МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження керованості англійсько-українського машинного перекладу на основі спеціалізованих корпусів. Моделі

Виконав:
студент 2 курсу групи ІПЗм-21-2
Максименко Д. В.
(прізвище, ініціали)

Спеціальність	121 – Інженерія програмного забезпечення
Тип програми	Освітньо-наукова
Керівник	доцент каф. ПІ Турута О. П.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

2023 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Програмної Інженерії _____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _____121 – Інженерія програмного_забезпечення_____

(код і повна назва)

Тип програми _____освітньо-наукова_____

Освітня програма _____Інженерія програмного забезпечення_____

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«___»_____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____Максименку Даніїлу Вікторовичу_____

(прізвище, ім'я, по батькові)

1. Тема роботи ____«Дослідження керованості англійсько-українського_____ машинного перекладу на основі спеціалізованих корпусів. Моделі»_____ затверджена указом університету від «29» _березня_ 20_23_р. № _302Ст__

2. Термін подання студентом роботи до екзамінаційної комісії «17» травня 2023р.

3. Вихідні дані до роботи машинний переклад, трансформери, кодувальник, декодувальники, Python, Pytorch, MarianMT, BERT, пояснювальна записка___

4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз предметної області, постановка задачі, дослідження технологій керованого машинного перекладу, порівняння запропонованої моделі з існуючими_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	05.04.2023	
2	Огляд існуючих рішень	10.04.2023	
3	Аналіз результатів експериментів	20.04.2023	
4	Підготовка пояснювальної записки	25.04.2023	
5	Спецчастина	27.04.2023	
6	Підготовка презентації та доповіді	03.05.2023	
7	Попередній захист	05.05.2023	
7	Нормконтроль, рецензування	08.05.2023	
8	Занесення диплома в електронний архів	15.05.2023	
9	Допуск до захисту у зав. кафедри	17.05.2023	

Дата видачі завдання 29 березня 20 23 р.

Студент _____
(підпис)

Керівник роботи _____ доцент каф. ПІ Турута О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить 155 стор., 45 джерела, 42 рис., 5 додатків, 8 таблиць.

КЕРОВАНІСТЬ, КОНТЕКСТ, КОРПУС, МАШИННИЙ ПЕРЕКЛАД, МЕТРИКИ, СЕМАНТИЧНИЙ ПОШУК, ТОКЕНІЗАЦІЯ, ТРАНСФОРМЕРИ, BERT, BERT SCORE, BLEU, MARIAN, METEOR.

Об’єктом дослідження є машинний переклад із передачею зовнішнього контексту для керування стилем результатів.

Метою роботи є підвищення керованості моделей машинного навчання на спеціалізованих корпусах за рахунок передачі векторів контексту, що вказують на стилістичні та тематичні характеристики.

Методи розробки базуються на таких технологіях як Python 3, Pytorch, huggingface transformers, мовні моделі BERT і MarianMT.

У результаті роботи пропонується модифікація архітектури кодувальників і декодувальників, яка додатково приймає вектори контексту при перекладі, що дозволяють підвищити керованість моделі та виконувати трансфер стилю на спеціалізованих корпусах. Модифікація була порівняна з існуючими рішеннями.

BERT, BERT SCORE, BLEU, CONTEXT, CONTROLLABILITY, CORPUS, MACHINE TRANSLATION, MARIAN, METEOR, METRICS, SEMANTIC SEARCH, TOKENIZATION, TRANSFORMERS.

The research objective is to develop a solution for machine translation with external context to control result style.

Aim of this paper is to increase controllability of machine learning models in specialized corpuses by passing context vectors, which should point towards style and topic features of the output.

Methods of development are based on technologies such as Python 3, Pytorch, huggingface transformers, language models MarianMT and BERT.

A modification of encoder-decoder architecture, which receives additional context vectors to increase controllability of a model and allows style transfer in specialized corpuses, was proposed and compared with existing solutions.

Умови публікації пояснювальної записки

Я,
_Максименко Даніїл Вікторович_____
(прізвище, ім'я, по батькові)

студент групи __ІПЗм-21-2__здобувач вищої освіти на другому (магістерському) рівні
кафедра_____програмної інженерії_____,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему
_____ Дослідження керованості англійсько-українського машинного перекладу на основі спеціалізованих корпусів. Моделі_____,
(назва роботи)

що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

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

ЗМІСТ

Вступ 8
1 Аналіз наукової галузі 11
1.1 Історія розвитку методів машинного перекладу……...11
1.2 Огляд сучасних рішень та літератури з приводу машинного перекладу.17
1.3 Огляд літератури з метрик оцінювання якості машинного перекладу…22
2 Формулювання методів і методики проведення дослідження.....29
2.1 Постановка задачі…….. 29
2.2 Обґрунтування напрямку дослідження 29
2.3 Обґрунтування методів дослідження.....31
2.4 Етапи проведення дослідження.....32
2.5 Методологія теоретичного дослідження 33
2.6 Методологія проведення експерименту 34
3 Огляд запропонованого рішення.....36
3.1 Архітектура рішення 36
3.2 Обґрунтування використання мереж для семантичного пошуку 40
3.3 Принцип використання запропонованого рішення.....43
4 План експерименту 47
4.1 Умови проведення 47
4.2 Використані дані 47
4.3 Критерії порівняння та принципи оцінки.....49
5 Аналіз результатів.....52
5.1 Контроль через навчання на малому корпусі 52
5.2 Контроль через додавання спеціального токену 56
5.3 Навчання запропонованої архітектури 58
5.4 Перевірка керованості перекладів.....65

5.5 Аналіз результатів запропонованої архітектури 70
6 Побудова програмної системи за результатами дослідження.....81
Висновки 84
Перелік джерел посилання.....86
Додаток А Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії 93
Додаток Б Звіт результатів перевірки кваліфікаційної роботи на унікальність тексту.....95
Додаток В Наукові публікації.....96
Додаток Г Слайди презентації.....136
Додаток Д Лістинг фрагменту коду для тренування моделі 148
Додаток Е Експертний висновок результатів перевірки кваліфікаційної роботи.....155

ВСТУП

Обсяги виробництва різноманітного медіа контенту зростають по всьому світу. На YouTube в середньому завантажують 700 тисяч годин контенту щодня, постійно публікуються нові статті, книги, ессе, оповіді чи просто дописи у соціальних мережах на актуальні теми. Виходить чимало фільмів від абсолютно різних країн, які за якістю не відстають від звичного голівудського кінематографу. Публікуються сотні наукових статей чи патентів. Вносяться правки у закони. У розумінні всього цього нас може суттєво обмежувати відсутність розуміння мови оригіналу.

Наразі моделі машинного перекладу дають вражаючі результати для більшості текстів [1]. Хоч вони і стали сприймати контекст набагато краще ніж раніше, такі моделі часто втрачають частину сенсу чи стилю при роботі з певними вузьконаправленими текстами. Той самий перекладач від Google не має можливості додати контексту до перекладу, а тому може давати доволі прямолінійні варіанти, коли людина намагалася б підібрати більш відповідне слово, використовуючи словники та шукаючи аналогії. Тобто автоматизовані рішення здатні зберігати сенс оригіналу, але часто програють і підборі відповідного стилю, структури фраз.

Тож, наразі основна проблема машинних перекладів – це відсутність можливості контролю контексту та стилю перекладу (умовно чи має це бути офіційно-діловий переклад, який настрій закладався в оригінал, якому домену має відповідати певне спірне слово) [2].

Актуальність дослідження методів контрольованого машинного перекладу полягає в потребі до підвищення розуміння контексту сучасними моделями глибокого навчання при спробах перекладу, а також відсутністю

великих структурованих і вручну перевірених наборів даних для української мови.

Робота пов’язана з науковою діяльністю кафедри програмної інженерії дослідженням методів глибинного навчання для української мови, що до цього також активно викладалося та перевірялося на експериментах у попередніх наукових роботах співробітників та студентів кафедри [3].

Мета дослідження полягає у підвищенні керованості моделей машинного навчання на спеціалізованих корпусах за рахунок передачі векторів контексту, що вказують на стилістичні та тематичні характеристики. Окрім основної мети дослідження має на меті вирішити набір супутніх задач:

- розробити модифікацію архітектури кодувальників і декодувальників, яка дозволяла б додавати між цими шарами додатковий контекст отриманий із зовнішньої моделі;
- порівняти запропоновану архітектуру із наявними рішеннями для керування перекладом на корпусах текстів різних стилів і тем за метриками.

Об’єктом дослідження є машинний переклад із передачею зовнішнього контексту для керування стилем результатів. Предметом дослідження є керованість англійсько-українського машинного перекладу на основі спеціальних корпусів.

У досліджені використовувалися кількісні методи для порівняння отриманого методу машинного перекладу із вже існуючими рішеннями за значеннями метрик.

Удосконалено роботу моделей для машинного перекладу, створених за архітектурою кодувальник-декодувальник, за рахунок передачі векторів контексту, що можуть формуватися зовнішньою моделлю та використовуватися для впливу на результати обробки вхідних текстів. Саме

використання даних із зовнішньої моделі відрізняє запропоноване рішення від уже наявних досліджень. Модифікація цих векторів дозволить задавати домен перекладу, стиль, бажану структуру та лексику.

Запропоноване рішення може використовуватися як для реалізації рішень з поглибленим контролем машинного перекладу, так і для пришвидшення тренування моделей для перекладу. Це досягається за рахунок контекстних векторів, що і дозволяють впливати на результати, і краще розмічати вхідні дані для швидшого досягнення прийнятного стану моделі-перекладача. Також дослідження надає рекомендації з використання та навчання вже наявних моделей перекладу при спробах адаптації під українську мову для певного домену.

У 2021 і 2022 роках було подано 2 статті, що досліджували якість роботи наявних мовних моделей на задачі класифікації для української мови. Публікації були подані на ICTERI 2021 і проіндексовані в SCOPUS. Перші результати дослідження були подані на CSIT 2022 ще у вересні 2022 року і показували порівняння вже наявних моделей машинного перекладу в контексті керованості перекладів на малих спеціалізованих корпусах. Статтю було представлено на конференції в листопаді 2022 року і проіндексовано в SCOPUS. У грудні 2022 року було подано статтю на IST 2022 з результатами обробки багатомовного, мультимодального набору даних Multi30k, що використовувався для подальшого навчання і тестування запропонованої архітектури. У лютому 2023 року було подано статтю на конференцію UNLP 2023, яка продовжувала та поглиблювала публікацію для IST 2022 за рахунок ідей з оцінювання керованості машинного перекладу і опису вже повністю перекладеного набору Multi30k.

1 АНАЛІЗ НАУКОВОЇ ГАЛУЗІ

1.1 Історія розвитку методів машинного перекладу

Автоматичному перекладу наразі вже більше 70 років і він пройшов довгий шлях від роботи за певним набором правил до інтелектуальних рішень, що можуть знають розмовну версію більшості мов не гірше за середнього носія. Перший крок до цього було зроблено в 1949 році Вореном Вівером. Він надав теоретичну базу для можливого автоматизованого рішення на основі теорії інформації та досягнень у декодуванні повідомлень, зроблених під час Другої Світової Війни. Кожне слово кодувалося частиною мови, числом, родом та іншими морфологічними характеристиками, а спеціальна клавіатура дозволяла передавати ці особливості для уточнення потрібного відповідника з іншої мови. Уже в 1954 році було розпочато експерименти на комп'ютері IBM 701, який зміг перекласти 60 російських речень англійською мовою [4]. Проте ці речення були лише найкращими зразками з усіх експериментів. У більш загальному використанні запропонований метод справлявся не сильно краще ніж умовний розмовник чи словник, але такі результати все одно довели можливість подальших досліджень у цій сфері та покращення результатів. До вирішення задачі доєдналися дослідники з Канади, Японії, Франції, Німеччини та багатьох інших країн.

Одним з важливих приводів до розвитку машинних перекладів також став початок Холодної війни. Америка активно продовжувала експерименти з покращення якості машинного перекладу, але у 1966 році такі дослідження були визнані не ефективними та було рекомендовано сфокусуватися на розвитку словників та прямих перекладів. Це на 10 років вивело США з гонки технологій машинного перекладу, хоча напрацювання з тих часів заклали основу для сучасної обробки природньої мови.

З 1970 року основним напрямком машинного перекладу став підхід заснований на правилах [5]. Його створювали на основі лінгвістичних правил та методів роботи звичайних перекладачів-людей. Така система зазвичай мала певний словник для пари мов та певний набір правил, що дозволяв інтерпретувати час, рід чи відмінок. Також поверх такої системи додавали модуль виправлення помилок, списки назв чи імен, які мають перекладатися за чітко заданим способом. Така система потребувала підтримки як програмних інженерів, так і лінгвістів, але контрольованість та здатність до оновлення таких систем дуже низька. Вони зазвичай давали сирий переклад, який все одно потребував редагування людиною. Класичний приклад сучасного перекладача заснованого на правилах – це PROMPT перекладач, який часто дає прямі переклади, може плутати відмінки і в принципі погано справляється з повноцінним перекладом чогось більш складного ніж якісь буденні прості фрази з кількох слів. Окремим варіантом такого методу виступає прямий переклад, що має заготовлені переклади для певних словосполучень, але такий метод ще менш контрольований та добре працює лише для малого набору опрацьованих випадків. Редагування такого алгоритму буде займати багато часу і вимагатиме фактично повного перебору можливих комбінацій для перевірки правильності роботи, але в середньому якість цих перекладів усе одно є дуже низькою.

Окремо існував підхід з використання певної метамови, що мала б позначки під певні дії чи об’єкти, які відповідали б слову в реальній мові [6]. Такі переклади часто не здатні формувати літературні переклади чи близькі до людського мовлення, але їх перевага була в швидкому перекладі з однієї мови на будь-яку кількість інших мов із передачею основних сенсів. Підхід з використанням так званої «інтерлінгва» (метамови) був також дуже дорогим для розробки і давав чимало синтаксичних і граматичних помилок. Розробка

такої універсальної мови займала цілі життя деяких вчених, але так і не дала достатньо прийнятного результату. Проте далі ці їдеї та напрацювання будуть використанні в більш просунутих, сучасних методах.

Усі ці методи фактично все одно підпадають під визначення перекладів заснованих на правилах. Кожний з них важкий для розробки і підтримки, добре охоплює лише певну область знань та погано узагальнюється. Проте ці підходи все ще популярні для певних вузько спрямованих текстів, бо при правильному налаштуванні системи слова не будуть плутатися і будуть відповідати чітко заданій предметній області (наприклад переклад сторінки з описом прогнозу погоди). Підхід з правилами все одно стикається зі складнощами при обробці суфіксів чи префіксів, може давати переклад, що звучить не натурально. Також йому дуже складно адаптуватися під омоніми (слова з однаковим написанням, але різними значеннями). Цей підхід прожив майже всю епоху Холодної війни і так і не довів свою перспективність, а тому залишився лише в окремих сферах, як дешевше за сучасні та відносно якісне рішення (наприклад переклади сторінок товарів на Aliexpress досі робляться саме за описаними вище алгоритмами).

Новий головний підхід прийшов з Японії, адже через структуру японської мови підхід за правилами було занадто важко адаптувати. Мови з використанням ієрогліфів не діляться на слова пробілами. Кожний символ фактично і є певним словом і може нести десятки різних значень або взагалі його значення визначиться тільки контекстом загального діалогу чи інтонацією при вимові. Новий підхід можна назвати перекладом по прикладам. У ньому ми маємо передавати алгоритму як можна більше пар речень із схожими сценаріями, але певними відмінностями, щоб програма навчилася будувати фрази по отриманим прикладам, розуміючи значення конкретних слів для загальної конструкції. Метод був запропонований Макото Нагао з

університету Кіото у 1980 році [7]. Він не став революцією одразу, але в теорії вже мав вирішити головну проблему правил – побудову гігантського набору умов, які також треба далі підтримувати. Фактично ця ідея стала основою для машинного перекладу на десятиліття вперед і навіть зараз залишається актуальною, хоч і певною мірою видозмінена.

Першим розвитком цієї ідеї став статистичний метод перекладу [8]. Йому передається певний корпус (набір) текстів на двох мовах. Тексти розбиваються на слова. Далі рахується скільки разів слово мало конкретний переклад. Саме цей варіант і буде використовувати система. Таким чином системі не задають жодний набір правил і не формують словник, а дають лише готові переклади. У теорії алгоритм має перекладати наближено до узагальненого людського перекладу. Запропоновано його було в 1990му році IBM Research Center. Відпадала потреба в лінгвістах, а робота метода в принципі була сильно більш ефективною та точною за переклад по правилам. З початкового опису залишається незрозуміло як саме алгоритм вирішує, що 2 слова з різних мов взагалі пов’язані. Коли він проходить по реченням і бачить з одного боку слово «man», а з іншого боку «чоловік» або «людина» і це постійно повторюється, то кореляція оригіналу та цих перекладів буде поступово зростати. На початку кожне слово буде однаково корелювати з будь-яким словом на іншій мові.

IBM назвали алгоритм за новим підходом «першою моделлю» [9]. Її проблема була у відсутності розуміння правильного порядку слів у реченні для мови перекладу, тож це вирішувала друга модель за рахунок пам’яті про звичну позицію певного слова. Третя модель додала до задачі статистичного перекладу задачу заповнення пропусків у текстах [10]. Вона оцінювала де може не вистачати слова і шукала за використанням найбільш оптимальний варіант для підстановки з урахуванням частини мови, яка б мала бути на місці

пропуску для підтримки граматичної цілісності. Четверта версія моделі запам’ятовувала відносний порядок слів, щоб розуміти як слова виставляються у реченні разом (умовно де саме має стояти прийменник при порівнянні з якимось іменником). П’ята модель скоріше просто виправляла явні помилки попередніх, але не зробила якогось явного прориву, що можна було б порівняти із попередниками. Така система все одно не справлялася з родами, омонімами, відмінками та в принципі дуже погано працювала для морфологічно багатих мов.

Розвитком статистичних перекладів стали статистичні фразові переклади [11]. Вони будували статистику використання перекладу не по окремим словам, а по нграмам. Нграма – поєднання якоїсь кількості слів, зазвичай від одного до трьох. Такий підхід порівнює не просто конкретні слова, а саме частини речень, що дозволяє йому певною мірою розуміти контекст через статистики використання певних перекладів для заданих комбінацій слів. Переклад став більш різноманітним і нарешті почав краще працювати з омонімами та навіть вчився правильно підбирати форму слова для гнучких мов. Цей метод став стандартом на 10 років, починаючи з 2006 року, для більшості онлайн перекладачів. Однією з його проблем була непередбачуваність. Переклад за правилами міг давати погані результати, але їх можна було прослідувати і передбачити. Статистичний переклад за фразами міг часом давати аномальні результати, які були викликані нестачею даних чи помилками в навчальному наборі.

Також цікавою була ідея поєднати переклад за правилами із статистичним методом, а саме синтаксично-статистичний переклад [12]. За ним речення аналізувалося для виявлення частин мов окремих слів і далі статистичним методом підбиралося слово, що відповідало як морфологічним показникам, так і статистично мало бути найбільш оптимальним перекладом

слова. Метод розроблявся як потенційне майбутнє машинних перекладів, але так і не перейшов далі стадії експериментів через якість синтаксичного аналізу автоматизованими алгоритмами. Як тільки задача побудови синтаксичного дерева стає більш-менш важчою за щось по типу виявлення підмету та присудку, то дерево синтаксичного аналізу може давати абсолютно неправильні результати.

Отже, у 2016 році галузь машинних перекладів нарешті прийшла до використання нейронних мереж автоматичної генерації текстів іншою мовою із збереженням сенсу, контексту та стилістики оригіналу.

У 2014 році Google опублікував дослідження використання рекурентних автокодувальників для статистичного машинного перекладу [13]. Уже в 2016 році було опубліковано продовження, яке пропонувало підхід близький до перенесення стилю зображень [14]. Отже, у цьому випадку текст буде вхідним зображенням, а мова – додатковою картинкою, з якої треба скопіювати стиль. Архітектура знову приймала вигляд автокодувальника, де кодувальник має створити матрицю, що виражала б сутність тексту, а декодувальник, знаючи лише цільову мову, має перетворити цю сутність на аналогічний текст іншою мовою. Тобто саме тут повернулася ідея проміжної мови або уявлення.

Саме ці 2 дослідження та їх імплементація в продукти Google з 2016 року дали поштовх новим роботам у сфері машинного перекладу тепер уже з використанням нейронних мереж, адже за достатнього розміру мережі, кількості даних та кількості ресурсів для розрахунків такий алгоритм здатний перебирати можливі зв’язки та показники доки не виявить найбільш дієві. При цьому від інеженерів не потребується побудови складних парсерів, правил і погано контрольованих статистичних моделей.

1.2 Огляд сучасних рішень та літератури з приводу машинного перекладу

Попередній розділ закінчився на ідеї використання рекурентних автокодувальників для задачі машинного перекладу. З того моменту галузь обробки природньої мови відійшла від рекурентних мереж у бік використання Attention механізму і трансформерів. Початок цьому дала стаття 2017 року під назвою «Attention is all you need» авторства Ашіша Васвані та ще 7 співавторів. Вони запропонували архітектуру, яка мала б відтворити якість рекурентних мереж, але витрачати менше часу та ресурсів на навчання, а також бути більш доступною для трансферного навчання та подальшого дотренування при наявності певного якісного стану мережі. Загалом архітектура також має N однотипних шарів у кодувальнику, які складаються з Attention механізму з кількома головами для паралелізму розрахунків і звичайної мережі прямої подачі [15]. Декодувальник будується аналогічним чином і на виході має повнозв'язний шар із певною функцією активації.

Фактично Attention має прийняти певний запит на пошук у словнику з ключами та значеннями. Результат рахується як зважена сума, де вага рахується як функція сумісності від значень вектору запиту та вектору ключів. Тоді було запропоновано варіант Attention механізму, який називається масштабованим Attention механізмом скалярним добутку (scaled dot-product Attention). Цей варіант рахує скалярний добуток запитів і ключів та ділить на корінь квадратний розмірності вектору ключів, огортає результат у softmax функцію та множить на матрицю значень, що показано у формулі 1.1.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \tag{1.1}$$

де Q – вектор запитів;

K – вектор ключів;

V – матриця значень;

d_K – розмірність вектору ключів.

Ця ідея повністю змінила підход до роботи з текстовими даними у глибокому навчанні, адже вона вирішила проблему із занадто глибокими рекурентними мережами, які переставали навчатися через затухання градієнту, потребували неймовірної кількості ресурсів та потужностей. Якщо спростити, то Attention механізм дав можливість розширити мережу замість поглиблення чим сильно спростив виконання таких важких задач як переклад. Перший трансформер, запропонований у цій статті, перетнув позначку у 40 за BLEU метрикою, про яку буде написано пізніше, за 3 дні тренування. Тобто мережі знадобилося лише 3 дні, щоб почати перекладати з англійської на німецьку не гірше за середньостатистичну людину.

Трансформери почали розходитися по іншим задачам обробки природньої мови. Найвідомішими є GPT для генерації текстів [16], BERT [17], який став універсальною основою для виконання більшості задач над текстами, XLM-R [18], яка вдосконалила результати BERT. Наразі Google Translate використовує саме трансформер архітектуру в поєднанні із статистичним перекладом. Відгуки, які даються після перекладу використовуються для подальшого навчання мережі методом навчання з підкріпленням. Варіант з найбільшою кількістю голосів має сприйматися мережею як найбільш вдалий. При цьому ці відгуки дозволяють розширяти набори даних для мов з малою кількістю ресурсів і в принципі збирати нові різноманітні тексти.

Окремо варто відмітити мережу Google під назвою T5 [19]. Вона створювалася для генерації тексту і мала конкурувати з GPT, але з часом виявилося що її багатомовна версія здатна добре вирішувати задачу перекладу,

якщо їй дати оригінал і явно прописати задачу (наприклад «переклади з української на польську»). У такому разі при подальшому тренуванні ця мережа також здатна перетнути межу людиноподібних перекладів доволі швидко. Наразі саме вона є основою Google Translate, проте не для всіх мов. Підхід з використанням багатомовної мережі для генерації тексту для перекладів дозволив також зробити одну велику модель для кількох мов одразу, що було важко зробити на рекурентній архітектурі через її розміри та складність.

Паралельно з Google інші дослідники нейронних мереж і штучного інтелекту також будували нові архітектури під машинні переклади з використанням Attention. Наприклад, Microsoft у 2018 презентували бібліотеку для C++ Marian за авторством Марчіна Юнкцис-Доумунта та ще 11 співавторів [20]. Ця бібліотека дозволяла тренувати одразу кілька різних архітектур від рекурентних автокодувальників до трансформерів з використанням C++ для пришвидшення важких розрахунків. Розглянемо саме трансформер версію Marian. Вона складається з кодувальника на 6 шарів і декодувальника також на 6 шарів. Може приймати чи генерувати текст до 512 токенів. Також мережа здатна працювати як з парою мов, так і з сімейством мов одразу (має передаватися спеціальний токен, що вказує на цільову мову перекладу). Кодувальник є двустороннім. Це означає, що він може враховувати не тільки попередні значення, але й наступні. Декодувальник є авторегресійним, тобто кожний наступний результат залежить від попердніх. Ця архітектура може подаватися через інтерфейс іншої нейронної мережі – BART і за рахунок цього її часто тренують на C++ і потім переносять на BART для використання у скриптових мовах [21]. На рисунку 1.1 можна побачити спрощений вигляд архітектури Marian.

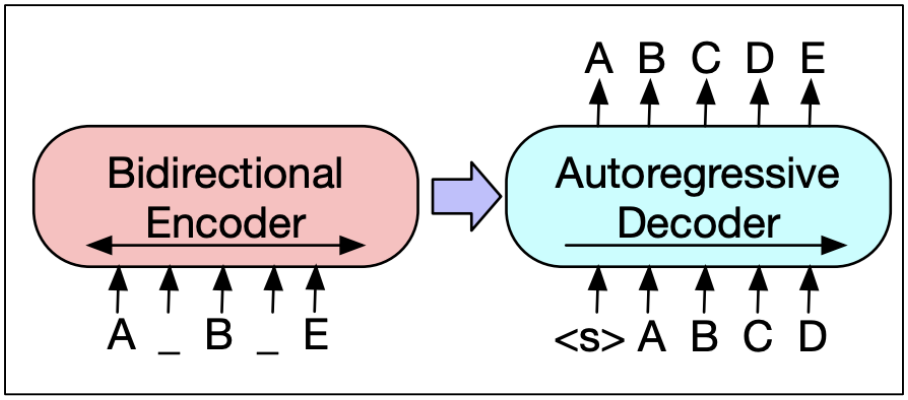

Рисунок 1.1 – Спрощений вигляд архітектури Marian

Також на якість машинного перекладу сильно вплинув новий підхід до токенізації тексту, який прийшов разом із трансформерами. Зазвичай токенізація відбувалася за словами, а тому щоб вивчити всі відмінки певного слова в українській мові мережі було потрібно створювати окремий токен під кожний варіант слова і створювати окремі вектори уявлень під них. Трансформери токенізують текст не по окремим словам, а по довільним шматкам тексту, якщо не знають повне слово. Наприклад, мережа може не знати слово «Харків», але вона розіб’є його на окремі токени «Ха», «##р», «##кі», «##в», а за рахунок векторів уявлень і порядку цих токенів зможе зрозуміти контекст і значення цих токенів разом. Це спростило роботу як з англійською мовою, адже тепер словник мережі скоротився лише до кількох десятків тисяч токенів (зазвичай беруть близько 32 тисяч) і не потрібно розширяти словник під кожне окреме нове слово, а далі перенавчати мережу. Також це серйозно покращило роботу для мов без пробілів по типу японської чи китайської та для морфологічно багатих мов, по типу української, де одне слово може змінюватися безліч разів за рахунок різних суфікісів, закінчень, префіксів, родів, числа і так далі. Такий підхід було запропоновано разом з BERT у 2018 році.

Усі ці архітектури створювалися для прямолінійного перекладу тексту між мовами. Тобто вони не дають можливостей впливати на свою роботу та налаштовувати результати. Єдиним способом контролю для таких моделей є додаткове тренування моделі на певному спеціалізованому корпусі з метою виконати трансфер стилю з цього корпусу на поведінку моделі загалом. Такий спосіб може дати високу якість для конкретного домену, але вимагає створення окремих моделей донавчених під конкретний стиль, або підтримки однієї великої архітектури, яка б навчалася на всіх цих текстах одразу. Проте останній варіант може дати гіршу якість на окремому домені, якщо інші стилі суттєво переважають його за кількість текстів. Таким чином, цей підхід не є раціональним через час на тренування і потребу збору високоякісних наборів даних під кожний новий домен.

Наступний спосіб контролю перекладу таких моделей – додавання спеціального токену маркеру, який би вказував на бажані характеристики кінцевого перекладу. Здавалося б один токен не має давати суттєвого впливу, але трансформери використовують двонаправлений кодувальник, а тому цей токен буде враховуватися для створення уявлень про всі інші вхідні токени. Авторегресійний декодувальник трансформерів-генераторів мав би дозволити скористатися цими модифікованими уявленнями для генерації кожного наступного токену з використанням маркеру стилю. Проте цей підхід дає обмежений вплив (під кожну характеристику треба буде додавати окремий спеціальний токен і ці токени займатимуть місця справжніх токенів вхідного тексту, що зменшить максимальний розмір вхідних даних моделі) і також потребує додаткового тренування, адже модель буде намагатися перекладати цей маркер як звичайний текст, якщо не вивчить спосіб його використання на прикладах.

Наступні способи модифікують зв’язок кодувальника і декодувальника. Одне рішення використовує результати моделювання теми і додає їх як ще один вектор до матриці токенів [22]. Проблема цього підходу полягає у додаванні лише одного нового рядку до матриці результатів кодувальника, що дає обмежений вплив на декодування матриці в токени, а також у використанні латентного розподілу Дирихле для моделювання теми, що є застарілим методом для цієї задачі. Інший підхід пропонує додавати на кожний рядок матриці уявлень про токени вектор з якоюсь кількістю характеристик (довжина, офіційність, ввічливість тощо) [23]. Цей підхід дає більший вплив і контроль над перекладом, але також потребує нових тренувань для кожної нової характеристики чи при зміні поведінки для вже наявної характеристики.

Отже, якщо як такий машинний переклад стабільно розвивався і досяг значних успіхів з приходом трансформерів, то контрольований переклад не може позбутися проблем із постійним тренуванням для внесення найменших модифікацій і наявні методи не надають значної гнучкості в керованості.

1.3 Огляд літератури з метрик оцінювання якості машинного перекладу

Паралельно з розвитком автоматичних перекладів наростала проблема автоматичної перевірки якості перекладу. Метрики для перевірки якості машинного перекладу можна поділити на 2 типи: токен-метрики та метрики засновані на уявленнях (embedding metrics). Кожний тип метрик може вирішувати свої задачі та по своєму описувати отриманий результат, дозволяючи виявити проблеми чи переваги отриманого алгоритму.

Токен-метрики є класичним підходом до оцінки машинного перекладу. Їх сутність полягає в порівнянні еталону та згенерованого тексту за відповідністю токенів (це може бути як слово, так і нграма чи просто набір

символів від однієї штуки). Цей підхід і досі використовується як головний спосіб заміряти та порівнювати якість перекладу різних моделей та архітектур.

Основна токен-метрика – BLEU [24]. Вона була запропонована у 2002 році Салімом Роукосом, Кішоре Папінені і Ві-Джін Жу. Їх ідея полягала в автоматизації оцінки перекладу, щоб пришвидшити цей процес та спростити роботу команд з машинного перекладу. Основна спрощена задумка – рахувати кількість співпадаючих нграм у кандидата та зразка. Чим більше нграм різного розміру знаходять відповідники серед зразків – тим краща якість перекладу. Зазвичай беруться юніграми, біграми та триграми і рахується середнє геометричне трьох варіантів. Тоді, якщо хоча б по одному набору нграм отримаємо нульову схожість, то і загальна оцінка схожості буде також 0 (формула 1.2).

$$BLEU(ref, cand, N) = \prod_{n=1}^{N} p_n^{w_n} \tag{1.2}$$

де ref – еталонний переклад;

$cand$ – переклад-кандидат;

N – кількість типів нграм, які мають перевірятися;

p_n – кількість співпадінь для нграми, яка враховує одразу n токенів з еталонів (ref) і кандидатів ($cand$);

w_n – зазвичай рівномірні ваги для кожного типу нграм (умовно перевіряємо всі нграми до 4 і тоді вага буде дорівнювати $\frac{1}{4}$), проте можуть налаштовуватися окремо під кожну нграму.

Проблема BLEU полягає у поганій роботі з гнучкими мовами, де нграма може змінитися просто після появи певного суфіксу чи префіксу, або коли при

перефразуванні у слова зміниться відмінок і закінчення. Для BLEU це все буде помилкою, бо через кілька нових символів нграма вже перестає збігатися, а тому оцінка стане несправедливо заниженою або взагалі нульовою.

Це частково вирішує інша токен-метрика METEOR, запропонована у 2005 році Сатанджівом Банерджі та Алоном Лаві. Основна ідея метрики збігається з BLEU, але METEOR додає до механізму порівняння словник синонімів і стемінг (зрізання закінчень). Таким чином морфологічно багаті мови вже мають кращу оцінку за METEOR, якщо переклад не ідеально відповідає оригіналу, але все одно передає необхідний сенс, але з іншим набором фраз.

METEOR рахується тільки по юніграмам (нграмам з одного токену) та використовує гармонійне середнє по точності та відклику (precision і recall) [25]. Формула 1.3 описує розрахунок METEOR.

$$METEOR = \frac{10PR}{R + 9P} = \frac{10 * \frac{m}{w_r} * \frac{m}{w_c}}{\frac{m}{w_c} + 9\frac{m}{w_r}} \tag{1.3}$$

де P – точність;

R – відклик;

m – кількість юніграм, що є і в кандидатах, і в еталонах;

w_c – кількість юніграм у кандидатах;

w_r – кількість юніграм у еталонах.

Так чи інакше токен-метрики обмежені лише перевіркою за схожістю за символами чи словами і потребують еталонних зразків для цільової мови. Навіть перевірка синонімів у METEOR не може вважатися повноцінною перевіркою значення, адже ядром цієї метрики все одно є розрахунок співпадінь нграм.

Метрики засновані на уявленнях є доволі новим явищем і були запропоновані у 2020 році Тіаний Жангом і Варшом Кішоре у статті «BERTScore: Evaluating Text Generation with BERT» [26]. Ідея полягає у використанні зовнішньої моделі, наприклад багатомовного BERT, для отримання векторів характеристик кожного токену і порівняння їх за косинусною відстанню. Для кожного токену обирається найбільш схожий відповідник, а також рахується IDF коефіцієнт, який показує важливість токену для загального корпусу. При малому розмірі корпусу IDF можуть прибирати, адже він може схибити уявлення про важливість певного токену.

Перевагою BERT Score саме для української мови є його незалежність від саме значення токенів та порівняння за уявленнями, бо тоді ми можемо вловити саме схожість значень, а не просто ідеальну відповідність одного набору нграм до іншого набору. Це важливо для мофрологічно багатих і гнучких мов, бо поява будь-якого суфіксу чи заміна на синонім може серйозно погіршити результат токен метрики. Так, METEOR компенсує це стемінгом і синонімами, але він все ще сильно залежить від якості словнику та розмітки закінчень. BERT Score залежить лише від якості зовнішньої моделі, а наразі їх більш ніж достатньо під різні мови.

BERT Score також рахує точність і відклик, де відклик – це максимальні схожості між токенами кандидату та еталону, поділені на модуль важливостей кожного токену еталону, а точність рахується так само, але замість кандидату береться еталон. Далі береться гармонійне середнє як F показник, що можна побачити у формулах 1.4 – 1.6.

$$R_{BERT} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\widehat{x_j} \in \hat{x}} x_i^T \widehat{x_j}\,, \tag{1.4}$$

$$P_{BERT} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^T \hat{x}_j \,, \tag{1.5}$$

$$F_{BERT} = 2\frac{PR}{P+R} \tag{1.6}$$

де x – вектор еталонних значень;

$\hat{x}$ – вектор значень кандидатів.

Головна проблема BERT Score та його подальших модифікацій для оцінювання задачі перекладу – метрика має працювати лише з текстами однієї мови, а тому для оцінки BERT Score ми все ще потребуємо певного еталонного перекладу і не можемо напряму порівняти оригінал з перекладом. Також BERT Score може давати посередні оцінки навіть коли тексти абсолютно не схожі (якщо наприклад декодувальник моделі було модифіковано і він втрачає зв’язок між розподілом результатів і вхідних значень). BERT Score може дати оцінку в районі 0.5-0.6 або через проблеми з нормалізацією по IDF, або через близькість окремих токенів, але загальну відмінність значень текстів.

Кожна з цих метрик вимагає наявності еталонного перекладу цільовою мовою, що далеко не завжди є в наявності в рамках задачі контрольованого машинного перекладу. Зазвичай буде лише один найбільш частий переклад, але в ідеалі задача контрольованого перекладу мала б замірятися за рахунок порівняння кожного нового перекладу із оригіналом мовою джерела. Тобто перевірка, що новий переклад не втратив сенсу оригінального тексту і не віддалився від нього занадто сильно.

Це може вирішувати порівняння вектору уявлень про текст загалом. Такий підхід є можливим за рахунок дослідження 2020 року від Нілса Реймерса у статті «Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation» [27]. Він запропонував спосіб модифікації

англомовних моделей для семантичного пошуку, щоб отримувати вектори уявлень про текст в одному просторі для одразу кількох мов. Їх ідея полягала у тренуванні моделі на мові з великою кількістю ресурсів та перенесенні отриманих знань на іншу менш насичену мову. Таким чином спочатку може тренуватися умовний BERT під англійську мову, а далі мультимовна модель, як до прикладу XLM-R, має повторювати вектори уявлень отримані BERT для перекладених на різні мови однакових текстів. Оцінка схожості двох текстів відбувається за косинусною схожістю векторів-уявлень (формула 1.7).

$$S_c(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{1.7}$$

де A і B – числові вектори, що порівнюються.

Подібний підхід реалізує одна з версій фреймворку COMET для оцінки перекладів [28]. але вона базується на простій XLM-R, що часом призводить до проблем при порівнянні фразеологізмів [29]. Проблеми трактування фразеологізмів чи певних виразів із переносним значенням можна спробувати усунути за рахунок мультимодального трансферу знань. Деякі вирази як «high five» англійською чи «дай п’ять» українською простіше зрозуміти маючи перед очима картинку, яка позначає відповідну дію. Для цього можна скористатися моделями для мультимодального семантичного пошуку, які тренуються не на задачі порівняння текстів. Вони пробують відтворити вектори-дескриптори зображень по їх описах і за рахунок цього отримують знання не тільки з текстів, але й з візуального домену чим покращують роботу з текстами, слова в яких не можна сприймати прямо.

Отже, жодна з розповсюджених, сталих метрик не дозволяє перевіряти переклад одночасно і на рівні токенів, і загалом за семантичним значенням без

наявності еталону для цільової мови. Останній згаданий метод ще не є достатньо перевіреним і також сильно залежить від якості тренування моделі-оцінювача. Використання лише текстових моделей для порівняння оригіналу з перекладом може призвести до значних проблем при порівнянні текстів з фразеологізмами чи переносними сенсами. Проте в рамках контрольованого перекладу класичні токен метрики чи метрики за уявленнями можна використовувати для порівняння на окремих доменах, щоб перевірити яка модель краще відрізняє певний конкретний набір текстів і їх характеристики. Метод порівняння оригіналу з перекладом через багатомовну модель для семантичного пошуку може допомогти при дослідженні ефективності методу керованого перекладу на окремих прикладах (щоб перевірити метод не тільки якісно за оцінкою людини, але й заміряти відповідність текстів кількісно за косинусною схожістю їх векторів-дескрипторів).

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Формулювання постановки задачі дослідження

Виходячи з аналізу предметної області та визначених проблем та недоліків наявних рішень можна визначити задачу дослідження наступним чином: визначити метод підвищення керованості моделей машинного перекладу на спеціалізованих корпусах за рахунок модифікації наявних архітектурних рішень.

У ході дослідження треба запропоновувати метод трансферу стилю з певного домену на переклад вхідного тексту. Запропонована архітектура має змінювати стиль, тон, структуру настрій, лексику кінцевого перекладу на основі характеристик обраного цільового домену.

Задача також потребує тренування існуючих рішень для керованого перекладу і їх подальшого порівняння із запропонованою архітектурою. Порівняння має виконуватися на корпусі текстів різних стилів, тем і структури.

Рішення щодо ефективності запропонованого рішення має бути прийнято на основі кількісного аналізу його результатів і результатів інших протестованих моделей.

2.2 Обґрунтування напрямку дослідження

Контрольованість моделей машинного перекладу є одним з найважливіших їх аспектів, адже наразі більшість комерційних рішень надають чорну скриньку, де ми часом можемо обирати з кількох варіантів або обрати переклад окремого слова. Останній варіант часто і досі реалізовується за принципом частотного перекладу, коли варіанти перелічуються за частотою

їх використання у тренувальному наборі алгоритму. Користувач не може прямо вказати моделі, який стиль перекладу має бути використано чи на який текст має бути схожий переклад за структурою чи формулюваннями.

Дотреновування моделей також ускладнюється для перекладів на мови з малою кількістю ресурсів, адже ймовірно знайти якийсь готовий датасет не вийде, а збір власного набору даних може бути непростою задачею, яка займе чимало часу та вимагатиме роботи одразу багатьох людей (для власне збору, очистки, редагування і структуризації). Додатковий токен дає занадто мало контролю та насправді слабо впливає на кінцевий переклад без додаткового тренування. Передача векторів власноруч визначених характеристик потребує чіткого розуміння цільового стилю та розуміння користувачем шкали цих оцінок, яку закладають розробники.

Зазвичай користувачу простіше дати приклад бажаного результату. Передача додаткового контексту могла б серйозно полегшити це, адже ми мали б одну модель, яка працюватиме одразу в кількох доменах та здатна порівнювати не тільки токени, а й тексти загалом за їх сенсом, темою, настроєм.

Характеристики для передачі контексту намагалися отримувати з використанням латентного розподілу Дирихле [30]. При цьому опис теми передається не до кодувальника, а до декодувальника, щоб кодувальник описав вхідні токени, а саме декодувальник змінював вигляд фінального тексту. Цей метод більш дієвий і дозволив швидко отримати BLEU більший за 30 на валідаційній вибірці для англо-німецьких перекладів. Проте латентний розподіл Дирихле є вже доволі старим статистичним методом моделювання теми і відстає за якістю від моделей глибокого навчання.

Отже, у цьому дослідженні ми пропонуємо використати архітектуру нейронного автоенкодера з Attention механізмом (трансформера) з передачею

вектора контексту, зміни якого можуть впливати на кінцевий стиль перекладу. Тобто надавати користувачу контроль над моделлю через опис текстів, що мають стиль і структуру подібний до необхідного в кінцевому перекладі.

2.3 Обґрунтування методів дослідження

Дослідження є кількісним, бо головними критеріями при його проведенні є токен-метрики і метрики за уявленнями. Такі критерії дозволять порівняти розроблені алгоритми з уже наявними дослідженнями за точністю відтворення еталонних перекладів і дослідити якість передачі сенсу оригінального тексту завдяки сучасним метрикам за уявленнями, навіть коли переклад не ідеально повторює зразок.

Метрики за уявленнями мають дозволити оцінити якісні критерії (відповідність сенсу, стилю оригіналу та перекладу) у автоматизованому режимі. Раніше такі речі можна було б оцінювати лише з використанням якісних методів дослідження із залученням лінгвістів і професійних перекладачів, щоб вони дали свою думку з приводу якості роботи алгоритму. Наразі за рахунок так званих state-of-the-art моделей (моделей, наближених до якості раніше доступної лише людині) ми можемо спробувати оцінити це автоматизовано.

Модель прийматиме якісні показники, а саме текст для перекладу. Він буде токенізуватися та кодуватися у набір чисел-ідентифікаторів, де кожне число визначає певне слово, знак пунктуації чи набір символів. За допомогою дедуктивного аналізу, виходячи з використання трансформер архітектури, можемо визначити, що текст не потребує жодної попередньої обробки чи очистки окрім власне токенізації та кодування при перекладі та декодування при розшифруванні результатів.

Класичні моделі приймають на вхід лише якісні показники. Запропонований у цій роботі алгоритм також має приймати вектор опису контексту, а тому нова модель зможе приймати якісні показники схожості певних текстів, їх семантичні характерситики у вигляді кількісних показників. Використовуючи методи індуктивного аналізу, можемо далі оцінити якість опису характеристик тексту. Для цього порівняємо окремі вектори показників, кластеризуємо їх і зробимо візуалізації.

2.4 Етапи проведення дослідження

Оскільки задача дослідження полягає у пошуку методів підвищення контрольованості алгоритмів машинного перекладу на спеціалізованих корпусах, то можемо виділити наступні етапи наукового дослідження:

- дослідити наявні архітектури для машинного перекладу з використанням нейронних мереж, адже наразі це є найбільш ефективний підхід;
- дослідити методи збору семантичних і стилістичних показників для передачі зовнішнього контексту до моделі-перекладача;
- побудувати архітектуру для машинного перекладу із можливістю передачі зовнішнього контексту;
- порівняти отриману архітектуру із наявними рішеннями за токен метриками;
- оцінити якість перекладу нової архітектури та наявних з використанням метрик за уявленнями, щоб визначити як вдало алгоритм передає семантичні та стилістичні характеристики оригіналу;

- оцінити ступінь контролю, який надає запропонована архітектура на прикладах із заміром семантичної схожості перекладів і оригінального тексту;
- розробити інтелектуальну програмну систему для машинного перекладу з використанням розробленого алгоритму.

2.5 Методологія теоретичного дослідження

Опишемо вхідні дані для навчання моделі та заміру якості моделей машинного навчання. Це допоможе конкретизувати методи аналізу та обґрунтувати їх використання. Отже, маємо наступні дані:

- OPUS корпуси для англійської та української мов на майже 2 мільйони пар текстів, які містять переклади TED лекцій, субтитри, статті з вікіпедії, літературні переклади [31];
- власноруч перекладений набор Multi30k, який містить 30 тисяч пар англійських і українських описів зображень [32];
- власноруч зібраний та структурований набір перекладів законів із сайту Верховної Ради (близько 2 тисяч речень) [33];
- власноруч зібраний та очищений набір перекладів рефератів до наукових робіт з сервісу Google Patent (майже 2000 пар для англійської та української мов).

Тож, наявні дані буде проаналізовано з використанням наступних методів дослідження:

- кластеризація та візуалізація текстів для виділення семантичних і стилістичних груп;

– індуктивний аналіз векторів контексту для кожного тексту з метою визначення їх ступеню впливу на кінцевий результат моделі;
– порівняльний аналіз моделей на основі тестового набору даних і додаткове порівняння по кожному стилістичному домену в тестовому наборі.

2.6 Методологія проведення експерименту

Метою експериментів є порівняння запропонованої архітектури контекстуального перекладу з наявними моделями машинного перекладу додатково натренованими на спеціалізованих корпусах версіях чи з токеном-вказівником потрібного стилю. Нова модель має дозволяти перекладати вузькопрофільні тексти в середньому не гірше за окремі спеціалізовані версії існуючих архітектур і надавати можливість контролю перекладу. Порівняння моделей має відбуватися за токен-метриками і метриками за уявленнями. Тоді функцію оптимальності архітектури можна визначити як у формулі 2.1.

$$E = f(tm, em) \tag{2.1}$$

де tm – значення токен-метрик;

em – значення метрик за уявленнями.

Будемо вважати експеримент із запропонованою архітектурою вдалим, якщо вона випередить наявні, протестовані моделі як на повному тестовому наборі даних, так і на окремих доменах у цьому наборі.

Після підготовки та тренування всіх обраних варіантів архітектури контрольованого машинного перекладу і порівняння їх результатів за

визначеним набором метрик на тестовому наборі варто також перевірити ступінь контролю стилю, структури та підбору лексики у запропонованій архітектурі. Для цього треба підібрати тексти, що можуть перекладатися кількома способами залежно від контексту. Далі за рахунок внесення змін у вектор контексту пробувати змінити характери перекладу та заміряти його відповідність тексту оригінальною мовою, що також можливо виконати за рахунок метрик за уявленнями.

3 ОГЛЯД ЗАПРОПОНОВАНОГО РІШЕННЯ

3.1 Архітектура рішення

При огляді літератури вже було згадано як більшість сучасних моделей машинного перекладу прийшли до вигляду кодувальників-декодувальників із різними реалізаціями цього підходу (рекурентні рішення, трансформери на основі Attention механізму). Тобто такі моделі приймають вектор токенів, які кодуються числами, а кожному токену присвоюється вектор-уявлення. На виході кодувальника маємо матрицю, яка містить характеристики кожного окремого вхідного токену. На основі цієї матриці декодувальник будує новий вектор, що розшифровується в текст. Також було згадано способи перетворення цих моделей на моделі керованого машинного перекладу (за рахунок токенів маркерів, тренування на малих корпусах, моделювання теми, додавання вектору фіксованого набору характеристик) [34]. Усі ці підходи впираються у потребу постійного ручного налаштування та тренування з нуля при найменшій зміні запланованої поведінки чи додаванні нових параметрів. Описані підходи не здатні виконувати саме трансфер стилю на кінцевий переклад.

Підхід запропонований у цій роботі відштовхується від попередньо згаданого методу та від архітектури SV2TTS (Speaker Verification to Text-To-Speech) [35], яка використовується для озвучування тексту із трансфером голосу певної людини без додаткового тренування мережі. Для цього їй потрібен лише один зразок цільового голосу з якого додаткова модель створює вектор-уявлення, що конкатенується на вектор кожної літери тексту, що має бути озвучений. Тобто у нас є набор основних моделей, що виконують головну задачу – у нашому випадку це модель машинного перекладу з архітектурою кодувальника-декодувальника, а також має бути зовнішня

модель, що збирає додатковий контекст, який і буде використовуватися для контролю результатів. Таким чином, ми можемо взяти вже наявну архітектуру машинного перекладу, ініціалізувати її шари з певного стану, який вже було натреновано на великому наборі даних, взяти зовнішню модель, що навчається виконувати будь-яке завдання, яке дозволить отримати семантичні та стилістичні характеристики тексту, а далі маємо поєднати уявлення з кодувальника-перекладача та зовнішньої моделі та декодувати їх у нові токени цільової мови.

У цьому дослідженні пропонується використати моделі для семантичного пошуку, адже вони вже завчасно натреновані під задачу підбору найбільш схожого тексту за необмеженою кількістю характеристик. Зазвичай вони тренуються як сіамські мережі, тобто мають 2 одинакових кодувальники, які мають створити вектори-уявлення двох текстів, а на виході дати оцінку схожості, яка буде близька до людської. Як міру оцінювання використовують попередньо згадану косинусну схожість. Ці моделі далі використовують для широкого спектру задач: кластеризація чи класифікація текстів або ж збір додаткових даних для більших задач (наприклад, уявлення про опис товару як додаткова характеристика часового ряду його продажів при прогнозуванні). Ці вектори можуть як додати більше контексту при перекладі за рахунок позиціонування перекладеного тексту в одному просторі з іншими схожими зразками, так і дозволити змінити переклад за рахунок зміщення оригінального вектору семантичних і стилістичних характеристик в кластер з іншими характеристиками.

Раніше вже згадувалося, що будь-яка модель кодувальник-декодувальник може бути використана як основа для подібного рішення, особливо якщо вона вже має добре навчені стани, щоб не починати навчання з чистого листа, адже досягти якісного результату для машинного

перекладу з використанням нейронних мереж без великого набору текстів і відповідних потужностей для навчання буде неможливо. У цьому дослідженні за основу буде взято попередньо згадану в огляді існуючих рішень MarianMT. Як модель семантичного пошуку можна використати дистильований сіамський англомовний BERT, що повертає вектор на 384 елементи з описом вхідного тексту. На рисунку 3.1 показано загальний вигляд запропонованої архітектури контрольованого машинного перекладу.

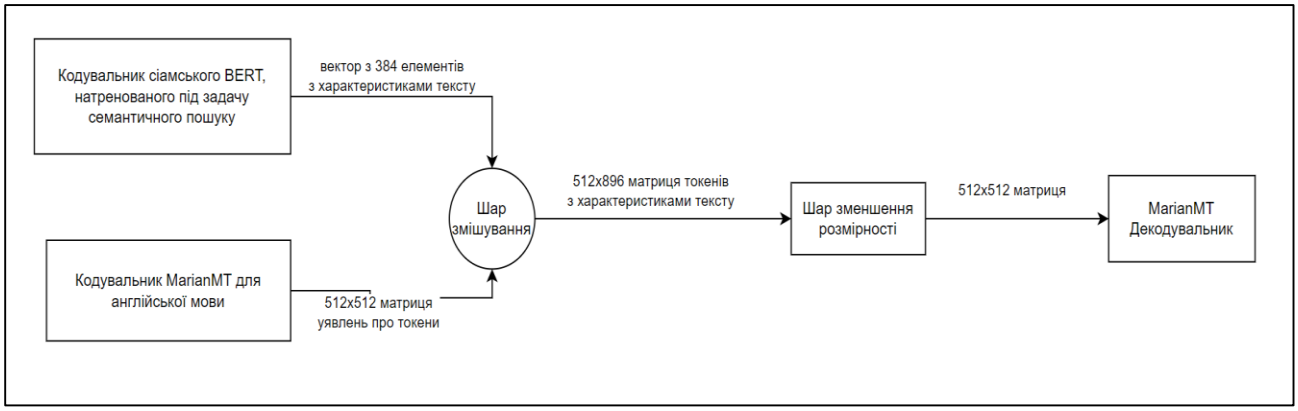

Рисунок 3.1 – Загальна архітектура запропонованого рішення для контрольованого машинного перекладу

Блок зменшення розмірності та відбору змінних складається з одного лінійного шару та SiLU активації [36], яка має також виступає як додаткова регуляризація для лінійного шару. Ця функція має працювати краще для глибоких моделей за класичну ReLU [37], яка рахується як $\max(0, x)$, де x – вхідне значення тензора. SiLU переходить від 0 до вищих значень більш плавно та пропускає частину негативних вагових коефіцієнтів і за рахунок цього градієнт повільніше затухає при обході глибоких мереж. SiLU досягає глобального мінімуму при $x \approx -1.28$ і глобальний мінімум дорівнює -0.28.

Назва функції розшифровується як Sigmoid Linear Unit або сигмоїдна лінійна одиниця (формула 3.1).

$$SiLU(x) = x * \sigma(x) = x * (1 + e^{-x})^{-1} \quad (3.1)$$

де x – вхідне число;

$\sigma(x)$ – функція сигмоїду;

e – експонента.

На рисунку 3.2 покажемо графіки SiLU і ReLU функцій, які явно доводять різницю двох активацій і показують більш плавний перехід до додатніх значень у SiLU.

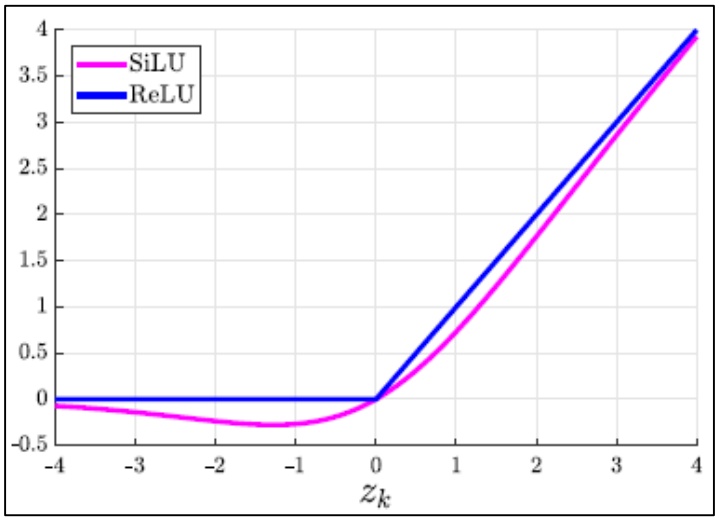

Рисунок 3.2 – Порівняння графіків ReLU і SiLU

Таким чином, при достатньому тренуванні оновленної моделі ми маємо отримати рішення, що зможе переносити характеристики певного набору текстів на переклад основного вхідного тексту (наприклад перекласти текст у

офіційно-діловому стилі чи навпаки зробити його структуру більш схожою на повсякденне мовлення, чи зробити переклад у стилі інструкції).

3.2 Обґрунтування використання мереж для семантичного пошуку

Для обґрунтування використання саме мереж для семантичного пошуку проаналізуємо проєкції на 2D площину уявлень отриманих з сіамського BERT. У ході аналізу скористаємося набором даних, що містить документацію до програмної бібліотеки, закони, описи фотографій і реферати наукових статей. Побудуємо уявлення для кожного з текстів у наборі. Вектори мають 384 елементи, а тому не можуть бути візуалізовані в їх звичайному, сирому вигляді. Для цього скористаємося іншою моделлю, а саме t-SNE (t-distributed stochastic neighbor embedding) [38]. Мета цієї моделі знайти місце певного вхідного вектора у просторі меншому за оригінальну розмірність вхідних даних. Алгоритм намагається створити нові вектори-відповідники меншої розмірності, Евклідова відстань між якими буде максимально відповідати оригінальним відстаням. PCA (Principal component analysis) працював би швидше за t-SNE [39], але цей алгоритм вловлює лише глобальну дисперсію значень у наборі даних і втрачає локальні дисперсії окремих груп при мінімізації розмірності. t-SNE має на меті зберігати дисперсію лише сусідніх значень і при цьому не є детермінованим алгоритмом. Тому результати для однакового набору можуть відрізнятися у кількох запусках, але все одно втримувати подібну структуру окремих груп.

У нашому випадку потрібно підібрати точку на 2D площині, яка б відповідала кожному тексту описаному 384-мірним вектором. На рисунку 3.3 бачимо фінальну візуалізацію уявлень побудованих по заданому набору даних.

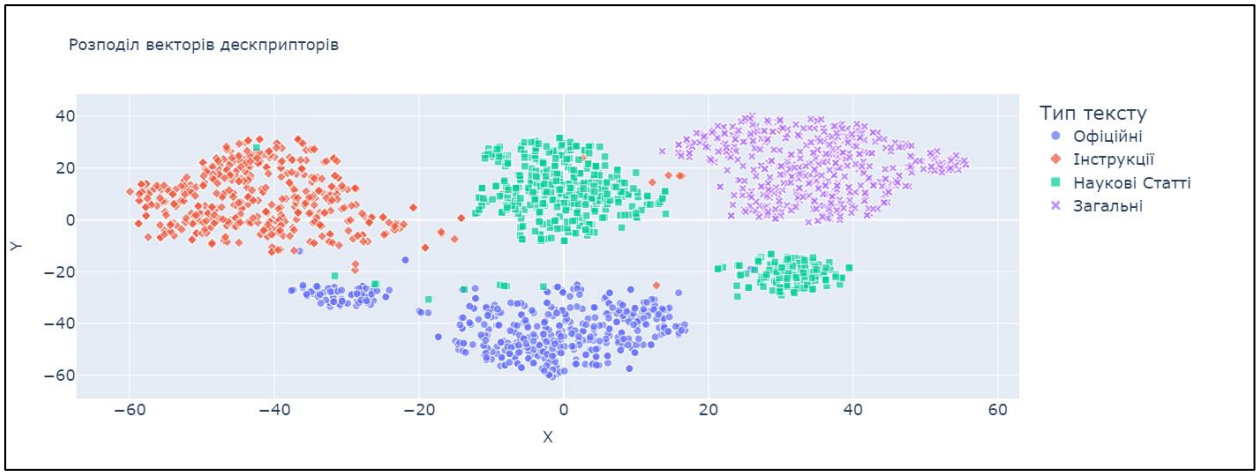

Рисунок 3.3 – Візуалізація уявлень про тексти на 2D площині

На цьому графіку чітко видні окремі групи за темою та навіть окремі менші кластери у кожній групі. Тобто вектори-дескриптори для семантичного способу справді добре відділяють тексти різних тем та стилів. Дослідимо ці кластери та окремо глянемо на випадки коли тексти різних стилей опинялися в одному просторі.

Зелені точки – це тексти з наукових статей. Вони поділилися на 2 окремі групи: тексти з природничих наук чи інженерії та тексти з правознавства чи економіки, які наблизилися до кластеру законів. Окремі тексти з посиланнями на певні закони навіть опиняються прямо серед юридичних текстів. Проте є і певні дивні рішення, як наприклад текст з хімії, що опинився посеред програмної документації. Це можна пояснити нестачею даних такого характеру для t-SNE, адже косинусна схожість текстів з кластеру документації та цього зразку близьку до 0 (тексти не є пов’язаними, але не протилежні одне одному).

Закони чітко розділилися за формулюванням. Частина, що пишеться як «стаття 2 визначає/робить щось», відділилася в окремий субкластер. Також окремо відділилися тексти, що регулюють виборче право.

Документація в більшості скупчилася в один великий кластер, проте тексти, що складаються з одного чи двох текстів і мають в собі просто якийсь термін чи назву потрапляють ближче до наукових статей, що виглядає логічно, адже там також можуть бути тексти, які пропонують певну назву, скорочення чи пояснюють терміни.

Останній кластер – описи фотографій, які прості і за структурою, і за лексикою. Вони віддалені від усіх вузькопрофільних текстів і це саме той результат, який потрібен для задачі контрольованості перекладу. При цьому навіть серед цих текстів можна побачити малі кластери, що відрізняються темами: діти, тварини, машини, чоловіки, жінки, групи людей. Жодних аномалій для цього кластеру помічено не було.

Таким чином, ці вектори справді достатньо добре дозволяють визначати теми та схожі характеристики текстів, що допоможе контролювати переклад та додати контексту моделі при пошуку відповідників цільовою мовою. Ми можемо розрізняти окремі семантичні та стилістичні групи та подавати моделі інформацію про схожість певного нового тексту на ті, що вона вже бачила раніше, що має також полегшити вивчення нових доменів і в принципі полегшити навчання. Тобто сам по собі вектор контексту не буде нести інформацію про стиль, але вказуватиме на групу текстів, що мали б переклад подібний до того, який бажає отримати користувач.

Варто зазначити, що існують мережі як distiluse (дистильована сіамська XLM-R [40] для семантичного пошуку), яка може порівнювати тексти одразу кількома мовами, бо її уявлення для різних мов знаходяться в одному просторі показників. У випадку з distiluse вектор уявлень має 512 дробових чисел від -1 до 1. Така мережа може бути ефективна при масштабуванні підходу на більшу кількість мов, де початкова мова не є англійською. Тому рішення може масштабуватися на більшу кількість випадків, ніж переклади з англійської.

Хоч ця мережа вже і навчалася на багатьох багатомовних наборах даних і ймовірно бачила тексти, що можуть використовуватися для тренування перекладача, це не викликатиме перенавчання (overfit), адже результати семантичної моделі є лише додатковими вказівниками на бажані характеристики в перекладі.

3.3 Принцип використання запропонованого рішення

Однією з проблем запропонованого підходу є потреба у початковому повноцінному тренуванні отриманої мережі, адже через додавання нового шару та нових вхідних даних на стадії між кодувальником і декодувальником, втрачаються зв’язки між значеннями, які вже вивчив декодувальник і тими, що він почне отримувати в модифікованій моделі. Тобто шари конкатенації та зменшення розмірності змінюють результати кодувальника достатньо сильно, щоб порушити генерацію перекладу, а тому цей зв’язок двох мереж треба відновити для отримання нормальних перекладів. Процес тренування в цьому випадку буде простішим ніж зазвичай, адже ми ініціалізуємо старі шари мережі з її нормального, робочого стану, а тому основна оптимізація має відбуватися у блоці зменшення розмірності та відбору змінних. Наведемо приклад того, як сильно просідає якість машинного перекладу нової архітектури без тренування. Маємо наступний вхідний текст: «He has to come back in the next movie». Його зразковий переклад мав би бути таким: «Він має повернутися в наступному фільмі», проте мережа повертає текст, який абсолютно не пов’язаний з оригіналом, а саме: «Це означає, що ми маємо справу з іншими людьми, а не з ними.».

Отже, нова архітектура вимагає попереднього тренування, але далі має працювати за принципом few-shot learning, тобто формувати нові переклади,

маючи лише кілька зразків того, що від неї хочуть отримати. Наприклад, ми хочемо перекласти текст більш офіційно, тож для цього треба зміщувати вектор текстових характеристик у кластер офіційно-ділових зразків.

Тобто модель має отримувати вхідний текст та його вектор уявлень отриманий з моделі для семантичного пошуку у початковому вигляді або модифікованому, щоб додати наблизити його до певного кластеру текстів. Далі за рахунок наближеності цього вектору до інших у просторі результатів моделі для пошуку можемо перенести частину характеристик з наближених до вектору текстів на основний вхідний текст. Сам по собі вектор не несе саме стилістичні характеристики і є лише вказівником на схожі тексти і їх характеристики.

Для перенесення стилю певної групи текстів ми маємо створити середні вектори уявлень про цю групу. Тобто будемо рахувати кожний елемент середнього вектору кластеру текстів (формула 3.2).

$$feature_i = \frac{\sum_N^{j=0} vec_{ji}}{N} \tag{3.2}$$

де $feature$ – кінцевий вектор опису характеристики;

i – індекс елементу цього вектору характеристики, що приймає значення від 0 до розміру вектору семантичної моделі;

N – кількість текстів і їх відповідних векторів для обраної характеристики;

j – індекс вектору уявлень про текст, який приймає значення від 0 до N;

vec_{ij} – і-тий елемент j-того вектору-декскриптору.

Далі щоб наблизити уявлення про текст до цього кластеру маємо порахувати різницю оригінального вектора уявлень про вхідний текст і середнього вектору кластеру. Далі будемо рахувати лінійну комбінацію

оригінального вектору та вектору різниці оригіналу та середніх значень по кластеру (формула 3.3).

$$descriptor = V_{original} - \alpha * \left(V_{original} - V_{mean\ cluster}\right) \quad (3.3)$$

де $descriptor$ – кінцева комбінація оригіналу та різниці;

$V_{original}$ – оригінальний вектор уявлень про вхідний текст;

$V_{mean\ cluster}$ – вектор усереднених характеристик кластеру;

α – коефіцієнт ступеню трансформації, який позначає міру того як мають накладатися нові характеристики на оригінальний вектор.

Таким чином можемо комбінувати одразу характеристики кількох доменів за рахунок наявності усереднених векторів характеристик для кожного з них.

Покажемо це на прикладі. На рисунку 3.4 показано 2D проєкції векторів-уявлень текстів, а помаранчевий хрест позначає початкову позицію тексту «He came to the throne at the age of 73, an age when most people are thinking more about retirement than taking up a big and important job.» на цій площині. Він розташований поміж 3 доменів: законів, наукових статей і описів зображень. Спробуємо змістити цей текст до домену законів, щоб зробити його переклад більш офіційним. Для цього порахуємо середнє значення кожного елементу вектора-уявлення про закони і отримаємо новий узагальнений вектор уявлень про офіційно-ділові тексти. Порахуємо різницю між кожним елементом оригінального вектору вхідного тексту і цього узагальненого офіційного вектору. Домножимо ці значення на коефіцієнт α, який позначає масштаб бажаних змін (почнемо із значення 1.5 і будемо збільшувати на 1 до 9.5). Домножений вектор різниць додається до оригінального вектору. Великі

блакитні кола – нові отримані уявлення про вхідний текст із зміщенням до юридичного домену (рис. 3.4).

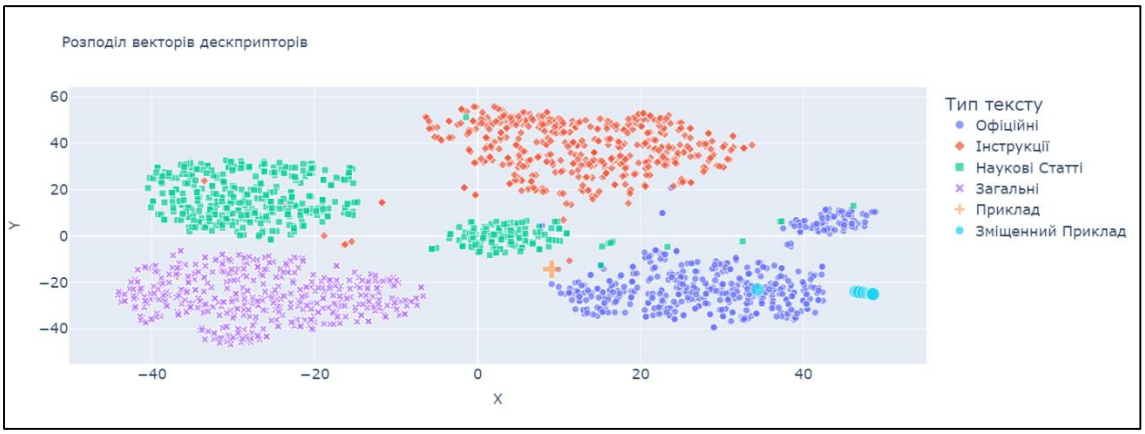

Рисунок 3.4 – Приклад зміщення тексту до заданого домену

Тож, за таким принципом модель має працювати для кожного тексту після навчання. Визначаємо вхідний текст та формуємо його вектор уявлень з моделі для семантичного пошуку, обираємо характеристики та значення коефіцієнту трансформації α для кожної з них. Виконуємо лінійні комбінації із векторами різниць оригінального вектору уявлень і усередненого вектору характеристики. Передаємо кінцевий отриманий вектор разом із текстом у модель і за рахунок цього комбінуємо вхідний текст із іншими, чиї властивості плануємо перейняти.

4 ПЛАН ЕКСПЕРИМЕНТУ

4.1 Умови проведення

Для навчання моделей та виконання експериментів було використано середовище з наступними характеристиками:

- CPU: Intel Core i7-10870H;
- RAM: 16 гігабайт;
- GPU: Nvidia RTX 2060 з 6 гігабайтами VRAM;
- операційна система Windows 10 з використанням WSL під Ubuntu 20.

Для роботи з моделями, модифікації їх поведінки та виконання тренування було використано мову Python 3.8 з окремим віртуальним середовищем під розробку та тренування моделей. Моделі використовують фреймворк PyTorch. Основа реалізації MarianMT була взята з бібліотеки huggingface transformers. Максимальна кількість вхідних і вихідних токенів – по 128 токенів, щоб зменшити час на навчання в описаному середовищі. Реалізації метрик взято з huggingface metrics і datasets. Для попередньої обробки даних та їх структуризації використано бібліотеку pandas.

4.2 Використані дані

Загалом у розділі 2.5 були дані загальні описи використаних наборів даних. Власноруч зібрані набори з українськими перекладами описів зображень з multi30k, переклади рефератів наукових статей, переклади документації VueJS і переклади законів із сайту Верховної Ради будуть використані як для навчання нової архітектури, так і для навчання окремих,

спеціалізованих версій наявної MarianMT для англо-українських перекладів, і для навчання з токеном-вказівником.

Частина наборів буде використана лише для тренування нової архітектури, щоб відновити зв’язок кодувальника та декодувальника і навчити шар зменшення розмірності. Це набори з OPUS. Вони вже були використані для тренування MarianMT під англо-українські переклади раніше, а тому не потрібні для інших варіантів моделі перекладу. Для запропонованої архітектури їх вистачить, щоб повернути її початкову якість. У таблиці 4.1 покажемо характеристики тренувальних наборів даних.

Таблиця 4.1 – Тренувальні набори даних

№	Назва набору	Опис текстів	Кількість пар
1	Multi30k українська версія	Описи фотографій з доволі простою структурою (суб’єкт, певна дія, опис оточення)	30 000
2	Переклади рефератів наукових статей	Містять описи складних тем, але часто використовують доволі прості терміни та мають легшу структуру за власне саму статтю.	2 000
3	Переклади законів	Багато спеціалізованої лексики, особлива структура та конструкція речень.	4 000
4	Переклади документації VueJS	Багато технічної лексики, структура подібна до інструкцій, що якісно відрізняє ці тексти від інших наборів.	2 500

Кінець таблиці 4.1

№	Назва набору	Опис текстів	Кількість пар
5	OPUS набори	Містять промови, енциклопедичні статті, субтитри фільмів і серіалів, огляди книг. Багато прикладів різних стилей, тем та використання однакових слів із різними сенсами.	1 942 849

Загалом маємо 1 мільйон 981 тисячу і 349 текстів. Для тестування моделей буде використано набір даних з 25% зібраних описів зображень, законів і наукових статей. Набір містить 8500 текстів і дозволить перевірити модель у випадках з різною структурою, лексичними формулюваннями і стилістикою тексту.

Отже, є 5 тренувальних наборів і 1 тестовий, які містять різні за стилем, темами і структурою тексти, що дозволять надати моделям достатньо інформації для контролю перекладу.

4.3 Критерії порівняння та принципи оцінки

Оберемо критерії порівняння моделей. Серед токен-метрик варто зупинитися на описаних до цього BLEU і METEOR. BLEU є стандартним методом перевірки та оцінки якості перекладу для всіх досліджень. Навіть попри проблеми з гнучкими мовами по типу української, BLEU може дати розуміння чи досягає алгоритм достатньої якості, щоб бути на рівні з людиною. Зазвичай BLEU беруть у межах від 0 до 100, але для простоти порівняння з іншими метриками масштабуємо його до інтервалу від 0 до 1. Тоді значення більше 0.2 і менше 0.4 зазвичай вважають перекладом, що вже

здатний передавати основні сенси, але має багато синтаксичних чи граматичних помилок. BLEU від 0.4 до 0.5 – це вже переклад рівня носія обох мов, проте він може бути не ідеально літературним. Значення від 0.5 до 1 буде літературним, професійним перекладом. Тому для нас у цьому дослідженні мінімальною планкою буде досягти BLEU, яке буде не менше за 0.35 (мінімальна кількість граматичних помилок, правильна передача сенсів). METEOR так само має дати оцінку за відповідністю токенів, проте буде швидше зростати за рахунок використання словника синонімів і зрізання закінчень. Якісним перекладом за METEOR вважається той, що набирає хоча б 0.3 з 1.

Використаємо BERT Score для порівняння за сенсом та стилем із еталонним перекладом.

Тож, маємо 2 токен метрики (BLEU, METEOR) і 1 метрику за уявленнями (BERT Score) у якості критеріїв порівняння моделей. Оцінка буде проводитися на описаному вище тестовому наборі.

Контрольованість перекладу має перевірятися за рахунок порівняння моделей як на всьому тестовому наборі, так і за рахунок заміру метрик для кожного домену окремо. Також має бути проведено серію експериментів з модифікації векторів контексту запропонованої моделі з метою перевірити їх вплив на результат та обсяг цього впливу. Заміряти такі переклади токен метриками чи через BERT Score буде неможливо, адже при зміні стилю чи структури перекладу ми зазвичай не маємо еталону для порівняння. Проте можемо скористатися мультимодальною мультимовною моделлю для семантичного пошуку, що тренувалася під відтворення уявлень про візуальні дані на основі їх текстових описів. Вона має заміряти якість перекладу лише з використанням оригіналу та власне самого машинного перекладу та повернути косинусну схожість двох текстів. Наближення до візуального

домену в теорії має полегшити роботу з метафорами та сталими виразами за рахунок наявності додаткового пояснення їх значень з іншої сфери, яка недоступна чисто лінгвістичним моделям. Косинусна схожість повертає оцінку від -1 до 1, де -1 – протилежні тексти, 0 – не пов’язані поміж собою, 1 – схожі. Будемо округляти оцінки нижче 0 до власне самого 0, бо для нас протилежні та непов’язані тексти є однаково поганими результатами.

5 АНАЛІЗ РЕЗУЛЬТАТІВ

5.1 Контроль через навчання на малому корпусі

У таблиці 5.1 показано результати всіх навчених версій моделі та оригіналу.

Таблиця 5.1 – Результати моделей за метриками на власному тестовому наборі

Модель	BLEU	METEOR	BERT Score F1
Оригінальна MarianMT	0.1120	0.2807	0.8115
Натренована на описах картинок	0.1270	0.3034	0.8380
Натренована на законах	0.2534	0.3861	0.8630
Натренована на рефератах наукових статей	0.1880	0.3347	0.8448
Натренована на всіх даних	0.3416	0.4754	0.8983

Ці результати доводять, що заявлена якість оригінальної моделі не відповідає дійсності при перекладі більш складних, специфічних текстів. Токен метрики доводять, що без додаткового тренування ця модель не здатна перекладати вузькопрофільні тексти якісно, адже вона робить багато синтаксичних і граматичних помилок. Проте BERT Score показує, що модель все одно здатна передати оригінальний сенс.

Тренування на описах фотографій слабко покращує результати моделі, що пояснюється простотою цих текстів. Зазвичай це прості, короткі речення, що містять лише опис дійової особи та саму дію і часом опис оточення на фото. У цих текстах відсутня рідкісна, специфічна, двозначна лексика, а структура майже завжди однакова і спрощена на фоні інших доменів. Ці тексти допомагають покращити загальну якість моделі та наситити її більшою кількістю лексики та форм вже відомих слів, але самі по собі вони не здатні сильно змінити поведінку моделі та її показники.

Версія натренована на перекладах законів зайняла друге місце на власному тестовому наборі. Складність цього домену пояснює суттєве покращення якості при тестуванні, бо модель стала здатна перекладати найбільш специфічні та неочевидні тексти з наявних у валідаційній вибірці, але погіршилася на більш звичайних, простих реченнях. Проте саме така поведінка від неї і очікувалася. Результати цього експерименту доводять, що навіть малий набір даних із певними унікальними характеристиками здатний змінити поведінку моделі та адаптувати її до нового домену чи стилю. При цьому BERT Score показує, що модель все ще не втратила сенс більшості текстів, хоч переклад не юридичних текстів і став гіршим.

Наступна версія тренувалася з використанням рефератів наукових статей. Цей експеримент не зміг повторити чи покращити якість юридичної моделі, що пояснюється специфікою саме рефератів. Їх мета – коротко і просто викласти основні ідеї наукової статті. У них зустрічається певна специфічна для домену лексика, але не в тих обсягах як у законах. Проте структура тексту суттєво ускладнюється на фоні описів фотографій, що дозволяє краще перекладати як і відривки рефератів, так і деякі більш прості речення із законів. Таким чином, це тренування також змогло внести відчутні зміни в поведінку моделі, а тому може вважатися успішним.

Очікувано модель, яка отримала всі тексти дала найкращий результат за всіма метриками, але навіть вона не може досягти якості людини зі знанням обох мов на високому рівні (BLEU більше за 0.4). BERT Score доводить, що модель здатна перекласти значення оригінального англійського тексту з невеликою кількістю помилок (неправильні форми слова або зайвий і надмірний переклад власних назв), що доводять значення токен метрик. Проте хоч ця модель і досягла найкращих результатів за метриками, вона не відповідає потребам контрольованого перекладу, адже приймає рішення про стиль та результат перекладу сама.

В середньому одне покоління MarianMT у описаному середовищі вчиться по 6 хвилин, коли приймає до двох тисяч текстів, а тому сам процес додаткового навчання не є довготривалим та важким з точки зору часу та ресурсів на його виконання.

На рисунку 5.1 покажемо функцію для попередньої обробки текстів перед їх подачею до звичайної MarianMT моделі. Тут нам потрібно тільки токенізувати та зробити маску для оригінального тексту і токенізувати фінальний текст. Для оригіналу створюються масиви однакового розміру для ідентифікаторів токенів (цілих чисел, що відповідають певному набору символів) і масив масок, який у цьому випадку завжди заповнюється одиницями до спеціального токену, що позначає завершення тексту, адже всі не службові токени мають бути доступні моделі. Якщо текст токенізується у більше ніж максимальну кількість токенів (128 у цьому випадку), то він обрізається. На виході маємо словник з трьома масивами, які описують кожний текст.

```
def preprocess_function(examples):
    inputs = examples['input']
    targets = examples['target']
    model_inputs = tokenizer(
        inputs,
        max_length=max_input_length,
        truncation=True
      )

    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            targets,
            max_length=max_target_length,
            truncation=True
          )

    model_inputs['labels'] = labels['input_ids']
    return model_inputs
```

Рисунок 5.1 – Токенізація та маскування текстів

Також покажемо код розрахунку метрик на валідаційному та тестовому наборах після навчання кожного покоління. Функція має рахувати BLEU, METEOR, BERT F1 Score і середню кількість токенів у кожному згенерованому перекладі. Вона повертає словник, де кожний ключ відповідає одному з розрахованих значень. Значення округляються до 4 знаків після коми. Для обрахунку METEOR попередньо перед навчанням мережі також необхідно завантажити словник синонімів та обрати стемер для мови текстів (алгоритм зрізання закінчень слова). Для BERT Score необхідно завантажити матрицю вагових коефіцієнтів багатомовного BERT і словник токенізатору для цієї моделі. Далі huggingface Trainer автоматично конвертує словник результатів у pandas Dataframe для кращої візуалізації результатів у вигляді таблички (рис. 5.2).

```python
def compute_metrics(eval_preds):
    preds, labels = eval_preds

    if isinstance(preds, tuple):
        preds = preds[0]
    decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)

    # Replace -100 in the labels as we can't decode them.
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # post-processing
    decoded_preds, decoded_labels = postprocess_text(decoded_preds, decoded_labels)

    result = {}
    for metric_name, metric in string_metrics.items():
      result[metric_name] = metric.compute(
          predictions=decoded_preds,
          references=decoded_labels
        ).get('score' if metric_name == 'sacrebleu' else 'meteor')

    for metric_name, metric in embedding_metrics.items():
      bert_score = metric.compute(
          predictions=decoded_preds,
          references=decoded_labels,
          model_type='bert-base-multilingual-cased'
        )
      result[f'{metric_name}_precision'] = np.mean(bert_score.get('precision'))
      result[f'{metric_name}_recall'] = np.mean(bert_score.get('recall'))
      result[f'{metric_name}_f1'] = np.mean(bert_score.get('f1'))

    prediction_lens = [np.count_nonzero(pred != tokenizer.pad_token_id) for pred in preds]
    result["gen_len"] = np.mean(prediction_lens)
    result = {k: round(v, 4) for k, v in result.items()}
    return result
```

Рисунок 5.2 – Функція розрахунку метрик

Тож, були навчені окремі моделі під кожний домен. Вони будуть використані як еталони для запропонованої архітектури, адже мають добре вивчати конкретний набір текстів і його стиль, але для контролю вимагають повноцінного тренування.

5.2 Контроль через додавання спеціального токену

У таблиці 5.2 наведено результати моделі без додаткового тренування із додаванням спеціального токену з позначкою теми і такої самої моделі, яка тренувалася на всіх зібраних текстах із додатково проставленим токеном.

Таблиця 5.2 – Результати моделей із спеціальним токеном теми на тестовому наборі

Модель	BLEU	METEOR	BERT Score F1
Оригінальна MarianMT	0.1120	0.2807	0.8115
Спеціальний токен без тренування	0.1172	0.3086	0.8085
Спеціальний токен з тренуванням	0.3708	0.4923	0.9019

Додавання маркеру стилю не дало суттєвого приросту за метриками і навіть погіршило результат за BERT Score. Це можна пояснити появою нового токену, який вказує на потрібний стиль, але так само перекладається моделлю як і будь-яке інше слово в оригінальному тексті, хоча у цільовому тексті цей токен не перекладається.

Проте тренування із наявним спеціальним токеном перед текстом суттєво покращує метрики у порівнянні з оригінальною моделлю та моделлю, що просто навчалася на всьому наборі даних без жодних додаткових маркерів (останній модель у таблиці 6.1). Маркер тепер не перекладається і дає моделі додаткову вказівку на бажаний стиль перекладу, що дозволяє краще розрізняти запропоновані домени (закони, загальні тексти і наукові статті).

Отже, такий підхід дозволяє краще розрізняти стиль чи тему текстів з метою зміни перекладу. Проте для додавання будь-якої нової характеристики модель прийдеться повністю перенавчати з новими спеціальними токенами. Для будь-якої категорії чи її підрозділу потрібно буде створювати новий

маркер, розмічати тексти власноруч і або навчати модель повністю з нуля, якщо зміни в категоріях є суттєвими, або донавчити модель на малому наборі для нової категорії. Також цей підхід не дозволяє змінити ступінь змін у тексті, адже маркер завжди буде змінювати характеристики перекладу фіксованим чином.

5.3 Навчання запропонованої архітектури

Наступним кроком було навчання нової, запропонованої архітектури з метою відновити зв’язок кодувальника і декодувальника. В середньому на кожну епоху при двох мільйонах тренувальних текстів йде по 8 годин, тож було натреновано 5 поколінь, чого власне і вистачило для повноцінного відновлення показників моделі та з додаванням механізму контролю.

У таблиці 5.3 показано результати навчання п’яти епох запропонованої архітектури та пораховані на власному тестовому наборі з текстами трьох стилей.

Таблиця 5.3 – Результати навчання запропонованої архітектури

Стан моделі	BLEU	METEOR	BERT Score F1
Оригінальна MarianMT без дотренування	0.1120	0.2807	0.8115
Запропонована модифікація MarianMT без навчання	0.0002	0.0147	0.5859

Кінець таблиці 5.3

Стан моделі	BLEU	METEOR	BERT Score F1
Покоління №1	0.2845	0.4387	0.8848
Покоління №2	0.3250	0.4627	0.8935
Покоління №3	0.3422	0.4730	0.8977
Покоління №4	0.3509	0.4781	0.8998
Покоління №5	0.3714	0.4930	0.9021

Отже, п’яте покоління запропонованої модифікації класичних кодувальників-декодувальників змогло не тільки відновити зв’язок між шарами оригінальної мережі, а й обійти кращу дотреновану версію оригінальної MarianMT, результати якої були в таблиці 5.1. Також результати за метриками майже аналогічні отриманим значенням для моделі, яка навчалася на всіх зібраних текстах із додаванням спеціального токену-маркеру стилю (таблиця 5.2).

Варто зазначити, що BERT Score проявив себе погано як метрика для перекладів у цьому випадку, адже він дає середню оцінку навіть не натренованій версії нової, запропонованої архітектури, хоча за токен метриками і суб’єктивними оцінками її переклади абсолютно не відповідають оригінальним текстам. Цьому є одразу кілька пояснень. BERT Score у своїй реалізації використовує звичайний багатомовний BERT, який працює для української мови гірше за будь-яку спеціалізовану модель по типу UkrROBERTA чи XLM-R [41]. Інша причина – він порівнює окремі токени, а тому може просто знайти певні схожі частини слів чи слова, які насправді не дають правильний переклад, схожий на еталонний. Токенізатор мультимовного BERT містить мало токенів, що описують хоча б корінь українського слова чи його значну частину. У більшості випадків він б’є текст

навіть не на слова, а на певні набори символів, які самі по собі не мають значного сенсу. Ці однакові токени все одно отримуватимуть різні вектори уявлень, адже механізм уваги враховує зв’язки кожного токену з усіма іншими в тексті, але вони все одно будуть близькі один до одного. Тому навіть абсолютно протилежні переклади можуть отримати високу оцінку, адже будуть розбиті на подібні набори токенів, що можуть отримати схожі вектори уявлень [42].

Наприклад, повернемося до випадку описаного у розділі 3.3, а саме неправильного перекладу тексту «He has to come back in the next movie». BERT Score дає оцінку схожості еталонного тексту «Він має повернутися в наступному фільмі» і перекладу «Це означає, що ми маємо справу з іншими людьми, а не з ними.» як 0.6341. Оцінка схожості двох таких текстів мала би бути близькою до 0, але вона ближче до максимального значенням метрики, ніж до мінімального. Це доводить що використання метрик за уявленнями без додаткового заміру токен метрик не може дати якісну оцінку моделей машинного перекладу.

Переглянемо модифікації коду, які були необхідні щоб запустити нову архітектуру. У досліджені було використано версію MarianMT з бібліотеки huggingface, а тому було вирішено модифікувати код їх Marian моделі, щоб зберегти ваги для вже навчених шарів. Для цього створимо клас MarianStyledModel, який наслідує клас MarianModel. У конструкторі класу з’явиться 1 новий параметр для розміру вектору уявлень. На його основі створимо лінійний шар, який отримує на вхід вектори розміру суми розмірності Marian кодувальника і розмірності вектору уявлення, а на виході повертає вектор розмірності кодувальника Marian. Також створимо SiLU активацію, бо ця функція має механізм самостабілізації та може виступати як

додатковий регуляризатор для шарів, що мають ваги високої величини (рис. 5.3).

```
class MarianStyledModel(MarianModel):
  def __init__(
      self,
      config: MarianConfig,
      topic_embed_size=384
      ):
    super().__init__(config)
    self.feature_reducer = nn.Linear(
        config.d_model + topic_embed_size,
        config.d_model
      )
    self.feature_activation = nn.SiLU()
```

Рисунок 5.3 – Конструктор класу MarianStyledModel

Далі маємо створити метод для додавання вектору уявлень з моделі для семантичного пошуку на кожний рядок матриці уявлень про токени вхідного тексту. Матриця уявлень про токени є тривимірним тензором (розмір порції на максимальну кількість токенів на розмір уявлення про токен), тоді як уявлення із семантичної моделі є двовимірним тензором (розмір порції на розмір вектору уявлення). Тож спочатку повторимо вектор отриманий із моделі для семантичного пошуку N разів, де N – максимальна кількість токенів, яку приймає модель (у випадку цього дослідження це 128). Далі змінимо форму цього тензора повторень, який має форму розмір порції на добуток максимальної кількості токенів і розміру вектора уявлень. Отримаємо тензор форми розмір порції на розмір вектора уявлень на максимальну кількість токенів. Транспонуємо цей тензор, щоб переставити останні 2 виміри місцями. Після цього можемо з’єднати отриманий тривимірний тензор з

тензором-результатом кодувальника. На виході матимемо тензор форми розмір порції на максимальну кількість токенів на суму розмірності уявлень про токени і розмірності вектору уявлень про текст з моделі для семантичного пошуку (рис. 5.4).

```
def add_style_embedding(self, encoder_output, topic_embedding):
  # The encoder output is a 3D tensor with size (batch_size, num_tokens, marian_embed_dim)
  # Style/topic embedding is also a 2D tensor with size (batch_size, style_embedding_size)
  # This concats the style embedding for each token in the encoder output

  batch_size = encoder_output.size()[0]
  num_tokens = encoder_output.size()[1]

  if topic_embedding.dim() == 1:
      idx = 0
  else:
      idx = 1

  # Start by making a copy of each style embedding to match the input text length
  # The output of this has size (batch_size, num_tokens * marian_embed_dims)
  topic_embeddings_size = topic_embedding.size()[idx]
  e = topic_embedding.repeat_interleave(num_tokens, dim=idx)

  # Reshape it and transpose
  e = e.reshape(batch_size, topic_embeddings_size, num_tokens)
  e = e.transpose(1, 2)

  # Concatenate the tiled style embedding with the encoder output
  encoder_output_style_transfered = torch.cat((encoder_output, e), 2)
  return encoder_output_style_transfered
```

Рисунок 5.4 – Код об’єднання уявлень про токени і уявлення про текст

Далі залишається модифікувати метод forward моделі. Код залишиться таким самим, але між викликом кодувальника та декодувальника додаються ще 2 кроки: об’єднання матриці уявлень про токени з уявленням про текст і зменшення розмірності рядків цієї матриці до розмірності результатів кодувальника. Перший крок було реалізовується методом add_style_embedding, що було описано раніше. Для другого кроку як раз

знадобиться лінійний шар і активація, які були створені в конструкторі. Викликаємо їх уже на тензорі отриманому з add_style_embedding та повертаємо тензор до розмірності, для якої будувався декодувальник. Таким чином можемо зберегти вже натреновані 12 шарів у кодувальнику і декодувальнику і все одно модифікувати їх роботу через конкатенацію з додатковим тензором (рис. 5.5).

```
if encoder_outputs is None:
    encoder_outputs = self.encoder(
        input_ids=input_ids,
        attention_mask=attention_mask,
        head_mask=head_mask,
        inputs_embeds=inputs_embeds,
        output_attentions=output_attentions,
        output_hidden_states=output_hidden_states,
        return_dict=return_dict,
    )
# If the user passed a tuple for encoder_outputs, we wrap it in a BaseModelOutput when return_dict=True
elif return_dict and not isinstance(encoder_outputs, BaseModelOutput):
    encoder_outputs = BaseModelOutput(
        last_hidden_state=encoder_outputs[0],
        hidden_states=encoder_outputs[1] if len(encoder_outputs) > 1 else None,
        attentions=encoder_outputs[2] if len(encoder_outputs) > 2 else None,
    )

encoder_output_with_style_features = self.add_style_embedding(
    encoder_outputs[0],
    topic_embedding
)
reduced_encoder_output = self.feature_reducer(
    encoder_output_with_style_features
)
swish_encoder_output = self.feature_activation(
    reduced_encoder_output
)
```

Рисунок 5.5 – Модифікація матриці результатів кодувальника у методі forward для додавання контекстного вектору

Зміни в архітектурі також спричинять зміни в коді попередньої обробки текстів перед навчанням моделі. Тепер до словнику вхідних даних додається новий ключ topic_embedding, який заповнюється списком векторів уявлень з моделі для семантичного пошуку про оригінальні англомовні тексти (рис. 5.6).

```
def preprocess_function(examples):
    inputs = examples['eng']
    targets = examples['ukr']
    model_inputs = tokenizer(
        inputs,
        max_length=max_input_length,
        truncation=True,
        padding=True
      )

    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            targets,
            max_length=max_target_length,
            truncation=True,
            #padding=True
          )

    model_inputs['labels'] = labels['input_ids']
    model_inputs['topic_embedding'] = topic_model.encode(
        examples['eng'],
        convert_to_numpy=False,
        convert_to_tensor=True
      )
    return model_inputs
```

Рисунок 5.6 – Код попередньої обробки тренувальних і тестових даних

В описаному середовищі обробка тренувального набору даних з усіма майже двома мільйонами текстів займала до 35 хвилин.

Також знадобиться мінімальна модифікація класу Seq2SeqTrainer, який власне виконує тренування моделі. У ньому треба всього лише модифікувати метод prediction_step і додати передачу аргументу topic_embedding до моделі. Загалом жодна інша частина цього класу не має змінюватися, щоб запрацювати із новими вхідними аргументами (рис. 5.7).

```
if "attention_mask" in inputs:
    gen_kwargs["attention_mask"] = inputs.get("attention_mask", None)
if "topic_embedding" in inputs:
    gen_kwargs["topic_embedding"] = inputs.get("topic_embedding", None)
if "global_attention_mask" in inputs:
    gen_kwargs["global_attention_mask"] = inputs.get("global_attention_mask", None)
```

Рисунок 5.7 – Модифікація Seq2SeqTrainer для тренування запропонованої архітектури

Тож, за результатами цього етап експерименту було натреновано запропоновану в дослідженні архітектуру контрольованих перекладів та порівняно її ефективність із окремими версіями MarianMT через замір токен метрик і метрик за уявленнями на тестовому наборі з кількома різними стилями і темами.

5.4 Перевірка керованості перекладів

Наступним кроком буде перевірка контрольованості теми та стилю для всіх навчених моделей із їх порівнянням. Для цього поділимо тестовий набір даних на окремі набори для кожного домену та зробимо замір для кожного з них. Таким чином зможемо перевірити якість того як кожна модель розрізняє стилі та змінює свої переклади, щоб підлаштуватися під заданий домен [43].

Почнемо з домену законів та юридичних текстів, адже він є найбільш складним серед зібраних. Він містить багато як спеціалізованої лексики, так і звичайної, але не в стандартному значенні, тому результати саме для цього набору є найбільш цікавими серед наявних. У таблиці 5.4 наведено результати заміру метрик для текстів цього домену з тестового набору.

Таблиця 5.4 – Результати моделей для юридичних текстів і законів

Модель	BLEU	METEOR	BERT Score F1
Оригінальна MarianMT без дотренування	0.0806	0.2444	0.7778
MarianMT натренована на законах	0.4948	0.6044	0.9247
MarianMT натренована на всіх даних	0.4860	0.5987	0.9239
Спеціальний токен без тренування	0.0910	0.2764	0.7862
Спеціальний токен з тренуванням	0.5193	0.6141	0.9285
Модифікована MarianMT з векторами контексту	0.5325	0.6473	0.9303

Запропонована модель із додатковими векторами контексту перемагає всі інші моделі, що були протестовані. Ця архітектура навіть перемогла модель, що додатково тренувалася виключно на законах. Також варто зазначити, що результати моделі натренованої на всіх текстах із додаванням спеціального токену є також вищими за окрему модель для законів та модель

натреновану на всіх текстах без жодних маркерів. Ця модель отримала більше додаткових знань української мови з інших доменів, але за рахунок маркеру продовжує чітко розмежовувати допоміжні тексти та власне необхідний домен. І модель натренована з маркером, і модель натренована з контекстними векторами отримали BLEU вищий за 0.5 (50), а отже ці моделі здатні надавати якісні, людиноподібні переклади юридичних текстів.

У таблиці 5.5 наведемо результати заміру метрик на загальних текстах (описи фотографій).

Таблиця 5.5 – Результати моделей для загальних текстів

Модель	BLEU	METEOR	BERT Score F1
Оригінальна MarianMT без дотренування	0.2290	0.3264	0.8599
MarianMT натренована на описах картинок	0.4240	0.4083	0.9181
MarianMT натренована на всіх даних	0.4006	0.3948	0.9128
Спеціальний токен без тренування	0.2202	0.3730	0.8428
Спеціальний токен з тренуванням	0.4089	0.4029	0.9164

Кінець таблиці 5.5

Модель	BLEU	METEOR	BERT Score F1
Модифікована MarianMT з векторами контексту	0.5346	0.5301	0.9264

Запропонована модель втримує високу якість перекладу на загальних текстах також. Вона все ще обходить інші протестовані моделі для цього домену та досягає BLEU вищого за 0.5, а отже все ще надає людиноподібні переклади з мінімальною кількістю синтаксичних чи пунктуаційних помилок. Цікаво відмітити, що додавання токену-маркеру без тренування навпаки погіршило результати MarianMT. Ймовірно це було викликано появою токену, що не мав би перекладатися, але модель без тренування все одно намагалася знайти йому аналог серед українських слів. При цьому моделі без векторів контексту, які також тренувалися на всіх текстах, програли моделі, яка навчалася лише на описах картинок. Це сталося через додавання великої кількості текстів із специфічною лексикою та структурою, які сильно ускладнені в порівнянні з прямолінійними описами картинок, які мають однакову структуру з суб’єкту, дії та короткого опису оточення чи самого суб’єкту. Токен-маркер все одно не дозволив достатньо відділити спеціалізовані тексти від загальних, хоча Marian і реалізує двонаправлений кодувальник, який мав би врахувати вплив цього токену на всі інші.

Останній домен – наукові статті, а саме їх реферати з коротким викладанням змісту роботи. У таблиці 5.6 наведено результати моделей для домену наукових статей (рефератів до наукових статей).

Таблиця 5.6 – Результати моделей для наукових статей

Модель	BLEU	METEOR	BERT Score F1
Оригінальна MarianMT без дотренування	0.1094	0.2710	0.7956
MarianMT натренована на рефератах наукових статей	0.2193	0.4127	0.8495
MarianMT натренована на всіх даних	0.2342	0.4291	0.8548
Спеціальний токен без тренування	0.1089	0.2736	0.7952
Спеціальний токен з тренуванням	0.2522	0.4523	0.8648
Модифікована MarianMT з векторами контексту	0.2664	0.4686	0.8618

Для цього домену всі протестовані моделі дали погані результати, хоча запропонована архітектура все ще перемагає інші варіанти. Низьке значення BLEU вказує на велику кількість граматичних і лексичних помилок, проте моделі з векторами контексту та з токеном-маркером мають METEOR, який

значно вищий за BLEU. Це вказує на використання синонімів чи однокореневих слів замість тих, що визначені в еталонному тексті. Це також підтверджується високим BERT Score. Проблеми можуть бути викликані ускладненою структурою українських перекладів, адже вони часто складаються з одного складного речення, коли англійський оригінал містить кілька простих речень. Також варто відмітити, що модель натренована на всіх текстах перемагає модель натреновану лише на рефератах. У випадку цього домену дані з інших галузей навпаки допомагають підвищити якість робити, адже доповнюють базу знань моделі для окремих підрозділів цієї великої категорії.

Отже, запропонована модель перемагає інші протестовані варіанти моделей машинного перекладу в усіх трьох підібраних доменах. Для двох з трьох галузей якість перекладів відповідає роботі людини, при цьому модель добре розмежовує різні групи та їх стильові та лексичні характеристики. Також важливо відмітити, що модель справляється з перекладом вузькопрофільних, спеціалізованих текстів краще за окремо натреновані варіанти MarianMT саме для заданої сфери.

5.5 Аналіз результатів запропонованої архітектури

Фінальним кроком буде перевірка перекладів запропонованої моделі з різними налаштуваннями векторів контексту. Кожний згенерований переклад буде перевірятися мультимодальною мультимовною моделлю для семантичного пошуку, що має принаймні дозволити на скільки новий переклад близький до оригінального англійського тексту.

Для цих експериментів також було визначено прийнятні значення коефіцієнту трансформації. Виходячи з результатів було вирішено варіювати

коефіцієнт на проміжку від 3.5 до 5.5. Вищі значення коефіцієнту часто призводять до поламок у поведінці моделі, як то повторення одного слова чи видача набору символів замість дійсного чи хоча б частково неправильного перекладу. Це ставалося через появу великої кількості значень вищих за 1 у матриці уявлень про токени, що порушувало подальшу поведінку блоку зменшення розмірності і декодувальника, який зазвичай приймав значення від -1 до 1. Значення менші за 3.5 часто не призводили до якихось значних змін кінцевого перекладу. Модель могла змінити кілька сполучників чи форму слова, але не давала суттєвих стилістичних оновлень.

Для тестування модифікації стилю перекладів створимо набір усереднених векторів уявлень про певні характеристики. Відберемо для тестування наступні стилі: офіційні тексти, документація та інструкції, загальні тексти, жарти, стара література, статті. Для створення 6 усереднених векторів зберемо тексти, що будуть відповідати характеристикам. Створимо для кожного окремого тексту вектор уявлення через модель для семантичного пошуку, яку використовуємо як джерело векторів контексту моделі (сіамський BERT, який повертає вектор на 384 елементи). Далі порахуємо середнє значення для кожної з 384 позицій у кожній з категорій. Отримаємо 6 векторів по 384 елементи, які будуть описувати середнє уявлення про характеристику.

Важливо відмітити, що уявлення про характеристики мають формуватися з текстів мовою джерела, тобто у нашому випадку ми збираємо англійські тексти та усереднюємо їх вектори уявлень, а потім використовуємо їх для розрахунку векторної різниці та лінійної комбінації різниці та оригінального вектору тексту, що перекладається. Для офіційних текстів візьмемо вже зібрані закони і відберемо з них 1000 випадкових речень. Для загальних текстів також відберемо 1000 випадкових описів картинок. Для інструкцій та документацій збираємо 1000 речень з документації фреймворку

VueJS. Для старої літератури зберемо 1000 випадкових речень зі старих літературних творів доступних на Project Gutenberg. Для жартів збираємо також 1000 речень з онлайн версії журналу для вчителів англійської як другої мови (The Internet TESL Journal). Для статей скористаємося вже зібраними рефератами і відберемо з них 1000 випадкових речень.

Почнемо з простого прикладу: «Give my money back». Це речення може перекладатися загально або більш офіційно чи в стилі інструкції. Усі такі переклади були б валідними для оригінального тексту, а тому спробуємо зміщувати вектор контексту та змінити стиль результату. Якщо не модифікувати вектор жодним чином і просто перекласти текст з використанням натренованої запропонованої моделі, то отримаємо наступний результат: «Поверни мої гроші» (оцінка схожості від моделі для семантичного пошуку: 0.9935). Якщо змістити вектор до домену офіційних текстів з коефіцієнтом 3.5, то отримаємо «Поверніть мої гроші» (оцінка схожості: 0.9940). Модель замінила форму дієслова і тепер звертається на ви, при цьому у перекладі не з’явилися якісь нові помилки, сенс було збережено, а оцінка семантичної моделі не зменшилася. Якщо підвищити коефіцієнт для домену офіційних текстів, то маємо переклад «Віддайте мої гроші назад» (оцінка схожості: 0.9957). Змінилася структура тексту та формулювання запиту, зберіглося офіційне звернення. Якщо ж змістити текст до домену старої літератури з коефіцієнтом 5.0, то можемо отримати «Віддайте мені мої гроші.» (оцінка схожості: 0.9877). Тепер модель пробує додавати пунктуацію, зберігає більш офіційний стиль спілкування, але змінює структуру звернення ще раз на фоні всіх попередніх варіантів. Також спробуємо перекласти текст зі зміщенням в домен інструкцій та документацій з коефіцієнтом 5.5, то отримаємо «Повернути мої гроші» (оцінка схожості: 0.9937). Модель

поставила дієслово у неозначену форму та зробила запит справді схожим на інструкцію, а не на пряме звернення до співрозмовника.

Спробуємо інший текст: «Then, about seven years after the gold rush began, it finished». Без жодних модифікацій модель дає переклад «Через сім років після початку золотої лихоманки все закінчилося» (оцінка схожості: 0.9755). Спробуємо комбінувати вектор уявлень цього текст з векторами старої літератури і статей з коефіцієнтами 3.5. Отримаємо наступний текст: «Потім, близько семи років після початку золотої лихоманки, вона завершилася» (оцінка схожості: 0.9801). Модифікація вектору уявлень зробила переклад ближчим за формулюванням до оригінального англійського варіанту. Спробуємо візуалізувати комбінування векторів для модифікації стилю. Покажемо кластери законів, статей, документації і загальних текстів. Також покажемо оригінальний текст і його модифіковані варіанти. Жовтий хрест позначає початкове положення вектору уявлень про вхідний текст на цій двовимірній проєкції. Голубі кола показують комбінації цього вектору із середнім вектором статей з коефіцієнтами від 1.5 до 9.5. При 1.5 вектор опиняється серед власне наукових статей, а при 2.5 і далі він зміщується в положення поміж статтями і законами. Тобто за рахунок положення вектору уявлень про текст у цьому просторі ми можемо підказати моделі властивості, яких текстів ми хочемо перенести на цей конкретний переклад [44]. Таким чином сам по собі вектор не несе в собі характеристики стилю, але дозволяє порівняти вхідний текст з іншими та підібрати найбільш правильний відповідник, який мав би як оригінальні властивості тексту, так і властивості заданих прикладів (рис. 5.8).

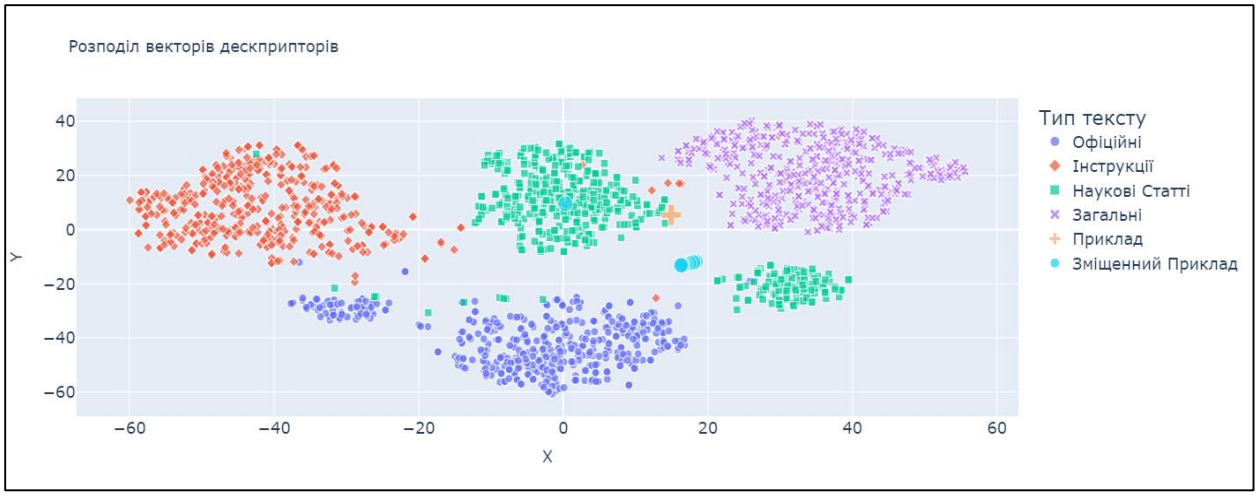

Рисунок 5.8 – Проєкція векторів уявлень тренувальних текстів і вхідного тексту та його лінійних комбінацій із середнім вектором-уявленням наукових статей

За рахунок порівняння текстів по додатковому контекстному вектору модель здатна підбирати нові варіанти перекладу під задані умови: зробити текст більш офіційним, зробити мовлення у тексті у більш старому стилі, спростити лексику і структуру перекладу чи зробити його на манер інструкції.

Спробуємо ще один приклад: «What are you going to eat with your sandwich?». Початковий переклад: «Що ти їстимеш зі своїм бутербродом?» (оцінка схожості: 0.9892). Спробуємо комбінувати вектор цього тексту із вектором загальних текстів з коефіцієнтом 5.5. Отримаємо текст: «Що ти будеш їсти зі своїм сендвічем?» (оцінка схожості: 0.9907). Тепер модель використала англіцизми, змінила формулювання дії.

Тож, модель здатна комбінувати характеристики різних текстів із вхідним текстом для перекладу, щоб на виході перекласти його з іншою структурою, набором слів, формулюваннями та стилістикою. Для цього їй потрібна лише бібліотека прикладів для різних характеристик, які користувач хотів би змінювати.

З попереднього висновку випливає, що можна спробувати комбінувати стилі в моделі без середніх векторів кожної окремої характеристики, яку треба модифікувати. Замість цього можна було б просто надати текст-приклад. Тоді модель могла б зробити трансфер його характеристик і характеристик подібних до нього текстів на вхідний текст для перекладу. Але наразі тестування довело, що модель нездатна модифікувати переклад з одного прикладу. Наприклад перший текст у цьому розділі «Give my money back» завжди отримує оригінальний переклад «Поверни мої гроші», який би текст-приклад не отримала модель. Ймовірно це викликано нестачею тренувальних даних, бо навіть 2 мільони текстів для такої моделі не є значним набором. У дослідженні було використано лише частину OPUS наборів через обмеження середовища для розрахунків. Тому припущення про використання моделі лише з одним прикладом замість модифікації окремих характеристик усе ще залишається можливим, але для його спростування чи підтвердження потрібне більш потужне середовище з більшою кількістю відеокарт і оперативної пам’яті.

Також у таблиці 5.7 наведемо більше прикладів роботи моделі для різних текстів з різними модифікаціями характеристик.

Таблиця 5.7 – Приклади роботи моделі

№	Оригінал	Переклад	Модифікації контекстного вектору	Оцінка схожості
1	acquire ownership of intellectual property rights	набуття права інтелектуальної власності	Відсутні	0.9714

Продовження таблиці 5.7

№	Оригінал	Переклад	Модифікації контекстного вектору	Оцінка схожості
2	acquire ownership of intellectual property rights	набути право власності на інтелектуальну власність	Лінійна комбінація з юридичними текстами з коефіцієнтом 3.5	0.9748
3	I began asking the students themselves to compile multiple translations of a single poem for class presentation.	Я почала просити студентів скласти кілька перекладів одного вірша для презентації класу.	Відсутні	0.9810
4	I began asking the students themselves to compile multiple translations of a single poem for class presentation.	Я почав просити самих студентів скомпілювати кілька перекладів одного вірша для презентації класу.	Комбінація із загальними текстами, де суб’єктом є чоловік. Коефіцієнт 4.5.	0.9849
5	In case you broke something you must pay for this	Якщо ти щось зламав, ти повинен заплатити за це	Відсутні	0.9812

Продовження таблиці 5.7

№	Оригінал	Переклад	Модифікації контекстного вектору	Оцінка схожості
6	In case you broke something you must pay for this	Якщо ви розбили що-небудь, ви повинні відповісти	Офіційний з коефіцієнтом 7	0.9683
7	I began asking the students themselves to compile multiple translations of a single poem for class presentation.	Я почав вимагати від студентів складання декількох перекладів єдиного вірша для презентації класу.	Попередня модифікація, а також комбінація з юридичними текстами з коефіцієнтом 3.5.	0.9854
8	why don’t you come sit down with me?	чому б тобі не присісти зі мною?	Відсутні	0.9766
9	why don’t you come sit down with me?	чому ви не приєднаєтеся до мене?	Комбінація з юридичними текстами з коефіцієнтом 5.5	0.9137
10	Do you want to hear a dirty joke? Ok. A white horse fell in the mud.	Ви хочете почути брудний жарт? Гаразд. Білий кінь впав у грязюку.	Відсутні	0.9692

Кінець таблиці 5.7

№	Оригінал	Переклад	Модифікації контекстного вектору	Оцінка схожості
11	Do you want to hear a dirty joke? Ok. A white horse fell in the mud.	Хочете почути грязну анекдоту? Гаразд. У грязюку впав білий кінь.	Комбінація з жартами і зі старою літературою з коефіцієнтами по 4.5	0.9720
12	Excuse me. Do you know the way to the zoo?	Вибачте, ви знаєте шлях до зоопарку?	Відсутні	0.9715
13	Excuse me. Do you know the way to the zoo?	Вибачте, ви знаєте, як пройти до зоопарку?	Комбінація із загальними текстами з коефіцієнтом 5.5	0.9702

У цих прикладах можна побачити як комбінування з різними доменами призводить до як незначних змін форм окремих слів, так і до повної зміни настрою тексту, його структури та формулювання думок. Оцінки схожості отримані із багатомовної мультимодальної моделі для семантичного пошуку не відрізняються суттєво між оригінальним перекладом моделі та модифікованими версіями, що свідчить про збереження сенсів.

На рисунку 5.9 показано код функції для комбінування векторів уявлень текстів.

```python
def transform_text_embedding(original_text, features=None, example_text=None):
    if features is not None:
        text_embedding = semantic_model.encode(
            [original_text],
            convert_to_numpy=False,
            convert_to_tensor=True
        )
        text_embedding = text_embedding.to('cpu')
        for text_feature in features:
            feature_embed = feature_embeddings[text_feature.name]
            text_embedding[0] = text_embedding[0] - ((text_embedding[0] - feature_embed) * text_feature.value)
    elif example_text is not None:
        text_embedding = semantic_model.encode(
            [example_text],
            convert_to_numpy=False,
            convert_to_tensor=True
        )
    else:
        text_embedding = semantic_model.encode(
            [original_text],
            convert_to_numpy=False,
            convert_to_tensor=True
        )
    return text_embedding.to('cpu')
```

Рисунок 5.9 – Функція для комбінування векторів уявлень текстів

У цій функції формуємо вектор уявлень про текст і можемо або не модифікувати його взагалі, або замінити його на текст приклад (що не дало бажаних результатів), або комбінувати його з усередненими векторами доменів за описаними у розділі 3.3 формулами. Комбінування відбувається послідовно за наданим списком словників, які містять назву домену і коефіцієнт α (коефіцієнт трансформації). На назвами підбираються відповідні усереднені вектори. При використанні тексту-прикладу просто береться його вектор без жодних додаткових модифікацій значень.

На рисунку 5.10 покажемо код перекладу тексту із контролем його характеристик.

```
def make_translation(original_text, features=None, example_text=None, max_input_length=128):
    text_tokenized = mt_tokenizer(
        [original_text],
        return_tensors='pt',
        max_length=max_input_length,
        truncation=True
    ).to('cpu')
    text_embedding = transform_text_embedding(
        original_text,
        features,
        example_text
    )
    generated_ids = mt_model.generate(
        topic_embedding=text_embedding,
        **text_tokenized
    )
    return mt_tokenizer.batch_decode(
        generated_ids,
        skip_special_tokens=True
    )[0]
```

Рисунок 5.10 – Функція контрольованого перекладу

У цій функції токенізуємо текст і одразу повертаємо вектори ідентифікаторів і масок у формі pytorch тензорів на CPU. Формуємо вектор уявлень про текст і модифікуємо його за потреби, використовуючи попередню функцію. Генеруємо ідентифікатори токенів перекладу і декодуємо їх.

Отже, на фінальному етапі експерименту було перевірено як саме запропонована модель здатна змінювати стиль і структуру перекладів та яким є обсяг цих змін. Наразі вона здатна змінювати переклад лише при модифікації окремих характеристик тексту. У режимі перекладу з трансфером характеристик із заданого тексту-прикладу результат залишається незмінним. Проте при комбінуванні вектору тексту із середніми векторами прикладів характеристик чи стилів можна суттєво змінити лексику, стиль чи структуру результату [45].

6 ПОБУДОВА ПРОГРАМНОЇ СИСТЕМИ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

За результатами дослідження було отримано модель для керованого машинного перекладу, визначено методику її використання. Створимо прототип інтелектуальної програмної системи для використання моделі.

Серверну частину прототипу реалізуємо з використанням мови програмування Python і фреймворку FastAPI. API має надавати токен сесії користувача, щоб запобігати неавторизованому доступу до моделі та її неконтрольованому використанню. API має реалізовувати створення профілю і зберігання історії перекладів користувача. Також мають бути закриті запити для експорту перекладів для подальшого тренування моделі.

На рисунку 6.1 можна побачити формат POST запиту на створення нового перекладу.

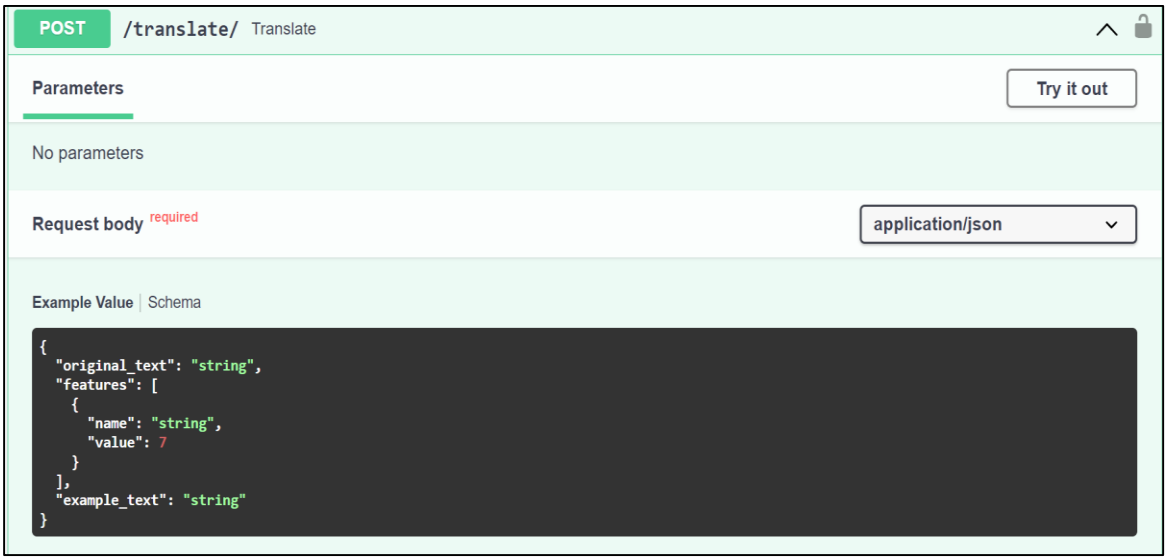

Рисунок 6.1 – POST запит на створення перекладу

Запит дозволяє виконувати простий переклад без жодного впливу на модель, виконувати трансфер стилю з окремих доменів або з тексту-прикладу. Код виконання самого трансферу вже було описано у розділі 5.5.

На рисунку 6.2 можна побачити GET запит на отримання історії перекладів з механізмом часткового експорту малими порціями.

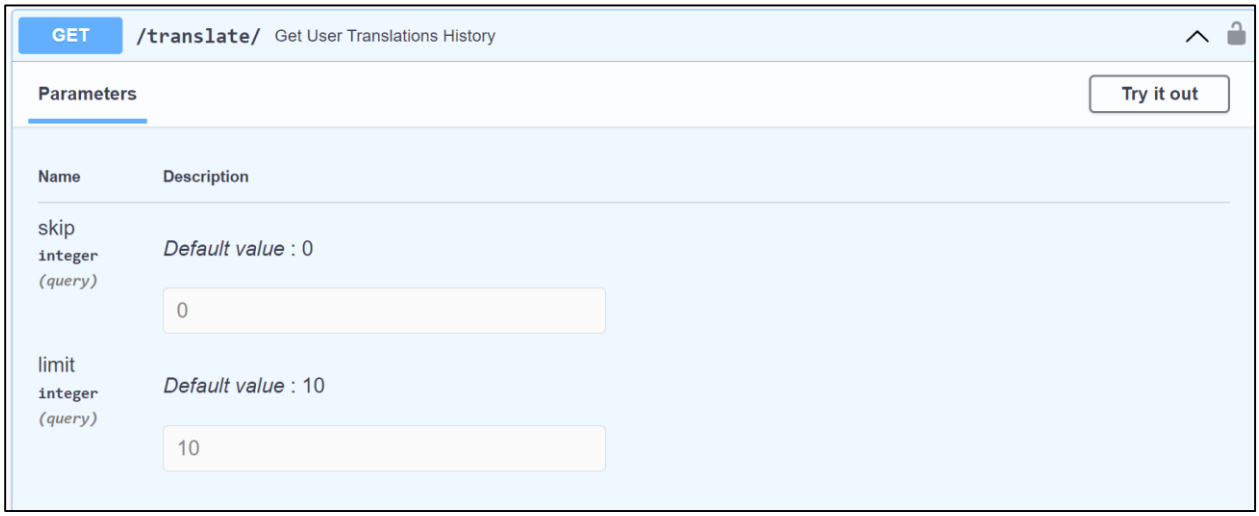

Рисунок 6.2 – Отримання історії перекладів користувача

Користувацька частина була реалізована за рахунок Vue JS і UIKit. Вона надає простий інтерфейс з формою для вхідного тексту і слайдерами для налаштування коефіцієнту трансформації для кожного окремого домену. При виконанні перекладу з'являється друга форма, яка показує переклад згенерований за заданими параметрами. Також повертається оцінка семантичної схожості оригінального тексту та його перекладу із заданими характеристиками. Оцінка виконується аналогічно розділу 5.5, тобто з використанням багатомовної мультимодальної моделі для оцінки наближеності двох текстів. Значення присвоюється від 0 до 1, адже значення менше за 0 (протилежні тексти) однаково погане для нас як і просто 0

(нейтральні тексти, які не мають спільного). На рисунку 6.3 показано форму для перекладів і результати.

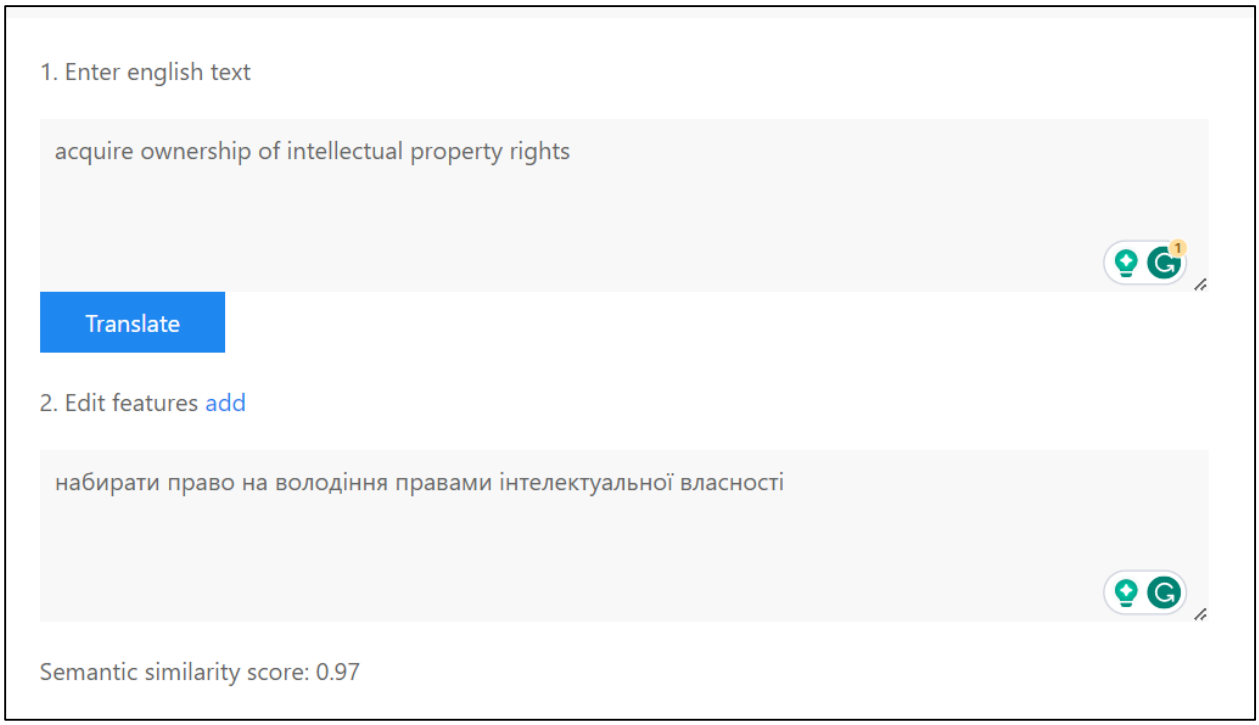

Рисунок 6.3 – Прототип для використання моделі керованого перекладу

Отже, прототип дозволяє використовувати запропоновану архітектуру в обох режимах: і за списком характеристик із коефіцієнтами трансформації (як через інтерфейс, так і через API), і за текстом-прикладом у one-shot learning режимі (тільки через API). Також прототип дає можливість оцінити переклад моделі, тож це має дозволити більш детально перевірити підхід заміру якості перекладу через модель для мультимодального семантичного пошуку, адже можна буде прослідкувати кореляцію оцінок моделі та користувачів прототипу.

ВИСНОВОК

Отже, у ході дослідження було вивчено та протестовано рішення для керованого машинного перекладу з англійської на українську на спеціалізованих корпусах, а також було запропоновано модифікацію архітектури кодувальників-декодувальників, яка отримує додатковий вектор контексту для трансферу стилю з певного домену на переклад вхідного тексту. Запропонована модель дає кращі результати не тестовому мультидоменному наборі даних як загалом, так і в окремих галузях, а також підвищує ступінь керованості за рахунок комбінування характеристик доменів.

Було досліджено наявні рішення та їх застосування для поставленої задачі. Визначено недоліки та головні проблеми. Досліджено метрики для машинного перекладу, визначено їх недоліки саме для задачі контрольованого перекладу, адже у ній часто відсутні еталони для порівняння.

За сформульованим планом експериментів було натреновано варіанти моделі MarianMT, що мали б дозволяти керувати її перекладами, на зібраних мультидоменних наборах даних: переклади законів України, переклади рефератів наукових статей, переклад описів зображень з набору Multi30k. За планом моделі натреновані для окремих доменів, модель натренована на всіх текстах, модель з токеном-маркером стилю і запропонована модифікація з векторами контексту мали бути порівняні на тестовому наборі, що складається з 25% від зібраних пар текстів за токен метриками BLEU і METEOR, а також за метрикою за уявленнями (embedding метрикою) BERT Score. Фінальним кроком стало тестування моделі на окремих текстах, що можуть мати кілька перекладів залежно від контексту. Для заміру якості цих варіацій перекладу було використано багатомовну мультимодальну версію XLM-R для

семантичного пошуку і косинусну схожість векторів-дескрипторів англомовного оригіналу і згенерованих текстів.

Тож, в результаті дослідження вдалося створити модифікацію моделей машинного перекладу, що здатна переносити характеристики кластерів текстів на вхідний зразок і було виконано кількісне порівняння рішення із існуючими варіантами. Проте не вдалося натренувати архітектуру достатньо, щоб виконувати траснфер стилю з єдиного зразка, адже це потребує більшої кількості даних і довшого тренування моделі.

Перспективним варіантом розвитку дослідження було б подальше тренування архітектури, щоб досягнути трансферу стилю з єдиного зразка. Таким чином архітектура надавала би 2 методи використання: як перенос стилю з конкретного тексту, так і модифікація окремих характеристик, яка доступна вже зараз. Також важливою є інтерпритація векторів-дескрипторів текстів, щоб надати більш гнучкий контроль над окремим характеристиками та краще розуміти вплив кожного значення вектору на кінцевий переклад.

Цікавою також є перспектива подальшого вдосконалення запропонованої модифікації з метою дозволити робити стилістичні модифікації лише на окремих токенах тексту, а не впливати одразу на весь текст загалом. Це дозволило б змінювати лише окремі невдалі чи невідповідні частини перекладу та підбирати інші варіанти не за частотою використання, як це часто робиться у наявних рішеннях, а за додатковим контекстом, що надається користувачем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A Brief History of Machine Translation Technlogy. K International. URL: https://www.k-international.com/blog/history-machine-translation/ (дата звернення: 29.04.2023).

2. The Pros and Cons of Google Translate | Language Connections. Language Connections. URL: https://www.languageconnections.com/blog/the-pros-cons-of-google-translate/ (дата звернення: 29.04.2023).

3. Дашенков Д., Смеляков К., Турута О. Methods of Multilanguage Question Answering. IEEE 8th International Conference on Problems of Infocommunications. 2021. № 2021 - Proceedings. С. 251–255. URL: https://doi.org/10.1109/PICST54195.2021.9772145 (дата звернення: 29.04.2023).

4. Пол Г. THE GEORGETOWN-IBM EXPERIMENT OF 1954: AN EVALUATION IN RETROSPECT. 1967. URL: https://aclanthology.org/www.mt-archive.info/Garvin-1967.pdf (дата звернення: 29.04.2023).

5. Торрегроса Д., Пасріча Н., Масуд М. Leveraging Rule-Based Machine Translation Knowledge for Under-Resourced Neural Machine Translation Models. MTSummit. 2019. Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks. С. 125–133. URL: https://aclanthology.org/W19-6725.pdf (дата звернення: 29.04.2023).

6. Ал Ансарі С. Interlingua-based Machine Translation Systems: UNL versus Other Interlinguas. The Egyptian Journal of Language Engineering. 2014. URL: https://doi.org/10.21608/ejle.2014.59863 (дата звернення: 29.04.2023).

7. Нагао М., Цуджі Ю., Мітамура К. A Machine Translation System From Japanese Into English - Another Perspective of MT Systems. COLING. 1980. COLING 1980 Volume 1: The 8th International Conference on Computational

Linguistics. URL: https://aclanthology.org/C80-1063.pdf (дата звернення: 29.04.2023).

8. Мат'яс Дж. An Overview of Statistical Machine Translation. 2015. URL: https://www.researchgate.net/publication/287196051_An_Overview_of_Statistical_Machine_Translation (дата звернення: 29.04.2023).

9. Колінз М. Statistical Machine Translation : IBM Models 1 and 2. 2011. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1162/handouts/Collins_annotated.pdf (дата звернення: 29.04.2023).

10. Шонеман Т. Computing Optimal Alignments for the IBM-3 Translation Model. CoNLL. 2010. Proceedings of the Fourteenth Conference on Computational Natural Language Learning. С. 98—106. URL: https://aclanthology.org/W10-2913 (дата звернення: 29.04.2023).

11. Коен Ф., Ох Ф., Марку Д. Statistical Phrase-Based Translation. NAACL. 2023. Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the As. С. 127—133. URL: https://aclanthology.org/N03-1017.pdf (дата звернення: 29.04.2023).

12. SYNTACTIC PHRASE-BASED STATISTICAL MACHINE TRANSLATION / H. Hassan та ін. 2006 IEEE Spoken Language Technology Workshop, м. Palm Beach, Aruba, 10—13 груд. 2006 р. 2006. URL: https://doi.org/10.1109/slt.2006.326799 (дата звернення: 29.04.2023).

13. Суцквевер І., Він'ялс О., Лі К. Sequence to Sequence Learning with Neural Networks. 2014. URL: https://arxiv.org/abs/1409.3215 (дата звернення: 29.04.2023).

14. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation / Й. Ву та ін. 2016. URL: https://arxiv.org/abs/1609.08144 (дата звернення: 29.04.2023).

15. Attention Is All You Need / A. Васвані та ін. 2017. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 29.04.2023).

16. Improving Language Understanding by Generative Pre-Training / А. Редфорд та ін. 2018. URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (дата звернення: 29.04.2023).

17. Девлін Дж., Ченг М.-В., Лі К. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 29.04.2023).

18. Unsupervised Cross-lingual Representation Learning at Scale / А. Конн'ю та ін. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, м. Online. Страудсбург, Пенсільванія, США, 2020. URL: https://doi.org/10.18653/v1/2020.acl-main.747 (дата звернення: 29.04.2023).

19. Раффел К., Шазір Н., Робертс А. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research. 2020. № 21. С. 1—67. URL: https://jmlr.org/papers/volume21/20-074/20-074.pdf (дата звернення: 29.04.2023).

20. Marian: Fast Neural Machine Translation in C++ / М. Юнкіс-Даунмут та ін. Proceedings of ACL 2018, System Demonstrations, м. Мельбурн, Австралія. Страудсбург, PA, США, 2018. URL: https://doi.org/10.18653/v1/p18-4020 (дата звернення: 29.04.2023).

21. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension / М. Л’юіс та ін. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, м. Online. Страудсбург, Пенсільванія, США, 2020. URL: https://doi.org/10.18653/v1/2020.acl-main.703 (дата звернення: 29.04.2023).

22. Гонг З., Джанг Й., Джоу Г. Statistical Machine Translation based on LDA. 2010 4th International Universal Communication Symposium (IUCS), м. Пекін, Китай, 18–19 жовт. 2010 р. 2010. URL: https://doi.org/10.1109/iucs.2010.5666182 (дата звернення: 29.04.2023).

23. Controlling Machine Translation for Multiple Attributes with Additive Interventions / A. Щіоппа та ін. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, м. Онлайн і Пунта Кана, Домініканська Республіка. Страудсбург, Пенсільванія, США, 2021. URL: https://doi.org/10.18653/v1/2021.emnlp-main.535 (дата звернення: 29.04.2023).

24. BLEU / K. Papineni та ін. the 40th Annual Meeting, м. Philadelphia, Pennsylvania, 7–12 лип. 2002 р. Морістаун, Нью-Йорк, США, 2001. URL: https://doi.org/10.3115/1073083.1073135 (дата звернення: 29.04.2023).

25. Банерджі С., Лаві А. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. WS. 2005. С. 65–72. URL: https://aclanthology.org/W05-0909 (дата звернення: 29.04.2023).

26. BERTScore: Evaluating Text Generation with BERT / T. Джанг та ін. 2019. URL: https://arxiv.org/abs/1904.09675 (дата звернення: 29.04.2023).

27. Multi-level Distillation of Semantic Knowledge for Pre-training Multilingual Language Model / M. Лі та ін. EMNLP. 2022. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. URL: https://aclanthology.org/2022.emnlp-main.202 (дата звернення: 29.04.2023).

28. COMET: A Neural Framework for MT Evaluation / P. Рей та ін. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 16 листоп. 2020 р. С. 2685–2702. URL: https://aclanthology.org/2020.emnlp-main.213.pdf (дата звернення: 29.04.2023).

29. Максименко Д., Сайчишина Н., Турута О. Controllability for English-Ukrainian Machine Translation Based on Specialized Corpora. Multi3Generation.

30. Чен В., Хадіві Ш., Торстен-Пітер Я. Guided Alignment Training for Topic-Aware Neural Machine Translation. 2016. URL: https://arxiv.org/abs/1607.01628 (дата звернення: 29.04.2023).

31. OPUS - an open source parallel corpus. OPUS - an open source parallel corpus. URL: https://opus.nlpl.eu/ (дата звернення: 29.04.2023).

32. Extension Multi30K: Multimodal Dataset for Integrated Vision and Language Research in Ukrainian / Н. Сайчишина та ін. UNLP-2023, м. Дубровнік. 2023. С. 54–61. URL: https://aclanthology.org/2023.unlp-1.7.pdf (дата звернення: 29.04.2023).

33. Improving the Machine Translation Model in Specific Domains for the Ukrainian Language / Д. Максименко та ін. EEE 17th International Conference on Computer Sciences and Information Technologies, м. Львів. 2022. С. 123–129. URL: https://ieeexplore.ieee.org/document/10000529 (дата звернення: 29.04.2023).

34. Neural Natural Language Generation: A Survey on Multilinguality, Multimodality, Controllability and Learning / О. Турута та ін. Journal of Artificial Intelligence Research. 2022. Т. 73. URL: https://dl.acm.org/doi/10.1613/jair.1.12918 (дата звернення: 29.04.2023).

35. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis / Є. Джіа та ін. 2018. URL: https://arxiv.org/abs/1806.04558 (дата звернення: 29.04.2023).

36. Елфвінг С., Учібе Е., Доа К. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. 2017. URL: https://arxiv.org/abs/1702.03118 (дата звернення: 29.04.2023).

37. Агарап А. Deep Learning using Rectified Linear Units (ReLU). 2019. URL: https://arxiv.org/abs/1803.08375 (дата звернення: 29.04.2023).

38. Маатен Л., Гінтон Дж. Visualizing Data using t-SNE. Journal of Machine Learning Research. 2008. Т. 9. URL: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf (дата звернення: 29.04.2023).

39. Мачк'євіч А., Ратайчжак В. Principal components analysis (PCA). Computers & Geosciences. 1993. Т. 19, № 3. С. 303–342. URL: https://www.sciencedirect.com/science/article/abs/pii/009830049390090R (дата звернення: 29.04.2023).

40. Unsupervised Cross-lingual Representation Learning at Scale / А. Конню та ін. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, м. Страудсбург, Пенсильванія, США, 2020. URL: https://doi.org/10.18653/v1/2020.acl-main.747 (дата звернення: 29.04.2023).

41. Панченко Д., Максименко Д., Турута О. Evaluation and Analysis of the NLP Model Zoo for Ukrainian Text Classification. Communications in Computer and Information Science. 2022. Т. 1698. URL: https://link.springer.com/chapter/10.1007/978-3-031-20834-8_6 (дата звернення: 29.04.2023).

42. Панченко Д., Турута О., Максименко Д. Ukrainian News Corpus as Text Classification Benchmark. Communications in Computer and Information Science. Т. 1635. URL: https://link.springer.com/chapter/10.1007/978-3-031-14841-5_37 (дата звернення: 29.04.2023).

43. Generalized Semantic Analysis Algorithm of Natural Language Texts for Various Functional Style Types / Н. Шаронова та ін. CEUR Workshop Proceedings : Міжнар. наук. конф., м. Гливиці, 12 трав. 2022 р. 2022. С. 16–26. URL: https://ceur-ws.org/Vol-3171/paper4.pdf (дата звернення: 29.04.2022).

44. Investigation of the deep learning approaches to classify emotions in texts / Д. Назаренко та ін. CEUR Workshop Proceeding : матеріали Міжнар. наук.

конф., м. Харків, 22 квіт. 2021 р. 2021. С. 206–224. URL: https://ceur-ws.org/Vol-2870/paper19.pdf (дата звернення: 29.04.2023).

45. Effectiveness of Preprocessing Algorithms for Natural Language Processing Applications / К. Смеляков та ін. 2020 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), м. Харків, 6 жовт. 2020 р. 2020. С. 1–5. URL: https://doi.org/10.1109/PICST51311.2020.9467919 (дата звернення: 29.04.2023).