Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІЖЕННЯ ВЕЛИКОМОВНОЇ МОДЕЛІ ДЛЯ ПЕРЕКЛАДУ УКРАЇНСЬКОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-2

Попов І.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Кузьомін О.Я.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Попову Іллі Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження великомовної моделі для перекладу української мови з використанням штучного інтелекту

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 грудня 2023 р.

3. Вихідні дані до роботи методи машинного перекладу, перелік використовуваних програмних засобів: теоретичні відомості про машинний переклад та обробку мови.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд основних методів машинного перекладу.
2. Огляд моделей нейронного машинного перекладу.
3. Порівняння великих мовних моделей.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Приклади та схеми роботи, та зображення архітектур мовних моделей, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	03.11.23-08.11.23	
3	Аналіз літератури з досліджуваної проблеми	08.11.23-20.11.23	
4	Аналіз технічних засобів	20.11.23-25.11.23	
5	Аналіз методів	25.11.23-01.12.23	
6	Програмна реалізація	01.12.23-05.12.23	
7	Оформлення пояснювальної записки	05.12.23-10.12.23	
8	Перевірка на плагіат	15.12.2023	
9	Рецензування	20.12.2023	
10	Підготовка презентації та доповіді	28.12.2023	
11	Занесення роботи в електронний архів	03.01.2024	
12	Попередній захист кваліфікаційної роботи	04.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кузьомін О.Я.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 48 с., 2 табл., 22 рис., 40 джерел.

МОВНІ МОДЕЛІ, НЕЙРОННИЙ МАШИННИЙ ПЕРЕКЛАД, УКРАЇНСЬКА МОВА, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єктом дослідження є можливості перекладу української мови з використанням різних видів нейронних мереж.

Метою дослідження є визначення найбільш ефективних методів та моделей, як з точки зору якості перекладу, так і ресурсів, що потрібні для тренування, налаштування, та використання моделі.

Проведено дослідження методів машинного перекладу. Проведено дослідження нейронних моделей машинного перекладу. Проведено тестування та порівняння великих мовних моделей на основі архітектури Transformer.

У результаті дослідження здійснене тестування та порівняння різних великих мовних моделей у задачу перекладу української мови у декількох мовних напрямах.

ARTIFICIAL INTELLIGENCE, LANGUAGE MODELS, NEURAL MACHINE TRANSLATION, UKRAINIAN LANGUAGE.

The object of the research is the possibilities of translating the Ukrainian language using different types of neural networks.

The purpose of the research is to determine the most effective methods and models, both in terms of translation quality and resources required for training, customization, and use of the model.

A research of machine translation methods was conducted. Research of models of neural machine translation was conducted. Testing and comparison of large language models based on the Transformer architecture was conducted.

As a result of the research, various large language models were tested and compared in the task of translating Ukrainian in several language directions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RBMT – Rules-Based Machine Translation (машинний переклад на основі правил)

НМП – нейронний машинний переклад

RNN – Recurrent Neural Networks (рекурентна нейронна мережа)

MoE – Mixture of Experts (суміш експертів)

FFN – Feedforward Neural Network (нейронна мережа прямого поширення)

ВСТУП

Останніми роками нейронний машинний переклад (НМП) виявився ефективнішим за фразові статистичні методи, завдяки чому швидко став найсучаснішим методом машинного перекладу. Однак системи НМП обмежені в перекладі мов з низьким рівнем ресурсів через значну кількість паралельних даних, необхідних для вивчення якісних відображень між мовами. Багатомовний НМП може допомогти вирішити проблеми, пов’язані з перекладом мов з обмеженими ресурсами. Переваги очевидні: відсутність розповсюдження помилок внаслідок використання вихідних даних однієї моделі як вхідних даних для іншої, зменшення накладних витрат і складності завдяки використанню однієї моделі для кількох мовних напрямків замість окремих моделей для кожного напрямку, покращення якості перекладу мовами з обмеженими ресурсами завдяки перенесенню знань зі споріднених мов, потенціал для перекладу з нуля для мовних напрямків, для яких немає прямих даних, тощо. Однак ці моделі мають і свої недоліки, зокрема вартість і складність перенавчання моделей, неможливість додавання додаткових мов без повного перенавчання моделі, а також майже повна неможливість точного налаштування.

Актуальність дослідження полягає у виявленні ефективної моделі для перекладу української мови. Перспектива розвитку може бути пов’язана із можливостями більш точного налаштування моделі, що може бути особливо корисно для мов з низьким рівнем ресурсів і рідкісних доменів, а також допоможе підвищити якість і ефективність перекладу.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ МАШИННОГО ПЕРЕКЛАДУ

1.1 Машинний переклад на основі правил

Системи машинного перекладу на основі правил (RBMT) були першими комерційними системами машинного перекладу і базуються на лінгвістичних правилах, які дозволяють ставити слова в різні місця і надавати їм різних значень залежно від контексту. Технологія RBMT застосовується до великих колекцій лінгвістичних правил на трьох різних етапах: аналіз, передача та генерація. Правила розробляються експертами з людської мови та програмістами, які докладають значних зусиль, щоб зрозуміти та зіставити правила між двома мовами. RBMT спирається на створені вручну перекладацькі лексикони, деякі з яких користувачі можуть редагувати та уточнювати для покращення перекладу.

RBMT дає змогу певною мірою контролювати лексику та уточнення користувацьких словників і є відносно передбачуваним у своїх результатах. Однак такі уточнення можуть забирати багато часу на впровадження та підтримку, а в деяких випадках можуть призвести до погіршення якості перекладу через неоднозначність термінів. Оскільки переклади виконуються за правилами, часто результат може мати більш «машинний» стиль написання, і хоча переклади можуть бути зрозумілими, вони часто не є вільно читабельними. Це часто називають якістю «суті», коли суть перекладу досить зрозуміла, але потрібно провести значну роботу з пост-редагування, щоб адаптувати результат перекладу до конкретної цільової аудиторії та стилю написання [1].

1.1.1 Перший метод

Перший метод складається з таких кроків:

Крок 1. Аналіз мови оригіналу на мову перекладу:

– семантичний аналіз;

– лексичний аналіз;

– виділення ключових слів;

– аналіз настроїв.

На цьому кроці не враховується розташування слів, тобто не розглядаються змістовні речення.

Крок 2. Отримані знання переносяться у модель.

Крок 3. Визначається, яке слово підійде для речення.

1.1.2 Другий метод

Другий метод складається з таких кроків:

Крок 1. Морфологічний аналіз вихідного тексту. Аналізується структура вихідної мови. Це подібно до Кроку 1 в першому методі. Аналіз базується на позиції слів.

Крок 2. Тегування слів на основі частин мови.

Крок 3. Переклад здійснюється на основі структури виявленої у першому кроці, та частин мови у Кроці 2.

1.1.3 Третій метод

Цей метод є комбінацією першого та другого методу. Позиція слів змінюється відповідно до речення, щоб отримати осмислене речення мовою перекладу (рис. 1.1 [2]).

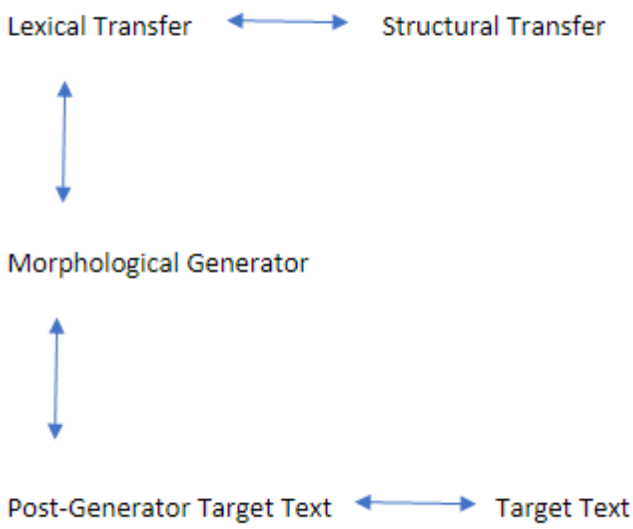

Рисунок 1.1 – Ілюстрація третього методу

Лексичний переклад (Lexical Transfer) – це процес зіставлення слів з мови оригіналу на мову перекладу. Він включає в себе вибір найбільш відповідного слова для тексту, який потрібно перекласти з мови оригіналу на мову перекладу.

Структурний переклад (Structural Transfer) – перекладає синтаксичні та структурні аспекти речення з мови оригіналу на мову перекладу. Зміст речення залишається незмінним, зберігаючи граматику.

Морфологічний генератор (Morphological Generator) – гарантує, що перекладені слова стоять у правильній формі (рід, число, час) відповідно до правил мови перекладу.

Цільовий текст після генератора (Post-Generator Target Text) – це кінцевий результат, тобто перекладені речення з вихідної мови.

1.1.4 Переваги та недоліки

Однією з головних переваг є те, що вона може обробляти складні речення за допомогою нейронних мереж і статистичних методів, які підвищили якість перекладу.

До мінусів моделі можна віднести складність обробки ідіоматичних і багатозначних виразів; розробка правил для складного синтаксису і граматики може бути складним завданням [2].

1.2 Статистичний машинний переклад

Статистичний машинний переклад – це підхід до машинного перекладу, який використовує великі обсяги двомовних даних для пошуку найвірогіднішого перекладу для заданих вхідних даних. Системи статистичного машинного перекладу навчаються перекладати, аналізуючи статистичні зв’язки між оригінальними текстами та їхніми перекладами, виконаними людиною.

Найважливішими компонентами статистичного машинного перекладу є модель перекладу та модель мови.

Модель мови будується на основі одномовних даних вихідної мови. Мовна модель знаходить найкращий варіант перекладу з кандидатів на основі мови перекладу. Модель може асоціюватися з вільним володінням мовою перекладу, оскільки вона надає перекладеному тексту його природний мовний потік.

Модель перекладу навчається на паралельних даних. Модель перекладу – це таблиця вирівняних фраз та їх переклад. Ці фрази називаються *n*-грамами. Мета моделі перекладу – передбачити можливі варіанти перекладу для конкретних вхідних текстів. Модель перекладу можна асоціювати з адекватністю, оскільки вона зберігає зміст оригіналу.

Процес складається з таких кроків:

Крок 1. Вхідний текст розбивається на фрази.

Крок 2. Фрази зіставляються з паралельними еквівалентами з моделі перекладу.

Крок 3. Мовна модель перевіряє ймовірність перекладу вихідною мовою.

Можливі підходи до статистичного перекладу включають:

– послівний переклад: модель генерує переклад слово за словом;

– фразовий переклад: модель перекладає послідовності слів;

– переклад на основі синтаксису: модель перекладає синтаксичні одиниці;

– ієрархічний фразовий переклад: модель поєднує методи на основі фраз із методами на основі синтаксису.

Основні проблеми, з якими стикається статистичний переклад:

– створення та навчання паралельних даних є дорогим і трудомістким процесом;

– статистичний машинний переклад вимагає великих паралельних даних, щонайменше 2 мільйони слів;

– специфічні помилки в перекладі важко передбачити та виправити;

– статистичний машинний переклад менш придатний для мовних пар із різним порядком слів, наприклад, для перекладу з англійської на китайську [3].

1.3 Гібридний переклад

Гібридний машинний переклад – це поєднання перекладу на основі правил та статистичного перекладу. Гібридний переклад використовує пам’ять перекладу, що робить його набагато ефективнішим з точки зору якості [4]. Деякі нещодавні роботи були зосереджені на гібридних підходах, які

поєднують підхід перенесення з одним із корпусних підходів. Це було зроблено для того, щоб працювати з меншою кількістю ресурсів і залежати від навчання та тренування правил перенесення. Основна ідея цього підходу полягає в автоматичному навчанні синтаксичних правил перенесення на основі обмеженої кількості вирівняних за словом даних. Ці дані містять всю необхідну інформацію для синтаксичного аналізу, перенесення та генерації речень [5].

1.4 Нейронний машинний переклад

Нейронний машинний переклад – це форма наскрізного навчання, яку можна використовувати для автоматичного виконання перекладів. У нейронному машинному перекладі нейронна мережа програми відповідає за кодування та декодування вихідного тексту, а не за виконання набору заздалегідь визначених правил із самого початку.

Таким чином, нейронний машинний переклад має потенціал для вирішення багатьох проблем традиційних фразових систем перекладу і, як було доведено, забезпечує кращу якість перекладу.

Нейромережеві моделі дуже відрізняються від фразових систем. Якщо останні розбивають вхідне речення на набір слів і словосполучень і зіставляють кожне з них зі словом або словосполученням у мові перекладу, то нейронні мережі враховують усе вхідне речення на кожному кроці при створенні вихідного речення [6].

1.4.1 Архітектура Seq2Seq

Архітектура навчання від послідовності до послідовності складається з кодера та декодера. Кодер і декодер, як правило, є рекурентними нейронними

мережами або більш сучасними варіантами, такими як довга короткочасна пам’ять (LSTM) або вентильний рекурентний вузол (GRU). Кодер перетворює вхідне речення на список векторів, по одному вектору на вхід. Маючи цей список, декодер виробляє один вихід за раз, поки не буде вироблено спеціальний токен кінця речення [7].

У цієї архітектури є два основних недоліки, обидва пов’язані з довжиною. По-перше, як і у випадку з людиною, ця архітектура має дуже обмежену пам’ять. Останній прихований стан LSTM – це те місце, куди ви намагаєтеся втиснути все речення, яке потрібно перекласти. Цей вектор зазвичай має довжину лише кілька сотень одиниць – чим більше ви намагаєтеся втиснути в цей вектор фіксованої розмірності, тим з більшими втратами змушена працювати нейронна мережа. Думати про нейронні мережі з точки зору «стиснення з втратами», яке вони повинні виконувати, іноді буває дуже корисно (рис. 1.2 [8]).

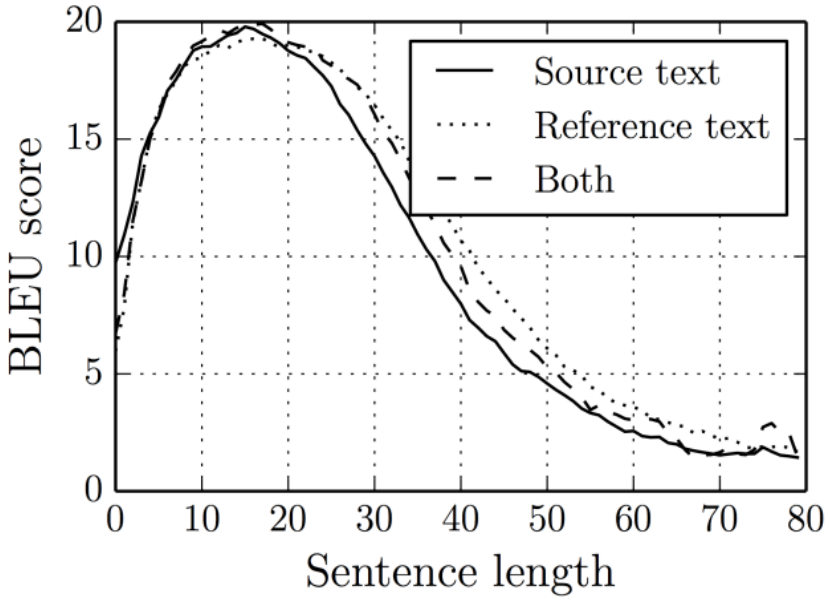

Рисунок 1.2 – Залежність якості перекладу від довжини речення

По-друге, як правило, чим глибша нейронна мережа, тим важче її навчати. Для рекурентних нейронних мереж, чим довша послідовність, тим глибша нейронна мережа в часовому вимірі. Це призводить до зникаючих

градієнтів, коли сигнал градієнта від об’єкта, на якому навчається рекурентна нейронна мережа, зникає, коли вона рухається в зворотному напрямку. Навіть для рекурентних нейронних мереж, спеціально створених для запобігання зникаючих градієнтів, таких як LSTM, це все ще залишається фундаментальною проблемою [8].

1.4.2 Архітектура Transformer

Моделі Transformer є найпопулярнішими та найпоширенішими в нейронному машинному перекладі. Архітектура Transformer, представлена в статті «Attention Is All You Need» у 2017 році [9], зробила значний стрибок у якості машинного перекладу. Відтоді вона домінує в галузі нейронного машинного перекладу і широко застосовується для різних завдань обробки природної мови, зокрема й для перекладу.

Моделі Transformer мають низку переваг, зокрема розпаралелювання обчислень, здатність обробляти далекі залежності та використання механізмів самоуваги для ефективного захоплення контекстної інформації. Такі моделі, як оригінальний Transformer, BERT, GPT та їхні різновиди, встановили нові стандарти якості перекладу.

Моделі Transformer також проклали шлях до багатомовного перекладу та перекладу з нуля, дозволивши навчити одну модель перекладати між кількома мовами і навіть мовами, з якими вона ніколи не стикалася під час навчання.

Хоча інші архітектури, такі як Seq2Seq моделі на основі RNN та згорткові моделі НМП, все ще використовуються в певних сценаріях, популярність та ефективність архітектури Transformer зробила її основним вибором для НМП в останні роки. Архітектуру Transformer зображено на рисунку 1.3 [10].

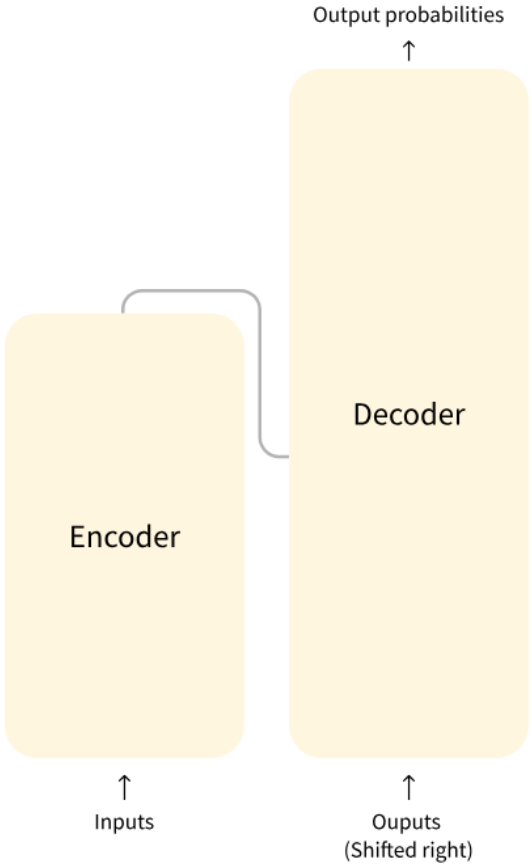

Рисунок 1.3 – Архітектура Transformer

У двох словах, завданням кодера, що знаходиться в лівій частині архітектури Transformer, є перетворення вхідної послідовності в послідовність безперервних відображень, яка потім подається на декодер.

Декодер, що знаходиться в правій частині архітектури, отримує вихід кодера разом з виходом декодера на попередньому часовому кроці для формування вихідної послідовності [10].

1.5 Постановка задачі дослідження

Об’єктом дослідження є можливості перекладу української мови з використанням різних видів нейронних мереж.

Метою дослідження є визначення найбільш ефективних методів та моделей, як з точки зору якості перекладу, так і ресурсів, що потрібні для тренування, налаштування, та використання моделі.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів та моделей нейронного машинного перекладу;

– провести порівняння ефективності різних методів та моделей, визначити різницю у якості перекладу та інших характеристиках, таких як швидкість тренування та перекладу, кількість необхідних даних для тренування;

– визначити, які моделі більш придатні для перекладу української мови.

2 МОДЕЛІ НЕЙРОННОГО МАШИННОГО ПЕРЕКЛАДУ

2.1 Рекурентні нейронні мережі для машинного перекладу

Рекурентні нейронні мережі призначені для того, щоб отримувати послідовності тексту на вхід або повертати послідовності тексту на вихід, або і те, і інше. Вони називаються рекурентними, тому що приховані шари мережі мають цикл, в якому вихід і стан комірки на кожному часовому кроці стають входами на наступному часовому кроці. Ця повторюваність слугує формою пам’яті. Вона дозволяє контекстній інформації проходити через мережу так, що відповідні виходи з попередніх часових кроків можуть бути застосовані до операцій мережі на поточному часовому кроці.

Це схоже на те, як ми читаємо. Коли ви читаєте текст, ви зберігаєте важливі фрагменти інформації з попередніх слів і речень і використовуєте їх як контекст для розуміння кожного нового слова і речення [11].

Базова модель рекурентної нейронної мережі для перекладу зображено на рисунку 2.1 [12]. Цей тип RNN – це RNN від послідовності до послідовності (seq2seq).

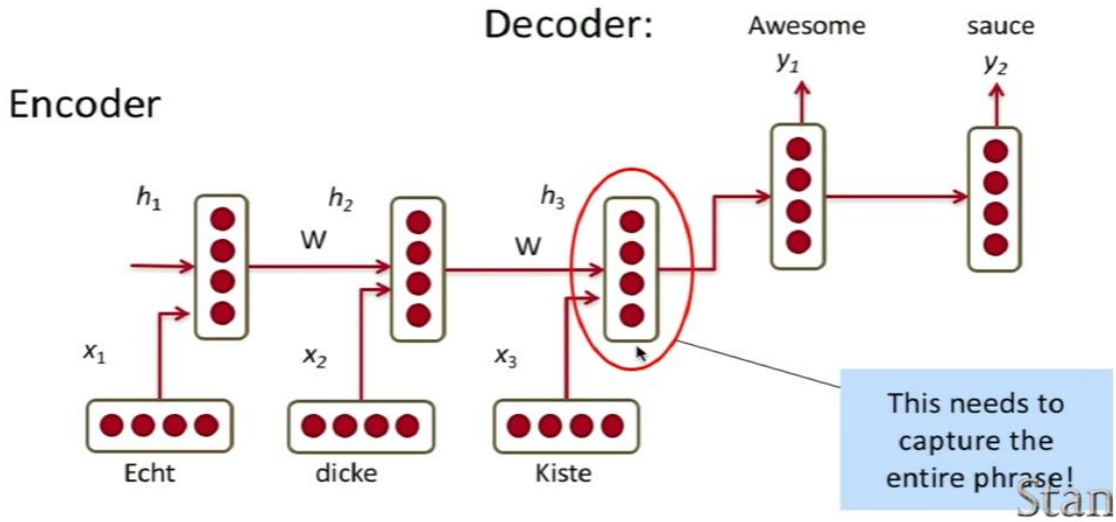

Рисунок 2.1 – seq2seq RNN

Модель починається з кодеру (encoder). Спочатку обчислюються вставки слів для вхідних векторів по одному гарячому слову. Потім надсилаються вбудовані слова для перекладу. Вбудований вектор слів множиться на деяку вагову матрицю $W^{(hx)}$. Попередній обчислений прихований стан (який є попереднім виходом вузла RNN) множиться на іншу вагову матрицю $W^{(hh)}$. Результати цих 2 множень додаються разом і застосовується нелінійність типу Relu/tanh. Тепер це наступний прихований стан h.

Цей процес повторюється для довжини нашого вхідного речення. Очевидно, що у першому вхідному слові x_0 немає попереднього прихованого стану, тому h_0 встановлюється рівним нулям.

Як дізнатися довжину вхідного речення? Речення може бути різної довжини, тому також потрібно мати стоп-токен (наприклад, крапку), який вказує на те, що був досягнутий кінець речення. Нам потрібно, щоб модель навчилася передбачати, коли потрібно виводити цей стоп-токен. Цей стоп-токен, по суті, є просто додатковим «словом» у навчальних даних.

Далі йде декодер. Як тільки досягається стоп-токен, йде перехід до вузла декодера RNN, щоб почати виробляти вихідні вектори.

Щоб отримати вихід у на кожному часовому кроці від декодера RNN, мається ще одна вагова матриця $W^{(S)}$, на яку множиться прихований стан h, щоб отримати векторний вихід. Потім до неї застосовується м’який максимум, який дає кінцевий результат. Цей кінцевий результат показує, який вектор слів було передбачено на цьому кроці.

$$h_i = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}). \tag{2.1}$$

$$\hat{y}_t = softmax(W^{(S)} h_t). \tag{2.2}$$

Ця модель дуже проста і на практиці може працювати лише для дуже простих речень (2 або 3 слова). Це пов’язано з тим, що ці базові RNN мають

проблеми з запам’ятовуванням більше, ніж на кілька кроків назад у минулому (проблема зникаючого градієнта).

Щоб покращити модель можна застосувати такі методи (рис. 2.2 [12]):

– тренувати різні ваги RNN для кодування та декодування. У наведеній вище моделі використовується один і той самий вузол RNN для кодування та декодування, що явно не є оптимальним для навчання;

– на етапі декодування замість того, щоб просто мати попередній прихований етап як вхідний, як показано вище, можна також включити останній прихований етап кодера (на рисунку – C). Разом з цим також додати останній передбачений вектор вихідного слова. Це має допомогти моделі зрозуміти, що вона щойно вивела певне слово і не виводитиме його знову.

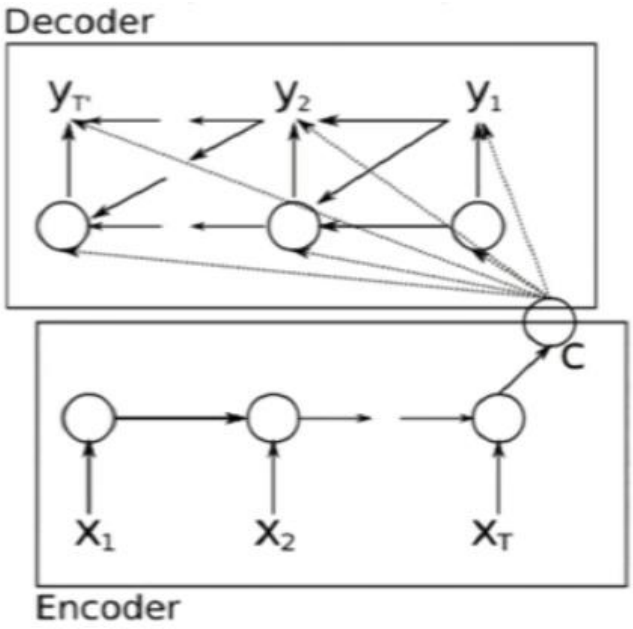

Рисунок 2.2 – Покращення моделі

Отже, у вузлі декодера тепер буде 3 вагові матриці (одна для попереднього прихованого стану *h*, одна для останнього передбаченого вектора слів *y* і одна для останнього прихованого стану кодера *C*), на які буде помножено відповідні вхідні дані, а потім додано їх, щоб отримати вихід декодера.

Інший спосіб перегляду всього процесу зображено на рисунку 2.3 [12].

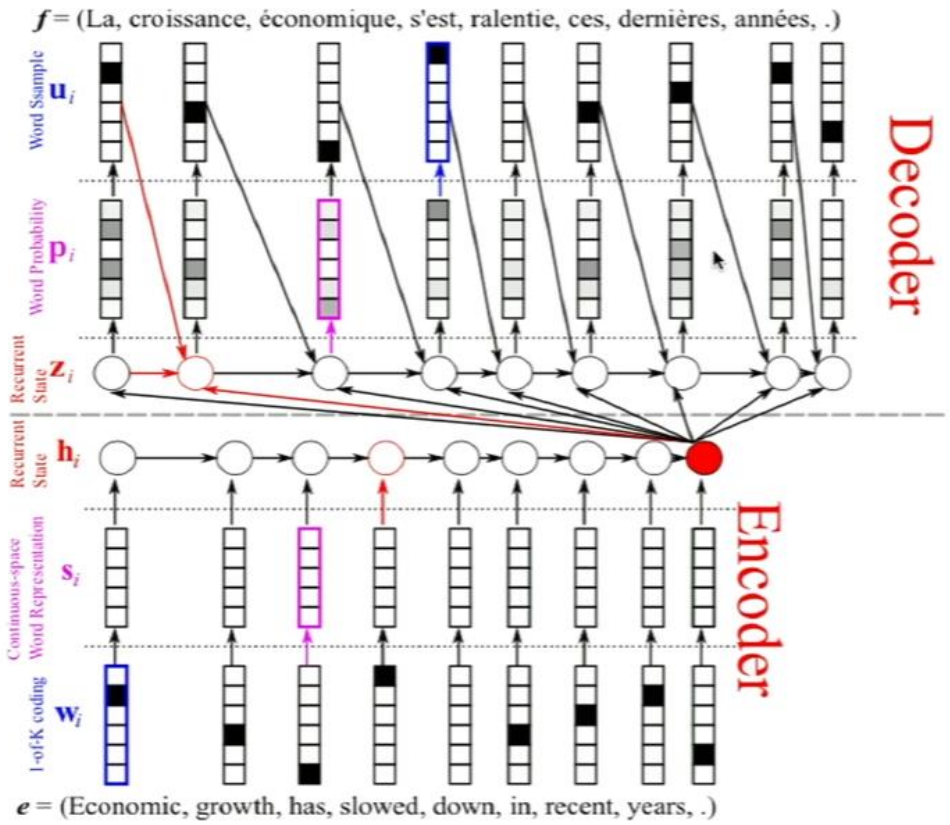

Рисунок 2.3 – Покращена модель

Важливо пам’ятати, що вагова матриця, яка використовується для множення входів на кожному кроці кодера, є абсолютно однаковою, вона не відрізняється для різних часових кроків.

На етапі декодування використовується інша вагова матриця, ніж на етапі кодування, але знову ж таки ця вагова матриця використовується на етапі декодування.

Інші можливі покращення включають:

– додавання більшої кількості шарів RNN до моделі;

– навчання двонаправленого кодеру, при цьому враховуючи прихований стан не тільки з останнього часового кроку прихованого шару, але й з наступного часового кроку прихованого шару;

– альтернативою навчанню двонаправленого кодера є навчання кодера у зворотному порядку, але це працює лише в тому випадку, якщо мови добре збігаються, наприклад, французька та англійська [12].

2.2 Моделі Transformer

Моделі Transformer використовуються для вирішення всіх видів завдань обробки мови: класифікація речень, класифікація кожного слова в реченні, генерація вмісту тексту, вилучення відповіді з тексту, генерація нового речення з вхідного тексту та переклад.

Архітектуру Transformer було представлено в червні 2017 року. Початкове дослідження було зосереджене на перекладацьких завданнях. Згодом було впроваджено кілька впливових моделей, зокрема:

– червень 2018 року: GPT [13], перша попередньо навчена модель Transformer, використана для точного налаштування на різних завданнях обробки природної мови і отримала найсучасніші результати;

– жовтень 2018 року: BERT [14], ще одна велика попередньо навчена модель, призначена для створення кращих резюме речень;

– лютий 2019: GPT-2 [15], покращена (і більша) версія GPT, яка не була одразу оприлюднена з етичних міркувань;

– жовтень 2019: DistilBERT [16], дистильована версія BERT, яка працює на 60% швидше, займає на 40% менше пам’яті та зберігає 97% продуктивності BERT;

– жовтень 2019: BART [17] і T5 [18], дві великі попередньо навчені моделі, що використовують ту саму архітектуру, що й оригінальна модель Transformer;

– травень 2020 року, GPT-3 [19], ще більша версія GPT-2, яка здатна добре виконувати різноманітні завдання без необхідності точного налаштування (так зване навчання з нуля).

Цей список далеко не вичерпний, і призначений лише для того, щоб висвітлити деякі з різних типів моделей трансформаторів. Загалом їх можна згрупувати в три категорії:

– GPT-подібні (авторегресійні моделі Transformer);

– BERT-подібні (Transformer моделі з автоматичним кодуванням);

– BART/T5-подібні (Transformer моделі від послідовності до послідовності).

Згадані вище моделі Transformer (GPT, BERT, BART, T5 тощо) були навчені як мовні моделі. Це означає, що вони були навчені на великих обсягах необробленого тексту в режимі самоконтролю. Самонавчання – це тип навчання, в якому мета автоматично обчислюється на основі вхідних даних моделі. Це означає, що людина не потрібна для маркування даних!

Цей тип моделі розвиває статистичне розуміння мови, на якій вона навчалася, але він не дуже корисний для конкретних практичних завдань. Через це загальна попередньо навчена модель проходить процес, який називається навчанням з переносом. Під час цього процесу модель налаштовується під наглядом, тобто за допомогою міток, що коментуються людиною, – на певне завдання.

Прикладом такого завдання є передбачення наступного слова в реченні, прочитавши n попередніх слів. Це називається каузальним мовним моделюванням, оскільки вихідні дані залежать від минулих і теперішніх входів, але не від майбутніх [20].

2.2.1 Архітектура Transformer

Більшість конкурентних моделей перетворення нейронних послідовностей мають структуру кодер-декодер. Кодер відображає вхідну послідовність символьних зображень $(x_1, \dots, x_n)$ у послідовність неперервних зображень $z = (z_1, \dots, z_n)$. За заданим z декодер генерує вихідну послідовність $(y_1, \dots, y_m)$ символів по одному елементу за раз. На кожному кроці модель є авторегресійною, використовуючи попередньо згенеровані символи як додаткові вхідні дані для генерації наступних.

Transformer слідує цій загальній архітектурі, використовуючи стекові самоконтролюючі та точкові, повністю з'єднані шари для кодера та декодера (рис. 2.4 [9]).

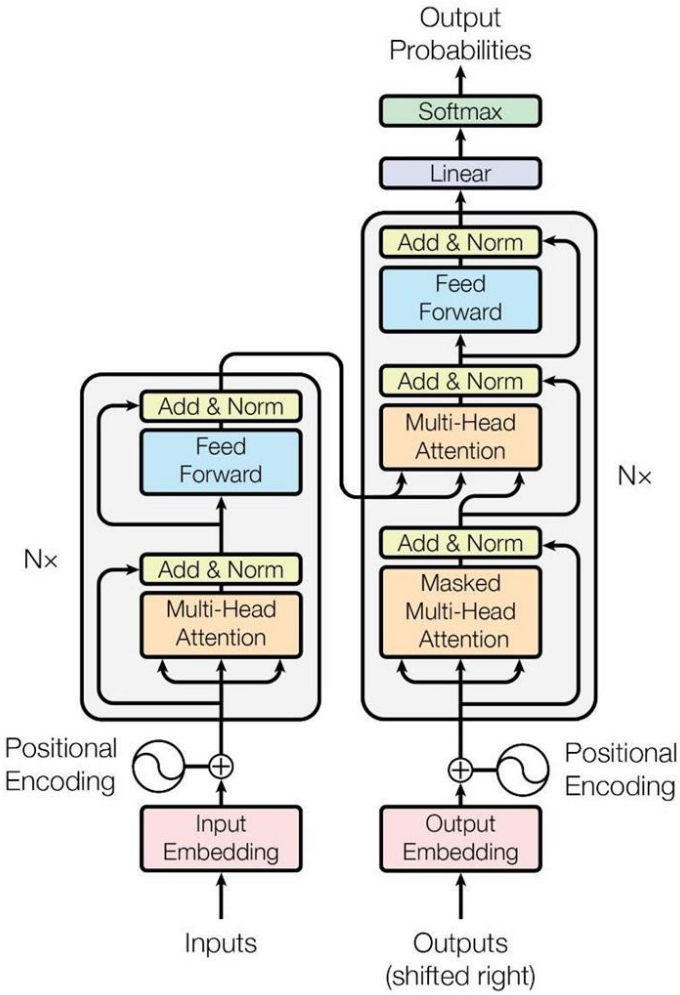

Рисунок 2.4 – Архітектура моделі Transformer

Кодер складається зі стеку з $N = 6$ однакових шарів. Кожен шар має два підшари. Перший – це механізм самоуваги з декількома головками, а другий – проста, позиційно повністю зв'язана мережа прямого поширення. Використовується залишковий зв'язок навколо кожного з двох підшарів, з подальшою нормалізацією шарів.

Декодер також складається зі стеку з $N = 6$ однакових шарів. На додаток до двох підшарів у кожному шарі кодера, декодер вставляє третій підшар, який виконує багатоголовочну увагу над виходом стеку кодера. Подібно до кодера,

використано залишкові зв’язки навколо кожного з підшарів, а потім нормалізовано шари. Також модифіковано підшар самоуваги в стеку декодерів, щоб запобігти впливу позицій на наступні позиції. Таке маскування у поєднанні з тим, що вихідні вставки зсунуті на одну позицію, гарантує, що передбачення для позиції і можуть залежати лише від відомих виходів у позиціях, менших за *i*.

Функцію уваги можна описати як відображення запиту і набору пар ключ-значення у вихідні дані, де запит, ключі, значення і вихідні дані є векторами. Результат обчислюється як зважена сума значень, де вага, присвоєна кожному значенню, обчислюється функцією сумісності запиту з відповідним ключем (рис. 2.5 [9]).

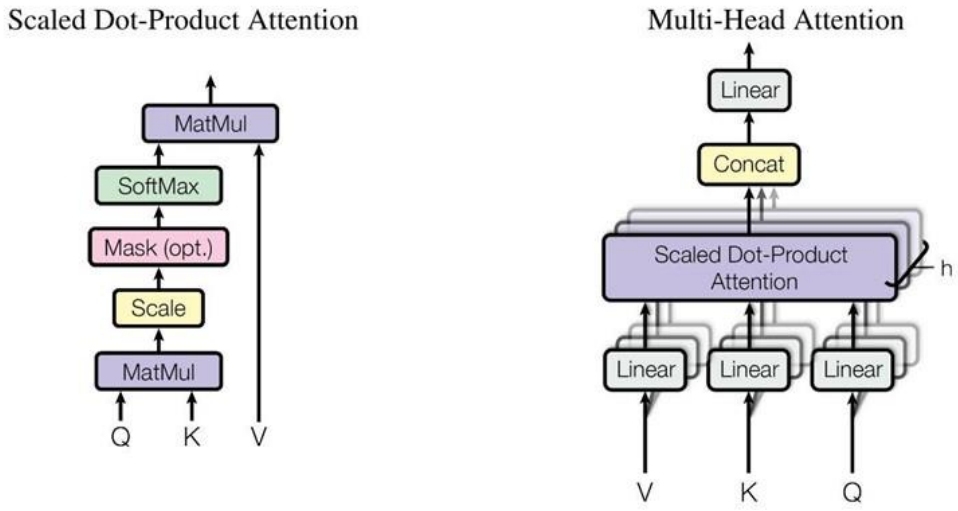

Рисунок 2.5 – Функція уваги

На додаток до підшарів уваги, кожен з шарів в кодері та декодері містить повністю підключену мережу прямого зв’язку, яка застосовується до кожної позиції окремо та ідентично. Вона складається з двох лінійних перетворень з активацією ReLU між ними.

$$\text{FFN}\,(x) = \max(0, xW_1 + b_1)W_2 + b_2. \qquad (2.3)$$

Хоча лінійні перетворення однакові в різних позиціях, вони використовують різні параметри від шару до шару. Інший спосіб описати це як дві згортки з розміром ядра 1. Розмірність входу та виходу $d_{\text{model}} = 512$, а внутрішній шар має розмірність $d_{ff} = 2048$.

Як і в інших моделях перетворення послідовностей, модель використовує вивчені вбудовування для перетворення вхідних та вихідних токенів у вектори розмірності d_{model}. Також використовуєтья звичайне вивчене лінійне перетворення і функція softmax для перетворення виходу декодера в передбачувані ймовірності наступних токенів. У моделі використовується однакова вагова матриця між двома шарами вбудовування та попереднє лінійне перетворення softmax. У шарах вбудовування ці ваги множаться на $\sqrt{d_{\text{model}}}$.

2.2.2 Кодерні моделі

Кодерні моделі використовують лише кодер моделі Transformer (рис. 2.6 [20]). На кожному етапі шари уваги мають доступ до всіх слів у початковому реченні. Ці моделі часто характеризуються як такі, що мають «двонаправлену» увагу, і їх часто називають моделями автокодування. До представників цього сімейства моделей відносяться BERT та DistilBERT.

Попереднє навчання цих моделей зазвичай зводиться до того, що задане речення певним чином спотворюється (наприклад, маскуванням випадкових слів), а модель отримує завдання знайти або реконструювати початкове речення.

Моделі кодерів найкраще підходять для завдань, що вимагають розуміння повного речення, таких як класифікація речень, розпізнавання іменованих об’єктів (і загалом класифікація слів), а також екстрактивні відповіді на запитання.

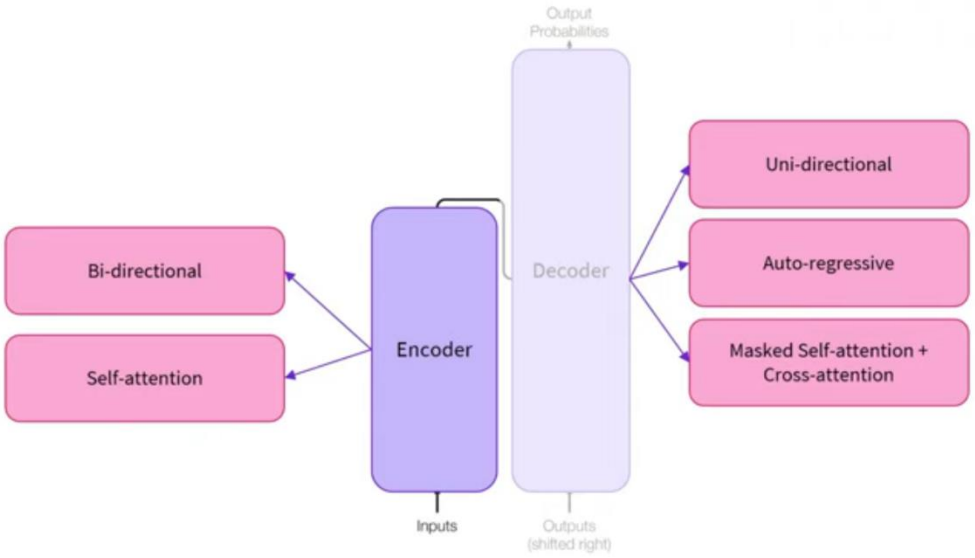

Рисунок 2.6 – Кодер у моделі Transformer

2.2.3 Декодерні моделі

Моделі з декодером використовують лише декодер моделі Transformer (рис. 2.7 [20]). На кожному етапі для певного слова шари уваги мають доступ лише до слів, розташованих перед ним у реченні. Такі моделі часто називають авторегресивними. До представників цього сімейства моделей відносяться GPT та GPT-2.

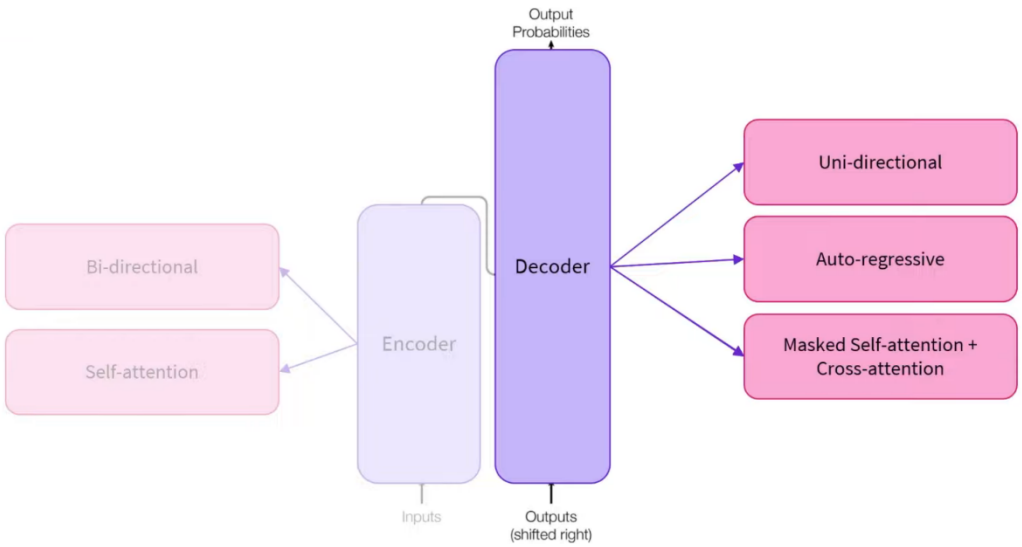

Рисунок 2.7 – Декодер у моделі Transformer

Попереднє навчання моделей-декодерів зазвичай обертається навколо передбачення наступного слова в реченні.

Ці моделі найкраще підходять для завдань, пов’язаних з генерацією тексту.

2.2.4 Моделі від послідовності до послідовності (seq2seq)

Моделі кодер-декодер (від послідовності до послідовності) використовують обидві частини архітектури Transformer (рис. 2.8 [20]). На кожному етапі шари уваги кодера мають доступ до всіх слів у початковому реченні, тоді як шари уваги декодера мають доступ лише до слів, розташованих перед даним словом у вхідних даних.

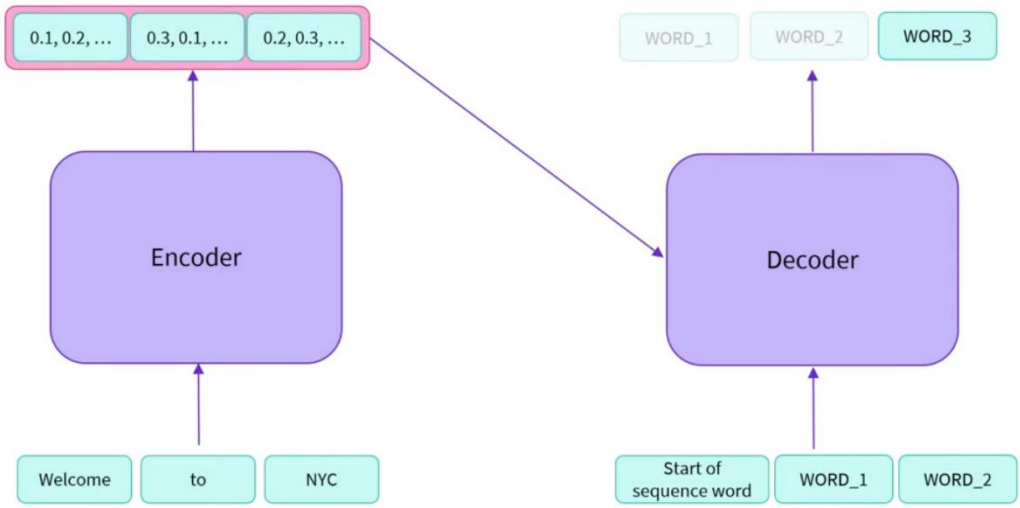

Рисунок 2.8 – Модель кодер-декодер

Попереднє навчання цих моделей можна виконати, використовуючи цілі моделей кодера або декодера, але зазвичай це дещо складніше завдання. Наприклад, T5 навчають, замінюючи випадкові фрагменти тексту (які можуть містити кілька слів) одним спеціальним словом-маскою, і завдання полягає в тому, щоб передбачити текст, який замінює це слово-маска.

Моделі від послідовності до послідовності найкраще підходять для завдань, що полягають у створенні нових речень залежно від заданих вхідних даних, таких як узагальнення, переклад або генерування відповідей на запитання.

3 ПОРІВНЯННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

3.1 Marian

Marian – це ефективний фреймворк нейронного машинного перекладу, написаний на чистій мові C++ з мінімальною кількістю залежностей. Наразі він використовується в багатьох європейських проектах і є основним перекладацьким і навчальним інструментом, що лежить в основі запуску нейронного машинного перекладу у Всесвітній організації інтелектуальної власності. У еволюціонуючій екосистемі інструментальних засобів нейронного мащинного перекладу з відкритим вихідним кодом Marian займає власну нішу, яка найкраще характеризується двома аспектами:

– він повністю написаний на C++11 і навмисно не містить прив’язок до Python; код моделі та мета-алгоритми призначені для реалізації в ефективному коді C++;

– він є самодостатнім з власним бек-ендом, який забезпечує автоматичну диференціацію в зворотному режимі на основі динамічних графів.

Marian має мінімальні залежності (лише Boostand CUDA або бібліотека BLAS) і забезпечує безбар’єрну оптимізацію на всіх рівнях: метаалгоритми, такі як багатовузлове навчання на основі MPI, ефективний пакетний пошук променів, компактні реалізації нових моделей, кастомні оператори та кастомні GPU-ядра. Intel зробила свій внесок у оптимізацію бекенду CPU і продовжує оптимізувати його.

Внутрішній інтерфейс глибокого навчання, що входить до складу Marian, базується на автодиференціації в зворотному режимі з динамічними обчислювальними графами і є однією з відомих платформ машинного навчання. Хоча бекенд можна використовувати для інших завдань, окрім машинного перекладу, він був оптимізований саме для цього та подібних випадків використання. Оптимізація на цьому рівні включає, наприклад,

ефективну реалізацію різних об’єднаних комірок RNN, механізмів уваги або оператора нормалізації атомних шарів.

На додаток до механізму автоматичного диференціювання та фреймворку кодера-декодера, було реалізовано багато ефективних метаалгоритмів. До них відносяться навчання на декількох пристроях (GPU або CPU), скоринг і пакетний пошук променів, об’єднання різнорідних моделей (наприклад, моделей DeepRNN і трансформаторних або мовних моделей), багатовузлове навчання і багато іншого [21].

3.2 M2M-100

Модель M2M-100 було запропоновано у 2021 році у статті «Beyond English-Centric Multilingual Machine Translation». До цієї моделі переклади між двома мовами залежали від англійської мови. Однак модель m2m-100 перекладає дані з однієї мови на іншу без використання англійської як посередника. Це робить її справжньою моделлю багатомовного перекладу «Багато до багатьох», яка може перекладати безпосередньо між будь-якою парою зі 100 мов [22].

Модель M2M-100 базується на архітектурі Transformer, яка складається з двох модулів: кодера та декодера. Кодер перетворює вихідну послідовність токенів у послідовність вставок однакової довжини. Потім декодер послідовно створює цільове речення, токен за токеном, або авторегресивно. Точніше, кодер бере послідовність токенів $W = (w_1, \dots, w_S)$ і вихідну мову ℓ_s, і виробляє послідовність вставок $H = (h_1, \dots, h_S)$, які потім подаються на декодер з цільовою мовою ℓ_t, щоб послідовно виробити послідовність цільових токенів $V = (v_1, \dots, v_T)$:

$$H = \text{encoder}\,(W, \ell_s). \tag{3.1}$$

$$\forall i \in [1, \dots, T], v_{i+1} = \text{decoder}\,(H, \ell_t, v_1, \dots, v_i). \quad (3.2)$$

Кодер і декодер складаються з однотипних шарів, які називаються трансформаторними шарами. Кожен шар-трансформатор приймає на вхід послідовність векторів і виводить послідовність векторів. У кодері шари-трансформатори складаються з двох підшарів, шару самоуваги та шару зворотного зв'язку. Вони накладаються послідовно, після чого відбувається залишкове з'єднання та нормалізація шарів:

$$Z = \text{norm}\,(X + \text{ self-attention}\,(X)). \quad (3.3)$$

$$Y = \text{norm}\big(Z + \text{ feed-forward}\,(Z)\big). \quad (3.4)$$

Шар самоуваги – це шар уваги, який оновлює кожен елемент послідовності, дивлячись на інші елементи, в той час як шар зворотного зв'язку пропускає кожен елемент послідовності незалежно через 2-шаровий персептрон. У декодері є додатковий третій підрівень, між самоувагою і прямою передачею, який обчислює увагу на виході кодера.

Архітектура Transformer була розроблена для двомовного випадку, коли цільова мова є фіксованою. У випадку багатомовного машинного перекладу цільова мова не є фіксованою, і можна застосувати кілька стратегій, щоб змусити мережу створювати речення потрібною цільовою мовою, в кодер було додано спеціальний токен, що позначає мову оригіналу, і спеціальний токен в декодер, що позначає мову перекладу.

Відправною точкою для вдосконалення моделей масового багатомовного перекладу є велика модель трансформатора з 12 кодувальними та 12 декодувальними шарами, розміром 8192 шарів і розмірністю вбудовування 1024. Загальна кількість параметрів становить 1,2 мільярди. Навчання виконується за допомогою оптимізатора Adam. Спочатку проводиться прогрів для 4000 оновлень, зі згладжуванням міток 0,1. Для

регуляризації було використано параметр відсіву між {0,1;0,2;0,3}. Для стабілізації навчання більш глибоких трансформерів було використано LayerDrop 0,05 і попередню нормалізацію.

Для навчання на мільярдах речень навчальні дані було розділено на 256 різних частин, щоб керувати споживанням пам'яті. Однак безпосередній поділ мов із середніми та низькими ресурсами на частини зменшив би варіативність даних кожної частини для мов із середніми та низькими ресурсами. Уявіть собі випадок, коли на шард припадає лише 100 речень певного мовного напрямку – модель легко перевантажиться. Таким чином, кожна мова поділяється на різну кількість частин залежно від рівня ресурсу, так що мови з високим рівнем ресурсу мають більше частин, а мови з низьким рівнем ресурсу мають лише одну частину. Згодом частини з меншими ресурсами реплікуються, поки не буде досягнуто повної кількості частин [23].

3.3 NLLB (No Language Left Behind)

Модель послідовного багатомовного машинного перекладу NLLB базується на архітектурі кодера-декодера Transformer. NLLB – це модель умовних обчислень на основі суміші експертів (група мереж-експертів або Mixture-of-Experts), яка навчається на даних, отриманих за допомогою нових та ефективних методів інтелектуального аналізу даних, пристосованих для мов з низьким рівнем ресурсів. Було запропоновано численні архітектурні та навчальні вдосконалення, щоб протидіяти надмірному пристосуванню під час навчання на тисячах завдань. Ця модель досягає покращення на 44% BLEU порівняно з попередніми моделями, що закладає важливий фундамент для реалізації універсальної системи перекладу.

Ця масштабна багатомовна модель перекладу навчається одразу за кількома напрямками перекладу, використовуючи ті самі спільні можливості моделі. Це може призвести до корисного міжмовного перекладу між

спорідненими мовами з ризиком збільшення інтерференції між неспорідненими мовами. Моделі з розрідженою сумішшю експертів (MoE) – це тип моделей умовних обчислень, які активують підмножину параметрів моделі на кожен вхід, на відміну від щільних моделей, які активують всі параметри моделі на кожен вхід. Це дозволяє досягти більш оптимального компромісу між міжмовним перенесенням та інтерференцією і покращення продуктивності для мов з обмеженими ресурсами.

Як показано на рисунку 3.1 [24], підшар FFN у щільних моделях було замінено на підшар MoE через кожні f_{MoE} шарів як у кодері, так і в декодері.

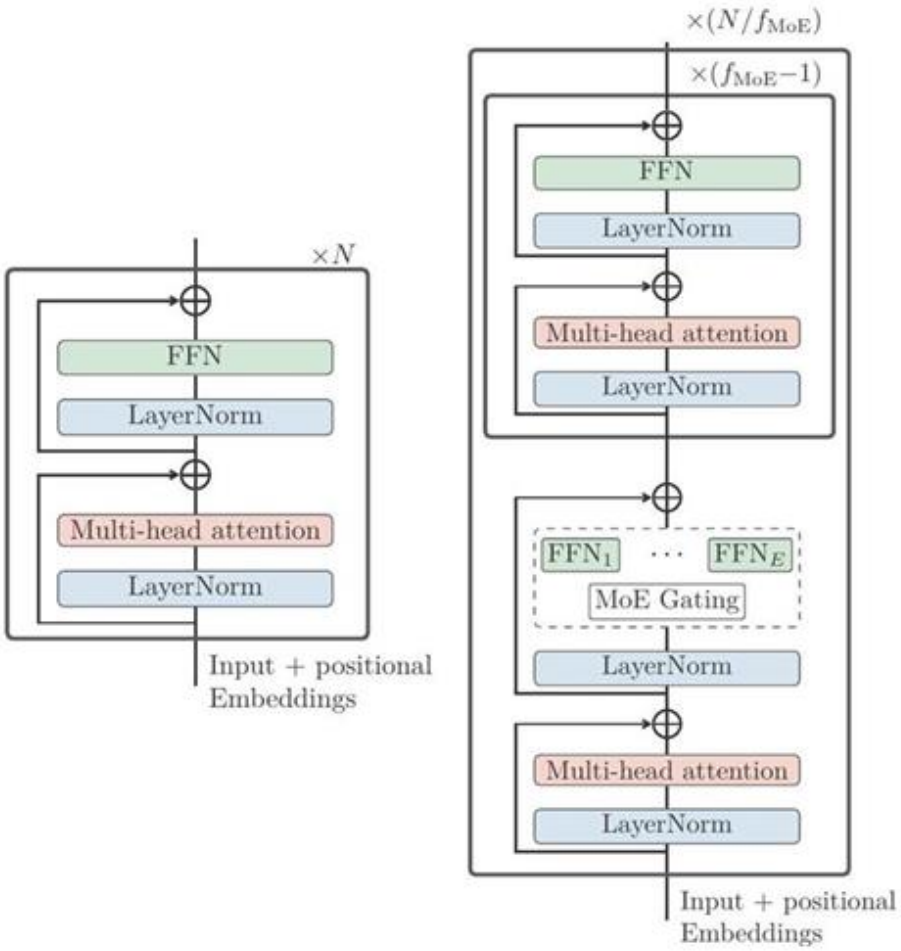

Рисунок 3.1 – Ілюстрація трансформаторного кодера з шарами MoE

Підшар MoE складається з E мереж прямого поширення (FFN), позначених через $(\mathrm{FFN}_1, \mathrm{FFN}_2, \ldots, \mathrm{FFN}_E)$, кожна з яких має вхідні та вихідні

проекції $W_i^{(e)}$ та $W_o^{(e)}$. До кожного підшару приєднано мережу, що складається з лінійного шару з нормалізацією за критерієм softmax та вагами W_g, яка вирішує, як пересилати токени експертам. За вхідним токеном x_t вихід підшару MoE оцінюється як:

$$\mathrm{FFN}_e\,(x_t) = W_o^{(e)}\mathrm{ReLU}\,\left(W_i^{(e)} \cdot x_t\right),\ (\forall e \in \{1, \dots, E\}), \tag{3.5}$$

$$G_t = \mathrm{softmax}\big(W_g \cdot x_t\big)\,,\ \mathcal{G}_t = \mathrm{Top} - \mathrm{k} - \mathrm{Gating}(G_t), \tag{3.6}$$

$$\mathrm{MoE}(x_t) = \sum_{e=1}^{E}\ \mathcal{G}_{te} \cdot \mathrm{FFN}_e(x_t), \tag{3.7}$$

де $\mathcal{G}_t \in \mathbb{R}^E$ – вектор маршрутизації, обчислений мережею, тобто для кожного експерта $\mathcal{G}_{t,e}$ – внесок експерта e (FFN_e) у вихід. Модель слідує алгоритму Top-k-Gating і відправляє токен не більше ніж $k = 2$ експертам. Завжди обирається 2 найкращих експерти з найбільшою кількістю балів для кожного токена.

Модель кодера-декодера Transformer, доповнена шарами MoE та їхніми відповідними мережами вентилів, навчається направляти вхідні токени до відповідних 2 найкращих експертів, оптимізуючи лінійно зважену комбінацію перехресної ентропії, згладженої мітками, та допоміжних втрат для балансування навантаження. Ця додаткова втрата підштовхує токени до рівномірного розподілу між експертами і оцінюється як:

$$LB = E \cdot \sum_{e=1}^{E}\ f_e p_e,\ p_e = \frac{1}{T} \cdot \sum_{t=1}^{T}\ G_{te}, \tag{3.8}$$

де f_e – частка токенів, що потрапляють до експерта *e*, як їх перший вибір, через Top-k-Gating;

p_e – середня ймовірність потрапляння до цього експерта серед T токенів у міні-батчі [24].

3.4 Тестування моделей

Для порівняння було взято моделі Helsinki-NLP/OPUS-MT [25], дві версії моделі M2M-100 (400 мільйонів параметрів та 1,2 мільярди параметрів), та модель nllb-200-distilled-600M. Моделі було взято з бібліотеки transformers від HuggingFace [26]. Спочатку було імпортовано потрібні бібліотеки (рис. 3.2).

```
from transformers import M2M100ForConditionalGeneration, M2M100Tokenizer, MarianMTModel, MarianTokenizer, AutoModelForSeq2SeqLM, AutoTokenizer
from sacrebleu import corpus_bleu
from datasets import load_dataset
from tqdm import tqdm
import numpy as np
import torch
```

Рисунок 3.2 – Імпорт бібліотек

Далі виконано завантаження та підготовку моделей для використання, для кожної моделі було завантажено кодер та модель (рис. 3.3).

```
device = "cuda:0" if torch.cuda.is_available() else "cpu"

marian_models = {
    "en-uk": MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-en-uk").to(device),
    "uk-en": MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-uk-en").to(device),
    "uk-pl": MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-uk-pl").to(device),
    "pl-uk": MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-pl-uk").to(device),
    "uk-hu": MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-uk-hu").to(device),
    "hu-uk": MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-hu-uk").to(device)
}

marian_tokenizers = {
    "en-uk": MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-en-uk"),
    "uk-en": MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-uk-en"),
    "uk-pl": MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-uk-pl"),
    "pl-uk": MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-pl-uk"),
    "uk-hu": MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-uk-hu"),
    "hu-uk": MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-hu-uk"),
}

m2m_model = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_418M").to(device)
m2m_tokenizer = M2M100Tokenizer.from_pretrained("facebook/m2m100_418M")

m2m_model_1b = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_1.2B").to(device)
m2m_tokenizer_1b = M2M100Tokenizer.from_pretrained("facebook/m2m100_1.2B")

nllb_model = AutoModelForSeq2SeqLM.from_pretrained("facebook/nllb-200-distilled-600M").to(device)
nllb_tokenizers = {
    "uk": AutoTokenizer.from_pretrained("facebook/nllb-200-distilled-600M", src_lang="ukr_Cyrl"),
    "en": AutoTokenizer.from_pretrained("facebook/nllb-200-distilled-600M", src_lang="eng_Latn"),
    "pl": AutoTokenizer.from_pretrained("facebook/nllb-200-distilled-600M", src_lang="pol_Latn"),
    "hu": AutoTokenizer.from_pretrained("facebook/nllb-200-distilled-600M", src_lang="hun_Latn")
}
```

Рисунок 3.3 – Підготовка моделей

Для оцінки моделей було взято датасет Flores 200, який включає набір з більше ніж 2000 речень на 200 мовах, з інформацією про теми та домени речень. Датасети було взято з бібліотеки datasets від HuggingFace [27] (рис. 3.4).

```
lang_map = {
    "uk": "ukr_Cyrl",
    "en": "eng_Latn",
    "pl": "pol_Latn",
    "hu": "hun_Latn"
}

sentences = {
    "uk": load_dataset("Muennighoff/flores200", lang_map["uk"]),
    "en": load_dataset("Muennighoff/flores200", lang_map["en"]),
    "pl": load_dataset("Muennighoff/flores200", lang_map["pl"]),
    "hu": load_dataset("Muennighoff/flores200", lang_map["hu"])
}
```

```
sentences['uk']['devtest'][0]['sentence']
```

```
'"Зараз у нас є 4-місячні миші, в яких немає діабету і які мали діабет раніше," – додав він.'
```

Рисунок 3.4 – Завантаження датасетів

Підготовано функції для перекладу речень використовуючи відповідні кодери та моделі (рис. 3.5).

```
def translate_m2m(model, tokenizer, input, input_lang, output_lang):
    tokenizer.src_lang = input_lang
    tokenized = tokenizer(input, return_tensors="pt", max_length=512, truncation=True).to(device)
    translation = model.generate(**tokenized, forced_bos_token_id=tokenizer.get_lang_id(output_lang))
    return tokenizer.batch_decode(translation, skip_special_tokens=True)[0]

def translate_marian(model, tokenizer, input):
    tokenized = tokenizer.encode(input, return_tensors="pt", max_length=512, truncation=True).to(device)
    translation = model.generate(tokenized, max_length=150, num_beams=5, early_stopping=True)
    return tokenizer.decode(translation[0], skip_special_tokens=True)

def translate_nllb(model, tokenizer, input, output_lang):
    tokenized = tokenizer(input, return_tensors="pt", max_length=512, truncation=True).to(device)
    translation = model.generate(
        **tokenized,
        forced_bos_token_id=tokenizer.lang_code_to_id[lang_map[output_lang]],
        max_length=512)
    return tokenizer.batch_decode(translation, skip_special_tokens=True)[0]
```

Рисунок 3.5 – Функції для перекладу

Моделі було використано для перекладу речень у декількох напрямках:

– Українська – Англійська;
– Англійська – Українська;
– Українська – Польська;
– Польська – Українська;
– Українська – Угорська;
– Угорська – Українська.

Оригінальні речення були збережені для оцінки моделей (рис. 3.6).

```
for idx, sentence in tqdm(enumerate(sentences['uk']['devtest'])):
    input_sentences = {
        "uk": sentence['sentence'],
        "en": sentences['en']['devtest'][idx]['sentence'],
        "pl": sentences['pl']['devtest'][idx]['sentence'],
        "hu": sentences['hu']['devtest'][idx]['sentence']
    }

    for langs in marian_models.keys():
        src_lang, target_lang = langs.split('-')
        input_sentence = input_sentences[src_lang]
        hypotheses_marian[langs].append(
            translate_marian(marian_models[langs], marian_tokenizers[langs], input_sentence))
        hypotheses_m2m[langs].append(
            translate_m2m(m2m_model, m2m_tokenizer, input_sentence, src_lang, target_lang))
        hypotheses_m2m_1b[langs].append(
            translate_m2m(m2m_model_1b, m2m_tokenizer_1b, input_sentence, src_lang, target_lang))
        hypotheses_nllb[langs].append(
            translate_nllb(nllb_model, nllb_tokenizers[src_lang], input_sentence, target_lang))

    references['uk'].append(input_sentences['uk'])
    references['en'].append(input_sentences['en'])
    references['pl'].append(input_sentences['pl'])
    references['hu'].append(input_sentences['hu'])
```

Рисунок 3.6 – Виконання перекладу та збереження результату

Для оцінки якості перекладу біло розраховано метрику BLEU для кожної моделі. Для оцінки було використано бібліотеку sacreBLEU (рис. 3.7) [28].

```python
for langs in hypotheses_m2m.keys():
    bleus_marian[langs] = corpus_bleu(hypotheses_marian[langs], [references[langs.split('-')[1]]], lowercase=True)
    bleus_m2m[langs] = corpus_bleu(hypotheses_m2m[langs], [references[langs.split('-')[1]]], lowercase=True)
    bleus_m2m_1b[langs] = corpus_bleu(hypotheses_m2m_1b[langs], [references[langs.split('-')[1]]], lowercase=True)
    bleus_nllb[langs] = corpus_bleu(hypotheses_nllb[langs], [references[langs.split('-')[1]]], lowercase=True)

for langs in bleus_marian.keys():
    print(f"BLEU Score for Marian model {langs}: {bleus_marian[langs]}")

for langs in bleus_m2m.keys():
    print(f"BLEU Score for M2M 418M model {langs}: {bleus_m2m[langs]}")

for langs in bleus_m2m_1b.keys():
    print(f"BLEU Score for M2M 1.2B model {langs}: {bleus_m2m_1b[langs]}")

for langs in bleus_nllb.keys():
    print(f"BLEU Score for NLLB model {langs}: {bleus_nllb[langs]}")
```

Рисунок 3.7 – Оцінка моделей за метрикою BLEU

Результати оцінки можна побачити у таблиці 3.1.

Таблиця 3.1 – Результати оцінки моделей за метрикою BLEU

Мови перекладу	**Helsinki-NLP/OPUS-MT**	**M2M-100 400M**	**M2M-100 1,2B**	**NLLB 600M**
Українська – Англійська	29,58	28,86	36,42	37,95
Англійська – Українська	20,42	22,51	27,38	23,05
Українська – Польська	12,85	14,49	17,79	14,00
Польська – Українська	12,50	15,00	17,63	14,30
Українська – Угорська	13,67	15,17	19,91	13,82
Угорська – Українська	11,77	15,41	18,68	13,87

Як бачимо, модель M2M-100 на 1,2 мільярди параметрів показує кращі результати майже на всіх напрямках, але варто мати на увазі, що ця модель також вимагає більше пам’яті та працює значно повільніше. Для оцінки швидкості роботи моделей було виміряно час, потрібний кожній з моделей для перекладу 100 речень з тестового набору (рис. 3.8).

```
for idx, sentence in tqdm(enumerate(sentences['uk']['devtest']['sentence'][:100])):
    input_sentences = {
        "uk": sentence,
        "en": sentences['en']['devtest'][idx]['sentence'],
        "pl": sentences['pl']['devtest'][idx]['sentence'],
        "hu": sentences['hu']['devtest'][idx]['sentence']
    }

    for langs in marian_models.keys():
        src_lang, target_lang = langs.split('-')
        input_sentence = input_sentences[src_lang]
        translate_marian(marian_models[langs], marian_tokenizers[langs], input_sentence)
```

```
100it [02:42,  1.63s/it]
```

Рисунок 3.8 – Тестування моделі на 100 реченнях

Результати вимірювань можна побачити у таблиці 3.2.

Таблиця 3.2 – Результати оцінки моделей за метрикою BLEU

Модель	**Час перекладу 100 речень**
Helsinki-NLP/OPUS-MT	2хв 42с
M2M-100 400M	6хв 19с
M2M-100 1,2B	10хв 54с
NLLB 600M	5хв 28с

Також було виміряно об’єм пам’яті, що використовується моделями при перекладі. Між вимірами пам’ять та кеш очищались з використанням функцій torch.cuda.empty_cache() та torch.cuda.reset_peak_memory_stats() з бібліотеки pytorch [29] та функцією gc.collect() що запускає збір сміття (рис. 3.9). Вимір здійснювався після перекладу тестового речення (рис. 3.10). Для вимірювання використаного об’єму пам’яті було використано функцію torch.cuda.max_memory_allocated() з бібліотеки pytorch (рис. 3.11).

```
def flush():
  gc.collect()
  torch.cuda.empty_cache()
  torch.cuda.reset_peak_memory_stats()
```

Рисунок 3.9 – Функція очищення пам’яті

```
sentences['uk']['devtest'][10]['sentence']
```

✓ 0.0s Python

```
'Одна експериментальна вакцина, схоже, знижує вірогідність смерті від еболи, проте станом на зараз немає жодних ліків, які підійшли б для лікування існуючої інфекції.'
```

Рисунок 3.10 – Тестове речення

```
model = MarianMTModel.from_pretrained("Helsinki-NLP/opus-mt-uk-en").to(device)
tokenizer = MarianTokenizer.from_pretrained("Helsinki-NLP/opus-mt-uk-en")

print(translate_marian(model, tokenizer, sentences['uk']['devtest'][10]['sentence']))
print(f"Max memory allocated: {bytes_to_megabytes(torch.cuda.max_memory_allocated())} megabytes")
```

✓ 6.4s Python

```
One experimental vaccine seems to reduce the likelihood of death from Ebola, but there is no known cure for an existing infection.
Max memory allocated: 318.59326171875 megabytes
```

```
model = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_418M").to(device)
tokenizer = M2M100Tokenizer.from_pretrained("facebook/m2m100_418M")
flush()

print(translate_m2m(model, tokenizer, sentences['uk']['devtest'][10]['sentence'], "uk", "en"))
print(f"Max memory allocated: {bytes_to_megabytes(torch.cuda.max_memory_allocated())} megabytes")
```

✓ 6.4s Python

```
One experimental vaccine seems to reduce the likelihood of death from Ebola, but there are no medications to be used to treat an existing infection.
Max memory allocated: 1970.255859375 megabytes
```

```
model = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_1.2B").to(device)
tokenizer = M2M100Tokenizer.from_pretrained("facebook/m2m100_1.2B")
flush()

print(translate_m2m(model, tokenizer, sentences['uk']['devtest'][10]['sentence'], "uk", "en"))
print(f"Max memory allocated: {bytes_to_megabytes(torch.cuda.max_memory_allocated())} megabytes")
```

✓ 11.6s Python

```
One experimental vaccine seems to reduce the likelihood of death from Ebola, but there is currently no medication that would treat the existing infection.
Max memory allocated: 4947.38525390625 megabytes
```

```
model = AutoModelForSeq2SeqLM.from_pretrained("facebook/nllb-200-distilled-600M").to(device)
tokenizer = AutoTokenizer.from_pretrained("facebook/nllb-200-distilled-600M", src_lang="ukr_Cyrl")
flush()

print(translate_nllb(model, tokenizer, sentences['uk']['devtest'][10]['sentence'], "en"))
print(f"Max memory allocated: {bytes_to_megabytes(torch.cuda.max_memory_allocated())} megabytes")
```

✓ 10.7s Python

```
One experimental vaccine appears to reduce the likelihood of death from Ebola, but there are currently no drugs that could treat an existing infection.
Max memory allocated: 2375.20458984375 megabytes
```

Рисунок 3.11 – Виміри використаної пам’яті та результати

Як видно з вимірів, моделі, що мають вищу якість перекладу, відпрацьовують значно повільніше, тож щоб підібрати конкретну модель для задачі перекладу, потрібно звернути увагу на багато різних факторів, наприклад, об’єм доступної пам’яті, кількість наявних обчислювальних ресурсів, мови, які братимуть участь у перекладі, тощо.

ВИСНОВКИ

У рамках кваліфікаційної роботи було проведено порівняння моделей перекладу Helsinki-NLP/OPUS-MT, двох версій моделі M2M-100 та моделі NLLB для перекладу української мови. Модель M2M-100 на 1,2 мільярди параметрів показала вищу якість перекладу тестових речень, але помітно програла у швидкості. Вибір конкретної моделі залежить від поставленої задачі, ресурсів, якими ви володієте, та вимог до якості перекладу. Зверніть увагу, що це спрощена схема, і розробка готової до використання системи перекладу може включати додаткові етапи, зокрема попередню обробку даних [30–39], оптимізацію моделі, міркування щодо масштабованості тощо. Крім того, не забувайте про етичні міркування, конфіденційність даних і потенційні упередження, які можуть виникнути в системах машинного перекладу.

Результати дослідження апробовано у вигляді статті у студентському науковому журналі «UNIVERSUM» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Rules-Based Machine Translation (RBMT)? URL: https://omniscien.com/faq/what-is-rules-based-machine-translation/ (дата звернення 10.10.2023).

2. Rule Based Machine Translation. URL: https://medium.com/@keerthanasathish/rule-based-machine-translation-7b0074a91d20 (дата звернення 10.10.2023).

3. Statistical machine translation. URL: https://machinetranslate.org/statistical-machine-translation (дата звернення 10.10.2023).

4. The Big Guide to Machine Translation. URL: https://localizejs.com/articles/types-of-machine-translation/ (дата звернення 10.10.2023).

5. Chéragui, M. A. (2012). Theoretical Overview of Machine translation. *ICWIT*, 160-169.

6. Machines That Think: The Rise of Neural Machine Translation. URL: https://phrase.com/blog/posts/neural-machine-translation/ (дата звернення 10.10.2023).

7. Neural Machine Translation using a Seq2Seq Architecture and Attention (ENG to POR). URL: https://towardsdatascience.com/neural-machine-translation-using-a-seq2seq-architecture-and-attention-eng-to-por-fe3cc4191175 (дата звернення 10.10.2023).

8. A Simple Introduction to Sequence to Sequence Models. URL: https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/ (дата звернення 10.10.2023).

9. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

10. The Transformer Model. URL: https://machinelearningmastery.com/the-transformer-model/ (дата звернення 10.10.2023).

11. Language Translation with RNNs. URL: https://towardsdatascience.com/language-translation-with-rnns-d84d43b40571 (дата звернення 10.10.2023).

12. Machine Translation Using RNN. URL: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/supervised_learning/recurrent_neural_networks/machine-translation-using-rnn (дата звернення 10.10.2023).

13. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

14. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

15. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

16. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

17. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

18. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, *21*(1), 5485-5551.

19. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901.

20. Hugging Face NLP Course. URL: https://huggingface.co/learn/nlp-course (дата звернення 10.10.2023).

21. Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., ... & Birch, A. (2018). Marian: Fast neural machine translation in C++. *arXiv preprint arXiv:1804.00344*.

22. How To Fine-Tune m2m-100 Model in fairseq?. URL: https://medium.com/@juanluis1702/how-to-fine-tune-m2m-100-model-in-fairseq-6670676ddf2b (дата звернення 10.10.2023).

23. Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., ... & Joulin, A. (2021). Beyond english-centric multilingual machine translation. *The Journal of Machine Learning Research*, *22*(1), 4839-4886.

24. Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., ... & NLLB Team. (2022). No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

25. Helsinki-NLP/Opus-MT: Open neural machine translation models and web services. URL: https://github.com/Helsinki-NLP/Opus-MT (дата звернення 10.10.2023).

26. HuggingFace Transformers. URL: https://huggingface.co/docs/transformers/index (дата звернення 10.10.2023).

27. HuggingFace Datasets. URL: https://huggingface.co/docs/datasets/index (дата звернення 10.10.2023).

28. sacreBLEU. URL: https://github.com/mjpost/sacrebleu (дата звернення 10.10.2023).

29. PyTorch. URL: https://pytorch.org/ (дата звернення 10.10.2023).

30. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

31. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

32. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

33. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

34. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

35. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

36. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

37. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium*, September 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25-27.

38. Yakovleva O., Kovač M., Ardasov V. & Yeremenko I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities, *Public Administration and Regional Development*, 19(1), pp. 158-184.

39. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, 11, pp. 126938-126949.

40. Попов І., Кузьомін О. ДОСЛІДЖЕННЯ МЕТОДІВ НЕЙРОННИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ПЕРЕКЛАДУ УКРАЇНСЬКОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ. *UNIVERSUM*. 2023. № 2. С. 95–99.