Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ УКРАЇНСЬКО-АНГЛІЙСЬКОГО ПЕРЕКЛАДАЧА НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Виконав: студент 4 курсу,групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
Кузьменко О.Ю.

Керівник: к.т.н., проф. каф.КН
Колесницький О. К.
«07» 06 2024 р.

Рецензент: к.т.н., проф каф.КСУ
Биков М. М.
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри _____ Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 03 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Кузьменку Олексію Юрійовичу

1. Тема роботи: Програмний модуль українсько-англійського перекладача на основі нейронної мережі.
Керівник роботи: Колесницький Олег Костянтинович, к.т.н.
затверджені наказом вищого навчального закладу "11" 03 2024 року №80

2. Термін подання студентом роботи 27.05.2024

3. Вихідні дані до роботи :
Формати файлів – TXT, JSON;
Мова програмування – об'єктно-орієнтована;
Максимальна довжина тексту – 500 символів;
Обсяг навчальної вибірки – 20000 речень;
Обсяг тестової вибірки – 4000 речень.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
проаналізувати існуючі технології, методи і моделі нейромереж по перекладу тексту та вибрати найбільш ефективні;
розробити нейромережу роботи модуля для українсько-англійського перекладу;
спроектувати та програмно реалізувати модуль збереження даних перекладу.
протестувати

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Алгоритм роботи нейронної мережі
Процеси обробки інформації
Архітектура трансформера
UML діаграма класів
Лістинг коду трансформера
Результати програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Колесницький О.К., к.т.н., проф. каф. КН		

7. Дата видачі завдання 07.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області задачі про українсько-англійський переклад	08.05.24	10.05.24	
2	Аналіз відомих методів щодо українсько-англійський перекладу;	11.05.24	15.05.24	
3	Аналіз переваг та недоліків існуючих ресурсів для перекладу з української на англійську мову	16.05.24	23.05.24	
4	Розробка алгоритму нейромережевого модуля	24.05.24	27.05.24	
5	Реалізація модуля українсько-англійського перекладача на основі нейромережі трансформер	28.05.24	03.06.24	
6	Розробка інструкції користувача	04.06.24	05.06.24	
7	Оформлення матеріалів до захисту БДР	05.06.24	06.06.24	

Студент _____ **Кузьменко О. Ю.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Колесницький О.К**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота скаладається з 72 сторінок формату А4, на яких є 17 рисунків, 2 таблиці, перелік використаних джерел містить 12 найменувань.

Метою роботи є підвищення достовірності перекладу з української на англійську мову за рахунок використання нейронної мережі трансформер.

У роботі було обґрунтовано вибір нейронної мережі трансформер для перекладу з української мови на англійську, яка приймає вхідний документ TXT або JSON форматів. Програмний модуль перекладу тексту створено на мові програмування Python у програмному середовищі Intellij IDEA. Навчання програмного модуля відбувалось з використанням датасету TED2020. Навчальна вибірка складалась із 200000 речень обох мов. Тестова вибірка складалась із 40000 речень. Розроблений програмний модуль має достовірність перекладу з української на англійську мову 98%, а програма-аналог Google Translate має достовірність - 95%.

Ключові слова: переклад, українсько-англійський, трансформер, токен.

ABSTRACT

Bachelor's qualification work is based on 72 pages of A4 format, on which there are 17 figures, 2 tables, the list of used sources contains 12 denominations

The purpose of the work is to increase the reliability of the translation from Ukrainian to English by using a transformer neural network.

In the work, the choice of a transformer neural network for translation from Ukrainian to English, which accepts an input document in TXT or JSON formats, was justified. The text translation software module was created in the Python programming language in the Intellij IDEA programming environment. The software module was trained using the TED2020 dataset. The training sample consisted of 200,000 sentences of both languages. The test sample consisted of 40,000 sentences. The developed software module has a reliability of translation from Ukrainian to English of 98%, and the analogue program Google Translate has a reliability of 95%.

Keywords: translation, Ukrainian-English, transformer, token.

ЗМІСТ

ВСТУП

Нині переклад текстів є надзвичайно важливим, оскільки ми щодня зіштовхуємося з труднощами розуміння різних статей, людей, робочих документів на інших мовах і т.д. Завдання перекладу текстів базується на обробці природної мови і має велику актуальність, оскільки може бути використане в різних галузях з різними цілями, особливо в періоди високих міжнародних відносин, коли важливо розуміти один одного і мати можливість читати документацію різними мовами та інше. Для тестування було обрано найточнішу модель серед багатьох.

Нейромережі мають значення у перекладі через їхню здатність навчатися на великих обсягах даних та виявляти складні зв'язки між мовами. Ось кілька причин, чому нейромережі вважаються ключовим інструментом у сфері перекладу:

- Здатність розуміти контекст: Нейромережі можуть аналізувати контекст та семантику речень, що допомагає їм краще розуміти сутність тексту та правильно перекладати його.
- Автоматизація процесу навчання: Нейромережі можуть навчатися на великій кількості паралельних текстів (тексти на одній мові та їх переклади на іншій мові), що сприяє автоматизації та покращенню якості перекладу з часом.
- Адаптація до різноманітних мовних структур: Нейромережі можуть адаптуватися до різних особливостей мовних структур, що дозволяє їм ефективно працювати з мовами різного типу та походження.
- Покращення якості перекладу: Використання нейромереж може призвести до більш точного та природного перекладу, оскільки вони здатні враховувати мовні варіації, вирази та ідіоми.
- Можливість використання у реальному часі: Деякі нейромережі можуть працювати швидко, що дозволяє використовувати їх для миттєвого

перекладу текстів у різних ситуаціях, таких як онлайн-спілкування, документація та інше.

У цілому, нейромережі є потужним інструментом для автоматизації та покращення процесу перекладу, що робить їх важливими в сучасній сфері мовного перекладу.

В даній роботі буде розглянута програмна реалізація згорткової нейронної мережі, завданням якої є переклад з української на англійську мову. Ця програма може бути застосована в різних випадках, коли необхідно перекласти текст з української на англійську мову. Також програма дає змогу для подальшої взаємодії з вихідними даними або зберігання, упорядкування в базах даних, оскільки формат вихідного файлу txt або json.

Об’єкт дослідження – процес комп’ютеризованого українсько-англійського перекладу.

Предмет дослідження – програмні засоби українсько-англійського перекладу на основі нейронних мереж та достовірність їх роботи.

Мета дослідження – підвищення достовірності перекладу з української на англійську мову за рахунок використання нейронної мережі трансформер.

В роботі роз’язуються такі задачі:

- Огляд існуючих методів перекладу текстів та вибір конкретного напрямку дослідження.
- Розробка структури програмного модуля для українсько-англійського перекладача.
- Обгрунтування вибору архітектури нейронних мереж трансформера для подальшого використання.
- Розробка алгоритму роботи програмного модуля перекладача.
- Програмна реалізація розробленого модуля українсько-англійського перекладача.
- Тестування програмного модуля українсько-англійського перекладача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПЕРЕКЛАДУ ТЕКСТІВ

1.1 Постановка задачі

Вхідні дані текст, необхідний для перекладу. Потрібно, використовуючи глибоку нейронну мережу перекладати текст з української мови на англійську. Дану задачу можна розглядати як задачу коректного перекладу тексту.

Метою цієї роботи є створення програмного модуля, який здатний автоматично перекладати тексти з однієї мови на іншу за допомогою комп'ютера та також підвищення точності перекладу тексту використовуючи згорткову нейромережу. Результат перекладу повинен бути еквівалентним за змістом вихідному тексту та придатним для спілкування в межах відповідної мовної групи.

План вирішення задач:

- Аналіз вихідного тексту та побудова внутрішньої моделі змісту.
- Підготовка перекладу на основі алгоритмів машинного навчання.
- Оцінка якості перекладу за допомогою експертної оцінки.
- Провести порівняння результатів роботи розробленої програми з аналогами.

Постановка завдання полягає в розробці програмного модуля, який здатний автоматично перекладати тексти з однієї мови на іншу. Програма повинна бути здатна адекватно відтворювати зміст вихідного тексту, зберігаючи структуру та смислові зв'язки.

Класифікація об'єкту в контексті даного тексту означає ідентифікацію та віднесення текстів різних мов до відповідних категорій або класів. Таким чином, класифікація об'єкту полягає у визначенні мови кожного тексту та автоматичному перекладі його на іншу мову з використанням комп'ютерних алгоритмів.

1.2 Огляд відомих методів перекладу тексту

Переклад тексту є важливою складовою міжкультурної комунікації, що сприяє обміну інформацією між різними мовами та культурами. Сьогодні існує багато методів перекладу, кожен з яких має свої переваги та недоліки.

У цьому розділі буде проведений огляд існуючих методів перекладу тексту з однієї мови на іншу. Для кращого розуміння і порівняння різних підходів до перекладу тексту будуть розглянуті та проаналізовані наступні методи:

1. Машинний переклад

Машинний переклад – це автоматичний переклад тексту за допомогою комп'ютерних алгоритмів. Існує кілька основних підходів до машинного перекладу:

– Статистичний машинний переклад (SMT)

Цей метод базується на статистичних моделях, які використовують двомовні корпуси текстів для створення ймовірнісних правил перекладу. Статистичний підхід дозволяє автоматично вивчати закономірності перекладу, але може давати неточні результати при перекладі складних або рідковживаних виразів.

– Нейронний машинний переклад (NMT)

Нейронний переклад використовує штучні нейронні мережі (рис. 1.1) для створення моделей перекладу.

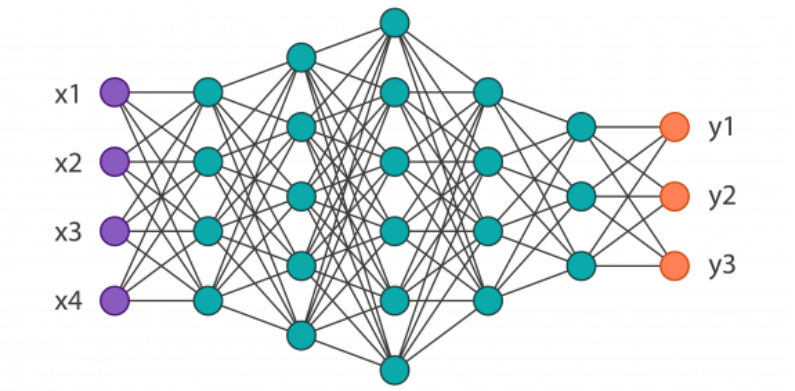


Рисунок 1.1 – Структурна схема нейронної мережі

Цей метод забезпечує більш природний та контекстуально правильний переклад у порівнянні зі статистичним підходом. Нейронні мережі вчаться на великих обсягах даних і можуть адаптуватися до різних стилів та контекстів.

2. Ручний переклад

Ручний переклад виконується професійними перекладачами, які володіють обома мовами та розуміють культурні нюанси. Основні види ручного перекладу включають:

– Літературний переклад

Цей вид перекладу застосовується до художніх текстів, де важливі не лише точність, але й емоційне забарвлення, стиль та авторські особливості. Літературні перекладачі часто стикаються з викликами адаптації метафор, ідіом та культурних реалій.

– Технічний переклад

Технічний переклад спеціалізується на текстах, пов'язаних з технікою, наукою, медициною та іншими галузями, що потребують точності та спеціалізованих знань. Технічні перекладачі повинні бути обізнаними в термінології та специфіці відповідної галузі.

3. Гібридні методи

Гібридні методи об'єднують машинний та ручний підходи для досягнення кращих результатів. Вони передбачають використання машинного перекладу для створення чорнового варіанту, який потім редагується професійним перекладачем. Це дозволяє скоротити час і знизити витрати, зберігаючи високу якість перекладу.

4. Постредагування машинного перекладу

Постредагування – це процес редагування машинного перекладу людиною з метою покращення якості кінцевого тексту. Існують два основні типи постредагування:

– Легке постредагування

Цей підхід передбачає мінімальні втручання, спрямовані на досягнення зрозумілості та загальної правильності тексту. Використовується в ситуаціях, коли висока точність не є критичною.

– Повне постредагування

Включає детальне редагування з метою досягнення максимальної точності, стилістичної відповідності та природності перекладу. Використовується для текстів, де висока якість є обов'язковою.

Різні методи перекладу мають свої переваги та сфери застосування. Вибір методу залежить від типу тексту, вимог до якості перекладу та ресурсів, що є в розпорядженні. Комбінація машинного та ручного перекладу, а також постредагування, часто дає найкращі результати, забезпечуючи ефективність та високу якість кінцевого продукту.

Цей огляд дозволить отримати уявлення про різноманітність підходів до перекладу тексту та їхні переваги та недоліки, що буде корисним для подальшого розгляду та вибору оптимального методу для реалізації програмного модуля перекладу.

Завдання нейромережі в перекладі тексту полягатиме в розробці моделі, яка здатна автоматично перекладати тексти з української мови на англійську. Нейронний машинний переклад є найпопулярнішим завдяки своїй ефективності та швидкості, тоді як ручний переклад залишається найдостовірнішим через увагу до деталей та культурних контекстів.

1.3 Обгрунтування вибору аналогу до програмного модуля українсько-англійського перекладача

На сьогоднішній день найбільш поширеними стали системи машинного перекладу, які базуються на статистичному аналізі. Ці системи є простішими у створенні та підтримці, оскільки вже існує велика кількість інформації, перекладеної на кілька мов, така як статті, книги, новини, що значно полегшує процес навчання систем.

Серед найпопулярніших онлайн-ресурсів машинного перекладу, доступних сучасним інтернет-користувачам, можна відзначити наступні:

– Google Translate є одним із найвідоміших і найбільш використовуваних онлайн-ресурсів для перекладу текстів. Він підтримує понад 100 мов і пропонує такі функції, як переклад тексту, голосовий переклад, переклад зображень та документів. Завдяки використанню нейронних мереж Google Translate забезпечує високий рівень точності та природності перекладу.

– Microsoft Translator (рис. 1.2) також відомий як Bing Translator, пропонує переклад текстів і розмов у реальному часі.

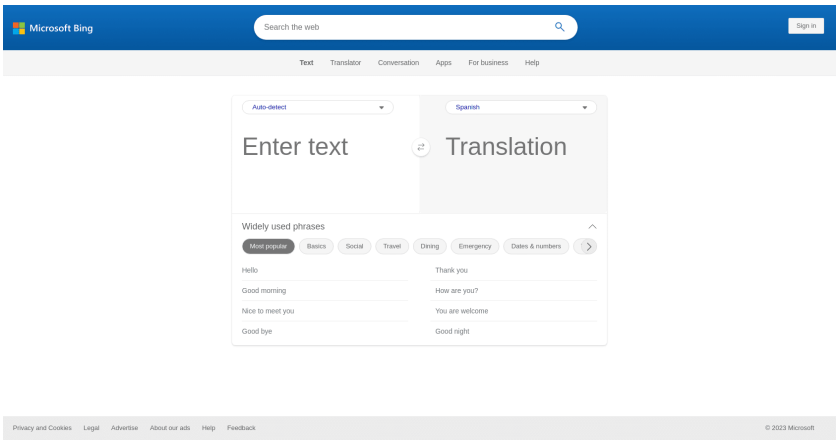


Рисунок 1.2 – Вікно програми Microsoft Translator

Платформа підтримує більше ніж 70 мов і інтегрується з іншими продуктами Microsoft, такими як Office, Skype та Cortana. Microsoft Translator також використовує нейронні мережі для покращення якості перекладу.

– DeepL Translator (рис. 1.3) є відносно новим, але швидко набирає популярність завдяки високій точності та природності перекладів.

Рисунок 1.3 – Вікно програми Microsoft Translator

Платформа використовує передові нейронні мережі, що дозволяють забезпечити якість перекладу, яка часто перевершує інших конкурентів. DeepL підтримує менше мов у порівнянні з Google та Microsoft, але активно розширює свій асортимент.

– Reverso пропонує переклад текстів та фраз із підтримкою кількох десятків мов. Особливістю Reverso є можливість переглядати приклади використання слів та виразів у контексті, що допомагає краще зрозуміти значення та вживання перекладених фраз. Сервіс також пропонує функції навчання мов.

– iTranslate є популярним додатком для перекладу текстів, голосу та веб-сторінок. Платформа підтримує більше ніж 100 мов і пропонує додаткові функції, такі як переклад в автономному режимі, словники та інструменти для вивчення мов.

– Papago розроблений південнокорейською компанією Naver, спеціалізується на перекладі азійських мов, таких як корейська, японська, китайська, а також підтримує інші мови. Сервіс пропонує переклад тексту, зображень, голосу та розмов у реальному часі, використовуючи нейронні мережі для забезпечення високої якості перекладу.

Ці онлайн-ресурси машинного перекладу забезпечують користувачів широким спектром можливостей для перекладу текстів, голосу, зображень та

документів. Використання нейронних мереж дозволяє досягати високої точності та природності перекладу, що робить ці інструменти надзвичайно корисними в повсякденному житті та професійній діяльності.

У додатку Google Перекладач (рис. 1.4) можна перекладати надрукований, рукописний, сфотографований і надиктований текст понад 100 мовами. Цей сервіс також доступний у веб-переглядачі [1].

Велика база доступних мовних пар та цілий ряд функціональних можливостей, що надаються сервісом машинного перекладу від компанії Google, стали причиною того, що даним сервісом щодня користуються більше 500 млн. людей.

Недоліки:

– поле введення тексту обмежується 5000 символами;

– різна точність перекладу кожної пари мов;

– періодичні збої в алгоритмах призводять до отримання неадекватного перекладу;

– достовірність перекладу великого об’єму тексту;

– відсутність функції прямого перекладу тексту під час використання мовних пар, де жодна з мов не є англійською (у такому разі англійська мова виступає в ролі посередника між мовою оригіналу та цільовою мовою перекладу).

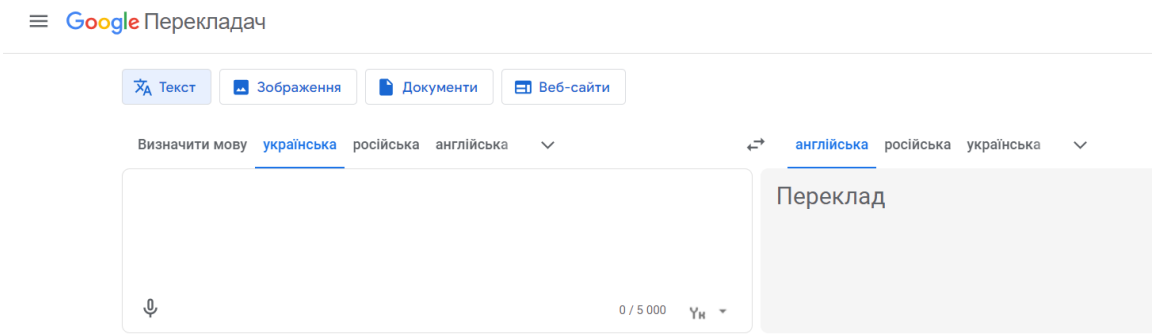


Рисунок 1.4 – Вікно програми Google Translate

Також варто зазначити, що через те, що вибір варіантів контролюється статистичним алгоритмом, при перекладі звичайних загальновживаних слів Google Перекладач може пропонувати в числі можливих варіантів нецензурні слова. На результат перекладу також може вплинути масове пропонування якого-небудь, в тому числі завідомо неправильного варіанту перекладу.

У зв'язку з цим, необхідно розробити новий програмний модуль для українсько-англійського перекладу з підвищеною достовірністю функціонування.

1.4 Висновок

В даному розділі роботи було сформульовано постановку завдання, яке полягає в розробці програмного модуля для автоматичного перекладу текстів з української мови на англійську. Метою роботи є створення модуля, який здатний адекватно перекладати тексти, зберігаючи їхній сенс та структуру. Для досягнення цієї мети було запропоновано план вирішення задачі, який включає аналіз вихідного тексту, підготовку перекладу на основі алгоритмів машинного навчання, оцінку якості перекладу та порівняння результатів роботи з аналогами.

Був проведений огляд існуючих методів та програмних засобів перекладу тексту, основними недоліком яких є невисока достовірність перекладу. На основі цього аналізу було визначено необхідність розробки нового програмного модуля з підвищеною достовірністю функціонування, що є головною метою подальших досліджень у цій області.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ УКРАЇНСЬКО-АНГЛІЙСЬКОГО ПЕРЕКЛАДАЧА

2.1 Структура процесів обробки інформації програмного модуля українсько-англійського перекладача

Машинний переклад на основі нейронних мереж - це метод перекладу слів з однієї мови на іншу, який використовує алгоритми нейронних мереж. Зараз відомо, що добре навчені моделі машинного перекладу здатні ефективно аналізувати контекст тексту та використовувати попередньо навчені моделі для точних перекладів. Google Translate, DeepL - це приклади відомих рішень машинного перекладу на основі нейронних мереж, доступних у будь-якому браузері.

Нейронна мережа (рис. 2.1) для перекладів побудована на основі концепції нейронних мереж, що імітують роботу людського мозку, навчаючись так, щоб наближатися до людського мислення. Нейрони утворюють шари, а мережа складається з вузлів, які взаємодіють між собою. Кожен нейрон і його зв'язки мають вагу, яка залежить від обсягу даних, що аналізуються.

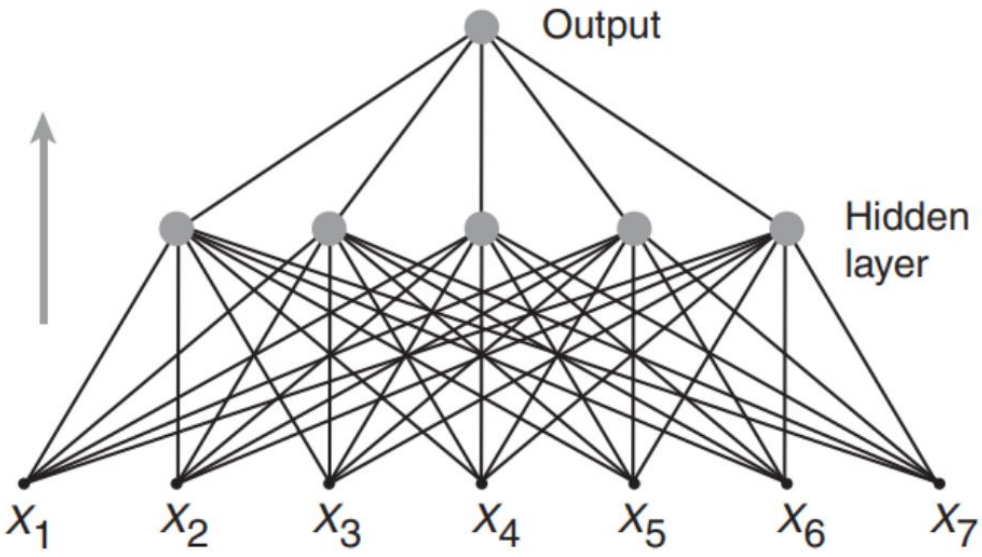


Рисунок 2.1 – Нейронна мережа

У традиційному підході до перекладу використовуються фрази, складені зі слів, у той час як у машинному перекладі використовується інший підхід. Нейронна мережа навчається перекладати, аналізуючи речення та контекст. Такі мережі можуть розуміти різні типи перекладу, такі як юридичний, медичний, приватний, і можуть бути навчені на власному словниковому запасі.

Машинний переклад на основі нейронних мереж (рис. 2.2) потребує наявності великої кількості даних для навчання, оскільки саме ці дані допомагають моделі у візуалізації процесу перекладу. Кожне слово чи речення у тексті мають однакове значення, оскільки вони всі впливають на модель і проходять однакові процеси та кроки алгоритму.

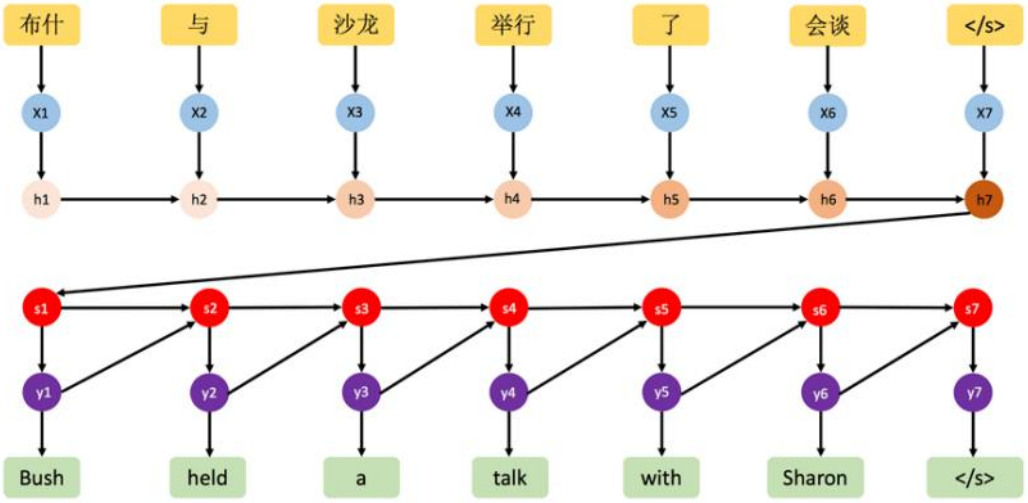


Рисунок 2.2 – Імітація перекладу тексту

Завданням цієї роботи є розробка програмного модуля для перекладу тексту з української мови на англійську. Таким чином, вхідними даними для цього модуля є файл з текстом, який потребує перекладу. Кроки обробки цього файлу представлені на рисунку 2.3.

Рисунок 2.3 – Процеси обробки інформації при українсько-англійському перекладі

Згорткова нейронна мережа (рис.2.4) - один з видів нейронних мереж, що використовується для розпізнавання та обробки зображень. Проте згорткові нейронні мережі також можуть бути застосовані для обробки природної мови. Ці нейронні мережі працюють з матрицею, тому необхідно її сформувати. Для цього потрібно зіставити точку в багатовимірному просторі об'єкта; у випадку сентимент-аналізу ми використовуємо слова. Ця процедура називається “вбудовування”.

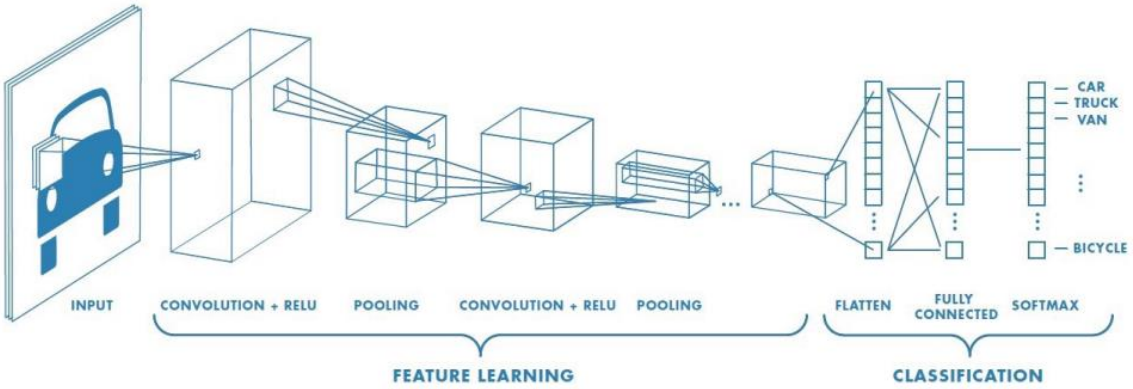


Рисунок 2.4 – Візуалізація алгоритму згорткової нейромережі

Хоча згорткові нейронні мережі (CNN) традиційно використовуються для обробки зображень, вони також можуть бути адаптовані для обробки тексту за допомогою методів, таких як Word2Vec (рис. 2.5).

Ефективність Word2Vec полягає у створенні векторів подібних слів. Використовуючи великі вибірки, Word2Vec здатний точно оцінити контекст вживання слова на основі його частоти у тексті. Алгоритм працює наступним чином: word2vec приймає вибірку текстів у ролі вхідних даних і надає кожному слову вектор, таким чином отримуючи координати слів. Спочатку генерується корпус зі слів, а потім на текстах відбувається навчання. Слова, що мають схожий сенс, матимуть схожі вектори. Отримані вектори зі слів застосовують для обробки природної мови та машинного навчання.

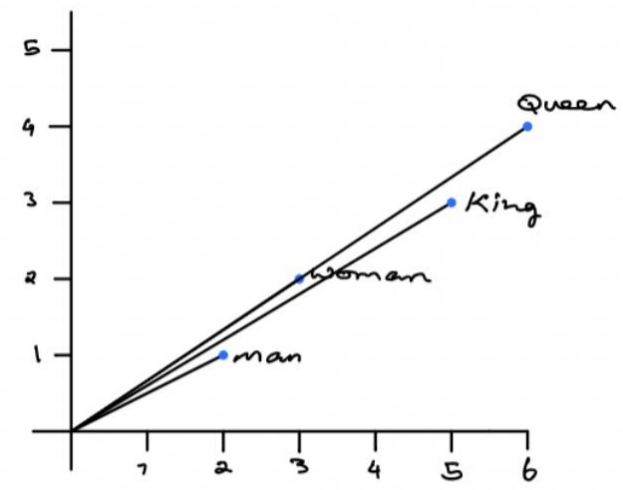


Рисунок 2.5 – Візуалізація Word2Vec

Як зазначено вище, для створення матриці використовується метод Word2Vec. Матриця зображення складається з таких вимірів, як ширина, висота та канали. Якщо використовувати лише ширину, слова втратять контекст і текст буде класифіковано неправильно. Спочатку обробляється отримана матриця шарами згортки. Для підбору розмірів фільтрів налаштовується лише один параметр - висота, тому розмір нашої матриці буде різним. Далі працюємо з картою ознак, що формується на виході кожного фільтра, з визначеним шаром дискретизації, через що розмір карт ознак зменшується. Таким чином, ми відкидаємо неважливу інформацію і залишаємо тільки найважливіше. Залишаються лише найбільш важливі n-грамми. Потім отримані карти ознак об'єднуються в одну векторну ознаку. Наступним кроком алгоритму є подача карти ознак на вхід у загальний повнозв’язковий шар, звідки потім отримуємо бажаний результат.

Проте, згорткові нейронні мережі не підходять для перекладу слів через їхню спеціалізацію на обробці просторових даних, таких як зображення, де важлива локальна структура. Обробка тексту вимагає врахування послідовності і контексту, що краще забезпечується архітектурою трансформерів, які можуть працювати з довгими залежностями в тексті завдяки механізму самоуваги. Тому для завдань перекладу доцільніше використовувати трансформери.

Рекурентні нейронні мережі (RNN) застосовуються для обробки послідовностей тексту як вхідних даних або при поверненні послідовностей тексту як вихідних даних, або в обох випадках. Їх також називають повторюваними, тому що приховані шари мережі мають цикл, у якому вихідні дані та стан комірки з кожного кроку часу стають входами на наступному кроці часу. Завдяки цій формі пам'яті контекстна інформація може проходити через нейронну мережу, щоб результати, отримані раніше, можна було використати у мережевих операціях в актуальний момент часу.

Згорткові нейронні мережі (CNN) не дозволяють такому типу контексту часових рядів протікати через мережу, як це роблять RNN (рис. 2.6). CNN зазвичай використовуються для обробки даних з фіксованими розмірами, таких як зображення, де важлива локальна структура.

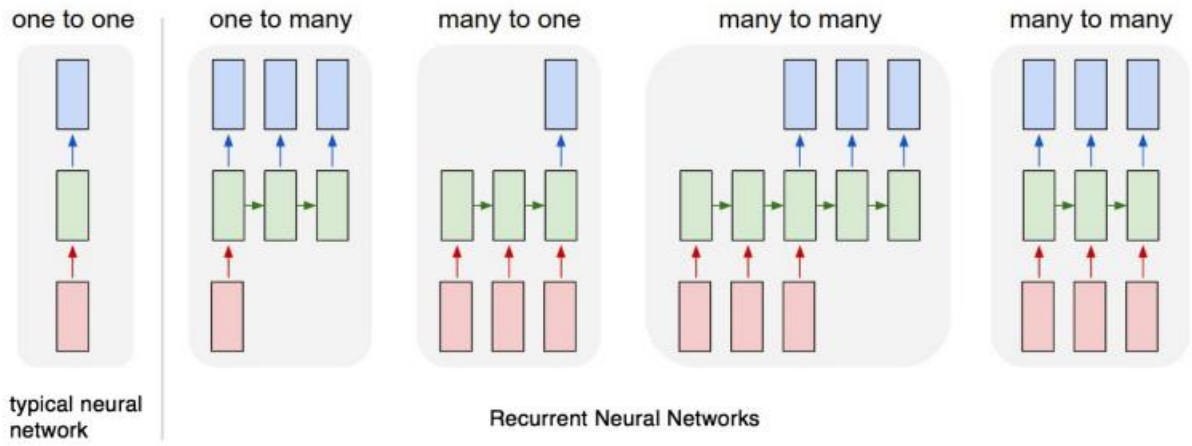


Рисунок 2.6 – Рекурентна нейронна мережа

Згідно з попереднім рисунком можна виділити такі складові:

- Входи: послідовність, що подається на вхід моделі, відповідно одне слово для окремого кроку часу. Всі слова кодуються унікально, тобто немає повторень.

- Вбудовування шарів: вбудовування застосовуються для зіставлення кожного слова у вектор. Його розмір напряму залежить від складності словникового запасу.

- Повторювані шари (Кодер): у даному випадку контекстні змінні попередніх моментів часу застосовуються в актуальний момент часу.

- Декодер: ці шари декодують закодовану послідовність в правильну послідовність класифікації.

- Виходи: вихідні дані представляються у вигляді послідовності цілих чисел або векторів.

Модель використовує об'єднання кодера і декодера. Суть кодера полягає в тому, що він підсумовує вхідні дані в певну контекстну змінну.

Наступним кроком є декодування цієї змінної для отримання кінцевої послідовності. Оскільки і кодер, і декодер є повторюваними, вони мають цикли, які обробляють кожну частину послідовності на різних кроках часу.

Оскільки і кодер, і декодер (рис. 2.7) є рекурентними, вони обробляють кожну частину послідовності поетапно, на різних тимчасових кроках.

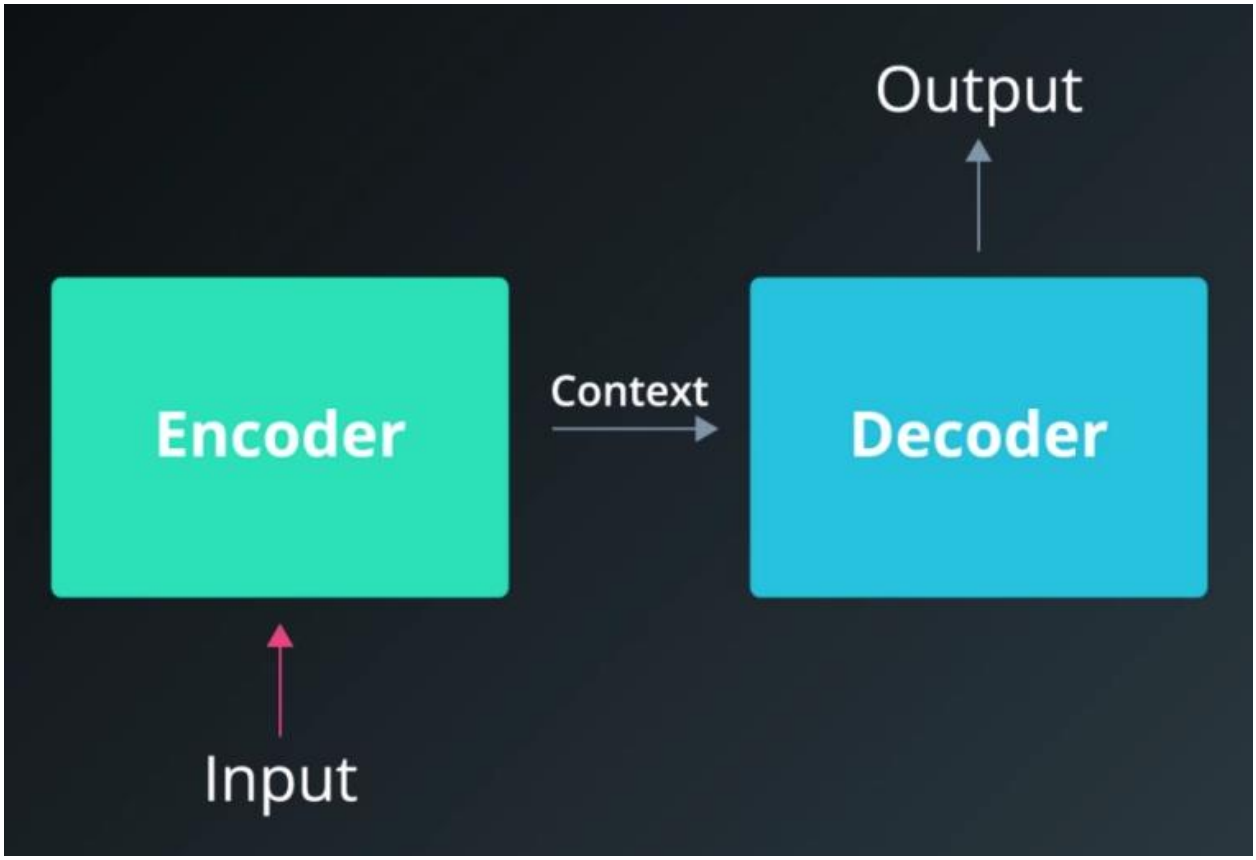


Рисунок 2.7 – Кодер і декодер

На кожному з моментів часу кодер «читає» вхідну послідовність і обробляє її прихований стан. Потім відбувається передача цього стану на наступний момент часу. Чим більше прихований стан, тим більша здатність до навчання моделі, але також вищі вимоги до обчислень.

Можна бачити, що для кожного актуального моменту часу є прихований стан і слово з послідовності. Для кодера цим є наступне слово у вхідній послідовності. Для декодера це попереднє слово з вихідної послідовності.

Рекурентні нейронні мережі (RNN) добре підходять для обробки послідовних даних, оскільки вони можуть зберігати контекст попередніх елементів у прихованих станах. Однак RNN мають деякі суттєві обмеження, які ускладнюють їх використання для перекладу тексту. По-перше, вони схильні до проблеми зникання градієнта, що ускладнює навчання при роботі з довгими послідовностями. Це означає, що інформація з початку тексту може бути втрачена або спотворена до того часу, коли вона потрібна на пізніших етапах.

По-друге, RNN обробляють послідовність слів по одному слову за раз, що робить їх повільними для навчання та застосування на великих текстових даних. На відміну від цього, трансформери, які використовують механізм самоуваги, можуть обробляти всі слова в послідовності одночасно, ефективно зберігаючи контекст і забезпечуючи більш високу точність і швидкість перекладу. Тому трансформери є більш підходящими для завдань машинного перекладу, ніж традиційні RNN.

Трансформерні мережі — це тип глибоких нейронних мереж, які найбільше використовуються для обробки мови. На даний момент такі мережі дозволяють розв’язувати практичні завдання перекладу, генерації текстів, створення відповідей на запитання, а також інші. Вперше вони були описані в роботі “ Attention is All You Need” Васвані і ін., 2017 р та є основою для багатьох сучасних моделей обробки мови.

Основні характеристики трансформерних мереж для перекладу текстів:

– Механізм уваги: завдяки трансформерам застосовується механізм самоуваги, який дозволяє кожному слову в реченні взаємодіяти з іншими словами. Це забезпечує кращу контекстуалізацію та залежності слів при меншому об’єми робочої пам’яті в нейромережі.

– Ефективність архітектури перекладача: трансформери можуть обробляти більші кількості даних за один раз, що забезпечує їм відносно швидке навчання.

– Зменшена потреба в попередній обробці: Як і Згорткові нейронні мережі (ЗНМ), трансформери вимагають мінімальної попередньої обробки тексту. Мережа самостійно навчається виявляти релевантні особливості завдяки механізму уваги, що зменшує залежність від попередніх знань та ручної роботи.

– Структура мережі: Класичний трансформер включає енкодер і декодер. Енкодер обробляє вхідний текст і перетворює його у внутрішнє представлення, тоді як декодер використовує це представлення для генерації вихідного тексту. Обидва компоненти складаються з кількох шарів самоуваги та повнозв'язних шарів.

– Використання позиційного кодування (Positional Encoding): Трансформери не мають вбудованого поняття послідовності, тому використовують позиційне кодування для додавання інформації про порядок слів у реченні, що є важливим для точного перекладу.

– Ефективність порівняно з рекурентними мережами: На відміну від рекурентних нейронних мереж, трансформери здатні обробляти весь текст одночасно, а не послідовно, що значно підвищує швидкість обробки та навчання, особливо на великих наборах даних.

– Запобігання проблемам з градієнтами: Завдяки механізму самоуваги та нормалізації шарів, трансформери допомагають уникати проблем зникання або вибуху градієнтів, які часто виникають у традиційних багатошарових нейронних мережах під час тренування за допомогою зворотного поширення.

Трансформери стали основою для багатьох сучасних досягнень в обробці природної мови і є ключовою технологією для створення ефективних систем машинного перекладу.

Глибоке навчання - це вид машинного навчання, що наділяє комп'ютери здатністю вчитися на досвіді та розуміти світ у термінах ієрархії концепцій [3].

Ця нейромережа дозволяє моделі ефективно зосереджуватися на різних частинах вхідних даних, навіть якщо вони знаходяться далеко одна від одної

в послідовності. Токени - це базові одиниці тексту, які трансформер обробляє; вони можуть бути окремими словами, частинами слів або навіть символами. Коли текст подається на вхід трансформера, він спочатку розбивається на токени, які потім перетворюються на векторні представлення. Ці вектори проходять через кілька шарів уваги і перетворень, що дозволяє моделі зрозуміти контекст і зв'язки між токенами, забезпечуючи високоякісні результати в завданнях обробки природної мови.

Перевагами нейромережевого перекладу є [4]:

– Глобальна доступність: пропонуючи автоматичний переклад на своєму веб-сайті, зробіть свій вміст доступним для глобальної аудиторії. Відвідувачі з різних регіонів і різних мов можуть легко зрозуміти ваш веб-сайт і взаємодіяти з ним, тим самим підвищуючи задоволеність і залученість користувачів.

– Покращена взаємодія з користувачем: автоматичний переклад забезпечує гарну взаємодію з користувачем. Відвідувачі можуть переглядати ваш веб-сайт улюбленою мовою, що забезпечує більш зручну та персоналізовану роботу.

– Збільшення охоплення та трафіку: ви можете розширити охоплення та залучити відвідувачів із різних мовних спільнот. Це може призвести до збільшення трафіку на вашому веб-сайті, потенційно збільшивши конверсії та продажі.

– Покращена оптимізація пошукових систем: коли ви надаєте перекладений вміст на своєму веб-сайті, пошукові системи можуть індексувати та ранжувати його за релевантними пошуковими запитами на певній мові. Це підвищує видимість вашого веб-сайту на сторінках результатів пошукової системи (SERP) кількома мовами, збільшує загальний органічний трафік і посилює зусилля з оптимізації пошукової системи (SEO).

– Локалізація для цільових ринків: нейронний автоматичний переклад дозволяє локалізувати вміст вашого веб-сайту. Ця локалізація може включати адаптацію описів продукту, маркетингових матеріалів і інтерфейсів

користувача, щоб резонувати з місцевою аудиторією, що призведе до збільшення залучення клієнтів і конверсій на цих ринках.

– Ефективність витрат і часу: за допомогою автоматизованої системи перекладу ви можете швидко й автоматично перекладати новий вміст або оновлення на своєму веб-сайті без допомоги перекладача, зменшуючи витрати та оптимізуючи процес локалізації.

Нейромережі трансформери виявляються ідеальним інструментом для завдань перекладу тексту з кількох причин. По-перше, вони використовують механізми самоуваги, що дозволяє їм одночасно розглядати всі слова вхідного речення, зберігаючи при цьому важливий контекст. Це сприяє кращому розумінню взаємозв'язків між словами і фразами, що допомагає у точнішому перекладі. По-друге, трансформери можуть працювати паралельно, що робить їх значно швидшими в порівнянні з рекурентними моделями, такими як RNN. Це дозволяє ефективно обробляти великі обсяги тексту і зменшує час, необхідний для навчання та використання моделі. Нарешті, трансформери демонструють високу рівновагу між швидкістю та якістю перекладу, забезпечуючи вражаючу точність і ефективність у вирішенні завдань перекладу тексту.

2.3 Методика навчання нейронних мереж трансформерів

Для навчання нейронної мережі трансформера необхідно пройти наступні етапи:

1. Підготовка даних

– Навчальні дані складаються з пар речень для вихідної та цільової мов. Ці пари речень використовуються для тренування моделі.

– Дані розділяються на тренувальні та валідаційні набори.

2. Токенізація тексту

– Текст кожного речення токенізується, тобто розбивається на окремі токени, які є найменшими одиницями тексту.

– Для цього може використовуватися метод субсловної токенізації, наприклад, з використанням алгоритму SentencePiece.

3. Побудова моделі трансформера

– Модель трансформера складається з енкодера та декодера.

– Енкодер приймає на вхід послідовності токенів вихідних речень, а декодер приймає послідовності токенів цільових речень.

4. Підготовка даних для навчання

– Токенізовані дані перетворюються на числові послідовності, які представляють індекси токенів у словнику.

– Ці дані розділяються на тренувальні та валідаційні набори. Це необхідно для оцінки якості навчання моделі та підбору оптимальних параметрів.

5. Навчання моделі

– Модель тренується на тренувальних даних з використанням методу зворотного поширення помилки та алгоритму оптимізації, такого як Adam.

– Під час навчання модель намагається мінімізувати функцію втрат, яка визначає різницю між прогнозованими та справжніми значеннями.

– Відбувається цикл тренування, де модель обробляє навчальні партії даних та оновлює ваги згідно зі значеннями функції втрати.

6. Оцінка моделі

– Модель оцінюється на валідаційних даних, щоб визначити її якість та виявити ознаки перенавчання.

– Це дозволяє покращити параметри моделі та вирішити проблеми перенавчання.

7. Ітерація

– Процес тренування може бути повторений кілька разів (екіпажі), щоб покращити якість моделі.

– Після завершення тренування модель може бути використана для перекладу нових текстів, які не використовувалися під час навчання.

8. Інференс

– Після навчання модель використовується для інференсу, тобто перекладу текстів.

– Реалізовані два методи декодування: жадібний та збірний. Вони використовуються для генерації перекладу вхідного речення.

– Додаткові етапи навчання трансформерів:

9. Регуляризація

Для уникнення перенавчання використовуються техніки регуляризації, такі як dropout, який випадково виключає певні нейрони під час тренування для запобігання надмірному пристосуванню моделі до тренувальних даних.

10. Розширення даних

Розширення даних, такі як аугментація, можуть бути використані для збільшення розмаїття тренувальних прикладів та покращення загальної здатності моделі до генералізації.

11. Файн-тюнінг

Після первинного навчання модель може бути додатково налаштована (файн-тюнінг) на специфічних наборах даних для конкретних завдань або галузей, що дозволяє підвищити точність перекладу у вузько спеціалізованих контекстах.

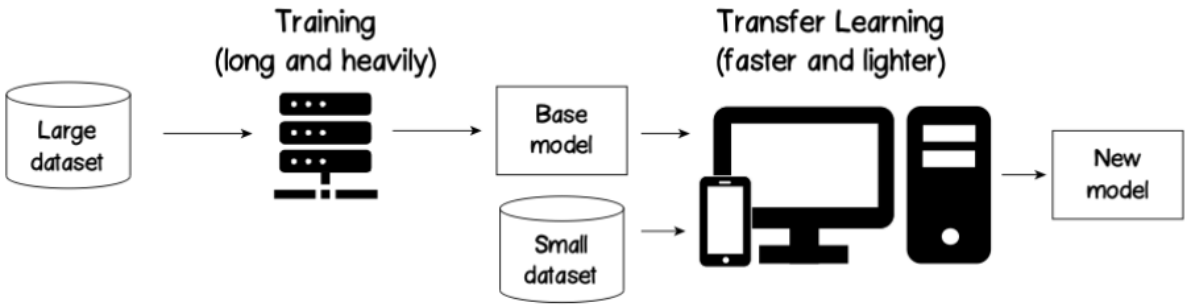


Рисунок 2.7 – Структурна схема Transfer Learning

Підготовка та навчання моделі трансформера (рис. 2.7) для машинного перекладу є складним та багатоетапним процесом. Першим кроком є

підготовка даних, яка включає розділення даних на тренувальні та валідаційні набори, а також токенізацію тексту для подальшої обробки. Побудова моделі трансформера включає створення енкодера та декодера, які опрацьовують вхідні та вихідні речення відповідно. Підготовлені дані перетворюються на числові послідовності, які передаються моделі для тренування.

Під час тренування моделі використовується алгоритм зворотного поширення помилки та метод оптимізації Adam для оновлення ваг моделі та мінімізації функції втрати. Після кожної епохи тренування модель оцінюється на валідаційних даних для контролю якості та уникнення перенавчання. Процес тренування може бути повторений кілька разів для покращення якості моделі. Після завершення тренування модель може бути використана для перекладу нових текстів за допомогою методу інференсу. Для цього реалізовані два методи декодування: жадібний та збірний, які забезпечують генерацію перекладу вхідних речень. Регуляризація та розширення даних також грають важливу роль у підвищенні якості моделі.

Отже, процес навчання моделі трансформера для машинного перекладу включає кілька кроків, від підготовки даних до інференсу, та вимагає уважності та систематичного підходу для досягнення найкращих результатів.

2.4 Розробка алгоритму роботи програмного модуля українсько-англійського перекладача

Алгоритм роботи програмного модуля українсько-англійського перекладача, розроблений на основі всіх описаних у цьому розділі процедур, представлено на рисунку 2.8 і він складається з двох етапів: 1) навчання та 2) використання.

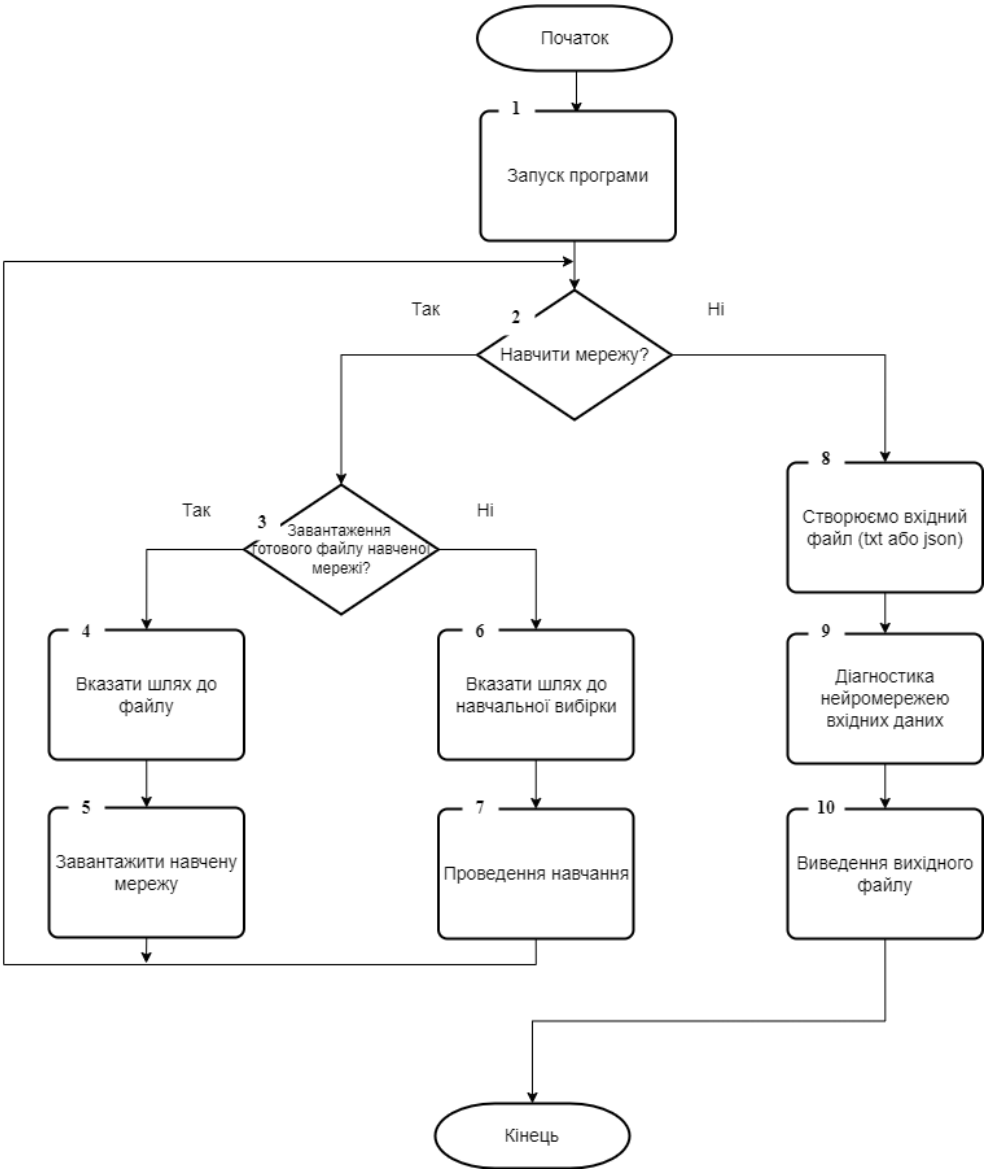


Рисунок 2.8 – Схема алгоритму роботи програми

Навчання нейромережі трансформера на датасеті, який складається з пар речень для вихідної та цільової мов, є складним та багатоетапним процесом. Починаючи з підготовки даних, текст кожного речення токенізується, щоб розбити його на окремі токени, які є найменшими одиницями тексту. Після цього дані перетворюються на числові послідовності, які передаються моделі для тренування. Під час навчання моделі намагається мінімізувати функцію втрати, яка визначає різницю між прогнозованими та справжніми значеннями. Після кожної епохи тренування модель оцінюється на валідаційних даних для контролю якості та уникнення

перенавчання. Після завершення тренування модель може бути використана для перекладу нових текстів за допомогою методу інференсу. UML діаграма класів розроблюваного програмного модуля представлена на рисунку 2.9.

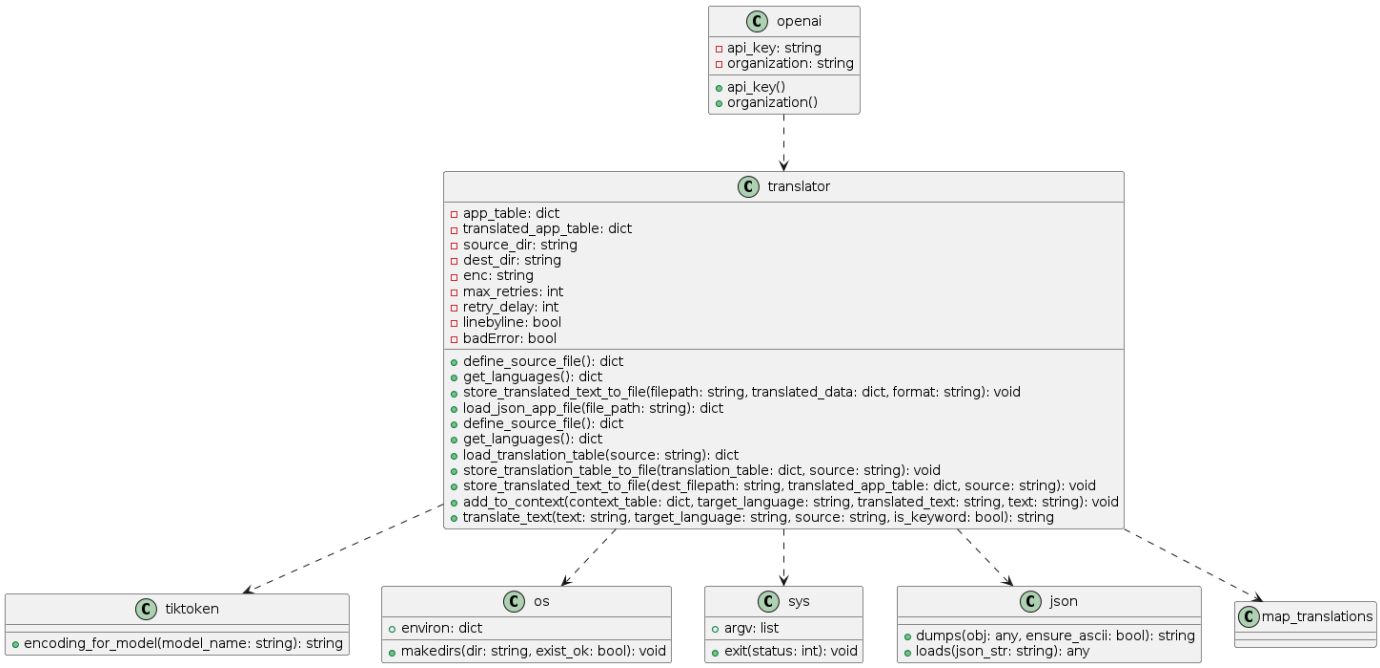


Рисунок 2.9 – UML діаграма класів

Таким чином, діаграма класів в контексті перекладу тексту нейромережею надає комплексне уявлення про структуру і взаємодію компонентів системи, що допомагає в її проектуванні, розробці і підтримці.

2.2 Вибір архітектури нейронної мережі трансформера

Архітектура трансформера складається з двох основних частин: енкодера та декодера. Енкодер обробляє вхідну послідовність, створюючи контекстні подання для кожного токена, тоді як декодер генерує вихідну послідовність, використовуючи контекстні подання з енкодера та раніше згенеровані токени вихідної послідовності. На рисунку 2.4 представлений лістинг коду нейромережі трансформера, щоб отримати уявлення про його програмну реалізацію.

```python
class Transformer(nn.Module):
    def __init__(self, src_vocab_size, trg_vocab_size):
        super().__init__()
        self.src_vocab_size = src_vocab_size
        self.trg_vocab_size = trg_vocab_size

        self.src_embedding = nn.Embedding(self.src_vocab_size, d_model)
        self.trg_embedding = nn.Embedding(self.trg_vocab_size, d_model)
        self.positional_encoder = PositionalEncoder()
        self.encoder = Encoder()
        self.decoder = Decoder()
        self.output_linear = nn.Linear(d_model, self.trg_vocab_size)
        self.softmax = nn.LogSoftmax(dim=-1)

    def forward(self, src_input, trg_input, e_mask=None, d_mask=None):
        src_input = self.src_embedding(src_input) # (B, L) => (B, L, d_model)
        trg_input = self.trg_embedding(trg_input) # (B, L) => (B, L, d_model)
        src_input = self.positional_encoder(src_input) # (B, L, d_model) => (B, L, d_model)
        trg_input = self.positional_encoder(trg_input) # (B, L, d_model) => (B, L, d_model)

        e_output = self.encoder(src_input, e_mask) # (B, L, d_model)
        d_output = self.decoder(trg_input, e_output, e_mask, d_mask) # (B, L, d_model)

        output = self.softmax(self.output_linear(d_output)) # (B, L, d_model) => # (B, L, trg_vocab_size)

        return output
```

Рисунок 2.4 – Лістинг коду структури нейронної мережі трансформер

Компоненти архітектури:

– Embedding-шари: Перетворюють вхідні токени у векторні подання певної розмірності (d_model).

– Positional Encoding: Додає інформацію про позиції токенів у послідовності, що дозволяє трансформеру враховувати порядок слів.

– Енкодер: Складається з декількох шарів, які містять механізм самоуваги та позиційні зворотні зв'язки.

– Декодер: Використовує механізм самоуваги для обробки вихідної послідовності, а також увагу до виходу енкодера.

Задача енкодера в нейронній мережі – перевести вхідні дані до проміжного багатомірного представлення. Задача декодера – трансформувати це проміжне подання до передбачення.

Кожен шар енкодера та декодера складається з механізму самоуваги та багатошарових перцептронів. Ці шари дозволяють моделі зосереджуватись

на різних частинах вхідної послідовності та навчатися складним відношенням між токенами.

Токенізація є одним з ключових кроків у підготовці текстових даних для використання в нейронних мережах, зокрема в трансформерах. Вона полягає в розбитті тексту на менші одиниці — токени, які можуть бути словами, частинами слів або навіть окремими символами. Правильна токенізація забезпечує кращу обробку тексту і розуміння його структури моделлю. Архітектура трансформера (рис. 2.5), розроблена для роботи з послідовними даними, використовує механізм самоуваги (self-attention), що дозволяє моделі ефективно визначати важливі частини вхідного тексту, встановлюючи вагові коефіцієнти для кожного токена. Це значно підвищує здатність моделі до контекстного розуміння та обробки великих обсягів текстових даних.

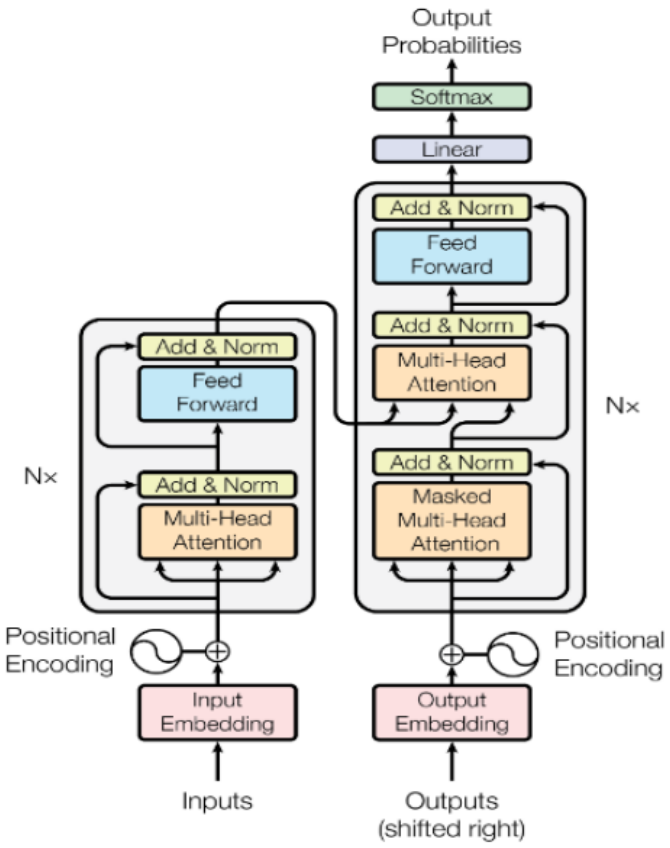


Рисунок 2.5 – Архітектура Transformer

Види токенізації:

– Word-level токенізація: Розбиває текст на окремі слова. Цей метод простий, але має недоліки, пов'язані з великою кількістю унікальних слів у словнику та проблемами з обробкою невідомих слів.

– Subword-level токенізація: Розбиває текст на частини слів або морфеми. Цей метод дозволяє краще обробляти рідкісні та невідомі слова, оскільки вони можуть бути представлені комбінацією часто вживаних частин. До популярних алгоритмів належать Byte Pair Encoding (BPE) та WordPiece.

– Character-level токенізація: Розбиває текст на окремі символи. Це дозволяє обробляти будь-які слова, включаючи неологізми та помилки, але призводить до дуже довгих послідовностей, що може ускладнити навчання моделі.

Використовувати у своїй роботі будемо Subword-level (рис. 2.6) токенізацію для досягнення більшої достовірності перекладу. Цей метод токенізації є гнучким і здатним працювати з різними мовами і текстами, забезпечуючи хорошу якість токенізації навіть для рідкісних і невідомих слів. Для токенізації в коді будемо використовувати бібліотеку sentencepiece для токенізації тексту

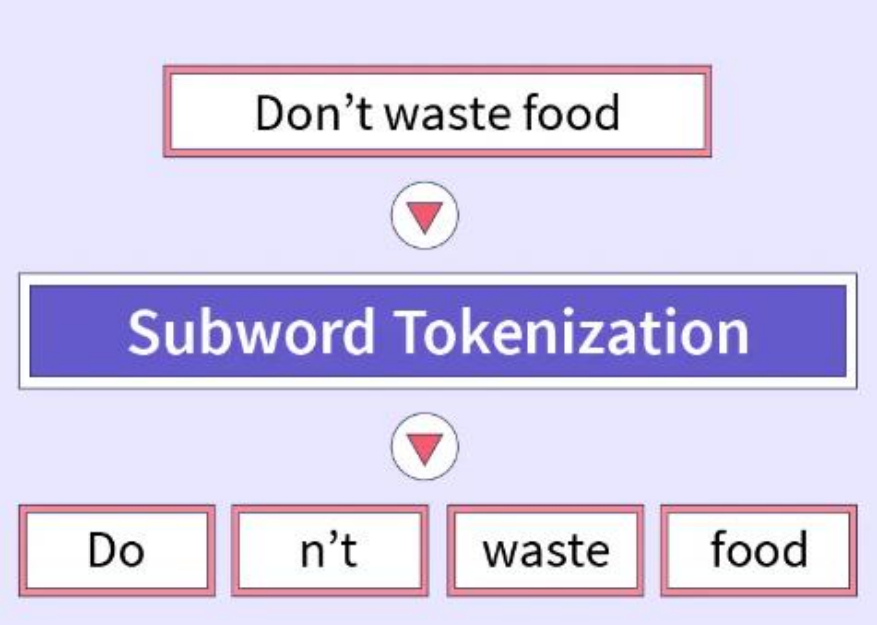


Рисунок 2.6 – Візуалізація Subword-level токенізації

Архітектура трансформера є потужним інструментом для задач машинного перекладу завдяки своїй здатності ефективно обробляти послідовності та враховувати контекст кожного слова. Використовуючи механізм самоуваги, трансформер може зосереджувати увагу на важливих частинах вхідної та вихідної послідовностей, що дозволяє досягти високої якості перекладу. Представлений код є прикладом реалізації базової моделі трансформера, яка може бути розширена та покращена для досягнення ще кращих результатів у перекладі текстів.

2.5 Висновок

У розділі було визначено структуру процесів обробки інформації у програмному модулі українсько-англійського перекладача, аргументовано вибір архітектури згорткової нейронної мережі для вирішення даної задачі. Проаналізовано процес навчання нейронних мереж трансформерів. Розроблено алгоритм функціонування програмного модуля українсько-англійського перекладача та побудовано UML діаграму класів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ УКРАЇНСЬКО-АНГЛІЙСЬКОГО ПЕРЕКЛАДАЧА НА ОСНОВІ НЕЙРОМЕРЕЖІ

3.1 Використання фреймворків та бібліотек

Для програмної реалізації процесів створення та навчання згорткової нейронної мережі було використано такі бібліотеки як Tensorflow та Tourch.

TensorFlow - це відкритий фреймворк для машинного навчання, розроблений компанією Google. Він надає широкий спектр інструментів для створення та навчання нейронних мереж, включаючи глибоке навчання.

Tensorflow [8] – це інтерфейс призначений для впровадження і розгортання великомасштабних моделей машинного навчання. Система підтримує обчислення на різних мобільних платформах, різних типах обладнання та на розподілених архітектурах різного масштабу. Єдина система, яка охоплює такий широкий спектр платформ, значно спрощує застосування машинного навчання в реальних умовах. Вона може використовуватися для реалізації великої кількості алгоритмів, включаючи навчання і тестування глибоких нейронних мереж. Ця система вже успішно застосовується в галузях комп'ютерного зору, робототехніки, обробки природної мови та інших сферах інформатики. Ми використовували TensorFlow при розробці наших архітектур. Основними перевагами TensorFlow є його гнучкість і масштабованість, що дозволяє створювати різноманітні моделі нейронних мереж для вирішення різних завдань у сфері штучного інтелекту та машинного навчання.

Основні особливості TensorFlow включають:

– Модульність: TensorFlow має модульну архітектуру, що дозволяє легко збирати різні типи штучних нейронних мереж, такі як згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN), автокодери, варіаційні автокодери та багато інших.

– Велика спільнота та підтримка: TensorFlow має велику та активну спільноту користувачів, що призводить до швидкого вирішення проблем та надання різноманітних ресурсів для навчання та розвитку.

– Вбудовані інструменти для обробки даних: TensorFlow має вбудовані інструменти для роботи з даними, включаючи завантаження, обробку та підготовку даних для навчання нейронних мереж.

– Підтримка різних пристроїв: TensorFlow може працювати на різних платформах, включаючи CPU, GPU та TPU (Tensor Processing Unit), що дозволяє ефективно використовувати різні обчислювальні ресурси для навчання моделей.

Усі ці особливості роблять TensorFlow потужним інструментом для розробки та навчання різноманітних моделей нейронних мереж для широкого спектру завдань у сфері машинного навчання та штучного інтелекту.

OpenAI [9] в контексті перекладача на основі нейромережі відіграє ключову роль у розвитку передових методів машинного перекладу та генерації тексту. Організація використовує передові алгоритми глибокого навчання та штучного інтелекту для створення мовних моделей, які можуть автоматично перекладати тексти з однієї мови на іншу.

Бібліотека torch [10] є однією з основних бібліотек для роботи з глибоким навчанням у середовищі мови програмування Python. Вона пропонує широкий спектр функцій для реалізації нейронних мереж, обробки даних, оптимізації, візуалізації та інших задач, пов'язаних з машинним навчанням і глибоким навчанням. Основні функції бібліотеки включають в себе автоматичне диференціювання, широкий вибір алгоритмів оптимізації (таких як SGD, Adam, Adagrad тощо), підтримку обчислень на графіках (GPU), різноманітні функції активації та шари нейронних мереж, а також зручний інтерфейс для роботи з даними та моделями. Бібліотека torch дозволяє ефективно реалізовувати різноманітні архітектури нейронних мереж та проводити їх навчання на великих обсягах даних.

SentencePiece [12] - це бібліотека для токенізації тексту, яка використовується в основному для систем генерації тексту на основі нейронних мереж, де розмір словника є ключовим аспектом. Ось деякі важливі аспекти та функції цієї бібліотеки:

– Unsupervised Text Tokenizer: SentencePiece є безнаглядним токенізатором тексту, що означає, що вона може працювати без великої кількості попередньо позначених даних або навчання на певних мовних корпусах.

– Subword Units: Вона реалізує підодиниці слів, такі як Byte Pair Encoding (BPE) або Unigram Language Model, що дозволяє ефективно розбивати слова на менші підодиниці для кращого розуміння тексту.

– Language Independence: SentencePiece є незалежним від мови та може бути успішно використаний для токенізації тексту на будь-якій мові.

– Детокенізація: Крім токенізації, бібліотека також надає можливість детокенізації, тобто об'єднання токенів у повноцінні слова або фрази.

– Гнучкість: SentencePiece дозволяє налаштовувати розмір словника, тип підодиниць слів та інші параметри для оптимальної токенізації тексту.

– Використання в Нейронних Мережах: Ця бібліотека широко використовується в нейронних мережах для обробки природної мови, таких як машинний переклад, генерація тексту та інші завдання, де обробка тексту є ключовою.

SentencePiece є потужним інструментом для токенізації тексту, який допомагає покращити якість та ефективність моделей обробки мовленнєвих даних у сфері штучного інтелекту.

Також інші бібліотеки для функціонування програми [11]:

– os

Бібліотека os в Python надає функції для взаємодії з операційною системою. Вона дозволяє отримувати доступ до різних функцій операційної системи, таких як робота з файлами, каталогами, змінними середовища тощо.

– sys

Модуль sys надає доступ до деяких функцій та змінних, пов'язаних з інтерпретатором Python та його середовищем. Це дозволяє отримувати інформацію про виконуване середовище та інтерпретатор Python.

– json

Модуль json в Python використовується для роботи з форматом даних JSON (JavaScript Object Notation). Він дозволяє кодувати дані в формат JSON або декодувати дані з цього формату

3.2 Програмна реалізація модуля українсько-англійського перекладача

Опишемо основні функції нейромережі трансформера.

```
class Transformer(nn.Module):

    def __init__(self, src_vocab_size, trg_vocab_size):
        super().__init__()
        self.src_vocab_size = src_vocab_size
        self.trg_vocab_size = trg_vocab_size

        self.src_embedding   =   nn.Embedding(self.src_vocab_size,
d_model)
        self.trg_embedding   =   nn.Embedding(self.trg_vocab_size,
d_model)
        self.positional_encoder = PositionalEncoder()
        self.encoder = Encoder()
        self.decoder = Decoder()
        self.output_linear          =          nn.Linear(d_model,
self.trg_vocab_size)
        self.softmax = nn.LogSoftmax(dim=-1)
```

Конструктор класу Transformer, ініціалізує параметри моделі та створює необхідні шари (Embedding, позиційний кодер, енкодер, декодер, лінійний шар та softmax).

```
forward(self, src_input, trg_input, e_mask=None, d_mask=None)
```

Метод для прямого поширення даних через модель. Виконує обчислення вхідних даних через всі шари моделі та повертає остаточний вивід.

Конструктор класу Encoder, створює шари енкодера та нормалізації:

```
def __init__(self):
        super().__init__()
        self.layers = nn.ModuleList([EncoderLayer() for i in range(num_layers)])
        self.layer_norm = LayerNormalization()
```

Метод, який пропускає вхідні дані через шари енкодування та повертає нормалізований вихід.

```
def forward(self, x, e_mask):
        for i in range(num_layers):
            x = self.layers[i](x, e_mask)
```

Конструктор класу Decoder, створює шари декодера та нормалізації.

```
def __init__(self):
        super().__init__()
        self.layers = nn.ModuleList([DecoderLayer() for i in range(num_layers)])
        self.layer_norm = LayerNormalization()
```

Метод, який пропускає вхідні дані через шари декодування та повертає нормалізований вихід.

```
def forward(self, x, e_output, e_mask, d_mask):
    for i in range(num_layers):
        x = self.layers[i](x, e_output, e_mask, d_mask)
```

Наступна частина коду описує ініціалізацію необіхдних параментрів для нейромережі:

```
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
learning_rate = 1e-4
batch_size = 80
seq_len = 200
num_heads = 8
num_layers = 6
d_model = 512
d_k = d_model // num_heads
num_epochs = 10
beam_size = 8
```

device: вибір пристрою для тренування моделі (GPU, якщо доступний, або CPU).

learning_rate: швидкість навчання для оптимізатора.

batch_size: розмір міні-пакету даних для одного кроку навчання.

seq_len: максимальна довжина послідовності (кількість токенів).

num_heads: кількість голов у багатоголовій увазі.

num_layers: кількість шарів в енкодері та декодері трансформера - 6.

d_model: розмірність векторного представлення (векторів) в моделі.

d_k: розмірність кожної голови у багатоголовій увазі (обчислюється як d_model // num_heads).

num_epochs: кількість епох для тренування моделі - 10.

beam_size: розмір бім-пошуку для генерації послідовностей.

Цей фрагмент коду завантажує модель трансформера та налаштовує оптимізатор Adam для її тренування.

```
print("Loading Transformer model & Adam optimizer...")
self.model = Transformer(src_vocab_size=len(self.src_i2w), trg_vocab_size=len(self.trg_i2w)).to(device)
self.optim = torch.optim.Adam(self.model.parameters(), lr=learning_rate)
self.best_loss = sys.float_info.max
```

«torch.optim.Adam» використовується оптимізатор Adam з бібліотеки PyTorch.

self.model.parameters(): Передає параметри моделі трансформера оптимізатору для оновлення під час тренування.

lr=learning_rate: Встановлює швидкість навчання (learning rate) для оптимізатора, значення якої визначене константою learning_rate.

Adam (Adaptive Moment Estimation) не є функцією активації нейрона. Це оптимізатор, який використовується для оновлення параметрів моделі під час тренування нейронних мереж.

Adam поєднує переваги двох інших методів оптимізації: AdaGrad та RMSProp. Він обчислює адаптивні швидкості навчання для кожного параметра, використовуючи середні значення перших моментів (середні значення градієнтів) та других моментів (квадрати градієнтів). Adam є дуже ефективним і широко використовується у тренуванні складних моделей нейронних мереж, включаючи трансформери.

У коді, який ви надіслали раніше, функція активації використовується у класі FeedFowardLayer. Там використовується функція активації ReLU (Rectified Linear Unit).

Ось відповідна частина коду:

```
class FeedFowardLayer(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear_1 = nn.Linear(d_model, d_ff, bias=True)
        self.relu = nn.ReLU()
        self.linear_2 = nn.Linear(d_ff, d_model, bias=True)
        self.dropout = nn.Dropout(drop_out_rate)
    def forward(self, x):
        x = self.relu(self.linear_1(x)) # (B, L, d_ff)
        x = self.dropout(x)
        x = self.linear_2(x) # (B, L, d_model)
        return x
```

У цій частині коду, self.relu створює об'єкт функції активації ReLU, яка використовується для активації після першого лінійного шару (self.linear_1).

Функція активації ReLU визначається наступним чином:

$ReLU(x)=max(0,x)$

Ця функція активації є дуже популярною в нейронних мережах завдяки своїй простоті та ефективності. Вона має такі переваги:

– Нелінійність: Вводить нелінійність в модель, що дозволяє нейронній мережі навчатися складним залежностям.

– Відсутність проблеми зникаючого градієнта: На відміну від функцій Sigmoid або Tanh, ReLU не насичується в позитивній частині, що дозволяє уникнути проблеми зникаючого градієнта.

Ось де в коді ця функція застосовується:

```
def forward(self, x):
    x = self.relu(self.linear_1(x)) # (B, L, d_ff)
    x = self.dropout(x)
```

```
        x = self.linear_2(x) # (B, L, d_model)
        return x
```

Тут ReLU застосовується до виходу першого лінійного шару self.linear_1(x), а потім результат передається через шар Dropout і другий лінійний шар.

У цьому конкретному коді функція активації не явно визначена, оскільки необхідні активаційні функції вбудовані в окремі шари.

Наступний код займається підготовкою даних для навчання моделі глибокого навчання, яка, ймовірно, є моделлю перекладу тексту.

Ініціалізація SentencePiece моделей:

```
src_sp = spm.SentencePieceProcessor()
trg_sp = spm.SentencePieceProcessor()
src_sp.Load(f"{SP_DIR}/{src_model_prefix}.model")
trg_sp.Load(f"{SP_DIR}/{trg_model_prefix}.model")
```

Завантажуються моделі токенізації для джерельної (src_sp) та цільової (trg_sp) мов.

Функція get_data_loader читає текстові файли з даними для джерельної та цільової мов, токенізує та доповнює (або обрізає) ці дані, створює та повертає DataLoader, який буде використовуватися для навчання.

Функції process_src та process_trg:

– process_src: Токенізує джерельний текст і додає кінцевий токен (eos_id).

– process_trg: Токенізує цільовий текст, додаючи початковий токен (sos_id) і кінцевий токен (eos_id).

Коротко кажучи, цей код відповідає за підготовку текстових даних для навчання моделі перекладу, включаючи токенізацію, доповнення або

обрізання послідовностей та створення набору даних для навчання з використанням PyTorch.

3.3 Опис набору данних

TED2020 Dataset – це набір даних, створений на основі субтитрів до виступів TED (Technology, Entertainment, Design). Він використовується в основному для задач машинного перекладу та аналізу текстів. TED Talks є відомими виступами, які охоплюють широкий спектр тем, включаючи науку, мистецтво, бізнес, та інші дисципліни. Ці виступи перекладені на багато мов, що робить TED2020 ідеальним набором даних для багатомовних задач обробки природної мови (NLP).

Основні характеристики TED2020 Dataset:

– Джерело: Субтитри до виступів TED, доступні на офіційному вебсайті TED.

– Багатомовність: Набір даних містить переклади субтитрів на багато мов, що робить його корисним для задач машинного перекладу.

– Різноманітність тем: Включає виступи на різні теми, що забезпечує різноманітний контекст і лексичний склад.

– Відкритий доступ:Зазвичай доступний для дослідників та розробників в області NLP.

TED2020 зазвичай включає наступні компоненти:

– Вхідний текст (source text): Текст субтитрів мовою оригіналу.

– Цільовий текст (target text): Текст субтитрів на мові перекладу.

– Метаінформація: Включає інформацію про виступи, такі як ідентифікатори виступів, тривалість, автори тощо.

TED2020 Dataset широко використовується для:

– Машинний переклад: Навчання та оцінка моделей перекладу, таких як трансформери.

– Багатомовний аналіз: Дослідження методів обробки багатомовних текстів.

– Навчання моделей обробки природної мови: Використання для задач сумісного навчання, навчання переносу тощо.

Даний датасет включає в собі 200 000 речень, необхідних для навчання нейромережі. Обсяг тестової вибірки складає 40 000 речень, максимальна довжина фрагменту тексту, необхідного для перекладу – 300 символів.

Проте розмір набору даних TED2020 може змінюватися в залежності від того, які саме дані включені в цей набір. Наприклад, він може містити тексти промов, метадані про доповіді (такі як дати, автори, теми тощо), аудіозаписи та відеозаписи промов.

Зазвичай набір даних TED2020 є досить обсяжним, оскільки він містить багато різних промов та презентацій, які були представлені на конференції TED у 2020 році. Цей датасет має велику кількість унікальних слів, про все ж має спільне з такими датасетами як наведено на рисунку 3.1.

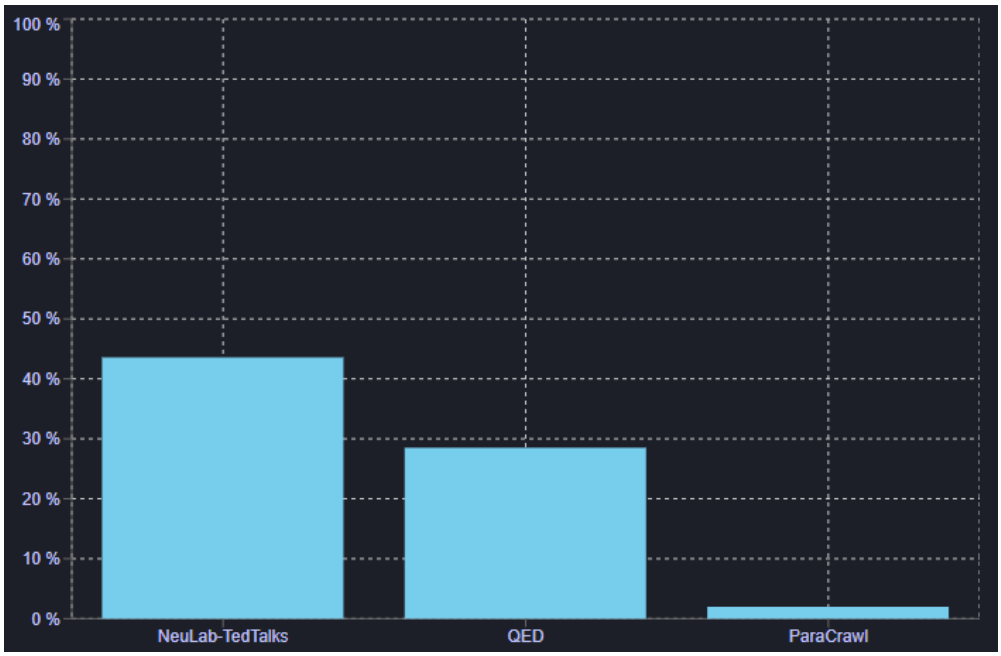


Рисунок 3.1 – Діаграма найбільш значних збігів з наборами даних

Розбиття даних на набори навчань, валідації та витримки дозволяє розробити високоточні моделі, що стосуються даних, які ви збираєте в майбутньому, а не лише даних, на яких навчалася модель. Навчаючи нейромережу, перевіряючи та тестуючи у наборі виплат, ми отмираэмо реальне відчуття того, наскільки достовырними будуть результати моделі, що призводить до кращих рішень та більшої впевненості у достовірності моделі.

Для цього коду набір даних буде складатися з 200000 речень, які будуть використовуватися для навчання моделі перекладу. Давайте детально опишемо структуру цього набору даних та етапи його обробки.

1. Структура набору даних:

Файли з текстами:

– Джерельна мова (Source Language): Файл, який містить 200000 речень на вихідній мові.

– Цільова мова (Target Language): Файл, який містить відповідні 200000 речень на цільовій мові.

2. Обробка набору даних

Читання текстових файлів:

– Код читає два файли, один для джерельної мови (source) і один для цільової мови (target).

– Кожен файл містить 200000 рядків, де кожен рядок є окремим реченням.

Токенізація текстів:

– Речення з файлу джерельної мови токенізуються за допомогою моделі SentencePiece (src_sp).

– До кожного токенізованого речення додається кінцевий токен (eos_id).

– Токенізовані речення доповнюються або обрізаються до фіксованої довжини (seq_len).

– Речення з файлу цільової мови токенізуються за допомогою моделі SentencePiece (trg_sp).

– Вхідний цільовий список, що починається з початкового токена (sos_id).

– Вихідний цільовий список, що закінчується кінцевим токеном (eos_id).

– Обидва списки доповнюються або обрізаються до фіксованої довжини (seq_len).

Створення набору даних:

– Токенізовані та підготовлені списки джерельних, вхідних цільових та вихідних цільових даних зберігаються у вигляді тензорів PyTorch.

– Вони об'єднуються у спеціальний клас CustomDataset, який наслідує клас Dataset з PyTorch.

– Створення завантажувача даних (DataLoader):

– Використовується DataLoader з PyTorch для створення батчів даних, які будуть використовуватися під час навчання.

– Дані розбиваються на батчі розміром, визначеним параметром batch_size.

– Дані перемішуються перед кожною епохою навчання (shuffle=True).

Структура класу CustomDataset:

```
import torch
from torch.utils.data import Dataset
class CustomDataset(Dataset):
    def __init__(self, src_list, input_trg_list, output_trg_list):
        self.src_data = torch.LongTensor(src_list)
        self.input_trg_data = torch.LongTensor(input_trg_list)
        self.output_trg_data = torch.LongTensor(output_trg_list)

        assert self.src_data.shape[0] == self.input_trg_data.shape[0]
== self.output_trg_data.shape[0], "Shapes are not aligned"
    def __getitem__(self, idx):
```

```
        return      self.src_data[idx],      self.input_trg_data[idx],
self.output_trg_data[idx]
    def __len__(self):
        return self.src_data.shape[0]
```

Цей клас дозволяє легко і ефективно працювати з набором даних під час навчання моделі.

3.4 Висновок

У цьому розділі описується можливість використання об'єктно-орієнтованої мови програмування Python та середовища розробки Intellij IDEA на високому рівні. Бібліотека Tensorflow була використана для реалізації процесу побудови та навчання згорткової нейронної мережі. Бібліотека OpenAI використовувалася для обробки тексту. Бібліотека torch для роботи з глибоким навчанням. Збереження реалізовано в форматі файлів txt або json для того, щоб далі працювати з даними або зручно обробляти для збереження в базу даних. В результаті був розроблений програмний модуль для перекладу українського тексту на англійську мову на основі згорткової нейронної мережі.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО МОДУЛЯ УКРАЇЕСЬКО-АНГЛІЙСЬКОГО ПЕРЕКЛАДАЧА

Для вибору потрібного для перекладу необхідно відкрити вхідний файл в піпці “translations”. Після цього нам необхідно ввести тексту для перекладу, якщо файл формату txt – вводимо текст в пустому файлі, якщо json – вводимо текст для перекладу в полі «"text":», як показано на рисунку 4.1.

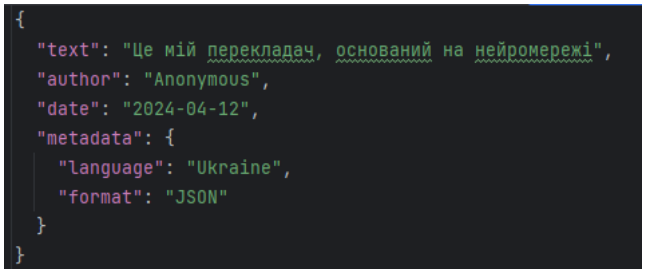


Рисунок 4.1 – Введення тексту для перекладу

Після введення тексту потрібно ввести в терміналі команду для перекладу, яка складається з виклику основного коду програми, вхідної та вихідної дерикторії файлу:

– Для .json:

```
python jsontranslator.py
C:\Users\efaefa3\IdeaProjects\autotranslate-ai\translations\input.json
C:\Users\efaefa3\IdeaProjects\autotranslate-ai\translated\json
```

– Для .txt:

```
python txttranslator.py
C:\Users\efaefa3\IdeaProjects\autotranslate-ai\translations\input.json
C:\Users\efaefa3\IdeaProjects\autotranslate-ai\translated\json
```

Після закінчиться виконання коду, ми отримаємо в папці “translated” наш вихідний файл з перекладом (рис. 4.2)

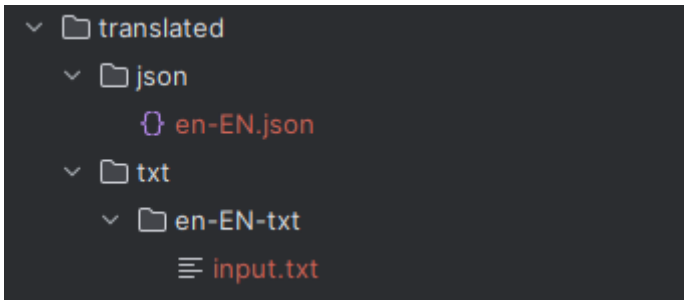


Рисунок 4.2 – Відображення вихідного файлу

Відкривши вихідні файли, ми можемо побачити перекладений текст, як показано на рисунку 4.3.

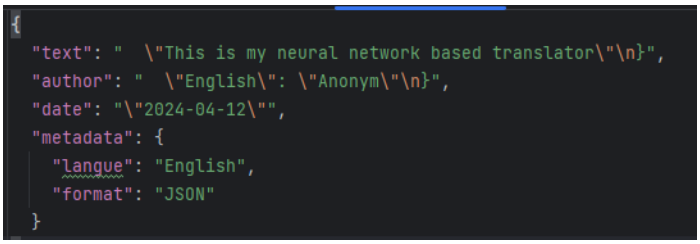


Рисунок 4.3 – Результат перекладу

Розроблений програмний модуль українсько-англійського перекладача було навчено за допомогою корпусу двомовних текстів.

Результати тестування запропонованого програмного модуля наведено у табл. 4.1.

Таблиця.4.1 – Результати випробувань запропонованого програмного модуля для перекладу з української на англійську мову.

	Результат перекладу	
На вході речення англійською мовою (загальна кількість речень P=100)	Вірно True – **TP=98**	Невірно False– **FP=2**

Оцінимо ключовий показник якості створеного програмного модуля, використовуючи дані з таблиці 4.1.

Достовірність.

$$Accuracy = \frac{TP}{TP + FP} \times 100\% = \frac{98}{100} \times 100\% = 98\%$$

Для доведення досягнення мети дослідження потрібно порівняти ці показники якості із показниками аналога (див табл 4.2). Таке порівняння наведено у табл. 4.2.

Таблиця.4.2 – Порівняння показників якості запропонованого програмного модуля із показниками аналога (Google Translate)

	Аналог (Google Translate)	Запропонований програмний модуль
Достовірність	95 %	98 %

Із табл. 4.2 видно, що розроблена програма має на 3% (98% проти 95%) достовірність ніж програма-аналог. Тобто мета роботи досягнута – достовірність перекладу з української мови на англійську підвищена у розробленому програмному засобі на 3%.

Висновок

Проведено тестування програмного модуля українсько-англійського перекладача, що ґрунтується на технології Deep Learning. Аналіз результатів функціонування розробленої програми показав, що її достовірність перевищує достовірність аналога на 3% (98% проти 95%). Іншими словами, мета роботи була досягнута: достовірність перекладу з української на англійську мову підвищена у створеному програмному продукті на 3%.

ВИСНОВКИ

У дослідженні розглянуто завдання перекладу текстів з української на англійську мову, використовуючи технологію Deep Learning. Під час аналізу предметної області досліджено постановку цього завдання, розглянуто різні методи перекладу текстів та обґрунтовано використання нейронних мереж, зокрема трансформерів, для цієї мети. Також проведено порівняльний аналіз існуючих програмних реалізацій перекладу тексту та обрано підходи, спрямовані на підвищення достовірності перекладу з української на англійську мову.

Були розроблені структура процесів програмного модуля перекладу текстів та алгоритм роботи нейронної мережі трансформера для перекладу, аргументовано вибір архітектури нейронної мережі трансформер для вирішення даної задачі. Проаналізовано процеси роботи нейронних мереж трансформерів та розроблено відповідну структуру нейронної мережі трансформера. Також розроблено алгоритм функціонування програмного модуля перекладу текстів та UML діаграму класів.

Обґрунтовано вибір архітектури та бібліотек для реалізації процесів створення та навчання моделі, де використано бібліотеки Tensorflow та Torch. В результаті було створено програмний модуль для українсько-англійського перекладу на основі нейронної мережі трансформер.

Було проведено тестування програмного модуля українсько-англійського перекладача, розробленого з використанням технології Deep Learning. Аналіз результатів показав, що розроблена програма має вищу достовірність виявлення медичних масок на обличчі на 3% порівняно з аналогами. Отже, мета дослідження була досягнута – програмний засіб виявлення медичних масок на обличчі демонструє підвищену достовірність на 3%.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Електронний ресурс]. – Режим доступу:// https://support.google.com/translate/answer/6350850?hl=uk&co=GENIE.Platform%3DDesktop#:~:text=У%20додатку%20Google%20Перекладач%20можна,також%20доступний%20у%20веб-переглядачі.&text=На%20сторінці%20Google%20Перекладача%20можна,веб-сайти%20понад%20100%20мовами.

2. S. Li., W.Deng. Deep Facial Expression Recognition: A Survey. IEEE Transactions onAffective Computing, 04. 2018. P. 1–10.

3. Гудфеллоу Я. Глубокое обучение / пер. с анг. А. А. Слинкина. 2-е изд., испр. М.: ДМК Пресс, 2018. 652 с.: цв. ил.

4. Linguise - Машинний переклад [Електронний ресурс] - Режим доступу до ресурсу: https://www.linguise.com/uk/блог/керівництво/що-таке-нейронний-машинний-переклад/

5. Загальний опис та реалізація Word2Vec за допомогою PyTorch [Електронний ресурс]. – Режим доступу://https://habr.com/ru/articles/801807/

6. Згорткова нейронна мережа, частина 2: навчання алгоритмом зворотного розповсюдження помилки [Електронний ресурс] - Режим доступу до ресурсу: https://habr.com/ru/articles/348028/

7. Python [Електронний ресурс]. – Режим доступу: https://www.python.org/.

8. M. Abadi., A. Agarwal., P. Barham., E. Brevdo., Z. Chen., C. Citro., G. S. Corrado. Tensorflow: Large-scale machine learning on heterogeneous systems:Tensorflow.org , 2015. URL:http://download.tensorflow. org/paper/whitepaper2015.pdf (Last accessed 24.04. 2021)

9. OpenAI [Електронний ресурс]. – Режим доступу://https://cases.media/en/article/openai-laboratoriya-doslidzhen-shtuchnogo-intelektu-ta-yiyi-rol-u-metavsesviti

10. PyTorch Documentation [Електронний ресурс]. – Режим доступу://https://pytorch.org/docs/stable/index.html

11. Стандартна бібліотека Python [Електронний ресурс]. – Режим доступу://https://docs.python.org/uk/3/library/index.html

12. SentencePiece [Електронний ресурс]. – Режим доступу://https://github.com/google/sentencepiece

13. Нейромережеві моделі та технології обчислювального інтелекту. Нейрокомп`ютери. Частина І : навчальний посібник / О. К. Колесницький, В. І. Месюра. – Вінниця : ВНТУ, 2021. – 66 с. (3 авт.арк./1,5 авт.арк.)

14. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» / Укладачі А.А.Яровий, О.В.Сілагін, І.В.Богач. – Вінниця. ВНТУ, 2022. – 47 с.

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний модуль українсько-англійського перекладача на основі нейронної мережі

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 91,19% Схожість 8,81%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Кузьменко О.Ю.

Керівник роботи Колесницький О.К.

Додаток Б (обов'язковий)

Лістинг програми

```
from tqdm import tqdm
from constants import *
from custom_data import *
from transformer import *
from data_structure import *
from torch import nn

import torch
import sys, os
import numpy as np
import argparse
import datetime
import copy
import heapq
import sentencepiece as spm

#TRAIN
class Manager():
    def __init__(self, is_train=True, ckpt_name=None):
        # Load vocabs
        print("Loading vocabs...")
        self.src_i2w = {}
        self.trg_i2w = {}

        with open(f"{SP_DIR}/{src_model_prefix}.vocab") as f:
            lines = f.readlines()
        for i, line in enumerate(lines):
            word = line.strip().split('\t')[0]
            self.src_i2w[i] = word

        with open(f"{SP_DIR}/{trg_model_prefix}.vocab") as f:
            lines = f.readlines()
        for i, line in enumerate(lines):
            word = line.strip().split('\t')[0]
            self.trg_i2w[i] = word

        print(f"The size of src vocab is {len(self.src_i2w)} and that of
trg vocab is {len(self.trg_i2w)}.")

        # Load Transformer model & Adam optimizer
        print("Loading Transformer model & Adam optimizer...")
        self.model = Transformer(src_vocab_size=len(self.src_i2w),
trg_vocab_size=len(self.trg_i2w)).to(device)
....
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ УКРАЇНСЬКО-АНГЛІЙСЬКОГО ПЕРЕКЛАДАЧА НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ»

Виконав: студент 4 курсу,групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Кузьменко О.Ю.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф.КН
Колесницький О.К.
(прізвище та ініціали)
« 07 » 06 2024 р.

Вінниця ВНТУ - 2024 рік

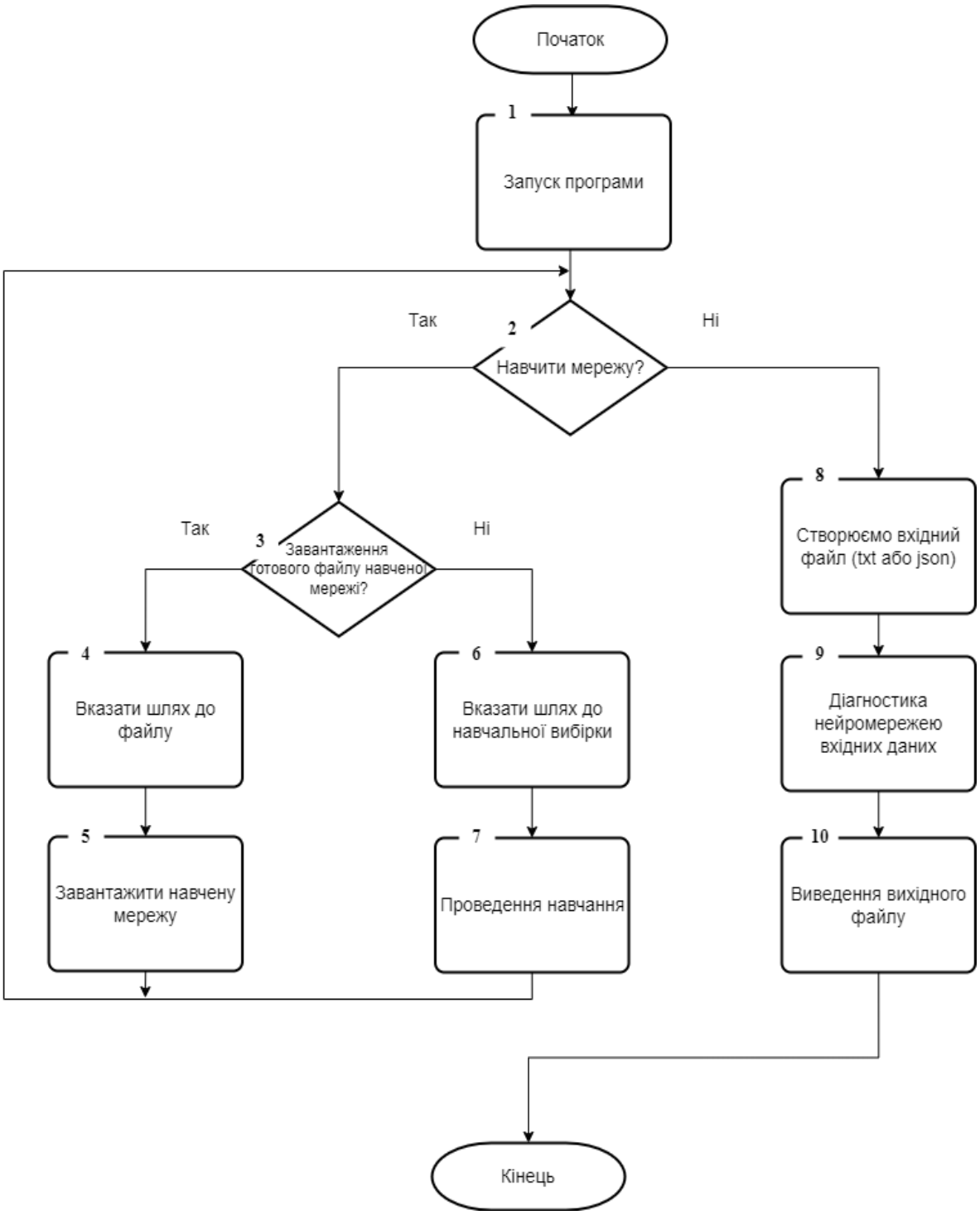


Рисунок В.1 – Схема алгоритму роботи програми

Рисунок В.2 – Процеси обробки інформації при українсько-англійському перекладі

```python
class Transformer(nn.Module):
    def __init__(self, src_vocab_size, trg_vocab_size):
        super().__init__()
        self.src_vocab_size = src_vocab_size
        self.trg_vocab_size = trg_vocab_size

        self.src_embedding = nn.Embedding(self.src_vocab_size, d_model)
        self.trg_embedding = nn.Embedding(self.trg_vocab_size, d_model)
        self.positional_encoder = PositionalEncoder()
        self.encoder = Encoder()
        self.decoder = Decoder()
        self.output_linear = nn.Linear(d_model, self.trg_vocab_size)
        self.softmax = nn.LogSoftmax(dim=-1)

    def forward(self, src_input, trg_input, e_mask=None, d_mask=None):
        src_input = self.src_embedding(src_input) # (B, L) => (B, L, d_model)
        trg_input = self.trg_embedding(trg_input) # (B, L) => (B, L, d_model)
        src_input = self.positional_encoder(src_input) # (B, L, d_model) => (B, L, d_model)
        trg_input = self.positional_encoder(trg_input) # (B, L, d_model) => (B, L, d_model)

        e_output = self.encoder(src_input, e_mask) # (B, L, d_model)
        d_output = self.decoder(trg_input, e_output, e_mask, d_mask) # (B, L, d_model)

        output = self.softmax(self.output_linear(d_output)) # (B, L, d_model) => # (B, L, trg_vocab_size)

        return output
```

Рисунок В.3 – Лістинг коду структури нейронної мережі трансформер

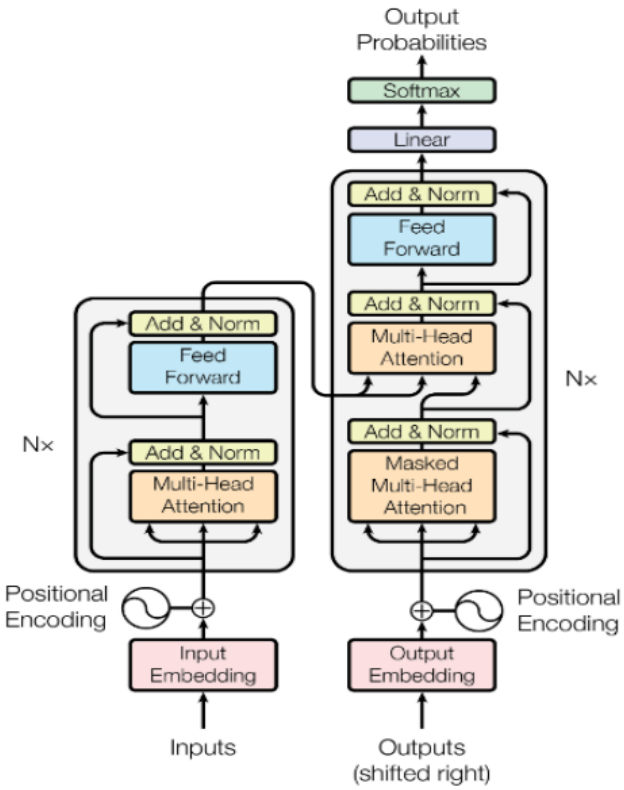


Рисунок В.4 – Архітектура Transformer

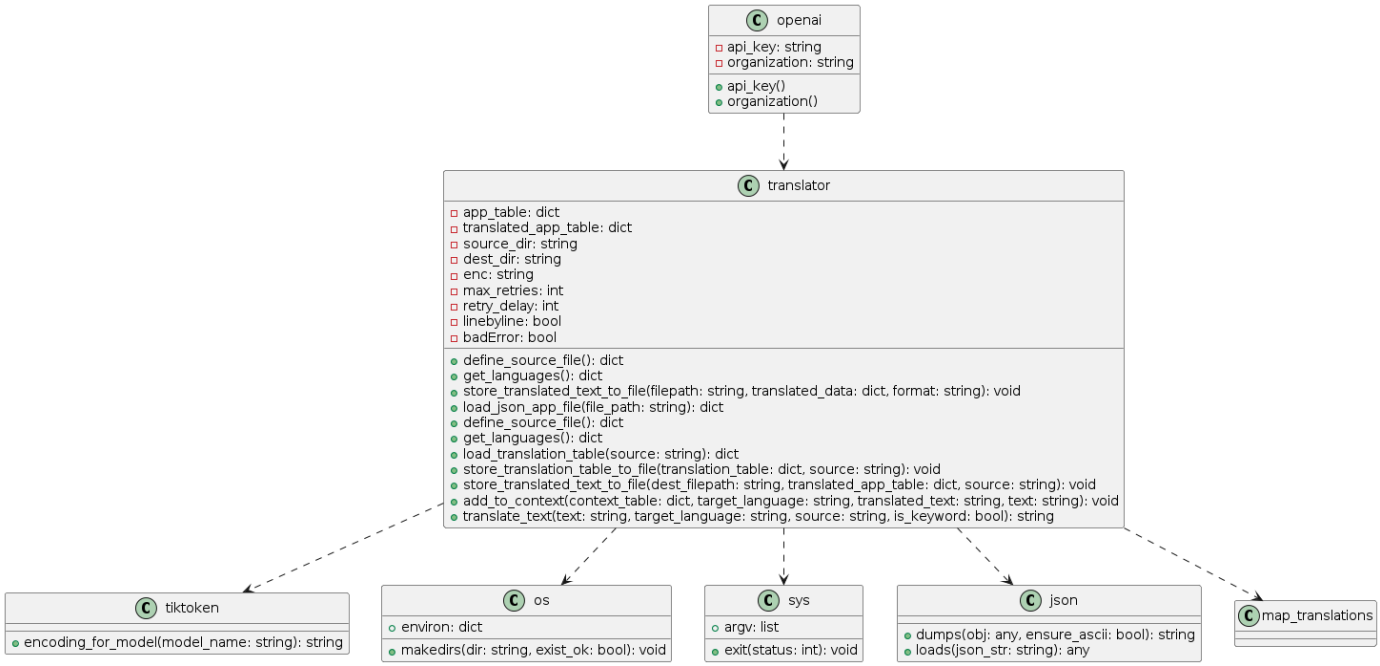


Рисунок В.5 – UML діаграма класів

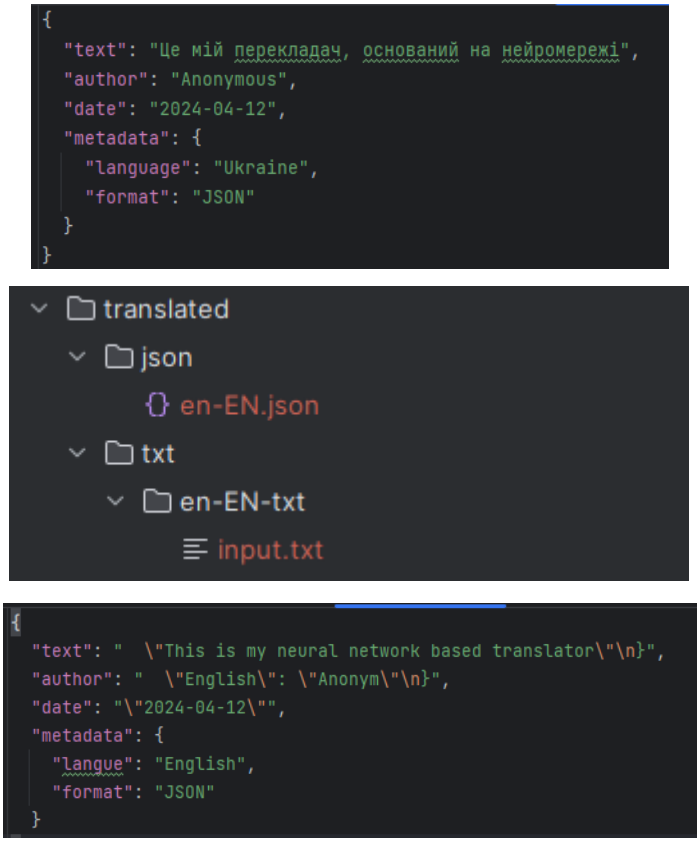


Рисунок В.6 – Результати програми

Додаток Г (довідниковий)

Інструкція користувача

На комп’ютері має бути встановлений Python 3.8+ версії та пакети, які необхідні для запуску програми (якщо спробувати запустити програму, то пакети, які не встановлені, будуть вказані в терміналі – командою pip install name встановлюємо її).

Тренування:

Для тренування потрібні вхідні дані (тобто набір двомовних пар речень української та відповідної їм англійської мови), їх потрібно помістити в папку DATA_DIR(див. рис.Г.1), які знаходяться в кореневому каталозі проекту (якщо папки немає, то потрібно створити).

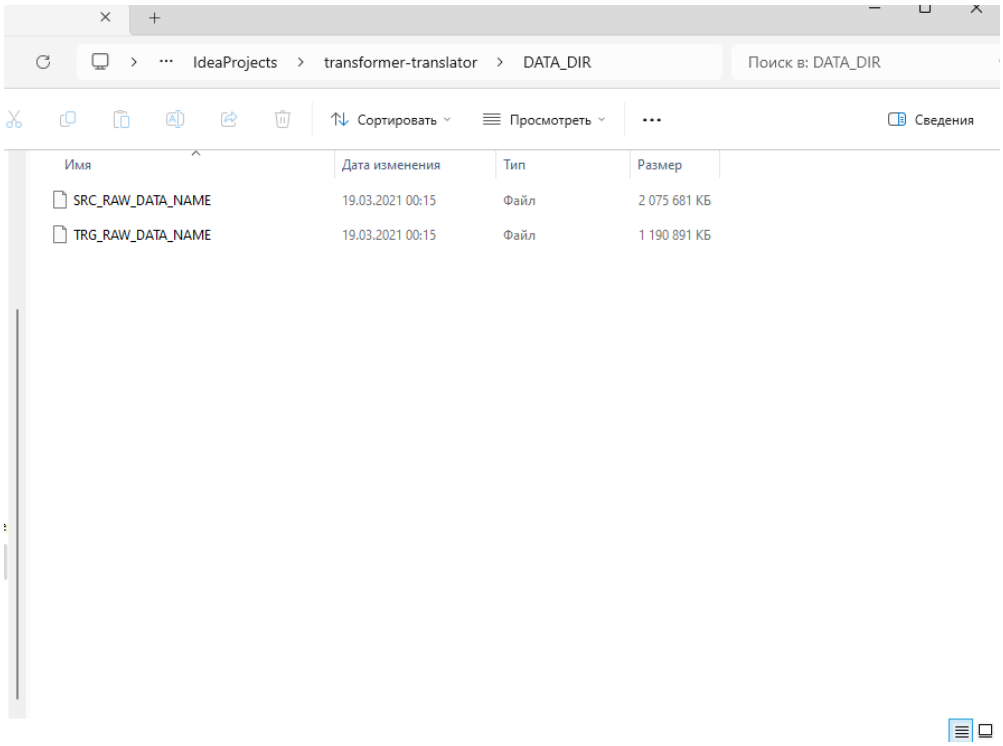
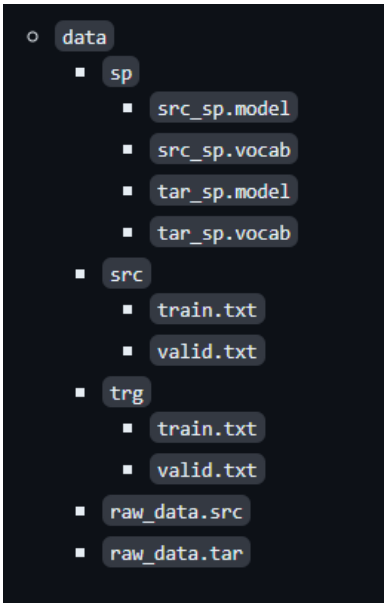


Рисунок Г.1 – Створення папки для навчальних даних

- Перейти до каталогу, що містить sentencepiece_train.py
- Ввести за запустити команду py src/sentencepiece_train.py в терміналі

```
(venv) PS C:\Users\efaefa3\IdeaProjects\transformer-translator> py src/sentencepiece_train.py
```

Після чого створиться необхідний каталок файлів та папок:



Після чого вводимо цю команду для навчання мережі python src/main.py --mode='train'

Після навчання мережі вводимо команду python src/main.py --mode='inference' --ckpt_name=CHECKPOINT_NAME --input=INPUT_TEXT --decode=DECODING_STRATEGY

--input: це послідовність введення, яку ви хочете перекласти.