**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ТАРАСА ШЕВЧЕНКА**
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

«До захисту допущено»
Завідувач кафедри
В. М. Терещенко _____
(підпис)

«___» _____20___ р.

# Кваліфікаційна робота
**на здобуття ступеня бакалавра**

за спеціальністю 122 Комп'ютерні науки
на тему:
**МАШИННИЙ ПЕРЕКЛАД ВІРШІВ**

Виконав студент 4 курсу
Томаш Станіслав Миколайович _____
(підпис)

Науковий керівник:
асистент
Федорус Олексій Мстиславович _____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021

# РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел (27 найменувань). Робота містить 20 рисунків та 6 таблиць. Загальний обсяг роботи становить 43 сторінки.

МАШИННЕ НАВЧАННЯ, ГЛИБИННЕ НАВЧАННЯ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, LSTM, МАШИННИЙ ПЕРЕКЛАД, ENCODER-DECODER, ПЕРЕКЛАД ВІРШІВ.

В даній роботі запропоновано алгоритм перекладу віршів зі збереженням ритму з використанням технологій машинного навчання та побудовано моделі необхідні для реалізації алгоритму. Також описано і сформовано набори даних для навчання відповідних моделей.

**Об'єктом** роботи є вірші, написані російською мовою, а також їх переклади на англійську мову.

**Предмет** роботи – переклад віршів з російської мови на англійську.

**Мета** роботи – побудова алгоритму для автоматичного перекладу віршованого тексту з російської на англійську мову зі збереженням ритму.

**Методи** дослідження – методи глибинного навчання, побудова моделей нейронних мереж.

**Технології** розробки: мова програмування Python, інтерактивні оболонки IPython та Jupyter Notebook, бібліотеки TensorFlow Keras, Numpy, Matplotlib, BeautifulSoup та набір інструментів NVIDIA CUDA Toolkit.

**Результат** виконання роботи – побудовані та натреновані нейронні мережі для генерації транскрипції слів, перекладу речень з російської на англійську мову, визначення розміру вірша, а також описано та реалізовано алгоритм для перекладу віршів з російської на англійську мову.

# ЗМІСТ

# ВСТУП

**Оцінка сучасного стану об’єкта розробки.** На сьогоднішній день існує доволі велика кількість систем машинного перекладу текстів. Найбільш відомими з них є онлайн-перекладачі на кшталт Google Translate чи Amazon Translate. Ці перекладачі використовують свою велику базу користувачів для того, щоб постійно вдосконалювати переклади, які вони пропонують. Всі сучасні системи застосовують машинне навчання та обробку величезної кількості даних для генерації перекладу. При цьому доволі мало досліджень було проведено в галузі машинного перекладу художніх текстів, а особливо віршів. Задача перекладу віршованого тексту є дуже складною, оскільки включає в себе доволі великий обсяг комп’ютерної креативності при дуже строгих обмеженнях, заданих оригінальним текстом.

**Актуальність роботи та підстави для її виконання.** Задачі, що включають в себе комп’ютерну креативність знаходяться на передньому фронті нових розробок в сфері машинного навчання. Задачі обробки природньої мови виникають в багатьох сферах сучасного життя: ми щодня користуємось голосовими асистентами чи чат-ботами, перекладаємо слова онлайн, або читаємо генеровані машиною субтитри в відеозаписах. Задача машинного перекладу віршів може бути особливо корисною в цілях легшого обміну культурними здобутками між різними національностями, оскільки робота перекладачів зазвичай займає доволі багато часу, і при цьому зазвичай перекладаються тільки найбільш відомі широкому загалу твори.

**Мета й завдання роботи.** Метою роботи є побудова алгоритму для автоматичного перекладу віршованого тексту з російської на англійську мову зі збереженням ритму. Для досягнення цієї мети поставлено такі завдання:

- Аналіз існуючих досліджень у сфері машинного перекладу текстів, віршів, а також генерації віршів.
- Проектування моделей машинного навчання, необхідних для побудови алгоритму.
- Формування наборів даних для навчання змодельованих моделей.
- Візуалізація та аналіз сформованих наборів даних.
- Імплементація та тренування запропонованих моделей.
- Оцінка якості роботи фінальних моделей.

**Об'єкт, методи й засоби розроблення.** Об’єктом дослідження є вірші, написані російською мовою, а також їх переклади на англійську мову. Предмет роботи – переклад віршів з російської мови на англійську. **Методи** дослідження – методи глибинного навчання, побудова моделей нейронних мереж.

В якості мови програмування було обрано мову Python та інтерактивне оболонку IPython з використанням Jupyter Notebook. Для роботи з масивами чисел використано бібліотеку Numpy, для побудови графіків – бібліотеку Matplotlib, для веб-скрейпінгу – бібліотеку BeautifulSoup, для побудови та тренування нейронних мереж – пакет TensorFlow Keras. Навчання нейронних мереж відбувалось з використанням NVIDIA CUDA Toolkit для навчання на графічному процесорі. Конфігурація комп’ютера для навчання: ЦП Intel Core i7-9750H @ 2.60ГГц, ОЗУ 32ГБ, ГП NVIDIA GeForce RTX 2060 @ 6ГБ.

**Можливі сфери застосування.** Запропонований алгоритм та архітектури моделей можуть бути застосовані для побудови перекладів віршів та можуть бути основою для подальших досліджень в даній предметній області.

# РОЗДІЛ 1.

# ОСНОВНІ МЕТОДИ ОБРОБКИ ПРИРОДНОЇ МОВИ

Обробка природної мови – це розділ комп’ютерних наук, направлений на надання комп’ютеру можливості розуміти текст чи людську мову таким самим чином, як їх розуміють люди. [1]

## 1.1. Початок розвитку обробки природньої мови

Першим успішним дослідженням в сфері обробки природної мови можна вважати Джорджтаунський експеримент [2]. Даний експеримент являв собою машинний переклад 60 російських речень на англійську мову. Алгоритм був побудований на словнику з 250 слів та 6 граматичних правилах. Успіх презентації сприяв збільшенню фінансування досліджень в напрямку розвитку машинного перекладу, але алгоритми на основі словників та правил не виправдали надій вчених та урядів, які фінансували дослідження, тому вже в кінці 60-х років фінансування було зменшено, аж до появи статистичних методів обробки природної мови в 80-х роках XX століття. Найуспішнішими розробками в цей період можна вважати системи на кшталт SHRDLU чи ELIZA.

Система SHRDLU оперувала в рамках обмеженого «світу», що складався з простих геометричних об’єктів – «блоків» та дій, що можна було з ними виконувати. [3] Програма могла «розуміти» запити користувача на виконання дії з певним об’єктом чи об’єктами та рапортувати про стан «світу». Розуміння запитів користувача було побудовано на семантичних та граматичних правилах занесених в систему. Система була написана на мові LISP та спеціально розробленій для задачі мові PLANNER.

ELIZA – система, розроблена Джозефом Вейзенбаумом в 1964-1966 роках, яка симулювала діалог між людиною та машиною. [4] Її алгоритмічну основу складали паттерни, з якими вона порівнювала речення користувачів, та видавала заготовлені відповіді, використовуючи інформацію з фраз людини. Поведінка ELIZA визначалася заведеними в систему «скриптами», найбільш відомим з яких є DOCTOR, який симулює розмову з психотерапевтом (рис.1.1).

```
ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

Рисунок 1.1. Лістинг «розмови» користувача з системою ELIZA

Бачимо, що системи, засновані на спробі прямолінійно описати правила побудови та аналізу фраз природної мови для розуміння комп’ютером не приводило до побудови систем, що могли самостійно аналізувати інформацію, задану користувачем. Натомість, поведінка комп’ютера повністю визначалась і обмежувалась паттернами, які задав автор програми. Такий підхід може бути ефективним і використовуватись і в наш час, але тільки в ситуаціях коли компетенція комп’ютера строго обмежена до розуміння чи виконання невеликої кількості дій. Наприклад, системи на основі порівняння паттернів та розбору регулярних виразів використовуються в різноманітних чат-ботах технічної підтримки.

## 1.2. Статистичні методи обробки природної мови

Використання статистичних методів для обробки природної мови стало можливим перш за все, через збільшення обчислювальної потужності комп'ютерної техніки, доступної дослідникам [2]. Для використання статистичних методів потрібна велика кількість попередньо розмічених даних. Це дозволяє використати припущення, що спостереження, які справджуються на розміченому наборі, також будуть спостерігатись і на всій іншій множині можливих речень чи фраз мови. Набір текстів природної мови називається корпусом. [5] На сьогоднішній день існують відкриті корпуси як і для загального аналізу та задач (наприклад Global Web-Based English [6]), так і для специфічних задач, які використовують тексти певної предметної області (наприклад Medical Corpus, побудований на медичних текстах і призначений для задач в області аналізу медичних текстів [6]).

Простою задачею, на якій можна продемонструвати використання статистичних методів обробки природної мови є задача тегування частин мови. [8] В цій задачі класифікатор повинен визначити до якої частини мови відноситься кожне слово в реченні. Позначимо через $T$ всі можливі частини мови. Тоді для ізольованого слова встановлена частина мови:

$$\mathrm{t} = \arg\max_{\mathrm{t}\in\mathrm{T}} p(t \mid w)$$

Такий підхід можна поширити на тегування всіх слів в реченні. Отримаємо:

$$t = \arg\max_{t_{1,n}\in T} \prod_{i=1}^{n} p(t_i \mid w_i)$$

Алгоритм, заснований на використанні даної формули досягає точності близько 90%. [8] Втім, його проблема полягає в тому, що він не враховує існування слів, які можуть в різних ситуаціях належати до різних частин мови і завжди ставить у відповідність одному слову одну й ту саму частину мови. Так, наприклад, в реченні *"the can rusts"* слово *can* буде позначено як іменник, хоча в даному контексті воно є дієсловом. Для того щоб враховувати дану особливість мови введемо поняття контексту слова. Контекстом розміру $m$ слова $w_i$ в реченні вважатимемо $m$ слів, що зустрічаються в реченні перед ним, тобто $context(w_i) = \overline{w_{i-m} w_{i-m+1} \dots w_{i-1}}$ . Тоді ми можемо розширити це поняття на частини мови та порахувати в корпусі ймовірність того що частина мови $t_i$ зустрічається у кожному з можливих контекстів. З урахуванням цього можемо побудувати нову формулу:

$$t = \arg\max_{t_{1,n} \in T} \prod_{i=1}^{n} p_{context}(t_i \mid context(t_i)) \cdot p(t_i \mid w_i)$$

Системи, що використовують таку формулу для тегування частин мови досягають точності в 96-97% вже на довжині контексту, що дорівнює одиниці.[8]

Статистичні методи обробки природної мови використовуються не тільки для задач класифікації, а й для більш специфічних для даної галузі задач, наприклад, для генерації текстів. Для генерації текстів за допомогою статистичних методів використовуються ланцюги Маркова. Вони застосовують ідею контексту, описану вище. Ланцюг Маркова – ймовірнісний автомат, в якому стани – контексти, а ймовірність переходів визначається як:

$$P(\overline{w_1 w_2 \dots w_m}, \overline{w_1 \dots w_m w}) = p(w \mid \overline{w_1 w_2 \dots w_m}),$$

де $p$ позначає ймовірність появи слова $w$ в контексті $\overline{w_1 w_2 \dots w_m}$. Також існують варіації генерації тексту на основі ланцюгів Маркова, що використовують модель мови, побудовану не на рівні слів, а на рівні символів.[9] В наступних розділах даної роботи також розглядатиметься використання ланцюгів Маркова та споріднених алгоритмів для генерації віршованого тексту.

## 1.3. Використання глибинного навчання для обробки природної мови

З подальшим розвитком обчислювальної техніки та збільшенням обчислювальних потужностей для обробки природної мови почали використовувати нейронні мережі. [2] Найчастіше в системах глибинного навчання для обробки природної мови використовуються рекурентні нейронні мережі, оскільки вони найкраще можуть емулювати поняття контексту, описане в попередньому підрозділі. Використання нейронних мереж принципово відрізняється від традиційних статистичних чи граматично-семантичних тим, що алгоритми, засновані на ньому націлені на розв’язання комплексних задач обробки природної мови в цілому, а не їх окремих підзадач.

# РОЗДІЛ 2.
# СТАН РОЗРОБОК В ОБЛАСТІ МАШИННОГО ПЕРЕКЛАДУ ВІРШІВ

## 2.1. Автоматичний переклад віршів

Дослідження в області машинного перекладу віршів майже не проводились раніше. Одною з небагатьох робіт є стаття команди дослідників компанії Google Дмітрія Гензеля, Якоба Узкорейта та Франца Оха «*"Poetic" Statistical Machine Translation: Rhyme and Meter*». [10] Метод, запропонований в роботі використовує статистичні моделі для побудови перекладів. Системою генерується $n$ найбільш вірогідних перекладів тексту вірша (строфи) і з них обирається найбільш «поетичний». «Поетичність» визначається за допомогою порівняння результуючого тексту з обмеженнями ритму, рими та форми обраного жанру вірша (наприклад, хоку чи сонета). В роботі використовується мовна пара французької та англійської мов. Результат досягнутий розробленої системою – 12 зі 109 перекладених строф задовольняли обмеження обраного жанру і при цьому зберігали зміст. Цей результат є доволі хорошим, враховуючи складність поставленої задачі.

## 2.2. Автоматична генерація віршів

Враховуючи відсутність значимих досліджень та результатів в області автоматичного перекладу віршів доцільно розглянути дослідження в спорідненій сфері – машинній генерації віршів.

Одною з перших спроб генерації віршів, що одночасно задовольнятимуть умови збереження ритму та рими, а також наявності якого-небудь змісту та граматичної коректності була робота Хісара Манурунга «*An evolutionary algorithm approach to poetry generation*». [11] Дана робота розглядає генерацію вірша як процес стохастичного пошуку. В

запропонованому алгоритмі для пошуку на першому кроці випадковим чином генеруються текстові послідовності. На кожному наступному кроці визначаються послідовності які найбільше підходять під задані критерії збереження ритму, рими та відповідності заданій темі. Ці послідовності комбінуються попарно між собою, а також генерують нові шляхом мутації. При цьому також застосовуються функції, які перебудовують послідовності слів в граматично правильні речення. З нової популяції знову обираються найкращі кандидати, поки відповідність критеріям не стане задовільною. Даний процес схожий на процес еволюції та називається генетичним алгоритмом. Проте, результат отриманий автором все ще продукує вірші, які задовольняють здебільшого тільки критерії ритму та граматичної коректності. Результуючі вірші зазвичай є здебільшого повторами заданих системі слів. Наприклад, вірш, побудований системою зі вхідними словами «*lion*» та «*waist*»: [11]

*A lion, it dwells in a waste.*
*A lion, it dwells in a waste.*
*A waste will be rare.*
*Its head will be rare.*
*Its waist, that is small, will be rare.*

Принципово інший підхід, побудований на використанні шаблонів з існуючих віршів використовує система описана в роботі Симона Колтона, Якоба Гудвіна та Тоні Вейла «*Full-FACE Poetry Generation*». [12] Для генерації використовується корпус з існуючих англомовних віршів (для побудови шаблонів), а також відкриті лексичні та орфоепічні словники. В процесі генерації система бере відкриту статтю новин та визначає її ключові слова і фрази. Після цього за допомогою евристичних алгоритмів вона намагається

знаходити слова-синоніми, що трансформують фрази в такі, що підпадають під заданий ритмічний шаблон. При цьому результат генерації також втрачає зміст і задовольняє здебільшого тільки критерії ритмічності і граматичної коректності: [12]

*the wiry militant arm of a doorman*
*a white middle-eastern man, like a snowball*
*spaceship-foreign embassies*
*foreign embassies, each like a stranger*

*an impersonal suvarnabhumi international airport*
*a white middle-eastern man, like the surface of a porcelain*
*the sturdy design of a bangkok post*

Ще один принципово інший підхід, описаний в роботі Джека Хопкінса та Доува Кіели «*Automatically Generating Rhythmic Verse with Neural Networks*» [13], використовує рекурентні нейронні мережі для генерації англомовних сонетів. Модель мови, що використовується в роботі побудована не на словах чи літерах, а на позначенні звуків у словах. Це дозволяє одразу бачити фонетичні особливості тексту, без необхідності вводити моделі, що визначатимуть ці особливості у словах. Така модель генерує сонети, що повністю задовольняють умову ритмічності, але також і мають деякий сенс, оскільки мережа намагається повторювати деякі послідовності, на яких була навчена, то й присутній і певний спільний сенс між вхідним корпусом та результатом. Приклад результату роботи описаної системи значно більш схожий на справжню поезію ніж в попередніх розглянутих роботах: [13]

*The crow crooked on more beautiful and free,*

*He journeyed off into the quarter sea.*
*his radiant ribs girdled empty and very –*
*least beautiful as dignified to see.*

Автори даної роботи також провели дослідження на групі добровольців, яке показало, що 54.8% учасників вважали тексти генеровані системою – справжніми віршами, написаними людьми. При цьому 51.4% учасників вважало справжні вірші дійсно написаними людьми. Отже можна зробити висновок, що генеровані в ході експерименту вірші є невідрізнимими від віршів такої ж форми, написаних живими авторами.

# РОЗДІЛ 3.
# ФОРМУВАННЯ ТА АНАЛІЗ НАБОРУ ДАНИХ

В даній роботі було прийнято рішення досліджувати переклади віршів з російської на англійську мову. З врахуванням даної інформації в даному розділі описуються набори даних які використовуються в роботі.

## 3.1. Embedding-вектори для слів російської та англійської мов

Embedding-вектор – це вектор дійсних чисел, який кодує значення певного слова. [14] З математичної точки зору набір embedding-векторів являє собою опис відношення вкладення з множини слів мови на векторний простір векторів з певної підмножини $\mathbb{R}^n$, де $n$ – розмірність embedding-вектору. Це відношення побудоване так, щоб слова, близькі за змістом перетворювались в близькі вектори. Embedding-вектори будуються за допомогою аналізу великої кількості природніх текстів. Ідея побудови базується на тому, що схожі за змістом слова будуть виконувати схожу функцію в схожих реченнях. В роботі використовуються попередньо натреновані набори векторів для слів англійської [15] та російської [16] мов. Детальний опис використаних наборів подано в таб.3.1.

| Мова | Джерело текстів | Розмірність векторів | Кількість слів в наборі даних для тренування векторів (млн.) | Кількість різних слів (тис.) |
|---|---|---|---|---|
| Російська | Twitter | 300 | 810 | 3100 |
| Російська | Wikipedia | 300 | 386 | 1500 |
| Англійська | Twitter | 200 | 27000 | 1200 |
| Англійська | Wikipedia | 300 | 6000 | 400 |

Таблиця 3.1. Опис наборів використаних embedding-векторів

## 3.2. Паралельний корпус віршів мовної пари російська-англійська

Корпус сформований з доступних у відкритому доступі в мережі Інтернет перекладів російських віршів на англійську мову. Джерелом перекладів виступив сайт RuVerses [17]. Для отримання тексту з сайту було написано веб-скрейпер на мові Python з використанням технології BeautifulSoup [18]. Після отримання пар текстів проводилось їх фільтрування. В першу чергу відфільтровувались тексти, в яких не співпадала кількість рядків в оригіналі та перекладі. Після цього методом перегляду корпусу було відкинуто переклади які не відповідали критерію ритмічності чи не зберігали зміст оригіналу. Фінальний варіант корпусу було збережено в форматі JSON-масиву, в якому кожним елементом є об'єкт, що включає в себе назву вірша російською мовою, назву вірша в перекладі, масив рядків вірша російською мовою та масив рядків вірша англійською мовою. В отриманому наборі присутні вірші 81 автора. Розподіл віршів за авторами наведено на рис.3.1. Найкоротші вірші в наборі складаються всього з двох рядків, наприклад вірш Анни Ахматової «*Молитесь на ночь, чтобы вам…*»:

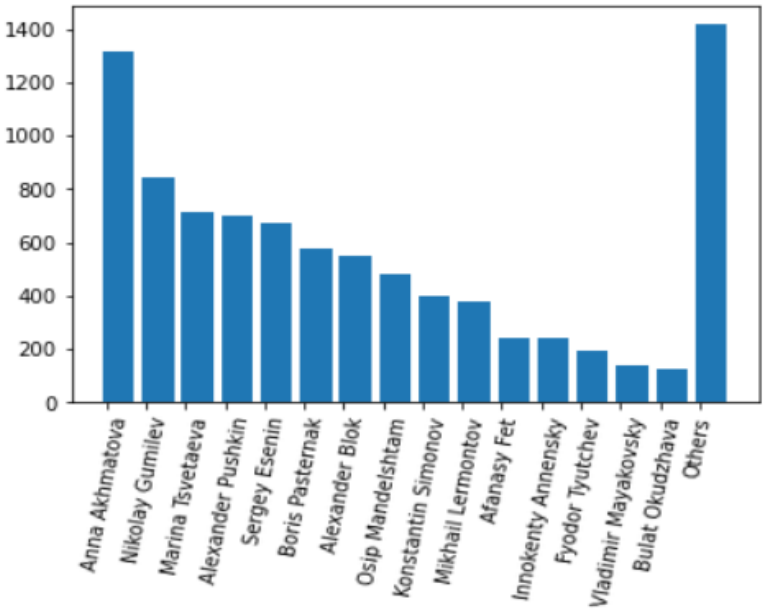


Рисунок 3.1. Розподіл віршів в наборі за авторами

*Молитесь на ночь, чтобы вам*

*Вдруг не проснуться знаменитым.*

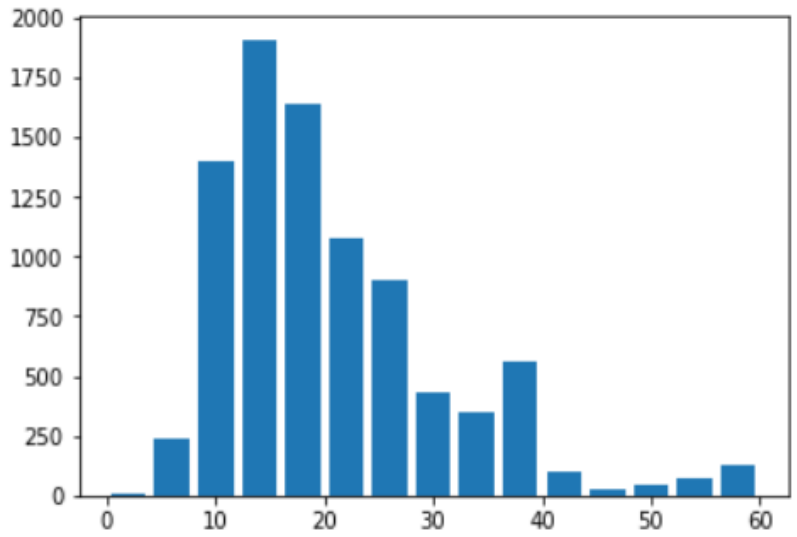


Рисунок 3.2. Розподіл віршів в наборі за кількістю рядків

Найдовший вірш в наборі – «*Двенадцать*» Олександра Блока складається з 336 рядків. Загальний розподіл віршів в наборі за кількістю рядків в них наведено на рис.3.2.

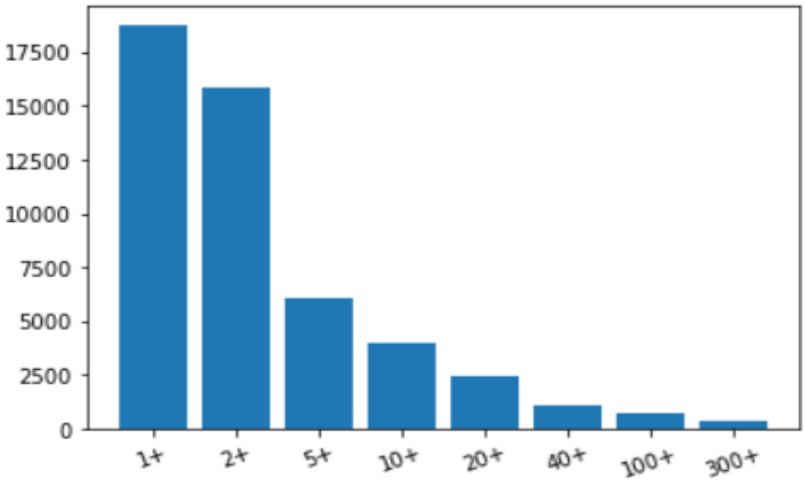


Рисунок 3.3. Розподіл російських слів в наборі за кількістю їх повторів

Ще одною важливою характеристикою корпусу є кількість різних слів у віршах та в корпусі в цілому. Всього в корпусі зустрічається 49262 різних слова, а загальна кількість слів в віршах-оригіналах – 831350. Серед російських слів в оригіналах найчастіше зустрічається слово-сполучник «*и*», яке присутнє в корпусі 41858 раз. Серед значимих слів найчастіше зустрічається слово «*я*» – 16734 рази. Загальний розподіл кількості російських слів від кількості їхніх повторів в корпусі наведено на рис.3.3. Можна відмітити, що 18760 слів зустрічається в корпусі тільки один раз, що становить близько 38% від всіх слів в корпусі. Такий розподіл значно ускладнює подальшу роботу з корпусом під час

навчання, але дана проблема є специфічною для поставленої задачі, оскільки автори віршів мають тенденцію до використання рідковживаних слів чи навіть неологізмів. Загальна кількість слів в англійських перекладах дещо більша – 1166455. Причиною цього може бути те, що в середньому в англійських словах менше складів, ніж в російських словах і для досягненя оригінальної віршової форми потрібно використати більше слів. Всього в корпусі перекладів зустрічається 26716 різних слів. Найчастіше зустрічається артикль «*the*» – 67431 раз. Серед значимих слів найчастіше зустрічається слово «*you*» – 15685 раз. Цікаво відмітити, що його російський відповідник, слово «*ты*» зустрічається в корпусі тільки 7437 раз, що більш ніж вдвічі менше. Загальний розподіл кількості англійських слів від кількості їхніх повторів в корпусі подано на рис.3.4. Знову можемо спостерігати велику кількість слів, що зустрічаються тільки раз – 7622, що становить 28% від всіх слів в корпусі.

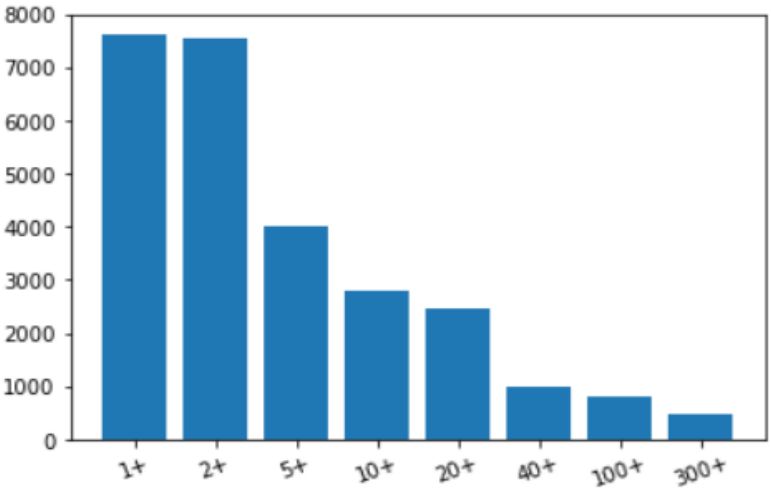


Рисунок 3.4. Розподіл англійських слів в наборі за кількістю їх повторів

## 3.3. Паралельний корпус текстів довільного змісту для мовної пари російська-анлійська

Окрім паралельного корпусу в області віршів в роботі також використовується паралельний корпус текстів довільного змісту для початкового грубого тренування перекладача. В якості такого корпусу використовується ParaCrawl Corpus [19]. Цей корпус сформований за

допомогою відкритих даних з перекладених веб-сайтів та документів в мережі Інтернет. Автори корпусу також стверджують про використання просунутих метрик для фільтрування корпусу від машинно-перекладеного тексту, чи тексту з неякісним перекладом.

### 3.4. Словники транскрипцій для російської та англійської мови

В роботі також використовуються словники транскрипцій слів у фонетичні коди ARPABET [20]. Коди ARPABET дозволяють записувати символи міжнародної фонетичної абетки (International Phonetic Alphabet, IPA) у вигляді послідовностей ASCII символів. Так наприклад звуку *tʃ*, що зазвичай відповідає літері «*ч*» російської та української мови, в ARPABET відповідатиме код «*CH*». Також в нотації ARPABET в коди, що відповідають голосним звукам включається наголошеність звуків. Цифра 0 в кінці коду відповідає ненаголошеному звуку, цифра 1 – головному наголосі в слові, більші цифри – побічним наголосам слова. Саме ця особливість кодів ARPABET робить їх найбільш привабливими для даної роботи, оскільки це дозволить зручніше працювати з ритмічними характеристиками тексту. В роботі використано словники з проекту CMU Sphinx Університету Карнегі [21, 22]. Використаний російський словник включає в себе 533668 різних слів, використаний англійський словник включає 120452 різних слів. Варто відмітити, що 6466 російських слів серед тих, що зустрічаються в оригінальних російських віршах та 7519 англійських слів з перекладів з корпусу віршів відсутні в словниках, що спонукає до побудови моделі для генерації транскрипцій слів, яких немає в словниках.

# РОЗДІЛ 4.

# ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ПЕРЕКЛАДУ ВІРШІВ

## 4.1. Модель для генерації транскрипції слів

Для задачі генерації транскрипції слів, будемо вважати слово послідовністю літер, а транскрипцію – послідовністю кодів ARPABET. Задача перетворення одної послідовності на іншу є задачею seq2seq навчання. Першим кроком в побудові моделі є перевід літер та звуків в числові значення. Найпростіший підхід до цього – використання one-hot encoding. При використанні цього способу кожному символу ставиться у відповідність єдине число. Другим підходом є використання для кодування літер embedding-векторів. В цьому підході кожній літері початково призначається вектор з випадкових чисел. Його використання можна обґрунтувати тим, що в процесі навчання моделі вона зможе привести матрицю embedding-векторів до стану, в якому символам які виконують схожі функції відповідають близькі вектори. В цьому підрозділі буде наведено результати навчання моделей як з використанням one-hot encoding, так і з використанням embedding-векторів.

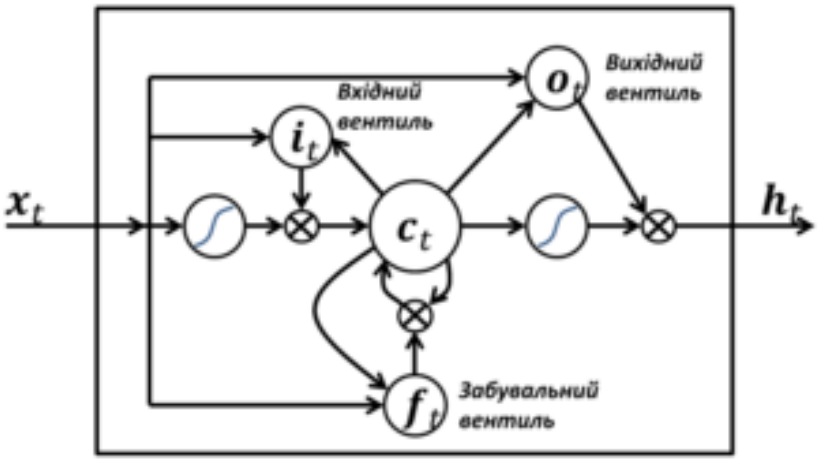


Рисунок 4.1. Схема клітини мережі LSTM

Після кодування літер та звуків перетворення послідовності кодів літер в коди відбувається за допомогою рекурентної нейронної мережі. В даній роботі в якості рекурентної нейронної мережі використовується мережа

довгої короткочасної пам’яті (Long Short-Term Memory, LSTM). [23] Клітина даної мережі складається з вхідного вентилю, вихідного вентилю, та забувального вентилю, з’єднаних так, як показано на рис.4.1. Величини $c_t, h_t$ є прихованими станами клітини. Величинами які оптимізуються при навчанні LSTM є матриці $W$ та $U$, які позначають ваги вхідних та рекурентних зв’язків відповідно. Отже, остаточно змінні клітини LSTM та функції переходу при обробці мережею з $h$ клітин елементу послідовності $x_t$, що є вектором розмірності $d$: [24]

$x_t \in \mathbb{R}^d$ – вхідний елемент,
$f_t \in \mathbb{R}^h$ – вектор активації забувального вентиля,
$i_t \in \mathbb{R}^h$ – вектор активації вхідного вентиля,
$o_t \in \mathbb{R}^h$ – вектор активації вихідного вентиля,
$h_t \in \mathbb{R}^h$ – вектор прихованого стану (вихідний вектор),
$c_t \in \mathbb{R}^h$ – вектор внутрішнього прихованого стану,
$W \in \mathbb{R}^{h \times d}$ – матриці ваги вхідних зв'язків для кожного вентиля,
$U \in \mathbb{R}^{h \times d}$ – матриці ваги рекурентних зв'язків для кожного вентиля,
$b \in \mathbb{R}^h$ – bias-вектори для кожного вентиля

$$\begin{cases} f_t = S\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t = S(W_i x_t + U_i h_{t-1} + b_i) \\ o_t = S(W_o x_t + U_i h_{t-1} + b_o) \\ c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_c x_t + U_c h_{t-1} + b_c) \\ h_t = o_t \circ tanh(c_t) \end{cases}$$

Символом $S$ в системі вище позначено функцію сигмоїди, що визначається за формулою:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

Функція гіперболічного тангенсу $tanh$ визначається за наступною формулою:

$$tanh\, x = \frac{sinh\, x}{cosh\, x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Символом $\circ$ позначається добуток Адамара – бінарний оператор для матриць однакової розмірності, який повертає матрицю такої самої розмірності, елементами якої є добутки відповідних елементів вхідних матриць. Формула обрахунку добутку Адамара:

$$\begin{pmatrix} A_{11} & \cdots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nm} \end{pmatrix} \circ \begin{pmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nm} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} & \cdots & A_{1m}B_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1}B_{1m} & \cdots & A_{nm}B_{nm} \end{pmatrix}$$

Або більш формально для елементу добутку Адамара:

$$(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$$

Матриці ваг в LSTM-мережі можна навчати за допомогою оптимізаційних алгоритмів, на кшталт градієнтного спуску, якщо навчання відбувається на наборі розмічених даних. В побудові мережі для генерації транскрипції в даній роботі використовується оптимізатор Adam, що є одним з видів стохастичного градієнтного спуску. Оптимізатор оптимізує значення ваги $w$ для мінімізації значення функції втрат $L$. Також на кожному кроці обраховуються допоміжні значення $m$ та $v$, в формулах для яких враховуються коефіцієнти «забування» для градієнту та моменту градієнту $\beta_1, \beta_2$. В наведених далі формулах нотація $w^{(t)}$ позначає значення змінної на $t$-му кроці оптимізації. Формули для ітеративного перерахунку ваги за допомогою Adam: [25]

$$\begin{cases} m_w^{(t+1)} = \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)} \\ v_w^{(t+1)} = \beta_2 v_w^{(t)} + (1-\beta_2)\left(\nabla_w L^{(t)}\right)^2 \\ w^{(t+1)} = w^{(t)} - \eta \dfrac{\dfrac{m_w^{(t+1)}}{1-\beta_1^{t+1}}}{\sqrt{\dfrac{v_w^{(t+1)}}{1-\beta_2^{t+1}}} + \epsilon} \end{cases}$$

Також варто зазначити, що LSTM може вивчати тільки паттерни послідовностей в порядку появи елементів в послідовності. Проте, звук в транскрипції який відповідає літері може залежати не тільки від попередніх літер, а й від наступних. Наприклад слово «*ape*» транскрибується в «*EY1 P*», а слово «*add*» в «*AE1 D*». Бачимо, що літера *a* відповідає різним звукам, не дивлячись на те, що знаходиться в одному місці в словах. Для того щоб обробляти такі ситуації в запропонованій моделі застосовується двонаправлений (Bidirectional) LSTM шар. Bidirectional LSTM шар являє собою дві LSTM мережі, в одну з яких послідовність передається в звичайному порядку, а в іншу – в зворотному. На виході шару отримуємо матрицю виходів шару після обробки кожного елементу вхідної послідовності.

Останнім кроком, що залишився для побудови мережі генерації транскрипції є перетворення вихідної матриці Bidirectional LSTM в символи транскрипції. Для цього використовується повнозв’язний шар з кількістю нейронів, що дорівнює кількості можливих різних символів транскрипції. Цей шар повинен повернути матрицю, в якій на j-му місці в і-му рядку знаходиться ймовірність того, що на і-му місці в транскрипції знаходиться символ з номером j.

Фактично поставлена задача генерації послідовності являє собою задачу класифікації для кожного елементу генерованої послідовності. Множина класів в задачі – множина можливих символів транскрипції. Для оптимізації параметрів нейронної мережі в задачі багатокласової класифікації використаємо функцію категоричної крос-ентропії:

$$L = \frac{-\sum_{i=0}^{N}\left(\log\left(\frac{e^{s_{ip_i}}}{\sum_{j\in C} e^{s_{ij}}}\right)\right)}{N}$$

В наведеній функції $C$ – множина можливих класів (в нашому випадку – символів транскрипції), $s_{ij}$ – передбачена нейронною мережею ймовірність того, що елемент $i$ відноситься до класу $j$, $p_i$ – клас, якому має належати елемент $i$, тобто символ, що стоїть на і-му місці в транскрипції. Схема моделі для генерації транскрипцій російських слів з використанням embedding-векторів наведена на рис.4.2. Схема моделі з використанням one-hot кодування

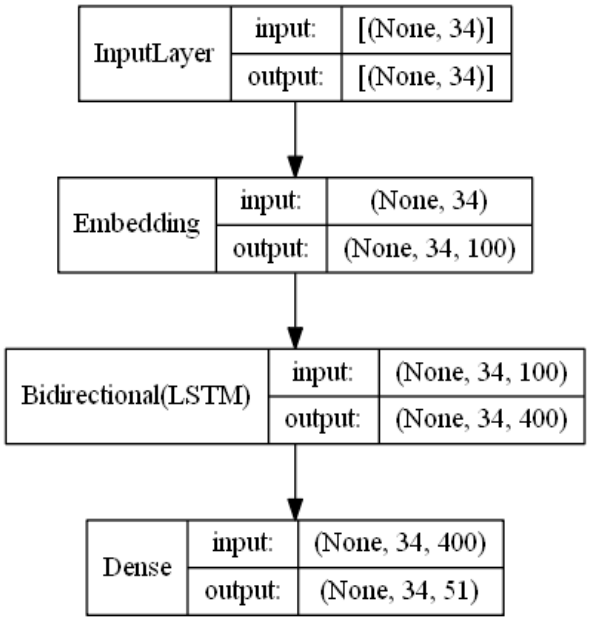


Рисунок 4.2. Схема нейронної мережі для генерації транскрипцій з використанням embedding-векторів

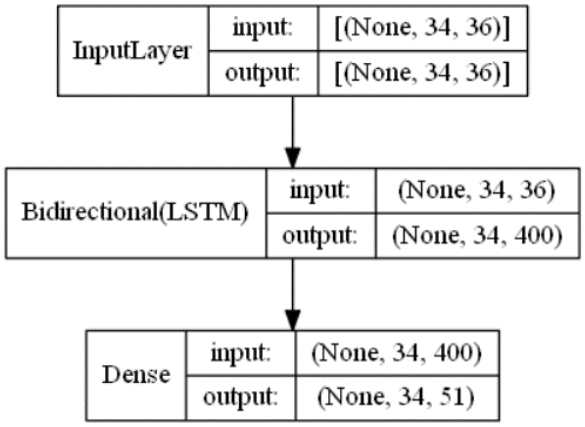


Рисунок 4.3. Схема нейронної мережі для генерації транскрипцій з використанням one-hot кодування

наведена на рис.4.3. Число 34 на схемах відповідає за максимальну довжину слова та транскрипції в словнику, на якому відбувається тренування моделі. Число 36 позначає кількість різних літер в словах Число 51 позначає кількість різних символів, можливих в транскрипції. Загальна кількість параметрів в моделі з embedding-векторами становить 505651, а в моделі з one-hot кодуванням – 399651. Графіки навчання обох моделей на російському словнику наведені на рис.4.4-4.7.

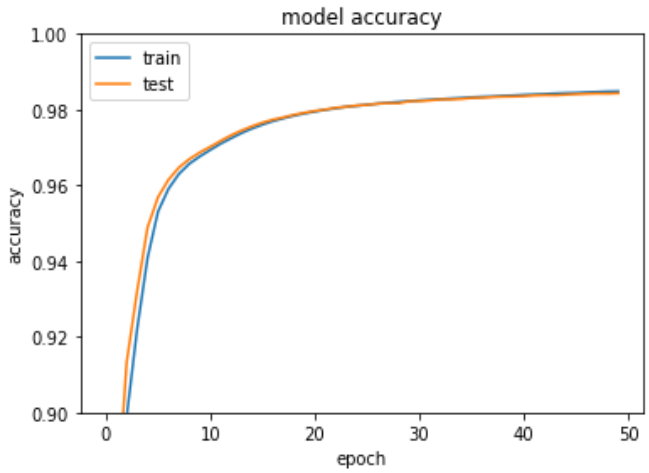


Рисунок 4.4. Графік зміни точності при навчанні мережі з one-hot кодуванням

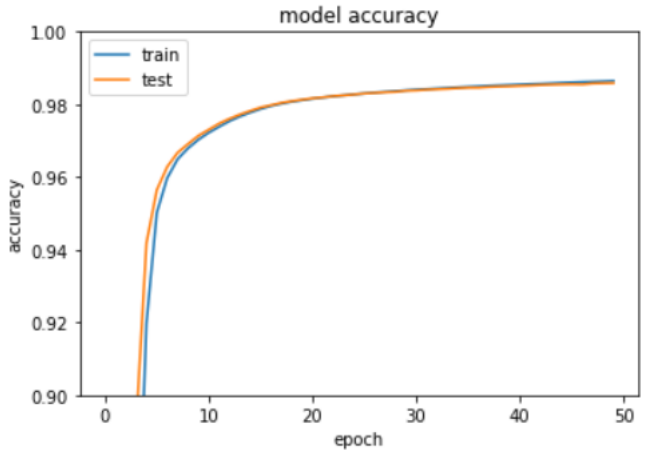


Рисунок 4.5. Графік зміни точності при навчанні мережі з embedding-векторами

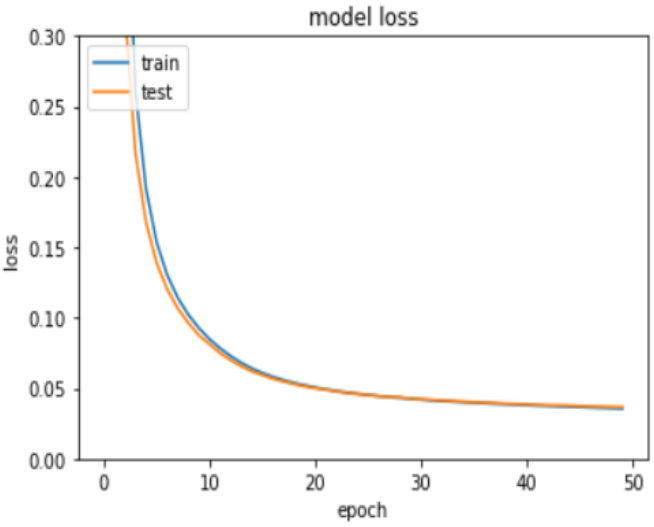


Рисунок 4.6. Графік зміни значення функції втрат для мережі з one-hot кодуванням

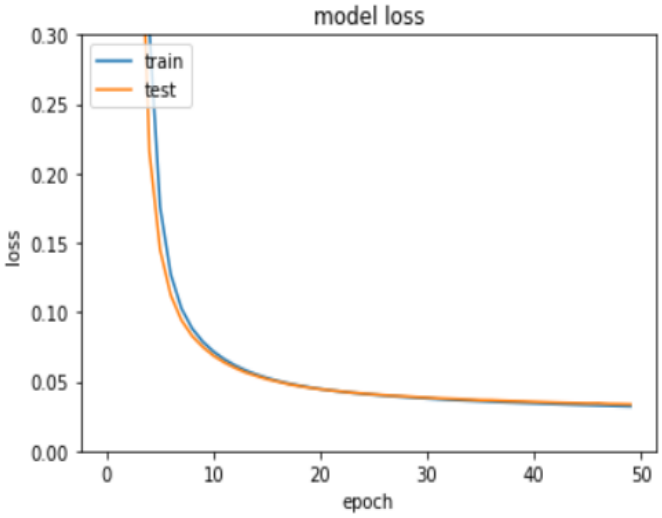


Рисунок 4.7. Графік зміни значення функції втрат для мережі з embedding-векторами

За 50 епох навчання отримана функція втрат для моделі з one-hot кодуванням стала дорівнювати 0.0356, а точність генерації 0.9848. Для моделі з embedding-векторами ці величини становлять відповідно 0.0322 та 0.9864, що свідчить про її вищу якість. Також, дивлячись на графіки, можемо зробити висновок, що моделі не зіткнулись з проблемою перенавчання, оскільки метрики на тренувальному та тестовому наборах майже не відрізняються. В таб.4.1. наведено транскрипції деяких слів, згенеровані обома моделями, а також правильні транскрипції даних слів:

Таблиця 4.1. Приклади транскрибування російських слів натренованими моделями

| Слово | One-hot pred. | Embedding pred. | Правильна транс. |
|---|---|---|---|
| заморозь | Z A0 M O0 R O1 ZJ | Z A0 M O0 R O1 ZJ | Z A0 M O0 R O1 ZJ |
| всползая | V S P O1 L Z A0 J A0 | V S P O1 L Z A0 J A0 | V S P O0 L Z A1 J A0 |
| просевая | P R O0 SJ E0 V A1 J A0 | P R O0 SJ E0 V A0 J A0 | P R O0 SJ E0 V A1 J A0 |
| застывши | Z A0 S T Y1 V SH I0 | Z A0 S T Y0 V SH I0 | Z A0 S T Y1 V SH I0 |
| звонкозвучной | Z V O0 N K O0 Z V U1 CH N O0 J | Z V O0 N K O0 Z V U1 CH N O0 J | Z V O0 N K O0 Z V U1 CH N O0 J |
| островитяне | O0 S T R O0 VJ I1 TJ A0 NJ E0 | O0 S T R O0 VJ I0 TJ A1 NJ E0 | O0 S T R O0 VJ I0 TJ A1 NJ E0 |

З таблиці бачимо, що обидві натреновані мережі добре справляються з визначенням звуків в транскрипції російських слів. Найчастішою є помилка з неправильним розставленням наголосів. Тренування цих самих моделей на англійських словах та транскрипціях привело до отримання значення функції втрат 0.1717 і точності генерації 0.9478 для one-hot кодування та 0.1220 і 0.9633 відповідно для моделі з використанням embedding-векторів за 200 епох навчання. Бачимо, що при тренуванні на англійському словнику різниця між

якістю моделей стала більш значимою. Гірші метрики при тренуванні моделей на англійському словнику можна пояснити одночасно тим, що розмір використаного англійського словника менший за розмір використаного російського словника, а також тим, що в англійській мові складніші правила побудови транскрипцій і існує більша кількість виключень і багатолітерних структур, що позначають один і той самий звук. В таб.4.2. наведено приклади транскрипцій англійських слів, побудованих моделями:

Таблиця 4.2. Приклади транскрибування англійських слів натренованими моделями

| Слово | One-hot pred. | Embedding pred. | Правильна транс. |
|---|---|---|---|
| squalling | S K W AO1 L IH0 NG | S K W AO1 L IH0 NG | S K W AO1 L IH0 NG |
| presentiments | P R IH0 S EH1 N T AH0 M AH0 N T S | P R IH0 Z EH1 N T AH0 M AH0 N T S | P R IH0 S EH1 N T AH0 M AH0 N T S |
| livelong | L IH1 V AH0 L AO0 | L IH1 V L L NG | L AY1 V L O0 NG |
| leatherbound | L EH1 DH ER0 ER0 ER0 AW2 AW2 | L EH1 DH ER0 B B AW2 AW2 | L EH1 DH ER0 B AW2 N D |
| metallical | M IH0 T AE1 L IH0 K AH0 L | M AH0 T AE1 L IH0 K AH0 L | M IH0 T AE1 L IH0 K AH0 L |
| moldering | M OW1 L D IH0 NG NG | M OW1 L D ER0 IH NG | M OW1 L D ER0 IH NG |

З таблиці бачимо, що моделі для транскрибування англійських слів краще справляються з розставленням наголосів, але гірше визначають звуки в цілому. Також можна відзначити, що модель з one-hot кодуванням частіше будує повністю правильні транскрипції, а модель з embedding-векторами в середньому помиляється в меншій кількості звуків в словах.

## 4.2. Модель для машинного перекладу текстів довільного змісту

Для побудови моделі машинного перекладу використано ідеї побудови encoder-decoder моделі запропоновані Налом Кальбреннером та Філом Блансомом в статті «*Recurrent Continuous Translation Models*». [26] Суть запропонованого методу полягає в використанні комбінації двох рекурентних нейронних мереж. Така архітектура призначена для задач, в яких вхід та вихід є певними послідовностями (seq2seq).

Перша мережа називається кодувальником. Вона приймає на вхід вхідну послідовність слів та повертає вектор фіксованої довжини – закодовану послідовність. В даній роботі запропоновано використовувати внутрішні стани LSTM після обробки всієї послідовності в якості виходу мережі-кодувальника. В такому випадку можна бути впевненим, що в значеннях, які повертає кодувальник міститься інформація про всю вхідну послідовність в певному виді.

Друга нейронна мережа в архітектурі називається декодувальником. Вона приймає в деякому виді вихідний вектор мережі-кодувальника і її завданням є генерація вихідної послідовності – перекладу.

Обидві мережі навчаються одночасно, в якості вхідних даних мережі-кодувальника при навчанні використовуються тексти оригіналів. В якості вхідних даних для мережі-декодувальника використовуються тексти перекладів, до яких на початку додано особливе слово START_TOKEN. Цей неінтуїтивний на перший погляд підхід дозволяє мережі одночасно тренуватись для генерації кожного елементу вихідної послідовності: при генерації першого елементу послідовності декодувальник володіє інформацією тільки про вихідний стан кодувальника, та отримує на вхід

START_TOKEN, який є спільним для всіх послідовностей в наборі даних, тобто не дає жодної інформації декодувальнику. На другому кроці декодувальник знову ж таки володітиме інформацією про стан кодувальника, а також інформацією про згенероване на попередньому кроці слово, а також про еталонне слово, яке мало бути згенероване на попередньому кроці перекладу. Такий підхід дозволяє уникати поширення помилки, зробленої мережею на початку генерації поширитись на всю наступну послідовність.

В даній роботі використовується модифікація encoder-decoder моделі з використанням в кодувальнику Bidirectional LSTM замість звичайної LSTM (рис.4.8.). Це дозволяє кодувальнику агрегувати й передавати в декодувальник більш повну інформацію про оригінальну послідовність. Внутрішнім станом Bidirectional LSTM є комбінація внутрішнього стану обох мереж LSTM, з яких вона складається. Bidirectional LSTM повертає четвірку станів $h_{forward}, c_{forward}, h_{backward}, c_{backward}$. Проте, декодувальник являє собою звичайну LSTM, оскільки використання для нього Bidirectional LSTM дозволило б мережі «підглядати» правильні слова при навчанні. Отже,

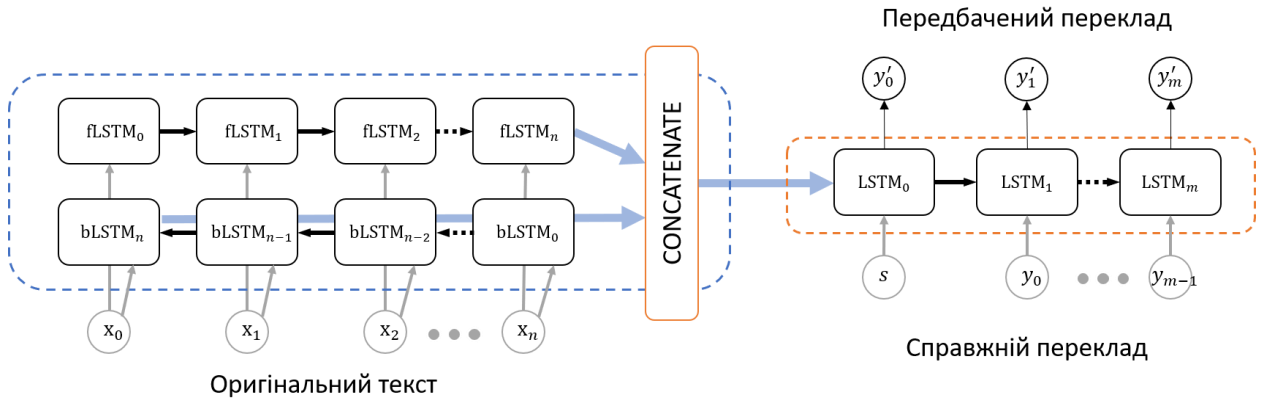


Рисунок 4.8. Концептуальна схема мережі для навчання encoder-decoder системи з використанням Bidirectional LSTM в кодувальнику

початкові стани декодувальника визначаються наступним чином (символ ⌒ позначає конкатенацію векторів):

$$h_0 = h_{forward} \frown h_{backward}$$

$$c_0 = c_{forward} \frown c_{backward}$$

Отже, модель-декодувальник включає в себе вдвічі більше LSTM-клітин, ніж кожна з LSTM модель-кодувальника, таким чином, що загальна кількість LSTM-клітин в мережах однакова. На практиці в системі використовується модель з одним Bidirectional LSTM шаром в кодувальнику та двома послідовно з'єднаними LSTM шарами в декодувальнику.

Модель не може працювати з текстами, тому для переводу текстів в числові вектори використовуються претреновані embedding-вектори (див. розділ 3.1.). При підготовці даних для навчання перш за все на корпусі вхідних текстів проводиться токенізація. У відповідність кожному слову ставиться певне число-індекс. Після цього будується матриця embedding-векторів, в якій і-ий рядок – embedding-вектор слова з індексом і. Після цього можемо в моделі передавати вхідні дані як список індексів слів, і обробляти їх Embedding шаром, проініціалізованим побудованою матрицею. Вихід Embedding-шару вже в свою чергу виступає входом рекурентної нейронної мережі.

Як і в моделі для генерації транскрипції при навчанні використовуємо оптимізатор Adam та функцію категоричної крос-ентропії для функції втрат. Так само, як і в попередній моделі можемо вважати, що задача генерації послідовності є еквівалентною до задачі класифікації кожного з елементів

генерованої послідовності. Конфігурацію моделі для тренування наведено на рис.4.9.

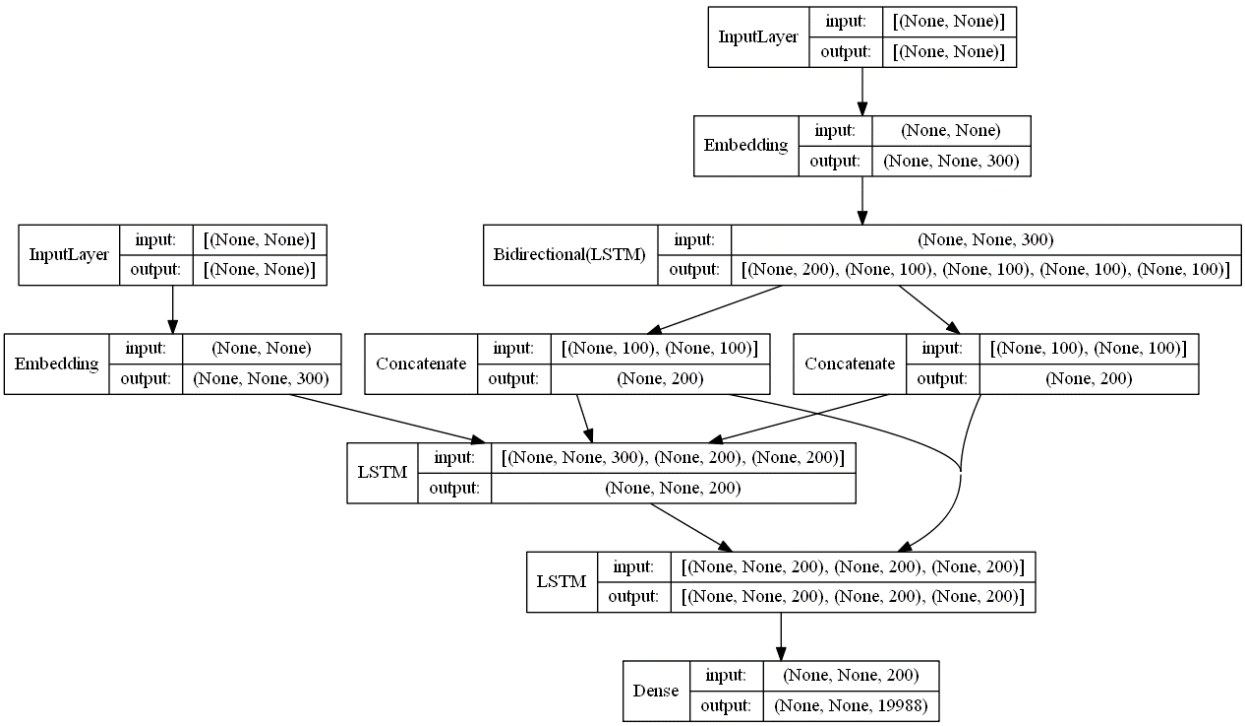


Рисунок 4.9. Конфігурація моделі для тренування системи encoder-decoder.

При використанні encoder-decoder системи для прогнозування послідовностей ми не маємо доступу до справжнього перекладу, отже ми маємо користуватись передбаченим перекладом, як справжнім (рис. 4.10).

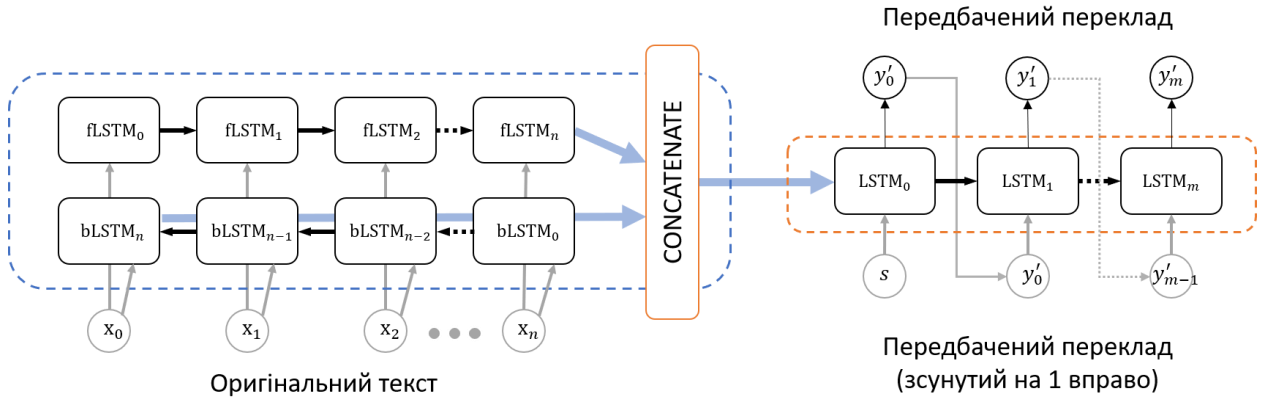


Рисунок 4.10. Концептуальна схема мережі для перекладу за допомогою encoder-decoder системи з використанням Bidirectional LSTM в кодувальнику

Отже, при роботі з системою encoder-decoder, як з перекладачем спочатку проводиться передбачення вихідних станів за допомогою окремо виділеної мережі-кодувальника (рис. 4.11.)

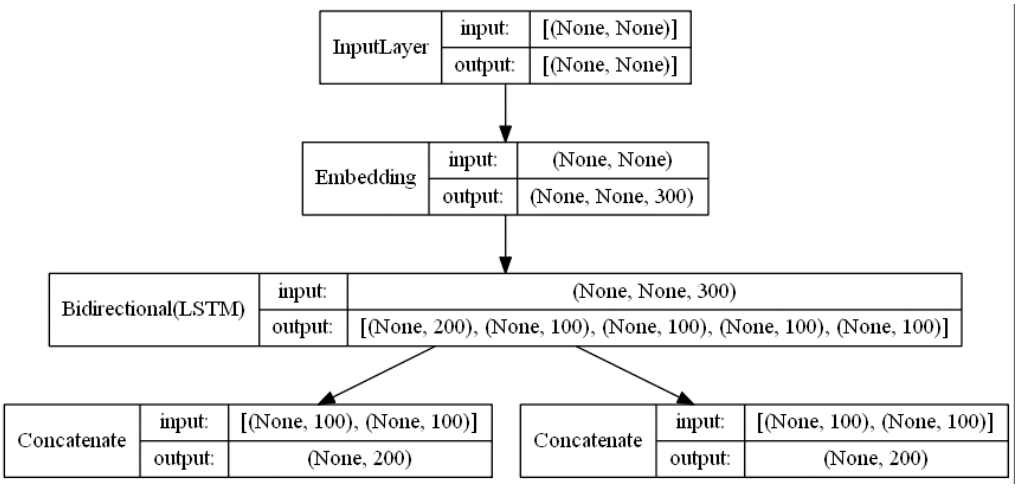


Рисунок 4.11. Конфігурація моделі кодувальника при використанні системи encoder-decoder для перекладу.

Декодувальник після цього повинен будувати переклад по одному слову. Модель декодувальника сконфігурована таким чином, що вона може приймати на вхід та видавати послідовності довільної довжини, про це свідчить значення None в розмірностях входів та виходів в конфігурації. (рис.4.12.) Отже, вхідними даними декодувальника в режимі генерації перекладу є:

- Послідовність з єдиного елемента – англійського слова, згенерованого на попередньому кроці. На першому кроці на вхід подається послідовність зі слова START_TOKEN.

- Стан LSTM після попереднього кроку (два вектори – $c, h$) На першому кроці на вхід подаються вихідні стани мережі-кодувальника після обробки оригінального тексту.

Відповідно виходами системи кодувальника є єдиний масив в якому на і-ій позиції знаходиться ймовірність того, що поточне слово перекладу буде словом з індексом i, а також стан LSTM після поточного кроку (два вектори – $c$ та $h$).

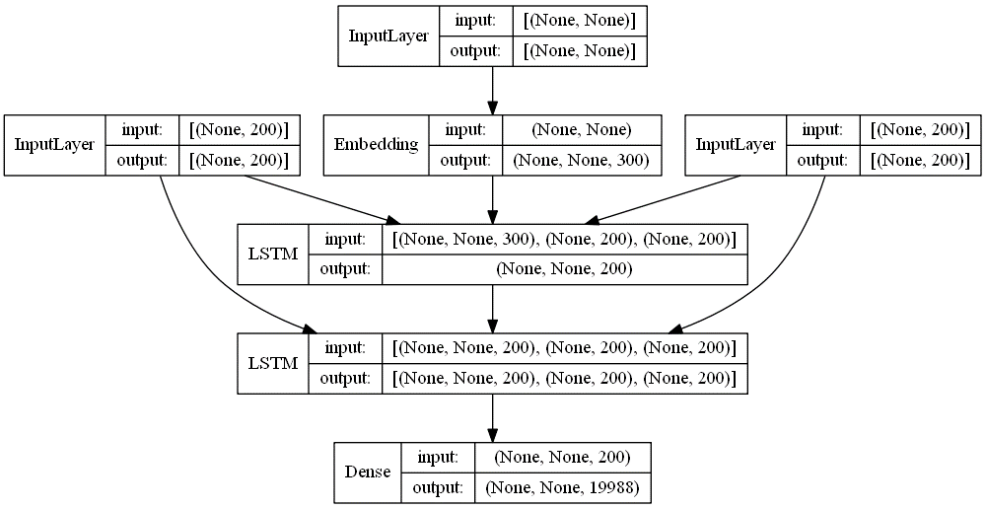


Рисунок 4.12. Конфігурація моделі декодувальника при використанні системи encoder-decoder для перекладу.

Генерація перекладу закінчується після того, як найбільш ймовірним згенерованим словом буде спеціальне слово END_TOKEN, яким закінчуються всі переклади в тренувальному та тестовому наборі. Тренування проводилось на наборі даних з розділу 3.3 протягом 300 тренувальних епох (рис.4.13, 4.14).

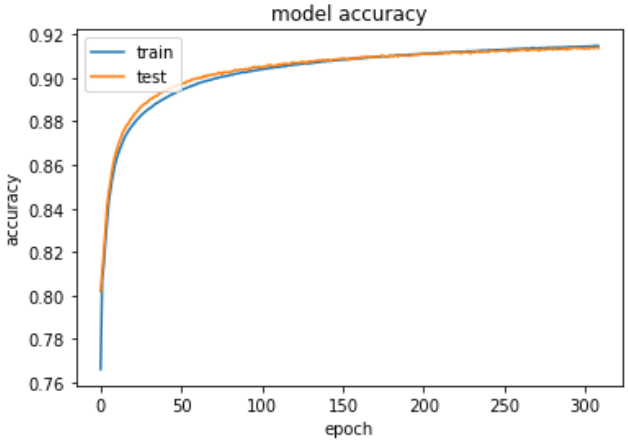


Рисунок 4.13. Графік зміни точності при навчанні encoder-decoder мережі

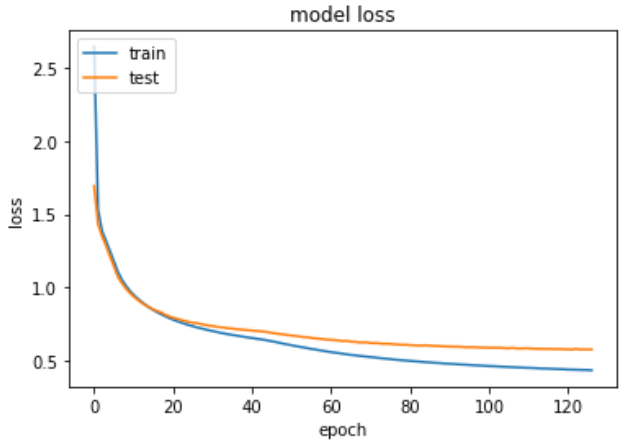


Рисунок 4.14. Графік зміни втрат при навчанні encoder-decoder мережі

Для тренування також була проведена попередня очистка набору даних:

- Видалено всі символи, що не є літерами кирилиці з текстів російською мовою та всі символи, що не є літерами латиниці з текстів англійською мовою
- Вилучено тексти, що складаються менш як з 5 та більш як з 25 слів
- Всі слова, що зустрічаються в текстах російською та англійською мовою відсортовано за кількістю їх повторів в наборі. Після цього всі слова, які не є одними з 20000 найчастіше вживаних було замінено на спеціальне слово UNKNOWN_TOKEN.
- Вилучено тексти, в яких більш ніж двічі зустрічається UNKNOWN_TOKEN.

Після препроцесингу набору він складав 4.8 мільйонів текстів. Для навчання було використано випадкову вибірку з 500 тисяч текстів. Приклади перекладу системою деяких речень з тестового набору наведено в таб.4.3.

Таблиця 4.3. Приклади перекладу речень натренованою encoder-decoder системою.

| | |
|---|---|
| Оригінальне речення | Беларусь по-прежнему заинтересована в развитии интеграционных процессов на постсоветском пространстве |
| Переклад системи | Belarus is interested in integration of the integration process in the post-soviet space |
| Еталонний переклад | Belarus is still interested in the development of integration processes in the post-soviet space. |
| | |
| Оригінальне речення | Сегодня пятница, так что мне завтра не надо идти в школу |

| | |
|---|---|
| Переклад системи | Today is Friday, so I’m glad you don’t want to go to school tomorrow. |
| Еталонний переклад | Today is Friday, so I don’t need to go to school tomorrow |
| | |
| Оригінальне речення | Глава региона также принял гостя с инвестиционным потенциалом |
| Переклад системи | The head of the region also visited the potential investors |
| Еталонний переклад | The head of the region also received a visitor with investment potential |
| | |
| Оригінальне речення | Каждый лично отвечает перед Богом за свои дела |
| Переклад системи | Everyone has his own personal responsibility for God |
| Еталонний переклад | The individual is responsible to god for his behavior |

Бачимо, що модель перекладає речення з корпусу доволі точно. При цьому видно, що вона вміє використовувати певні структури англійської мови які не мають чітких відповідників в російській, наприклад в останньому прикладі фраза «*каждый лично отвечает»* була перекладена як «*everyone has his own personal responsibility»*, а не «*everyone personally responses*», як швидше за все б зробили більш прості системи машинного перекладу, що базуються на словниках. При цьому модель часто помиляється «додумовуючи» контекст до речення. Це можна пояснити тим, що такий контекст міг бути присутнім у схожих реченнях з тренувального набору. Чисельний результат тренування моделі на 150 епохах показав досягнення точності перекладу в 68%. Точність визначається відсотком слів, що співпадають з еталонними перекладами в перекладах, запропонованих системою.

## 4.3. Модель для визначення віршованого розміру

Для побудови моделі визначення віршованого розміру було вручну розмічено набір віршів з розділу 3.2, та кожному віршу було присвоєно категорію одного з п’яти віршованих розмірів, або ж позначено, що вірш не належить до жодного з розглянутих розмірів [27] (таб.4.4, наголошений склад позначається символом /, ненаголошений – символом -). Також в рамках даної роботи використовується припущення, що віршований розмір в еталонному перекладі такий самий, як віршований розмір в оригіналі вірша.

Таблиця 4.4. Розподіл віршів в корпусі за їх віршованим розміром

| Назва віршованого розміру | Схема віршованого розміру | Кількість віршів в наборі |
|---|---|---|
| хорей | / - | 1970 |
| ямб | - / | 4382 |
| дактиль | / - - | 454 |
| амфібрахій | - / - | 940 |
| анапест | - - / | 1029 |
| невизначено | | 222 |

Можна обґрунтовано вважати, що відповідність вірша віршованому розміру визначається його транскрипцією. Модель, запропонована в розділі 4.1 використовується для перетворення кожного вірша в корпусі на його транскрипцію. На місці переходу рядка в транскрипцію також додано спеціальний елемент EOL_TOKEN. Більшість віршів, як в запропонованому корпусі, так і в цілому, написані з використанням чотирирядкових строф, тому доцільно для навчання розбити кожну транскрипцію на транскрипції чотирирядкових строф. Після цього можна побудувати модель класифікатора, яка складається з LSTM-шару, що власне аналізує послідовність токенів транскрипції, та прихованого повнозв’язного шару, який класифікує результат

аналізу послідовності в один з описаних в таб.4.4 класів (рис.4.15). Для додаткової точності також додано шар Embedding, який тренує embedding-вектори для символів транскрипції.

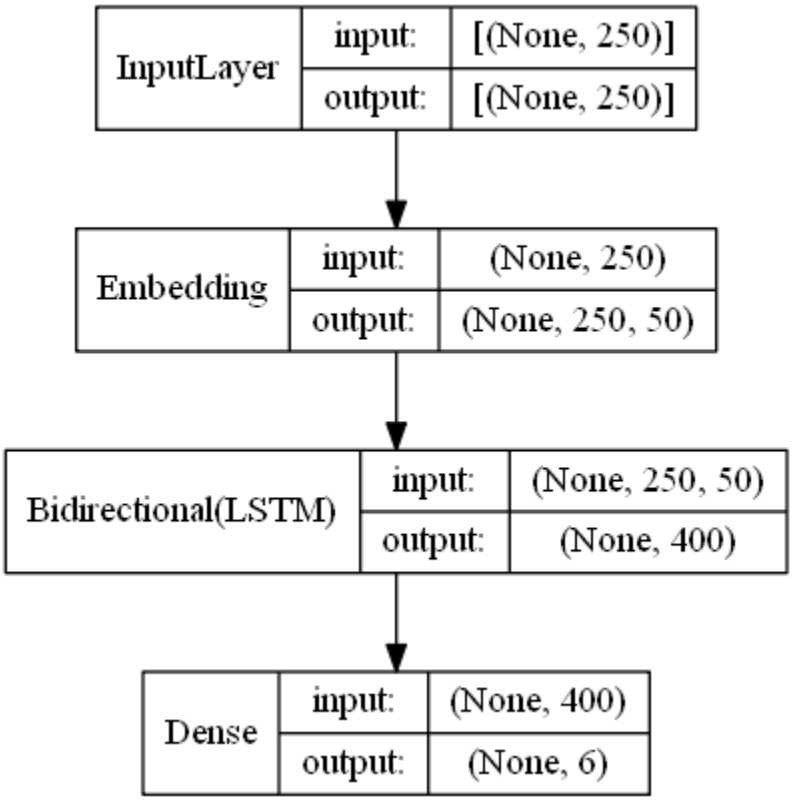


Рисунок. 4.15. Конфігурація моделі для визначення віршованого розміру вірша по його транскрипції.

## 4.4. Модель для генерації перекладу віршів

Ідея запропонованої моделі полягає в тому, що якщо використовувати encoder-decoder модель для перекладу, то слова перекладу генеруються послідовно і при цьому кожне наступне слово будується з урахуванням згенерованих до нього слів. Це дозволяє втручатись в процес генерації. Отже, можна побудувати наступний алгоритм генерації перекладу вірша, намагаючись задовольнити тільки умову ритмічності:

1. Генеруємо вихідні стани кодувальника INITIAL_STATE, передавши в натреновану мережу-кодувальник оригінал вірша.

2. Користаємось моделлю 4.3 для визначення розміру вірша-оригіналу FOOT.

3. Передаємо в декодувальник послідовність з одного слова START_TOKEN разом зі станом INITIAL_STATE в якості початкового стану декодувальника. На виході отримуємо масив ймовірностей того, що певні слова є першим словом перекладу. Обираємо $M$ слів з найвищими ймовірностями. Для кожного зі слів за допомогою моделі 4.3 визначимо ймовірність того, що воно є віршем з розміром FOOT. Запам'ятаємо отримані послідовності в список CANDIDATES разом з ймовірностями їх відповідності розміру FOOT.

4. Передаємо в декодувальник всі послідовності слів $\mathrm{SEQ}_i$ (префікси перекладу) зі списку CANDIDATES разом зі станом INITIAL_STATE в якості початкового стану декодувальника. Для кожної з них знову обираємо M слів з найвищими ймовірностями бути наступними у перекладі. Для кожного з цих слів за допомогою моделі 4.3 обчислюємо ймовірність того, що послідовність $\mathrm{SEQ}_i + \mathrm{M}$ є віршем з розміром FOOT. Записуємо послідовність $\mathrm{SEQ}_i + \mathrm{M}$ в список NEW_CANDIDATES разом з обчисленою ймовірністю.

5. Обираємо з списку NEW_CANDIDATES M послідовностей з найвищою ймовірністю бути віршем з віршованим розміром FOOT. Якщо серед обраних M послідовностей є такі, що закінчуються на END_TOKEN, тобто завершені вірші, то їх прибираємо зі списку NEW_CANDIDATES та додаємо до списку FINISHED_CANDIDATES, а також зменшуємо M на одиницю за кожну таку послідовність.

6. Повторюємо кроки 4-5 поки M не стане рівним нулю.

7. Зі списку FINISHED_CANDIDATES вибираємо послідовність з найвищою ймовірністю бути віршем з віршованим розміром FOOT. Вона й буде перекладом вірша зі збереженням віршованого розміру.

На практиці в даній роботі генеровані таким алгоритмом переклади для справжніх віршів є доволі неточними. Це можна пов'язати з відсутністю корпусу паралельних перекладів художніх текстів достатнього розміру для тренування encoder-decoder системи, що не дозволяє їй належним чином генерувати варіанти перекладів для віршів, оскільки вони є творами з доволі складними як граматичною, так і лексичною структурами. В таб.4.5. наведено приклад перекладу, виконаного системою.

Таблиця 4.5. Приклад перекладу віршів запропонований системою.

| | |
|---|---|
| Оригінал<br>(А. Ахматова, Широк и жёлт вечерний свет…) | Сюда ко мне поближе сядь,<br>Гляди весёлыми глазами,<br>Вот эта синяя тетрадь,<br>С моими детскими стихами |
| Переклад системи | H**e**re sit cl**o**ser n**e**xt to m**e**<br>L**oo**k with h**a**ppy **e**yes<br>H**e**re this **a**zure n**o**tebook **i**s<br>W**i**th my ch**i**ldhood p**o**ems |
| Еталонний переклад<br>(Ilya Shambat) | Come and sit right next to me,<br>With the happy eyes come look:<br>Here, my childhood poetry<br>Is in this blue notebook. |

Бачимо, що наведений переклад є доволі вдалим і добре зберігає як ритм, так і зміст строфи-оригіналу і підтверджує здатність запропонованого алгоритму вирішувати задачу перекладу віршів зі збереженням ритму.

# ВИСНОВКИ

У роботі було запропоновано та реалізовано алгоритм для машинного перекладу віршів, а також спроектовано та натреновано моделі нейронних мереж, необхідні для його роботи. Технологією розробки була обрана мова Python з бібліотекою TensorFlow Keras, оскільки вона пропонує гнучкий та зручний інтерфейс для імплементації нейронних мереж.

В рамках кваліфікаційної роботи було виконано поставлені завдання, а саме:

- Проаналізовано існуючі дослідження у сфері машинного перекладу текстів, віршів та у сфері генерації віршів.
- Спроектовано нейронні мережі для побудови транскрипції слів, для перекладу текстів та для визначення віршованого розміру вірша.
- Сформовано та проаналізовано набори даних для навчання моделей.
- Запропоновані моделі імплементовано та натреновано за допомогою TensorFlow Keras.
- Наведено результати роботи моделей та алгоритму для перекладу віршів.

Результатом кваліфікаційної роботи є побудований алгоритм для автоматичного перекладу текстів, а також його реалізація, та реалізація всіх моделей нейронних мереж, необхідних для його роботи.

Мета по побудові алгоритму для автоматичного перекладу віршованого тексту з російської на англійську мову зі збереженням ритму була досягнута.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. IBM Cloud Learn Hub: Natural Language Processing [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/learn/natural-language-processing
2. Natural Language Processing [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Natural_language_processing
3. Winograd, Terry (1970-08-24). "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language". MIT AI Technical Report 235. URI: http://hdl.handle.net/1721.1/7095
4. Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. Commun. ACM 9, 1 (Jan. 1966), 36–45. DOI: 10.1145/365153.365168
5. Corpus linguistics [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Corpus_linguistics
6. Corpus of Global Web-Based English [Електронний ресурс] – Режим доступу до ресурсу: https://www.english-corpora.org/glowbe/
7. Sketch Engine English Medical Corpus [Електронний ресурс] – Режим доступу до ресурсу: https://www.sketchengine.eu/medical-web-corpus/
8. Charniak, E. (1997). Statistical Techniques for Natural Language Parsing. AI Magazine, 18(4), 33. DOI: 10.1609/aimag.v18i4.1320
9. Build a Deep Learning Text Generator Project with Markov Chains [Електронний ресурс] – Режим доступу до ресурсу: https://www.educative.io/blog/deep-learning-text-generation-markov-chains
10. Dmitriy Genzel, Jakob Uszkoreit, Franz Och. “Poetic” Statistical Machine Translation: Rhyme and Meter. EMNLP (2010), pp. 158-166. URI: https://www.aclweb.org/anthology/D10-1016

11. Hisar Manurung. An evolutionary algorithm approach to poetry generation. PhD Thesis, University of Edinburgh, 2003. URI: http://hdl.handle.net/1842/314
12. Simon Colton, Jacob Goodwin, Tony Veale. Full-FACE poetry generation. In Proceedings of the 3rd International Conference on Computational Creativity (ICCC), pp. 95–102, 2012.
13. Jack Hopkins, Douwe Kiela.Automatically Generating Rhytmic Verse with Neural Networks. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 168-178, DOI: 10.18653/v1/P17-1016
14. Word Embedding [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Word_embedding
15. Global Vectors [Електронний ресурс] – Режим доступу до ресурсу: https://nlp.stanford.edu/projects/glove/
16. Deep Pavlov Pre-trained Russian Embeddings [Електронний ресурс] – Режим доступу до ресурсу: http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html
17. RuVerses [Електронний ресурс] – Режим доступу до ресурсу: https://ruverses.com/
18. Документація бібліотеки Beautiful Soup [Електронний ресурс] – Режим доступу до ресурсу: https://www.crummy.com/software/BeautifulSoup
19. ParaCrawl Corpus [Електронний ресурс] – Режим доступу до ресурсу: https://www.paracrawl.eu/index.php
20. ARPABET [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ARPABET
21. Словник англійських транскрипцій проекту CMUSphinx [Електронний ресурс] – Режим доступу до ресурсу: http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/

22. Словник російських транскрипцій проекту CMUSphinx [Електронний ресурс] – Режим доступу до ресурсу: https://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/Russian/
23. Long short-term memory [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Long_short-term_memory
24. Sepp Hochreiter. Long Short-term Memory (1997). DOI:10.1162/neco.1997.9.8.1735
25. Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization (2017). arXiv: 1412.6980
26. Nal Kalchbrenner, Phil Blunsom. Recurrent Continuous Translation Models. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1700–1709. ACL: D13-1176
27. Віршований розмір [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Foot_(prosody)