Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів машинного навчання та data lake для навчальних систем
(тема)

Виконав: студент	2 курсу групи ІПЗм-20-4
	Латиш А.С.
	(прізвище, ініціали)
Спеціальність	121 – Інженерія програмного забезпечення
Тип програми	Освітньо-наукова
Керівник	проф. Єрохін А.Л.
	(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 - Інженерія програмного забезпечення
(код і повна назва)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ *20 ___ р*

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Латишу Артему Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Дослідження методів машинного навчання та data lake для навчальних систем»
затверджена наказом по університету від ___ _____2022 р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії ___ _____2022 р.
3. Вихідні дані до роботи: *У програмного забезпеченні передбачено: авторизація, перегляд моделей навчання, вибір моделі навчання для використання, проходження навчальної сесії, перегляд статистики по навчальним сесіям.*
Використовувати: СКБД MySQL 8, ОС Windows 10 Pro, мову php 8, мову python 3.9
4. Перелік питань, що потрібно опрацювати в роботі: *Вступ, аналітичний огляд, аналіз існуючих методів або алгоритмів , архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень , висновки, перелік посилань.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін	Відмітки про виконання
1	Аналітичний огляд	5 березня 2022	виконано
2	Аналіз існуючих методів або алгоритмів	21 березня 2022	виконано
3	Розробка моделі регресії періоду напіврозпаду	12 квітня 2022	виконано
4	Формування вимог ПЗ	15 квітня 2022	виконано
5	Проектування та розробка ПЗ	25 квітня 2022	виконано
6	Написання пояснювальної записки	1 травня 2022	виконано
7	Перевірка записки керівником на нормоконтролером	5 травня 2022	виконано
8	Оцінка роботи стороннім рецензентом, отримання відгуку від керівника атестаційної роботи, попередній захист	7 травня 2022	виконано
9	Здача роботи у електронний архів, допуск роботи до захисту завідувачем кафедри та передача готової роботи секретарю ЕК	15 травня 2022	виконано
10	Здача готової роботи	25 травня 2022	виконано

Дата видачі завдання 17.01.2022 р.

Керівник проф. _____(Єрохін А.Л.)

Завдання прийняв до виконання_____ (Латиш А.С.)

РЕФЕРАТ / ABSTRACT

Звіт з практики містить: 70 с., 15 рис., 2 табл., 8 формул, 22 джерел.

МАШИННЕ НАВЧАННЯ, DATA LAKE, МОДЕЛЬ, ОПТИМІЗАЦІЯ АЛГОРИТМУ

Об’єктом аналізу та дослідження є алгоритми, які допомагають прискорити навчання людини з використання машинного навчання та data lake.

Метою кваліфікаційної роботи є аналіз алгоритмів та методів машинного навчання, які можуть допомогти у навчанні та опрацьовані методики вивчення інформації, а також розробка власної моделі, яка буде використовувати алгоритм інтервального повторення і передбачувати забування слова користувачем.

У результаті кваліфікаційної роботи було проаналізовано алгоритм інтервального повторення, та розроблена модель навчання з використання великої кількості даних, завантажених з відкритих джерел.

MACHINE LEARNING, DATA LAKE, MODEL, OPTIMIZATION ALGORITHM

The object of analysis and research is algorithms that help speed up human learning using machine learning and data lake.

The purpose of the qualification work is to analyze algorithms and methods of machine learning that can help in learning and developed methods for studying information, as well as to develop your own model that will use the interval repetition algorithm and predict the user's forgetting of a word.

As a result of the qualification work, the interval repetition algorithm was analyzed, and a training model was developed using a large amount of data downloaded from open sources.

ЗМІСТ

ВСТУП

В сьогоднішніх реаліях машинне навчання займає вагому роль. Машинне навчання може використовуватися майже в будь-який областях. В основі машинного навчання лежить використання методів статистики, аналізу, оптимізації і тд.

Яку роль у навчальних системах грає машинне навчання? Машинне навчання може використовувати набір даних для аналізу результату. А саме використання набору схожого до вхідного. За допомогою цього методу можна визначити найкращий спосіб вивчення інформації.

Що ж до data lake, використовуючи цю технологію в хмарі, інженери, аналітики та фахівці з обробок даних можуть дуже просто обробляти, зберігати та отримувати доступ до потрібної інформації для машинного. Це допомагає впроваджувати інновації, перевіряти гіпотези або аналізувати дані.

В Абердіні провели опитування, яке показало, що компанії, які використовували data lake разом з машинним навчанням, перевершили аналогічні компанії на 9% зростанням доходу.

То ж, завдання цієї атестаційної роботи – опрацювати та розробити методи навчання системи з використанням машинного навчання та data lake.

Метою дослідження кваліфікаційної роботи є аналіз існуючих алгоритмів машинного навчання, які можуть допомогти у навчанні та опрацьовані методики вивчення інформації, а також розробка власної моделі, яка буде використовувати алгоритм інтервального повторення і передбачувати забування слова користувачем.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Загальні поняття машинного навчання

Розглянемо поняття. Машинне навчання – це деякі методи з використанням штучного інтелекту, які дозволяють системі самовдосконалюватися на власному досвіді, без будь-яких додаткових затрат на кодування. Це робить систему схожою на людину.

Розглянемо простенький приклад. Ви заходите до сайту інтернет-магазину, вибираєте товар, на сторінці цього товару система пропонує інші товари за даною категорію, або товари які покупали інші товари разом з даним, а отже вам це також може згодитися. Ця проста функція збільшую продажі магазину, а вам допомагає знайти потрібний вам товар.

Основа машинного навчання – це навчання системи за допомогою безлічі схожий завдань. Тобто такий спосіб не розв’язує напряму потрібну задачу, але так компанії не потрібно витрачати додаткові години, дні, а то і місяці розробника, який кодує систему. За нього це зробить штучний інтелект. Так, спочатку метод буде не точним, але з часом він буде накопичувати досвід та корегувати результати до більш точніших.

Вибір алгоритму навчання залежить від того типу даних, який система буде мати у достатньому обсязі та виду автоматизації, яку потрібно виконати.

У чому різниця від звичайного програмування? У класичному програмуванні ми пишемо програму яка протестована на деяких вхідних даних. Якщо ж ми говоримо про машинне навчання, ми передаємо вхідні та вихідні дані, а машина сама розроблює програму. На рисунку 1.1 представлена загальна схема.

Машинне навчання автоматизує безліч різноманітних справ, а особливо завдання для яких потрібен інтелект людини, тиражування якого може бути досягнуто тільки за допомогою машинного навчання.

Machine learning актуально і на підприємствах, де воно може швидко створювати моделі аналізу даних або автоматизувати рутинні проблеми. Промислові підприємства залежать від неймовірних обсягів даних. Вони потрібні

для прийняття розсудливих рішень та оптимізації діяльності. Machine learning створює моделі, які можуть аналізувати та обробляти колосальні обсяги проблемних даних для отримання правильних висновків. Моделі повинні бути масштабованими, точними та функціонувати з мінімальними витратами часу для виконання тих чи інших робіт. Підприємства досить легко можуть використовувати можливості та уникати невідомих ризиків, створюючи такі точні моделі машинного навчання.

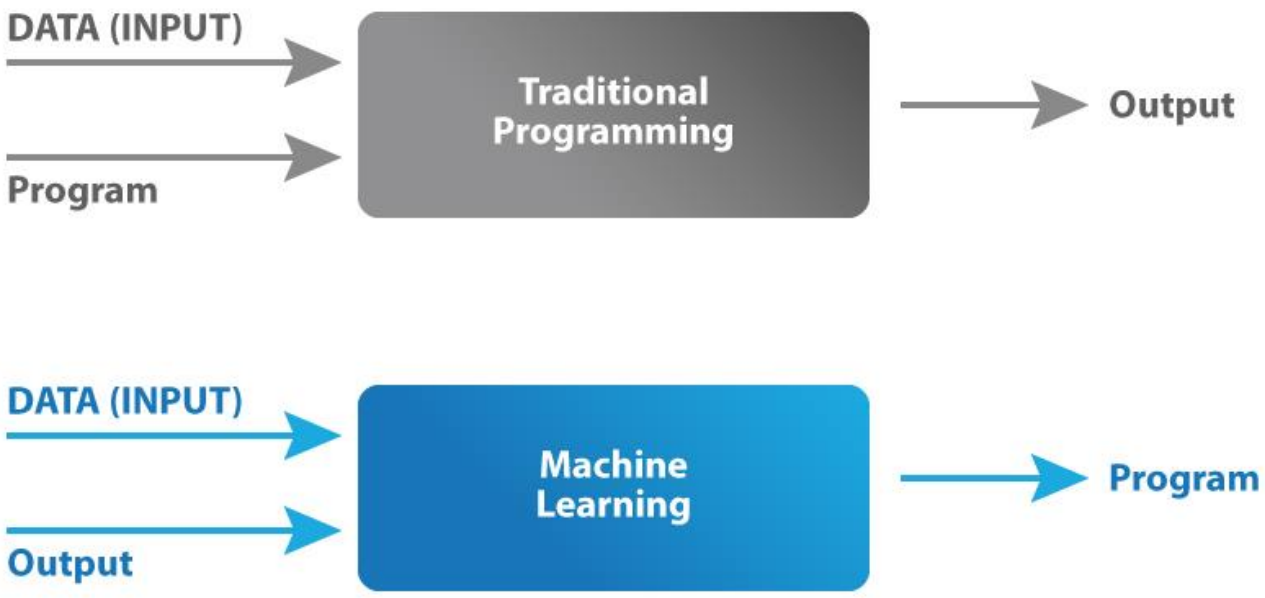

Рис. 1.1 – Загальна схема програмування на машинного навчання [2]

У реальному світі можна знайти багато варіантів використання цієї технології, наприклад: генерація тексту, розпізнавання зображень і тд. Що в свою чергу приводить до розширення можливостей експертів машинного навчання, і становлення затребуваними професіоналами.

Для опанування методології, потрібно знати основні поняття. Давайте їх розглянемо.

– Модель або гіпотеза – являється математичним представленням реального процесу. Модель будується за допомогою навчальних даних та алгоритму машинного навчання.

– Характеристика – характеристикою описують параметр набору даних або вимірювану властивість.

– Навчання. В якості вхідних даних алгоритм приймає деякий набір даних, відомих як «навчальні дані». Алгоритм повинен знайти закономірність, а далі навчає модель очікуваним цілям. В результаті процесу навчання виходить модель машинного навчання.

– Вектор об'єктів – представляє собою набір з декількох числових об'єктів. Він використовується в якості вхідних даних для моделей машинного навчання з метою прогнозування та навчання.

– Прогнозування. Одразу після створення моделі машинного навчання, вона може приймати вхідні дані для отримання прогнозованого результату.

– Перенавчання. Означає витягування навчальних моделей з неточних вхідних даних та шуму. Як правило це виходить в результаті обробки великої кількості інформації. В даному випадку модель не може правильно охарактеризувати дані.

– Недостатня відповідність – сценарій неможливості моделі розшифрувати тенденцію вхідних даних. А отже, це руйнує точність моделі. Простіше кажучи, алгоритм або модель недостатньо добре відповідають даним.

– Мета – передбачуване значення моделі.

1.2 Етапи машинного навчання

Всього існує сім основних етапів:

1. Збір даних (Gathering Data)
2. Підготовка даних (Preparing data)
3. Вибір моделі (Choosing a model)
4. Навчання (Training)

5. Оцінка (Evaluation)
6. Оптимізація гіперпараметрів (Hyperparameter Tuning)
7. Передбачення (Prediction)

Для початку, вивчення мови програмування є обов'язковим. Переважно використовують Python, тому що ця мова програмування ідеально підходить для роботи с такими завданнями, але вам ніхто не заперечує використати інші, улюблені мови. Також вам потрібно мати необхідні математичні та аналітичні знання. Як правило для вирішення задач машинного навчання потрібно освіжити наступні математичні області:

– математичний аналіз: градієнти та похідні;

– лінійна алгебра для аналізу даних: матриці, вектори, скаляри і тензори;

– багатовимірне обчислення;

– складні оптимізації та алгоритми;

– статистика і теорія ймовірностей.

Як працює машинне навчання? Трьома центральними поняттями є учень, модель і параметри.

– Учень коректує модель і параметри, для приведення прогнозів до фактичного результату.

– Модель – система, яка робить прогнози.

– Параметри – фактори прогнозів для врахування моделлю.

Сам процес машинного навчання складається з трьох етапів: навчання, перевірка, тестування. Схема представлена на рисунку 1.2.

Навчання на основі навчального набору. Цей набір використовується як вибірка даних для відповідності моделі, тобто фактичний набір, який ми використовуємо для навчання моделі (зміщення у випадку нейронної мережі та ваги). З допомогою цих даних модель створює уроки.

Набір даних перевірки використовується для того, щоб забезпечити неупереджену оцінку, при налаштуванні гіперпараметрів моделі, відповідності набору навчальних даних. Оцінка стає точнішою, бо навички в наборі перевірки долучаються до конфігурації моделі.

Набір перевірки часто використовують для частотної оцінки даної моделі. Інженери використовують ці дані для чітких налаштувань гіперпараметрів. З цього випливає, що модель іноді бачить такі дані, але ніколи не робить уроки них. Результати набору перевірки використовуються для оновлення гіперпараметрів високого рівня. А отже, набір перевірки побічно впливає на саму модель. Також цей набір відомий як набір для розробки або набір розробників, та допомагає на етапі розробки моделі.

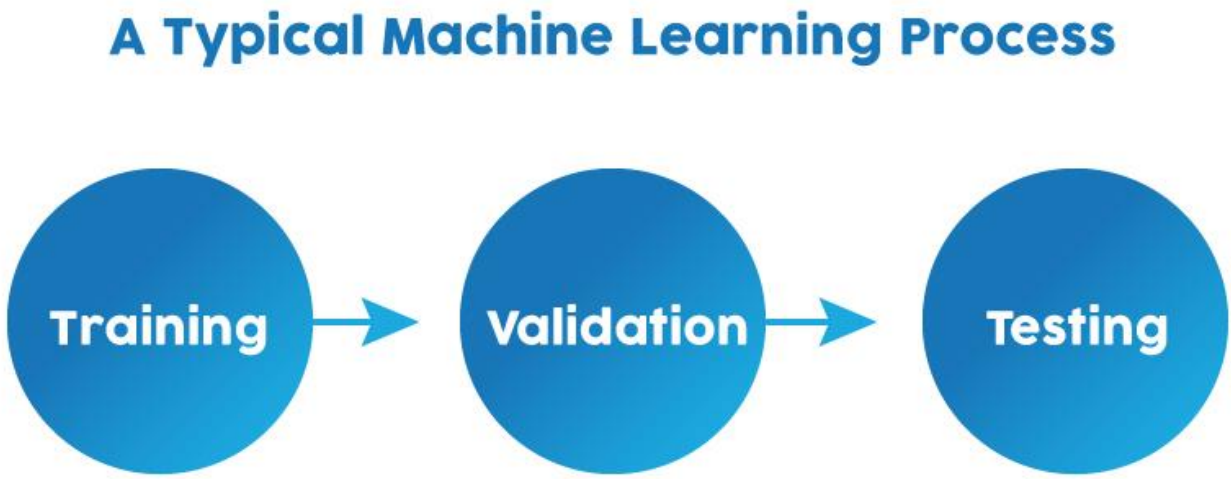

Рис. 1.2 – Схема етапів машинного навчання [3]

Тестовий набір даних використовується при забезпеченні неупередженої оцінки відповідності моделі набору даних.

Тестовий набір використовується для оцінки моделі та забезпечує ідеальний стандарт. Але він може бути використаний тільки після повного навчання моделі, з використанням наборів перевірки та навчання. Тестовий набір використовується, як правило, для оцінки конкурентних моделей. Візьмемо, для прикладу, змагання Kaggle. Вони спочатку випускають набір перевірки разом з навчальним набором, а дійсний тестовий набір показують тільки під завершення змагання, і саме кінцевий набір визначає переможця. Набір для перевірки можна використовувати у якості тестового набору, але практика показала, що ідея не працездатна. Набір тестів

повинен бути добре підібраним та містити детально відібрані дані, охоплюючи різні класи зіткнення моделі при використанні на реальних прикладах. На рисунку 1.3 представлено співвідношення етапів між собою.

Поділ набору даних. Розглянемо рекомендації до розподілу даних на навчальні, валідаційні та тестові набори.

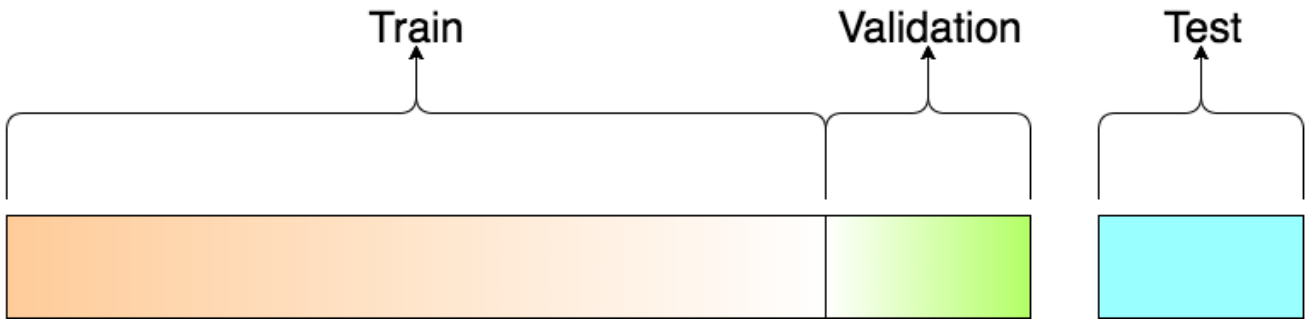

Рис. 1.3 – Співвідношення етапів машинного навчання [4]

В основі розподілу лежить дві речі. По-перше, сумарна кількість вибірок на фактичній моделі, яку ви навчаєте і, по-друге у ваших даних.

Моделі які потребують колосальних даних для навчання необхідно оптимізувати для таких обсягів навчальних наборів. Моделі ж з малою кількістю гіперпараметрів набагато легше налаштовувати та перевіряти, тому інженеру потрібно зменшити розмір набору перевірки, але для моделей з багатьма гіперпараметрів рекомендується мати великий набір перевірки. Також є один нюанс. Інженеру не потрібно розроблювати набір перевірки для моделі без гіперпараметрів або для складно налаштовуваних.

В загальному, як і багато інших речей в машинному навчанні, співвідношення достатньо специфічне під власне використання для поділу. Під час створення великої кількості моделей судити стає все простіше і простіше.

Варто пам’ятати, що вибір неправильної навчальної моделі машинного навчання напряму позначиться на результатах виконання. Детальніше про цю тему ви можете дізнатися у доповіді, яка знаходиться у другому томі тезисів «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» [5]

1.3 Постановка задач дослідження

Отже, для того щоб побудувати модель машинного навчання, яка буде підходити для потреб потрібно:

- проаналізувати та вибрати алгоритм, який буде використовуватися для моделі;
- знайти найбільш підходящу функцію затрат моделі;
- підготовити тренувальні дані на яких буде навчатися модель;
- проаналізувати результати та зробити висновки щодо доцільності використання вибраного алгоритму;;
- якщо справжні результати не будуть співпадати з очікуваними, знайти причину і скоректувати модель.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ АБО АЛГОРИТМІВ

2.1 Загальний аналіз

В останні роки з’явився вкрай цікавий інструмент в арсеналі наукових діячів: машинне навчання. Ця технологія уже довела свою актуальність та цінність у здатності прискорити як прикладні розробки, так і фундаментальні теоретичні дослідження. У цьому розділі ми розглянемо аналіз деяких методів для прискорення навчання застосовуючи machine learning.

Машинне навчання може використовуватися в будь-яких сферах навчання: від перевірки домашнього завдання до проведення дослідження в області генетики. Давайте розглянемо область вивчення іноземної мови. У цій області широкий спектр можливостей для використання data lake технологій та машинного навчання.

2.2 Вибір алгоритму інтервального повторення

Що ж, розглянемо алгоритм інтервального повторення. Основа цього алгоритму полягає у збільшені проміжків для повторення уже вивченої інформації для збільшення ефективності користі від розподілу.

2.2.1 Значимість

Багато різноманітних наукових дисциплін боролися з проблемою розуміння пам’яті людини ще з давніх часів. Первинні праці були сконцентровані на охарактеризуванні пам’яті людини та з використанням контрольованих дрібномасштабних тестів. Пізніше, данні емпіричні розробки стали центральним стрижнем розробки ефективних алгоритмів інтервального повторення. А втім, сьогоденні алгоритми рознесеного повторення представляють евристику, основану на принципах, з конкретними непластичними властивостями, які не застосовують ретельний автоматичний нагляд та високий рівень контролю, які пропонують та використовують новітні системи онлайн-навчання. Далі буде показано розробку

структури обчислення для отримання оптимального алгоритму повторення, яка буде адаптована під успішність учнів. Дослідивши даний алгоритм з розглядом популярних систем для навчання, ми зможемо переконатися у перевагах даного методу проти альтернатив.

2.2.2 Структура моделі

Вибір моделі пам’яті не впливає на дану структуру, бо надає набір підходів для відбору графіків діагностики, що має оптимальну якість в рамках досліджуваної моделі пам’яті. Для більш простого розуміння, ми розглянемо структуру для всесвітньо відомої моделі пам’яті з психології та літератури: експоненціальної моделі дуги забування з двійковими відгуками [6] та додаймо до нього підхід машинного навчання, регресію періоду напіврозпаду [7], для того щоб оглянути вплив параметрів цієї моделі.

Для більшої конкретики, розглянемо учня, що має бажання вивчити набір предметів $\mathscr{I}$, використовуючи метод інтервального повторення, отже повторне проглядання предметів з періодом, ми прописуємо кожний випадок перегляду як триплет (1)

$$e := \begin{pmatrix} & time & \\ & \downarrow & \\ i & , & t & , & r \\ \uparrow & & & & \uparrow \\ item & & & & recall \end{pmatrix} \tag{1}$$

З чого випливає, що учень, в момент часу *t*, переглянув елемент $i \in \mathscr{I}$ та або забув його (r = 0), або згадав (r = 1). Необхідно звернути уваги, що кожний випадок перевірки включає результати даного відкликання. Тобто, з основної роботи Карпіке і Рейдігера, учня перевіряють у кожній ітерації.

Давайте змоделюємо ймовірність учня згадати чи забути елемент *i* в момент часу *t*, при цьому враховуючи приведене вищу уявлення

та використовуючи експоненціальну модель кривої забування (2)

$$m_i(t) \coloneqq \mathbb{P}(r) = \exp\big(-n_i(t)\,(t - t_r)\big), \qquad (2)$$

де t_r – момент часу останнього перегляду, а $n_i(t) \in \mathbb{R}^+$ - частота забування в момент часу t, вона може залежати від безлічі факторів, наприклад, кількості попередніх згадувань/забувань або складності даного елементу.

Потім потрібно відстежити час вивчення, за допомогою багатогранного процесу підрахування $N(t)$, у $i_{му}$ елементі, $N_i(t)$ рахує кількість переглядів учнем елементу i до часу t, звернувшись до книги по тимчасових точках [8], ми можемо охарактеризувати процеси обчислення, з використанням їх відповідних проміжків $u(t)$, наприклад $\mathbb{E}\,[dN\,(t)] = u(t)dt$, і думаємо про вивчення r по двійкових мітках. А головне, щоразу як учень переглядає елемент, то згадування r впливає на швидкість забування цього елементу. Це було експериментально доведено. Ми можемо оцінити наступний ефект з використанням регресії періоду напіврозпаду, яка неявно передбачає, що згадування елемента i під час тестування має накопичуваний ефект на швидкість забування $n_i(t)$. Успішне згадування під час t_r корегує швидкість забування на $(1 - a_i)$, наприклад $n_i(t) = (1 - a_i)n_i(t_r), a_i \leq 1$, що ж до невдалого згадування, то воно корегує на $(1 + \beta_i)$, тобто $n_i(t) = (1 + \beta_i)n_i(t_r), \beta_i \geq 0$. Початкова швидкість забування $n_i(0)$, у цьому контексті, описує важкість елемента, при цьому, слід мати на увазі, складніші елементи мають більш високі вихідні показники забування, порівнюючи з легшими елементами, а властивості a_i, β_i та $n_i(0)$ повинні бути оцінені використовуючи перевіренні реальні дані.

2.2.3 Експеримент

Для перевірки даного алгоритму запам’ятовування, були взяті дані з відкритих джерел, а саме набір інформації для вивчення мов. Набір, який було взято складається з дванадцяти мільйонів сеансів вивчення мови, та складає понад п’ять

мільйонів унікальних пар (слово - користувач). Позначимо їх літерою $\mathcal{D}$. Варто зауважити, що дані були зібрані за двотижневий період. Користувач, за один сеанс навчання, відповідає на декілька тестів, кожен з яких містить два або більше слів. Кожне слово у списку зіставляється з i, також, в сеансі, частка правильних згадувань, що містять слово i, розглядається в якості ймовірності його емпіричної оцінки запам'ятовування $\hat{m}(t)$ під час сеансу t. Слово вважається успішним тільки тоді, коли воно згадується ідеально під час сеансу, тобто $r_i(t) = 1$, при інших випадках – невдалим, тобто $r_i(t) = 0$. Так як ми можемо сподіватися, що оцінка властивостей моделі буде достатньо точною тільки у користувачів та слів зі значимою кількістю подій тестування, то ми візьмемо користувачів що найменш з 30 подіями тестування та слова повинні бути переглянуті як мінімум 30 разів. Після такої обробки даних, виявилось, що була відкинута лише двадцята частина даних, а отже досліджувані початкові дані було взято досить влучно.

Під час дослідження було порівняно та опрацьовано продуктивність методу з двома ключовими показниками: (i) загальний графік перевірки, який відправляє елементи на тестування з деякою постійною швидкістю μ, та (ii) графік перевірки на основі встановленого порогу, що прискорює інтенсивність перевірки елементів під час ймовірності відкликання при пороговому значенні m^{th} на c exp $((t-s)/\zeta)$ в момент часу s. Базова порогова лінія має аналогічна евристиці описаній раніше. Вона передбачає, коли учень майже забуває елемент та вносить цей елемент до розкладу наступної перевірки. Ми могли б порівняти цей алгоритм з методом Редді, але не будемо, тому що той алгоритм спеціально зроблений для системи Лейтнера. Що передбачає роздільний набір значень швидкості забування, а отже зовсім не підходить для наших досліджуваних параметрів.

Навіть, якщо фактичне втягування неможливе для перевірки продуктивності алгоритму, розуміння таких налаштувань призводить до іншого значного експерименту. За даними відомого навчального проекту Duo, учні доволі часто не пропускають рекомендований час навчання, що означає, пари: елемент – користувач будуть ближче до єдиного, ніж запам’ятовування або граничне значення. Це справедливо і навпаки, що показано на рисунку 2.1.

На рисунку показані приклади пар елемент - учень, які відповідають часу перевірки та мають високу ймовірність запам’ятовування – перший графік. Розклад на основі порога – це другий графік. І нарешті, однаковий розклад перевірки - третій.

Кожний графік має свічки, які відповідають сесії тестування з зеленим кольором, якщо успішно та червоним, якщо невдало. Час t = 0 – це час, коли учень вперше побачив вивчаємий елемент в наборі даних, що також може бути першою подією тестування. Якщо учень більш успішніше запам’ятовує елемент, то час між сесіями перевірки зростає, і навпаки, при кожній невдачі сесії становляться частішими.

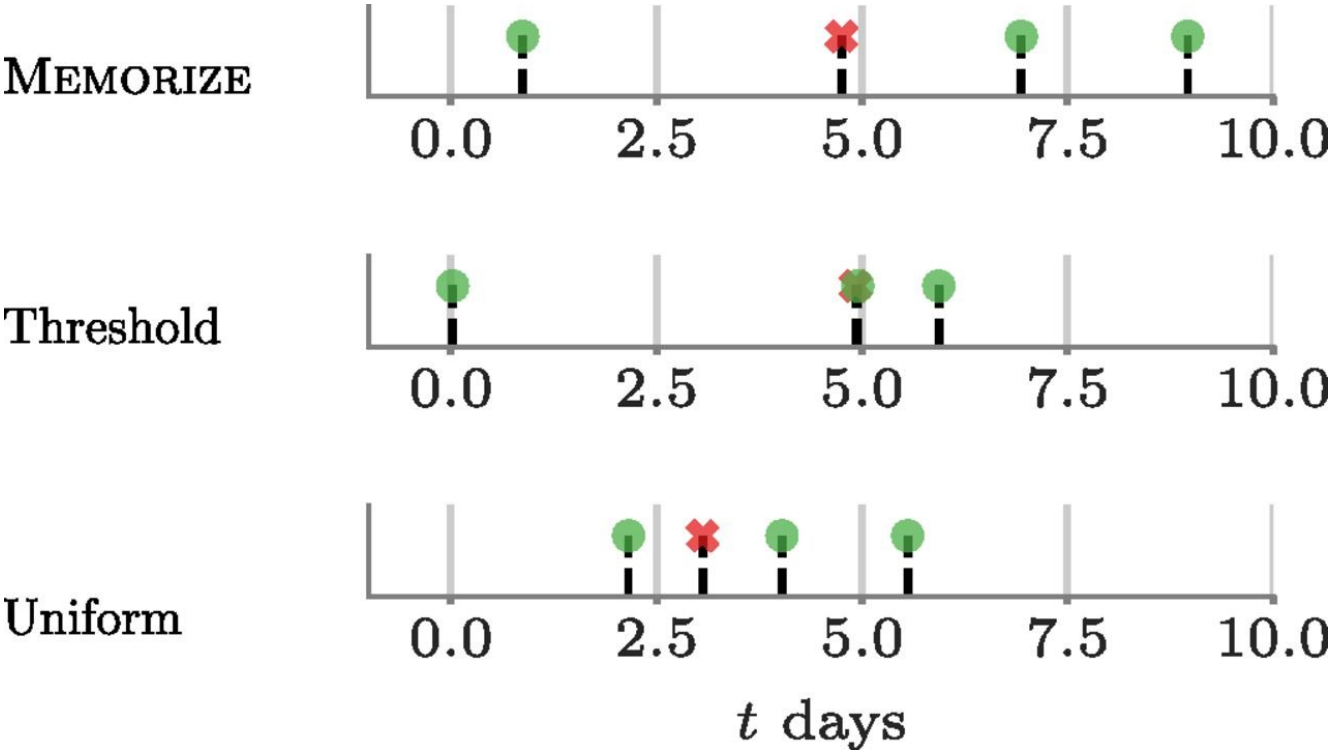

Рис. 2.1 – Графіки різних пар елемент – учень до часу

В наслідок цього, ми можемо зробити висновок, що потрібно кожну пару призначити деякій контрольній групі або групі обробки (тобто пороговій або однаковій).

Для обробки стійкої процедури оцінювання, ми використовуємо більш детальне розуміння, що ґрунтується на (*i*) ймовірних порівняннях для визначення

ретельності учня слідувати графіку тестування під час всіх перевірок, крім останньої в послідовності перевірки, тобто $e_1, \ldots, e_{n-1}$ в послідовності n перевірок, та (ii) якісний показник, емпірична швидкість забування $\hat{n}$. Але вона повинна отримати оцінку за допомогою останньої перевірки e_n (та інтервалу затримки $t_n - t_{n-1}$) кожної черговості перевірки, а також не залежить від точного вибору моделі пам'яті.

2.2.3 Підсумки

Для початку, ми повинні згрупувати пари елемент – учень за кількістю перевірок n та періоду навчання, тобто $t_{n-1} - t_1$. Потім для кожного шаблону згадування ми створюємо групи контролю (threshold і uniform) та обробки (memorize). Далі, обраховуємо емпіричну частоту забування для кожної перевірочної сесії у групі.

На рисунку 2.2 показано графіки результуючого підсумку для послідовностей, які включають від трьох до семи навчальних сесій.

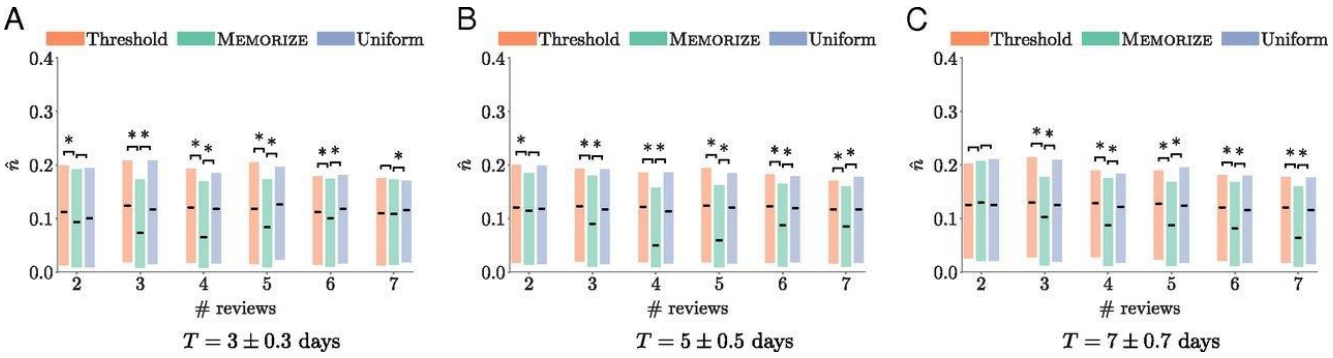

Рис. 2.2 – Графіки результуючого підсумку на відрізці від трьох до семи днів

Результати показують, що графік memorize має невелику перевагу порівнюючи з базовими показниками, заснованими на графіках threshold та uniform. Також при збільшені періоду навчання, збільшується кількість перевірок, при яких графік memorize досягає найбільшої переваги. З цього, ми можемо зробити

висновок, що ця перевага є наслідком упередженості через складність елементу вивчення.

Як видно з графіків, середня емпірична частота забування для двадцяти п’яти відсотків найліпших елементів: memorize графік – ймовірність, uniform графік – розклад перевірок та threshold графік – розклад перевірок для різних елементів n та різними навчальними періодами $T = t_{n-1} - t_1$. Квантилі двадцять п’ять та сімдесят п’ять відсотків вказані у прямокутниках, середні значення вказані на суцільних лініях, варто звернути увагу, що нижчі показники говорять про кращу продуктивність.

Далі ми робимо наступний висновок: щоразу, коли користувач приділяє більше уваги memorize, його продуктивність виростає. Якщо розглядати детальніше, для кожного користувача, який має як мінімум сімдесят сесій навчання з періодом $T = 8 \pm 3.2$ дня.

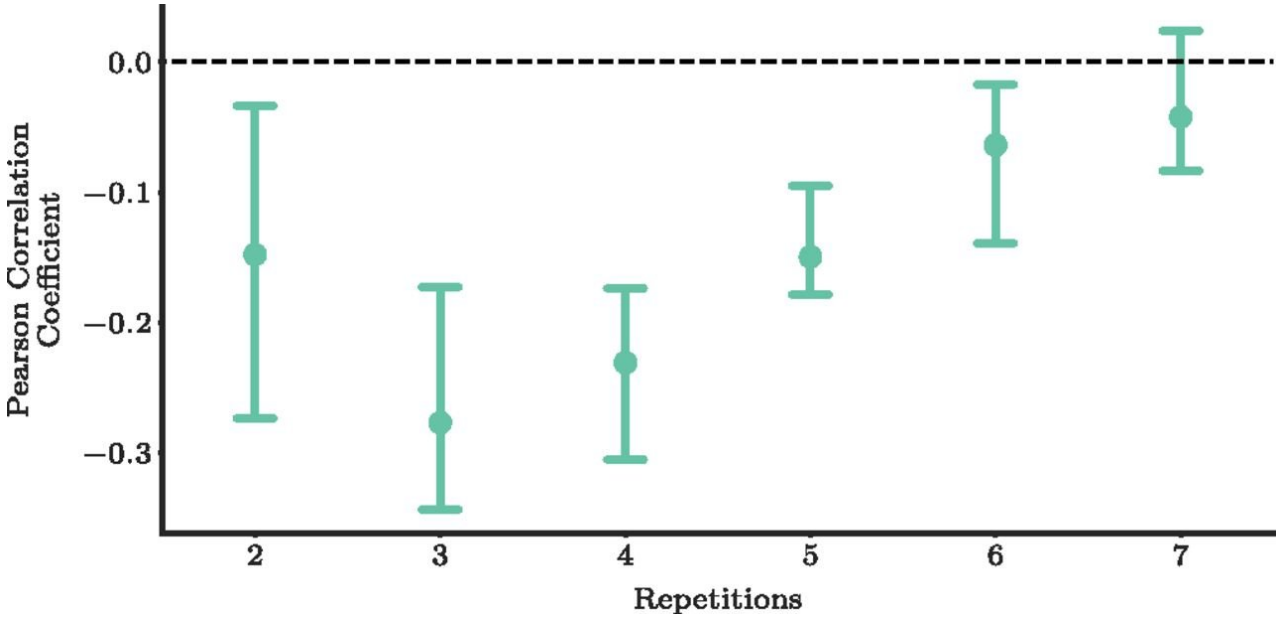

Рис. 2.3 – Графік коефіцієнту кореляції Пірсона до кількості повторень

Нам потрібно обчислити коефіцієнт кореляції Пірсона між емпіричною частотою забування і значеннями логарифмічної ймовірності, а також вибрати нижні та верхні п’ятдесят відсотків послідовностей перевірки з точки зору логарифмічної ймовірності при memorize.

З рисунка 2.3 зрозуміло, що учні, які приділяють більше уваги memorize, в середньому, досягають нижчих емпіричних показників забування.

Якщо звернутися до з системи Лейтнера [9], то можна дізнатися про існуває багатьох різних алгоритмів рознесеного повторення. Але розробка адаптивних алгоритмів основаних на даних з доказовими гарантіями було недостатньо.

У цій роботі ми розглянули основну частину для моделювання власної розробки алгоритму інтервального повторення з гарантіями для адаптації учнів. Така структура моделювання, що використовує структуру процесів із зазначеними часовими точками і з переходами, розглядає розробку алгоритмів інтревального повторення як задачу оптимального управління такими стрибкоподібними SDE. Вона надає надпотужний інструмент для пошуку алгоритмів інтервального повторення, тому що така структура не залежить від конкретних варіантів моделювання.

3 МОДЕЛЬ ІНТЕРВАЛЬНОГО ПОВТОРЕННЯ

3.1 Вибір алгоритму

Основою моделі завжди є алгоритм. Проаналізувавши існуючі алгоритми а також перевіривши деякі з них [10], було прийнято рішення використати алгоритм регресійного напіврозпаду. Цей алгоритм дуже добре підходить для конкретних цілей кваліфікаційної роботи.

3.2 Опис моделі

У цьому розділі буде описано регресію періоду напіврозпаду, опираючись на теорію психології та використовуючи сучасні методи машинного навчання. Якщо ми говоримо про теорію пам’яті, то варто згадати модель Еббінга-Хауса, або криву забування [11]. Вона передбачає, що пам'ять забувається експоненціально з плином часу:

$$p = 2^{-\Delta/h}, \tag{3}$$

де p – це ймовірність правильного запам'ятовування слова, яка є функцією Δ – часу затримки з моменту останнього запам’ятовування слова, та h – показник сили довготривалої пам’яті користувача або період напіврозпаду.

Що таке час затримки, або більш відомий як ефект затримки? Ефект затримки був введений у 1970 році та говорить, що людина краще запам’ятовує матеріал, якщо відстань між навчальними практиками поступово збільшується [12].

На рисунку 3.1 представлена крива забування при періоді напіврозпаду h = 1. Розглянемо випадки:

1. Якщо Δ = 0. Слово практикувалося зовсім недавно, отже $p = 2^0 = 1.0$, що відповідає ідеї про те, що воно свіже в пам'яті і має бути правильно згадано незалежно від періоду напіврозпаду.

2. Якщо Δ = h. Тобто час затримки дорівнює періоду напіврозпаду, тому $p = 2^{-1} = 0.5$, що означає учень знаходиться на грані згадування слова.
3. Якщо Δ > h. Щодо періоду напіврозпаду, слово не практикувалося довгий час, тому ймовірно, було забуто, наприклад, p ≈ 0.

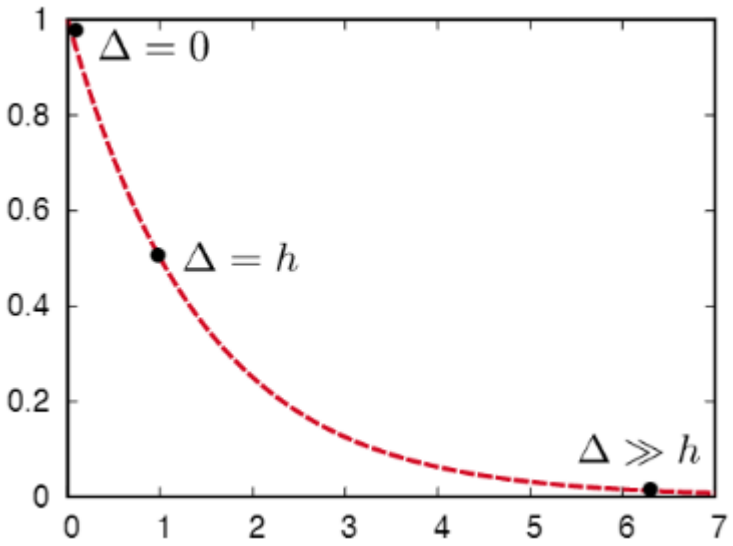

Рис. 3.1 – Модель Еббінга-Хауса. Прогнозована швидкість згадування, представлена у функції часу затримки Δ та періоду напіврозпаду h = 1

Нехай деякий вектор ознак позначається x, який сумує попереднє знайомство користувача з конкретним словом, та нехай Θ – це вектор параметрів, що містить ваги, відповідні *x*, кожній змінній ознак. Якщо ми візьмемо звичайну практику дослідження ефекту затримування та інтервалів, то отримаємо припущення, що період напівозапу повинен експоненціально збільшуватися з кожним повторним згадуванням. Розрахунковий період напіврозпаду позначимо $\hat{h}\Theta$, та він заданий формулою:

$$\hat{h}\Theta = 2^{\Theta x}, \tag{4}$$

Взагалі-то, як окремий випадок (4) можна представити алгоритми Лейтнера та Пімслера з використанням специфічних ваг, які ми підібрали власноруч.

Що ж до наших конкретних цілей, то нам потрібно емпірично відповідати Θ – даним трасування навчання, а також не забувати про величезний набір функцій. Припустимо, ми маємо деяку вибірку даних $\mathfrak{D} = \{\langle p, \Delta, x \rangle_i\}_{i=1}^{D}$. Вона складається з навчальних сесій користувачів. Кожний екземпляр вибірки складається зі: $p -$ спостережуваної швидкості запам'ятовування, $\Delta -$ часу затримки з моменту останнього перегляду слова та $x -$ вектора функцій, спеціально зробленого для персоналізації процесу навчання. Нам потрібно знайти якомога кращі ваги моделі Θ^*, щоб мінімізувати затрати наступної функції (5):

$$\Theta^* = \arg\min_{\Theta} \sum_{i=0}^{D} \ell(\langle p, \Delta, x \rangle_i; \Theta) \qquad (5)$$

Для ілюстрування даної функції, потрібно розглянути рисунок 3.2, на якому показано графік вивчення користувачами слів протягом 30ти днів.

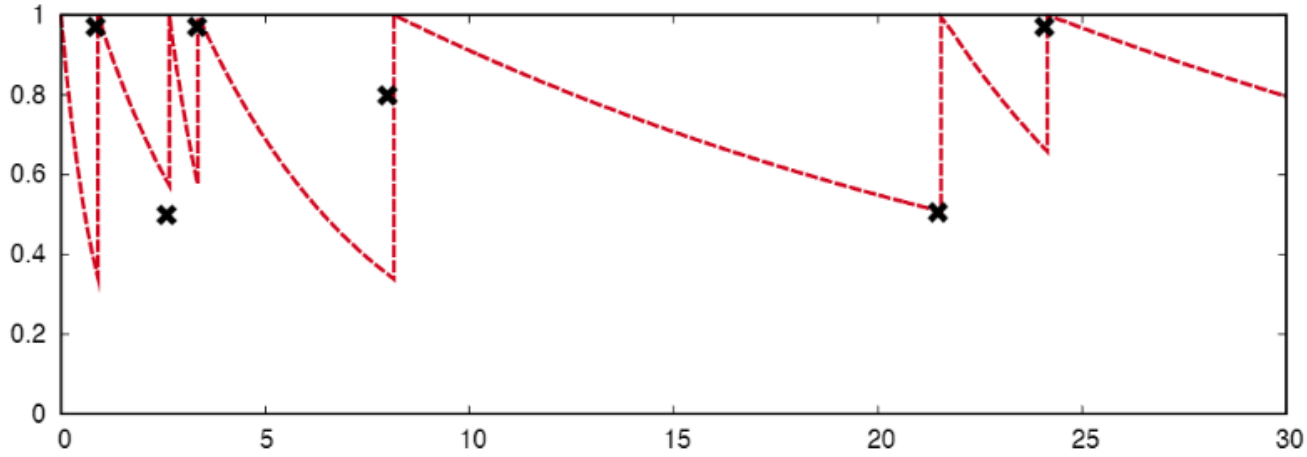

Рис. 3.2 – Вивчення слів користувачем за останні 30 днів. Хрестики позначають спостережувану частоту згадування p для кожної сесії, а для відповідності моделі прогнозування $\hat{p}\Theta$ (пунктирні лінії) цим точкам, спрямована регресія періоду напіврозпаду

Кожен хрестик вказує на екземпляр даних: вертикальне положення – це спостережувана частота згадування p для кожної сесії, а горизонтальна відстань

між точками – це час затримки Δ між сесіями. Об'єднуючи формули (3) та (4), ми отримуємо модель прогнозування:

$$\hat{p}\Theta = 2^{-\Delta/\hat{h}\Theta}, \tag{6}$$

яка відображується на рисунку 3.2 у вигляді пунктирної лінії з плином часу (яка відновлюється до 1.0 після кожної експозиції, бо Δ = 0). Для стикування результатів прогнозованих кривих забування, використовується функція втрат при навчанні (6).

В якості квадратичної функції затрат було вибрано L_2 регуляризовану квадратну функцію втрат [13], яка в своїй основній формі задається наступною формулою:

$$\ell(\mathbf{x}; \Theta) = (p - \hat{p}\Theta)^2 + \lambda||\Theta||_2^2, \tag{7}$$

де $\mathbf{x} = \langle p, \Delta, x \rangle$ є скороченням для тренувального екземпляра даних, і λ – це параметр, керуючий терміном регуляризації і допомагає запобігти перенавчання.

Для розуміння, потрібно описати що таке функція затрат (7). Для навчання використовується ітеративний підхід. Тобто кожний наступний результат залежить від значення попередніх, або висловлюючись терміном машинного навчання, навчається. А як зрозуміти наступний результат кращий чи гірший за попередній? Тут і вступає функція затрат. Вона обчислює середній штраф за всіма навчальними прикладами. А отже наша ціль полягає у підборі таких коефіцієнтів, щоб функція затрат була мінімальному. Чим менше результат функції затрат, тим кращі коефіцієнти.

Наступним кроком розглянемо що таке L_2 регуляризація. Регуляризація – це така форма регресії, що зводить оцінки коефіцієнтів до нуля. Іншими словами, для уникнення ризику перенавчання, цей метод блокує звертання для складнішої та більш гнучкої моделі.

В загальній практиці використовують L_1 та L_2 регуляризації. Регресійна модель, L_1 називається регресією Лассо, а модель, що використовує L_2, називається регресією Ріджа. В нашій моделі ми використовуємо регресію Ріджа.

На практиці, з'ясувалося корисним оптимізувати період напіврозпаду h, окрім спостережуваної частоти згадування p. Так як визначити «справжній» період напіврозпаду даного слова у пам'яті користувача не представляється можливим, ми алгебраїчно його наближаємо за допомогою функції (3), використовуючи p і Δ. Ми розв'язуємо для $h = \frac{-\Delta}{\log_2(p)}$ та використовуємо фінальну функцію затрат:

$$\ell(\mathbf{x}; \Theta) = (p - \hat{p}\Theta)^2 + \alpha(h - \hat{h}\Theta)^2 + \lambda||\Theta||_2^2, \qquad (8)$$

де α – параметр у загальній цільовій функції навчання, використаний для управління важливості періоду напіврозпаду. Тепер, оскільки функція є гладкою по відношенню до Θ, ми можемо досить просто підібрати ваги, використовуючи історію вивчення слів користувачем, за допомогою градієнтного спуску.

Варто зауважити, що сам алгоритм регресії взятий з системи Duolingo, який ви можете переглянути в додатку Б.

3.3 Тренувальні дані

Машинне навчання в більшій мірі опирається на статистику. Тому при навчанні нашої моделі ми повинні надати їй значну статистичну випадкову вибірку в якості тренувальних даних. Чому це так важливо? При наданні не набору не випадкових даних, наша модель може використати якість шаблони машинного навчання, котрих насправді немає. Також важливим пунктом є розмір навчального набору. Причина дуже добре описана в законі великих чисел [14].

За законом великих чисел в статистиці та теорії ймовірносте, при збільшенні розміру набору середнє значення наближається до середнього значення всієї вибірки. Але це не означає, що така група вибірок буде відображати справжній

характер набору, особливо для малих наборів. При цьому це підштовхує до висновків, що при відхиленні набору від істинного середнього, закон не буде гарантувати, що послідовні вибірки змістять середнє значення спостережень до середнього значення всього набору, на відміну від іншого закону - помилки гравця.

Отже, при недостатній кількості вхідних даних, ми не тільки не отримаємо достатньо інформації, а навіть можемо прийти до неправильних висновків на основі цих даних. Наприклад, проаналізувавши дані користувачів, які навмання вибрали свою рідну мову для навчання, то ми отримаємо зовсім не ті результати на які сподівалися.

Перейдемо до самого набору даних. Тренувальний набір даних взятий з великого озеру даних Гарварду, а саме Harvard dataverse. Набір складається з csv файлів та має більш ніж з тринадцяти мільйонів строк, взятих з різних популярний сервісів по вивчанню іноземних мов таких як Memorize, Lingualeo, Duolingo. Набір поділений на навчальні сесії різних користувачів, та включає знеособлені дані користувача, а саме зашифрований id для конфіденціальності.

Перейдемо до структури файлу. На рисунку 3.3 показано приклад вхідного набору даних.

```
p_recall,timestamp,delta,user_id,learning_language,ui_language,lexeme_id,lexeme_string,history_seen,history_correct,session_seen,session_correct
1.0,1362076081,27649635,u:FO,de,en,76390c1350a8dac31186187e2fe1e178,lernt/lernen<vblex><pri><p3><sg>,6,4,2,2
0.5,1362076081,27649635,u:FO,de,en,7dfd7086f3671685e2cf1c1da72796d7,die/die<det><def><f><sg><nom>,4,4,2,1
1.0,1362076081,27649635,u:FO,de,en,35a54c25a2cda8127343f6a82e6f6b7d,mann/mann<n><m><sg><nom>,5,4,1,1
0.5,1362076081,27649635,u:FO,de,en,0cf63ffe3dda158bc3dbd55682b355ae,frau/frau<n><f><sg><nom>,6,5,2,1
1.0,1362076081,27649635,u:FO,de,en,84920990d78044db53c1b012f5bf9ab5,das/das<det><def><nt><sg><nom>,4,4,1,1
1.0,1362076081,27649635,u:FO,de,en,56429751fdaedb6e491f4795c770f5a4,der/der<det><def><m><sg><nom>,4,3,1,1
1.0,1362076081,27649635,u:FO,de,en,1bacf218eaaf9f944e525f7be9b31899,kind/kind<n><nt><sg><nom>,4,4,1,1
1.0,1362082032,444407,u:dDwF,es,en,73eecb492ca758ddab5371cf7b5cca32,bajo/bajo<pr>,3,3,1,1
1.0,1362082044,5963,u:FO,de,en,76390c1350a8dac31186187e2fe1e178,lernt/lernen<vblex><pri><p3><sg>,8,6,6,6
0.75,1362082044,5963,u:FO,de,en,7dfd7086f3671685e2cf1c1da72796d7,die/die<det><def><f><sg><nom>,6,5,4,3
0.888888888889,1362082044,5963,u:FO,de,en,35a54c25a2cda8127343f6a82e6f6b7d,mann/mann<n><m><sg><nom>,6,5,9,8
0.8,1362082044,5963,u:FO,de,en,0cf63ffe3dda158bc3dbd55682b355ae,frau/frau<n><f><sg><nom>,8,6,5,4
0.8,1362082044,5963,u:FO,de,en,84920990d78044db53c1b012f5bf9ab5,das/das<det><def><nt><sg><nom>,5,5,5,4
1.0,1362082044,5963,u:FO,de,en,56429751fdaedb6e491f4795c770f5a4,der/der<det><def><m><sg><nom>,5,4,5,5
1.0,1362082044,5963,u:FO,de,en,1bacf218eaaf9f944e525f7be9b31899,kind/kind<n><nt><sg><nom>,5,5,3,3
1.0,1362082130,77,u:dDwF,es,en,73eecb492ca758ddab5371cf7b5cca32,bajo/bajo<pr>,5,5,1,1
0.0,1362082194,150,u:FO,de,en,84920990d78044db53c1b012f5bf9ab5,das/das<det><def><nt><sg><nom>,10,9,1,0
1.0,1362082194,150,u:FO,de,en,35a54c25a2cda8127343f6a82e6f6b7d,mann/mann<n><m><sg><nom>,15,13,1,1
0.0,1362082194,6032,u:FO,de,en,4fcb6bb8e44d7b618999721071862827,mädchen/mädchen<n><nt><sg><nom>,1,1,1,0
1.0,1362082194,6032,u:FO,de,en,a6834806c43ea1be9eb3e4fdae6f98db,apfel/apfel<n><m><sg><nom>,1,1,1,1
```

Рис. 3.3 – Приклад тренувального набору даних

Як ми бачимо з рисунка, вхідні дані поділені на дванадцять колонок: p_recall, timestamp, delta, user_id, learning_language, ui_language, lexeme_id, lexeme_string, history_seen, history_correct, session_seen, session_correct.

Опишемо їх:

- p_recall – частка вправ цієї сесії, в яких слово були успішно згадано, у діапазоні від 0 до 1.0, де 0 згадано у 0% випадків, а 1.0 – у 100%;
- timestamp – часова мітка даної сесії;
- delta – час у секундах з останньої сесії, де було це слово;
- user_id – знеособлене id учня, який проходив сесію;
- learning_language – мова, яку вивчає учень;
- ui_language – мова, яку знає учень;
- lexeme_id – id слова;
- lexeme_string – слово та його теги;
- history_seen – загальна кількість разів, коли учень побачив це слово під час урока;
- history_correct – загальна кількість разів учень успішно згадав слово під час урока;
- session_seen – загальна кількість разів, коли учень побачив слово під час сесії;
- session_correct – загальна кількість разів учень успішно згадав слово під час сесії.

Теги слова. В колонці lexeme_string представлено не тільки слово, а й теги цього слова. Вони використовуються для кращої класифікації форм слова. Так наприклад, <num> означає множинну форму слова, <prn> означає що це займенник і тд. Деякі теги мають таку форму <*…>. Наприклад <*sf> відповідає загальній лексемі без будь-яких поверхневих форма.

Повний список тегів та їх розшифровку знаходиться у текстовому файлі разом з тренувальним набором даних.

3.4 Практична частина

Так як був вибраний алгоритм періоду напіврозпаду, то давайте порівняємо його з іншим алгоритмом рознесеного повторення.

Перейдемо до мови програмування. Так як ми маємо досить великий обсяг даних, було обрано мову програмування python 3.9, а точніше мову Pypy, спеціально розроблену для обчислення складних даних. Pypy основується на python, а отже ніяких додаткових знать під час програмування використовувати не потрібно.

Детальніше про алгоритми. Для порівняння було вибрано три категорії алгоритмів рознесеного повторення:

- регресія періоду напіврозпаду (HLR – у скобках позначено скорочення, використане у таблиці, приведеній нижче). Щоб отримати більш точні результати ми розглянемо наступні варіанти: з періодом напіврозапу та без нього у функції затрат (-h), та з ознаками лексеми і без них. (-lex);
- Лейтнер і Пімслер, які є базою, а також окремими випадками алгоритму періоду напіврозпаду. У цьому випадку використано фіксовані ваги, які були заздалегідь обчислені;
- логістична регресія (LR), є ще одним стандартом машинного навчання. Ми використовуємо два варіанти: з ознаками лексеми і без них (-lex). Для моделей логістичної регресії було також підібрано час затримки $x\Delta$ в якості додаткової функції.

Для початку, потрібно було визначати початкові значення, то було вирішено взяти перший мільйон строк навчальної вибірки для коректування параметрів алгоритму. В кінці-кінців, параметри прийняли такі значення $\lambda = 0.1, \alpha = 0.01$, а також швидкість навчання $\eta = 0.001$. Однакові параметри були використані для алгоритму періоду напіврозпаду та логістичної регресії. Алгоритми Пімслера та Лейтнера не потребують у тренувальному наборі, так як мають фіксовані ваги.

Результати навчання представлені у таблиці 3.1 з використання описаного у розділі 3.3 набору даних. При цьому перші дев'яносто відсотків були використанні для навчання, а десять відсотків – для тестування.

Таблиця 3.1 – Результати оцінки з використанням тренувального набору даних. Стрілка (↓) вказує, що кращі результати нижчі, а (↑) – що вищі. При цьому статистично важливі (p < 0.001) відзначені знаком *

Модель	MAE↓	AUC↑	COR_h↑
HLR	0.129*	0.511*	0.203*
HLR -lex	0.129*	0.507*	0.135*
HLR -h	0.350	0.498*	-0.074*
HLR -lex -h	0.350	0.498*	-0.07*
Лейтнер	0.235	0.526*	-0.105*
Пімслер	0.445	0.482*	-0.112*
LR	0.211	0.491*	
LR -lex	0.211	0.492*	
Константа $\bar{p} = 0.859$	0.175		

Для всебічного порівняння, ми розглядаємо кілька оціночних показників:

- середня абсолютна помилка (MAE) вимірює, наскільки близько прогнози до їх спостережуваних результатів: $\frac{1}{D}\sum_{i=1}^{D}|p - \hat{p}\Theta|_i$;
- критерій суми рангів Вілкоксона (AUC). Це показний якості ранжирування. В даній ситуації від представляє ймовірність моделі оцінити випадкове правильно згадане слово як ймовірне, порівнюючи з випадковим неправильно згаданим словом. У нашій моделі AUC використовується для допомоги в оцінці рейтингу;
- кореляція періоду напіврозпаду (COR_h) представляє собою кореляцію Спірмена-Ранка між $\hat{h}\Theta$ та алгебраїчною оцінкою h. Це ще один показний якості ранжирування.

Для всіх трьох показників, регресії періоду напіврозпаду з ознаками тегів лексем є найкращим (або ж другим кращим) підходом, потім слідує регресія періоду напіврозпаду без тегів лексем.

Насправді, ці дві регресії є єдиними в яких MAE знаходиться нижче ніж базовий постійний прогноз середньої швидкості згадування в навчальних даних (таблиця 3.1, останній рядок – 0.175). Крім того дані варіанти регресії періоду напіврозпаду є також єдиними з позитивним значенням COR_h, що досить логічно, тому що ці дві регресії є єдиними двома, які спеціально оптимізуються для нього.

Якщо оцінити вплив тега лексем -lex, то ми побачимо обмежений вплив на результат.

А ось функція затрат -h у регресії періоду напіврозпаду показує суттєві результати. Без цього показника MAE збільшується більш ніж у два рази, також варіанти -h, як ми бачимо, гірше інших базових показників, щонайменше за одним із них.

Метод Лейтнера, що не став несподіванкою, дійсно виявив найвищі значення AUC серед випробуваних алгоритмів. Але варто зауважити, що перші два варінати регресії періоду напіврозпаду не сильно гірші, крім того вони також знижують MAE в порівнянні з Лейтнером як мінімум на сорок відсотків.

Розглянемо критерій суми рангів Вілкоксона. Значення 0.5 передбачає випадкове вгадування [15], тому значення AUC можуть здаватися низькими. Для початку, це може бути пов’язано з зашумлевим завданням прогнозування, але також і з обмеженим діапазоном: $\bar{p} = 0.859$, тому більшість слів запам'ятовується правильно, а прогнози, як правило, високі.

3.5 Аналіз вагових коефіцієнтів

При тренуванні, регресія періоду напіврозпаду також аналізує складність слів та тегів лексем, яка притаманна концепціям. Простіші слова та теги мають позитивну вагу, тому що рідша сесія, обумовлена тривалішим періодом напіврозпаду. В той час як складніші концепції мають негативну вагу (більш часта сесія, обумовлена коротшим періодом напіврозпаду).

Я ми можем побачити, у таблиці 3.2 приведені моделі регресії періоду напіврозпаду для кількох слів з французької, німецької, англійської та іспанської

мов. Ваги, що більше за нуль асоціюються з простими короткими або загальновідомими словами, тому розумно що їх простіше запам’ятати. Що ж до вагів, які менше за нуль, то такі слова складні, мають неправильну форму, пов’язані з минулим або майбутнім словом або винятки. Такі слова значно складніше запам’ятовувати.

Таблиця 3.2 – Приклад вагових коефіцієнтів тегів лексем для французької (FR), німецької (DE), англійської (EN) та іспанської(ES) мов

Мова	Слово	Тег	Θ_k
FR	fallait	fallour. V.IMPERF .P3.SG	-0.0036
FR	ceci	ceci.PN .NT	-0.0021
FR	suis	^etre.V.PRES .P1.SG	0.0067
FR	visite	visiter. V.PRES .P3.SG	0.0101
DE	war	sein. V.IMPERF .P1.SG	-0.0594
DE	den	der. DET .DEF .M.SF.ACC	-0.0002
DE	sprechen	sprechen. V.INF	0.0009
DE	Baby	Baby. N.NT.SG.ACC	0.0014
EN	writing	write. V.PRESP	-0.0103
EN	rose	rise. V.PST	-0.0001
EN	circle	circle. N.SG	0.0001
EN	camera	camera. N.SG	0.0041
ES	quedado	quedar.V.PP.M.SG	-0.0998
ES	est’a	estar. V.PRES .P3.SG	-0.0002
ES	como	comer. V.PRES .P1.SG	0.0008
ES	liberal	liberal. ADJ .SG	0.0047

Отже ми отримали зручний інструмент початкового налаштування системи. Все дуже просто, чим більше коефіцієнт ваги, тим простіше слово, чим нижче тим слово складніше. Дані вагові коефіцієнти дають уявлення про аспекти іноземних мов для їх вивчення.

3.5.1 Аномальні результати

Під час тренування було проведено декілька експериментів над тренувальними даними для отримання вагових коефіцієнтів. Приклад можна розглянути на рисунку 3.4. Як ми бачимо, крім коефіцієнтів лексем, ми отримали коефіцієнти для правильного згадування слів та неправильного. Правильне згадування «right» повинно бути більше нуля, як і показано на рисунку.

```
right   0.3727
wrong   0.1666
time    0.0054
bias    0.3343
en:what/what<prn><itg><m><sp>   0.0090
pt:alô/alô<ij>  0.0109
fr:thé/thé<n><m><sg>    0.0080
en:horse/horse<n><sg>   0.0297
en:a/a<det><ind><sg>    -0.0395
en:tomorrow/tomorrow<adv>       0.0041
en:also/also<adv>       0.0028
es:lee/leer<vblex><pri><p3><sg> 0.0144
es:queso/queso<n><m><sg>        0.0297
en:an/a<det><ind><sg>   0.0099
it:caffè/caffè<n><m><sp>        0.0110
en:blue/blue<adj>       0.0075
```

Рис. 3.4 – Приклад вихідних коефіцієнтів, використовуючи регресію періоду напіврозпаду з лексемами

А тепер розглянемо такі ж коефіцієнти, але для регресії періоду напіврозпаду без лексем. Як показано на рисунку 3.5, коефіцієнт «right» для правильного згадування від'ємний. Але чому так вийшло? Давайте проаналізуємо графік першого мільйону строк, основаного на тренувальних даних, описаного в розділі 3.3, який показаний на рисунку 3.6 та спробуємо зрозуміти.

На графіку є два стовпця: history_seen, який показує загальну кількість правильно та неправильно згаданих слів, тобто кількість разів, коли слово було переглянуте, та стовпець p_recall, який показує оцінку за сеанс для кожного слова.

Спочатку це може збити з пантелику, як збило і мене. Я звернувся до джерела тестових даних та зауважив наступну річ. Тестові дані, можуть складатися з багаторазового перегляду одного і того ж слова в одному тесті або сеансі.

```
right   -0.0133
wrong   -0.2190
bias    7.5371
en:am/be<vbser><pri><p1><sg>    0.0113
en:book/book<n><sg>     0.0090
fr:femme/femme<n><f><sg>        0.0076
pt:corta/cortar<vblex><pri><p3><sg>     0.0017
en:use/use<vblex><pres> 0.0038
es:segundos/segundo<n><m><pl>   0.0035
es:es/ser<vbser><pri><p3><sg>   0.0106
en:soldiers/soldier<n><pl>      0.0023
de:lernt/lernen<vblex><pri><p3><sg>     0.0069
en:clothes/clothes<n><pl>       0.0046
fr:une/un<det><ind><f><sg>      0.0072
es:vieja/viejo<adj><f><sg>      0.0029
it:torte/torta<n><f><pl>        0.0022
es:un/uno<det><ind><m><sg>      0.0080
```

Рис. 3.5 – Приклад вихідних коефіцієнтів, використовуючи регресію періоду напіврозпаду без лексем

Навіть якщо припустити, що слова з history_seen більше тисячі являються фальсифікованими або помилковими, хоча вони також можуть бути реальними, але я сумніваюсь, зверніть увагу, наскільки широкий діапазон p_recall при високих значеннях history_seen. Припустимо, користувач переглядав слово сотні разів. Іноді він може забути слово, іноді зробити помилку, а система не розпізнає такі нюанси, тому і карає користувача за його неуважність.

Після збору даних їх потрібно збалансувати і єдиний спосіб, яким регресія може це зробити на такому широкому діапазоні p_recall – це встановити майже нульові ваги для «right» та «wrong» значень.

Це наводить на думку, що більша частина прогностичної сили регресії походить від «bias», який дає середню абсолютну помилку (MAE) в оцінці за сесію

для регресії прогнозування, яка завжди повертає константу 0.859. Це призводить до того, що така оцінка перевершує всі інші, крім регресії періоду напіврозпаду з лексемами та без них (-lex), що може допомогти переконати нас у наведеному вище аналізі.

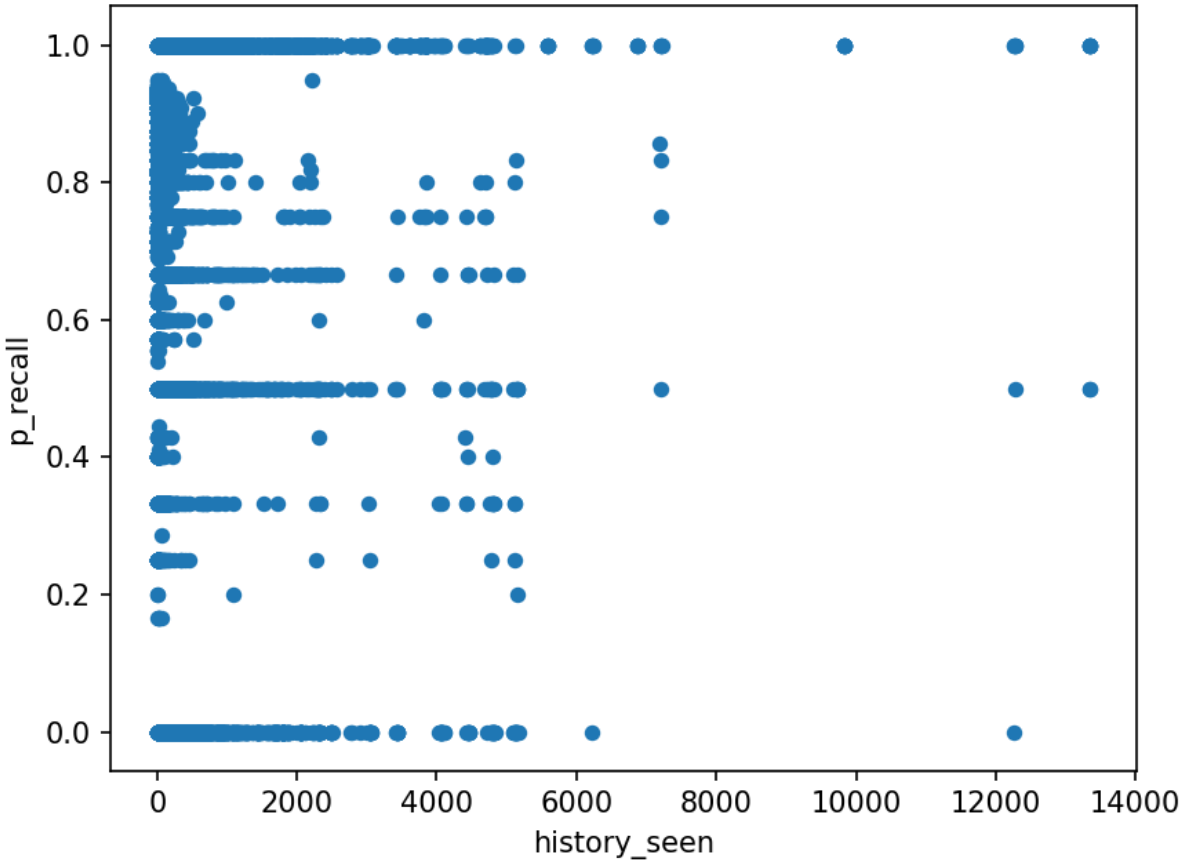

Рис. 3.6 – Графік першого мільйона строк у вхідному файлі

Якщо даний аналіз правильний, то це може означати, що оптимізація для ваг призводять до помилки в формулах оцінки за сесію $(p - \hat{p}\Theta)^2$ та передбачувані періоду напіврозпаду $(h - \hat{h}\Theta)^2$, як використовуються у фінальній функції затрат (8).

Якщо привезти і ту і другу формулу до нуля, то для кінцевої моделі це проблематично.

Це дуже схоже на Баєсове висновування. Що ж, після сотень або навіть тисячі слів користувач повинен був уже запам’ятати слово. Якщо ж ні, то це не повинна бути проблема регресії, ми повинні скорегувати модель, або відкинути такі дані.

Як варіант, можна використати бібліотеку Ебісу, яка й намагається це зробити.

Ви можете заперечити, що при наявності достатньої кількості даних це має відбутися автоматично. Враховуючи, що термін «bias» набагато більший, ніж ваги двох інших у регресії періоду напіврозпаду без лексем (-lex), може виявитись, що цього не відбувається. Крім того, покладатися на велику кількість даних для отримання правильної відповіді (у нашому випадку позитивну і, можливо, значну вагу для правильного), також проблематично, тому що ми покладаємося тільки на правильність вхідних даних.

4 ФОРМУВАННЯ ВИМОГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Моделювання програмної системи це важливий етап у створенні майбутнього продукту. Під час цього етапу а також проектування програмної системи було прийнято рішення розділити систему на наступні складові:

– сервер та база даних, до якої звертаються система. Ця частина створена для обробки запитів, а також прогнозування відповідно до моделі описаної у третьому розділі;
– клієнтська частина, або її ще називають фронтенд. Призначена для візуального відображення результатів роботи серверу.

А тепер давайте розглянемо детальніше кожну з частин.

4.1 Загальні відомості до системи

То ж, як уже згадувалося вище, програмна система складається з:

– сервер та база даних, до якої звертаються система оброблює запити клієнта, а саме: запит на авторизацію, запит на створення та редагування уроків, запит на проходження сесії уроку.
– клієнтська частина дає змогу користувачеві зареєструватися, авторизуватися, створювати уроки зі слів, проходити сесії уроків а також отримати прогноз, забув користувач те чи інше слово чи ні.

Під час розробки даної програмної системи були використані наступні технології.

Серверна частина написана за допомогою мови програмування Php 8 [16]. Окрім цього, слід зауважити, що для виконання алгоритму моделі було використано мову програмування Python 3.9 [17].

Для клієнтської частини були використані такі мови програмування: Vue.js 3 [18], Javascript [19].

4.2 Серверна частина

Серверна частина поділена на такі підрозділи: авторизація, створення уроків, редагування уроків, проходження сесії, прогнозування забування слів. Кожен підрозділ, окрім прогнозування, використовує мову програмування Php 8. Доступ до бази здійснюється за допомогою MySQL 8 [20]. MySQL досить популярна база даних, а також, як вказує розробник, може працює з великими об’ємами даних, але навіть так, база даних не встигає витягувати записи для обробки моделі, яка виконується за допомогою мови програмування Python, тим самим створює так зване «bottleneck», тобто вузьке місце. Тому замість витягування потрібних елементів поштучно, база даних віддає весь обсяг у сирому вигляді та записує її до файлі, звідки уже Python оброблює дані.

4.3 Клієнтська частина

Клієнтська частина складається з веб сторінок, на яких користувач може зареєструватися, створити урок, відредагувати урок, пройти сесію уроку.

На сторінці створення уроку можна вибрати назву, описати тему уроку а також вибрати мову та слова для вивчення зі списку.

На сторінці редагування уроку можна відредагувати поля, які перечисленні вище.

Сторінка проходження сесії уроку складається з карток, які були заздалегідь посортовані за часом забування, або, якщо користувач ніколи не бачив таких слів, за ваговими коефіцієнтами, які описані у розділі 3.5.

5 ПРОЕКТУВАННЯ ТА АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Проектування UML програмного забезпечення

Під час проектування даного програмного забезпечення було використано веб застосунок draw.io створена діаграма Use Case.

На рисунку 5.1 можна переглянути Use case діаграму.

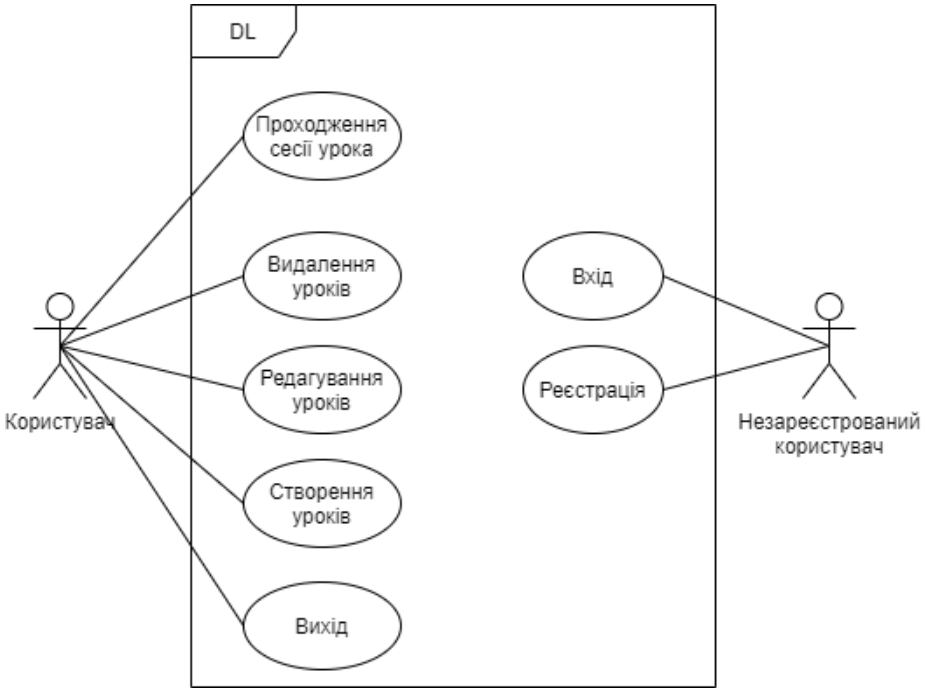

Рис. 5.1 – Діаграма варіантів використання програмного забезпечення

Діаграма представляє собою двох акторів: користувача та незареєстрованого користувача.

Незареєстрований користувач може виконувати не так багато дій, а саме: війти до системи зареєструватися. Процедура реєстрації досить проста і може бути проведена за декілька хвилин. Користувач повинен надати свою електронну почту та ввести пароль. Після цього його автоматично посилає на головну сторінку, де він стає «користувачем» у діаграмі.

Авторизований користувач може створювати уроки, редагувати їх та видаляти, проходити сесії цих уроків та виходити з системи.

Створення, перегляд та редагування уроків відбувається в розділі зі списком всіх уроків на вибраній мові. Видалити урок можна зі сторінки редагування. На сторінці перегляду уроку можна натиснути кнопку пройти урок, яка пересилає авторизованого користувача на сторінку проходження навчальної сесії. Якщо користувач хоче вийти з системи для зміни користувача, то кнопка знаходиться у верхньому меню справа.

5.2 Проектування бази даних

Для програмного забезпечення було обрано СКБД MySQL 8.

На основі аналізу моделі машинного навчання, а також вхідних тренувальних даних, що будуть оброблятися системою була побудована схема бази даних, що зображена на рисунку 5.2. Схема була спроектована у веб сервісі dbdesigner.

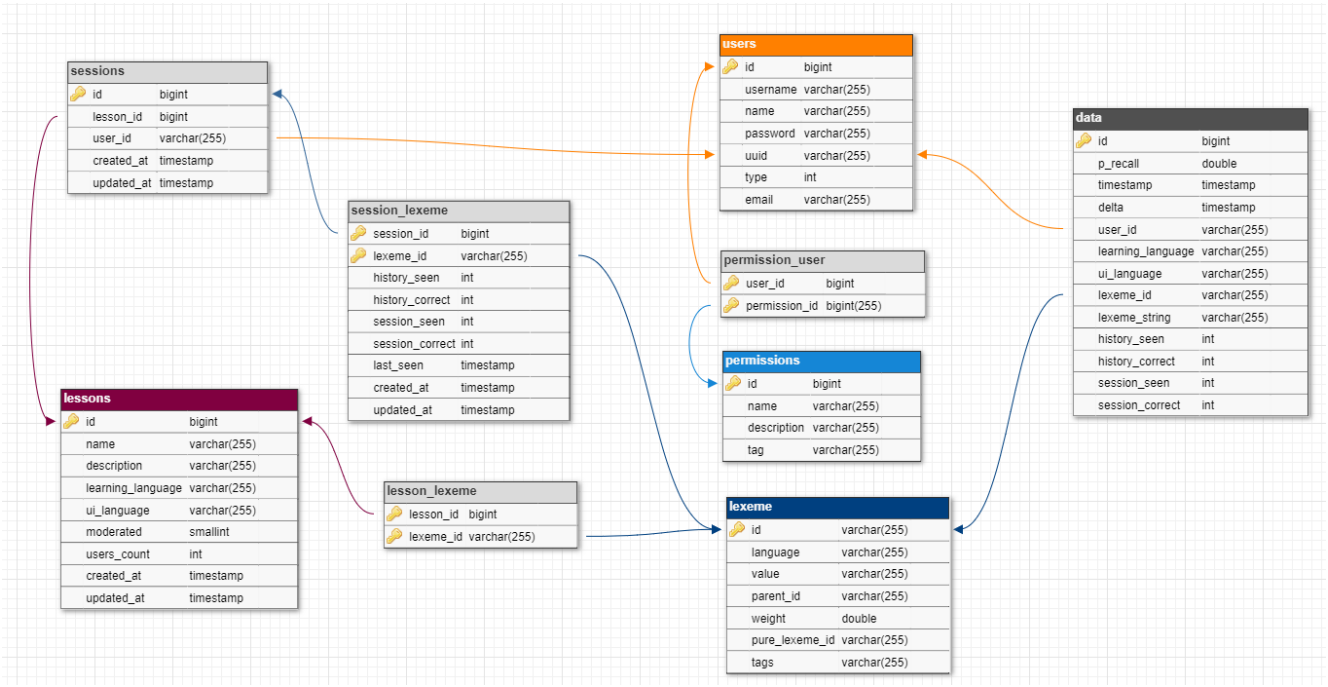

Рис. 5.2 – Схема бази даних програмного забезпечення

Основою бази даних є таблиця data, що включає до себе всі дані с тренувальної вхідної вибірки, описаної в розділі 3.3. Колонки цієї таблиці повністю дублюють колонки вибірки, тому повторно описувати їх не буду.

Таблиця lexeme представляє собою список всіх слів які будуть вивчати користувачі. Ці слова напряму взяті із тренувальних даних. Таблиця складається з колонки id – взятого з lexeme_id таблиці data, language – мова на якій написано слово, value – значення самого слова, parent_id – id з цієї ж таблиці але зведеної до англійської мови, weight – ваги взяті з прогону тестових даних через регресію періоду напіврозпаду (детальніше у розділі 3.5), pure_lexeme_id – id початкової форми слова, наприклад wrote – write, tags – теги лексеми взятої все ж з тієї ж таблиці data, колонки lexeme_string.

Таблиця lessons зберігає в собі всі створені користувачем уроки в системі, та складається з таких полів: id – унікальний ключ, name – ім’я уроку, description – опис, learning_language – мова яку будуть вивчати в цьому уроці, ui_language – мова яку знає користувач, moderated – чи пройшов урок модерацію та доступний у глобальному пошуку, users_count – загальна кількість користувачів, що проходить урок, created_at – дата створення уроку, updated_at – дата останнього редагування урока.

Таблиці lessons та lexeme з’єднані проміжною таблицею lesson_lexeme, яка допомагає прив’язати слова до уроку.

Таблиця users представляє собою всіх користувачів, що зареєструвалися в системі, та має такі поля: id – унікальний ключ, username – логін, один з варіантів авторизації та пошуку користувачів в системі, name – справжнє ім’я (можна залишити пустим), password – пароль, uuid – анонімізований id, type – тип користувача, email – електронна пошта, що вказується при реєстрації.

Таблиця permissions потрібна для створення дозволів, що прикріплюються до користувачів і дозволяють робити ті чи інші речі без модерації, зв’язується з таблицею users та допомогою проміжної permission_user. Має наступні колонки: name – ім’я дозволу, description – короткий опис, що робить цей дозвіл, tag – системний тег для перевірки дозволів.

Таблиця sessions зберігає всі навчальні сесії користувача та приймає активну участь у прогнозування забування слів користувачем та має наступні поля: lesson_id – зовнішній ключ для прив’язки до таблиці lessons, user_id – зовнішній

ключ для прив’язки до таблиці users, created_at – дата початку сесії, updated_at – дата останнього проходження сесії.

Таблиця session_lexeme представляє собою всі слова, що користувач побачив у сесії грає пряму роль у прогнозуванні моделі та складається з таких полів: session_id – зовнішній ключ для прив’язки до таблиці sessions, lexeme_id – зовнішній ключ для прив’язки до таблиці lexeme, history_seen – загальна кількість разів користувач побачив слово, history_correct – загальна кількість слів користувач згадав слово правильно, session_seen – кількість разів користувач побачив слово за сесію, session_correct – кількість разів користувач згадав слово правильно за сесію, last_seen – дата останнього перегляду слова в даній сесії, created_at – дата першого перегляду слова, updated_at – загальна дата останнього перегляду слова.

5.3 Проектування класів

Розглянемо сервер програмного забезпечення та переглянемо функції, які вони виконують. На рисунку 5.3 знаходиться частина діаграми класів.

Загальний клас Model описує всі таблиці бази даних, які унаслідуються від нього. Цей клас зв’язаний з іншим класом Abstract, який реалізує усі базові методи для роботи з базою даних, наприклад пошук по полю, рахування кількості записів, завантаження зв’язаних таблиць і так далі.

Клас User реалізує методи для роботи з користувачем, такі як вхід, реєстрація, відновлення паролю, підтвердження електронної пошти і тд.

Клас Lesson створений для маніпуляції з уроками та вирішує питання створення нових уроків, редагування уже існуючих та видалення з бази.

Клас Session відповідає за створення нової навчальної сесії користувача та редагування даних, що були додані до старих сесій.

Клас Data працює з тренувальними даними та зберігає їх у таблиці.

Клас Lexeme створений для додавання нових слів в систему (як правило це відбувається тільки на стороні серверу, користувач ніяк не може це зробити з

інтерфейсу програмного забезпечення), редагування коефіцієнтів вагів уже існуючих слів, а також відповідає за прогнозування забування слів користувачем.

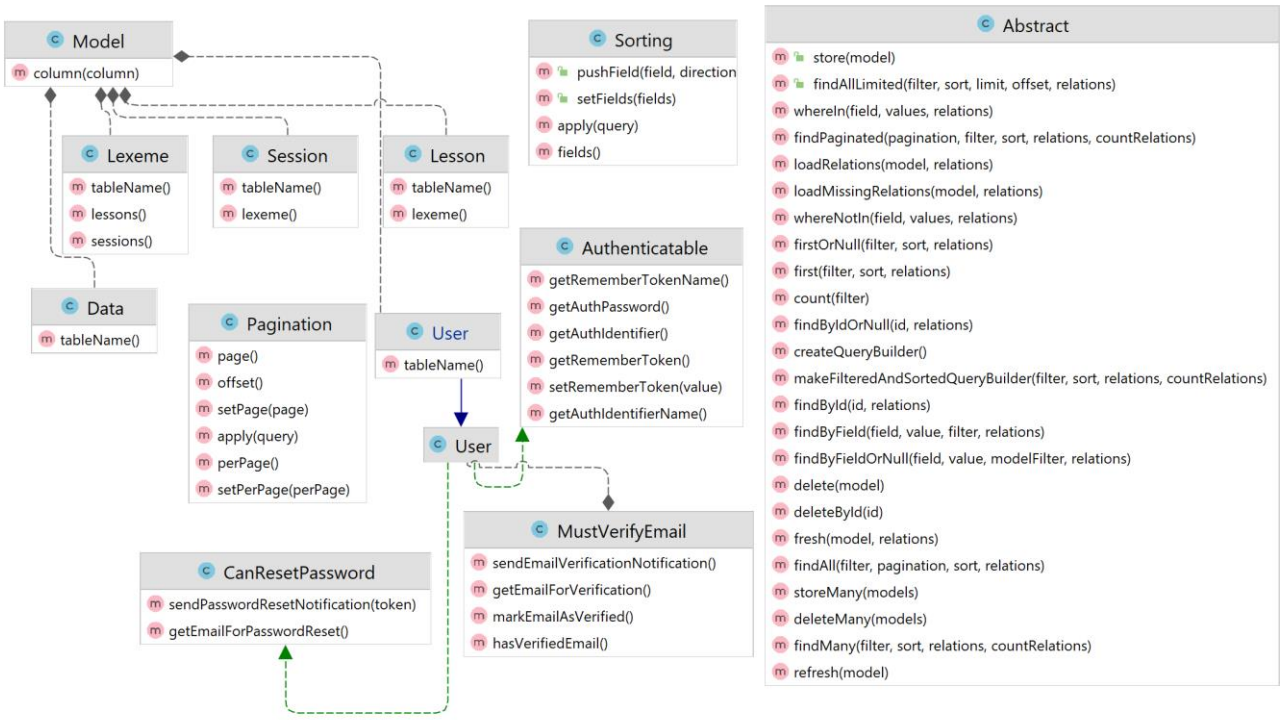

Рис. 5.3 – Частина діаграми класів

Також існують додаткові класи Pagination та Sorting, які допомагають у відображені списку всіх існуючих даних посторінково, а також у сортуванні по полям таблиць.

У додатку В ви можете ознайомитися з кодом програмної системи та знайти приклад деяких методів, описаних вище.

ВИСНОВКИ

В результаті кваліфікаційної роботи була опрацьована тема машинного навчання, розглянуто основні етапи роботи, а саме: збір даних, підготовка даних, вибір моделі, навчання, оцінка, оптимізація гіперпараметрів, передбачення а також створена навчальна модель з використанням алгоритму регресії періоду напіврозпаду.

Деякі ідеї були взяті з опису нечіткої нейронної мережі [21] та з системи аналізу медичних даних. [22]

На основі цієї моделі було створено програмне забезпечення, яке наглядно демонструє результати її роботи, а саме аналіз існуючих даних та прогнозування наступних даних користувача.

В ході досліджень було виявлено, що використання моделей машинного навчання для навчання учнів прискорює запам’ятовування матеріалу та може бути використано в інших сферах навчального процесу.

Що ж до алгоритму регресії періоду напіврозпаду, що розглядався як основний у моделі описаній у кваліфікаційній роботі, то він виявився кращий у деяких сферах. Середня абсолютна помилка майже у два раз краща ніж у алгоритму Лейтнера, на однакових даних і майже у три рази ніж у алгоритму Пімслер.

Також варто зауважити, що по критерію суми рангів Вілкоксона, який являє собою показником якості ранжирування регресія періоду напіврозпаду майже не відстає, хоча вона не була оптимізована для цього критерію, що свідчить про можливість подальшого покращення моделі, наприклад з використанням сторонніх бібліотек, які збільшують цей коефіцієнт.

ПЕРЕЛІК ПОСИЛАНЬ

1. Lock M. Angling for Insight in Today's Data Lake / Michael Lock. – Aberdeen: RR, 2017. – 8 с. – (RR). – (1; вип. 1).
2. Earnado [Електронний ресурс] – Режим доступу до ресурсу: https://earnado.com/tr/makine-ogrenmesi-nedir-nasil-yapilir-ornekler-ile-anlatim. (дата звернення: 10.01.2022)
3. GKTCS [Електронний ресурс] – Режим доступу до ресурсу: https://gktcs.com/content_management/blog_details/120. (дата звернення: 12.01.2022)
4. CIKSITI [Електронний ресурс] – Режим доступу до ресурсу: https://ciksiti.com/el/chapters/5693-top-50-frequently-asked-machine-learning-interview-questions. (дата звернення: 15.01.2022)
5. Єрохін А.Л., Латиш А.С. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Baku, 27.04.2022 - 28.04.2022. Том 2, Секція 5. С. 172.
6. Ebbinghaus H. Memory : a contribution to experimental psychology / Hermann Ebbinghaus. – New York: Columbia University, 2013. – 511 с.
7. Meeder B. A Trainable Spaced Repetition Model for Language Learning / B. Meeder, B. Settles., 2016. – 1848 с.
8. Aalen O. Survival and Event History Analysis: A Process Point of View (Statistics for Biology and Health) / O. Aalen, O. Borgan, H. Gjessing. – Oslo: Springer, 2008. – 558 с.
9. Leitner Sebastian. So Lernt Man Lernen / Leitner. – Herder, Freiburg, 2003. – 317 с.
10. Predicting the Optimal Spacing of Study: A Multiscale Context Model of Memory / Mozer, M. C., Pashler, H., Cepeda, N. та ін.]., 2021. – 700 с.
11. Ebbinghaus H. Memory; A Contribution to Experimental Psychology / Hermann Ebbinghaus., 2011. – 134 с.

12. Melton A. W. The situation with respect to the spacing of repetitions and memory / Arthur W. Melton., 1970. – 10 с. – (Journal of Verbal Learning and Verbal Behavior).
13. Hilt D. E. Ridge, a computer program for calculating ridge regression estimates / D. E. Hilt, D. W. Seegrist., 2013. – 201 с.
14. Pranab K. Sen. Large Sample Methods in Statistics / Pranab K. Sen, Julio M. Singer. – Boca Raton, 2017. – 435 с. – (3rd Edition).
15. Fawcett T. An introduction to ROC analysis / Tom Fawcett., 2005. – 874 с.
16. Prettyman S. Learn PHP 8: Using MySQL, JavaScript, CSS3, and HTML5 / Steve Prettyman., 2020. – 452 с. – (2nd Edition).
17. Shaw Z. Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code (Zed Shaw's Hard Way Series) / Zed Shaw., 2017. – 320 с.
18. Heitor R. R. Vue.js 3 Cookbook: Discover actionable solutions for building modern web apps with the latest Vue features and TypeScript / Ramon Ribeiro Heitor. – 562, 2020.
19. Herman D. Effective JavaScript: 68 Specific Ways to Harness the Power of JavaScript / David Herman., 2021. – 276 с. – (2nd Edition).
20. Vanier E. Advanced MySQL 8: Discover the full potential of MySQL and ensure high performance of your database / E. Vanier, B. Shah, T. Malepati., 2019. – 288 с.
21. Yerokhin, O. Zolotukhin. Fuzzy probabilistic neural network in document classification tasks // Interbranch collection of scientific papers «Information Extraction and Processing». National Academy of Sciences of Ukraine. 46 (122) – 2018. – P.69–71. https://doi.org/10.15407/vidbir2018.46.068
22. Yerokhin, A., Turuta, O., Babii, A., Nechyporenko, A. Intelligent information system of heterogeneous medical data analysis // Proceedings of the 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2017. Lviv, Ukraine, 2017. – p.p. 332-335.