ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ОЛЕСЯ ГОНЧАРА

Факультет прикладної математики
Кафедра обчислювальної математики та математичної кібернетики

Кваліфікаційна робота
перший (бакалаврський) рівень вищої освіти
спеціальність 124 Системний аналіз

РОЗРОБКА АЛГОРИТМУ НАВЧАННЯ ТОЧКОВОЇ МОДЕЛІ РАНЖУВАННЯ СПИСКУ, ЗАЛЕЖНОЇ ВІД ЙОГО КОНТЕКСТУ

Виконавець
студент групи ПС–18–1
Каманцев Артем Сергійович _____

Керівник
професор кафедри обчислювальної
математики та математичної кібернетики
д-р. фіз.-мат. наук, доц.
Притоманова О.М. _____

Завідувач кафедри обчислювальної
математики та математичної кібернетики
канд. фіз.-мат. наук, доц.
В.А. Турчина _____

Дніпро – 2022

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 124 Системний аналіз

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної математики та математичної кібернетики

_____Валентина ТУРЧИНА

«___» _____ 2022 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ

Каманцев Артем Сергійович

1. Тема роботи Розробка алгоритму навчання точкової моделі ранжування списку, залежної від його контексту.
керівник роботи Притоманова Ольга Михайлівна д-р. фіз.-мат. наук, професор кафедри обчислювальної математики та математичної кібернетики, затверджені наказом по Університету від 21.03.2022 №303-с.
2. Строк подання роботи 7.06.2022
3. Вхідні дані до роботи
Задача ранжування списку, методи класифікації, набори даних: CUAD V1, відгуки користувачів про додатки у Google Play.
4. Зміст кваліфікаційної роботи
Визначення впливу введення конкуруючих елементів до моделей класифікації, що використовуються як локальні функції ранжування, на точність розв’язку задачі ранжування точковим підходом (poitwise).
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

не передбачений

6. Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
	Не передбачено		

7. Дата видачі завдання 17.01.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Формулювання теми кваліфікаційної роботи, складання графіку її виконання.	19.02.2022	Виконано
2	Здійснення пошуку інформації в галузі системного аналізу за темою роботи, використовуючи наукові журнали, on-line ресурси.	1.03.2022	Виконано
3	Вивчення алгоритмів та моделей ранжування списку.	8.03.2022	Виконано
4	Попередня обробка даних.	1.04.2022	Виконано
5	Проведення експериментів.	15.05.2022	Виконано
6	Аналіз результатів.	22.05.2022	Виконано
7	Оформлення кваліфікаційної роботи та супроводжуючої документації.	5.06.2022	Виконано
8	Надання в електронному вигляді примірника кваліфікаційної роботи до випускової кафедри.	7.06.2022	Виконано
9	Надання паперового примірника кваліфікаційноїроботи з власноручним підписом до випускової кафедри.	15.06.2022	Виконано

Студент _____ Артем КАМАНЦЕВ

Керівник роботи _____ Ольга ПРИТОМАНОВА

РЕФЕРАТ

Кваліфікаційна робота: 43с., 6 рис., 5 табл., 19 джерел, 1 додатків.

Об’єкт дослідження: точкова (pointwise) модель ранжування.

Мета роботи: розробити алгоритм навчання точкової моделі ранжування списку, залежної від його контексту.

Методи дослідження: методи математичної статистики, комп’ютерний експеримент.

Одержані результати та їх новизна: запропоновано новий метод формування даних для точкової (pointwise) моделі ранжування, з’ясовано вплив такого підходу на якість розв’язку задачі ранжування.

Ключові слова: РАНЖУВАННЯ, ЗАДАЧА РАНЖУВАННЯ, ТОЧКОВА МОДЕЛЬ РАНЖУВАННЯ, WINNER-TAKES-ALL, NAMED ENTITY RECOGNITION, НАВЧАННЯ З УЧИТЕЛЕМ.

ANNOTATION

The graduation research of the 4-year student A. Kamantsev (Oles Honchar Dnipro National University, Faculty of Applied Mathematics, Department of Calculating Mathematics and Mathematical Cybernetics) deals with learning to rank problem.

The study is focused on the learning to rank problem with fixed query and Winner-Takes-All metrics as a special case of Named Entity Recognition task. The pointwise approach with classification model as a local ranking function is used to solve the ranking problem. The research is conducted to figure out the impact of an alternative approach to the creation of input data on the classification model. It is proposed to input into the model not only single element of the ranked list, but also all other elements of this list as additional features at each step. The following classification models were used as a local ranking function: Naive Bayes, Logistic Regression, K Near Neighbors, Support Vector Machine, Decision Tree, XGBoost.

The proposed data generation approach improves ranking results only when Logistic Regression classifier is used as a local ranking function (or XGBoost, in case there is not much noise in the data). When other models are used, the ranking results does not improve or deteriorate.

The results of the study can be used to understand the boundaries of the feasibility of the proposed data generation approach or as a starting point for further research on the feasibility of the proposed data generation approach.

Pages 43, bibliography 19, pictures 6, tables 5, supplement 1.

ЗМІСТ

ВСТУП

Процес добування інформації – процес, що отримує структуровану інформацію з неструктурованих даних у формі сутностей та їх зв’язків. Добута інформація використовується для аналізу даних. Не структуровані дані можуть бути у вигляді тексту, картинки, аудіо чи відео і для кожного виду даних були розроблені численні техніки добування структурованої інформації з них [1].

У роботі приділено увагу обробці неструктурованого тексту, тому розглянемо детальніше задачі і методи добування структурованої інформації саме з цього виду даних.

Natural Language Processing (NLP) – група методів, що дозволяє інтерпретувати усну чи письмову мову людей. При застосуванні NLP для обробки мови людей виникає низка задач високого рівня, таких як: добування інформації, корекція інформації, машинний переклад, система відповідей на питання і розуміння натуральної мови. У рамках задачі добування інформації з текстових даних виділяють такі підзадачі [1]:

1) Named Entity Recognition (NER)
2) Relation Extraction (RE)
3) Event Extraction (EE) and salient facts extraction

Розглянемо детальніше підзадачу NER.

NER – задача ідентифікації домен-незалежних сутностей, таких як локація, особа чи організація та домен-залежних, таких як хвороби, ліки, хімікати, протеїни та інше. Традиційні NER системи можна поділити на: засновані на

правилах (Rule-Based Methods, RBM), засновані на навчанні (Learning-Based Methods, LBM) та гібридні підходи [1].

Один з гібридних підходів можна описати так [2]:

1) Знайти у тексті, за допомогою шаблонів, послідовності слів або символів, які є ймовірними кандидатами бути сутністю, що необхідно знайти у тексті. Такими кандидатами можуть виступати дати, числа, валюти, речення чи слова. На цьому кроці застосовуються RBM.
2) За допомогою деякої моделі, попередньо натренованої на розмічених даних, визначити, які з кандидатів є шуканою сутністю. Оскільки модель попередньо тренується на розмічених даних, то цей крок можна віднести до LBM.

Іноді можна спростити задачу, зробивши припущення, що в тексті шукана сутність має бути одна і лише одна.

У цьому випадку задача NER зводиться до задачі ранжування, що не залежить від запитів, із метрикою Winner-Takes-All (WTA). Задача ранжування полягає у тому, щоб за деяким критерієм впорядкувати скінчену множину елементів. Критерій може не бути задано у явному вигляді, тому функцію ранжування вивчають за тренувальними даними за допомогою деякої моделі [3]. Метрика WTA полягає у тому, що якість моделі оцінюється лише за вірністю ранжування першого елементу кожного списку (це частинний випадок метрики Precision at k при k=1 [4]).

Існує три основних підходи до розв’язку задачі ранжування [4]:

1) Pointwise - кожний документ списку ранжується окремо і функція помилки обчислюється окремо для кожного документу.

2) Pairwise – до моделі по черзі вводяться пари документів, і їй необхідно визначити, чи правильно впорядкована введена пара. На основі відповідей моделі відбувається впорядкування списку.
3) Listwise – враховує усі документи, що необхідно впорядкувати. Основною проблемою такого підходу є змінна довжина списку кандидатів у документі, тому в рамках даного підходу існують досить різні алгоритми: деякі обчислюють функцію помилки за усім списком документів, виконуючи оцінку кожного документу окремо; деякі роблять семплінг списку документів, деякі використовують складну архітектуру мережі, яка дозволяє вводити змінну кількість документів.

У роботі пропонується поєднати ідею про важливість впливу усіх елементів списку документів на значення функції ранжування, обчисленої для кожного документу, listwise підходу з pointwise підходом. Для цього пропонується для кожного документу вводити у модель, що обчислює функцію ранжування, не тільки сам цей документ (згідно pointwise підходом) а й усіх його конкурентів.

Об’єктом дослідження є точкова (pointwise) модель ранжування.

Предметом дослідження є вплив введення на кожному кроці у модель класифікації не лише одного елементу списку, що ранжується, а і його конкурентів з цього списку, на якість результатів ранжування за pointwise підходом.

Мета роботи полягає у тому, щоб з’ясувати, чи покращиться якість ранжування, якщо на кожному кроці вводити у модель не лише один елемент списку, що ранжується, а і його конкурентів з цього списку.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Задача ранжування

Задача ранжування може виникати в найрізноманітніших випадках у сферах Information Retrieval (IR), Natural Language Processing (NLP), і Data Mining (DM). Типові застосування: пошук документів, експертний пошук, пошук визначень, спільна фільтрація, відповіді на запитання, отримання ключових фраз, узагальнення документа та машинний переклад. Без втрати загальності, візьмемо пошук документів як приклад при розгляді теоретичної частини задачі ранжування, а потім проведемо аналогії з прикладними задачами, що розв'язуються у роботі.

Пошук документів виконується наступним чином (рис. 1.1). Система містить колекцію документів. За заданим запитом, система отримує документи, що містять слова запиту, з колекції, ранжує документи, і повертає документи сортовані за спаданням рейтингу. Задача ранжування розв'язується за допомогою моделі ранжирування $f(q, d)$ для сортування документів, де q позначає запит, а d позначає документ.

Традиційно, модель ранжування $f(q, d)$ створюється без навчання. Новий тренд у пошуку документів полягає у використанні методів машинного навчання для автоматичного конструювання функції ранжування $f(q, d)$ [3].

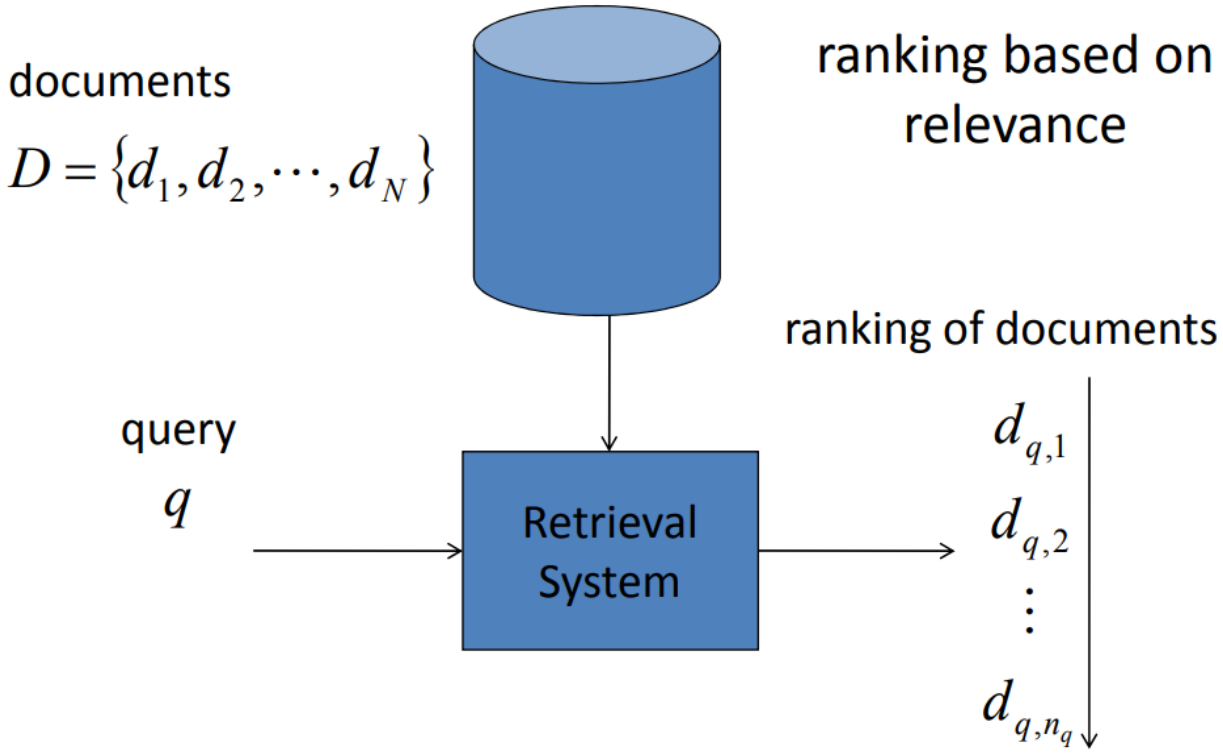


Рисунок 1.1 – Пошук документів

1.2 Тренування та тестування моделі ранжування

Задача ранжування це задача навчання з вчителем і тому має фази тренування та тестування (рис. 1.2).

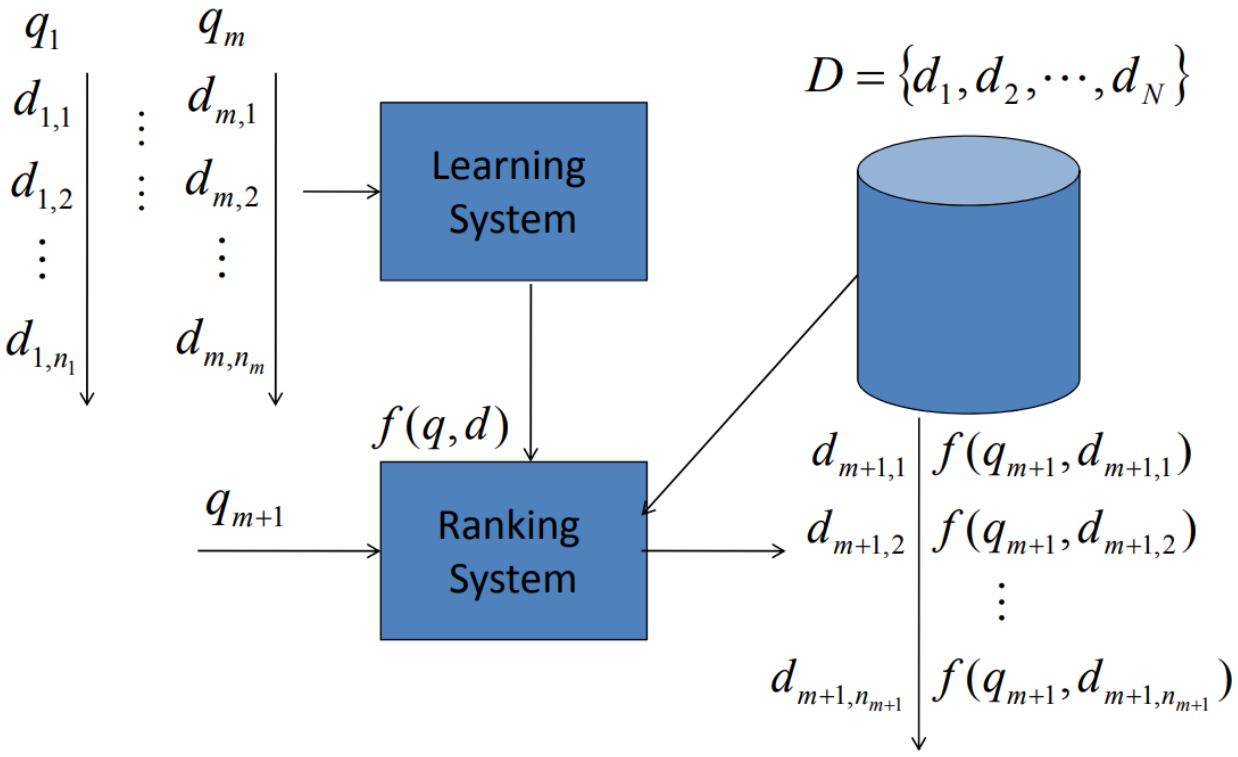


Рисунок 1.2 – Задача ранжування для пошуку документів

Дані для тренування складаються з запитів та документів. Кожний запит асоційований з певними документами. Релевантність документів відносно запиту задано у вигляді лейблів, які означають деякі оцінки (рівні). Чим вищу оцінку має документ, тим більш релевантним документ є.

Нехай Q це множина запитів, D – множина документів, $Y = \{1,2, \dots l\}$ – множина лейблів, де лейбли представляють оцінки. Існує повне відношення порядку на множині оцінок $l \succ l-1 \succ \cdots \succ 1$, де $\succ$ позначає відношення порядку. Далі покладемо, що $\{q_1, q_2, \dots, q_m\}$ це множина запитів для тренування і q_i це $i-$й запит. $D_i = \{d_{i,1}, d_{i,2}, \dots, d_{i,n_i}\}$ це набір документів, асоційованих із запитом q_i і $y_i = \{y_{i,1}, y_{i,2}, \dots, y_{i,n_i}\}$ це набір лейблів асоційованих із запитом q_i, де n_i позначає розміри D_i і y_i; $d_{i,j}$ позначає j-й документ у D_i; і $y_{i,j} \in Y$ позначає j-тий лейбл оцінки у y_i, що представляє ступінь релевантності $d_{i,j}$ відносно q_i. Оригінальний тренувальний набір позначається як $S = \{(q_i, D_i), y_i\}_{i=1}^{m}$.

Вектор ознак $x_{i,j} = \phi(q_i, d_{i,j})$ створюється з кожної пари запит-документ $(q_i, d_{i,j}), i = 1, 2, \dots, m; j = 1,2, \dots, n_i,$ де ϕ позначає функцію ознак. Варто зазначити, що ознаки визначені як функції пар запит-документ. Позначивши $x_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,n_i}\}$, ми представляємо тренувальні дані як $S` = \{(x_i, y_i)\}_{i=1}^{m}$. Тут $x \in X$, і $X \subseteq \mathbb{R}^d$

Нашою ціллю є натренувати (локальну) модель ранжування $f(q, d) = f(x)$ яка може приписувати оцінки заданим парам запит-документ (q і d) або, що еквівалентно, заданим векторам x. Більш загально, ми можемо розглядати тренування глобальної моделі ранжування $F(q, D) = F(x)$. Локальна модель ранжування виводить скалярну оцінку, у той час як глобальна модель ранжування виводить вектор оцінок.

Нехай документи у D_i будуть ідентифікуватись цілими числами $\{1, 2, \dots, n_i\}$. Визначимо перестановку (ранжований список) π_i на D_i як бієкцію з $\{1, 2, \dots, n_i\}$ у себе. Будемо використовувати Π_i для позначення множини усіх можливих перестановок на D_i, а $\pi_i(j)$ для позначення рангу (позиції) j-го

документу (або $d_{i,j}$) у перестановці π_i. Ранжування є нічим іншим як вибором перестановки $\pi_i \in \Pi_i$ для заданого запиту q_i і асоційованих документів D_i використовуючи оцінки надані моделлю ранжування $f(q_i, d_i)$.

Дані для тестування складаються з нового запиту q_{m+1} і асоційованих документів D_{m+1} (дані для тестування можуть складатись і з більшої кількості запитів, проте для простоти не будемо це враховувати). $T = \{(q_{m+1}, D_{m+1})\}$. Ми створюємо вектор ознак x_{m+1}, використовуючи натреновану модель ранжування, призначаємо оцінки документам D_{m+1}, сортуємо їх за оцінками, і надаємо ранжований список документів як вивід π_{m+1}.

Дані для тренування та тестування схожі, проте відмінні від даних, що використовуються у таких традиційних задачах навчання з учителем як класифікація і регресія. Запит та асоційовані з ним документі формують групу. Групи є незалежними і мають однаковий розподіл, у той час коли елементи всередині групи не є такими. Локальна модель ранжування це функція запиту і документу, або, що еквівалентно, це функція вектору ознак отриманого із запиту і документу [3].

1.3 Постановка формалізованої задачі ранжування

Формалізуємо задачу ранжування як задачу навчання з вчителем. Нехай $\aleph$ це вхідний простір (простір ознак), що складається зі списків векторів ознак, а γ це вихідний простір, що складається зі списків оцінок. У подальшому вважатимемо, що x це елемент простору $\aleph$, а y елемент простору γ. Нехай $P(X, Y)$ це невідомий сумісний розподіл ймовірностей де випадкова змінна X приймає x

як своє значення, а випадкова змінна Y приймає y як своє значення.

Нехай $F(\cdot)$ це функція відображення зі списку векторів ознак x у список оцінок y. Ціллю навчання є автоматичне вивчення функції $\hat{F}(x)$ за заданими тренувальними даними $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$. Кожний елемент тренувальної вибірки складається із вектору ознак x_i і відповідних оцінок y_i $(i = 1, \dots, m)$. Тут m позначає кількість елементів тренувальної вибірки.

$F(x)$ і y можуть бути записані як $F(x) = \big(f(x_1), f(x_2), \dots, f(x_n)\big)$ і $y = (y_1, y_2, \dots, y_n)$. Вектори ознак представляють об'єкти, що ранжуються. Тут $f(x_i), i = \overline{1,n}$ позначає локальну функцію ранжування і n позначає кількість векторів ознак и оцінок.

Задачею роботи є створення локальної функцій ранжування, що буде залежати від усього вектору x, а не лише від одної з його компонент.

Функція втрат $L(\cdot, \cdot)$ використовується для оцінки результатів прогнозів $F(\cdot)$. Спочатку, вектори ознак x ранжуються у відповідності до $F(x)$, потім перші n результатів ранжування оцінюються з використанням відповідних оцінок y. Якщо вектори ознак з високими оцінками ранжовано високо, то втрати будуть малими. Інакше, втрати будуть великими. Функція втрат конкретно представлена як $L(F(X), y)$. Зазначимо, що функція втрат для ранжування дещо відрізняється від функції втрат у інших статистичних задачах навчання, у тому сенсі, що вона використовує сортування.

Визначимо функцію ризику $R(\cdot)$ як очікувану функцію втрат відносно сумісного розподілу $P(X, Y)$,

$$R(F) = \int_{\aleph \times \gamma} L(F(x), y) dP(x, y)$$

За наявними тренувальними даними, ми обчислюємо емпіричну функцію ризику:

$$\hat{R}(F) = \frac{1}{m} \sum_{i=1}^{m} L(F(x_i), y_i)$$

Тоді задача навчання стає задачею мінімізації емпіричної функції ризику, як у інших задачах навчання. Мінімізувати емпіричну функцію ризику може бути складно через природу функції втрат (вона не є неперервною і використовує сортування). Ми можемо розглянути використання сурогатної функції втрат $L`(F(x), y)$.

Відповідна емпірична функція втрат:

$$\hat{R}'(F) = \frac{1}{m} \sum_{i=1}^{m} L'(F(x_i), y_i)$$

Ми також можемо ввести регуляризацію для проведення мінімізації регуляризованого емпіричного ризику. У таких випадках задача навчання перетворюється на задачу мінімізації (регуляризованої) емпіричної функції ризику заснованої на сурогатній функції втрат.

Зазначимо, що ми використали формулювання машинного навчання тут. Під час добування даних, вектори ознак отримуються із запиту і асоційованих із ним документів. Оцінки y представляють ступені релевантності документів відносно запиту. Ми використовуємо глобальну функцію ранжування $F(\cdot)$. На практиці, це може бути локальна функція ранжування $f(\cdot)$. Кількість векторів

ознак у x може бути дуже великою, навіть нескінченою. Проте оцінка (функція втрат) враховує лише n результатів.

У задачах добування даних, справжня функція втрат може бути визначена на основі Normalized Discounted Cumulative Gain або Mean Average Precision. Для сурогатної функції втрат існують також різні способи визначення, що веде до різних підходах до розв’язку задачі ранжування. Наприклад, можна визначити pointwise, pairwise, listwise функції втрат [3].

2 РОЗРОБКА АЛГОРИТМУ НАВЧАННЯ ТОЧКОВОЇ МОДЕЛІ РАНЖУВАННЯ, ЗАЛЕЖНОЇ ВІД КОНТЕКСТУ СПИСКУ

2.1 Ранжування за pointwise підходом

Pointwise підхід перетворює задачу ранжування на задачу класифікації, регресії чи порядкової класифікації і застосовуються існуючі методи класифікації, регресії чи порядкової класифікації. Тому структура групи ранжування ігнорується у цьому підході.

У роботі приділено увагу підходу, що перетворює задачу ранжування на задачу класифікації. Кожному з кандидатів приписується лейбл приналежності до одного з 2х класів:

0 – якщо кандидат не є шуканою сутністю

1 – якщо кандидат є шуканою сутністю

Тоді у якості локальної функції ранжування $f(x)$ можна використовувати модель класифікації, вихідним значенням якої є вірогідність того, що кандидат є шуканою сутністю (тобто вірогідність приналежності кандидата до класу «1»).

2.2 Алгоритм навчання точкової моделі ранжування, залежної від усього списку

Процес розв’язку задач, згідно із pointwise підходом, можна зобразити наступним чином (див. рис. 2.1):

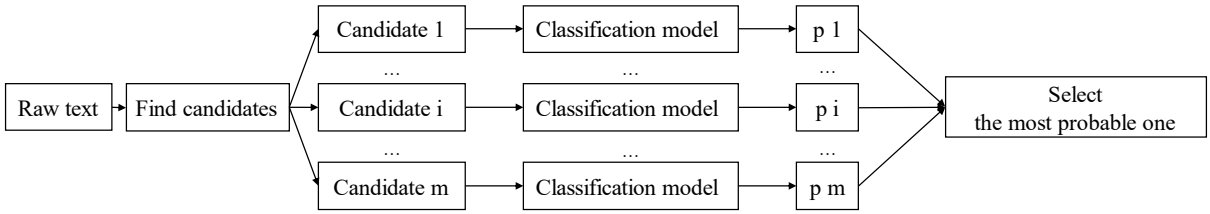


Рисунок 2.1 – Процес розв’язку задачі за pointwise підходом

Запропонований варіант полягає у тому, щоб разом із кожним кандидатом вводити у модель класифікації і його конкурентів. Тоді процес матиме вигляд (див. рис. 2.2):

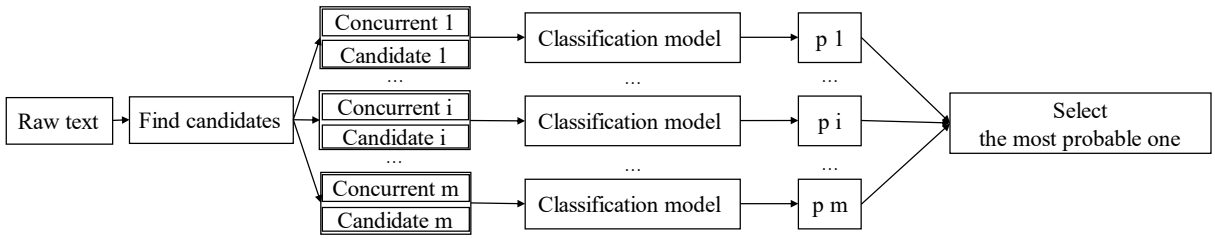


Рисунок 2.2 – Запропонований процес розв’язку задачі

де:

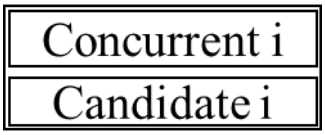
Raw text – документ, представлений у вигляді рядку тексту

Find Candidates – етап пошуку кандидатів на роль шуканої сутності у документі за допомогою алгоритмів.

$Candidate\ 1, \dots, Candidate\ m$ – вектори ознак знайдених кандидатів

$Concurrent\ i$ – сума всіх векторів ознак знайдених кандидатів, окрім того вектору, що відповідає i-му кандидату, тобто:

$$Concurrent\ i = \sum_{\substack{j=1 \\ j \neq i}}^{m} Candidate\ j\,, \qquad i = \overline{1, m}$$



- конкатенація векторів $Concurrent\ i$ і $Candidate\ i$, $i = \overline{1, m}$.

Classification model – модель класифікації (для кожного кандидата використовується одна і та ж сама модель)

$p\ i$ – вірогідність того, що $Candidate\ i$ є шуканою сутністю, оцінена моделлю класифікації.

Запропонований підхід до формування вхідного вектору даних для моделі класифікації у подальшому будемо називати *конкурентними*.

2.1 Метрики

Оскільки якість ранжування необхідно визначати лише за першим елементом ранжованого списку, то необхідною метрикою для контроля якості ранжування буде метрика WTA, що обчислюється як:

$$wta = \begin{cases} 1, & \text{якщо кандидат на першій позиції є шкуканою сутністю} \\ 0, & \text{якщо кандидат на першій позиції не є шуканою сутністю} \end{cases}$$

Точність моделі за усім набором даних обчислюється за формулою:

$$precision = \frac{\sum_{i=1}^{n} wta_i}{n}$$

де

wta_i – метрика WTA, обчислена для i-го документу з набору даних

n – кількість документів у наборі даних

Така метрика у деяких випадках може виявитись не сильно чутливим до змін у моделі, особливо при невеликій кількості даних. Наприклад, у моделі відбулись деякі зміни, проте порядок для жодного елементу при жодному ранжуванні не змінився, але надала елементам інші ймовірності приналежності до класів. Чутливою до таких змін буде метрика $ROC - AUC$ [6]. Зміст її значення можна описати так:

$$ROC - AUC = p\left(p^{+}_{positive} > p^{+}_{negative}\right)$$

де

p – ймовірність

$p^{+}_{positive}$ - ймовірність приналежності до позитивного класу у задачі бінарної класифікації, яку надає модель елементу, що насправді належить до *позитивного* класу

$p^{+}_{negative}$ - ймовірність приналежності до позитивного класу у задачі бінарної класифікації, яку надає модель елементу, що насправді належить до *негативного* класу

Значення метрики ROC-AUC, обчислені для кожного документа, агрегуються за тактикою $micro - average$ [7].

2.2 Задача ранжування у термінах предметної галузі

Розглянемо загальну постановку прикладних задач, що розглядаються у роботі.

Наявна множина документів, кожний документ представлено у вигляді рядку тексту. Відомо, що у кожному документі присутня що найменш одна сутність деякого класу. Для кожної прикладної задачі, що розглядається у роботі, клас сутностей є фіксованим. У якості таких класів можуть виступати:

- Слово, що має найпозитивнішим у тексті (може бути лише одне у документі)
- Закон, що регулює дію контракту (може бути декілька у документі)
- Сума контракту (може бути декілька у документі)
- Дата підписання контракту (може бути декілька у документі)

та інші.

Для кожного документу може бути сформовано список з кандидатів на роль шуканої сутності. Кандидат представляється у вигляді під рядка рядку, яким представлено документ. Такий список може бути сформований за допомогою алгоритмів, що засновані на регулярних виразах (наприклад, мовою Python [5]).

Для кожного документу серед усіх кандидатів необхідно визначити одного такого, який дійсно являє собою шукану сутність. Іншими словами, необхідно список з кандидатів впорядкувати таким чином, щоб його першим елементом був той кандидат, який являє собою шукану сутність.

Для усіх наявних документів відомо, які з кандидатів кожного документу є шуканою сутністю. Використовуючи ці дані, необхідно побудувати модель, що зможе розв’язати задачу для нових документів, для яких не відомо, які з кандидатів є шуканою сутністю.

Отже, розглядувані задачі є задачами ранжування. У цих задачах документи виступають у якості запитів, а кандидати – у якості документів, з постановки задачі ранжування для пошуку документів.

Зауважимо, що у постановці задачі ранжування для задачі пошуку документів різні множини d_i, що відповідають різним запитам q_i, можуть перетинатись, проте множини кандидатів для різних документів перетинатись не можуть.

Задача роботи полягає у тому, щоб визначити, який ефект на pointwise підхід ранжування має запропонована модель формування вхідного вектору даних для моделі класифікації.

3 ХІД ЕКСПЕРИМЕНТУ

3.1 Збір і обробка даних

Для експерименту було зібрано дані з двох джерел:

1) Відкритий набір даних CUAD V1 [8]
2) Google Play Reviews – відгуки та оцінки користувачів сервісу Google Play [9], які вони ставлять додаткам.

Розглянемо детальніше кожне джерело.

3.1.1 CUAD V1

3.1.1.1 Збір даних і їх структура

Це джерело є відкритим і містить 510 юридичних договорів. Воно складається з:

- Документів у форматі «.pdf»
- Текстових файлів, що містять текст, отриманий з pdf-документів
- Файлу «master_clauses.csv», що містить мітки для кожного документу

У роботі використано мітки:

- «Governing Law» - одне або декілька речень, що містять інформацію про те, за якими законами контракт має інтерпретуватись.
- «Governing Law-Answer» - одна або декілька назв регіонів, у яких контракт має інтерпретуватись.

Приклад міток з файлу «master_clauses.csv» (див. рис. 3.1):

Filename	Governing Law	Governing Law-Answer
CybergyHoldingsInc_201	['This Agreement is accepted by Comp	Nevada
EuromediaHoldingsCorp_	['This Agreement is subject to all laws	Ontario, Canada
FulucaiProductionsLtd_2	['All questions with respect to the con	Florida
GopageCorp_20140221_	['This Agreement shall be governed by	Nevada
IdeanomicsInc_2016033(	['This Agreement shall be governed by	New York
DeltathreeInc_19991102	['This Agreement shall be governed by	New York
EdietsComInc_20001030	['This Agreement shall be interpreted	California
IntegrityMediaInc_2001(	['This Agreement has been entered int	Tennessee
MusclepharmCorp_2017	['This Agreement has been executed a	California
TomOnlineInc_20060501	['This Agreement will be governed by	England, United Kingdon

Рисунок 3.1 – Приклад міток

3.1.1.2 Постановка задачі

Задача полягає у знаходженні в тексті кожного документа будь-якого з речень, що містять інформацію про те, за якими законами контракт має інтерпретуватись.

Кандидатами будуть виступати усі речення документу, що містять хоча б одну з назв регіонів, які перелічені у стовпчику «Governing Law-Answer», чи назву будь-якого штату США.

3.1.1.3 Очистка даних

Очистка даних складається з таких кроків:

1) Прибрати з розгляду документи, що хоча б у одному зі стовпчиків «Governing Law» чи «Governing Law-Answer» не містять значень, або містять порожній масив «[]» чи «[* * *]». В результаті залишилось 429 документів.
2) У тексті кожного документу та елементів стовпців «Governing Law» і «Governing Law-Answer» виконати заміни, за такими правилами (за допомогою регулярних виразів):

$$\# \rightarrow \text{ " ";}$$

$$\backslash n \rightarrow \text{ " ";}$$

$$\backslash xa0 \rightarrow \text{ " ";}$$

$$"\backslash x0c" \rightarrow \text{ " ";}$$

$$" < omitted > " \rightarrow " < omitted > \text{ ";}$$

$$" \, . " \rightarrow ".";$$

$$"\_" \rightarrow \text{ " ";}$$

$$" - - + " \rightarrow \text{ " ";}$$

$$"\backslash * +" \rightarrow " * ";$$

$$" + " \rightarrow \text{ " ";}$$

3.1.1.4 Формування кандидатів та міток для задачі ранжування

Після очистки даних, у кожному документі було знайдено список кандидатів, кожний з яких представляє собою речення, яке може описувати закон, за яким контракт має інтерпретуватись. Для цього виконуються такі кроки:

1) Знайти у тексті кожного документа початок та кінець усіх речень, що перелічені у стовпчику «Governing Law» для цього документу. Для тих документів, для яких це не вдалось зробити – необхідно виправити помилки у текстових файлах, які виникли внаслідок автоматичного добування тексту з pdf-документів, і повторити спробу.
2) Розділити кожний елемент стовпця «Governing Law-Answer» за допомогою регулярного виразу: "; ?|,?" і сформувати з отриманих елементів і списку штатів США множину S.
3) Для кожного тексту документу виконати:
    а) Розбити текст на речення за допомогою класу nltk.tokenize.punkt бібліотеки nltk [10].
    б) Відібрати ті речення, що містять хоча б один елемент множини S.
    в) Якщо після відбору залишилось менше двох речень – прибрати документ з подальшого розгляду (залишилось 419 документів), оскільки ранжування списку з одного елементу не є задачею, яка потребує розв’язку.
    г) Для кожного відібраного речення визначити, чи перетинаються його межі із межами речень, що містяться у стовпчику «Governing Law» документу. Якщо так, то присвоїти цьому реченню мітку «1» інакше «0».

д) Текст кожного відібраного речення зконкатенувати з попереднім реченням (якщо наявне).

е) Виправити помилки, що пов’язані з тим, що розбиття на речення за допомогою бібліотеки nltk може не співпадати із тими реченнями, що містяться у стовпчику «Governing Law».

3.1.1.5 Векторизація

Для векторизації даних для кожного кандидата кожного документу було виконано наступні дії:

1) Прибрати знаки наголосу з тексту.
2) Привести текст до нижнього регістру.
3) Розбити текст на слова, за допомогою регулярного виразу:

   '(?u)(\b[a-z]{2,}\b)'

   Тобто словом вважається будь-яка послідовність символів від «a» до «z» довжиною 2 та більше.
4) Лематизувати кожне слово за допомогою класу nltk.WordNetLemmatizer бібліотеки nltk.
5) Лематизувати усі слова зі списку sklearn.feature_extraction.text. ENGLISH_STOP_WORDS бібліотеки scikit-learn [11] і прибрати усі такі слова зі слів документу.
6) Зі слів сформувати біграми.
7) Прибрати слова і біграми, що зустрічаються менше ніж 10 разів серед усіх документів.

8) Векторизувати за технікою “Bag of words” (бінаризований).
9) Якщо отримано порожній вектор – видалити поточного кандидата та мітку, що йому відповідає з набору даних.

3.1.2 Google Play Reviews

3.1.2.1 Збір даних і їх структура

За допомогою бібліотеки-скраперу [12] було отримано 25 000 000 коментарів користувачів сервісу Google Play до додатків з різних категорій (але не більше 2000 коментарів до одного додатку). З них було виділено 25 000 коментарів так, щоб розподіл оцінок користувачів збігався з розподілом оцінок вихідного набору даних.

Отримані дані зберігаються у json форматі і мають вигляд (див. рис. 3.2):

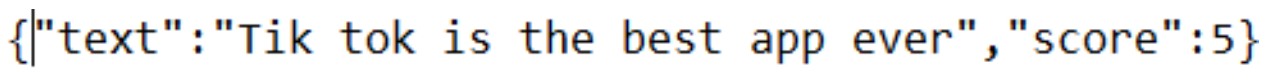


Рисунок 3.2 – Приклад відгуку користувача про додаток, отриманий з сервісу Google Play

де

- text – текст коментаря, що залишив користувач
- $score \in \{1, 2, 3, 4, 5\}$ – оцінка, яку поставив користувач додатку

Надалі будемо називати це джерело Google Play Reviews.

3.1.2.2 Постановка задачі

Задача полягає у визначенні одного слова, що є найбільш позитивним, для кожного коментарю. Кандидатами будуть виступати усі слова.

Хоч ця задача не має прикладної користі, проте її розв’язок сприяє досягненню мети роботи.

Перед розв’язанням задачі необхідно розв’язати підзадачу числової оцінки того, наскільки кожне слово є позитивним для формування лейблів задачі ранжування.

3.1.2.3 Очистка даних. Приведення до формату, необхідного для розв’язку підзадачі

На етапі очистки даних, були видалені оцінки користувачів, що не містили тексту коментаря.

Список кандидатів, які можуть бути найпозитивнішим словом у коментарі, на даному кроці не формується, оскільки спочатку необхідно розв’язати підзадачу.

Було зроблено припущення, що якщо користувач поставив додатку оцінку «3» і вище, то його коментар є позитивним, інакше - негативним. Тому коментарям, що мали оцінку «3» і вище було присвоєно мітку «1», усім іншим – «0».

3.1.2.4 Векторизація

Для розв’язку підзадачі для набору даних Google Play Reviews необхідно векторизувати ці дані. Векторизовані дані будуть використані в подальшому при приведенні до формату необхідному для розв’язання основної задачі.

Для кожного коментаря виконаємо наступні дії:

1) Прибрати знаки наголосу з тексту.
2) Привести текст до нижнього регістру.
3) Розбити текст на слова, за допомогою регулярного виразу:

   r'(?u)(\b[a-z]{2,}\b|[\u263a-\U0001f645]|!|\?)'

   Тобто словом вважається будь-яка послідовність символів від «a» до «z» довжиною 2 та більше, смайлики та знаки оклику («!») і питання («?»).
4) Якщо коментар містить менше ніж 2 слова, прибрати його з розгляду
5) Лематизувати кожне слово за допомогою класу WordNetLemmatizer бібліотеки nltk [10].
6) Лематизувати усі слова зі списку sklearn.feature_extraction.text. ENGLISH_STOP_WORDS бібліотеки scikit-learn і прибрати усі такі слова зі слів документу.
7) Зі слів сформувати біграми.
8) Прибрати слова і біграми, що зустрічаються менше ніж 10000 разів серед усіх документів.
9) Векторизувати за технікою “Bag of words” (бінаризований).
10) Якщо отримано порожній вектор – видалити поточного кандидата та мітку, що йому відповідає з даних.

3.1.2.5 Розв’язок підзадачі

Добути чисельну оцінку того, наскільки «позитивним» є кожне зі слів коментарів можна наступним чином:

1) Провести процедуру тренування класифікатору MultinomialNB бібліотеки scikit-learn для визначення, чи є увесь коментар позитивним чи негативним.
2) У якості числових оцінок «позитивності» кожного слова, взяти відповідні умовні ймовірності, що модель оцінила за даними: $P(x_i|y = 1)$, $i = \overline{1,n}$, де n – довжина словника, сформованого при векторизації даних.

Таким чином, підзадача знаходження числових оцінок «позитивності» кожного слова розв’язана.

3.1.2.6 Приведення формату векторизованих даних до формату, необхідного для розв’язку основної задачі

Наразі, кожний коментар представлено вектором, сформованим за технікою «Bag of words» (кодує більше одного слова). Для того, щоб ці дані можна було використовувати для розв’язку основної задачі, для кожного вектору необхідно виконати наступні дії:

1) Перетворити вектор «Bag of words» (довжини n) на масив «One hot encoded» векторів, тобто на масив векторів, кожний елемент якого є

вектором довжини n, але містить лише один не нульовий елемент рівний «1».

2) Присвоїти мітку «1» тому з отриманих векторів, який кодує слово, з максимальною оцінкою «позитивності» серед усіх слів поточного коментаря, іншим векторам присвоїти мітку «0» (згідно оцінок, отриманих при розв’язанні підзадачі).

Отримані дані можна використовувати для розв’язку основної задачі ранжування.

3.2 Алгоритм експериментів

3.2.1 Програмна реалізація

Усі експерименти проводились за допомогою мови програмування Python у обчислювальному середовищі Jupyter Notebook у середовищі розробки DataSpell.

Було використано бібліотеки: Scikit-learn, Xgboost, nltk, Numpy, Pandas, Scipy, Joblib.

Код функції, за допомогою якої відбувається перетворення даних, сформованих за класичним pointwise підходом до формату даних, сформованих за конкурентним підходом, наведено у Додатку А.

Увесь інший код відповідає процесам обробки даних, описаним у попередніх розділах, тому детально розглядатись не буде.

3.2.2 Формат даних

Усі дані, що використовуються під час проведення експерименту, векторизовані за технікою bag-of-words (one hot encoding). Дані є 3-вимірними і мають такі розмірності:

$$(samples, candidates, n)$$

Де

samples – кількість документів/коментарів

candidates – кількість кандидатів (для кожного документа/коментаря різна)

n – розмір словника, створеного за тактикою «Bag-of-words» (різний для кожної задачі)

Зауваження**.** Перед тренуванням моделі, тренувальна вибірка (алгоритм отримання описано далі) приводиться до двовимірного формату шляхом вертикальної конкатенації кандидатів усіх документів/коментарів.

Вхідний вектор моделі складається з двох зконкатенованих частин однакової розмірності n:

1) Вектору ознак кандидата
2) Вектору ознак конкурентів кандидата

Таким чином, вхідний вектор моделі має розмір $2 \cdot n$.

3.2.1 Тренування і тестування моделей

Тренування і тестування моделі відбувається по різному для великих і малих наборів даних, тому розглянемо ці процеси для кожного набору даних окремо.

3.2.1.1 CUAD V1

Цей набір даних відносно невеликий, тому для отримання коректних результатів тренування та тестування потрібно проводити згідно з процесом cross-validation. З метою зменшення стандартної помилки метрик, тренування та тестування було проведено згідно з процесом repeated-cross-validation [13].

Зауваження**.** Для коректної оцінки якості моделі, векторизацію даних необхідно проводити *після* розбиття даних на множини для тренування і тестування, що є одним з етапів repeated-cross-validation. Але оскільки метою роботи є відносне порівняння результатів моделей в залежності від вхідних даних, то з метою спрощення процесу експерименту, векторизацію було проведено перед розбиттям на множина для тренування і тестування.

Параметр k для кожного повтору cross-validation було обрано рівним 10.

Оптимальне значення кількості повторів процесу cross-validation, було визначено з даних таблиці 3.1:

Таблиця 3.1 – Метрики та стандартні помилки

r	roc-auc mean	roc-auc sem	precision mean	presision sem
1	0,990098	0,003348	0,949942	0,010896
2	0,990137	0,002107	0,951103	0,006566
3	0,989915	0,001807	0,951510	0,005285
4	0,990159	0,001687	0,951699	0,004930
5	0,990430	0,001443	0,951823	0,004271
6	0,990272	0,001312	0,951094	0,004018
7	0,990122	0,001234	0,951261	0,003672
8	0,990279	0,001108	0,951684	0,003543
9	0,990103	0,001066	0,951491	0,003289
10	0,990052	0,001035	0,951812	0,003155
11	0,990175	0,000979	0,952069	0,002989
12	0,990214	0,000924	0,951689	0,002879
13	0,990305	0,000874	0,951925	0,002812
14	0,990339	0,000835	0,95212	0,002725
15	0.990288	0.000813	0.951967	0.002617

Тут

roc-auc mean – середнє значення метрики ROC-AUC за усіма ітераціями

roc-auc sem – стандартна помилка метрики ROC-AUC за усіма ітераціями

precision mean – середнє значення метрики precision за усіма ітераціями

precision sem – стандартна помилка метрики precision за усіма ітераціями

Як бачимо, після 7 повторів, швидкість зменшення стандартної помилки знижується, тому для подальших експериментів було обрано саме 7 повторів.

3.2.1.2 Google Play Reviews

Цей набір даних доволі великий, тому для отримання коректних результатів немає необхідності використовувати repeated-cross-validation. Тому експерименти було проведено за допомогою 4-fold cross-validation.

3.2.2 Моделі класифікації

У якості локальних функцій ранжування у роботі були використані наступні класифікатори:

1) MultinomialNB бібліотеки scikit-learn.
2) LogisticRegression бібліотеки scikit-learn.
3) KNeighborsClassifier бібліотеки scikit-learn.
4) SVC бібліотеки scikit-learn.
5) DecisionTreeClassifier бібліотеки scikit-learn.
6) XGBClassifier бібліотеки xgboost [14].

Для кожної моделі перед її застосуванням до кожного набору даних було підібрано такі гіперпараметри, що забезпечують максимальне значення метрики $precision$ при розв'язку задачі ранжування (див. табл. 3.2). Для моделі SVC для набору даних Google Play Reviews параметри підібрати не вдалось через високу складність обчислень, тому і експерименти з цією моделлю на даному наборі даних проводитись не будуть.

Таблиця 3.2 – Гіперпараметри моделей класифікації

Модель	Параметри	CUAD V1	CUAD V1 Concurrent	GP Reviews	GP Reviews Concurrent
MultinomialNB	alpha	0.8	0.4	0.001	0.01
LogisticRegression	solver	liblinear	liblinear	liblinear	liblinear
	penalty	l1	l2	l2	l2
	C	0.01	0.01	2	10
KNeighborsClassifier	n_neighbors	5	uniform	15	3
	weights	5	distance	uniform	uniform
SVC	C	2	auto	-	-
	gamma	1	auto	-	-
DecisionTreeClassifier	criterion	gini	entropy	entropy	entropy
	min_samples_leaf	35	9	1	1
	max_depth	None	None	1200	None
	min_samples_split	2	2	2	100
XGBClassifier	booster	gbtree	gbtree	gbtree	gbtree
	num_parallel_tree	10	10	1	1
	n_estimators	20	20	100	100
	max_depth	20	20	900	900
	learning_rate	0.3	0.1	0.5	0.25
	reg_alpha	0.2	0.15	0.01	0.5
	reg_lambda	0.2	0.15	1	0.5
	colsample_bytree	0.1	1	1	1
	colsample_bylevel	1	1	1	0.2

Тут

GUAD V1 – дані з набору CUAD V1, сформовані за класичним pointwise підходом.

CUAD V1 Concurrent - дані з набору CUAD V1, сформовані за конкурентним підходом.

GP Reviews – дані з набору Google Play Reviews, сформовані за класичним pointwise підходом.

GP Reviews Concurrent - дані з набору Google Play Reviews, сформовані за конкурентним підходом.

Для набору даних CUAD V1 підбір параметрів здійснювався за допомогою процесу $cross - validation$ з параметром $k = 10$ (без повторів, оскільки повтори потребують у цьому випадки багато обчислювальних ресурсів).

Для набору даних Google Play Reviews підбір параметрів здійснювався за допомогою розбиття набору даних на вибірки для тренування і тестування об’ємом 75% і 25% відповідно від загальної кількості даних у наборі.

Дані, сформовані за конкурентним підходом, бінаризувались для моделей MultinomialNB і KNeighborsClassifier, оскільки вони, через свою природу, показували значно гірші результати на не бінаризованих даних. Інші ж моделі навпаки, показували кращий результат на не бінаризованих даних, оскільки такі дані містять більше інформації.

3.2.3 Порівняння результатів

Оскільки експерименти для кожної моделі на кожному наборі даних відбуваються за процедурою cross-validation або repeated-cross-validation, то в результаті отримуємо не одне значення, а вибірку значень за кожною метрикою.

Виникає необхідність перевірити, чи середнє значення вибірки однієї з метрик, отриманої при тестуванні моделі на даних, які сформовані за класичним pointwise підходом, відрізняється від середнього значення вибірки тієї ж метрики, отриманої при тестуванні тієї ж моделі на даних, які сформовані за конкурентним підходом.

Оскільки не вдалось визначити, що вибірки метрик мають нормальний розподіл, то для порівняння середніх значень таких вибірок було вирішено використовувати непараметричний Mann-Whitney U тест [15]. Критичне значення для p-значення даного тесту було обрано 0.05 (якщо отримане p-значення менше, то гіпотеза про рівність середніх значень вибірок відхиляється).

3.3 Результати експерименту

Результати експерименту для набору даних CUAD V1 наведено у таблиці 3.3:

Таблиця 3.3 – Результати експерименту. CUAD V1

Модель	Метрики	CUAD V1	CUAD V1 Concurrent	p-значення
MultinomialNB	precision	92.5 %	93.3 %	0.12
	ROC-AUC	96.6 %	95.8 %	$5e-5$
LogisticRegression	precision	94 %	96.4 %	$1.3e-5$
	ROC-AUC	96.7 %	99.3 %	2.2e-19
KNeighborsClassifier	precision	95.7 %	89 %	$6e-10$
	ROC-AUC	97.6 %	97 %	$6e-2$
SVC	precision	96.3 %	96.2 %	$7.1e-1$
	ROC-AUC	99.2 %	99.2 %	1.0
DecisionTreeClassifier	precision	94.2 %	92 %	$1.2e-2$
	ROC-AUC	98 %	96 %	$3.6e-9$
XGBClassifier	precision	96 %	94.9 %	$1e-1$
	ROC-AUC	99.4 %	98.6 %	$2.3e-7$

Результати експерименту для набору даних Google Play Reviews наведено у таблиці 3.4:

Таблиця 3.4 – Результати експерименту. Google Play Reviews

Модель	Метрики	GP Reviews	GP Reviews Concurrent	p-значення
MultinomialNB	precision	81.4 %	80.1 %	$2.9e-2$
	ROC-AUC	94.4 %	94 %	$2e-1$
LogisticRegression	precision	81 %	90.6 %	$2.9e-2$
	ROC-AUC	94.2 %	97.2 %	$2.9e-2$
KNeighborsClassifier	precision	71.5 %	63 %	$2.9e-2$
	ROC-AUC	88.2 %	80.5 %	$2.9e-2$
DecisionTreeClassifier	precision	81.4 %	75.5 %	$2.9e-2$
	ROC-AUC	94.2 %	87.6 %	$2.9e-2$
XGBClassifier	precision	81.2 %	87.5 %	$2.9e-2$
	ROC-AUC	93.9 %	96 %	$2.9e-2$

У таблиці 3.3 і 3.4:

precision – середнє значення метрики precision за результатами (repeated-) cross-validation.

ROC-AUC – середнє значення метрики ROC-AUC за результатами (repeated-) cross-validation.

p-значення – p-значення Mann-Whitney U тесту

Червоні комірки – середнє значення метрики на даних, сформованих за класичним pointwise підходом, *нижче* за середнє значення цієї ж метрики на даних, сформованих за конкурентним підходом, і різницю у середніх значеннях

можна вважати статистично значимою.

Зелені комірки – середнє значення метрики на даних, сформованих за класичним pointwise підходом, *вище* за середнє значення цієї ж метрики на даних, сформованих за конкурентним підходом, і різницю у середніх значеннях можна вважати статистично значимою.

3.4 Логістична регресія. Аналіз коефіцієнтів

З усіх розглянутих моделей класифікації, параметри моделі логістичної регресії, що визначаються під час тренування, є найбільш простими і зрозумілими. Тому аналіз її параметрів може покращити розуміння впливу конкурентного підходу до формування даних на якість розв'язку задачі ранжування.

Для кращого розуміння змісту параметрів моделі логістичної регресії, розглянемо алгоритм навчання цієї моделі:

1) Обчислити:

$$y(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + \cdots + x_{2 \cdot n} \cdot w_{2 \cdot n}$$

Де

x_i – елемент вхідного вектору даних

w_i – коефіцієнти (ваги/параметри) моделі

2) Обчислити:

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

Де

$p(x)$ – ймовірність приналежності вектору x до позитивного класу у задачі бінарної класифікації

3) Обчислити значення логістичної функції помилки:

$$error\_binary(x) = -\big(y \cdot log\big(p(x)\big) + (1 - y) \cdot \log\big(1 - p(x)\big)\big)$$

4) Оновити значення ваг, з метою зменшення значення функції помилки. У якості реалізації логістичної регресії у роботі використовується модель LogisticRegression бібліотеки sklearn, яка для оновлення ваг використовує алгоритм Limited-memory Broyden–Fletcher–Goldfarb–Shanno [16].

Отже, кожний з коефіцієнтів моделі логістичної регресії відповідає деякому елементу вектору ознак, а оскільки вектор ознак створюється за технікою bag-of-words (one-hot-encoding), то можна сказати, що кожний з коефіцієнтів моделі відповідає деякому слову із словника, що утворено з унікальних слів набору даних при векторизації.

Тоді, якщо модель має позитивний коефіцієнт, що відповідає деякому слову словника, то це означає, що наявність даного слова у кандидата збільшує ймовірність того, що цей кандидат є шуканою сутністю (негативні коефіцієнти зменшують цю ймовірність).

Розглянемо детальніше коефіцієнти моделі логістичної регресії, навченої на даних, які сформовані за конкурентним підходом.

Якщо розмір словника, сформованого під час векторизації bag-of-words (one-hot-encoding), дорівнює $n \in \mathbb{N}$, то розмір вектору вхідних даних моделі дорівнює $2 \cdot n$, а отже і розмір вектору коефіцієнтів моделі дорівнює $2 \cdot n$. Позначимо вектор коефіцієнтів моделі як $c \in \mathbb{R}^{2 \cdot n}$.

Будемо проводити порівняння i-го коефіцієнту з $(i+n)$-м коефіцієнтом $(i=\overline{1,n})$.

Говоритимемо, що коефіцієнт *змінив знак,* якщо:

$$\mathrm{c}[i]\cdot c[i+n]<0, i=\overline{1,n}$$

Говоритимемо, що коефіцієнт *змінив знак сильно*, якщо:

$$c[i]>0.1 \text{ і } c[i+n]<-0.1 \text{ або } c[i]<-0.1 \text{ і } c[i+n]>0.1, i=\overline{1,n}$$

i −у ознаку вхідного вектору називатимемо *сильною позитивною,* якщо $c[i]>0.1$.

i −у ознаку вхідного вектору називатимемо *сильною негативною,* якщо $c[i]<-0.1$.

Тобто, порівнюватимемо коефіцієнти моделі, що відповідають одним і тим самим словам у першій половині вектору вхідних даних (яка кодує кандидата) і у другій його половині (яка кодує конкурентів).

У таблиці 3.5 наведено відсотки сильно позитивних і сильно негативних коефіцієнтів, які змінили знак і змінили знак сильно.

Таблиця 3.5 – Результат порівняння коефіцієнтів логістичної регресії

	CUAD V1		Google Play Reviews	
	Змінили знак	Змінили знак сильно	Змінили знак	Змінили знак сильно
Сильно позитивні коефіцієнти	95%	76%	85%	80%
Сильно негативні коефіцієнти	92%	52%	50%	32%

3.5 Аналіз результатів

Як бачимо з таблиці 3.3 та таблиці 3.4, конкурентний підхід до формування даних не покращив або погіршив результати для усіх моделей на обох наборах даних, окрім логістичної регресії і XGBClassifier на наборі даних Google Play Reviews.

З таблиці 3.5 бачимо, що серед коефіцієнтів з сильним позитивним значенням змінили знак і змінили знак сильно суттєво більше коефіцієнтів у порівнянні з коефіцієнтами з сильно негативними значеннями.

Це можна пояснити тим, що модель вивчила доволі інтуїтивну закономірність:

- Якщо *хоча б один* з конкурентів має сильну ознаку, то з високою ймовірністю *можна* стверджувати, що поточний кандидат *не є вірним*.
- Якщо сильна негативна ознака присутня в *одного* конкурента, то цього *не достатньо* аби з високою ймовірністю стверджувати, що поточний

кандидат *є вірним*.

- Якщо сильні негативні ознаки присутні в *багатьох* конкурентів, то з високою ймовірністю *можна* стверджувати, що поточний кандидат *є вірним.*

Отримані результати можуть мати дві причини:

1) Прокляття розмірності. Це явище полягає у тому, що зі збільшенням розмірності даних, зростає складність їх аналізу, і тому кількість прикладів даних для тренування моделі повинна зростати з експоненційною швидкістю [17]. В експериментах, вектори вхідних даних збільшили свій розмір удвічі, проте кількість елементів у кожному наборі даних лишилась незмінною.
2) Завдяки особливостям своєї будови і різним рівням складності (зашумленості) наборів даних, деякі моделі змогли вивчити закономірність, яка з’явилась у даних завдяки конкурентному способу їх формування, що дозволило їм не тільки не постраждати від «прокляття розмірності» але і значно покращити свій результат.

ВИСНОВКИ

У роботі розглянуто задачу ранжування з метрикою WTA, як окремий випадок задачі NER а також запропоновано і перевірено ефективність конкурентного підходу до формування вхідного вектору даних для моделі класифікації, яка використовувалась у якості локальної функції ранжування.

Було з’ясовано, що конкурентний підхід до формування даних має не однозначний вплив на якість ранжування, а саме:

1) При конкурентному підході до формування даних, якість ранжування з використанням логістичної регресії і XGBClassifier (на наборі даних Google Play Reviews) суттєво покращилась. Отже можна зробити висновок, що таких підхід до форматування даних дозволяє моделі отримати більше корисної інформації про дані, що ранжуються, у порівнянні з класичним pointwise підходом.
2) В усіх інших випадках, при конкурентному підході до формування даних якість ранжування, не покращилась або погіршилась. Можливими причинами можуть бути:
    а) Явище «прокляття розмірності», оскільки при конкурентному підході розмірність векторів, що вводяться у модель класифікації, збільшується вдвічі.
    б) Нова інформація, яку може отримати модель при конкурентному підході, має складну структуру і не усі моделі можуть отримати від неї користь, через особливості своєї будови.

    З’ясування точної причини потребує подальших досліджень.

Виходячи з отриманих результатів, подальшими напрямками досліджень можуть бути:

1) Провести експерименти на даних із зниженою розмірністю. Наприклад, можна залишити фіксовану кількість найбільш інформативних ознак або застосувати такі техніки зниження розмірності як PCA або t-SNE [17].
2) Провести експерименти, у яких тренувальні дані, сформовані за конкурентним підходом, будуть містити більше елементів, у порівнянні з тренувальними даними, сформованими за класичним pointwise підходом.
3) Провести експерименти з використанням нейронних мереж у якості класифікатору. Оскільки вони за будовою схожі на логістичну регресію, то конкурентний підхід по формування даних може значно покращити результати ранжування. Наприклад, можна протестувати такі архітектури нейронних мереж як FCNN, CNN, RNN [18], Fully Convolutional Neural Networks [19] (можливо, така архітектура дозволить проводити ранжування за один запуск моделі).
4) Провести порівняння pointwise підходу з використанням даних, сформованих за конкурентним підходом, з pair-wise і list-wise підходами.
5) З’ясувати вплив конкурентного підходу до формування даних на якість ранжування з використанням метрик відмінних від WTA.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adnan Kiran, Akbar Rehan An analytical study of information extraction from unstructured and multidimensional big data. *Journal of Big Data*. October 2019. №6. DOI: 10.1186/s40537-019-0254-8.
2. Nadeau David, Sekine Satoshi A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*. August 2007. №30. DOI: 10.1075/li.30.1.03nad.
3. LI Hang A Short Introduction to Learning to Rank. *IEICE TRANSACTIONS on Information and Systems.* 01.10.2011. P. 1854-1862. DOI: 10.1587/transinf.E94.D.1854.
4. LETOR: A benchmark collection for research on learning to rank for information retrieval / Qin Tao, Liu Tie-Yan, Xu Jun, Li Hang. *Information Retrieval*. 2010. P. 346-374. DOI: 10.1007/s10791-009-9123-y.
5. re — Regular expression operations / Python Software Foundation. Дата оновлення: 08.06.2022. URL: https://docs.python.org/3/library/re.html (дата звернення: 09.06.2022).
6. Bradley Andrew P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition.* 1997. №30. P. 1145-1159. DOI: 10.1016/S0031-3203(96)00142-2.
7. Margherita Grandini, Enrico Bagli, Giorgio Visani Metrics for Multi-Class Classification: an Overview. *arXiv*. 2020. URL: https://arxiv.org/pdf/2008.05756.pdf.
8. The atticus project / The Atticus Project, Inc. 10.03.2021. URL: https://www.atticusprojectai.org/cuad (дата звернення: 09.06.2022).
9. Google Play / Google. URL: https://play.google.com/store/apps (дата звернення: 09.06.2022).

10. NLTK::Natural Language Toolkit / NLTK Project. URL: https://www.nltk.org/ (дата звернення: 09.06.2022).
11. scikit-learn: machine learning in Python. URL: https://scikit-learn.org/stable/index.html (дата звернення: 09.06.2022).
12. google-play-scraper / Olano Facundo. Дата оновлення: 04.02.2019. URL: https://github.com/facundoolano/google-play-scraper (дата звернення: 09.06.2022).
13. Kim Ji-Hyun Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*. 2009. №53. P. 3735-3745. DOI: 10.1016/j.csda.2009.04.009.
14. XGBoost Documentation. URL: https://xgboost.readthedocs.io (дата звернення: 09.06.2022).
15. scipy.stats.mannwhitneyu. URL: https://docs.scipy.org/doc/scipy/reference/ generated/scipy.stats.mannwhitneyu.html (дата звернення: 09.06.2022).
16. Morales Jos'e Luis, Nocedal Jorge Remark on Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound Constrained Optimization. *ACM Trans. Math. Softw.* 12.2011. №38. P. 4. DOI: 10.1145/2049662.2049669.
17. Carreira-perpiñán, Miguel Á. A review of dimension reduction techniques. 1997. URL: http://faculty.ucmerced.edu/mcarreira-perpinan/papers/ cs-96-09.pdf.
18. From Classical Machine Learning to Deep Neural Networks: A Simplified Scientometric Review / Mukhamediev Ravil I., Symagulov Adilkhan, Kuchin Yan, Yakunin Kirill, Yelis Marina. *Applied Sciences*. 2021. №11. DOI: 10.3390/app11125541.
19. Multi-Digit Recognition Using a Space Displacement Neural Network /

Matan Ofer, Burges Christopher J. C., LeCun Yann, Denker John. *Advances in Neural Information Processing Systems.* 1991. №4. URL: https://proceedings.neurips.cc/paper/1991/file/6e2713a6efee97bacb63e52c54f0ada0-Paper.pdf.

ДОДАТОК А

Функція формування конкурентних даних

```
def doc_concurrent_data(doc, binarize=False):
    """
    Convert list of candidates, to list of concurrent
candidates
    :arg doc - list of candidates in format of
scipy.sparse.csr.csr_matrix
    :arg binarize - whether to binarize each of
concurrent candidates
    :returns - list of concurrent candidates in format of
scipy.sparse.csr.csr_matrix
    """
    concurrent_list = []
    n = doc.shape[0]
    for idx, candidate in enumerate(doc):
        concurrent_mask = [True] * n
        concurrent_mask[idx] = False
        concurrent_candidates = doc[concurrent_mask]
        concurrent = concurrent_candidates.sum(axis=0)
        if binarize:
            concurrent[concurrent > 0] = 1
        concurrent = csr_matrix(concurrent)
        concurrent_list.append(concurrent)

    concurrent_sparse = vstack(concurrent_list,
format='csr')
    result = hstack([doc, concurrent_sparse],
format='csr')

    return result
```

Перевірку кваліфікаційної роботи на унікальність було проведено за допомогою сайту https://plag.com.ua/

Нормоконтроль пройдено.