Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ РАНЖИРУВАННЯ НА ОСНОВІ МАШИННОГО НАВЧАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF RANKING METHODS BASED ON MACHINE LEARNING

Виконав: студент(ка) 2 курсу, групи КНТ-119м
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Інженерія програмного забезпечення
Анурєєв С.В.
(прізвище та ініціали)
Керівник Шитікова О.В.
(прізвище та ініціали)
Рецензент Терещенко Е.В.
(прізвище та ініціали)

2020

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121Інженерія програмного забезпечення
(код і назва)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н., проф.
_____С.О. Субботін
“____” _____2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Анурєєва Сергія Володимировича
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів ранжирування на основі машинного навчання. Research and Software Implementation of Ranking Methods Based on Machine Learning

керівник проєкту (роботи) Шитікова Олена Вікторівна, к.т.н.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 03 листопада 2020 року №301
2. Строк подання студентом проєкту (роботи) 5 грудня 2020 року
3. Вихідні дані до проєкту (роботи) технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали та методи. 3. Розробка архітектури програми. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Економіко-організаційна частина. 7. Охорона праці та безпека у надзвичайних ситуаціях. 8. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–5 Основна частина	Шитікова О.В., доцент		
6 Економіко-організаційна частина	Пожуєва Т.О., професор		
7 Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Андреєв М.О., асистент		

7. Дата видачі завдання 02.10.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	1 тиждень	
2	Розробка та затвердження технічного завдання	2 тиждень	Технічне завдання
3	Проєктування структури системи	3-4 тиждень	Структурна схема
4	Розробка схеми функціонування системи та модулів програми	5-6 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи	7-8 тиждень	Текст програми
6	Тестування та відлагодження програми	9 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки	10 тиждень	ПЗ
8	Розробка слайдів презентації	11-12 тиждень	Слайди презентації
9	Захист роботи	13 тиждень	

Студент(ка) _____ Анурєєв С.В.
(підпис) (прізвище, ініціали)

Керівник проєкту (роботи) _____ Шитікова О.В.
(підпис) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 123 с., 21 рис., 18 табл., 4 додатка, 43 джерела.

РАНЖИРУВАННЯ, МАШИННЕ НАВЧАННЯ, ІНФОРМАЦІЙНИЙ ПОШУК, РЕЛЕВАНТНІСТЬ, КЛАСТЕРИЗАЦІЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, ДОСЛІДЖЕННЯ, РОЗРОБКА.

Об’єкт дослідження – процес ранжирування на основі машинного навчання.

Предмет дослідження – ранжирування на основі машинного навчання.

Мета магістерської роботи – дослідження та програмна реалізація методів ранжирування на основі машинного навчання, а саме на основі використання кластеризації та штучних нейронних мереж.

Матеріали, методи та технічні засоби: для успішної реалізації поставленої мети у роботі використовуються поєднання сучасних методів та підходів, а саме методу кластеризації k-means та штучні нейронні мережі Кохонена. Для роботи буде використаномову програмування Python, середовище розробки Anaconda та бібліотека kohonenv.1.1.2. Ноутбук HP Pavilion Gaming 16 в наступній конфігурації: Intel Core i7-10750H (2.6 - 5.0 ГГц), RAM 8 ГБ, SSD 1 ТБ, nVidia GeForce GTX 1660 Ti Max-Q, 6 ГБ.

Результати. Розроблено програмний комплекс, що реалізує запропонований метод ранжирування на основі машинного навчання для підвищення релевантності при інформаційному пошуку, а саме використовується методкластеризації k-means та штучні нейронні мережі Кохонена.

Висновки. Було запропоновано новий метод ранжирування на основі машинного навчання, а саме методу кластеризаціїї k-середніх та штучних

нейронних мереж Кохонена. Також було розроблено програмний комплекс, що реалізує роботу запропонованого методу.

Галузь використання – систем інформаційного пошуку.

ABSTRACT

Explanatory note of the final qualifying work of the master: 123 pages, 21 figures, 18 tables, 4 appeendixes, 43 sources.

RANKING, MACHINE LEARNING, INFORMATION SEARCH, RELEVANCE, CLUSTERING, ARTIFICIAL NEURAL NETWORKS, RESEARCH, DEVELOPMENT.

Object of study is the process of the ranking methods based on machine learning.

Subject of study is ranking based on machine learning.

The purposes of the work are research and software implementation of ranking methods based on machine learning, namely, based on the use of clustering and artificial neural networks.

Materials, methods and tools: to successfully implement this goal, the paper uses a combination of modern methods and approaches, namely the k-means clustering method and Kohonen artificial neural networks. The Python programming language, the Anaconda development environment, and the kohonen v.1.1.2 library will be used for this work. HP Pavilion Gaming 16 laptop in the following configuration: Intel Core i7-10750h (2.6-5.0 GHz), 8 GB RAM, 1 TB SSD, nVidia GeForce GTX 1660 Ti Max-Q, 6 GB.

Results. A software package has been developed that implements the proposed ranking method based on machine learning to increase relevance in information search, namely, the K-means clustering method and Kohonen artificial neural networks are used.

Conclusions. A new ranking method based on machine learning was proposed, namely the K-core clustering method and Kohonen artificial neural networks. A software package was also developed that implements the proposed method.

The field of use information search systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GUI – graphical user interface;

HITS – hyperlink-induced topic search;

MAP – mean average precision;

TREC – text retrieval conference;

URL – uniform resource locator;

БД – база даних;

ІП – інформаційний пошук;

НМ – штучними нейронними мережами;

ПЕОМ – персональна електронна обчислювальна машина;

ПЗ – програмного забезпечення;

ПП – програмний продукт;

СТ – середня точність.

ВСТУП

Пошук в колекціях документів є важливим завданням. Про це свідчить як велика кількість пошукових систем, так і їх постійний розвиток. Колекції документів можуть бути різних типів: блоги, новинні стрічки, наукові статті або всі безліч веб-сторінок. Пошукові системи, такі як Google або Yahoo, оперують з останнім типом колекцій. Принцип роботи пошукової системи наступний: користувач вводить запит, після чого система повертає ті документи з колекції, які найкращим чином задовольняють запиту. Як правило, в традиційних пошукових системах ранжирування документів (визначення релевантності документа запиту) проводиться на основі статистичної інформації про безліч слів в запиті і в документі. На відміну від пошуку в базі даних, де результатом є безліч записів з ключами, що задовольняють логічній умові, результат пошуку в колекції документів не визначений точно. Тому вводяться параметри точності і повноти, що відображають якість пошуку.

Те, які документи будуть видані користувачеві у відповідь на конкретний запит, визначає функція ранжирування, або функція схожості. Традиційні пошукові системи оперують з векторною моделлю документа, в якій запит і документ розглядаються як N-мірні вектори, де N – загальна кількість термінів у системі [1]. Компоненти вектора – дійсні ваги, що відображають важливість кожного відповідного терміна. Функція схожості визначається як косинус кута між векторами запиту і документа. Завдання ранжирування в даному випадку зводиться до задачі визначення ваг для термінів.

Існує ряд проблем, які не вдається вирішити, якщо функції ранжирування ґрунтуються тільки на вмісті самих документів і запиту, як це відбувається у векторній моделі. Наведемо приклади небажаних документів, тобто тих документів, які не є релевантними до запиту, але мають високий рейтинг в рамках векторної моделі. Наприклад, це документи, що містять

рекламу і підвищують свій рейтинг одним або декількома з наступних нечесних способів:

– штучне підвищення частоти ключового слова для підвищення його ваги;

– додавання невидимого тексту, колір якого дорівнює кольору фону або розмір шрифту якого дорівнює 0;

– маскування – передача пошуковій машині тексту документа, що відрізняється від того, який бачать користувачі.

Інший приклад – документ-програма конференції з інформаційної безпеки. Набираючи в рядку пошуку назву якогось з криптографічних алгоритмів, користувач цілком може отримати текст-програму конференції, незважаючи на те, що цей текст не містить в собі опис цього алгоритму. Цей текст містить багато різних термінів, що відносяться до певної області, але може задовольнити лише малу частину інформаційних потреб, пов'язаних з даною областю знань, так як не описує конкретних понять, а лише перераховує їх.

Саме тому, актуальною задачею є вивчення поведінки алгоритмів текстового ранжирування, для підвищення якості та релевантності результатів пошукових семантичних систем, в тому числі при вирішенні завдань розпізнавання і адаптивної класифікації об'єктів за даними.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Перший розділ буде присвячено детальному аналізу проблемної області ранжирування та розглядається задача інформаційного пошуку. Детельно розглядаються різні методи визначення статичної якості документів, а також варіанти побудови підсумкової функції ранжирування, що комбінує кілька ознак різної природи. Розглядаються вимоги, що пред'являються до колекції документів (наприклад, наявність посилальної структури) в кожному конкретному методі, складність реалізації, можливості та ефективність використання різних характеристик в якості ознак для ранжирування.

1.1 Інформаційний пошук та ранжирування

Термін «інформаційний пошук» (ІП) має багато значень. У даній роботі він використовується в якості перекладу оригінального варіанту англійською мовою «information retrieval» [1]. Тому під пошуком більш мається на увазі «Витяг інформації», ніж сам «пошук». Термін має багато значень, і зчитування номера кредитної картки теж можна віднести до способу пошуку інформації.

ІП – це знаходження матеріалів (зазвичай документів) неструктурованої природи (зазвичай текст), які задовольняють потребу в інформації всередині великих колекцій (зазвичай збережених на обчислювальних пристроях) [1], [2].

Знаходження листів в електронній пошті, сайтів в Інтернет – все це відноситься до пошуку інформації в даному визначенні. Такий метод доступу до інформації починає домінувати серед інших форм доступу до інформації, переважаючи, наприклад, над пошуком в реляційних базах даних, що мають сувору структуру. Термін «неструктуровані дані» відноситься до даних, які не мають структури, яка могла б бути легко оброблена комп'ютером. В реальності, по-справжньому неструктурованих даних практично немає. Текст природною мовою має «лінгвістичну» структуру або, принаймні, виділені параграфи,

абзаци та речення. ІП також використовується в разі напівструктурованого пошуку в разі, наприклад, пошуку документів, в заголовку яких міститься слово «Java», а в основному тексті – слово «багатозадачність» [3].

Системи ІП можна розділити на три категорії відповідно до масштабу колекцій, з якими вони оперують [4]. Системи пошуку в інтернет оперують з мільярдами документів, збереженими на мільйонах різних обчислювальних пристроїв. Такі системи повинні володіти рядом відмінних рис, щоб мати можливість здійснювати ефективний пошук документів в таких масштабах, а також обробляти гіпертекст і запобігати можливість підвищення релевантності окремих сайтів обманними шляхами. Інший вид пошукових систем – персональні. До даної категорії можна віднести, наприклад, інтегрований в операційну систему пошук по комп'ютеру, або пошук серед листів електронної пошти [5]. Поштові клієнти можуть також надавати функції класифікації тексту, такі як, наприклад, виявлення небажаної пошти. До відмінних рис систем персонального пошуку можна віднести наступні: виконання на одній машині, необхідність низького споживання системних ресурсів, таких як процесорний час і дисковий простір, малий час ініціалізації. Проміжну позицію займають системи корпоративного пошуку, а також пошуку всередині предметної області. Це можуть бути системи пошуку по внутрішній базі документів корпорації, базі даних патентів, або базі наукових статей в області біохімії. У таких системах документи, як правило, зберігаються в централізованому сховищі, а пошук здійснюється однією або декількома машинами. У даній роботі увага акцентується на першому з трьох типів систем. Для наочності, в наступному розділі описується простий приклад завдання інформаційного пошуку.

1.1.1 Приклад завдання інформаційного пошуку

Розглянемо приклад з пошуком в повному зібранні творів Вільяма Шекспіра [6], [7]. Припустимо, ставиться завдання знаходження всіх творів, в

яких зустрічаються слова Брут і Цезар, але не зустрічається Калпернія. Очевидний спосіб вирішення-прочитати всі твори від початку до кінця, відзначаючи ті, які містять необхідні слова, і викреслюючи ті, які містять зайві. Так як повне зібрання творів Шекспіра містить менше мільйона слів, на сучасному комп'ютері метод лінійного проходу за всіма документами досить ефективний. Але в багатьох випадках цього недостатньо, потрібен більш ефективний алгоритм, якщо необхідно [7], [8]:

– обробляти великі колекції. Обсяг даних в мережі зростає, принаймні, так само швидко, як і продуктивність комп'ютерів. Потрібно вміти обробляти колекції, що містять близько декількох трильйонів слів;

– здійснювати більш гнучкі запити. Наприклад, при лінійному проході не практикується обробляти запит «знайти всі документи, в яких слово Цар міститься близько слова «Іван», причому під «близько» мається на увазі в інтервалі довжиною не більше п'яти слів або в одному реченні;

– виконувати ранжируваний пошук: у багатьох випадках потрібен найбільш підходящий документ серед безлічі тих, які містять конкретні слова.

Щоб уникнути лінійного сканування тексту для кожного пошукового запиту, всю колекцію заздалегідь індексують, тобто створюють спеціальні структури даних, що дозволяють досягати потрібної продуктивності. Повернемося до нашого прикладу зі зборами творів Вільяма Шекспіра. Нехай в колекції міститься n документів та m різних слів (Шекспір використовував у своїх творах близько 32000 слів). Побудуємо таблицю, елемент (i, j) якої буде дорівнює 1 або 0 в залежності від того, чи міститься слово i в документі j (табл. 1.1).

Термін не завжди є словом, він може представляти кілька слів або число, але ми будемо вживати нарівні вирази термін та слово, маючи на увазі один індексований об'єкт.

Таблиця 1.1 – Зустрічальність різних термінів у творах Шекспіра

	Антоній та Клеопатра	Юлій Цезар	Буря	Гамлет	Отелло	Макбет	…
Антоній	1	1	0	0	0	1	
Брут	1	1	0	1	0	0	
Цезар	1	1	0	1	1	1	
Калпернія	0	1	0	0	0	0	
Клеопатра	1	0	0	0	0	0	
Милість	1	0	1	1	1	1	
Гірше	1	0	1	1	1	1	
…							

Щоб відповісти на вихідний пошуковий запит за допомогою таблиці 1.1, достатньо двох булевих операцій над векторами: 110100 AND 1101111 AND 101111 = 100100 [8], [9].

Отже, відповідними творами є «Антоній і Клеопатра» та «Гамлет». Розглянутий сценарій відноситься до булевої моделі пошуку, в якій терміни комбінуються за допомогою операторів AND, OR та NOT. Ця модель розглядає документи як множини термінів.

Розглянемо більш реалістичний сценарій. Нехай є $N = 1000000$ документів. Під документом мається на увазі мінімальна одиниця пошуку, це може бути глава в книзі, або повідомлення в блогу. Весь набір документів будемо називати колекцією. Припустимо, що кожен документ містить близько тисячі слів, що приблизно дорівнює двом-трьом сторінкам друкованого тексту. Для англійського тексту на 1 слово потрібно в середньому 6 байт, враховуючи прогалини і знаки пунктуації. Колекція в цьому випадку буде займати близько 6 гігабайт. Кількість різних термінів буде порядку $M = 500000$ [6-9]. Представлені міркування дають уявлення про порядок величин в даній задачі, вони не є формальними викладками.

Система повинна знаходити ті документи в колекції, які відповідають потреби в інформації. Під останньою розуміється певна тема, в області якої користувач бажає дізнатися більше. Користувач формує запит, який побічно

представляє його потреба в інформації, хоча і відрізняється від неї. Запит – це спроба отримати від системи потрібні документи. Кажуть, що документ є релевантним, якщо Користувач сприймає його як такий, що містить необхідну інформацію, що відповідає його потребі в інформації. Перший приклад був штучним, в ньому потреба в інформації полягала в тому, містить той або інший документ певні слова, в той час як зазвичай цікавиться, наприклад, «витоками трубопроводу», і хотів би отримати релевантні документи незалежно від того, містять вони вихідне поняття або виражають його через інші, такі як, наприклад, «розрив трубопроводу» [7] – [9].

Щоб оцінити ефективність пошукової системи, обчислюють наступні статистичні величини [1], [6] – [9]:

– точність – кількість релевантних документів в результаті, поділена на загальну кількість документів;

– повнота – кількість релевантних документів в результаті, поділена на загальну кількість релевантних документів в колекції.

При обсязі даних, описаних раніше, матриця зустрічальності термінів в документах, побудована наївним способом, буде займати занадто багато пам'яті: (500000×1000000 біт інформації. Дана матриця є сильно розрідженою, так як через те, що кожен документ містить близько 1000 слів, 99.8% елементів матриці є нулями.

Використання цієї особливості-центральна ідея побудови інвертованого індексу.

1.1.2 Інвертований індекс

Опишемо в загальному створення інвертованого індексу (по-іншому, інвертованого файлу). Є словник всіх термінів, існуючих в системі-об'єднання всіх окремих термінів документів. Для кожного терміна зберігається список документів, в яких зустрічається цей термін (інвертований список) [4] – [9].

Таблиця 1.2 – Приклад інвертованих списків

Словник	Інвертовані списки					
Брут	→	1	2	11	31	…
Цезар	→	1	2	5	6	…
Калпернія	→	2	31	54	101	…
…	…	…	…	…	…	…

Після побудови інвертованого індексу відповідь на запит "Брут і Калпенрія" може бути отримана за наступним алгоритмом:

– знайти «Брут» в словнику;

– отримати відповідний інвертований список;

– знайти «Калпернія» в словнику;

– отримати відповідний інвертований список;

– знайти перетин двох списків.

Останній пункт є вузьким місцем алгоритму. Грунтуючись на тому, що обидва списки відсортовані за зростанням ідентифікатора документа, існує простий алгоритм знаходження перетину цих списків За $O(n+m)$, де n та m – довжини цих списків, тобто $F(N)$, де N – загальна кількість документів.

1.1.3 Ранжируваний пошук

Ми описали Індекс, що підтримує булевий пошук, тобто відповідь на питання, які документи в точності задовольняють запиту, без будь-якого осмисленого упорядкування позицій в результуючому списку [9], [10]. У разі великих колекцій людина не має практичної можливості переглянути всі результати, так як їх занадто багато. У зв'язку з цим, необхідно проводити упорядкування позицій результату. Для вирішення цього завдання пошукова машина оцінює ступінь відповідності запиту для кожного відповідного документа (здійснює ранжирування документів).

Розглянутий в даній частині метод ранжирування ґрунтується на вирішенні двох основних задач [10]:

– визначення важливості (ваги) кожного конкретного терміна в документі, ґрунтуючись на статистиці його зустрічальності в тексті;

– побудова векторної моделі документа для підрахунку рейтингу (score) кожного документа.

Нехай тепер запит являє собою текст у вільній формі, тобто без спеціальних операторів (таких як І, АБО, НЕ та ін.). Такий стиль формування запитів домінує серед всіх пошукових сервісів в Інтернет, в цьому випадку запит розглядається як безліч слів. Алгоритм ранжирування повинен в якомусь вигляді підсумувати для всього цього безлічі ступінь їх «участі» в документі. У контексті цього завдання потрібно призначити вагу кожному терміну в документі. Найбільш простий метод-використовувати частоту зустрічальності терміна ($tf_{t,d}$, де t,d – термін і документ відповідно)) [10] – [13]. У цьому випадку порядок слів перестає мати значення при підрахунку ступеня релевантності, і пропозиції типу «Паша швидше, ніж Саша» та «Саша швидше, ніж Паша» ідентичні в цій моделі. Проте, вважається, що два документи, схожі за множиною слів, схожі і за своїм змістом [6], [10] – [13].

1.1.4 Інвертована частота документа

Використання тільки частоти терміна в документі недостатньо, так як не всі терміни документа однаково важливі в підрахунку релевантності: наприклад, слово «трубопровід» дає більше інформації про тематику документа, ніж, наприклад, союз «або»; в колекції документів в області автомобільної індустрії, слово «авто» буде зустрічатися практично в кожному документі. Іншими словами, різні терміни надають на релевантність різний вплив, навіть якщо вони зустрічаються в документі однаково часто. Можна було б використовувати частоту терміна в колекції (cf) для визначення понижуючого коефіцієнта для тих термінів, які занадто поширені в колекції і мають малий вплив на релевантність. Однак загальноприйнятим рішенням є використання частоти документа (df_t) [10]. Сенс в тому, що для поділу

документів вигідніше використовувати статистичні величини рівня документа, а не рівня терміна.

Є певна причина, через яку df вигідніше, ніж cf. Наприклад, є необхідність щоб невелика частина документів містить 1-ий термін отримала більшу оцінку при запиті «1-ий термін», ніж багато документів, що містять «2-ий термін» при запиті «2-ий термін», при тому, що частота зустрічальності в колекції для цих термінів збігається. Щоб використовувати частоту документа, вводиться спеціальна величина-інвертована частота документа, яка визначається за допомогою формули (1.1) [11]:

$$idf_t = \log \frac{N}{df_t}, \tag{1.1}$$

де N – загальна кількість документів у колекції.

Відзначимо, що idf для терміна, який рідко зустрічається – високий, а idf для терміна, який часто зустрічається– низький. У табл. 1.3 представлена статистика значень idf в колекції Reuters[12] для декількох термінів.

Таблиця 1.3 – Статистика значень idf в колекції Reuters для декількох термінів

Термін	df_t	idf_t
Car	18165	1,65
Auto	6723	2,08
Insurance	19241	1,62
Best	25235	1,5

У розглянутій ваговій схемі, званої $tf-idf$, ваги термінам призначаються відповідно до формули (1.2):

$$tf - idf_{t.d} = tf_{t,d} \times idf_t \tag{1.2}$$

та відповідають наступним характеристикам [10] – [13]:

– найбільшу вагу мають терміни, що зустрічаються багато разів, але в невеликій кількості документів;

– меншу вагу мають терміни, які або менше зустрічаються в документі, або зустрічаються в більшій кількості документів;

– найменшу вагу мають терміни, які зустрічаються практично у всіх документах.

1.1.5 Векторна модель

Тепер ми можемо розглянути кожен документ як вектор, розмірність якого дорівнює кількості термінів у словнику, А кожен компонент обчислюється за формулою (1.2). Такий вектор документа будемо позначати $\vec{V}(d)$. Колекція документів може бути розглянута як безліч векторів в єдиному векторному просторі, в якому кожна вісь відповідає за один термін. Як вже було зазначено, таке подання втрачає порядок проходження термінів у документах [13].

Після того, як була побудована векторна модель, необхідно визначити спосіб обчислення міри близькості двох векторів, що належать векторному простору. Простою ідеєю є модуль різниці двох векторів, але цей метод не підходить з наступної причини: два документи з дуже схожим змістом можуть сильно відрізнятися в даній мірі тільки тому що один з них набагато довше іншого (відносні частоти зустрічальності термінів однакові, але їх абсолютні величини сильно розрізняються) [11]. Щоб компенсувати ефект довжини документа, стандартним рішенням є обчислення косинуса кута між векторами $\vec{V}(d_1)$ та $\vec{V}(d_2)$ (1.3):

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{\left|\vec{V}(d_1)\right\|\vec{V}(d_2)\right|}, \tag{1.3}$$

де чисельник являє собою скалярний добуток векторів, а знаменник - добуток модулів векторів. Якщо позначити $\vec{v}(d) = \vec{V}(d) / \left|\vec{V}(d)\right|$, тоді (1.3) перепишеться відповідно до формули (1.4):

$$sim(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2). \quad (1.4)$$

Визначимо, яке застосування може мати метод обчислення близькості двох документів. Можна, маючи документ *d* знайти всі схожі документи – така функціональність підтримується деякими пошуковими машинами. Така функція може називатися «прочитати ще» або «прочитати схожі». Але більш важливе застосування – запит теж можна розглядати як вектор, вважаючи його документом, тільки дуже коротким. Повертаючись до підрахунку релевантності між запитом і різними документами, ми можемо призначати кожному документу значення його рейтингу (score) за такою формулою $score(q,d) = \vec{v}(q) \cdot \vec{v}(d)$, де $\vec{v}(q) = \vec{V}(q) / \left|\vec{V}(q)\right|$ – вектор запиту [12]. Тоді результуюча формула (1.4) матиме вигляд (1.5):

$$score(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{\left|\vec{V}(q)\right|\left|\vec{V}(d)\right|}. \quad (1.5)$$

Відзначимо, що документ може мати високе значення score, навіть якщо він не містить всіх термінів із запиту.

1.1.6 Оцінка результатів роботи системи інформаційного пошуку

Маючи деяку модель ранжирування в інформаційно-пошуковій системі, необхідно вміти відповідати на питання, наскільки ефективно працює конкретна функція ранжирування.

Для оцінки ефективності роботи пошукової системи потрібні три сутності:

– колекція документів;

– колекція тестових запитів;

– дані про те, які документи в колекції є релевантними для кожного тестового запиту.

Стандартний підхід полягає в тому, що для пари документ-запит відомо, чи є він релевантним чи ні, і не більше того. Тестова колекція повинна мати розумний розмір, щоб усереднені величини ефективності мали сенс. За необхідний мінімум приймається кількість в 50 запитів [6], [12]. Для прикладу наведемо одну широко використовувану тестову колекцію Text Retrieval Conference (TREC). Вона розробляється в Національному інституті стандартів і технологій в США. Колекція має поділ за типами розв'язуваних завдань, при цьому всього в ній міститься 1.89 мільйонів документів (в основному, новинних статей) і 450 різних тестових запитів, а також оцінки релевантності кожної пари документ – запит.

Вже зазначалося, що в якості показників ефективності можуть використовуватися точність і повнота пошуку. Ці величини не залежать від порядку ранжирування результатів, а засновані тільки на множинах релевантних документів і результату, що видається системою. Крім цього, важко оцінювати зміни в ефективності системи, якщо, наприклад, збільшилася точність пошуку, але знизилася повнота.

Для оцінки результатів ранжируваного пошуку в TREC часто використовується єдина величина, звана MeanAveragePrecision (MAP). Сенс цієї величини в тому, що для кожного запиту підраховується середнє серед значень точності, отриманих після кожного нового повернутого системою документа. Це значення називається Середня точність (СТ). Для отримання MAP, СТ усереднюється за всіма тестовими запитами. Формула (1.6) представлена нижче.

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} precision(R_{jk}), \quad (1.6)$$

де $Q = \{q_1, q_2, ..., q_N\}$;

$q_j = \{d_1, d_2, ..., d_{mj}\}$;

R_{jk} – підмножина q_j від документа d_1 до документа d_k.

Зазвичай MAP буде використовуватися для оцінки результатів роботи системи. Трохи інший підхід до оцінки ефективності роботи системи: підрахунок обмежується першими *k* результатами пошуку. Такий підхід може бути корисний в разі, якщо користувач переглядає, наприклад, тільки першу сторінку результатів пошуку. У такому випадку релевантність результатів на першій сторінці набагато важливіша, ніж на наступних, і має сенс максимізувати.

1.2 Ранжирування на основі посилань

В даному підрозділі описуються методи виявлення статичних ознак ранжирування, використовувані крім векторної або імовірнісної моделі пошуку для поліпшення результатів пошуку [10] – [13].

Якщо відбувається пошук в Інтернет, то розширення методів ранжирування за рахунок обліку посилальної структури може дати істотне поліпшення результатів пошуку. Для початку введемо поняття веб-графа. Статична частина веба, що складається з документів типу HyperTextMarkupLanguage (HTML), і гіперпосилань між ними може бути представлена у вигляді спрямованого графа, в якому кожен вузол є веб-сторінкою, а кожне спрямоване ребро – гіперпосиланням. Сукупність таких вузлів і направлених ребер називається веб-графом [6] – [12].

Посилання в веб-графі не розподілені випадковим чином. Число ребер, що входять у вузол, розподілено швидше статечному закону, а не за законом

Пуассона, як було б, якби посилання були розставлені випадковим чином. Розглянемо два інтуїтивних припущення, на основі яких базується виклад методів аналізу посилань у веб-графі [11].

1. Текст посилання, що вказує на сторінку *B*, є хорошим описом сторінки *B*.

2. Гіперпосилання зі сторінки *A* на сторінку *B* являє собою визнання авторитетності сторінки *B* з боку творця сторінки *A*. Це не завжди так; наприклад, багато посилань зі сторінки на сторінку всередині одного сайту зобов'язані своєю появою загальному шаблону сайту. Наприклад, більшість корпоративних веб-сайтів мають на кожній сторінці посилання на повідомлення про авторське право. Зрозуміло, що це не є свідченням схвалення. Відповідно, алгоритми аналізу посилання враховують такі посилання з меншою вагою.

1.2.1 Метод PageRank

Розглянемо методи обчислення ваг і ранжирування, що випливають виключно зі структури посилань. Перший метод присвоює кожному вузлу веб-графа вагу в діапазоні від 0 до 1, відомий як PageRank [10] – [13]. Вага вузла залежить від структури посилань.

Розглянемо процес випадкового переміщення по мережі, що починається з будь-якої сторінки (вузла веб-графа), і підкоряється наступним правилам. У кожен момент часу користувач з сторінки a переходить на випадкову сторінку, на яку зі сторінки в веде гіперпосилання. Наприклад, якщо зі сторінки *A* гіперпосилання ведуть на сторінки *B, C* і *D*, то «мандрівник», перебуваючи у вузлі *A*, в наступний момент часу з рівною ймовірністю ($p = 1/3$) перейде на одну зі сторінок *B, C* і *D*.

Ясно, що мандрівник буде потрапляти в одні вузли частіше за інших; інтуїтивно зрозуміло, що на ці вузли є багато посилань, що знаходяться на інших часто відвідуваних вузлах. Ідея алгоритму PageRank полягає в тому, що

вузли, часто відвідувані в результаті випадкового блукання по мережі, мають більшу важливість, ніж рідко відвідувані вузли.

Якщо в вузлі a немає вихідний посилань, то відбувається телепортація в випадковий вузол мережі. Ця операція еквівалентна тому, що мандрівник набирає UniformResourceLocator (URL) сторінки в рядку браузера. Імовірність потрапити в кожен вузол дорівнює $1/N$, де N – загальне число вузлів в графі.

Операція телепортації використовується двояко:

– якщо вузол не має вихідних посилань, то «мандрівник» викликає операцію телепортації;

– якщо вузол має вихідні посилання, то телепортація відбувається з ймовірністю α, де α зазвичай дорівнює 0,1, а з ймовірністю $1-\alpha$ переходить за випадковою вихідною посиланням.

Застосовуючи теорію марковських ланцюгів, можна показати, що «мандрівник», наступний за описаним комбінованим алгоритмом, знаходиться в кожному вузлі v фіксовану частку часу $\pi(v)$, яка залежить від [10] – [13]:

– структури веб-графа;

– від значення α. Величина $\pi(v)$ називається вагою PageRank вузла v.

Для підрахунку PageRank можна використовувати степінний метод. Він імітує випадкове блукання: почавши з певного стану і виконавши велику кількість кроків *t*, ми відстежуємо частоти відвідування кожного стану. Після великої кількості кроків t ці частоти встановлюються, так що різниця між обчисленими частотами опускається нижче заданого порігового значення. Ці частоти оголошуються вагами PageRank.

Значення ваг PageRank не залежать від запиту користувача. Таким чином, вага PageRank є мірою статичної якості веб-сторінки, що не залежить від запитів користувачів. Зрозуміло, що результат рейтингу повинен залежати від запиту. Методи обліку різних рангів сторінки для побудови єдиного списку будуть розглянуті пізніше.

1.2.2 Метод Hyperlink-InducedTopicSearch

Розглянемо метод Hyperlink-InducedTopicSearch (HITS) [9] – [11], в якому за заданим запитом кожна веб-сторінка отримує два показники: показник портальності і показник авторитетності. У цьому методі обчислюється два ранжируваних списку сторінок, а не один. Метод ґрунтується на поділі сторінок на два основних типи: портали і авторитетні джерела. Такий поділ особливо доречно при широкому пошуку, коли запити мають вигляд «Я хочу знати про університет». Авторитетним джерелом є офіційний сайт університету. Порталами є сайти, що містять посилання на авторитетні джерела. Ці сторінки не містять в собі необхідну інформацію, але містять посилання на неї, зібрані зацікавленими людьми.

Показник портальности будемо позначати $h(v)$, а показник авторитетності – як $a(v)$. Гіперпосилання зі сторінки v на сторінку α будемо позначати $v \to \alpha$. На початку для усіх вузлів $v : h(v) = a(v) = 1$. Тоді ітеративний процес буде виглядати відповідно до формул (1.7) та (1.8):

$$h(v) \leftarrow \sum_{v \to y} a(y), \tag{1.7}$$

$$a(v) \leftarrow \sum_{y \to v} h(y). \tag{1.8}$$

Таким чином, показник портальності сторінки визначається як сума показників авторитетності сторінок, на які вона посилається. З іншого боку, якщо на сторінку посилаються хороші портали, то показник її авторитетності збільшується. Досліджуємо результат багаторазового ітеративного застосування зазначених вище формул.

Нехай A – квадратна матриця суміжності підмножини веб-графа, в якій кожній сторінці відповідає один рядок і один стовпець. Елемент A_{ij} дорівнює одиниці, якщо існує гіперпосилання зі сторінки i на сторінку j, та нулю – в

іншому випадку. Тоді формули (1.7) та (1.8) перепишуться наступним чином та буде виглядати відповідно до формул (1.9) та (1.10):

$$\vec{h} \leftarrow A\vec{a}, \tag{1.9}$$

$$\vec{a} \leftarrow A^T\vec{h}. \tag{1.10}$$

після взаємної підстановки вираховується за формулою (1.11) та (1.12)

$$\vec{h} \leftarrow AA^T\vec{h}, \tag{1.11}$$

$$\vec{a} \leftarrow A^TA\vec{h}. \tag{1.12}$$

Якщо замінити символ «$\leftarrow$» символами «=» і ввести невідоме власне значення, то отримаємо рівняння для власних векторівт(1.13) та (1.14).

$$\vec{h} = \frac{1}{\lambda_h} AA^T\vec{h}, \tag{1.13}$$

$$\vec{a} = \frac{1}{\lambda_a} A^TA\vec{h}, \tag{1.14}$$

де λ_h – власне значення матриці AA^T;

λ_a – власне значення матриці A^TA.

Звідси слідують важливі наслідки.

1. Результат ітеративного обчислення прагне до стаціонарного значення, визначеного структурою графа.

2. Для обчислення результату можна застосовувати будь-який метод обчислення власного вектора стохастичної матриці, не обов'язково обмежуватися ітеративним алгоритмом.

Метод HITS має наступний вигляд:

– вибираємо цільове безліч веб-сторінок, створюємо веб-граф, індукований гіперпосиланнями, і обчислюємо матриці AA^T і A^TA;

– обчислюємо головні власні вектора і створюємо вектор показників портальності і вектор показників авторитетності;

– повертаємо веб-сторінки з високим показником портальності і високим показником авторитетності.

Також опишемо алгоритм вибору підмножини веб-сторінок, відповідних певній темі.

1. За заданим запитом за допомогою текстового індексу знаходимо сторінки, що містять терміни із запиту. Це безліч назвемо кореневим.

2. Побудуємо базову множину сторінок, що включає в себе кореневу множину, а також будь-яку веб- сторінку з кореневої множини, яка або сама посилається на сторінку з кореневої множини, або на неї посилається якась сторінка з кореневої множини.

Базова множина буде використовуватися для створення списків авторитетності та списків портальності.

1.3 Поєднання різних ознак для ранжирування

Ми розглянули векторну модель документів і ранжирування за косинусною мірою близькості запиту і документа [12], [13]. Цей захід визначається на основі пари запит, документ. У той же час існує безліч ознак документа, які можуть виявитися корисними при ранжируванні документа, і їх облік збільшить показники ефективності системи: PageRank, вік документа, довжина документа, довжина URL (якщо документ є веб-сторінкою) та інші. Для побудови функції ранжирування, відповідно до якої користувачі системи будуть бачити результати пошуку, необхідно в якомусь вигляді комбінувати кілька ознак. Різні способи, існуючі для вирішення цього завдання, розглянуті нижче.

1.3.1 Ручна настройка параметрів

Для початку розглянемо методи суміщення різних ознак документа ранжирування на прикладі роботи [8]. У цій роботі досліджуються можливості поліпшення результатів ранжирування з використанням декількох статичних ознак веб-сторінок. В якості базового ранжирування використовується Okapi BM25, дослідження проводиться в рамках тестової колекції документів TREC. В якості статичних параметрів сторінок використовуються наступні величини, що залежать від конкретного документа, але не залежать від запиту: PageRank, Indegree, URL Length, ClickDistance. PageRank -міра документа, що базується на посилальної структурі колекції; Indegree - кількість посилань, що ведуть на сторінку; URL Length – міра довжини базової адреси сторінки, яка «воліє» короткі шляхи або поділяє їх на кілька типів (кореневі, звичайні, файли тощо) і приписує кожному типу своє значення; ClickDistance - число, що дорівнює мінімальному числі переходів, яке потрібно щоб потрапити на цю сторінку з деякою кореневої сторінки (у випадку недосяжності приймається деяке усереднене значення).

Результати показали, що всі перераховані вище характеристики збільшують ефективність пошуку в рамках тестів TREC. Крім того, при використанні PageRank, додавання додаткових характеристик Indegree і ClickDistance не покращує результати, на відміну від URL Length. Користь від перерахованих статичних величин при поєднанні їх з ранжируванням *BM25* різна, послідовність в порядку убування зростання ефективності наступна: PageRank, Indegree, URLLength, ClickDistance.

Позначимо через s величину статичного параметра, через M – вміст документа, через *BM25* – величину близькості, підраховану за допомогою однойменної вагової схеми. R – випадкова величина, індикатор релевантності документа. У [8] показано, що в рамках імовірнісної моделі в якості підсумкової формули ранжирування може бути використана сума BM25 і

деякої величини моделює S по відношенню до M і R. у роботі порівнювалися наступні комбіновані функції (1.15) та (1.16):

$$Score(q,d) = BM25(q,d) + wS, \tag{1.15}$$

$$Score(q,d) = BM25(q,d) + w\frac{S^a}{k^a + S^a}, \tag{1.16}$$

де w, k та a – параметри, що налаштовуються.

У (1.5) передбачається незалежність S і M. у разі залежності S і M налаштування параметрів у (1.16) дозволяє домогтися кращої ефективності. Крім того, у випадку з PageRank більш високі результати були досягнуті з використанням логарифма цієї величини, у зв'язку з тим, що значення PageRank розподілено за статечним законом.

1.3.2 Повторне ранжирування

Повторне ранжирування – метод обліку статичної якості документа, розглянутий в [5], [10]. Автори використовували статичне значення «не специфічності» документа, визначаючи частину документів як занадто загальні, що містять недостатньо інформативних термінів. Поріг неспіцифічності визначався вручну в результаті тестування на різних значеннях. У роботі застосовується два різних методи обліку обраної характеристики в підсумковому ранжируванні:

– видаляти всі неспецифічні документи з колекції;

– збільшувати позицію для неспецифічних документів.

1.4 Ранжирування на основі машинного навчання

Різні параметри документів (косинусна близькість, PageRank і т.д.) можна розглядати як ознаки в задачі навчання, і не налаштовувати функції

ранжирування вручну, а оптимізувати параметри класифікатора на навчальній безлічі [14] – [17]. Якщо сформулювати задачу за допомогою пар документ-запит, яким призначено оцінку релевантний-нерелевантний, то її можна інтерпретувати як задачу класифікації текстів. Такий підхід не завжди є найкращим, і будуть розглянуті альтернативні варіанти.

Розглянемо простий приклад ранжирування на основі машинного навчання [15]. Будемо вважати, що функція ранжирування являє собою лінійну комбінацію двох факторів:

– косинусної міри подібності між запитом і документом;

– мінімальної ширини вікна w, всередині якого лежать терміни запиту.

Обидва фактори є хорошими індикаторами релевантності. Будемо працювати з навчальними прикладами, кожен з яких являє собою пару, що складається із запиту і документа, а також оцінку релевантності цього документа даному запиту: релевантний і не релевантний [15], [16]. Для кожного такого прикладу можна обчислити косинусну міру подібності, а також ширину вікна w.

Тут дві ознаки (a та w) є дійсними числами. Позначимо оцінку релевантний одиницею, а оцінку нерелевантний – нулем. Тепер завдання можна сформулювати так: знайти функцію ранжирування, що комбінує значення ознак і приймає значення, близькі до нуля або одиниці. Ця функція повинна бути якнайкраще узгоджена з сукупністю навчальних прикладів. Не обмежуючи спільності, будемо вважати, що шуканий лінійний Класифікатор має наступний вигляд (1.17) [17].

$$Score(q,d) = Score(a,w) = aa + bw + c, \qquad (1.17)$$

де a, b, c– обчислюються під час навчання.

Це завдання можна сформулювати як завдання мінімізації помилки і вирішувати одним з існуючих методів машинного навчання (наївна байєсівська Класифікація, метод k найближчих сусідів, метод опорних векторів і т.д.). Якщо

колекція навчальних прикладів досить велика, то можна повністю уникнути ручного налаштування функції. Зрозуміло, вузьким місцем розглянутого підходу є репрезентативність безлічі навчальних прикладів, релевантність яких повинні оцінювати експерти [16], [17].

У даній постановці задача відноситься до класу задач класифікації, в якій передбачається категоріальна змінна – релевантний документ чи ні. Щоб вирішити задачу ранжирування, можна використовувати навчений бінарний Класифікатор і сортувати документи за ймовірністю їх релевантності, однак такий підхід до ранжирування не завжди обгрунтований. Статистики зазвичай розрізняють задачі класифікації та регресійні задачі (в яких передбачається дійсне число). Між цими полюсами розташована спеціальна область порядкової регресії, в якій передбачаються ранги (ранжирування). Машинне навчання для пошуку за запитом слід розглядати, перш за все, як завдання порядкової регресії, мета якої –ранжируватисукупність документів по відношенню до запиту на основі навчальних прикладів того ж роду.

1.5 Постановка завдання

Аналіз методу ранжирування є завданням інтелектуального аналізу даних [14] – [17], для вирішення якого необхідно виявити фактори, що роблять істотний вплив на результат ранжирування, і залежність позицій документів від значень цих факторів. Досліджувані дані подаються у векторному вигляді, де компонентам векторів відповідають фактори, що характеризують документи і запити. Прикладами таких факторів є довжина документа, цитованість документа, кількість слів у запиті та інші.

Метод ранжирування пошукової системи будує функцію релевантності, яка порівнює пари векторів (q, d), що описують документ і текстовий запит відповідно, числову оцінку релевантності rel (1.18) [15]:

$$f(q, d) \to rel\,. \tag{1.18}$$

Рішення задачі ідентифікації методу ранжирування зводиться до побудови моделі, яка за заданими векторами (q, d) визначає ступінь релевантності документа d запитом q. Ступінь релевантності визначається як ранг, притаманний документу d при ранжируванніза запитом q. Зазвичай поділяють кілька ступенів релевантності (наприклад, високорелевантний, середньорелевантний і низькорелевантний) і кожного ступеня зіставляється кілька рангів.

Тож головним завданням роботи є вивчення нейромережевої моделі методу ранжирування текстових документів у базах даних, побудованої на основі методів машинного навчання.

1.6 Висновки за розділом 1

Розглянуто проблемну область ІП та ранжирування. В результаті проведеного аналізу було з’ясовано, що одним з найбільш актуальних та безпечних базисів для ранжирування та підвищення релевантності результатів пошуку є використання методів машинного навчання. Використання таких підходів дозволить відповісти на питання, якими властивостями повинен володіти документ, релевантний заданому запиту. Виходячи з такої постановки задачі ідентифікації, на вхід системі подається запит, а на виході система надає значення значущих факторів.

2 МАТЕРІАЛИ ТА МЕТОДИ

В першому розділі було проаналізовано сучасне становище у питанні ІП, ранжирування та релевантності. Попередній аналіз довів, що методів машинного навчання є перспективною основою для підвищення точності методів ранжирування.

Тому актуальним є дослідження та реалізація методів ранжирування на базі машинного навчання. Проте для цього слід визначити, які саме методи та засади машинного навчання будуть використовуватися.

Другий розділ присвячено детальному ознайомленню та дослідженню проектуванню методу ранжирування на основі машинного навчання.

2.1 Підготовка даних

Виходячи з попередньої постановки задачі ідентифікації, на вхід проектованому методу подається запит, а на виході метод надає відповіді із високою релевантністю. Така релевантність буде залежати від певних факторів.

Фактори, що описують документ і текстовий запит, можуть бути як числовими (такі, як обсяг тексту документа), так і номінальними (наприклад, тематика документа). Метою попередньої обробки даних для інтелектуального аналізу є приведення даних до однорідного вигляду, яке включає в себе три етапи [14], [16]:

– апріорне виключення малозначущих компонентів векторів q та d. На даному етапі перевіряється наявність кореляційних зв'язків між характеристиками як вхідних, так і вихідних даних окремо з подальшим виключенням малозначущих факторів;

– представлення входів і виходів моделі в числовому вигляді для номінальних факторів за допомогою двійкового кодування, при якому кожному значенню фактора зіставляється вектор, компоненти якого відповідають розрядам двійкового представлення номера значення;

– нормування даних за допомогою біполярної сигмоїдальної функції активації (2.1)[16]:

$$f(x) = \tanh(\beta x), \tag{2.1}$$

де β – заданий коефіцієнт;

x – значення фактору.

2.2 Використання нейромережевих моделей

Під штучними нейронними мережами (НМ) передбачають обчислювальні структури, що складаються з великої кількості однотипних елементів, кожен з яких виконує щодо прості функції [17]. Процеси в штучних НМ іноді асоціюють з процесами, що відбуваються в нервовій системі живих організмів. Навчити нейронну мережу – значить, повідомити їй, чого ми від неї домагаємося. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу. Вони роблять це без жодного апріорного знання. Натомість, вони розвивають свій власний набір доречних характеристик з навчального матеріалу, який вони оброблюють.

НМ ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами (аналогічно до біологічних нейронів у головному мозку тварин). Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого [13], [14]. Штучний нейрон, що отримує сигнал, може обробляти його, й потім сигналізувати штучним нейронам, приєднаним до нього. На рисунку 2.1 кожним круговим вузлом представлено штучний нейрон, а стрілкою – з'єднання виходу одного штучного нейрону зі входом іншого.

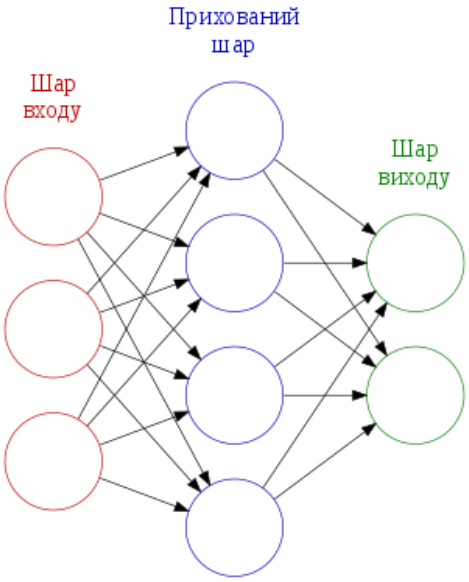

Рисунок 2.1 – Штучна нейронна мережа [14]

В поширених реалізаціях НМ сигнал на з'єднанні між штучними нейронами є дійсним числом, а вихід кожного штучного нейрону обчислюється нелінійною функцією суми його входів. Штучні нейрони та з'єднання зазвичай мають вагу, яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається лише якщо сукупний сигнал перетинає цей поріг. Штучні нейрони зазвичай організовано в шари. Різні шари можуть виконувати різні види перетворень своїх входів. Сигнали проходять від першого (входового) до останнього (виходового) шару, можливо, після проходження шарами декілька разів.

У зв'язку з успіхом застосування згорткових нейронних мереж до задачі класифікації зображень з'явилося безліч спроб використовувати нейронні мережі в інших завданнях. Останнім часом їх стали активно використовувати для завдання класифікації текстів.

2.3 Нейронні мережі Кохонена

НМ Кохонена – клас нейронних мереж, основним елементом яких є шар Кохонена. Шар Кохонена складається з адаптивних лінійних суматорів («лінійних формальних нейронів»). Як правило, вихідні сигнали шару Кохонена обробляються за правилом «Переможець отримує все»: найбільший сигнал перетворюється в одиничний, інші звертаються в нуль [18].

За способами налаштування вхідних ваг суматорів і за розв'язуваними завданнями розрізняють багато різновидів мереж Кохонена [19]. Найбільш відомі з них:

– мережі векторного квантування сигналів [20], тісно пов'язані з найпростішим базовим алгоритмом кластерного аналізу (метод динамічних ядер або k-середніх);

– самоорганізуються карти Кохонена;

– мережі векторного квантування, що навчаються з учителем.

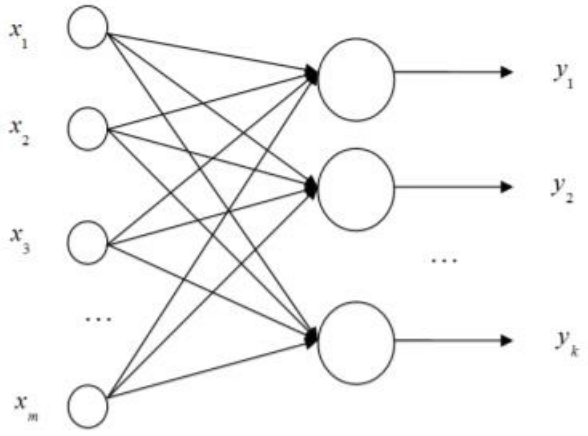

Рисунок 2.2 – Архітектура нейронної мережі Кохонена [19]

Як видно з рисунку 2.2 вектор вхідних сигналів $x = (x_1, x_2, ..., x_m)$ надходить на входи всіх нейронів вихідного шару. Нейрони вихідного шару є лінійними адаптивними суматорами і навчаються за правилом «Переможець отримує все» [21]. Кожному нейрону вихідного шару відповідає деякий кластер простору E.

Навчання мережі Кохонена полягає в підборі ваг w_{ij}, $i=1..m$, $j=1..k$ нейронів вихідного шару. Ваги нейрона, що відповідає за кластер, є компонентами вектора w^j, що є центром *j*-го кластера. Нейрон $j \in \{1,...,k\}$, який переміг при навчанні на деякому вхідному векторі, змінює свої ваги згідно з правилом (2.2):

$$w^j(t+1) = w^j(t) + \eta\left(x - w^j(t)\right), \tag{2.2}$$

де w^j – вектор вагових коефіцієнтів нейрону-переможця;

$\eta \in (0,1)$ - коефіцієнт навчання;

x – вхідний вектор.

Одночасно з нейроном-переможцем з меншою інтенсивністю свої ваги змінюють і інші нейрони $j \in \{1,...,k\}$, $i \neq j$. Чим далі нейрон знаходиться від переможця, тим менше змінюються його ваги (2.3):

$$w^j(t+1) = w^j(t) + \eta G(i,j)\left(x - w^j(t)\right), \tag{2.3}$$

де $G(i,j)$ – функція відстані між нейроном-переможцем *j* та даним нейроном *i*, наприклад: $G(i,j) = \exp\left(-\frac{d^2(i,j)}{2\lambda^2}\right)$. Тут $d(i,j)$ – може бути мірою відстані між ваговими векторами нейронів (наприклад, $d(i,j) = \left\| w^i - w^j \right\|$), а λ – заданий параметр.

Вихідний сигнал нейрона *i* тоді буде дорівнювати (2.4):

$$y = G(i,j), i = 1,...,k. \tag{2.4}$$

2.4 Метод кластеризації k-means

Метод k-means – найбільш популярний метод кластеризації. Був винайдений в 1950-х роках математиком Гуго Штейнгаузом і майже одночасно Стюартом Ллойдом. Особливої популярності набув після роботи Маккуїна[22].

Дія алгоритму така, що він прагне мінімізувати сумарне квадратичне відхилення точок кластерів від центрів цих кластерів (2.5) [22], [23]:

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2 , \tag{2.5}$$

де k – число кластерів;

S_i – отримані кластери $i = 1,2,...,k$;

μ_i – центри усіх векторів x із кластеру S_i.

За аналогією з методом головних компонент центри кластерів називаються також головними точками, а сам метод називається методом головних точок і включається в загальну теорію головних об'єктів, що забезпечують найкращу апроксимацію даних [24].

Алгоритм являє собою версію EM-алгоритму, застосовуваного також для поділу суміші гауссіан. Він розбиває безліч елементів векторного простору на заздалегідь відоме число кластерів k [25].

Основна ідея полягає в тому, що на кожній ітерації переобчислюється центр мас для кожного кластера, отриманого на попередньому кроці, потім вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився ближче за обраною метрикою.

Алгоритм завершується, коли на якійсь ітерації не відбувається зміни внутрішньокластерної відстані. Це відбувається за кінцеве число ітерацій, так як кількість можливих розбиття кінцевої множини звичайно, а на кожному кроці сумарне квадратичне відхилення V зменшується, тому зациклення неможливо.

В нашому випадку, нехай задано безліч векторів $\{x_1,...,x_m\} \subseteq E$, E – простір векторів. Потрібно розбити цю множину на k кластерів. Для цього знаходиться безліч кластерних центрів $w_j \subseteq E$, $j = 1,...,k$, що мінімізують функціонал (2.6):

$$D = \sum_{i,j} d(x_i, w_j), \tag{2.6}$$

де $d(x_i, w_j)$ – міра відстані між векторами x_i та w_j в просторі E.

2.5 Використання мереж Кохонена та кластеризації для методу ранжирування

Самоорганізована мережа Кохонена виділяє значущі фактори у вхідних векторах. Ці фактори утворюють вхідний шар нової мережі (кількість входів нової мережі збігається з кількістю кластерів, виділених мережею Кохонена). Кількість вихідних нейронів збігається з кількістю значущих характеристик, де кожна характеристика значима хоча б для одного кластера. Кількість прихованих нейронів визначається експериментально.

Гібридна мережа обробляє всі кластери [18], але при цьому потрібна додаткова модифікація задачника, необхідність якої викликана наявністю великої кількості факторів, несуттєвих для деяких кластерів. Модифікація задачника полягає в обнуленні тих компонент вихідних векторів, які є малозначущими для кластера, відповідного вхідному вектору.

Для кожного кластера вхідних векторів, виділеного мережею Кохонена, пропонується використовувати окремий персептрон з одним прихованим шаром. Задачник мережі являє собою безліч пар «вхід-вихід» вихідного задачника, де вхід належить досліджуваному кластеру. Кількість вхідних нейронів персептрона збігається з кількістю нейронів мережі Кохонена, кількість прихованих нейронів визначається експериментально, кількість

вихідних нейронів визначається числом значущих факторів досліджуваного кластера.

Дана модель дозволяє вирішити проблему нульових значень в задачнику для гібридної нейронної мережі, проте більш громіздка, оскільки для кожного кластера створюється окрема апроксимуюча нейронна мережа. Така модель виправдана при невеликій кількості значущих характеристик для кластерів або невеликій кількості кластерів.

Алгоритм навчання нейромережевих моделей має ряд особливостей:

– якість результату кожного етапу (кластерний аналіз, факторний аналіз, регресійний аналіз) надає ключовий вплив на наступний етап;

– якість навчання не погіршується, якщо число вихідних нейронів мережі Кохонена перевищить кількість кластерів вхідних векторів. Це пояснюється тим, що розбиття безлічі схожих векторів на більш дрібні кластери не впливає негативно на проведення етапу факторного аналізу;

– якщо в кластер потрапляє велика частка невірно розпізнаних векторів, при використанні Моделі на основі гібридної нейронної мережі значення помилки в проблемному кластері знижається, що пов'язано з малою кількістю значущих характеристик в цьому кластері.

2.6 Висновки за розділом 2

У другому розділі проаналізовано метод ранжирування на основі машинного навчання.

У якості методів машинного навчання обрано НМ Кохонена та метод кластеризації k-means. Таким чином запропонований метод використовує значущі фактори для ранжирування використовуючи кластеризацію даних на основі використання мережі Кохонена. Це дозволяє в подальшому спростити навчання моделізавдяки виключенню малозначимих для ранжирування характеристик.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Опис основних вимог до програми

Об'єктом розробки є програмний продукт (ПП) для реалізації методу ранжирування при пошуку у інформації. Мета роботи – розробити систему, що в повній мірі дозволить імплементувати та дослідити роботу запропонованого методу.

Необхідно реалізувати 2-основні модуля для системи авторизації. Перший модуль представляє клієнтську частину, де від користувача приймаються вхідні дані запиту. Другий модуль представляє сервер для обробки даних від клієнта та власне ранжирування результатів запиту. Такий сервер власно можна умовно розділити також на 2 модулі: певний контролер, що опрацьовує клієнтські дані та власне модуль ранжирування.

Тож серед функціональних вимог можна виділити:

– отримання та передача пошукового запиту;
– опрацювання пошукового запиту клієнта;
– ранжирування відповідей на запит;
– виведення результатів;
– надання та забезпечення графічного інтерфейсу користувача (GUI).

Для забезпечення надійності програми, вона має базуватися на комп’ютері. На протязі усього сеансу роботи додаток не повинен збоїти або зависати. Після оновлення інформаційної бази або функціоналу, застосунок треба перезапустити.

Більш того до інформаційної бази, яку використовує застосунок, висовуються наступні умови:

– вносити зміни до бази може тільки адміністратор;
– оновлення бази слід виконувати дотримуючись правил зберігання.

3.2 Проектування структури програмної системи

При проектуванні структури майбутньої програми було виділено 4 основних програмних модуля, які представлено на рисунку 3.1.

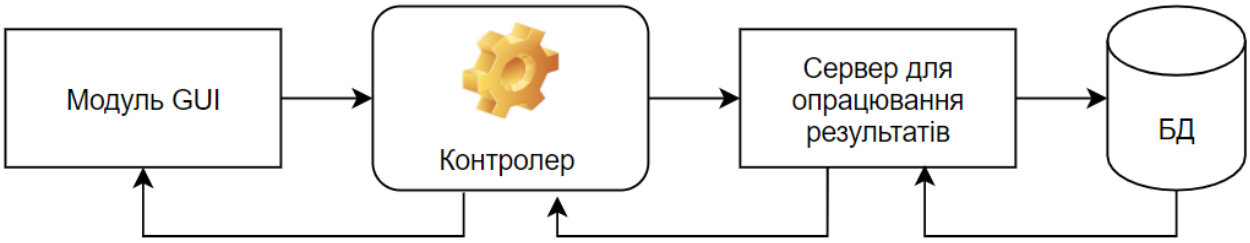

Рисунок 3.1 – Основні програмні модулі

Розробку представлених модулів можна аргументувати тим, що саме розділення основного функціоналу програмного комплексу за таким принципом гарантує формування функціонально завершених блоків програми.

Так модуль GUI отримує клієнтські дані та власне запит. Контролер слугує для передачі користувацьких даних на сервер для обчислень. Власне сервер підтягує данні із бази даних (БД) та виконує ранжирування відповіді на запит.

3.3 Вибір мови програмування

Мова програмування Python – це потужний інструмент для створення програм найрізноманітнішого призначення, доступний навіть для новачків [26]. З його допомогою можна вирішувати завдання різних типів.

Мова Python володіє деякими примітними особливостями, які обумовлюють її широке поширення. Тому слід проаналізувати її переваги та недоліки.

Python – інтерпретована мова програмування. З одного боку, це дозволяє значно спростити налагодження програм, з іншого – обумовлює порівняно низьку швидкість виконання [26].

Динамічна типізація. У Python не треба заздалегідь оголошувати тип змінної, що дуже зручно при розробці.

Хороша підтримка модульності. Ви можете легко написати свій модуль і використовувати його в інших програмах.

Вбудована підтримка Unicode в рядках. У Python необов'язково писати все англійською мовою, в програмах цілком може використовуватися ваша рідна мова.

Підтримка об'єктно-орієнтованого програмування. При цьому його реалізація в Python є однією з найбільш зрозумілих [27].

Автоматичне збирання сміття, відсутність витоків пам'яті.

Інтеграція з C / C++, якщо можливостей Python недостатньо.

Зрозумілий і лаконічний синтаксис, що сприяє ясному відображенню коду. Зручна система функцій дозволяє при грамотному підході створювати код, в якому буде легко розібратися іншій людині в разі необхідності. Також ви зможете навчитися читати програми та модулі, написані іншими людьми.

Величезна кількість модулів, як входять в стандартну поставку Python 3, так і сторонніх. У деяких випадках для написання програми достатньо лише знайти відповідні модулі і правильно їх скомбінувати. Таким чином, ви можете думати про складання програми на більш високому рівні, працюючи з уже готовими елементами, що виконують різні дії [26], [27].

Кросплатформність. Програма, написана на Python, буде функціонувати абсолютно однаково незалежно від того, в якій операційній системі вона запущена. Відмінності виникають лише в рідкісних випадках, і їх легко заздалегідь передбачити завдяки наявності докладної документації.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована

мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ [27].

Інтерпретатор мови Python і багата Стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

У таблиці 3.1 наведено порівняння зв’язки Python з іншими мовами програмування.

Таблиця 3.1 – Порівняння мов програмування

Критерії порівняння	Мови програмування			
	Python	Julia	Java +Deeplearning4j	Scilab
1	2	3	4	5
Наявність та актуальність оновлення матеріалів	10	5	7	9
Підтримка парадигми модульного програмування	8	5	8	7

Продовження таблиці 3.1

1	2	3	4	5
Продуктивність роботи	8	7	9	7
Паралельна обробка запитів та реалізація	8	5	7	7
Простота розроблення графічного інтерфейсу	7	3	7	5

Як видно з таблиці Python добре себе зарекомендувала. Цю мову програмування можна досить доречно використовувати для низки задач пов’язаних із машинним навчанням, наприклад, задач класифікації, оцінювання, моделювання та розпізнавання.

3.4 Вибір середовища розробки

Anaconda – це вільно та відкрито розповсюджуваний дистрибутив різних програмних продуктів, зокрема, мов програмування Python та R [28]. Платформа спеціалізується на «наукових обчисленнях»: наука про дані, застосуванні методів машинного навчання, широкомасштабна обробка даних, передбачувальна аналітика тощо. Використання платформи має на меті спрощення управління пакетами та їх розгортання. Версіями пакунків керує система управління пакетами Conda. Дистрибутив Anaconda використовується понад 15 мільйонами користувачів і містить більше 1500 популярних пакетів наукових даних, придатних для Windows, Linux та MacOS, наприклад, NumPy, SciPy та Ggplot2.

Дистрибутив Анаконда має широкі можливості під'єднання модулів (більше 1500), зокрема, власним Conda та менеджером віртуального середовища. Графічний інтерфейс, Anaconda Navigator слугує графічною альтернативою інтерфейсові командного рядка (CLI) [28].

Велика різниця між Conda та менеджером пакетів Pip полягає в тому, як управляти залежностями під'єднуваних пакунків, що є суттєвим викликом при роботі з Data Science у Python та головною причиною появи Conda.

Коли Pip встановлює деякий потрібний клієнтові пакет, то він автоматично встановлює весь перелік залежних від нього пакетів Python, не перевіряючи при цьому, чи вони будуть конфліктувати з раніше встановленими пакунками. Через це користувач із коректно встановленим, наприклад, Google Tensorflow, може виявити, що той різко перестає працювати: Pip при інсталяції нового пакета встановить інакшу версію NumPy (якого потребують одночасно і встановлюваний пакунок і вже наявний Tensorflow), наприклад 3.6, тоді як Tensorflow коректно працюватиме лише з 3.5. У деяких випадках може спочатку здаватися, що новий пакет працює як слід, але насправді він даватиме відмінні від правильних результати, що буде видно лише у деяких подробицях.

На відміну від цього, Conda структурно аналізує все поточне програмне середовище і вже після цього встановлює новий пакет, враховуючи всі обмеження сумісностей, версій різних пакунків, вірніше пропонує поєднуваний набір пакетів. У деяких випадках Conda попередить користувача про неможливість одночасного використання деяких пакетів [28].

Пакети з відкритим кодом можуть бути встановлені індивідуально як зі сховища Anaconda, так і з Anaconda Cloud чи з вашого власного сховища чи дзеркала, використовуючи команду conda install . Anaconda Inc компілює та створює всі пакунки у самому сховищі Anaconda та надає бінарні файли для Windows 32/64 біт, Linux 64 біт та MacOS 64-біт. Все, що доступне на PyPI, може бути встановлено в середовищі Conda за допомогою Pip, і Conda буде відслідковувати, що саме було встановлено самим пакунком і що саме встановлено за допомогою Pip.

Окремі, власні пакети можна створити за допомогою conda build, ними можна поділитись з іншими, завантаживши їх у будь-яке сховище, як-от: Anaconda Cloud, PyPI.

Установка Anaconda2 за замовчуванням тягне за собою Python 2.7, а Anaconda3 включає Python 3.7. Одначе за допомогою Conda завжди можна створити середовища, задані по-новому, щоб містити будь-яку версію Python.

У таблиці 3.2 наведено порівняння найбільш популярних середовищ розробки (IDE)для Python.

Таблиця 3.2 – Порівняння IDE

Критерії порівняння	IDE			
	Anaconda	PyCharm	Atom	PhpStorm
Наявність безкоштовної версії (open-sourse)	+	+	+	–
Підтримка бібліотек та налаштувань	10	8	6	10
Кросс-платформність	10	10	10	–
Швидкість роботи	10	10	10	8
Підтримка та оновлення	10	6	6	10

Як видно з таблиці зв’язка Anaconda найбільш оптимальний варіант IDE для подальшої розробки. Додатково слід зазначити, що саме Anaconda під лаштована під наукові обчислення, а отже легко інтегрує оновлені версії бібліотек необхідних для подальшої роботи.

3.5 Вимоги до технічних засобів

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

– наявність IBM-сумісної персональної електронної обчислювальної машини (ПЕОМ), на якій буде працювати додаток;

– процесор із тактовою частотою 2,0 GHz та більше;

– оперативна пам'ять 2,0 ГБ та більше;

– вільний простір на жорсткому диску не менше, ніж 2 ГБ (для збереження бази відповідей на запити).

Вимоги до програмної частини необхідно встановити дистрибутив Python.

3.6 Висновки за розділом 3

Під час роботи над розділом було спроектовано архітектуру майбутнього ПП. Були спроектовані модулі майбутньої програми. Було проаналізовано найбільш популярні мови програмування та після ретельного порівняння перевага була віддана Python.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Модульна структура проекту має передумову: модульна структура ПП полегшує її розуміння і дозволяє здійснювати модифікацію проекту за потреби.

Для повноцінної роботи із системою були розроблені модулі, які за функціоналом повторюють раніше спроектовані модулі ПП, проте мають більшу роздробленість за функціоналом. Серед розроблених модулів можна виділити:

– модуль GUI – з точки зору паттерну Model-View-Controllerце і буде представлення, тобто певна сторінка, що дозволяє отримати дані користувача;

– контролер (Controller) розбиває запит на терміни, формує пошукову команду – модуль, що містить список термінів і параметри пошуку;

– модуль опрацювання результатів – ExtendedSearcher отримує від сховища (БД) дані, що задовольняють запиту. ExtendedSearcher відправляє ранжируваний список повідомлень;

– модуль сховища (БД).

Розглянемо детальніше роботу розроблених модулів на рисунку 4.1.

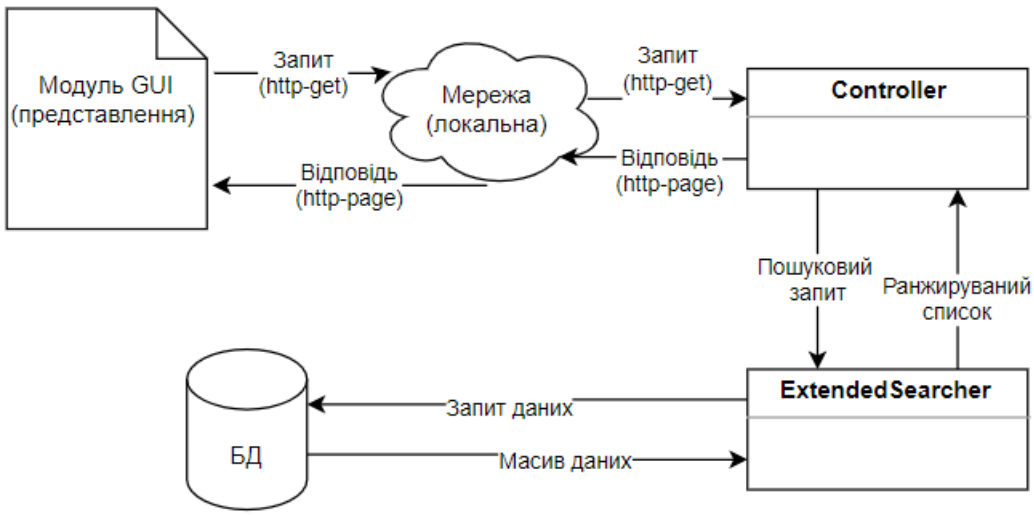

Рисунок 4.1 – Робота розроблених модулів

4.1 Модуль Controller

Controller розбиває запит на терміни, формує пошукову команду. Іншими словами, це модуль, що містить список термінів і параметри пошуку. На діаграмі класів модулю, що наведено на рисунку 4.2, наведено основний клас модулю.

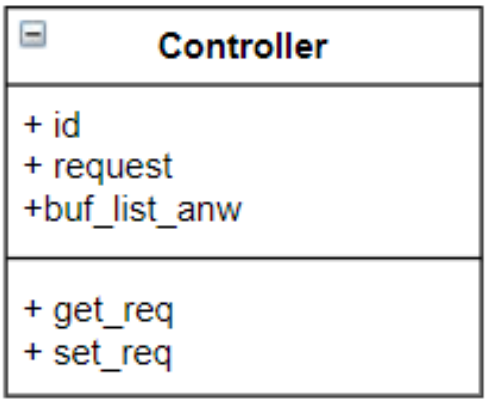

Рисунок 4.2 – Клас Controller

Модуль містить основний робочий клас. Робота зводиться до опрацювання запиту від користувача, що отримується через мережу. Також, клас надає користувачу відповідь у вигляді відсортованого списку результатів.

4.2 Модуль ExtendedSearcher

Модуль ExtendedSearcher отримує від БД, що задовольняють запиту. ExtendedSearcher відправляє ранжируваний список повідомлень. На діаграмі класів модулю, що наведено на рисунку 4.3, наведено основні класи модулю.

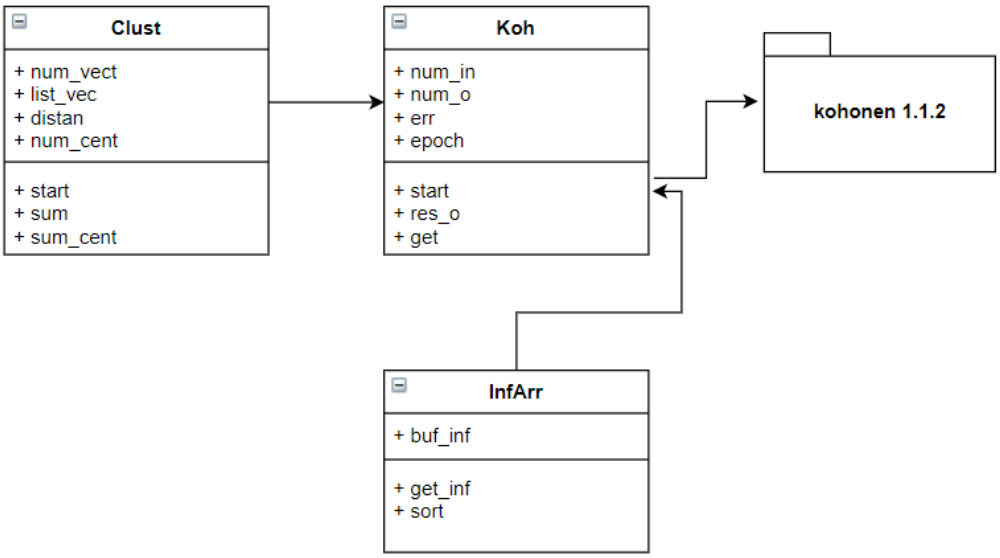

Рисунок 4.3 – Основні класи модулю

З рисунку видно, що клас Koh, який відповідає за тренування мережі Кохонена підтягує бібліотеку «kohonenv.1.1.2» [29]. Ця бібліотека відповідає за навчання мережі на основі даних.

Клас Clust відповідає за попередню кластеризіцію. Використовуючи множину векторів з простору векторів розділяє на k кластерів. Для цього знаходиться безліч кластерних центрів, що мінімізують функцію міру відстані між векторами.

Клас InfArr відовідає за завантаження масиву несортованої (без ранжирування) інформації із бази даних.

4.2.1 Бібліотека kohonenv.1.1.2

Цей модуль містить наступні реалізації карт Кохонена [29]:

– карта. Стандартна прямокутна N-мірна карта Кохонена;

– газ. Векторний квантувальник, який не має фіксованої топології. Нейрони в Газі сортуються для оновлення на основі їх відстані від сигналу,

причому порядок сортування визначає топологію для кожного представлення сигналу;

– GrowingGas. Квантувальник на основі газу, який може динамічно додавати нейрони для пояснення областей з високою помилкою вхідного простору;

– фільтр. Оболонка над базовим екземпляром карти, яка підтримує явну оцінку ймовірності кожного нейрона.

Вони тестуються за допомогою kohonen_test.py файл у цьому вихідному дистрибутиві.

Існує також колекція метрик відстані:

– косине_метричний. Викликається об'єкт, який обчислює косинусну відстань між сигналом і кожним нейроном в карті Кохонена;

– евклідова метрика. Об'єкт, що викликається, обчислює Евклідову відстань між сигналом і кожним нейроном в карті Кохонена;

– манхеттен_метричний. Викликається об'єкт, який обчислює манхеттенську відстань між сигналом і кожним нейроном на карті Кохонена.

4.3 Висновки за розділом 4

Розроблено ПП для реалізації методу ранжирування на основі машинного навчання з використанням кластеризації та мереж Кохонена. Обрано механізм реалізації відповідних методів та алгоритмів, а також елементи керування, з якими користувачу буде зручно використовувати розроблений ПП.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Цей розділ буде присвячено опису розробленого програмного комплексу. Буде означено сферу застосування, та наведено алгоритм користування. Також у цьому розділі буде проведено тестування та оцінка отриманих результатів.

5.1 Опис застосування програми

В якості вихідного алгоритму ранжирування розглянуто класичний алгоритм OkapiBM25 [14] – [17], в якому значення функції релевантності розраховується наступним чином за формулою (5.1) та (5.2):

$$BM25(d,q) = \sum_{t \in q} BM25w_{t,d}\,, \tag{5.1}$$

$$BM25w_{t,d} = idf_t \cdot \frac{(k_1+1)tf_{t,d}}{k_1\left(1-b+b\dfrac{dl}{avdl}\right)+tf_{t,d}}\,, \tag{5.2}$$

де d – документ;

q – запит;

t – леми запиту;

dl – довжина документу;

$avdl$ – середня довжина документу в колекції;

$f_{t,d}$ – частота леми t в документі d;

idf_t – зворотна частота зустрічає мості слова t;

$k_1 = 2$;

$b = 0{,}75$.

Навчальна вибірка будується на основі текстової колекції TREC-2017 та запитів із завдання TREC-2018 [30], [31]. Із завдання TREC-2018 [31] відібрані запити, кількість слів яких варіюється від 2 до 5, не включають в себе цифри, слова з друкарськими помилками, невідомі слова і при цьому в текстовій колекції міститься не менше 5 документів, що включають всі слова запиту. Вектор q складається з 11 компонент, частина з яких може бути нульовими. Перші 5 пар компонент містять значення $f_{t,d}$ й idf_t, $t \in q$, які є факторами, що описують запит, остання компонента визначає кількість слів запиту.

Вхідними векторами задачника є вектори параметрів запитів q, вихідними – вектори d параметрів документів, які отримали 1 ранг при ранжуванні за алгоритмом ОкаріBM25. Вектор d містить значення $tf_{t,d}$ та dl, довжина вектора d дорівнює 6. У задачник увійшов 141 запит довжини 2, 158 запитів довжини 3, 157 запитів довжини 4 і 133 запиту довжини 5. 80 % векторів використані в якості навчальних, а 20 % – в якості тестових.

В якості функції помилки котра вираховується за формулою (5.3):

$$MSE = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(y_{i,j} - d_{i,j}\right)^2, \tag{5.3}$$

де N – кількість прикладів;

M – розмірність вихідних векторів;

$y_{i,j}$ – компоненти вихідного вектору НМ;

$d_{i,j}$ – компоненти очікуваного вихідного вектору.

Кількість нейронів мережі Кохонена (кількість кластерів, що виділяються мережею) нарощувалася, починаючи з двох, до 8, виходячи з якості роботи гібридної мережі або комплексу мереж. В якості критерію якості використовувалася помилка навчання. Подальше збільшення числа нейронів не призводить до поліпшення якості роботи мережі, але призводить до збільшення обсягу обчислень.

В ході попередньої обробки даних виключена компонента, що відповідає кількості слів запиту, оскільки вона є лінійною комбінацією 5 інших компонент. В результаті аналізу значущих факторів вхідні вектори розділилися на кластери за кількістю слів у запиті. Оптимальне розбиття отримано при використанні 8 нейронів в шарі Кохонена: 2 кластера відповідали довжині запиту 2, 2 кластера відповідали довжині запиту 3, 2 кластера – довжині запиту 4 і 2 кластера довжині запиту 5. Для кожного кластера в якості значущих факторів визначені $tf_{t,d}$ та dl, де $t \in q$ для більшої частки документів. Розмір частки визначається параметрами ε і p. У дослідженні використані значення $\varepsilon = 0{,}01$ і $p = 0{,}25$, що відповідає 75%-ій частці. У прихованому шарі мережі використовувалося 16 нейронів.

Таблиця 5.1 – Результати навчання моделі на основі мережі Кохонена

№ кластеру	Розмір кластеру (дані для навчання)	Помилка на навчальних даних	Доля невірних відповідей (тестові дані)
1	51	0,0001	0
2	63	0,0001	
3	58	0	
4	61		
5	64	0,0533	0,03125
6	50	0,0003	0
7	60	0,0002	0,01667
8	63	0,0001	0

Відповіддю навченої НМ на вхідний вектор запиту є значення компонент вектора й, які забезпечать високий ранг документа за даним запитом. Відповідь НМ вважається невірним, якщо релевантність документа нижче, ніж у векторів - документів задачника.

В табл. 5.1 представлені помилки, отримані на навчальних і тестових даних (частка невірних відповідей), на восьми різних кластерах, виділених мережею Кохонена. Результати тестування методу показують успішне навчання моделей і низькі значення помилок навчання і тестування.

Тестування продуктивності проводилося на машині з наступними характеристиками: Intel Core i7-10750H (2.6 - 5.0 ГГц), RAM 8 ГБ, SSD 1 ТБ. Результати тестування представлені в таблиці 5.2.

Таблиця 5.2 – Продуктивність роботи системи

Вид показнику	Результат
Середній час обробки одного запиту	0,19 с
Середній час обробки одного запиту (вихідна система)	0,12 с
Середня кількість запитів в секунду	10,5
Середня кількість запитів в секунду (вихідна система)	16,6
Споживання оперативної пам'яті	1,2 Гб
Споживання оперативної пам'яті (вихідна система)	1,3 Гб

Відзначимо, що погіршення продуктивності системи природним чином пов'язано з її ускладненням і виконанням додаткових операцій. У процесі розробки акцент робився більше на гнучкість моделі даних і реалізації, ніж на мінімізацію кількості додаткових дій. Тому для отримання значень для більшої кількості кластерів потрібні додаткові звернення до бази та об'єктно-реляційне перетворення отриманих даних, що займає додатковий час.

5.2 Висновки за розділом 5

Запропоновано метод ранжирування на базі машинного навчання для підвищення релевантності пошуку у системах текстового пошуку, що засновано на оцінці значущості факторів. Метод включає кластеризацію даних на основі використання мережі Кохонена. Результати тестування методу показують успішне навчання моделей і низькі значення помилок навчання і тестування.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

6.1 Опис ідеї

Перед початком створення будь-якого програмного продукту важливо провести аналіз ринку, за допомогою якого можна зрозуміти чи є попит на розроблюваний продукт, визначити потенційних клієнтів. Окрім того попередній аналіз дозволить обчислити кошторис витрат на розробку проекту.

Розроблено проект на тему «Дослідження та програмна реалізація методів ранжирування на основі машинного навчання».

В результаті дослідження розроблено систему пошуку в колекціях документів, яка базується на моделях машинного навчання. Пошук в колекціях документів є важливим завданням. Про це свідчить як велика кількість пошукових систем, так і їх постійний розвиток. Колекції документів можуть бути різних типів: блоги, новинні стрічки, наукові статті або всі безліч веб-сторінок. Пошукові системи, такі як Google або Yahoo, оперують з останнім типом колекцій. Принцип роботи пошукової системи наступний: користувач вводить запит, після чого система повертає ті документи з колекції, які найкращим чином задовольняють запиту. Як правило, в традиційних пошукових системах ранжирування документів (визначення релевантності документа запиту) проводиться на основі статистичної інформації про безліч слів в запиті і в документі. На відміну від пошуку в базі даних, де результатом є безліч записів з ключами, що задовольняють логічній умові, результат пошуку в колекції документів не визначений точно. Тому вводяться параметри точності і повноти, що відображають якість пошуку.

Те, які документи будуть видані користувачеві у відповідь на конкретний запит, визначає функція ранжирування, або функція схожості. Традиційні пошукові системи оперують з векторною моделлю документа, в якій запит і документ розглядаються як N-мірні вектори, де N – загальна кількість термінів у системі [32]. Компоненти вектора – дійсні ваги, що

відображають важливість кожного відповідного терміна. Функція схожості визначається як косинус кута між векторами запиту і документа. Завдання ранжирування в даному випадку зводиться до задачі визначення ваг для термінів.

Існує ряд проблем, які не вдається вирішити, якщо функції ранжирування ґрунтуються тільки на вмісті самих документів і запиту, як це відбувається у векторній моделі. Наведемо приклади небажаних документів, тобто тих документів, які не є релевантними до запиту, але мають високий рейтинг в рамках векторної моделі. Наприклад, це документи, що містять рекламу і підвищують свій рейтинг одним або декількома з наступних нечесних способів:

- штучне підвищення частоти ключового слова для підвищення його ваги;
- додавання невидимого тексту, колір якого дорівнює кольору фону або розмір шрифту якого дорівнює 0;
- маскування – передача пошуковій машині тексту документа, що відрізняється від того, який бачать користувачі.

Інший приклад – документ-програма конференції з інформаційної безпеки. Набираючи в рядку пошуку назву якогось з криптографічних алгоритмів, користувач цілком може отримати текст-програму конференції, незважаючи на те, що цей текст не містить в собі опис цього алгоритму. Цей текст містить багато різних термінів, що відносяться до певної області, але може задовольнити лише малу частину інформаційних потреб, пов'язаних з даною областю знань, так як не описує конкретних понять, а лише перераховує їх.

Саме тому, актуальною задачею є вивчення поведінки алгоритмів текстового ранжирування, для підвищення якості та релевантності результатів пошукових семантичних систем, в тому числі при вирішенні завдань розпізнавання і адаптивної класифікації об'єктів за даними. Найближчими аналогами є проекти Google або Yahoo.

6.2 Попередня характеристика потенційного ринку

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів конкурентів.

Знаходження листів в електронній пошті, сайтів в Інтернет – все це відноситься до пошуку інформації в даному визначенні. Такий метод доступу до інформації починає домінувати серед інших форм доступу до інформації, переважаючи, наприклад, над пошуком в реляційних базах даних, що мають сувору структуру. Термін «неструктуровані дані» відноситься до даних, які не мають структури, яка могла б бути легко оброблена комп'ютером. В реальності, по-справжньому неструктурованих даних практично немає. Текст природною мовою має «лінгвістичну» структуру або, принаймні, виділені параграфи, абзаци та речення. ІП також використовується в разі напівструктурованого пошуку в разі, наприклад, пошуку документів, в заголовку яких міститься слово «Java», а в основному тексті – слово «багатозадачність» [3].

Прикладами програмних за стосунків, які аналізують стан здоров’я людини за показниками є Google або Yahoo. В свою чергу перераховані програмні системи є сильними конкурентами, що зайняли своє місце на ринку, оскільки це повноцінні продукти, які можуть обчислювати великі об’єми даних, враховують їх структурованість та наявні пропусків. Проте є доволі дорогими, є складними, та через роботу з великими об’ємами даних є порівняно повільними в роботі.

Результати проведеного аналізу потенційного ринку наведені в таблиці 6.1.

Формування ринку обумовлено потребою знаходження документів в колекції, які відповідають вимогам. Під останньою розуміється певна тема, в області якої користувач бажає дізнатися більше. Користувач формує запит,

який побічно представляє його потреба в інформації, хоча і відрізняється від неї. Запит – це спроба отримати від системи потрібні документи. Кажуть, що документ є релевантним, якщо Користувач сприймає його як такий, що містить необхідну інформацію, що відповідає його потребі в інформації. Перший приклад був штучним, в ньому потреба в інформації полягала в тому, містить той або інший документ певні слова, в той час як зазвичай цікавиться, наприклад, «витоками трубопроводу», і хотів би отримати релевантні документи незалежно від того, містять вони вихідне поняття або виражають його через інші, такі як, наприклад, «розрив трубопроводу» [7] – [9].

Таблиця 6.1 – Попередня характеристика потенційного ринку

№	Показники стану ринку (найменування)	Характеристика
1	Головні конкуренти	Google
		Yahoo
2	Динаміка ринку (якісна оцінка)	Зростає
3	Наявність обмежень для входу (вказати характер обмежень)	Надвеликі об’єми даних, наявність пропусків у даних, необхідність налаштування програмного продукту під конкретну галузь, проблема неструктурованих даних.
4	Специфічні вимоги до стандартизації та специфікації	Відсутні

Щоб оцінити ефективність пошукової системи, обчислюють наступні статистичні величини [1], [6] – [9]:

– точність – кількість релевантних документів в результаті, поділена на загальну кількість документів.

– повнота – кількість релевантних документів в результаті, поділена на загальну кількість релевантних документів в колекції.

При обсязі даних, описаних раніше, матриця зустрічальності термінів в документах, побудована наївним способом, буде займати занадто багато пам'яті: (біт інформації. Дана матриця є сильно розрідженою, так як через те, що кожен документ містить близько 1000 слів, 99.8% елементів матриці є нулями.

Використання цієї особливості-центральна ідея побудови інвертованого індексу. Цільовою аудиторією є заклади ЗВО та звичайні користувачі. Даний продукт може зацікавити їх високою точністю та швидкістю обчислення даних. Попередня характеристика потенційних клієнтів наведена в таблиці 6.2.

Таблиця 6.2 – Попередня характеристика потенційних клієнтів

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів (користувачів)
Потреба точної та швидкої пошуку інформації.	Державні та приватні установи	Система має бути доступною для користувача як за ціною так і за зрозумілістю використання.	Користувачі очікують, що програмний продукт буде проводити обробку та аналіз даних швидко та якість, забезпечуватиме результат високої точності.

В таблиці 6.3 наведено SWOT-аналіз розробленого застосування. Описано сильні сторони, такі як впровадження модифікованого генетичного алгоритму, за рахунок якого підвищується точність та швидкість роботи програмного продукту, та недоліки, а саме нездатність обробляти великі об’єми даних та швидкий темп розвитку галузі. Основними можливостями є впровадження інноваційних рішень та спрямованість на медичну діагностику взагалі. До загроз можна віднести появу сильних конкурентів на ринку.

Таблиця 6.3 – SWOT - аналіз

Сильні сторони	Слабкі сторони
- Іноваційність - Унікальність - Швидкість роботи алгоритму - Висока точність	- Нездатність працювати з великими об’ємами даних - Швидкий темп розвитку галузі
Можливості	Загрози
- Можливість впровадження проекту дляпошуку інформації. - Використання нейроалгоритмів	- Поява сильних конкурентів

6.3 Розроблення ринкової стратегії проекту

Потенційними клієнтами розроблюваного програмного продукту є державні і приватні заклади. Даний продукт може зацікавити їх високою точністю та швидкістю обчислення даних. Маючи деяку модель ранжирування в інформаційно-пошуковій системі, необхідно вміти відповідати на питання, наскільки ефективно працює конкретна функція ранжирування.

Для оцінки ефективності роботи пошукової системи потрібні три сутності:

– колекція документів;

– колекція тестових запитів;

– дані про те, які документи в колекції є релевантними для кожного тестового запиту.

Стандартний підхід полягає в тому, що для пари документ-запит відомо, чи є він релевантним чи ні, і не більше того. Тестова колекція повинна мати розумний розмір, щоб усереднені величини ефективності мали сенс. За необхідний мінімум приймається кількість в 50 запитів [6], [12]. Для прикладу наведемо одну широко використовувану тестову колекцію Text Retrieval Conference (TREC). Вона розробляється в Національному інституті стандартів і технологій в США.

Колекція має поділ за типами розв'язуваних завдань, при цьому всього в ній міститься 1.89 мільйонів документів (в основному, новинних статей) і 450 різних тестових запитів, а також оцінки релевантності кожної пари документ - запит.

Вже зазначалося, що в якості показників ефективності можуть використовуватися точність і повнота пошуку. Ці величини не залежать від порядку ранжирування результатів, а засновані тільки на множинах релевантних документів і результату, що видається системою. Крім цього, важко оцінювати зміни в ефективності системи, якщо, наприклад, збільшилася точність пошуку, але знизилася повнота.

Конкуренцію в сегментів складають як вітчизняні, так і закордонні компанії-розробники програмного забезпечення. Зазвичай, в Україні, незалежно від форми власності медичних закладів, використовують аналоги з меншою ціною, тож інтенсивність конкуренції оцінена з точки зору цінової політики. Відповідно до проведеного аналізу простота входу у сегмент визначена як середня.

Розроблення ринкової стратегії запропоновано в таблиці 6.4.

Таблиця 6.4 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегментів	Простота входу у сегмент
1	Державні і заклади	Середня	Помірний	Низька	Середня
2	Приватні заклади	Висока	Помірний	Середня	Середня

6.4 Калькуляція витрат на реалізацію проекту

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає:

– розрахунок витрат на заробітну плату працівникам;
– відрахування на єдиний соціальних внесок;
– розрахунок матеріальних витрат;
– витрати на спеціальне обладнання;
– накладні витрати.

6.4.1 Визначення складу, чисельності та фонду заробітної плати працівників

Для реалізації проекту необхідними учасниками є інженер-програміст та консультант. Розробка розпочинається 31 серпня і повинна бути виконана до 13 листопада 2020 року.

Ефективний фонд робочого часу визначається шляхом обчислення різниці між тривалістю календарного часу, кількості вихідних та святкових днів та кількості днів невиходу за причиною (6.1).

$$Ф_{РЧ} = T_К - T_{Вих} - T_{Невих}, \quad (6.1)$$

де $T_К$ – календарний час;

$T_{Вих}$ – вихідні і святкові дні;

$T_{невих}$ – невиходи за причиною.

Тривалість календарного часу складає 75 днів. За період розробки маємо 1 святковий день та 20 вихідних. Отже, ефективний фонд робочого часу складає:

$$Ф_{РЧ} = 75 - (20 + 1) = 56 \ днів.$$

Розрахунок заробітної плати за виконання проекту здійснюється на основі календарного плану робіт (табл. 6.5), в якому визначається трудомісткість, тривалість робіт, кількість виконавців.

Відповідно до табл. 1.5 програміст задіяний на проекті протягом 55 днів, консультант – 6 днів. Робочий день складає 8 годин.

Отже, тарифний заробіток кожного зі спеціалістів розраховується за формулою (6.2):

$$T_З = T_С \cdot РД \cdot Ф_{РЧС}, \quad (6.2)$$

де $T_С$– тарифна ставка за розрядом;

$РД$ – тривалість робочого дня (8 годин);

$Ф_{РЧС}$ – фонд робочого часу спеціаліста, тобто скільки днів спеціаліст задіяний на проекті.

Таблиця 6.5 – Етапи розробки продукту

Назва роботи	Трудомісткість		Виконавці	Тривалість, днів
	люд.–дні	% до підсумку		
1. Розробка календарного плану	1	1,64	Консультант	1
2. Аналіз предметної області	3	4,92	Програміст	3
3. Визначення вимог до програмного продукту та постановка завдання	2	3.28	Програміст	2
3. Розробка і затвердження технічного завдання	2	3.28	Програміст, консультант	1
3. Аналіз принципів оптимізації нейронних мереж	4	6.55	Програміст	4
4. Дослідження методів оптимізації	5	8.2	Програміст	5
5. Побудова моделі системи	6	9.84	Програміст	6
6. Створення програмного продукту	25	40.98	Програміст	25
7. Тестування і налагодження програмного продукту	5	8.2	Програміст	5
8. Складання програмної документації	8	13.11	Програміст, консультант	4
Разом	61	100		56

Відповідно до (6.2) тарифний заробіток програміста складає:

$$T_з = 120 \cdot 8 \cdot 55 = 52800\ грн.$$

Тарифний заробіток консультанта складає:

$$T_з = 70 \cdot 8 \cdot 6 = 3360\ грн.$$

Розрахунок річного фонду заробітної плати виконується на основі чисельності працівників, тарифної ставки, розміру преміальних виплат та додаткових виплат. Обчислення річного фонду оплати праці здійснюється за допомогою формули (6.3).

$$Ф_{ЗП} = \sum_{i=1}^{n} (Тс + П) \cdot Р_Г\ , \qquad (6.3)$$

де $Т_С$– тарифна ставка за розрядом;

$П$ – розмір премії;

$Р_Г$ – кількість робочих годин на рік;

n – кількість працівників.

Відповідно до (6.3) річного фонду заробітної плати становить:

$$Ф_{ЗП} = ((120 + 24) \cdot 2002) + (70 + 14) \cdot 2002) = 456456 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становлять 22% і беруться від основної та додаткової заробітної плати. Вони розраховуються за формулою (6.4):

$$ЄСВ = (Т_з + П) \cdot 0{,}22\ , \qquad (6.4)$$

де $Т_з$ - тарифний заробіток;

$П$ – розмір премії.

ЄСВ для програміста складає:

$$ЄСВ = (52800 + 10560) \cdot 0{,}22 = 13939{,}2\ грн.$$

ЄСВ для консультанта складає:

$$ЄСВ = (3360 + 672) \cdot 0{,}22 = 887.02\ грн.$$

Загалом додаткові соціальні відрахування становлять 14826,22 грн.

Витрати на заробітну плату працівників, премій, відрахування ЄСВ проводиться на підставі даних таблиці 6.5. Для наочного відображення всі проведені обчислення зведені до таблиці 6.6.

Таблиця 6.6 – Склад, чисельність та фонд заробітної плати працівників

Категорії працівників	Наявна чисельність осіб	Тарифна ставка за розрядом виконуваних робіт, грн/год	Ефективний фонд робочого часу, днів	Тарифний заробіток, грн	Преміальний відсоток до тарифного заробітку	Розмір премії, грн	Річний фонд заробітної плати, грн	ЄСВ, грн
1	2	4	5	6	7	8	9	10
Інженер-програміст	1	120	55	52800	20	10560	2882888	19939,2
Консультант	1	70	6	3360	20	672	168168	887,02
Всього	2	X	X	X	X	11232	456456	14826,22

6.4.2 Визначення витрат на матеріали та вартості спожитих послуг

У загальну вартість реалізації проекту необхідно включити вартість використаних матеріалів та спожитих послуг. Ціну кожного матеріалу та

відповідних послуг визначають за прейскурантами чи по іншим довідковими даними.

Витрати на матеріали розраховуються за формулою (6.5):

$$M = \sum_{i=1}^{n}(Ц_i N_i(1 + K_{т.з.}) - Ц_{io} N_{io}), \quad (6.5)$$

де M – витрати на матеріали, покупні напівфабрикати та комплектуючі, грн.;

$K_{т.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ – ціна і–го найменування чи комплектуючого, грн.;

N_i – потреба в і–му матеріалі, напівфабрикаті, компклетуючому;

$Ц_{io}$ – ціна зворотних відходів і–го найменування матеріалу, грн.;

N_{io} –кількість зворотних відходів і–го найменування;

n – кількість найменувань матеріалів, напівфабрикатів та комплектуючих.

Витрати на матеріали для розробки наведені в табл. 6.7.

Таблиця 6.7 – Розрахунок матеріальних витрат

Матеріальні витрати, найменування	Одиниця виміру	Ціна за одиницю, грн	Кількість використаних одиниць	Сума, грн
Папір офісний А4	пачка	85	1	85
Ручка	шт.	5	2	10
Блокнот	шт.	20	1	20
USB-флеш накопичувач 8Gb	шт.	130	1	130
Картридж принтера чорний	шт.	410	1	410
Разом				655

З урахуванням можливих транспортних витрат (5% від вартості

матеріалів), сумарна вартість матеріалів становить 687.75 грн.

$$M = 1{,}05 \cdot 655 = 687.75 \text{ грн.}$$

При виконанні роботи електроенергія споживалась комп'ютерною технікою (комп'ютер з периферійними пристроями) і настільною освітлювальною лампою; розрахуємо їх щоденні витрати.

Для розрахунку вартості експлуатації обладнання використаємо формулу (6.6):

$$C_E = N_H \cdot Ф_{еф} \cdot K_{ЗЧ} \cdot K_{ЗП} \cdot Ц_e \qquad (6.6)$$

де $N_н$ – номінальна потужність ЕОМ, кВт;

$Ф_{еф}$ – річний ефективний фонд часу роботи ЕОМ;

$K_{зм}$ – середні коефіцієнт завантаження за часом;

$K_{зп}$ – коефіцієнт завантаження за потужністю;

$Ц_е$– ціна одного кВт · год. Електроенергії, грн./(кВт · год.).

Для оцінки собівартості годин машинного часу необхідно визначити дійний фонд робочого часу ЕОМ. Дійсний фонд робочого часу ЕОМ обчислюється як число фактично відпрацьованих годин спеціаліста, за винятком часу на профілактику та ремонт ЕОМ. Час профілактики: щомісячно – 5 годин, використаємо формулу (6.7).

$$Ф_{еф} = Ф_{РЧ} - Ч_{ПР}, \qquad (6.7)$$

де $Ф_{РЧ}$ – фонд робочого часу працівника (448 години);

$Ч_{ПР}$ – часу на профілактику та ремонт ЕОМ.

Отже, річний фонд часу роботи ЕОМ становить 436 години. Середні коефіцієнти завантаження за часом та за потужністю становлять 0.6 та 0.9.

Відповідно, до формули 1.6 вартість експлуатації ЕОМ складає:

$$C_E = 0{,}3 \cdot 436 \cdot 0{,}8 \cdot 1{,}68 = 175{,}8\ грн.$$

Також при роботі використовувалась настільна освітлювальна лампа, яка використовувалася протягом 2 год/день (112 год протягом усього проекту). Отже, вартість її експлуатації становить:

$$C_E = 0{,}1 \cdot 112 \cdot 0{,}8 \cdot 1{,}68 = 15.05\ грн.$$

Загалом на електропостачання витрачено 190,85 грн.

Витрати на водопостачання визначаються з розрахунку на 1 особу 4м^3/місяць. Оскільки розробка ведеться протягом 2,5 місяців то споживання води приймаємо у розмірі 12м^3/місяць з урахуванням того що програміст працює 55 днів, а консультант 6 днів. Вартість 1м^3 становить 20,77 грн. Отже витрати на водопостачання становлять 249,24 грн.

Витрати на теплопостачання обчислюються з урахуванням площі приміщення(42 м^2) та дати початку опалення (02.11.2020) та дати завершення проекту (13.11.2020). Вартість опалення 1м^2 становить 34,56 грн/місяць. Отже, вартість опалення у період з 02.11 до 13.11 з врахуванням площі приміщення становить 580,6 грн.Розрахунок вартості спожитих послуг наведен в табл. 6.8.

Таблиця 6.8 – Розрахунок вартості спожитих послуг

Вид послуг	Норма витрат	Ціна, грн	Сума, грн
Електропостачання	0,4 кВт·год	1,68	190,85
Водопостачання	4м^3/ос·міс	20,77	249,24
Теплопостачання	1м^2	34,56	580,6
Разом			1020,69

6.4.3 Розрахунок суми амортизації

Суму амортизаційних відрахувань від балансової вартості універсального обладнання, апаратів і приладів, які відносяться до основних фондів, розраховується відповідно до їх зайнятості в даній роботі. Для використаного комп'ютерного обладнання сума амортизації становить 25% на рік від його балансової вартості:

$$A = Ф_б \cdot H_a \qquad (6.8)$$

де, $Ф_б$ – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, %.

Підставивши у формулу (6.8) значення балансової вартості використовуваних ЕОМ DELLInspiron 15, маніпулятору типу миша та принтеру HP Deskjet 2710 (разом 17 500 грн.) отримаємо:

$$A = 17500 \cdot 0{,}25 = 4375 \ грн$$

Амортизаційні відрахування становлять 4375 грн.

Результати розрахунків зводимо в таблицю 6.9.

Таблиця 6.9 – Розрахунок суми амортизаційних відрахувань

Обладнання	Балансова вартість, грн	Норма амортизації, %	Сума амортизації, грн
Комп'ютерна техніка	17500	25	4375

6.4.4 Калькуляція кошторису витрат

До накладних витрат при виконанні даної роботи відносяться витрати за інтернет-послуги, оренду приміщення та інші витрати, які складають 40% від основної заробітної плати розробника і становлять 22464 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведено в таблиці 6.10. У таблиці розрахована підсумкова сума вартості робіт, а так само питома вага кожної статті витрат у відсотках від підсумкової суми.

Таблиця 6.10 – Кошторис витрат

Калькуляційні статті	Витрати за період виконання проекту, грн	Питома вага від підсумку, %
Електропостачання	190,85	0,19
Водопостачання	249,24	0,25
Теплопостачання	580,6	0,58
Заробітна плата	56160	56,43
ЄСВ	14826,22	14,9
Амортизація універсального обладнання	4375	4,4
Витрати на матеріали	655	0,65
Накладні витрати	22464	22,6
Всього	99500,91	100

6.5 Висновки за розділом

Розробка програмного забезпечення проводиться за рахунок гранту в 98 тисяч гривень. Оскільки витрати на розробку програмного забезпечення є нижчими, ніж сума гранту, розробка є економічно обґрунтованою.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

Сучасний розвиток науки та техніки привносить принципові нововведення у всі сфери матеріального виробництва, докорінно змінюючи знаряддя та предмети праці, технологію, методи обробки інформації. Разом з тим, захопившись вдосконаленням засобів праці їх творці залишили поза увагою проблеми людини в рамках своєрідної технічної та комп'ютерної революції.

Комп'ютери, телебачення, системи зв'язку та інші засоби, що використовують досягнення радіоелектроніки є генераторами цілої низки електромагнітних випромінювань, вплив яких на організм людини ще не зовсім вивчений.

Методологічною основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей впливу на організм людини.

Дослідження та програмнареалізація методів ранжирування на основі машинного навчання передбачає роботу в обчислювальному центрі невеликої команди програмістів. Розміри приміщення $9 \times 4 \times 3.5 м$.

Площа приміщення становить $9 \cdot 4 = 36 м^2$, об’єм – $9 \cdot 4 \cdot 3.5 = 126 м^3$.

Небезпечні та шкідливі виробничі фактори за природою виникнення поділяються на такі групи: фізичні (підвищена і знижена температура повітря, надмірна запиленість і загазованість повітря та ін.); хімічні (вміст у робочій зоні шкідливих речовин та ін.); психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.); біологічні (підвищений вміст у повітрі мікроорганізмів та ін.).

У даному розділі розглянуто наступні питання: мікроклімат та склад повітря робочої зони; виробниче освітлення; шум; виробничі випромінювання;

безпека щодо організації робочих місць; безпечність технологічного обладнання та процесу; електробезпека; пожежна безпека.

7.2 Заходи із забезпечення безпеки

При роботі в обчислювальному центрі головним небезпечним фактором є електрика.

За ступенем небезпеки ураження електричним струмом, приміщення обчислювального центру відноситься до приміщень без підвищеної небезпеки.

Питання правильного електроживлення при об'єднанні комп'ютерів в локальні мережі мають важливе значення як для забезпечення електробезпеки користувачів, так і для безаварійної і безперебійної роботи обладнання.

Заходи щодо забезпечення електробезпеки розроблено відповідно до ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним» [33].

Приміщення живиться електричною енергією від централізованого трансформатора, який знаходиться на поверсі поза межами приміщення і понижує напругу до 220 В, змінним струмом частотою 50 Гц. Комп’ютери мають власні блоки живлення, що перетворюють змінний струм мережі у постійний (5 В, 12 В). Потужність даних блоків живлення: 400 – 450 Вт. ПК у приміщенні відносяться до категорії І захисту від ураження електричним струмом, оскільки усі прилади при підключенні до електричної мережі яка заземляються, використовуючи систему TN. Тобто система, у якій живильні мережі має глухе заземлення однієї точки струмопровідних частин джерела живлення, а електроприймачі і відкриті провідні частини електроустановки приєднуються до цієї точки за допомогою відповідно нейтрального і захисного провідників. Корпуса всіх електричних пристроїв виготовляються з неструмопровідних матеріалів, а живлення здійснюється спеціальним кабелем, так щоб виключити ураження людини електричним струмом. Зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів

відповідно до міжнародного стандарту CE. У приміщенні електропроводка трифазна, чотирьохжильна, з подвійною ізоляцією. Всі кабелі сховані в коробах. Штепсельні розетки встановлені на висоті одного та 0,2 метри від підлоги. Вимикачі на стінах розташовані на висоті 1,2 метра від підлоги.

Електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником).

Для запобігання ураження людини струмом служить захисне заземлення. Сучасні ПК включаються в розетку за допомогою триполюсної вилки, один з полюсів якої використовується для заземлення. З ним електрично з'єднані металеві неструмоведучі частини ПК, які можуть опинитися під напругою. В обчислювальному центрі передбачається використання тільки триполюсних розеток, у яких відповідний полюс з'єднаний з шиною заземлення.

Кожен блок живлення комп'ютера або периферійного пристрою має мережевий фільтр. Конденсатори цього фільтра призначені для шунтування високочастотних перешкод живлячої мережі на землю через дріт захисного заземлення.

При з'єднанні двох пристроїв (комп'ютера та принтера) інтерфейсним кабелем, загальний провід інтерфейсів послідовних і паралельних портів пов'язаний зі «схемної землею» і корпусом пристрою. Якщо пристрої, які з'єднуються, надійно заземлені через окремий провід на загальний контур, проблеми різниці потенціалів не виникає [17].

Згідно з «Правилами улаштування електроустановок» (ПУЕ) опір захисного заземлення в обчислювальному центрі становить 3 ОМ (за стандартом – не більше 4 Ом) [34].

Згідно з ГОСТ 12.2.007.0–75 «Вироби електротехнічні. Загальні вимоги безпеки» в приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру.

7.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного ПК з ВДТ розроблені відповідно до вимог Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу», МЮУ 06.05.2014 р. за № 472/25249, ДСанПіН 3.3.2.007-98 «Державні стандартні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [19] і НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям» [20].

Тип роботи програмістів в обчислювальному центрі належить до операторської роботи, даний тип роботи відносить до категорії 1а, тобто до неї належать роботи, що виконуються сидячи і не потребують фізичного напруження, робоче місце є постійним, робота проводиться у холодний та теплий періоди року, тому рекомендується підтримувати оптимальні показники мікроклімату.

Роботи, що виконуються сидячи і не потребують фізичного напруження, при яких витрати енергії складають до 139 Вт. Для забезпечення оптимального рівня параметрів повітряного виробничого середовища використовуємо ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [21]. (табл. 7.1).

Таблиця 7.1 – Норми мікроклімату в приміщенні

Період Року	Категорія робіт	Температура повітря, °С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодний	Легка	22–24	40–60	0,1
Теплий	Легка	23–25	40–60	0,1-0,2

В обчислювальному центрі передбачено: устрій системи водяного опалення приміщення для забезпечення необхідної температури повітря в холодний період року відповідно ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціювання».

Проведені дослідження мікрокліматичних умов на комп'ютеризованих робочих місцях показали, що зимою значення відносної вологості повітря часто можуть нижчими за встановлені норми і становити в середньому 30–40%. Це призводить не лише до надмірного висихання слизових оболонок очей, носа, горла, а й до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера. Для усунення таких проблем, в приміщені передбачена наявність 3 зволожувачів повітря, які допомагають підняти вологість до 60%. Кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м^2.

У теплий період року для підтримки нормальної температури повітря в приміщенні використовуються кондиціонери типу «спліт-система». Два кондиціонера розташовані у протилежних частинах обчислювального центру дозволяють підтримувати температуру повітря не вище 24°С. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціювання приміщення площею до 18 м^2[35].

Вимоги до освітлення регламентовані ДБН В.2.5-28-2018 [23], згідно з якими освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним. Природне освітлення бічне, односторонне.

В обчислювальному центрі коефіцієнт природної освітленості (КПО) має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду), це відповідає до вимогам ДБН В.2.5-28-2018. Для забезпечення відносної постійності природного освітлення вікна обладнанні сонцезахисними металевими жалюзі з системою регульовання. В таких жалюзі коефіцієнт відбиття становить 0,7. Робочі місця розташовано так, щоб у поле зору не потрапляли вікна або світлі поверхні світильників.

Найбільш прийнятними для приміщень, де працюють ПК, є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 20, 40 або 80 Вт.

Розрахуємо кількість ламп необхідних для освітлення приміщення розміром $9 \times 4 \times 3.5 м$ освітленістю $E_p = 400 лк$. Коефіцієнт відбиття стелі 70% і стін 50%. Для освітлення використовуються люмінесцентні лампи типу ЛБ в світильниках ЛПО.

Знаходимо індекс приміщення, використаємо формулу (7.1):

$$i = \frac{A \cdot B}{h \cdot (A + B)} = \frac{9 \cdot 4}{3{,}5 \cdot (9 + 4)} = 0{,}8 . \qquad (7.1)$$

Приймаємо коефіцієнт запасу k = 1,6 і коефіцієнт нерівномірності освітлення, використаємо формулу (7.2):

$$z = \frac{B_{ср}}{B_{хв}} = 1.1. \qquad (7.2)$$

При індексі i=0,8 отримуємо η=32%.

Світильники розміщуємо в два ряди (Np = 2).

Визначаємо необхідний світловий потік ламп в кожному ряду використаємо формулу (7.3):

$$\Phi_p = \frac{E_н \cdot S \cdot z \cdot k}{N_p \cdot \eta} = \frac{400 \cdot 36 \cdot 1{,}1 \cdot 1{,}6}{2 \cdot 0{,}32} = 39600 лм . \qquad (7.3)$$

Якщо в світильнику встановити по дві лампи ЛБ (n = 2) потужністю 80 Вт і світловим потоком Фл = 5400лм, то необхідне число світильників в ряду складе (7.4):

$$N_р = \frac{Ф_р}{n \cdot Ф_л} = \frac{39600}{2 \cdot 5400} = 4. \quad (7.4)$$

Для виключення засвічення екранів дисплеїв прямими світловими потоками світильники загального освітлення мають в своєму розпорядженні збоку від робочого місця, паралельно лінії зору програміста і стіні з вікнами.

Згідно із технічним паспортом обладнання (ПК, монітори, принтери, клавіатури (при натисканні), маніпулятори типу «миша» (при натисканні)), що використовується для роботи, рівень шуму складає: 43 дБА. Згідно із ДСН 3.3.6-039-99 нормоване значення рівня шуму більше, ніж фактичне, тож додаткових робіт для усунення надмірного шуму не проводилося. Роботи в даному приміщенні з таким обладнанням відповідають нормативним вимогам.

Обладнання і організація робочого місця в обчислювальному центрі мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності.

Площа на одне робоче місце з ПК складає по 8 м^2, а об’єм – по 28 м^3. Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку.

Робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам.

Робочі стільці мають підйомно-повторним механізми та регулюються по висоті та куту нахилу сидіння і спинки, а також по відстані спинки від переднього краю сидіння. Регулювання кожного параметра незалежне, легко здійснюється та має надійну фіксацію. Екран монітору повинний знаходитися на оптимальній відстані 700 мм.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташовує

елементів робочого місця, що важливе для підтримки оптимальної робочої пози людини-оператора.

Також приміщення обладнано шафами для зберігання документів, магнітних дисків, полицями, стелажами та тумбами.

Колір приміщень і меблів сприяє створенню комфортних умов для зорового сприйняття.

Загальний план обчислювального центру та робочих місць наведено на рис.7.1.

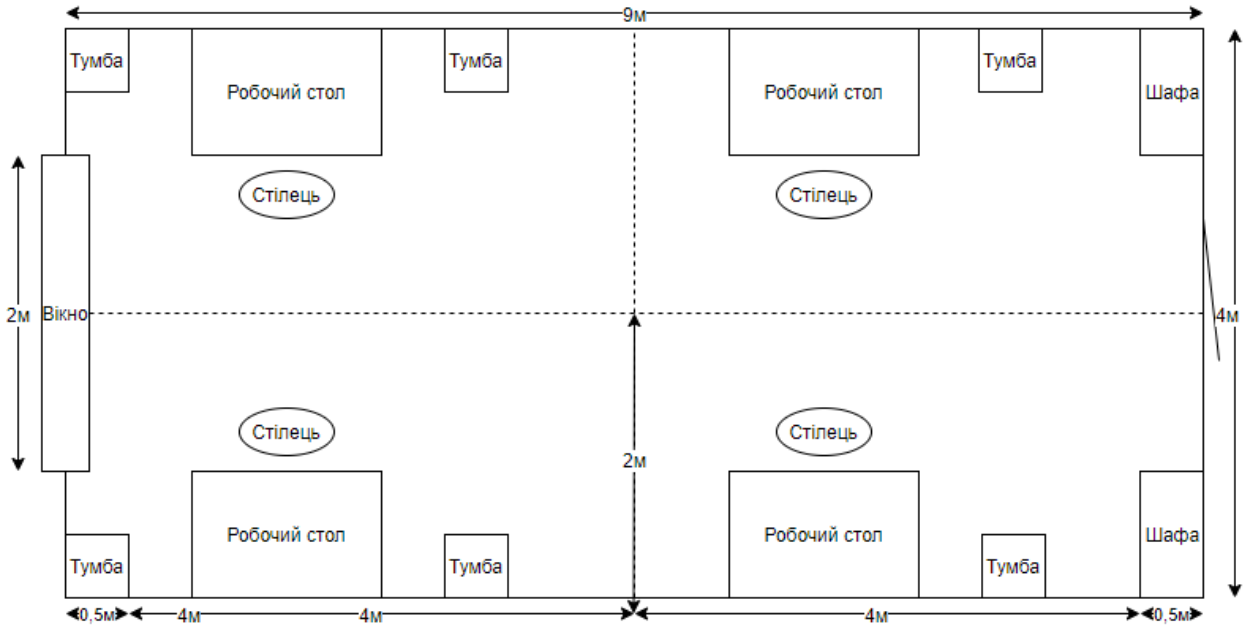

Рисунок 7.1 – Загальна схема обчислювального центру

7.4 Заходи з пожежної безпеки

Пожежі в обчислювальному центрі становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами. Як відомо, пожежа може виникнути при взаємодії горючих речовин, окислення і джерел запалювання. У приміщеннях ВЦ присутні всі три основні чинника, необхідні для виникнення пожежі.

Обчислювальний центр, згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [34] – [36] відноситься до категорії «П-IIа», а клас

можливої пожежі, згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги», визначається, як «А» [25].

До засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться пожежні стовбури, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. п.

Оскільки приміщення (дослідницької лабораторії, конструкторського бюро, тощо) що обладнане ПК з ВДТ має площу 36 м^2, тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 2 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м^2 площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

Усі працівники обчислювального центру під час прийняття на роботу і в процесі праці повинні проходити протипожежний інструктаж, перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними та нічим не захаращуються.

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі.

Фільтри припливно-витяжної вентиляції слід чистити згідно з затвердженим графіком, але не рідше двох разів на рік. Роботи з ремонту вузлів (блоків) ЕОМ, як правило, проводити не дозволяється. Їх необхідно проводити в окремому приміщенні (майстерні). У разі необхідності проведення ремонту або технічного обслуговування ЕОМ безпосередньо в залі допускається мати не більше 0,5 л легкозаймистих рідин у небиткій, щільно закритій тарі. Для

промивання деталей необхідно застосовувати, як правило, негорючі миючі препарати. Чарунки та інші знімні пристрої допускається промивати горючими рідинами тільки в спеціальному приміщенні, обладнаному припливно-витяжною вентиляцією.

Відповідальний за пожежну безпеку приміщень обчислювального центру зобов'язаний негайно сповістити технічну службу у разі виявлення незадовільного технічного стану димових та теплових сповіщувачів автоматичної пожежної сигналізації, а також про несправності автоматичної установки об'ємного (газового) пожежогасіння [17].

Усі працівники обчислювального центру повинні вміти привести в дію (в разі потреби) первинні засоби пожежогасіння. Регулярно, але не рідше одного разу на квартал, слід очищати від пилу ЕОМ, контрольновимірювальну апаратуру, кабельні траншеї та фальшпідлоговий простір за допомогою пиловсмоктувача.

7.5 Заходи забезпечення безпеки у надзвичайних ситуаціях

Надзвичайна ситуація – це будь-яка екстрена обстановка, при якій існує загроза безпеці людей. Наприклад, це може бути стихійне лихо, техногенна катастрофа, оголошення війни або теракт. У побутовому плані надзвичайна ситуація може скластися в результаті ДТП, вибуху побутового газу або іншій події, небезпечній для оточуючих.

Виробнича або транспортна катастрофа-це велика аварія, що спричинила за собою людські жертви, значний матеріальний збиток та інші важкі наслідки.

Відомо, що будь – яка діяльність потенційно небезпечна, а самі небезпеки носять перманентний характер (перманентний – постійний, безперервно триває, від латинського permaneo-залишаюся, продовжуюся).

Потенційна небезпека-це небезпека прихована, Невизначена в часі і просторі. Реалізується потенційна небезпека через причини і в разі, якщо

небажані наслідки будуть значні, то ця подія класифікується як надзвичайна ситуація.

У житті всі відхилення від звичайного, нормального ми називаємо надзвичайною подією або ситуацією. У нормативних документах даються наступні визначення.

Надзвичайна ситуація – це обстановка на певній території, що склалася в результаті аварії, небезпечного природного явища, катастрофи, стихійного чи іншого лиха, які можуть спричинити або спричинили за собою людські жертви, шкоду здоров'ю людей або навколишньому природному середовищу, значні матеріальні втрати і порушення умов життєдіяльності людей.

Екстремальна подія – це відхилення від норми процесів або явищ.

Аварія – це екстремальна подія техногенного характеру, що сталася за конструктивними, виробничим, технологічним або експлуатаційним причин, або через випадкові зовнішніх впливів, і полягає в пошкодженні, виході з ладу, руйнуванні технічних пристроїв або споруд.

Небезпечне природне явище – це стихійна подія природного походження, яке за своєю інтенсивністю, масштабом поширення і тривалості може викликати негативні наслідки для життєдіяльності людей, економіки і природного середовища.

Стихійне лихо – це катастрофічне природне явище (або процес), яке може викликати численні людські жертви, значний матеріальний збиток та інші важкі наслідки.

Екологічна катастрофа (екологічне лихо) – надзвичайна подія особливо великих масштабів, викликана зміною (під впливом антропогенних факторів) стану суші, атмосфери, гідросфери і біосфери, що супроводжується масовою загибеллю живих організмів і економічним збитком.

Всю сукупність можливих надзвичайних ситуацій доцільно спочатку розділити на конфліктні і безконфліктні.

До конфліктних, перш за все, можуть бути віднесені військові зіткнення, економічні кризи, екстремістська політична боротьба, соціальні вибухи,

національні та релігійні конфлікти, тероризм, розгул кримінальної злочинності, великомасштабна корупція та ін.

Безконфліктні надзвичайні ситуації, в свою чергу, можуть бути класифіковані (систематизовані) за значною кількістю ознак, що описують явища з різних сторін їх природи і властивостей.

Всі надзвичайні ситуації можна класифікувати за трьома основними принципами-масштабом поширення, темпу розвитку і природою походження.

Класифікація надзвичайних ситуацій за масштабом поширення

При класифікації надзвичайних ситуацій за масштабом поширення слід враховувати не тільки розміри території, що піддалася впливу надзвичайні ситуації, а й можливі її непрямі наслідки. До них відносяться важкі порушення організаційних, економічних, соціальних та інших істотних зв'язків, що діють на значних відстанях. Крім того, береться до уваги тяжкість наслідків, яка і при невеликій площі НС може бути величезною і трагічною.

Локальні (приватні) надзвичайні ситуації не виходять територіально і організаційно за межі робочого місця або ділянки, малого відрізка дороги, садиби або квартири. До локальних відносяться надзвичайні ситуації, в результаті яких постраждало не більше 10 осіб, або порушені умови життєдіяльності не більше 100 осіб, або матеріальний збиток становить не більше 1 тис.мінімальних розмірів оплати праці.

Якщо наслідки надзвичайної ситуації обмежені територією виробничого або іншого об'єкта (тобто не виходять за межі санітарно-захисної зони) і можуть бути ліквідовані його силами і ресурсами, то ці надзвичайні ситуації називаються об'єктовими.

Надзвичайні ситуації, поширення наслідків яких обмежено межами населеного пункту, міста (району), області, краю, республіки і усуваються їх силами і засобами, називаються місцевими. До місцевих відносяться надзвичайні ситуації, в результаті яких постраждало понад 10, але не більше 50 осіб, або порушені умови життєдіяльності понад 100, але не більше 300 осіб,

або матеріальний збиток становить понад 1 тис., але не більше 5 тис. мінімальних розмірів оплати праці.

Регіональні надзвичайні ситуації-такі надзвичайні ситуації, які поширюються на територію кількох областей (країв, республік) або економічний район. Для ліквідації наслідків таких надзвичайні ситуації необхідні об'єднані зусилля цих територій, а також участь федеральних сил. До регіональних належать НС, в результаті яких постраждало від 50 до 500 осіб, або порушені умови життєдіяльності від 500 до 1000 осіб, або матеріальний збиток становить від 0,5 до 5 млн.мінімальних розмірів оплати праці.

Національні (федеральні) надзвичайні ситуації охоплюють великі території країни, але не виходять за її межі. Тут задіюються сили, засоби і ресурси всієї держави. Часто вдаються і до іноземної допомоги. До національних відносяться НС, в результаті яких постраждало понад 500 осіб, або порушені умови життєдіяльності понад 1000 осіб, або матеріальний збиток становить понад 5 млн.мінімальних розмірів оплати праці.

Глобальні (транскордонні) надзвичайні ситуації виходять за межі країни і поширюються на інші держави. Їх наслідки усуваються силами і засобами як постраждалих держав, так і міжнародного співтовариства.

Високий ступінь нервового потрясіння, загроза життю і здоров'ю, екстрене становище в країні – все це може викликати у людей стан шоку. Воно може проявлятися наступними ознаками:

– прискорене дихання і серцебиття або, навпаки, "втрата" дихання (пульс слабо прощупується);

– розширення зіниць;

– блідість;

– заціпеніння, неможливість адекватно реагувати на ситуацію;

– поява холодного поту;

– загальна слабкість

Що ж робити, якщо ви опинилися в надзвичайній ситуації.

По-перше, зберігати спокій. Дуже важливо дотримуватися холоднокровність в будь-якій надзвичайній ситуації. На жаль, найчастіше небезпека полягає в тому, що люди роблять необдумані дії, які і призводять до їх загибелі. Ви точно повинні знати, що будете робити в наступну хвилину і які кроки зробите. Однак намагайтеся не витрачати на це занадто багато часу, пам'ятайте, що зволікання в небезпечній ситуації також загрожує серйозними наслідками.

По-друге, підготуйте комплект першої необхідності (а краще два комплекти) для будь-якої надзвичайної ситуації слід скласти і зберігати в окремій непромокаючої сумці, в легко доступному місці Комплект першої необхідності. Він включає в себе наступні речі:

а) аптечку першої допомоги;

б) консервовані продукти в банках і питну воду в пластикових пляшках (можна ще банку кави і пачку чаю, коробочку з сіллю);

в) транзисторний приймач і електричний ліхтарик;

г) комплект запасних батарейок для приймача і ліхтарика;

д) сірники в герметичній коробці, запальничку, папір на випадок розпалювання багаття;

е) інструменти і столове приладдя-складаний ніж, ложки, виделки, миски і гуртки, відкривачку для консервів, легкий туристський топірець;

ж) ліки, що вживаються членами сім'ї за медичними показниками регулярно (якщо інструкція не вимагає їх зберігання в холодильнику.

По-третє, не беріть нічого зайвого.Зберігайте документи та цінні речі в одному місці. Це потрібно для того, щоб в разі необхідності ви могли без зволікання зібрати їх і покинути житло.

Також, постарайтеся покинути небезпечне місце. Важливо пам'ятати про те, що якщо ви не є фахівцем (наприклад, рятувальником або лікарем), не потрібно намагатися допомогти професіоналам. Цим ви можете тільки погіршити ситуацію.

Важливо, у разі надзвичайної ситуації природного або техногенного характеру ніколи не користуйтеся ліфтом.

У таких ситуаціях електроенергія, як правило, відключається в першу чергу. Це стосується і пожежі, і землетрусу, і повені, і багатьох інших випадків.

Нарешті, викличте екстрені служби. Номери екстрених служб обов'язково повинні бути записані в телефоні. Пам'ятайте, що в надзвичайних обставинах навіть легкі номери можна забути [37] – [43].

ВИСНОВКИ

Під час виконання магістерської роботи, було запропоновано новий метод ранжирування на основі машинного навчання, а саме методу кластеризаціїї k-середніх та штучних нейронних мереж Кохонена. Також було розроблено програмний комплекс, що реалізує роботу запропонованого методу.

Для цього були поставлені і вирішені такі задачі:

– ознайомлення та аналіз предметної області;

– розгляд мови програмування Python та бібліотеки kohonenv.1.1.2;

– розроблення програмного комплексу для реалізації та тестування запропонованого методу.

Розроблений програмний комплекс, реалізує основні функції, а саме:

– отримання та передача пошукового запиту;

– опрацювання пошукового запиту клієнта;

– ранжирування відповідей на запит;

– виведення результатів;

– надання та забезпечення GUI.

Запропонований метод значнопідвищує рівень релевантності відповіді при інформаційному пошуку.

Розроблений програмний комплекс має практичну спрямованість та готовий до використання.

Під час розробки програми та аналізу розробленого програмного комплексу з’явилися ідеї з удосконалення, такі як: використання факторного аналізу на основі методу головних компонент для відбору значущих факторів для визначення релевантності, а також використання Байєсових мереж та нечіткої логіки або ж генеративних нейронних мереж для підвищення точності.

Нові можливості зможуть розширити функціонал програмного комплексу та підвищити його продуктивність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кравченко Р.І. Інформаційний пошук як ресурс технічного регулювання у сфері цивільного захисту [Текст] / Р.І. Кравченко, О.В. Савченко, В.В. Ніжник // Науковий вісник УкрНДІПБ. 2013. – № 2. – С. 178–18.

2. Збанацька О. Інформаційний пошук за стандартами інформаційної сфери: термінологічний аспект [Текст] / О. Збанацька // Наукові праці Національної бібліотеки України імені В.І. Вернадського, –2017. – Вип. 48. – С. 45–63.

3. П’ятчаніна Т.В. Патентно–інформаційний пошук як сучасний інструмент досліджень у галузі медико–біологічних наук [Текст] / Т.В. П’ятчаніна, А.М. Огородник, О.В. Васильєв, В.В. Чьочь // Наука та інновації, – 2020. – Т. 16, № 3. – С. 81–87.

4. Шаравара Т.О. Інформаційний пошук і робота з бібліотечними ресурсами [Текст] : навч. посіб. для аспірантів / Шаравара Т.О. // Полтав. держ. аграр. акад., Каф. інозем. мов та українознавства. – Київ : Ліра–К, 2017. – 255 с.

5. Васильєв О.В. Створення змішаних документографічних інформаційних систем на базі міжнародних інформаційних мереж [Текст] / О.В. Васильєв // НТІ, – 2000. – № 2. – С. 42–45.

6. Lytvyn V.V. Method of data expression from the Ukrainian content based on the ontological approach [Text] / V.V. Lytvyn, V.A. Vysotska, M.H. Hrendus // Радіоелектроніка. Інформатика. Управління, – 2018. – № 3. – С. 144–157.

7. Москаленко В.В. Модель і алгоритм навчання детектора шкідливого трафіку на основі модифікації зростаючого нейронного газу [Текст] / В.В. Москаленко, А.С. Москаленко, М.О. Зарецький // Радіоелектрон. і комп'ютер. Системи, – 2018. – № 3. – С. 11–19.

8. Кириченко І.В. Концепція пошукової системи агентного типу в системах електронного навчання [Текст] / І.В. Кириченко, І.Ю. Шубін // Системи оброб. Інформації, – 2018. – Вип. 2. – С. 100–107.

9. Шумейко О.О. Ранжування документів за інформаційним запитом [Текст] / О.О. Шумейко, Г.Я. Шевченко // Систем. технології, – 2017. – № 2. – С. 110–117.

10. Швед А.В. Моделі та інформаційні технології кластеризації та ранжирування групових експертних оцінок в умовах невизначеності [Текст] : автореф. дис. канд. техн. наук : 05.13.06 / Швед Альона Володимирівна. – Миколаїв: ЧДУ ім. Петра Могили, 2013. – 20 с.

11. Маргасов Д.В. Алгоритмізація та корекція рівня узгодженості під час ранжирування факторів в інформаційній системі аналізу енергоефективності [Текст] / Д.В. Маргасов, В.І. Литвин // Технічні науки та технології, – 2015. – № 2. – С. 175–179.

12. Овчинникова Ю.Ю. Типологічне ранжирування ключових територій екологічної мережі Східного Поділля [Текст] / Ю.Ю. Овчинникова // Науковий вісник НЛТУ України, – 2018. – Т. 28, № 2. – С. 81–85.

13. Євсєєв В. Результати ранжирування факторів, що впливають на ступінь захищеності важливих державних об’єктів [Текст] / В. Євсєєв, В. Пащенко // Збірник наукових праць Національної академії Державної прикордонної служби України. Сер. : Військові та технічні науки, – 2019. – № 1. – С. 47–58.

14. Струнгар А. Пертинентність і релевантність інформаційних ресурсів при пошуку інформації в електронних бібліотеках [Текст] / А. Струнгар // Наукові праці Національної бібліотеки України ім. В. І. Вернадського, – 2014. – Вип. 39. – С. 407–416.

15. Свідрик Т.І. Показники в системі діагностики витрат: якісна характеристика і релевантність [Текст] / Т.І. Свідрик // Наукові записки [Української академії друкарства], – 2008. – № 2. – С. 83–89.

16. Alvo M. Statistical Methods for Ranking Data (Frontiers in Probability and the Statistical Sciences) [Text] / M. Alvo, P. L. H. Yu. – Berlin : Springer, 2016. – 284 p.

17. Strevens M. The Knowledge Machine: How Irrationality Created Modern Science [Text] / M. Strevens. – New York : Springer, 2020. – 368 p.

18. Синько А.І. Застосування кластеризації на основі самоорганізаційних карт кохонена для розподілу користувачів на групи [Текст] / А.І. Синько, А.М. Пелещишин // Вісник Хмельницького національного університету. Економічні науки, – 2020. – № 1. – С. 100–105.

19. Воєводін Є.В. Порівняння ефективності стратегій розподілення з використанням самоорганізаційних карт Кохонена в системах оркестрування віртуальних контейнерів [Текст] / Є.В. Воєводін // Вісник Черкаського університету. Серія : Прикладна математика. Інформатика, – 2017. – № 1–2. – С. 91–100.

20. Маникаева О.С. Машинное обучение самоорганизующегося слоя кохонена в системах нейросетевого распознавания образов по статистической информации [Текст] / О.С. Маникаева, Е.А. Арсирий, А.П. Василевская // Системні технології, – 2015. – Вип. 5. – С. 93–105.

21. Годич О.В. Індуктивні методи та алгоритми самоорганізації моделей даних на основі карт Кохонена [Текст] : дис. канд. техн. наук : 01.05.03 / Годич Олесь Васильович. – Л. : ЛНУ ім. І. Франка, 2010. – 171 с.

22. Poryev G.V. Efficiency of the Overlay Network Clustering based on the Locality Metric [Text] / G.V. Poryev // Реєстрація, зберігання і обробка даних, – 2011. – Т. 13, № 2. – С. 47–52.

23. Chechko V.E. Qualitative analysis of clustering in aqueous alcohol solutions. II [Text] / V.E. Chechko, V.Ya. Gotsulskiys, T.V. Diieva // Ukrainian journal of physics, – 2019. – Vol. 64, № 2. – С. 143–150.

24. Shapovalova S. Increasing the share of correct clustering of characteristic signal with random losses in self–organizing maps [Text] / S. Shapovalova, Yu. Moskalenko // Восточно–Европейский журнал передовых технологий, – 2019. – № 2(4). – С. 13–21.

25. Melnyk R. Measurement of material surface defect intensity by distributed cumulative histogram and clustering [Text] / R. Melnyk, R. Kvit // Технологический аудит и резервы производства, – 2020. – № 4(2). – С. 36–45.

26. Wesley C.J. Core python applications programming [Text] / Wesley J. Chun. – 3rd ed. – Upper Saddle River, NJ[etc.] : Prentice Hall, 2012. – XXXII, 852 p.

27. Lutz M. Python pocket reference [Text] : Python in your pocket / Mark Lutz. – 5th ed. – Beijing [etc.] : O'Reilly, 2014. – VII, 254 p.

28. Anaconda [Electronic resource]. – Access mode: https://www.anaconda.com/products/individual

29. Kohonen 1.1.2 documentation [Electronic resource]. – Access mode: https://pythonhosted.org/kohonen/index.html

30. Text REtrieval Conference [Electronic resource]. – Access mode: https://trec.nist.gov/

31. Text REtrieval Conference – Data [Electronic resource]. – Access mode: https://trec.nist.gov/data.html

32. Методичні вказівки по економічному обґрунтуванню дипломних проектів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проектування» і7.091501 «Комп'ютерні та інтелектуальнісистеми і мережі »всіх форм навчання для усіх форм навчання [Text] / Укл.: Остапенко В.В., Касьян Е.М., Сердюк Є.М. – Запоріжжя: ЗНТУ, 2009. – 22 с.

33. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [Текст] : ДСТУ Б В.2.5–82:2016. – На заміну ДБН В.2.5–27–2006. – [Чинний від 2017–04–01]. – К. : ДП «УкрНДНЦ», 2016. – 109 с.

34. Жидецький В.Ц. Основи охорони праці: підручник [Текст] / В.Ц. Жидецький. – 5–те вид., доп. – К. : Знання, 2014. – 373 с.

35. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5–те, перероблене й доповнене : затв. М-

вом енергетики та вугільної промисловості України 20.06.14. – [Чинний від 2014–11–20]. – Х. : Міненерговугілля України, 2014. – 793 с.

36. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно–обчислювальних машин [Електронний ресурс]. – Режим доступу : http://mozdocs.kiev.ua/view.php?id=2445.

37. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям [Електронний ресурс]. – Режим доступу: http://zakon.rada.gov.ua/laws/show/z0508–18.

38. ДСН 3.3.6.042–99 Санітарні норми мікроклімату виробничих приміщень. – [Чинний від 1999–12–01] – Харків: Форт, 2000 – 24 с.

39. Голінько, В.І. Основи охорони праці [Текст] / В.І. Голінько. – Дніпропетровськ : Національний гірничий університет, 2010. – 271с.

40. ДБН В.2.5–28:2018. Природне та штучне освітлення. [Текст]. – [Чинний від 2019–03–01]. – К: Мінрегіон України, 2019. – 137 с.

41. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою : ДСТУ Б В.1.1–36:2016 [Текст]. – На заміну НАПБ Б.03.002–2007. – [Чинний від 2017–01–01]. – К. : Мінрегіонбуд України, 2016. – 66 с.

42. ДБН В.1.1–7:2016 Пожежна безпека об'єктів будівництва. Загальні вимоги [Text]. – [Чинний від 2017–01–31] – Київ : Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 47 с.

43. Васійчук В.О. Основи цивільного захисту: Навч. Посібник [Текст] / В.О. Васійчук, В.Є. Гончарук, С.І. Качан, С.М. Мохняк. – Львів, – 2010. – 384 с.

ДОДАТОК А

Технічне завдання

А.1 Вступ

А.1.1 Призначення та область застосування

Розроблений програмний комплекс може використовуватися для ІП на базі текстової інформації. Така система може використовуватися у державних установах на етапах менеджменту та документообігу для підвищення рівня автоматизації.

Програма представляє інтерфейс для зручної роботи із реалізованими функціями.

А.2 Підстави для розробки

Підставою для розробки є завдання на магістерськуроботу.

Існує ряд проблем, які не вдається вирішити на основі інформації тільки про текст та запит. Тому алгоритм ранжирування розширюється за рахунок використання додаткових ознак документів, що визначаються, наприклад, на основі посилальної структури колекції.

Використання таких систем значно підвищує рівень автоматизації.

А.3 Призначення розробки

Розроблений програмний комплекс може використовуватися для ІП під час документообігу.

Розроблений програмний комплекс, має реалізовувати основні функції, а саме:

- отримання та передача пошукового запиту;
- опрацювання пошукового запиту клієнта;
- ранжирування відповідей на запит;
- виведення результатів;
- надання та забезпечення GUI.

А.4 Вимоги до програми

А.4.1 Вимоги до функціональних характеристик

ПЗ повинне забезпечувати виконання нижче перерахованих функцій:

– отримання та передача пошукового запиту;

– опрацювання пошукового запиту клієнта;

– ранжирування відповідей на запит;

– виведення результатів;

– надання та забезпечення GUI.

А.4.2 Вимоги до надійності

Для забезпечення надійності програми, вона має базуватися на комп’ютері. На протязі усього сеансу роботи додаток не повинен збоїти або зависати. Після оновлення інформаційної бази або функціоналу, додаток треба перезапустити.

А.4.3 Вимоги до технічних та програмних засобів

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

– наявність IBM-сумісної ПЕОМ, на якій буде працювати додаток;

– процесор із тактовою частотою 2,0 GHz та більше;

– оперативна пам'ять 2,0 ГБ та більше;

– вільний простір на жорсткому диску не менше, ніж 2 ГБ (для збереження бази відповідей на запити).

Вимоги до програмної частини необхідно встановити дистрибутив Python.

А.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

– технічне завдання;

– опис проектування;

– опис тестування;

– опис програми.

А.6 Стадії і етапи розробки

А.6.1 Стадії розробки

Розробка повинна бути проведена в три стадії:

– розробка технічного завдання;

– робоче проектування;

– впровадження.

А.6.2 Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, погодження та затвердження справжнього технічного завдання.

На стадії робочого проектування повинні бути виконані перераховані нижче етапи робіт:

– розробка програми;

– розробка програмної документації;

– випробування програми.

На стадії впровадження повинен бути виконаний етап розробки підготовка і передача програми.

А.6.3 Зміст робіт по етапах

На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи:

– постановка задачі;

– визначення і уточнення вимог до технічних засобів;

– визначення вимог до програми;

– визначення стадій, етапів і термінів розробки програми та документації на неї;

– узгодження і затвердження технічного завдання.

На етапі розробки програми повинна бути виконана робота з програмування (кодування) і налагодженні програми.

На етапі розробки програмної документації повинна бути виконана розробка програмних документів відповідно до вимог до складу документації.

На етапі випробувань програми повинні бути виконані перераховані нижче види робіт:

– розробка, узгодження та затвердження та методики випробувань;

– проведення приймально-здавальних випробувань;

– коректування програми і програмної документації за результатами випробувань.

ДОДАТОК Б

Опис програми

Б.1 Загальні свідчення

ПЗ розроблено за допомогою мови програмування Python, середовища розробки Anaconda та бібліотеки kohonenv.1.12. Тобто реалізації бізнес логіки програми використовується мова програмування Python.

Б.2 Функціональне призначення

Розроблений програмний комплекс може використовуватися для ІП при документообігу.

Розроблений програмний комплекс, реалізує основні функції, а саме:

– отримання та передача пошукового запиту;
– опрацювання пошукового запиту клієнта;
– ранжирування відповідей на запит;
– виведення результатів;
– надання та забезпечення GUI.

Б.3 Опис логічної структури

На структурній схемі (рис. Б.1) основні програмні модулі, що були виділені при проектуванні системи.

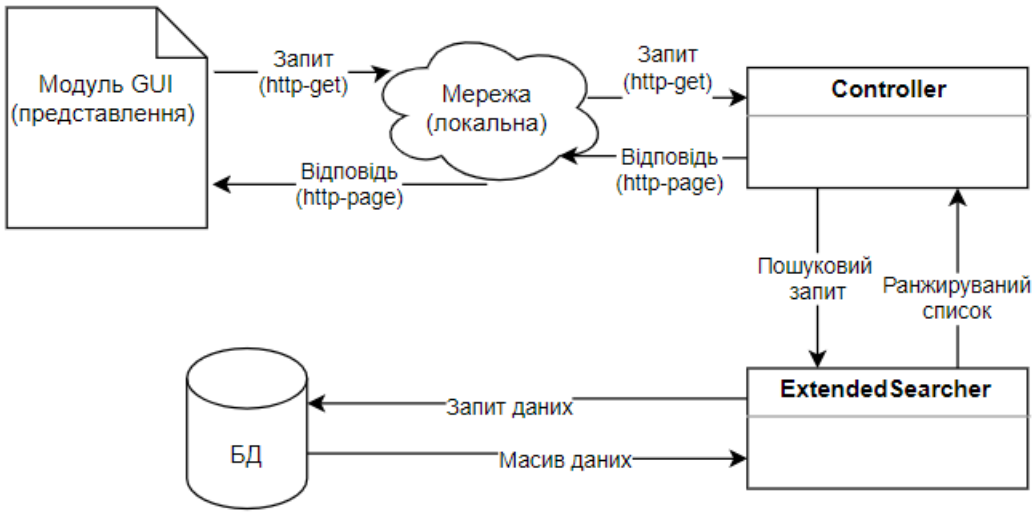

Рисунок Б.1 – Структурна схема проекту

Розробку представлених модулів можна аргументувати тим, що саме розділення основного функціоналу програмного комплексу за таким принципом гарантує формування функціонально завершених блоків програми.

Controller розбиває запит на терміни, формує пошукову команду. Іншими словами, це модуль, що містить список термінів і параметри пошуку. Модуль містить основний робочий клас. Робота зводиться до опрацювання запиту від користувача, що отримується через мережу. Також, клас надає користувачу відповідь у вигляді відсортованого списку результатів.

Модуль ExtendedSearcher отримує від БД, що задовольняють запиту. ExtendedSearcher відправляє ранжируваний список повідомлень. Клас Koh, який відповідає за тренування мережі Кохонена підтягує бібліотеку «kohonenv.1.1.2». Ця бібліотека відповідає за навчання мережі на основі даних.

Клас Clustвідповідає за попередню кластеризіцію. Використовуючи множину векторів з простору векторів розділяє на *k* кластерів. Для цього знаходиться безліч кластерних центрів, що мінімізують функцію міру відстані між векторами.

Клас InfArrвідовідає за завантаження масиву несортованої (без ранжирування) інформації із бази даних.

Б.4 Використані технічні засоби

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

– наявність IBM-сумісної ПЕОМ, на якій буде працювати додаток;

– процесор із тактовою частотою 2,0 GHz та більше;

– оперативна пам'ять 2,0 ГБ та більше;

– вільний простір на жорсткому диску не менше, ніж 2 ГБ (для збереження бази відповідей на запити).

Вимоги до програмної частини необхідно встановити дистрибутив Python.

ДОДАТОК В

Текст програми

```
import numpy as np
import pandas as pd
import math
import pickle
import random

trainl = 70
testl=25
prevWin = [-1, 0]
epoch = 0
D0 = 10
a0 = 0.9

Claster = np.zeros((trainl,2))
X = np.array(pd.read_csv('mnist_train.csv', nrows=trainl))
test_num = np.array(pd.read_csv('mnist_train.csv',
nrows=trainl))
#W = np.random.uniform(0.0, 0.3, size=(784, 10) )
W = np.zeros((784,10))
W = W.astype('float64')
X = X.astype('float64')


for i in range(trainl):
    for j in range(784):
        X[i][j] = X[i][j]/254
```

```
def vdist(V, W):
    S = 0
    for i in range(len(V)):
        S += (V[i] - W[i]) ** 2
    return math.sqrt(S)

while epoch < 25:
    Sum = 0
    epoch += 1
    for a in range(trainl):
        Dis = list()
        for j in range(10):
            d = 0
            for i in range(784):
                d += (X[a][i]-W[i][j])**2
            Dis.append(math.sqrt(d))
        #print(Dis)
        win = Dis.index(min(Dis))
        if prevWin[0] == win:
            if prevWin[1] > 3:
                win = random.randint(0, 9)
                prevWin = [win, 1]
            else:
                prevWin[1] += 1
        else:
            prevWin = [win, 1]
        DN = round(D0)
```

```
            for j in range(10):
                if vdist(W[:,j], W[:,win]) > DN and j != win:
                    continue
                for i in range(784):
                    W[i, j] += a0 * (X[a][i] - W[i, j])
                    Sum += abs(a0 * (X[a][i] - W[i, j]))
        if a0 - 0.05 >= 0.1:
            a0 -= 0.05
        if  D0 - 0.5   >= 0:
            D0 -= 0.5

    for a in range(trainl):
        index = test_num[a][0]
        Dis = list()
        for j in range(10):
            d = 0
            for i in range(784):
                d += (X[a][i] - W[i, j]) ** 2
            Dis.append(math.sqrt(d))
        win = Dis.index(min(Dis))
        Claster[a][0] = win
        Claster[a][1] = index

            class NetWork(self. n,self. m)

                Elem=new List<Element>()
                Count = m
                Random rnd = new Random()
```

```
    Nrs = new Neuron[Count]
    for (i = 0 i < Count i++)
        Nrs[i] = new Neuron(n, rnd)


def Add(Element x)

    Elem.Add(x)


def MinDist(Element x)

    self.min = self..MaxValue
    self.mini=0
    for (i = 0 i < Count i++)

        self. sss = Nrs[i].GetDist(x)
        if (min > Nrs[i].GetDist(x))

            mini = i
            min = Nrs[i].GetDist(x)


    return mini

self. Hauss(self. t, self. i, self. j)

    self. d = Nrs[i].GetDist(Nrs[j].Weight)
```

```
            self. zn = Math.Exp(-t)
            return Math.Exp(-d / zn)

        def Teacher()

            self.DeltaW = 1
            self.DeltaW1= 1
            self.DW = 1
            self.Epoch = -1
            while (Math.Abs(DW) > 0.010)

                Epoch++
                DW = 0
                DeltaW = 0
                for (i = 0 i < Elem.Count i++)

                    self. k = MinDist(Elem[i])

                    for (int j = 0 j < Count j++)

                        self. Hau = Hauss(Epoch, k, j)
                        DeltaW +=
Math.Abs(Nrs[j].ChangeWeight(Hau, Elem[i].Condition))



                DW = DeltaW - DeltaW1
                DeltaW1 = DeltaW
class Neuron
```

```
        def Weight
        self.OldWeight

        self.Count
        public Neuron(self. n, Random rnd)

            Count = n
            Weight = new self.n
            OldWeight = new self.n
            for (i = 0 i < Count i++)

                Weight[i] = rnd.Nextself.()
                OldWeight[i] = 0


        def GetDist(Element nr)

            return GetDist(nr.Condition)

        def GetDist(self.Cond)

            self. sum = 0
            for (i = 0 i < Count i++)
                sum += (Cond[i] - Weight[i]) * (Cond[i] -
Weight[i])
            return Math.Sqrt(sum)
```

```
        def ChangeWeight(self. Alpha, self.[] y)

            self.res = 0
            for (i = 0 i < Count i++)

                res +=0.1*Alpha * (y[i] - Weight[i])
                Weight[i] = Weight[i] + 0.1 * Alpha * (y[i] -
Weight[i])

            return res

        def Training()

            self.l = 0
            while (true)

                if (l++ > 10000)

                    // MessageBox.Show("========")
                    return

                int k = 0
                for (i = 0 i < Lst.Count i++)

                    self.y = GetSum(Lst[i])
                    if (y > Threshold) y = 0
                    else y = 1
                    if (y != Lst[i].Result)
```

```
                        Threshold += Speed * (Lst[i].Result -
y)
                        for (int m = 0 m < Size.X m++)
                            for (int n = 0 n < Size.Y n++)

                                Weight[m, n] += Speed *
(Lst[i].Result - y) * Lst[i].Condition[m, n]

                        k = 1



                if (k == 0) break



        def CheckImage(self.[,] img)

            self.y = GetSum(img)
            if (y > Threshold) return true
            return false
def Initial(self,
                 dimension=None,
                 shape=None,
                 metric=None,
                 learning_r=None,
                 neighb_s=None,
                 nois_var=None):
        assert dimension is not None
        self.dimension = dimension
```

```
assert shape is not None
self.shape = shape

self.metric = metric or euclidean_metric

ET = exponen_TS
CT = constan_TS

self.learning_r = learning_r
if isinstance(learning_r, (float, int)):
    self.learning_r = CT(learning_r)
if learning_r is None:
    self.learning_r = ET(-1e-3, 1, 0.2)

self.neighb_s = neighb_s
if isinstance(neighb_s, (float, int)):
    self.neighb_s = CT(neighb_s)
if neighb_s is None:
    self.neighb_s = ET(-1e-3, max(shape), 1)

self.nois_var = nois_var
if isinstance(nois_var, (float, int)):
    self.nois_var = CT(nois_var)
```

ДОДАТОК Г

Слайди презентації

Рисунок Г.1 – Слайд №1

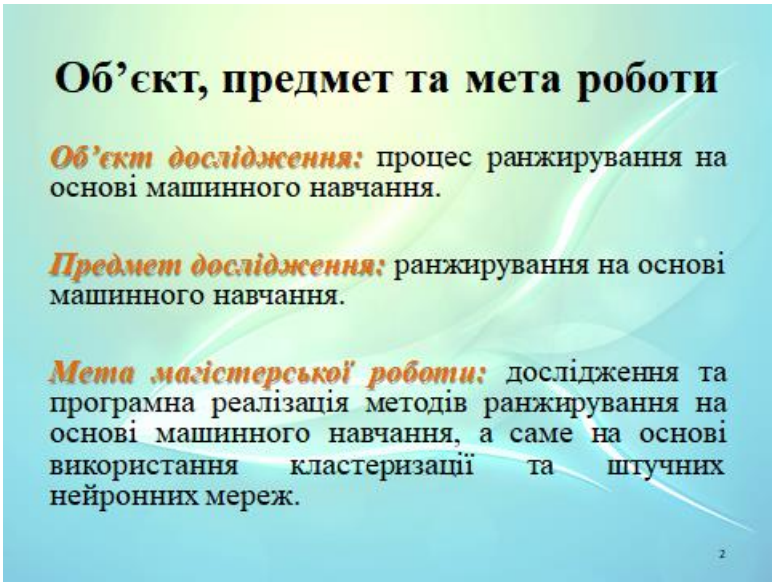

Рисунок Г.2 – Слайд №2

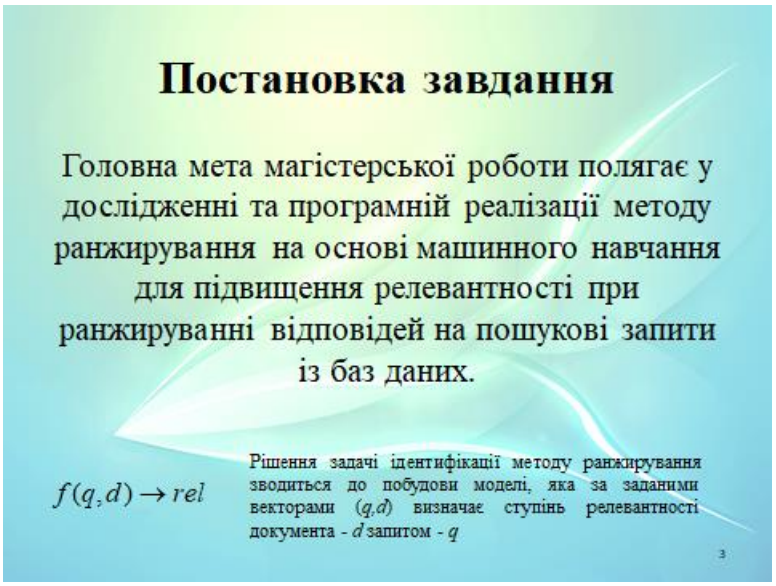

Рисунок Г.3 – Слайд №3

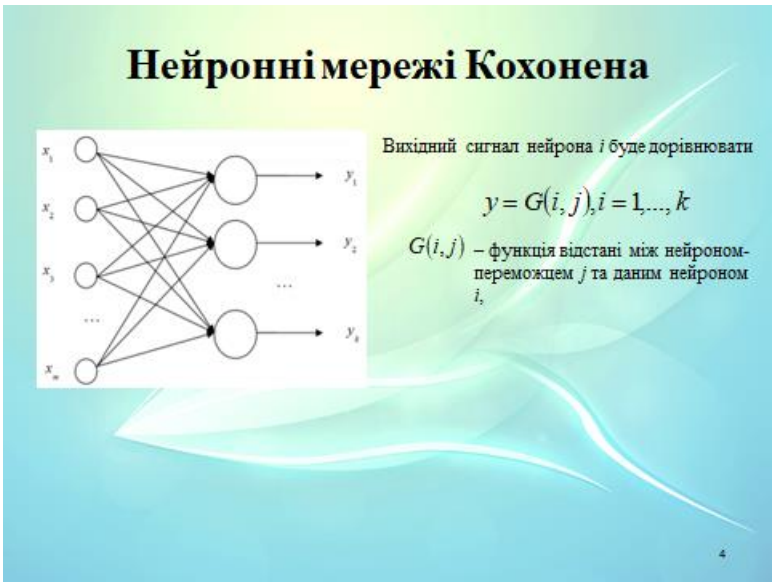

Рисунок Г.4 – Слайд №4

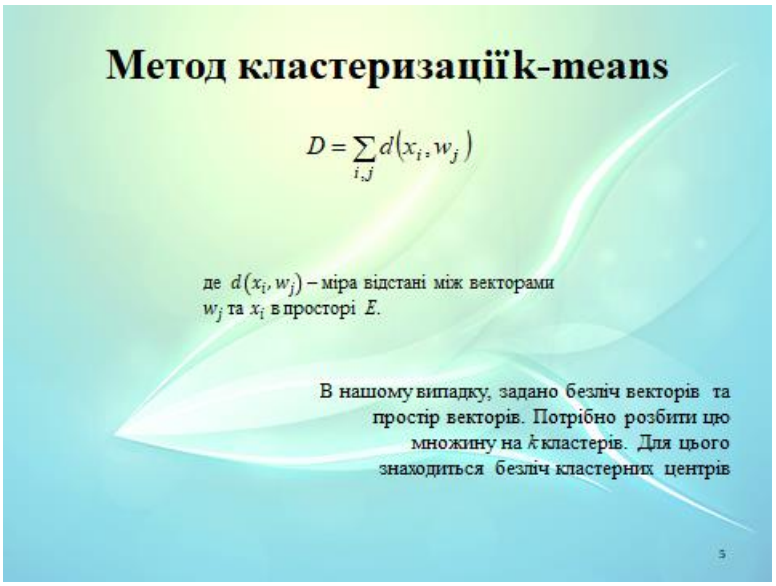

Рисунок Г.5 – Слайд №5

Рисунок Г.6 – Слайд №6

Рисунок Г.7 – Слайд №7

Рисунок Г.8 – Слайд №8

Рисунок Г.9 – Слайд №9

Рисунок Г.10 – Слайд №10

Рисунок Г.11 – Слайд №11

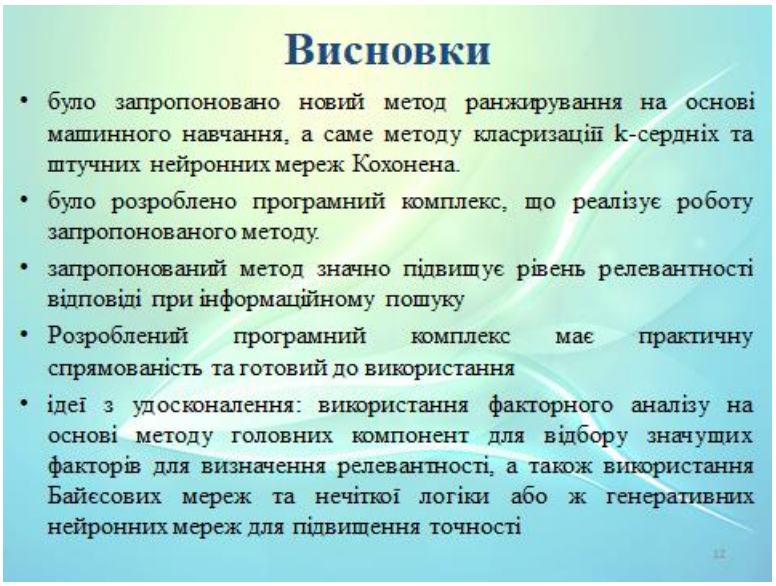

Рисунок Г.12 – Слайд №12

Рисунок Г.13 – Слайд №13