Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмно-алгоритмічні засоби формування шаблонів опрацювання частково структурованих файлів даних

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Гашинський Р.І.
(підпис) (прізвище та ініціали)

Керівник _____ Кравчук Г.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_ __ _» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Гашинському Роману Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-алгоритмічні засоби формування шаблонів опрацювання частково структурованих файлів даних

Керівник роботи Кравчук Галина Богданівна, асистент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-470*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Наукові публікації про автоматизований аналіз схеми документів з частково-структурованими даними

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання до програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних. 2. Проєктна частина розробки програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих даних. 3. Практична частина розробки програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Мета, Цінність. 3. Постановка проблеми. 4. Шаблони опрацювання. 5. Внутрішня схема даних. 6. Проєктна частина. Рішення. 7. Практична частина. Кроки. 8. Практична частина. Зчитування заголовків. Порівняння їх з внутрішньою схемою даних. 9. Практична частина. Виявлення коефіцієнту наповненості. 10. Практична частина. Визначення відповідності регулярним виразам. 11. Практична частина. Оцінка унікальності значень. 12. Практична частина. Результати тренування нейронної мережі для класифікації стовпців вхідного файлу. 13. Практична частина. Аналіз результатів роботи нейронної мережі. 14. Практична частина. Аналіз результатів роботи нейронної мережі. 15. Практична частина. Формування шаблонів опрацювання. 16. Цільові системи. 17. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В. С., кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про сучасні методи обробки частково структурованих даних	31.01.2024-03.02.2024	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	*Виконано*
4.	Виконання дослідження щодо існуючих інструментів та технологій для обробки частково структурованих файлів	07.02.2024-11.02.2024	*Виконано*
	Розроблення програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних		
5.	Оформлення розділу «Аналіз предметної області та постановка завдання до програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних	03.06.2024-05.06.2024	*Виконано*
	»		
6.	Оформлення розділу «Проєктна частина розробки програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих даних	06.06.2024-08.06.2024	*Виконано*
	»		
7.	Оформлення розділу «Практична частина розробки програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних	09.06.2024-11.06.2024	*Виконано*
	»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи охорони		
	праці»		
10.	Оформлення кваліфікаційної роботи		
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи		

Студент _____ Гашинський Р.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кравчук Г.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмно-алгоритмічні засоби формування шаблонів опрацювання частково структурованих файлів даних // Кваліфікаційна робота освітнього рівня «Бакалавр» // Гашинський Роман Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2024 // С. 59, рис. – 16, табл. – 4, слайд. – 19, додат. – 2 , бібліогр. – 42.

Ключові слова: дані, метадані, аналіз, опрацювання, менеджмент даних, контейнеризація, машинне навчання.

Кваліфікаційна робота присвячена розробці програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних.

В першому розділі кваліфікаційної роботи описано актуальність теми та основні цілі дослідження. Висвітлено основні методи та підходи до обробки частково структурованих даних. Розглянуто сучасні інструменти та технології для автоматизації обробки даних. Проаналізовано вимоги до програмно-алгоритмічних засобів.

В другому розділі кваліфікаційної роботи наведено проєктування архітектури програмно-алгоритмічних засобів. Досліджено алгоритми для автоматизованого формування шаблонів обробки даних.

В третьому розділі кваліфікаційної роботи описано реалізацію програмно-алгоритмічних засобів для автоматизації обробки даних. Проаналізовано результати тестування та ефективність розроблених засобів. Проведено аналіз точності та надійності роботи програмно-алгоритмічних засобів.

Об'єкт дослідження: програмно-алгоритмічні засоби формування шаблонів обробки даних.

Предмет дослідження: процес обробки частково структурованих файлів даних.

ANNOTATION

Software and Algorithmic Means of Templates Forming for Processing of Semi-Structured Data Files // Qualification work of the educational level «Bachelor» // Roman Gashynskyi // Ternopil Ivan Puluj National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2024 // P. 59, fig. – 16 , tabl. – 4, slides – 19, annexes. – 2, references – 42.

Keywords: data, metadata, analysis, processing, data management, containerization, machine learning.

The qualification work is dedicated to the development of software and algorithmic means of templates forming for processing of semi-structured data files. The goal of the work is to ensure effective and accurate processing of data with minimal manual intervention.

The first section of the qualification paper considers the relevance of the topic and the main objectives of the research. It highlights the primary methods and approaches for processing partially structured data. It reviews modern tools and technologies for data processing automation. The requirements for the software and algorithmic means are analyzed.

In the second section of the qualification work, the architecture of the software and algorithmic means is considered. The algorithms for automatic template creation for data processing are investigated. A detailed description of the containerization infrastructure using Docker is provided.

In the third section of the qualification work, the implementation of the software and algorithmic means for data processing automation is described. The results of testing and the efficiency of the developed tools are analyzed. A detailed analysis of the accuracy and reliability of the software-algorithmic tools is conducted.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programing Interface) – прикладний програмний інтерфейс.

Docker – платформа для контейнеризації, яка дозволяє ізольовано запускати додатки з усіма необхідними залежностями.

Dropout – техніка регуляризації, що випадково відключає нейрони під час тренування нейронної мережі, щоб запобігти перенавчанню.

HTTP (англ. HyperText Transfer Protocol) – протокол передачі гіпертексту.

FastAPI – сучасний, високопродуктивний веб-фреймворк для створення API за допомогою Python.

JSON (англ. JavaScript Object Notation) – нотація об’єкту JavaScript.

REST (англ. Representational State Transfer) – передача репрезентативного стану.

Swagger – інструмент для документування та тестування API.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДО ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ФОРМУВАННЯ ШАБЛОНІВ ОПРАЦЮВАННЯ ЧАСТКОВО СТРУКТУРОВАНИХ ФАЙЛІВ ДАНИХ 10
1.1 Аналіз предметної області програмно-алгоритмічний засобів 10
1.2 Формування списку обмежень до програмно-алгоритмічних засобів 13
1.3 Актанти та варіанти використання програмно-алгоритмічних засобів 15
1.4 Визначення переліку вимог до програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних 16
1.5 Висновок до першого розділу 17
РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА РОЗРОБКИ ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ФОРМУВАННЯ ШАБЛОНІВ ОПРАЦЮВАННЯ ЧАСТКОВО СТРУКТУРОВАНИХ ДАНИХ 19
2.1 Проєктування структури потоків даних для програмно-алгоритмічних засобів……19
2.2 Проєктування отримання файлів даних програмно-алгоритмічними засобами 20
2.3 Проєктування аналізу файлів даних програмно-алгоритмічними засобами 23
2.4 Проєктування класифікації стовпців програмно-алгоритмічними засобами 27
2.5 Проєктування формування шаблону опрацювання частково структурованого файлу даних 30
2.6 Проєктування інфраструктури контейнеризації для програмно-алгоритмічних засобів 31

2.7 Висновок до другого розділу 33
РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА РОЗРОБКИ ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ФОРМУВАННЯ ШАБЛОНІВ ОПРАЦЮВАННЯ ЧАСТКОВО СТРУКТУРОВАНИХ ФАЙЛІВ ДАНИХ ... 35
3.1 Реалізація механізму отримання даних в програмно-алгоритмічних засобах……35
3.2 Реалізація аналізу файлу програмно-алгоритмічними засобами 37
3.3 Реалізація класифікації стовпців програмно-алгоритмічними засобами ……38
3.4 Реалізація генерації шаблону опрацювання частково структурованого файлу даних 41
3.5 Реалізація інфраструктури контейнеризації для програмно-алгоритмічних засобів 43
3.6 Аналіз результатів роботи програмно-алгоритмічних засобів 44
3.7 Висновок до третього розділу 46
РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 47
4.1 Принципи ергономіки робочого місця..... 47
4.2 Правила електробезпеки при роботі з електронними пристроями.. 51
ВИСНОВКИ..... 54
ПЕРЕЛІК ДЖЕРЕЛ..... 55
ДОДАТКИ

ВСТУП

Актуальність теми. У сучасному інформаційному суспільстві, де дані швидко збільшуються в об’ємі та стають різноманітнішими, забезпечення високої якості даних є важливим для забезпечення міцної основи для управлінських і бізнес-рішень.

Більшість організацій працюють з даними, які можуть бути як бізнес-об'єктом, так і результатом діяльності самих організацій. Якщо організація не може довіряти даним для задоволення потреб бізнесу, усі зусилля, витрачені на збір, зберігання, захист і забезпечення доступу, будуть витрачені даремно [1].

Частково структуровані дані є поширеним форматом. Вони часто мають різні схеми даних, що ускладнює їх автоматизовану обробку, для якої необхідні шаблони опрацювання, які зазвичай складаються вручну, що сповільнює інтеграцію нових джерел і створює ризики, пов’язані з людським фактором. Тому розробка програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних є актуальним напрямком для дослідження.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- Проаналізувати стан досліджень у галузі обробки частково структурованих даних.
- Сформувати список вимог до програмно-алгоритмічних засобів.
- Спроєктувати архітектуру та описати основні компоненти.
- Реалізувати програмно-алгоритмічні засоби.
- Оцінити ефективність на прикладах реальних даних.

Виконання зазначених завдань сприятиме створенню ефективних інструментів для автоматизованої обробки частково структурованих даних, що

зменшить ризики, пов'язані з людським фактором, і прискорить інтеграцію нових джерел даних.

Практичне значення одержаних результатів. Практичне значення одержаних результатів полягає у створенні програмно-алгоритмічних засобів, які дозволяють автоматизувати процес формування шаблонів опрацювання частково структурованих файлів даних. Це забезпечить підвищення ефективності роботи з даними. Розроблені програмно-алгоритмічні засоби можуть бути використані для автоматизації рутинних задач обробки даних, зменшуючи час та ресурси, необхідні для їх виконання, та підвищуючи точність і надійність результатів аналізу.

Публікації. Основні результати кваліфікаційної роботи опубліковано у тезах. У співавторстві, здобувачу освітнього рівня бакалавр належить: [2] – автоматизований аналіз схеми документів з частково-структурованими даними (див. додаток А).

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДО ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ФОРМУВАННЯ ШАБЛОНІВ ОПРАЦЮВАННЯ ЧАСТКОВО СТРУКТУРОВАНИХ ФАЙЛІВ ДАНИХ

1.1 Аналіз предметної області програмно-алгоритмічних засобів

Частково-структуровані дані – це дані, які поєднують в собі елементи як структурованих, так і неструктурованих даних. Традиційні методи опрацювання даних, що орієнтовані на чітко структуровані формати, не завжди можуть бути ефективно застосовані до частково структурованих. Опрацювання частково структурованих даних є складною задачею через варіативність їх схем даних. Особливо важливо це при інтеграції даних з різних джерел даних. Дану проблему зображено на рисунку 1.1.

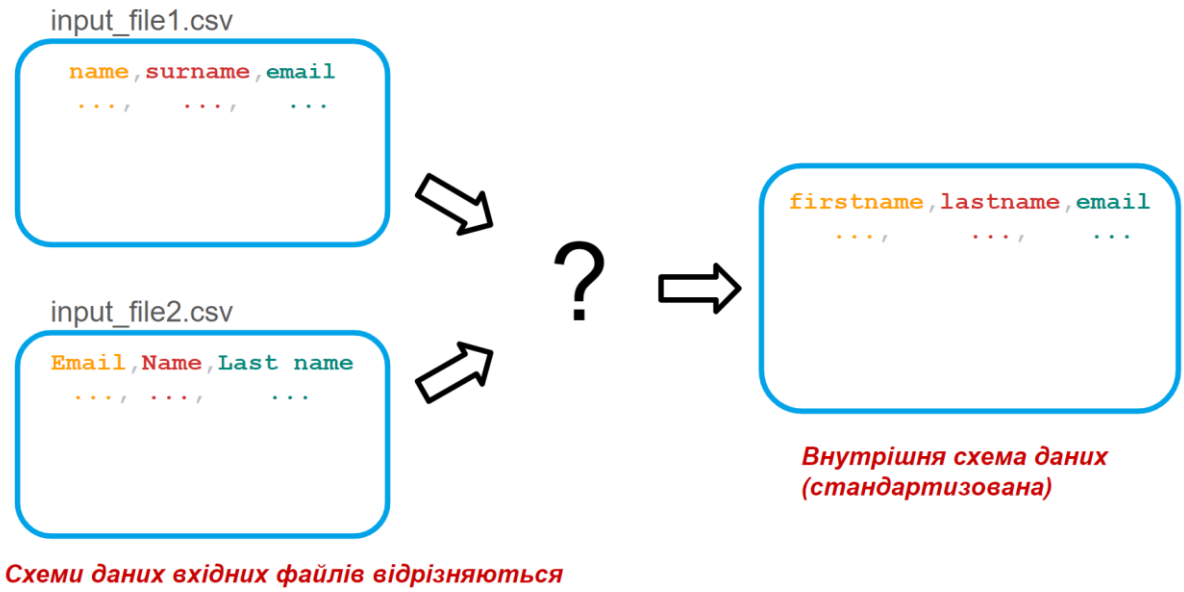


Рисунок 1.1 – Обробка частково структурованих даних з різних джерел

При опрацюванні файлів є необхідною можливість опрацьовувати файли з різною схемою даних, використовуючи один механізм обробки. Для процесу обробки йому необхідно надати інформацію, яка дозволяє працювати з

конкретною схемою даних. Найлегшим і найочевиднішим підходом є використання шаблонів імпорту, приклад застосування яких зображено на рисунку 1.2.

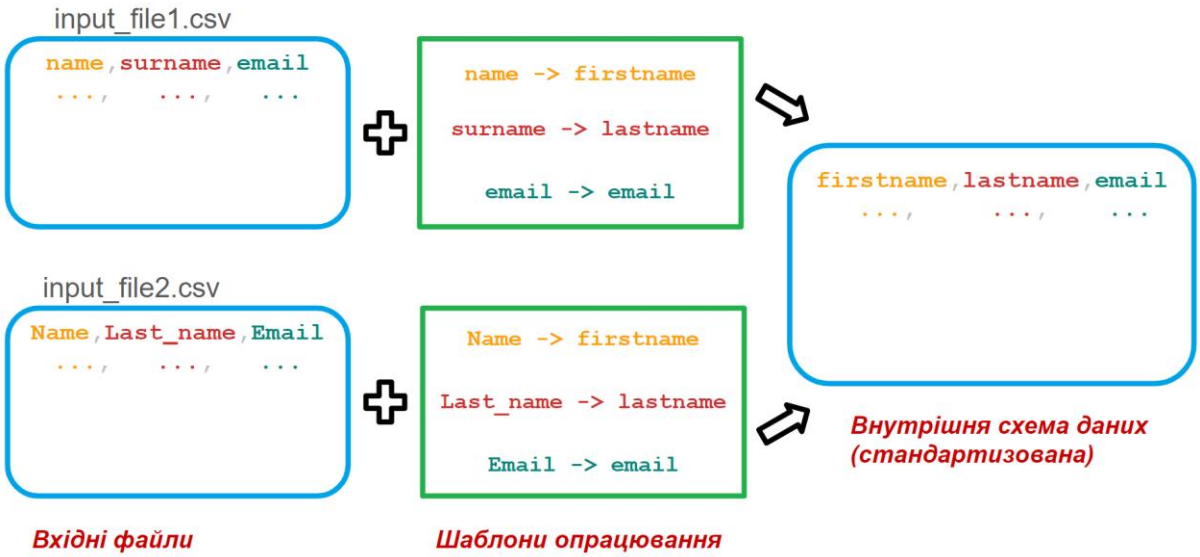


Рисунок 1.2 – Використання шаблонів імпорту для опрацювання різних схем даних

Такі шаблони дозволяють вказувати, як саме слід опрацьовувати файли для приведення їх схеми даних до внутрішньої схеми. Існують рішення, які можуть опрацьовувати схему даних при наданні шаблону імпорту, що дозволяє уникнути ручної роботи для процесу опрацювання і прискорити процес інтеграції нових джерел. Проте необхідність створювати шаблони вручну створює вразливості до людського фактору, включаючи помилки при визначенні схеми даних та інші дрібні неточності, що можуть вплинути на якість обробки даних. Ці проблеми стають особливо критичними при роботі з великими обсягами даних або при частій зміні схеми даних вхідних файлів, що потребує постійного оновлення шаблонів.

Вирішити ці нові проблеми може автоматизоване формування шаблонів обробки на основі схеми даних вхідного файлу. Такий підхід дозволяє автоматизовано визначати її без значного втручання людини. Це значно знижує ризик помилок та забезпечує високу точність і узгодженість обробки даних.

Процес автоматизованого формування шаблонів опрацювання, зображений на рисунку 1.3, дозволяє швидко адаптуватися до змін у схемі даних, покращуючи гнучкість та ефективність інтеграції нових джерел.

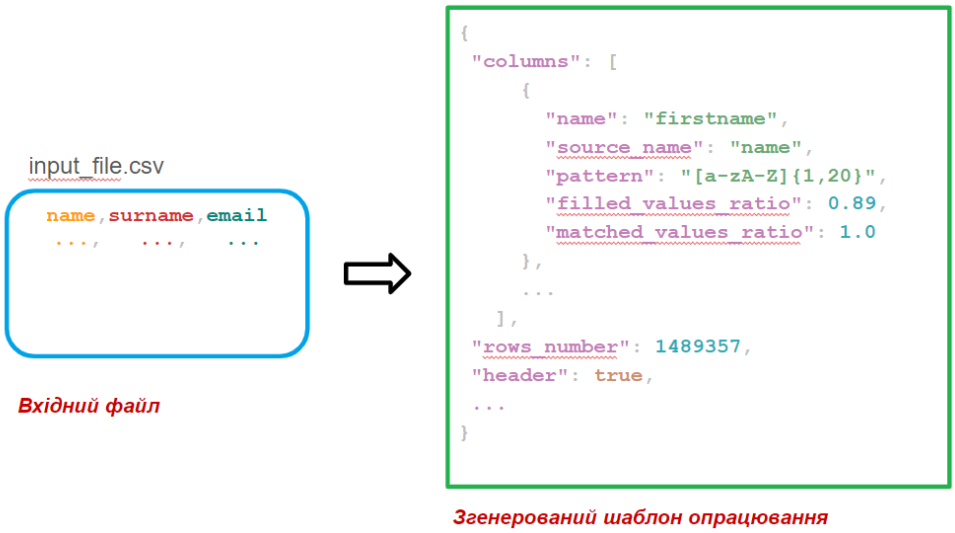


Рисунок 1.3 – Автоматизоване формування шаблонів обробки частково-структурованих файлів даних

Для реалізації цього зазвичай використовують різні підходи, найпоширеніші з яких описані в таблиці 1.1.

Таблиця 1.1 – Підходи до автоматизованого аналізу схеми частково-структурованих файлів даних

Підхід	**Переваги**	**Недоліки**
Регулярні вирази	Простота використання, швидкість	Обмежена гнучкість
Парсери та аналізатори	Висока точність для чітко структурованих частин даних	Потреба у чіткій специфікації внутрішньої схеми, обмежена гнучкість
Машинне навчання	Адаптивність, висока точність	Потреба у великих обсягах навчальних даних, обчислювальна складність

Запропоновані програмно-алгоритмічні засоби об’єднують ці підходи. Парсери і регулярні вирази дозволяють дізнатися аналітику про вміст файлу, а алгоритм машинного навчання на її основі робить висновки про схему даних.

Переваги:

- простота використання;
- швидкість;
- висока точність і гнучкість;
- малі обсяги навчальних даних.

Недоліки:

- потреба у чіткій специфікації внутрішньої схеми даних.

Отже, об’єднання методів дозволить вдало використати їх переваги і мінімізувати недоліки, а, враховуючи, що більшість процесів опрацювання даних вимагають саме приведення даних до чіткої внутрішньої схеми, головний недолік не є критичним в межах даної предметної області.

1.2 Формування списку обмежень до програмно-алгоритмічних засобів

У даній кваліфікаційній роботі освітнього рівня «Бакалавр» програмно-алгоритмічні засоби, які запропоновані для використання в стандартизації даних, мають певні обмеження.

По-перше, для вхідних даних доступний лише формат CSV (англ. comma-separated values – значення, розділені комами), оскільки він є найпоширенішим форматом частково структурованих даних в процесах обробки даних.

По-друге, як зазначено в підрозділі 1.1, програмно-алгоритмічні засоби вимагають чітку специфікацію внутрішньої схеми даних, що створює обмеження на набір вхідних даних, які аналізуються. Це означає, що програмно-алгоритмічні засоби будуть намагатися будь-які стовпці вхідних даних визначити та класифікувати до стовпців внутрішньої схеми даних.

Для даної роботи використано внутрішню схему даних, описану в таблиці 1.2.

Таблиця 1.2 – Внутрішня схема даних

Назва стовпця	**Шаблони назв**	**Регулярний вираз для валідації даних**
email	email, email address	.*@.*
firstname	firstname, name, fname, given_name	^[A-Za-z]{1,20}$
lastname	lastname, lname, surname	^[A-Za-z]{1,20}$
ip	ip, ip_address	^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$
country	country, country_name	^[A-Z]{2}$
phone_number	phone_number, phone, mobile, cell	^\+?[1-9]\d{1,14}$
zip	zip, zip_code, postal_code	^\d{5}$

Важливо зауважити, що шаблони назв, вказані у внутрішній схемі, не є єдиними варіантами назв стовпців, а лише пропозиціями, з якими буде здійснюватися порівняння, якщо такі будуть знайдені в файлі.

Також у внутрішньої схеми є свій ряд визначених обмежень, які впливали на вибір регулярних виразів:

- Імена і прізвища розраховані на англомовні варіанти.
- IP-адреси розраховані лише на формат IPv4.
- Для позначення країни використаний формат дволітерного коду країни, відповідно до стандарту ISO 3166-1 alpha-2 [3].
- Номери телефону відповідають міжнародному формату, відповідно до рекомендації E.123 [4].
- Для поштових індексів використаний п’ятицифровий формат.

Ці обмеження визначають специфіку та області застосування програмно-алгоритмічних засобів.

1.3 Актанти та варіанти використання програмно-алгоритмічних засобів

Як було визначено в підрозділі 1.1, цільовими системами запропонованих програмно-алгоритмічних засобів є процеси опрацювання даних, тому основним актором є оператор процесу опрацювання даних. До системних компонент програмно-алгоритмічних засобів належать:

- Інтерфейс користувача – забезпечує зручний доступ до функцій системи, включаючи завантаження файлів та перегляд результатів.
- Система зберігання – зберігає завантажені файли для обробника файлів.
- Обробник файлів – виконує аналіз та класифікацію даних, формування шаблонів.

На рисунку 1.4 зображено варіанти використання актора «Оператор процесу опрацювання даних».

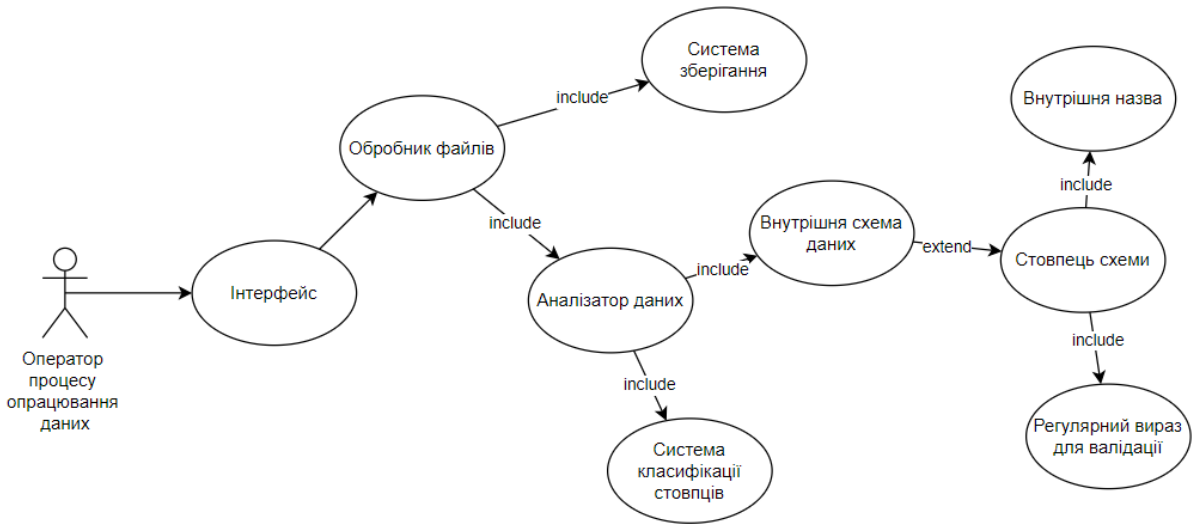


Рисунок 1.4 – Діаграма варіантів використання актора «Оператор процесу опрацювання даних»

Оператор процесу опрацювання даних взаємодіє з інтерфейсом користувача для завантаження файлів даних. Завантажені файли передаються до обробника файлів, який зберігає їх у системі зберігання та передає аналізатору

даних для обробки. Аналізатор даних виконує аналіз, включаючи визначення схеми даних за допомогою системи класифікації відповідно до внутрішньої схеми даних.

1.4 Визначення переліку вимог до програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних

Для розробки програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних важливо визначити перелік функціональних та нефункціональних вимог до них [5].

Функціональні вимоги описують конкретні функції та можливості, які повинні бути реалізовані у програмно-алгоритмічних засобах. Основні функціональні вимоги включають:

- Аналіз схеми даних файлів – програмно-алгоритмічні засоби повинні аналізувати вхідні файли даних для визначення їх схеми даних.
- Валідація даних – програмно-алгоритмічні засоби повинні перевіряти дані на відповідність заданим регулярним виразам для кожного типу стовпця.
- Формування шаблонів – програмно-алгоритмічні засоби повинні автоматизовано створювати шаблони опрацювання частково структурованих файлів даних на основі аналізу їх схеми даних.
- Інтерфейс для завантаження файлів – програмно-алгоритмічні засоби повинні мати зручний інтерфейс для завантаження файлів користувачами.

Нефункціональні вимоги описують характеристики програмно-алгоритмічних засобів, які не пов'язані безпосередньо з функціональністю, але є важливими для їх ефективної роботи. Основні нефункціональні вимоги включають:

- Продуктивність – програмно-алгоритмічні засоби повинні обробляти файли даних швидко та ефективно, забезпечуючи мінімальний час відгуку.

- Надійність – програмно-алгоритмічні засоби повинні бути стійкими до помилок та забезпечувати коректну обробку навіть у випадку некоректних вхідних даних.
- Легкість розгортання – програмно-алгоритмічні засоби повинні передбачати легкий запуск незалежно від середовища.
- Масштабованість – програмно-алгоритмічні засоби повинні мати можливість обробляти великі обсяги даних без значного зниження продуктивності.
- Зручність використання – інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні, забезпечуючи легкий доступ до основних функцій.

Запропоновані функціональні і нефункціональні вимоги є основою для розробки програмно-алгоритмічних засобів, які забезпечать ефективне і надійне формування шаблонів опрацювання частково структурованих файлів даних. Чітке визначення вимог дозволяє створити засоби, що відповідають потребам користувачів, забезпечуючи високу продуктивність, масштабованість та зручність використання.

1.5 Висновок до першого розділу

У першому розділі кваліфікаційної роботи «Програмно-алгоритмічні засоби формування шаблонів опрацювання частково структурованих файлів даних» було проведено аналіз предметної області та постановку завдання для розробки програмно-алгоритмічних засобів обробки частково структурованих файлів даних. Зокрема, було розглянуто актуальність теми, а також проаналізовано існуючі підходи та методи обробки таких даних.

Аналіз предметної області показав, що частково структуровані дані є складними для обробки через варіативність їх схеми даних. Існуючі методи мають свої переваги та недоліки. Було визначено, що для ефективної автоматизації процесу обробки частково структурованих даних необхідно

розробити спеціалізовані програмно-алгоритмічні засоби, які будуть поєднувати підходи для використання їх переваг і мінімізації недоліків.

Також було сформульовано вимоги до цих засобів, які включають як функціональні, так і нефункціональні аспекти. Функціональні вимоги охоплюють можливості імпорту даних, аналізу їх структури, формування шаблонів та забезпечення зручного інтерфейсу користувача. Нефункціональні вимоги включають продуктивність, масштабованість, надійність та легкість розгортання.

Проведений аналіз і визначення вимог створюють міцну основу для подальшої розробки програмно-алгоритмічних засобів, які забезпечать ефективну автоматизацію обробки частково-структурованих даних.

РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА РОЗРОБКИ ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ФОРМУВАННЯ ШАБЛОНІВ ОПРАЦЮВАННЯ ЧАСТКОВО СТРУКТУРОВАНИХ ДАНИХ

2.1 Проєктування структури потоків даних для програмно-алгоритмічних засобів

Проєктування структури потоків даних включає визначення та опис логічних потоків даних, що проходять через систему, від моменту завантаження даних до отримання кінцевих результатів [6].

Основна мета проєктування потоків даних полягає в тому, щоб забезпечити ефективну та надійну обробку частково структурованих даних шляхом оптимізації кожного етапу обробки, використання відповідних алгоритмів та інструментів, а також забезпечення безперебійної взаємодії між компонентами системи.

Потоки даних для програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих даних зображені на рисунку 2.1.

Рисунок 2.1 – Діаграма потоків даних програмно-алгоритмічних засобів

Ця схема наочно демонструє, як дані переміщуються через систему, починаючи з їх завантаження у вихідному форматі, проходячи через різні етапи обробки та аналізу, і закінчуючи генерацією кінцевих результатів та звітів.

Початковою точкою входу для даних є API, що приймає файли від користувачів. Після того, як вхідний файл отриманий, API передає його для

подальшої обробки. Основні функції API – прийом файлів і передача їх для аналізу.

На наступному етапі програмно-алгоритмічні засоби здійснюють аналіз отриманого файлу, при якому збирають основну аналітику про вміст файлу, включно з наповненістю вмісту стовпців.

Після цього проходить етап класифікації стовпців, на якому, використовуючи частину аналітики з попереднього кроку, алгоритм машинного навчання визначає схему даних вхідного файлу відповідно до внутрішньої схеми.

Останній етап – формування шаблону опрацювання даних і повернення його через API, що завершує процес обробки, забезпечуючи користувачам повну інформацію про структуру та якість даних.

2.2 Проєктування отримання файлів даних програмно-алгоритмічними засобами

Першим етапом роботи програмно-алгоритмічних засобів є отримання файлів через інтерфейс. Для забезпечення ефективного та надійного отримання даних було вирішено використовувати REST API [7]. Цей підхід було обрано з кількох причин:

- Стандартизація REST API забезпечує високу сумісність та легку інтеграцію з іншими системами. Завдяки стандартизованому підходу до взаємодії з веб-сервісами, REST API дозволяє забезпечити взаємодію між різними компонентами системи без необхідності в складних налаштуваннях та адаптаціях. Це спрощує інтеграцію з іншими системами та сервісами, що є важливим у контексті сучасних розподілених додатків.
- REST API є простим у використанні та розумінні, що полегшує розробку та підтримку програмного забезпечення. Простота вивчення і застосування REST API дозволяє розробникам швидко освоювати його і зосереджуватися на вирішенні основних завдань, а не на налаштуванні інтерфейсів. Це також знижує вартість навчання нових членів команди.

- REST API гнучкий у реалізації різних операцій, що дозволяє легко адаптувати API до вимог конкретного додатку та забезпечити необхідний функціонал. Це робить REST API універсальним інструментом для побудови взаємодії між компонентами програмної системи.

- REST API підтримує горизонтальне масштабування, що дозволяє обробляти великі обсяги запитів без зниження продуктивності, а також розподіляти навантаження між кількома серверами. Це забезпечує стабільну роботу системи під час пікових навантажень і покращує її надійність. Завдяки цьому, система може бути легко розширена для обробки збільшеного обсягу даних, зберігаючи при цьому високу продуктивність.

Для створення API на Python було обрано FastAPI. Це один з найшвидших веб-фреймворків для побудови API на Python [8, 9], що досягнуто завдяки використанню асинхронних викликів та ефективному управлінню потоками. Це забезпечує швидку обробку запитів та високу продуктивність навіть при значних навантаженнях на сервіс. Також FastAPI має зручний та інтуїтивно зрозумілий синтаксис, що полегшує розробку та тестування програмного забезпечення і сприяє зниженню кількості помилок та підвищенню якості коду.

Найпростіший приклад коду на FastAPI [10], описаний в лістингу 2.1, дозволяє запустити сервіс, який буде прослуховувати вхідні HTTP GET-запити по маршруту «/» і відповідати у форматі JSON «{"message": "Hello World"}».

Лістинг 2.1 – Приклад коду на FastAPI

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

FastAPI підтримує автоматизовану генерацію документації API за допомогою OpenAPI та Swagger, що дозволяє описати маршрути, вхідні і вихідні

типи даних [11]. Це значно спрощує взаємодію з клієнтами та іншими розробниками про можливості та структуру API. Автоматично згенерована документація дозволяє швидко перевіряти та тестувати різні кінцеві точки API.. Приклад Swagger документації зображено на рисунку 2.2.

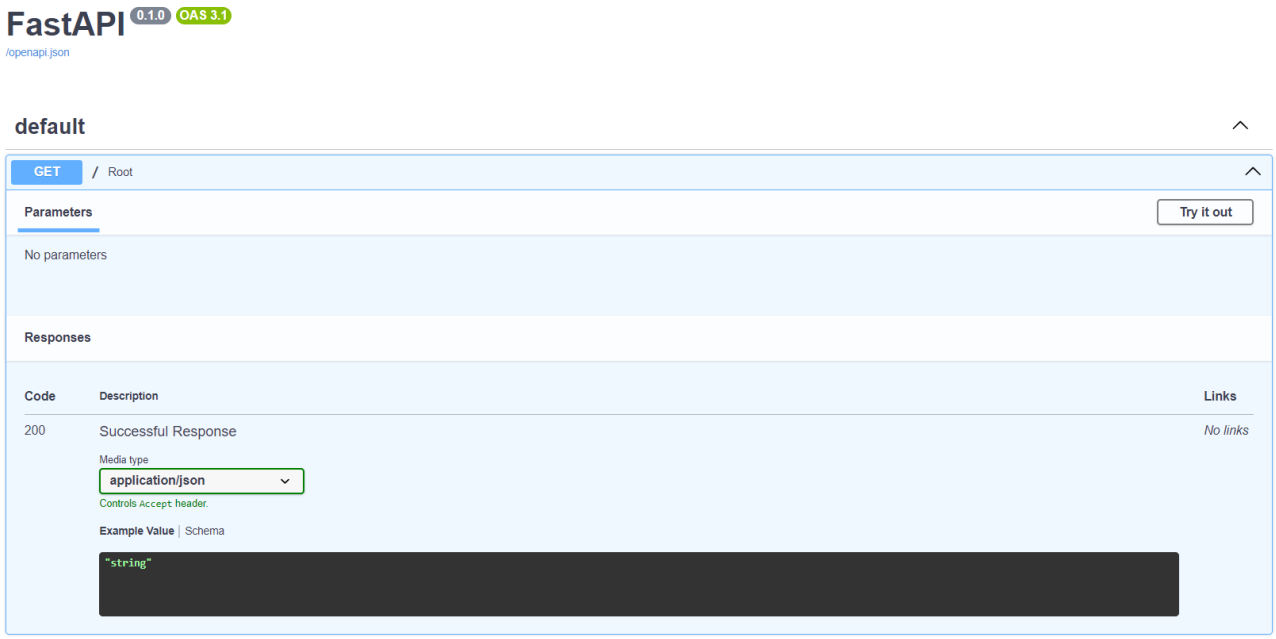


Рисунок 2.2 – Swagger документація, згенерована FastAPI

FastAPI дозволяє легко працювати з різними форматами даних, що необхідно для обробки частково структурованих файлів, Також особливу роль відіграють зручні автоматизовані механізми валідації вхідних даних [12]. Якщо на вхід подано неправильний тип даних або такий, що не пройде перевірку, якщо така запропонована, то запит буде відхилено з HTTP кодом 422 (вміст, що не підлягає обробці) [13].

Отримані файли необхідно передавати для подальшого аналізу програмно-алгоритмічними засобами, тому їх варто зберігати. Для цього найкраще підходять тимчасові файли, в які можна записати вміст отриманих файлів, а після аналізу видалити їх для очищення ресурсів.

2.3 Проєктування аналізу файлів даних програмно-алгоритмічними засобами

Розгляд файлу містить кілька послідовних етапів, кожен з яких відіграє важливу роль у забезпеченні точного аналізу даних. Послідовність даних етапів зображена на рисунку 2.3.

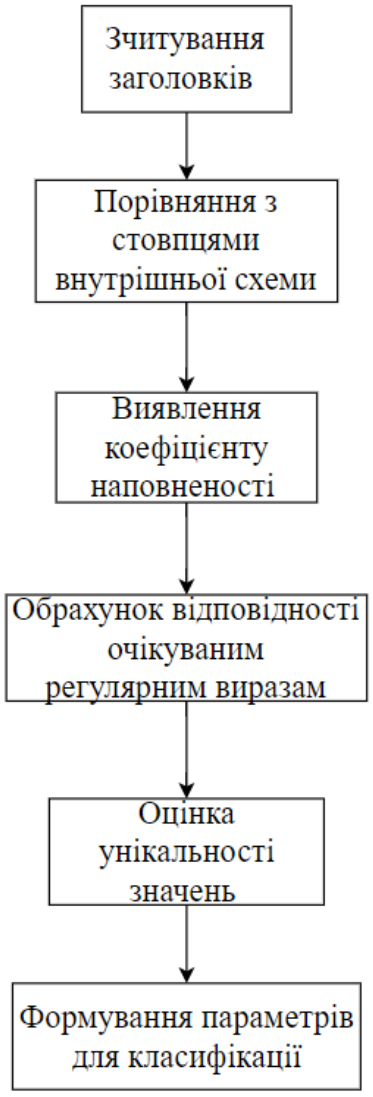
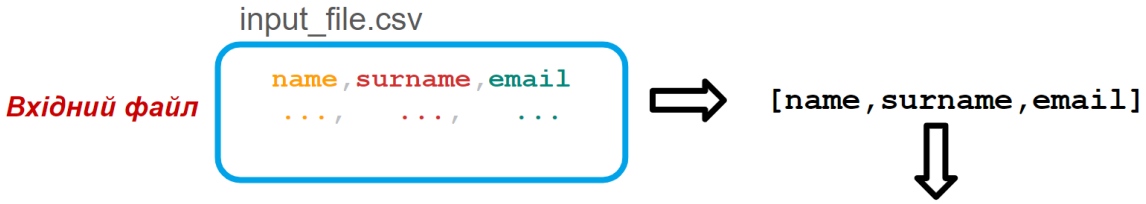


Рисунок 2.3 – Послідовність аналізу файлу даних програмно-алгоритмічними засобами

Першим етапом аналізу схеми даних є зчитування заголовків стовпців з вхідного файлу. Заголовки не є обов’язковими для файлів формату CSV, проте

їх використання доволі поширене, відповідно, вони можуть містити інформацію про те, які дані представлені в кожному стовпці, що є критично важливим для класифікації стовпців файлу до стовпців внутрішньої схеми даних. Процес зчитування заголовків включає ідентифікацію першого рядка файлу, який містить назви стовпців.

Після зчитування заголовки порівнюють зі стовпцями внутрішньої схеми. Цей етап дозволяє автоматизувати процес визначення відповідності між заголовками вхідного файлу та внутрішньою схемою даних, забезпечуючи точність класифікації та мінімізуючи можливі помилки. Процеси зчитування і порівняння заголовків зображено на рисунку 2.4.



	email	firstname	lastname	ip	country	phone_number	zip
0	0.2	1.0	0.8	0.2	0.33333	0.25	0.181818
1	0.153846	0.571429	1.0	0.1	0.5	0.285714	0.181818
2	1.0	0.2	0.2	0.2	0.08333	0.333333	0.2

Матриця схожості заголовків з стовпцями внутрішньої схеми

Рисунок 2.4 – Зчитування заголовків і порівняння з шаблонами назв стовпців внутрішньої схеми даних

Для отримання матриці схожості запропоновано використовувати алгоритми порівняння схожості тексту в назві стовпця і шаблонах назв стовпців внутрішньої схеми. Найкраще значення, яке буде обраховано для кожного стовпця і групи шаблонів назв з внутрішньої схеми даних, буде вважатися за схожість між ними. Результатом буде створення матриці схожості, яка показує ступінь відповідності кожного заголовка з внутрішньої схеми даних із заголовками у файлі. Наступним етапом аналізу схеми даних є виявлення

коефіцієнту наповненості значень. Процес аналізу коефіцієнту наповненості значень зображено на рисунку 2.5.

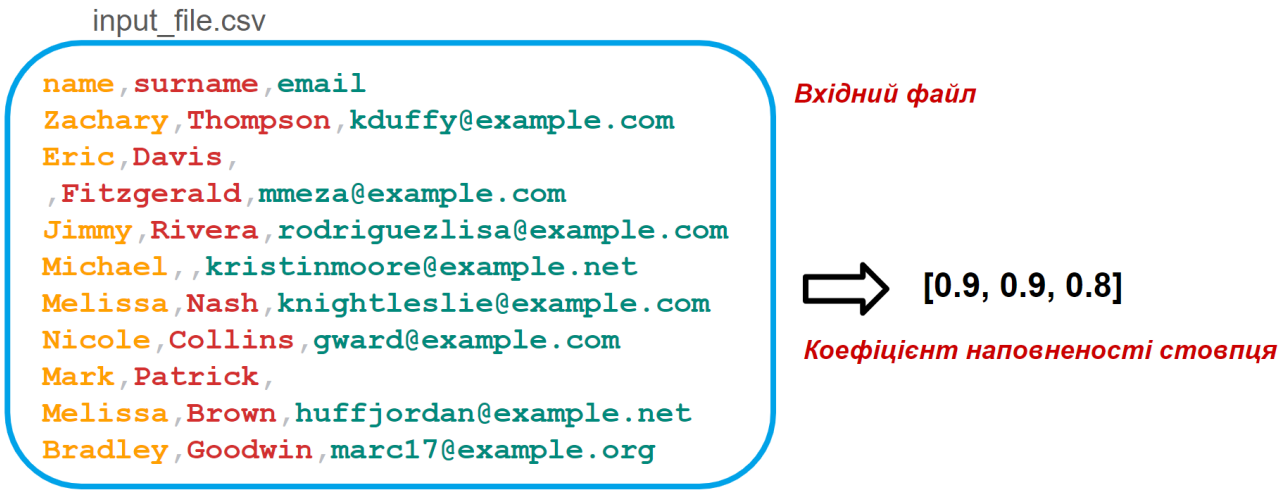


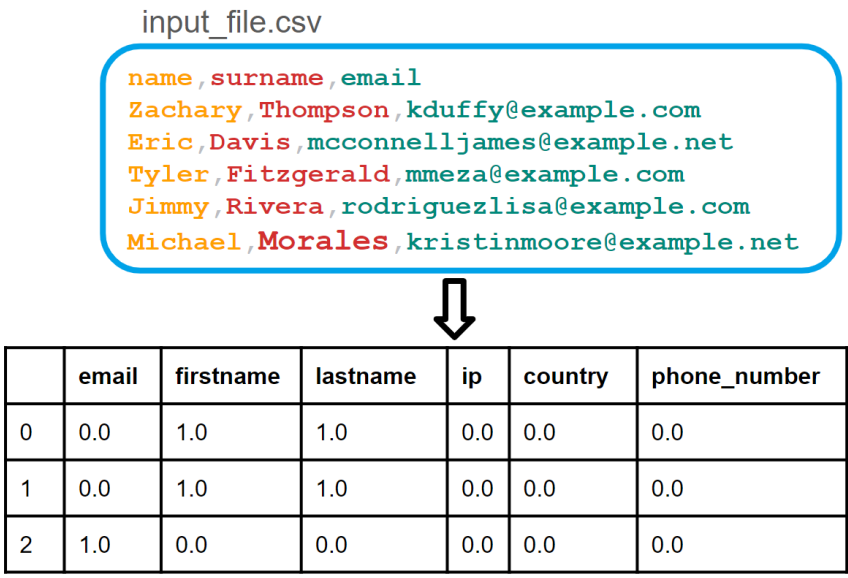
Рисунок 2.5 – Обрахунок коефіцієнту наповненості стовпців вхідного файлу частково структурованих даних

Важливо зазначити, що для цієї візуалізації та наступних, зображених в даній кваліфікаційній роботі освітнього рівня «Бакалавр», використані синтетичні дані, згенеровані бібліотекою Faker на Python.

Коефіцієнт наповненості необхідний для шаблону опрацювання, оскільки він часто є важливим для прийняття рішень при обробці даних. Коефіцієнт наповненості відображає, наскільки повними є дані в кожному стовпці, що допомагає виявити проблемні місця у файлі та прийняти рішення про подальшу обробку або корекцію даних.

Після обчислення коефіцієнту наповненості стовпців вхідного файлу програмно-алгоритмічні засоби обраховують відповідність між значеннями у стовпцях файлу та очікуваними регулярними виразами для стовпців внутрішньої схеми даних. Це включає аналіз кожного значення у стовпцях та перевірку його відповідності регулярним виразам. Цей етап допомагає підтвердити, що дані у кожному стовпці відповідають очікуваним форматам і не містять аномалій або некоректних значень. Процес обрахунку відповідності між значеннями у

стовпцях файлу та регулярними виразами для стовпців внутрішньої схеми даних зображено на рисунку 2.6.



	email	firstname	lastname	ip	country	phone_number	zip
0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
1	0.0	1.0	1.0	0.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Матриця відповідності регулярним виразам

Рисунок 2.6 – Обрахунок відповідності між значеннями у стовпцях файлу та регулярними виразами для стовпців внутрішньої схеми даних

Далі програмно-алгоритмічні засоби визначають унікальність значень у кожному стовпці. Це дозволяє оцінити різноманітність даних, що відіграє критичну роль для подальшої класифікації деяких стовпців, які мають схожі або ідентичні регулярні вирази, при відсутності заголовків.

Наприклад, для вказаної в підрозділі 1.4 внутрішньої схеми даних, стовпці firstname і lastname мають ідентичні регулярні вирази, що робить неможливим розділення лише за ними.

Найпростішим рішенням є порівняння унікальності значень, адже імена набагато менш унікальні, ніж прізвища. Обрахувавши значення унікальності для даних стовпців у останнього буде помітно більший показник, що дозволить моделі класифікації розділити стовпці правильно. Процес обрахунку коефіцієнту наповненості стовпів вхідного файлу даних зображено на рисунку 2.7.

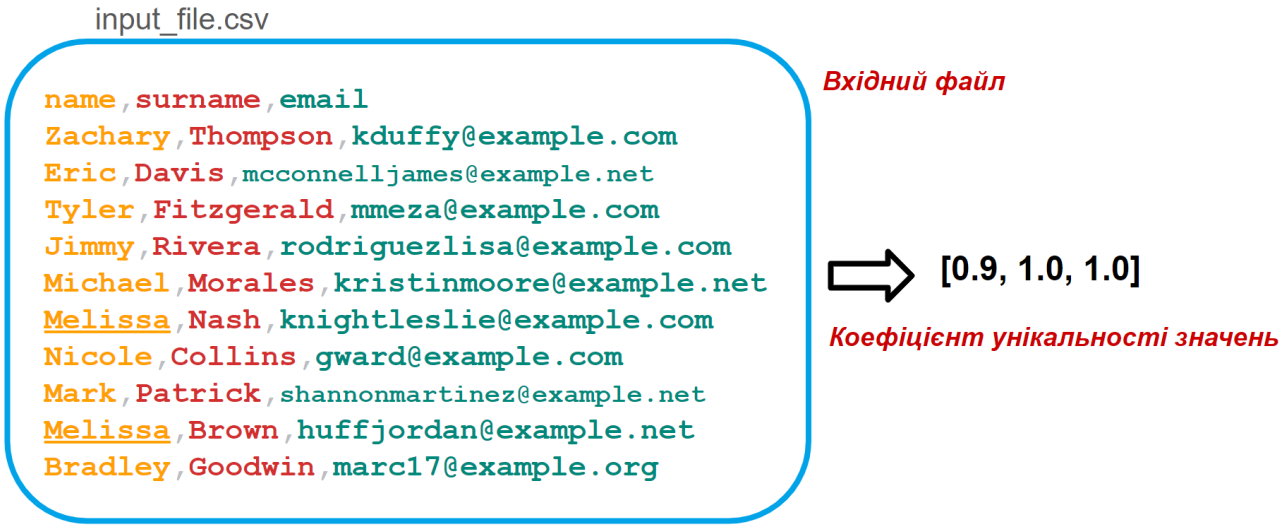


Рисунок 2.7 – Обрахунок коефіцієнту наповненості

Після виконання всіх попередніх етапів програмно-алгоритмічні засоби для кожного стовпця формують параметри, необхідні для їх класифікації. Ці параметри включають результати порівняння заголовків, відповідностей та оцінки унікальності.

2.4 Проєктування класифікації стовпців програмно-алгоритмічними засобами

На вхід алгоритму машинного навчання для кожного стовпця надходить набір параметрів, які описані в пункті 2.3. Для класифікації стовпців найкраще підходить нейронна мережа, яка буде приймати на вхід параметри стовпця файлу даних і повертати ймовірність відповідності до кожного стовпця внутрішньої схеми [14].

Отже, кількість вхідних нейронів буде дорівнювати 2n + 1, а кількість вихідних буде дорівнювати n,

де n – це кількість стовпців внутрішньої схеми даних.

Для даного випадку – це 15 і 7 нейронів відповідно.

Архітектура моделі класифікації стовпців із врахуванням кількості входів і виходів зображена на рисунку 2.8.

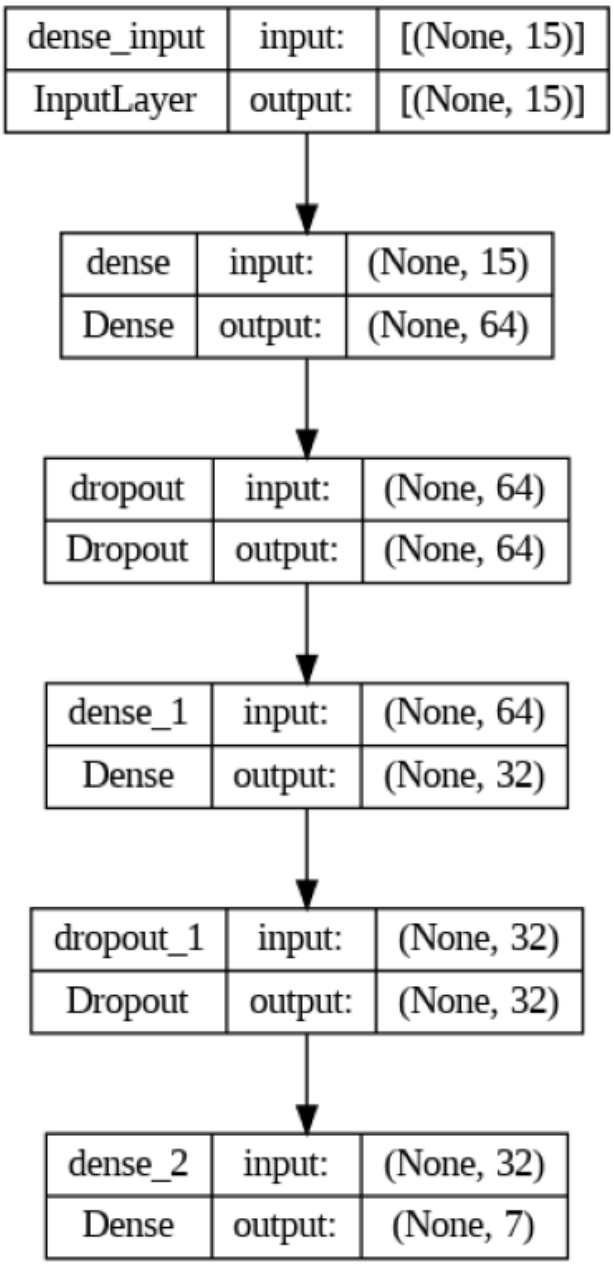


Рисунок 2.8 – Архітектура нейронної мережі для класифікації стовпців програмно-алгоритмічними засобами

Запропонована архітектура складається з наступних шарів:

1. Вхідний шар (15 нейронів) приймає вхідні параметри стовпця, які описують його характеристики.

2. Перший прихований шар (64 нейрони) забезпечує достатню кількість нейронів для вивчення складних зв'язків між вхідними параметрами.

3. Другий прихований шар (32 нейрони) додатково обробляє інформацію, зменшуючи кількість нейронів для уникнення перенавчання та підвищення загальної ефективності моделі.

4. Вихідний шар (7 нейронів) повертає ймовірність відповідності до кожного стовпця внутрішньої схеми.

Кількість прихованих шарів і нейронів на них вибрана таким чином, щоб модель була достатньо гнучкою і потужною для вивчення складних зв’язків, але при цьому не була занадто складною, що сповільнювало б її роботу і викликало перенавчання. Два приховані шари з 64 і 32 нейронами відповідно забезпечують достатню гнучкість і потужність для розв'язання задачі класифікації.

Для прихованих шарів обрана функція активації ReLU (англ. Rectified Linear Unit – зрізаний лінійний вузол), оскільки вона ефективна для вивчення складних зв’язків в даних [15]. Функція ReLU є нелінійною і дозволяє моделі навчатися складним взаємозв'язкам, швидко обчислюючи значення та допомагаючи уникнути проблеми зникання градієнта (див. формулу 2.1).

$$ReLU(x) = \max(0, x) \tag{2.1}$$

Для запобігання перенавчання нейронної мережі також варто використовувати техніку Dropout на прихованих шарах моделі, випадково замінюючи частину нейронів на нулі під час тренування. Це особливо важливо для невеликого або середнього обсягу даних для навчання. Dropout допомагає моделі узагальнювати знання і не звикати до даних, використаних під час навчання. Рекомендованими параметрами Dropout є 0.2-0.5 [16].

Для вихідного шару обрано функцію активації softmax для багатокласової класифікації [17], оскільки її результатом (див. формулу 2.2) є ймовірність приналежності до кожного класу, що дозволяє легко визначити найвідповідніший із них для кожного входу.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (2.2)$$

Така архітектура моделі дозволяє ефективно вивчати складні зв'язки між різними типами вхідних характеристик, забезпечуючи високу точність класифікації.

2.5 Проєктування формування шаблону опрацювання частково структурованого файлу даних

Після аналізу файлу даних і класифікації стовпців програмно-алгоритмічні засоби формують шаблон опрацювання частково структурованих файлів даних. Створений шаблон повинен бути зрозумілим, інформативним та зручним для аналізу. Для цього він має включати наступні основні компоненти:

- Кількість рядків у вхідному файлі – це дає уявлення про обсяг даних, які необхідно опрацювати.
- Наявність заголовків – цей параметр вказує, чи містить вхідний файл заголовки стовпців. Це важливо для правильного читання і інтерпретації даних.
- Список стовпців даних.

Кожен стовпець даних програмно-алгоритмічні засоби описують, використовуючи наступну інформацію:

- Назва стовпця у внутрішній схемі.
- Назва вхідного стовпця.
- Регулярний вираз, використаний для валідації даних у стовпці.
- Коефіцієнт заповнених значень, який вказує на наповненість значень у стовпці.
- Коефіцієнт відповідності значень заданому регулярному виразу.

Для повернення шаблону найкраще підходить формат JSON через його гнучкість, інтуїтивне розуміння та сумісність з REST API. Крім того, структура JSON забезпечує зручне зберігання вкладених об'єктів даних, що підходить для

списку стовпців в шаблоні. Приклад сформованого шаблону опрацювання частково структурованих даних у форматі JSON описаний в лістингу 2.2.

Лістинг 2.2 – Приклад шаблону опрацювання частково структурованих даних

```
{
  "columns": [
    {
      "name": "firstname",
      "source_name": "name",
      "pattern": "[a-zA-Z]{1,20}",
      "filled_values_ratio": 0.89,
      "matched_values_ratio": 1.0
    },
    {
      "name": "email",
      "source_name": "email address",
      "pattern": ".*@.*",
      "filled_values_ratio": 0.95,
      "matched_values_ratio": 0.98
    }
    // інші стовпці
  ],
  "rows_number": 1489357,
  "header": true
}
```

Такий шаблон опрацювання частково структурованого файлу даних дає інформацію про те, як саме необхідно взаємодіяти зі стовпцями для приведення їх до внутрішньої схеми даних, а також дозволяє приймати рішення на основі кількості, наповненості і відповідності вхідних значень.

2.6 Проєктування інфраструктури контейнеризації для програмно-алгоритмічних засобів

Для контейнеризації програмно-алгоритмічних засобів було вирішено використовувати Docker [18]. Це створює ряд переваг, зокрема:

- Ізоляція середовища – кожен компонент працює в окремому контейнері, що запобігає конфліктам залежностей. Це забезпечує стабільність та

передбачуваність роботи програмно-алгоритмічних засобів, оскільки зміни в одному контейнері не впливають на інші. Ізоляція також підвищує безпеку, оскільки кожен контейнер має свої власні ресурси і обмеження доступу.

- Легкість розгортання – контейнери можна легко розгортати на будь-якому сервері, який підтримує Docker. Це забезпечує гнучкість у виборі середовища для розгортання, знижуючи витрати на налаштування та підтримку інфраструктури. Завдяки Docker можна швидко налаштувати нові середовища для розробки, тестування або виробничої експлуатації.
- Масштабованість – можливість швидкого масштабування системи за рахунок додавання нових контейнерів. Це дозволяє легко адаптувати систему до зростаючих вимог, збільшуючи її продуктивність і забезпечуючи стабільну роботу під великим навантаженням. Автоматичне масштабування за допомогою оркестраційних інструментів, таких як Kubernetes, додатково спрощує управління масштабуванням.
- Відтворюваність середовища – гарантована робота програмно-алгоритмічних засобів в однакових умовах незалежно від середовища. Це забезпечує стабільність та передбачуваність роботи, що особливо важливо при розробці, тестуванні та розгортанні нових версій. Відтворюваність полегшує діагностику проблем, оскільки можна бути впевненим, що програмно-алгоритмічні засоби працюють однаково в усіх середовищах.

Додатково, використання Docker спрощує процес управління версіями та оновленнями компонентів системи. Кожен контейнер може бути налаштований з окремою версією програмного забезпечення, що дозволяє тестувати та впроваджувати оновлення без ризику вплинути на всю систему. Це також забезпечує швидке відновлення роботи у випадку збоїв або неполадок, оскільки можна швидко замінити несправний контейнер на новий.

Контейнеризація також полегшує процеси CI/CD (безперервної інтеграції та доставки), оскільки забезпечує стандартизоване середовище для тестування та розгортання. Це сприяє більш швидкому випуску нових функцій та виправлень,

підвищуючи загальну ефективність розробки та експлуатації програмно-алгоритмічних засобів. Структура Docker-контейнера зображена на рисунку 2.9.

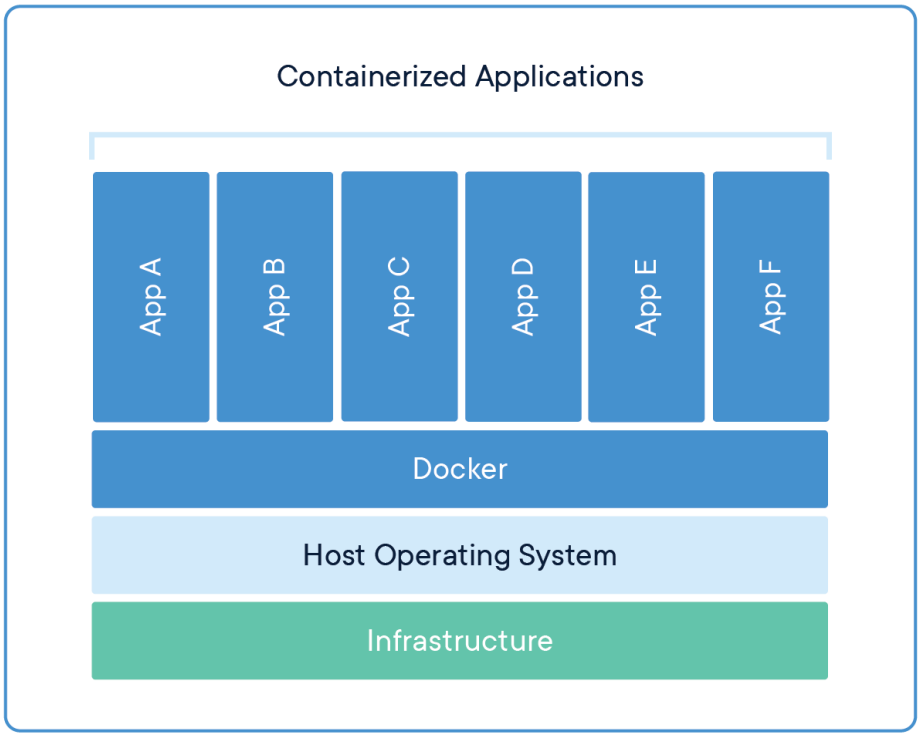


Рисунок 2.9 – Структура Docker-контейнера

В програмно-алгоритмічних засобах можливість легкого розгортання і масштабування системи забезпечує швидку адаптацію до зростаючих вимог та ефективне управління ресурсами, що дозволяє підтримувати стабільну і надійну роботу навіть під великим навантаженням.

2.7 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було розглянуто проєктну частину розробки програмно-алгоритмічних засобів формування шаблонів опрацювання частково структурованих файлів даних.

Основні результати проєктної частини включають визначення потоків даних, опис інтерфейсу для прийняття файлів даних, планування кроків для аналізу даних, визначення структури нейромережі для класифікації.

Було розроблено архітектуру програмно-алгоритмічних засобів, яка включає користувацький інтерфейс у вигляді REST API для отримання файлів даних, а також обробку цих файлів та модель машинного навчання для їх класифікації. Архітектура забезпечує ефективну взаємодію між компонентами та можливість розширення функціональності у майбутньому.

Були розроблені ключові алгоритми для аналізу структури файлів даних, класифікації стовпців, валідації даних та генерації шаблонів опрацювання. Ці алгоритми забезпечують точність та ефективність обробки частково структурованих файлів, автоматизуючи процес створення шаблонів та забезпечуючи високу якість результатів.

Описано REST API, що забезпечує зручний спосіб взаємодії користувачів із програмно-алгоритмічними засобами через HTTP POST-запити. Були розглянуті основні методи API, процес обробки файлів та формати відповідей. Це забезпечує можливість інтеграції з іншими системами та подальшого розвитку користувацького інтерфейсу.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА РОЗРОБКИ ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ФОРМУВАННЯ ШАБЛОНІВ ОПРАЦЮВАННЯ ЧАСТКОВО СТРУКТУРОВАНИХ ФАЙЛІВ ДАНИХ

3.1 Реалізація механізму отримання даних в програмно-алгоритмічних засобах

Як описано в підрозділі 2.1, для отримання файлів даних у програмно-алгоритмічних засобах було обрано підхід REST API, який реалізовано за допомогою фреймворку FastAPI. Цей механізм забезпечує прийом файлів від користувачів, їх тимчасове зберігання для аналізу та подальшої обробки. У даному випадку FastAPI використаний для маршруту, який приймає файли для обробки [19]. Код, який створений для отримання файлів даних при виконанні кваліфікаційної роботи, описано в лістингу 3.1.

Лістинг 3.1 – Код інтерфейсу на FastAPI

```
app = FastAPI()


@app.post("/file/")
async def analyze_file(file: UploadFile = File(...)):
    with tempfile.NamedTemporaryFile(delete=False) as temp_file:
        temp_file.write(await file.read())
        ...
    os.remove(temp_file.name)
    ...
```

Основний маршрут для завантаження файлу визначений за допомогою декоратора @app.post("/file/"). Цей маршрут дозволяє приймати файли через HTTP POST-запит. Завантажені файли обробляє асинхронна функція analyze_file, яка приймає файл у форматі UploadFile. Використання асинхронної обробки дозволяє ефективно обробляти кілька запитів одночасно, що підвищує продуктивність системи.

Для тимчасового зберігання завантажених даних використаний модуль tempfile, що створює тимчасовий файл, у який буде записано дані з завантаженого файлу. Це дозволяє уникнути необхідності зберігати великі обсяги інформації на постійній основі, забезпечуючи ефективне використання ресурсів. Після зберігання даних у тимчасовому файлі буде відбуватися аналіз файлу. Після завершення аналізу тимчасовий файл буде видалено для звільнення ресурсів.

Для запуску інтерфейсу на FastAPI використано Uvicorn, високопродуктивний сервер для ASGI-додатків (англ. Asynchronous Server Gateway Interface – асинхронний інтерфейс веб-серверів) [20, 21]. На рисунку 3.1 зображено Swagger документацію, згенеровану написаним в кваліфікаційній роботі кодом для FastAPI.

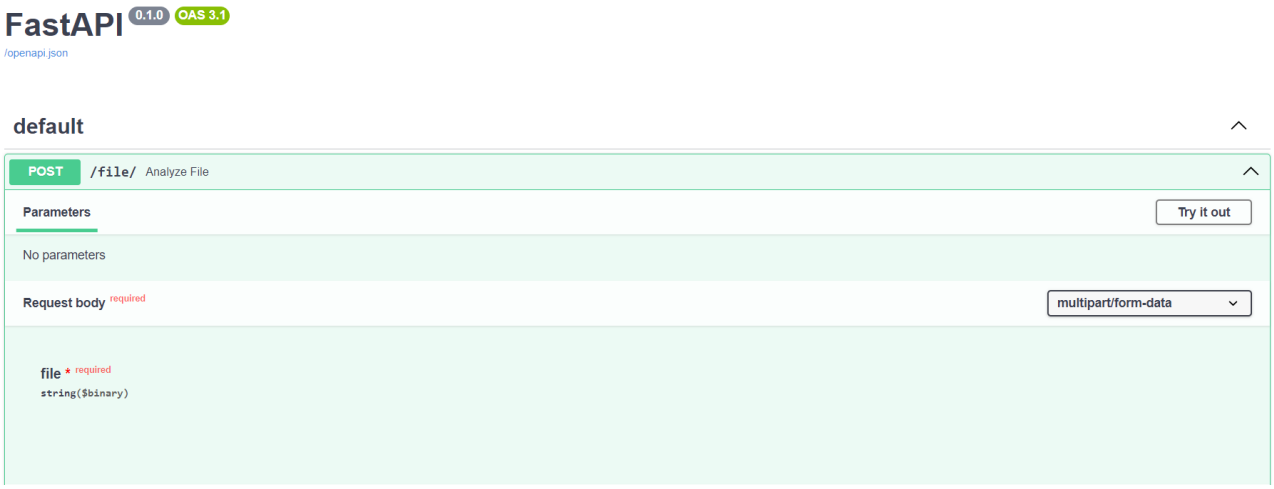


Рисунок 3.1 – Swagger документація інтерфейсу програмно-алгоритмічних засобів

Ця документація дозволяє відправляти запити на інтерфейс для його використання та тестування.

Для інтеграції з іншими системами необхідно виконати з них POST-запит, передавши файл в параметрах. В лістингу 3.2 показано приклад такого коду, який дозволяє відправити файл на обробку програмно-алгоритмічними засобами і вивести результат у форматі JSON.

Лістинг 3.2 – Код для відправки файлу на програмно-алгоритмічні засоби

```
import requests

url = 'http://localhost:8000/file/'

file_path = 'input_file1.csv'

with open(file_path, 'rb') as file:
    files = {'file': file}
    response = requests.post(url, files=files)

if response.status_code == 200:
    print(response.json())
else:
    print(f'Error: {response.status_code}')
    print(response.text)
```

Формат вхідних даних в Swagger документації обмежений типом, вказаним в конфігурації, проте, якщо використовувати HTTP POST-запити на інтерфейс напряму, як описано в лістингу 3.2, і вказати неправильний тип даних, FastAPI відхилить запит з кодом 422.

3.2 Реалізація аналізу файлу програмно-алгоритмічними засобами

Реалізація аналізу файлу включає визначення відповідності даних певним регулярним виразам та створення звіту про обробку. У цьому підрозділі наведено опис впровадження механізму здійсненого в кваліфікаційній роботі освітнього рівня «Бакалавр».

У лістингу 3.3 наведено створену при виконанні роботи функцію, яку програмно-алгоритмічні засоби викликають після того, як отримали файл даних через API і зберегли його в тимчасовий файл.

Лістинг 3.3 – Код аналізу файлу даних

```
header = get_header(file_path)
header_similarity_df = get_header_similarity_matrix(header,
column_registry)
df = pd.read_csv(file_path, header=None, skiprows=1,
skipinitialspace=True)
matches_df = get_matches_matrix(df, column_registry)
```

```
uniqueness = get_columns_uniqueness(df)
params = []
for index in range(len(header)):
    row_params = list(header_similarity_df.loc[index]) +
list(matches_df.loc[index].values) + [uniqueness[index]]
    params.append(row_params)
```

Створена функція get_header зчитує заголовки стовпців з файлу.

Наступним кроком є порівняння цих заголовків з реєстром відомих стовпців за допомогою реалізованої в ході роботи функції get_header_similarity_matrix, що повертає матрицю схожості, яка показує, наскільки заголовки файлу відповідають очікуваним заголовкам з реєстру. Для порівняння використовується алгоритм відстані Левенштейна [22], код наведено в додатку Б.

Написана у ході виконання роботи функція get_matches_matrix створює матрицю відповідностей між даними у стовпцях файлу та очікуваними регулярними виразами стовпців внутрішньої схеми даних. Це дозволяє визначити, наскільки значення у стовпцях відповідають очікуваним регулярним виразам.

Унікальність значень у кожному стовпці визначають за допомогою реалізованої функції get_columns_uniqueness. Ця інформація є важливою для розуміння структури та якості даних.

У результаті виконання функцій для кожного стовпця буде сформовано параметри, які включають порівняння заголовків, відповідностей регулярним виразам та оцінки унікальності. Ці числові параметри об’єднані у список, який містить дані для подальшої класифікації нейронною мережею [23].

3.3 Реалізація класифікації стовпців програмно-алгоритмічними засобами

Для реалізації класифікації стовпців використано нейронну мережу, архітектура якої розроблена в підрозділі 2.3. Нейронна мережа, побудована за

допомогою бібліотеки TensorFlow та Keras, код реалізації наведено в лістингу 3.4 [24].

Лістинг 3.4 – Реалізація нейронної мережі

```
model = Sequential([
    Dense(64, input_dim=X.shape[1], activation='relu'),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dropout(0.5),
    Dense(y.shape[1], activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

Першим кроком є збирання та підготовка даних для навчання моделі. Для цього у кваліфікаційній роботі використано файли з різними схемами даних. У конфігураційному файлі збережена інформація, в якій вказані назви стовпців для кожного файлу. Приклад конфігураційного файлу columns.yaml наведено в лістингу 3.5.

Лістинг 3.5 – Конфігураційний файл для набору навчальних даних

```
file_1.csv:
- country
- firstname
- phone_number
- lastname
file_2.csv:
- country
- lastname
- firstname
file_3.csv:
- ip
- email
- firstname
- lastname
- zip
- phone_number
```

Розроблена модель компілюється з використанням оптимізатора adam і функції втрат categorical_crossentropy, яка підходить для багатокласової класифікації. Після цього модель навчається на підготовлених даних. Навчання

проводиться на тренувальному наборі даних, а точність моделі оцінюється на валідаційному наборі.

Після навчання моделі проведено оцінку її ефективності на тестовому наборі даних. Для цього використано обраховані нейронною мережею метрики точності, які показують наскільки добре модель здатна класифікувати стовпці. Тестування моделі включало використання набору даних, який містив різні типи файлів для забезпечення різноманітності даних та оцінки адаптивності моделі.

Для відтворення реального середовища, в якому дані не завжди хорошої якості, в тренувальному датасеті використано значну кількість файлів з відсутніми заголовками і такими, що не відповідають жодному з шаблонів назв стовпців внутрішньої схеми даних, а також частину значень замінено на пошкоджені або відсутні. Це дозволяє моделі навчитись працювати з неповними даними, що часто зустрічаються в реальних сценаріях обробки даних. Процес тренування моделі, що був реалізований у ході виконання кваліфікаційної роботи, зображено на рисунку 3.2.

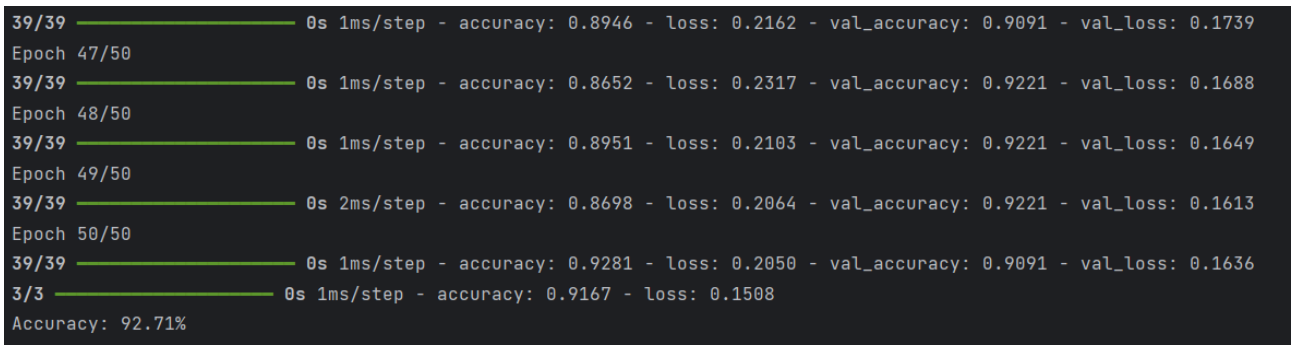


Рисунок 3.2 – Тренування нейронної мережі

У результаті тренування досягнута точність роботи нейронної мережі 92.71% [25]. Такий високий показник свідчить про здатність моделі правильно визначати типи стовпців на основі вхідних даних.

Після завершення навчання та оцінки точності модель зберігають для подальшого використання. При кожному наступному застосуванні нейронної

мережі для класифікації стовпців у нових файлах даних збережена модель завантажується без потреби у повторному навчанні.

3.4 Реалізація генерації шаблону опрацювання частково структурованого файлу даних

Генерація шаблону опрацювання файлу частково структурованих даних і його повернення є останнім етапом у роботі програмно-алгоритмічних засобів.

Для зручного представлення результатів використовується два основних класи даних [26, 27]: ColumnReport та Report.

Клас ColumnReport, описаний в лістингу 3.6, містить інформацію про окремий стовпець файлу. Він включає назву стовпця, його початкову назву, регулярний вираз для валідації, відсоток заповнених значень та відсоток значень, що відповідають заданому регулярному виразу.

Лістинг 3.6 – Клас ColumnReport для інформації про стовцець з даними вхідного файлу

```
@dataclass
class ColumnReport:
    name: str
    sourceName: str
    pattern: str
    filledValuesRatio: float
    matchedValuesRatio: float
```

Клас Report, описаний в лістингу 3.7, містить загальну інформацію про файл та список об’єктів ColumnReport, які характеризують кожен стовпець. Він включає загальну кількість рядків, інформацію про наявність заголовків та список звітів по стовпцях.

Лістинг 3.7 – Клас Report для інформації про вхідний файл

```
@dataclass
class Report:
    rowsNumber: int
```

```
    header: bool
    columns: list[ColumnReport]
```

Після обробки файлу і отримання необхідних метрик, таких як заповненість та відповідність значень регулярним виразам, описаних в підрозділі 3.2, програмно-алгоритмічні засоби генерують шаблон опрацювання частково структурованого файлу даних. Для кожного стовпця створюють об’єкт ColumnReport, який включає всі необхідні дані. Ці об’єкти збирають у список, який потім використовується для створення об'єкта Report, код наведено в лістингу 3.8.

Лістинг 3.8 – Збір даних в шаблон опрацювання

```
column_reports = [ColumnReport(column.name, header_column if
is_header else index, column.regex, filled, match) for index,
(column, header_column, filled, match) in enumerate(zip(columns,
header, filled_ratios, match_ratios))]
report = Report(len(df), is_header, columns=column_reports)
```

У цьому коді columns містить список об’єктів стовпців, header – заголовки стовпців файлу, filled_ratios – відсотки заповнених значень, а match_ratios – відсотки значень, що відповідають шаблонам. Кожен стовпець обробляють окремо і результати збирають у column_reports.

Для повернення результату у форматі JSON використано створену функцію dataclass_to_dict, яка рекурсивно перетворює об'єкти класів даних у словники. Код даної функції описано в лістингу 3.9.

Лістинг 3.9 – Приведення шаблонів опрацювання у формат JSON для повернення через інтерфейс

```
def dataclass_to_dict(obj):
    if isinstance(obj, list):
        return [dataclass_to_dict(item) for item in obj]
    elif hasattr(obj, '__dataclass_fields__'):
        result = {}
        for field in obj.__dataclass_fields__:
            value = getattr(obj, field)
            result[field] = dataclass_to_dict(value)
        return result
```

```
    else:
        return obj
```

Дана функція перетворює об’єкти, у яких є атрибути класів даних, у словники. Ця трансформація забезпечує гнучкість і сумісність із різними структурами даних.

3.5 Реалізація інфраструктури контейнеризації для програмно-алгоритмічних засобів

Для контейнеризації використано Docker, як було визначено в підрозділі 2.5. Він дозволяє упакувати програмно-алгоритмічні засоби разом з усіма їхніми залежностями в єдиний образ, що забезпечує стабільність і відтворюваність середовища виконання. Для збірки програмно-алгоритмічних засобів у Docker-образ створено Dockerfile. Dockerfile містить інструкції для побудови образу, включаючи копіювання файлів, встановлення залежностей та налаштування середовища виконання. Його вміст наведено в лістингу 3.10.

Лістинг 3.10 – Код Dockerfile

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt /app/
COPY *.py /app/
COPY utils /app/utils
COPY model/column_classification_model.keras /app/model/

RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 8000

CMD ["uvicorn", "main:app", "--port", "8000"]
```

Цей код дозволяє створити Docker-образ, що включає всі необхідні компоненти для запуску і розгорнути середовище, передбачене для роботи програмно-алгоритмічних засобів. Для програмно-алгоритмічних засобів було

обрано базовий образ python:3.9-slim, який базується на мінімалізованому варіанті офіційного образу Debian. Команда EXPOSE у Dockerfile використана для індикації того, що контейнер буде прослуховувати порт 8000 під час виконання. Це дозволяє зовнішнім клієнтам взаємодіяти з API, що працює всередині контейнера.

Також при старті контейнера автоматично буде стартувати інтерфейс програмно-алгоритмічних засобів, готовий до отримання вхідних частково структурованих файлів даних. Для створення Docker-образу необхідно виконати команду docker build.

Для запуску контейнера на основі створеного образу виконується команда docker run з вказуванням якому порту на хостовій машині відповідає порт 8000 всередині контейнера.

Це забезпечує ізольоване середовище, легке розгортання та масштабованість програмно-алгоритмічних засобів. Використання Docker дозволяє швидко масштабувати їх, додаючи нові контейнери для обробки збільшених обсягів даних або розподіляючи навантаження між кількома серверами.

3.6 Аналіз результатів роботи програмно-алгоритмічних засобів

Для оцінки ефективності розроблених програмно-алгоритмічних засобів були проведені тестування на різних наборах даних, які містили частково структуровані файли з різноманітними схемами даних. Ключовими метриками для оцінки точності роботи засобів був відсоток правильного визначення типів стовпців, що показує, наскільки правильно засоби визначають типи даних у стовпцях.

Результати тестування показали високу ефективність та точність програмно-алгоритмічних засобів. Для тестового набору даних відсоток правильного визначення типів стовпців перевищував 92%, що свідчить про високу точність алгоритмів класифікації.

Правильність розпізнавання стовпців продемонстровано в таблиці 3.1.

Таблиця 3.1 – Правильність розпізнавання стовпців у тестовому наборі даних

Назва стовпця	**Відсоток правильності класифікації при наявності заголовків**	**Відсоток правильності класифікації (загальний)**
email	100%	100%
firstname	100%	91.8%
lastname	100%	78.8%
ip	100%	100%
country	100%	100%
phone_number	100%	100%
zip	100%	100%

Результати показують, що програмно-алгоритмічні засоби здатні доволі коректно обробляти файли з різними схемами даних.

Нижчий відсоток правильності у стовпців firstname і lastname при відсутності заголовків у великої частини даних тестового набору зумовлений ідентичними регулярними виразами, що робить важчим їх розпізнавання. Матрицю помилок класифікації наведено в таблиці 3.2 [28].

Таблиця 3.2 – Матриця помилок класифікації для стовпців firstname і lastname при відсутності заголовків

	firstname розпізнано	**lastname розпізнано**
firstname було	57	11
lastname було	17	42

Для покращення цих результатів в подальшому заплановано розглянути існуючі моделі машинного навчання розраховані на ідентифікацію і розділення імен та прізвищ.

3.7 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи програмно-алгоритмічні засоби, які були розроблені, показали високу ефективність у вирішенні задачі формування шаблонів для опрацювання частково структурованих файлів даних. Вони дозволяють автоматизувати процес визначення схеми даних, що значно спрощує їх подальшу обробку та аналіз.

Подальший розвиток програмно-алгоритмічних засобів може включати інтеграцію складніших алгоритмів машинного навчання для кращої обробки форматів даних, а також розширення функціоналу для роботи з великими обсягами даних.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Принципи ергономіки робочого місця

Ергономіка вивчає діяльність людини в умовах сучасного виробництва з метою оптимізації знарядь і процесу праці та забезпечення необхідних зручностей для людини.

Згідно наукових досліджень з безпеки та гігієни праці – одна четверта причин лікарняних в офісі зумовлені проблемами зі спиною. Вартість простою виробництва на рік через хвороби опорно-рухового апарату становить мільярди доларів. Велика кількість працівників щодня пропускають роботу через так званий офісний синдром – групу симптомів, які виникають від тривалого перебування в одному й тому ж положенні, переважно під час роботи за комп’ютером в офісі (міофасциальний больовий синдром, синдром зап’ястного каналу, тендиніт тощо).

Тому забезпечення ергономічного робочого місця, середовища і меблів для людей – важливе завдання.

Робоче місце – це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників.

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпуса, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні положення тіла.

Проведені дослідження показують, що при раціональній організації робочих місць продуктивність праці зростає на 15-25%.

В Україні основні ергономічні вимоги до проєктування робочого місця користувача комп’ютера в системі «людина – техніка – виробниче середовище» визначаються державними стандартами: ДСТУ 8604:2015. Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги [37]; ДСТУ 7299:2013. Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки [38]; ДСТУ 7234:2011. Дизайн і ергономіка. Обладнання виробниче. Загальні вимоги дизайну та ергономіки [39].

Організація робочого місця передбачає:

- Правильне розміщення робочого місця у виробничому приміщенні.
- Вибір ергономічно обґрунтованого робочого положення, виробничих меблів з урахуванням антропометричних характеристик людини.
- Раціональне компонування обладнання на робочих місцях.
- Урахування характеру та особливостей трудової діяльності.

Статичні напруження працівника в процесі праці пов’язані з підтриманням у нерухомому стані предметів і знарядь праці, а також підтриманням робочої пози.

Робоча поза – це основне положення працівника у просторі: зручна робоча поза має забезпечувати стійкість положення корпуса, ніг, рук, голови працівника під час роботи, мінімальні затрати енергії та максимальну результативність праці.

Найпоширенішими у процесі праці є пози сидячи і стоячи. Проєктуючи робоче місце, потрібно враховувати, що при виконанні роботи з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи.

Робоча поза стоячи втомлює людину більше, ніж сидяча. Вона вимагає на 10 % більше енергії, спричиняє підвищення артеріального і венозного тиску крові, розширення вен на ногах, пошкодження ступень, викривлення хребта.

Під час роботи сидячи нижня частина корпуса розслаблена, а основне статичне навантаження припадає на м’язи шиї, спини, таза, стегон. Неправильна

сидяча поза може викликати застій крові в ногах, а якщо виконується великий обсяг роботи для пальців рук – запалення суглобів.

Організація робочого місця користувача комп'ютера повинна забезпечувати відповідність усіх елементів робочого місця та їх взаємного розташування ергономічним вимогам, які зображено на рисунку 4.1.

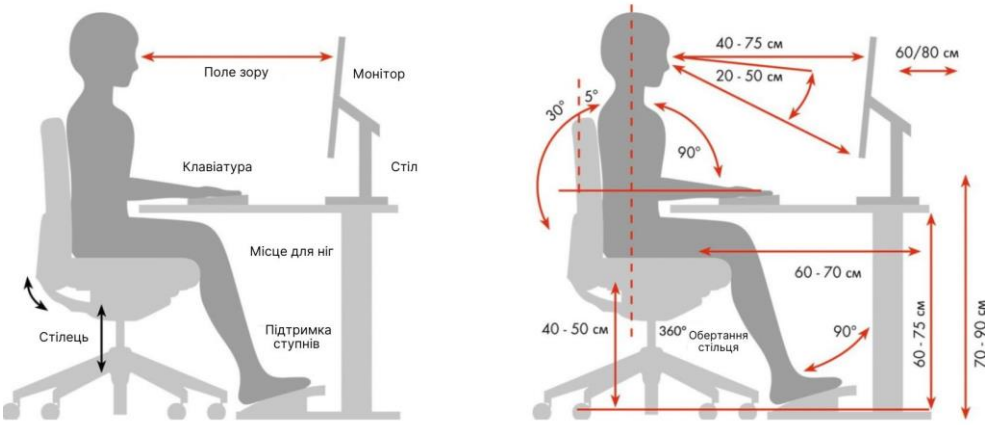


Рисунок 4.1 – Ергономічно ідеальне облаштування робочого місця

При організації робочого місця необхідно враховувати такі аспекти ергономіки в інтер'єрі:

Підтримка тіла – створення умов, за яких людина може правильно і комфортно розташувати своє тіло під час роботи, мінімізуючи навантаження на хребет і знижуючи ризик виникнення болю і напруги в м'язах. Для забезпечення підтримки тіла необхідно звернути увагу на такі чинники:

- Сидіння має володіти регульованою висотою, спинкою, підлокітниками, бути досить піддатливим, щоб адаптуватися до форми тіла і забезпечувати оптимальну підтримку спини, шиї та попереку. Подушка або спеціальний підголівник можуть бути використані для додаткової підтримки шиї.
- Стілець потрібно вибирати з правильною формою і підтримкою, щоб він забезпечував стійку основу і комфортне сидіння. Регульована висота дасть змогу адаптувати його під індивідуальні потреби.

- Підставка для ніг допомагає знизити навантаження на нижню частину спини і поліпшити кровообіг у ногах.

- Використання ергономічної клавіатури та миші з підтримкою зап’ястя може знизити навантаження на руки та запобігти розвитку синдрому карпального каналу. Вони мають бути розташовані на такій відстані та висоті, щоб руки та зап’ястя перебували в природному та комфортному положенні.

Розміщення та відстань допомагають створити комфортні умови роботи, крутий дизайн будинку, мінімізують навантаження на тіло та сприяють ефективності діяльності. Принципи, які слід враховувати під час розміщення меблів та обладнання:

- Висота робочого столу і крісла має бути регульованою, щоб можна було налаштувати її під оптимальний рівень. Важливо, щоб стіл був досить просторим, щоб розмістити всі необхідні предмети і забезпечити свободу руху. Відстань між робочою поверхнею і нижньою частиною стільниці має бути достатньою для комфортного розміщення ніг.

- Монітор слід розмістити на рівні очей, щоб мінімізувати напруження на шию та очі. Відстань між очима та монітором має бути приблизно 50-70 сантиметрів.

- Меблі в робочому просторі розмістіть таким чином, щоб мінімізувати необхідність поворотів, прогинів тіла, підтримувати правильну висоту допоможе ергономічне крісло.

- Необхідно передбачити достатній простір перед столом, кріслом та іншими робочими елементами, щоб користувач міг вільно переміщатися і розтягуватися під час роботи.

Правильно організоване освітлення допомагає запобігти втомі очей, знижує ризик виникнення напруження та головного болю, а також сприяє підтримці гарного настрою та концентрації. Максимально використовуйте природне освітлення в приміщенні. Розташуйте меблі так, щоб вікно або джерело світла знаходилися збоку або спереду, використовуйте штори або жалюзі, щоб регулювати інтенсивність світла.

Обирайте стабільні лампи з природним світлом, які не викликають мерехтіння та відблисків. Зверніть увагу на можливість регулювання яскравості та колірної температури джерел світла. Це дасть можливість адаптувати освітлення під свої індивідуальні потреби та вподобання.

Доступність означає, що вся необхідна ергономіка меблів, матеріали та обладнання мають бути зручно розташованими для користувача і відповідати необхідній висоті для нього. Переконайтеся, що всі необхідні предмети та інструменти знаходяться в безпосередній близькості від робочого місця або на його поверхні. Розмістіть їх таким чином, щоб вони були легкодоступні, без необхідності сильно нахилятися, прогинатися або витягували руки. Використовуйте системи зберігання та організації, такі як: шухляди, полиці, шафи та контейнери.

Загальні принципи організації робочого місця:

- На робочому місці не повинно бути нічого зайвого. Усі необхідні для роботи предмети мають бути поряд із працівником, але не заважати йому.
- Ті предмети, якими користуються частіше, розташовуються ближче, ніж ті предмети, якими користуються рідше.
- Предмети, які беруть лівою рукою, повинні бути зліва, а ті предмети, які беруть правою рукою – справа.
- Якщо використовують обидві руки, то місце розташування пристосувань вибирається з урахуванням зручності захоплювання його двома руками.
- Робоче місце не повинно бути захаращене.
- Організація робочого місця повинна забезпечувати необхідну оглядовість.

4.2 Правила електробезпеки при роботі з електронними пристроями

В двадцять першому столітті важко уявити собі життя без електронних пристроїв. Електроніка дозволяє нам завжди підтримувати зв’язок з нашими

рідними та близькими, електронні пристрої та програми допомагають нам здобувати освіту легше, швидше та ефективніше, електроніка допомагає нам працювати більш ефективно та продуктивно, майже вся праця вже перейшла у комп’ютер. Однак функціонування практично всіх електронних пристроїв забезпечується електроенергією. Тому робота з ними потребує дотримання правил електробезпеки.

Електробезпека - це система організаційних і технічних заходів і засобів, які забезпечують захист людей від шкідливого і небезпечного впливу електричного струму, електричної дуги, електромагнітного поля і статичної електричної електрики.

Правильне користування електроенергією виключає випадки ураження електричним струмом. Основні вимоги, яких потрібно дотримуватись при користуванні електроенергією:

1. Візуальна перевірка обладнання – перед вмиканням електронного пристрою завжди перевіряйте його шнур на наявність пошкоджень, щоб уникнути короткого замикання або ураження електричним струмом.

2. Захист від коротких замикань – автомати, пробкові запобіжники в електропроводці повинні бути завжди справними. Заміна заводських запобіжників, навіть тимчасово, усілякими металевими провідниками може стати причиною нещасного випадку чи пожежі.

3. Заземлення – електроприлад повинен бути надійно заземлений згідно з правилами установки приладу.

4. Правильне підключення – використовуйте лише справні та сертифіковані прилади. Ніколи не підключайте одночасно багато пристроїв до одного подовжувача, щоб уникнути перевантаження електромережі. Необхідно електрошнур спочатку підключити до приладу, а потім до мережі, а не навпаки.

5. Захист від вологи – електронні пристрої не повинні використовуватися в умовах високої вологості або поблизу води, щоб уникнути короткого замикання. Забороняється працювати з електроприладом вологими руками.

6. Очищення від пилу – регулярно очищуйте пристрої від пилу, щоб уникнути перегрівання або загоряння.

7. Уникання перегріву – забезпечте достатню вентиляцію для пристроїв, щоб уникнути їх перегріву. Не накривайте прилади під час роботи.

8. Вимикання пристроїв – Коли пристрій не використовується, вимикайте його з розетки. Це знижує ризик виникнення пожежі або пошкодження приладу при скачках напруги.

9. Категорично заборонено виконувати будь-які ремонтні роботи самостійно.

Якщо вже так сталося, що електричне обладнання загорілося, то перш за все потрібно вимкнути живлення в приміщенні. Якщо знеструмити електромережу неможливо, то слід пам’ятати: не можна застосовувати для гасіння воду та пінні вогнегасники, можна лише порошкові.

Порятунок життя людини, ураженої струмом, у багатьох випадках залежить від швидкості та правильності дій осіб, що надають допомогу. Передусім потрібно якнайшвидше звільнити потерпілого від дії електричного струму. Якщо неможливо відключити електричне обладнання від мережі, потрібно одразу звільнити потерпілого від струмоведучих частин, не торкаючись при цьому потерпілого.

Під час робити з комп’ютерною технікою також потрібно керуватися Вимогами щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями (НПАОП 0.00-7.15-18) [40] та Державними санітарними правилами і нормами роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [41].

До роботи на комп’ютеризованому робочому місці з екранними пристроями допускаються працівники, що пройшли вступний та первинний інструктажі з питань охорони праці та пожежної безпеки.

Користувач ПЕОМ один раз на 6 місяців повинен проходити повторний інструктаж з питань охорони праці.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи спроєктовано та реалізовано програмно-алгоритмічні засоби для автоматизованого формування шаблонів обробки частково структурованих даних.

У першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

- Подано актуальність теми обробки частково структурованих даних.
- Розглянуто аналіз предметної області.
- Визначено вимоги до програмно-алгоритмічних засобів.
- Проаналізовано існуючі методи та підходи до обробки частково структурованих даних.

У другому розділі кваліфікаційної роботи:

- Досліджено потоки даних програмно-алгоритмічних засобів.
- Обґрунтовано використання алгоритмів для автоматизованого формування шаблонів обробки.
- Сформовано інтерфейси та методи взаємодії між компонентами системи.
- Запропоновано метод класифікації стовпців на основі нейронних мереж.
- Спроєктовано інфраструктуру контейнеризації для забезпечення ізоляції та масштабованості.

У третьому розділі кваліфікаційної роботи:

- Розроблено код реалізації отримання даних та їх обробки.
- Протестовано ефективність роботи розроблених засобів на реальних даних, де в результаті досягнуто 92.79% точності класифікації.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено принципи ергономіки робочого місця і електробезпеки про роботі з електронними приладами.

Розроблені програмно-алгоритмічні засоби розміщені на GitHub в репозиторії кафедри [42].

ПЕРЕЛІК ДЖЕРЕЛ

1. Henderson D. DAMA International. DAMA-DMBOK: Data Management Body of Knowledge (2nd ed.). / D. Henderson, S. Earley, L. Sebastian-Coleman., 2017.- 612 с.

2. Матеріали VII Міжнародної студентської науково - технічної конференції / Тернопіль: Тернопільський національний технічний університет ім. І.Пулюя (м. Тернопіль, 25-26 квітня 2024 р.), 2024.- 365 с.

3. ISO 3166-1:2020(en) Codes for the representation of names of countries and their subdivisions – Part 1: Country code [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.iso.org/obp/ui/en/#iso:std:72482:en.

4. E.123: Notation for national and international telephone numbers, e-mail addresses and web addresses [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: https://www.itu.int/rec/T-REC-E.123-200102-I/en.

5. Функціональні та нефункціональні вимоги [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.guru99.com/uk/functional-vs-non-functional-requirements.html?gpp&gpp_sid.

6. Діаграми потоків даних [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.maxzosim.com/data-flow-diagrams/.

7. Sharma G. Rest API: Data retrieval and applications / G. Sharma, G. Lavania, D. Goyal., 2023.- 128 с.

8. Документація FastAPI [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://fastapi.tiangolo.com/.

9. Web Framework Benchmarks [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.techempower.com/benchmarks/#section=data-r22&hw=ph&test=query&l=zijzen-6.

10. FastAPI First Steps [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://fastapi.tiangolo.com/tutorial/first-steps/.

11. What Is OpenAPI? [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://swagger.io/docs/specification/about/.

12. Lathkar M. High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python / Malhar Lathkar., 2023.- 253c.

13. Fielding R. HTTP Semantics [Електронний ресурс] / R. Fielding, M. Nottingham, J. Reschke. – 2022. – Режим доступу до ресурсу: https://httpwg.org/specs/rfc9110.html#status.422.

14. What Is Neural Network Architecture? [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://h2o.ai/wiki/neural-network-architectures/.

15. Функції активації: ступінчаста, лінійна, сигмоїда, ReLU та Tanh [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh.

16. Dropout Regularization in Deep Learning Models with Keras [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/.

17. A Simple Introduction to Softmax [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://medium.com/@hunter-j-phillips/a-simple-introduction-to-softmax-287712d69bac.

18. DevOps. Посібник: Як домогтися гнучкості, надійності й безпеки світового рівня в технологічних компаніях / Д.Кім, Д. Хамбл, П. Дебуа, Д. Віллис. – Харків: Фабула, 2023. – 384 с.

19. Shah A. FastAPI Handbook [Електронний ресурс] / Atharva Shah. – 2023. – Режим доступу до ресурсу: https://www.freecodecamp.org/news/fastapi-quickstart/.

20. ASGI (Asynchronous Server Gateway Interface) Specification [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://asgi.readthedocs.io/en/latest/specs/main.html.

21. Документація Uvicorn [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.uvicorn.org/.

22. Understanding the Levenshtein Distance Equation for Beginners [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:

https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0.

23. Pandas User Guide [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://pandas.pydata.org/docs/user_guide/basics.html.

24. Документація TensorFlow [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.tensorflow.org/.

25. Machine Learning – Evaluating classification models [Електронний ресурс]. – 2023. – Режим до ресурсу: https://medium.com/@brandon93.w/machine-learning-evaluating-classification-models-18713af3d764.

26. How to Use Python Data Classes [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.dataquest.io/blog/how-to-use-python-data-classes/.

27. Python dataclasses [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://docs.python.org/3/library/dataclasses.html.

28. Evaluating Multi-Class Classification Model using Confusion Matrix in Python [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://medium.com/@gubrani.sanya2/evaluating-multi-class-classification-model-using-confusion-matrix-in-python-4d9344084dfa.

29. V. Kozlovskyi, Y. Balanyuk, H. Martyniuk, O. Nazarevych, L. Scherbak and G. Shymchuk, «Information Technology for Estimating City Gas Consumption During the Year,» 2022 International Conference on Smart Information Systems and Technologies (SIST), Nur-Sultan, Kazakhstan, 2022, pp. 1-4, doi: 10.1109/SIST54437.2022.9945786.

30. Approach to gas consumption process forecasting on the basis of a mathematical model in the form of a random cyclic process / Serhii Lupenko, Iaroslav Lytvynenko, Oleg Nazarevych, Grigorii Shymchuk, Volodymyr Hotovych // ICAAEIT 2021, 15-17 December 2021. – Tern. : TNTU, Zhytomyr «Publishing house „Book-Druk“» LLC, 2021. – P. 213–219. – (Mathematical modeling in power engineering and information technologies).

31. Lytvynenko, S. Lupenko, O. Nazarevych, G. Shymchuk and V. Hotovych, «Mathematical model of gas consumption process in the form of cyclic random

process,» 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 2021, pp. 232-235, doi: 10.1109/CSIT52700.2021.9648621.

32. Additive mathematical model of gas consumption process / Iaroslav Lytvynenko, Serhii Lupenko, Oleh Nazarevych, Hryhorii Shymchuk, Volodymyr Hotovych // Scientific Journal of TNTU. – Tern. : TNTU, 2021. – Vol 104. – No 4. – P. 87–97.

33. O. Nazarevych, Y. Leshchyshyn, S. Lupenko, V. Hotovych, G. Shymchuk and N. Shabliy, «Method of Gas Consumption Change-point Detection Based on Seasonally Multicomponent Model,» 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020, pp. 152-155, doi: 10.1109/ACIT49673.2020.9208924.

34. Y. Leshchyshyn, L. Scherbak, O. Nazarevych, V. Gotovych, P. Tymkiv and G. Shymchuk, «Multicomponent Model of the Heart Rate Variability Change-point,» 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Polyana, Ukraine, 2019, pp. 110-113, doi: 10.1109/MEMSTECH.2019.8817379.

35. Лапін В. М. Безпека життєдіяльності людини / В. М. Лапін., 2007. – 332 с. – (6-те вид., перероб. і доп.).

36. Голобородько В. М. Вибрані глави проективної ергономіки. Антропометричний фактор: навчальний посібник / В. М. Голобородько., 2004. – 216 с.

37. ДСТУ 8604:2015. Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги [Текст]. - Чинний від 2017-07-01. - Київ : УкрНДНЦ, 2016. 7 с. (Національний стандарт України).

38. ДСТУ 7299:2013. Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки [Текст]. - Чинний від 2014-01-01. - Київ : УкрНДНЦ, 2014. 4 с. (Національний стандарт України)

39. ДСТУ 7234:2011. Дизайн і ергономіка. Обладнання виробниче. Загальні вимоги дизайну та ергономіки. [Текст]. - Чинний від 2011-08-01. - Київ : УкрНДНЦ, 2011. 7 с. (Національний стандарт України)

40. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями [Текст]. - Чинний від 2011-08-01. - Київ : УкрНДНЦ, 2018. 5 с. (Національний стандарт України)

41. ДСанПІН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Текст]. - Чинний від 1998-12-10. - Київ : УкрНДНЦ, 1998. 20 с. (Національний стандарт України)

42. Репозиторій GitHub [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://github.com/TNTU-122-Computer-Science/FIS-CS-SN-41-RGashynskyi-2023-2024-Thesis_paper.

ДОДАТКИ

Додаток А

Тези конференції

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2024 р.

(збірник тез конференції)

Тернопіль 2024

Марчук Д. **РОЗРОБКА БЛОГ-ПЛАТФОРМИ З ВИКОРИСТАННЯМ REACT ТА NODE.JS**	**326**
Ільїн В. **СУЧАСНІ СИСТЕМИ ОПЛАТ З ВИКОРИСТАННЯМ КЛЮЧІВ ІНДПОТЕНТНОСТІ**	**327**
Соловій Т. **ОПТИМІЗАЦІЯ УПРАВЛІННЯ ДАНИМИ В ІНДУСТРІЇ ГАЗОПОСТАЧАННЯ**	**329**
Марків К. **РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КРУЇЗНОЇ КОМПАНІЇ З ВИКОРИСТАННЯМ SPRING FRAMEWORK**	**331**
Музика В. **АСИНХРОННІСТЬ ПРОГРАМУВАННЯ В РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**	**332**
Петришин Я., Марцинюк Я. **РОЛЬ CI/CD У ПІДВИЩЕННІ ЕФЕКТИВНОСТІ ТА НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**	**333**
Бурса В., Мудрик І. **АВТОМАТИЗОВАНА СИСТЕМА ОЦІНКИ РІВНЯ ЗНАННЯ ІНОЗЕМНОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ**	**335**
Антонюк Д., Пастух О. **АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АНСАМБЛЮВАННЯ АЛГОРИТМІВ**	**336**
Чорна Х., Пастух О. **ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУ ПАСАЖИРОПЕРЕВЕЗЕНЬ**	**337**
Бабинець К., Пастух О. **ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ**	**338**
Бойко Д., Пастух О. **ПРОГРАМА ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ ЩОДО КРЕДИТУВАННЯ**	**339**
Яковів Б., Пастух О. **ПРОГРАМНА СИСТЕМА ДЛЯ НЕВРОЛОГІЧНОЇ ДІАГНОСТИКИ**	**340**
Гашинський Р., Мельник Н. **АВТОМАТИЗОВАНИЙ АНАЛІЗ СХЕМИ ДОКУМЕНТІВ З ЧАСТКОВО-СТРУКТУРОВАНИМИ ДАНИМИ**	**341**

УДК 004.67
Гашинський Р. – ст. гр. СН-41, Мельник Н. – ст. гр. СІм-52
Тернопільський національний технічний університет імені Івана Пулюя

АВТОМАТИЗОВАНИЙ АНАЛІЗ СХЕМИ ДОКУМЕНТІВ З ЧАСТКОВО-СТРУКТУРОВАНИМИ ДАНИМИ

Науковий керівник: асистент Кравчук Г. Б.

Gashynskyi R., Melnyk N.
Ternopil Ivan Puluj National Technical University

AUTOMATED ANALYSIS OF THE SCHEMA OF DOCUMENTS WITH SEMI-STRUCTURED DATA

Supervisor: assistant Kravchuk H.

Ключові слова: дані, метадані, менеджмент даних
Keywords: data, metadata, data management

In today's information society, where data rapidly increases in volume and becomes more diverse, ensuring high quality of the data is important to provide solid foundations for management and business decisions. Most organizations work with data that can be both a business object and a result of the activities of the organizations themselves. If an organization can't trust data to meet business needs, all efforts spent on collecting, storing, protecting, and making it available will be wasted [1].

Usually data comes in the form of documents from different sources and in different formats. The lack of a common document schema can create significant problems for data processing and further analysis. The problem of semi-structured data [2] manifests itself in differences in data storage formats, inconsistencies in the nomenclature of columns in tables. For example, one data source may use the date format "mm/dd/yyyy" and another data source may use "dd/mm/yyyy". Such differences in formats not only complicate the data processing, but also significantly increase the risk of errors during their analysis. If the schema of the input document does not correspond to the internal schema of documents in the organization, bringing it to the proper format is one of the tasks of data management.

There are solutions that work with the document schema, but for them you need to specify a specific algorithm for processing. Also, the specifics of the data lifecycle in a particular organization can be complex, as data can have different lineage. The better an organization understands the lifecycle and lineage of data, the better it can manage its data [1].

The solution to the described problems is an automated analysis of the schema of incoming documents. For semi-structured data, this schema is the list of input columns, their content, compliance with business needs. To bring the incoming document to a fixed structure, instructions are to be created on how the incoming columns should correspond to the columns of the organization’s internal document schema, the content of each of them is to be reviewed to match the request, if necessary, and the number of rows is to be counted and saved in a commonly accepted format, for example JSON (Fig. 1).

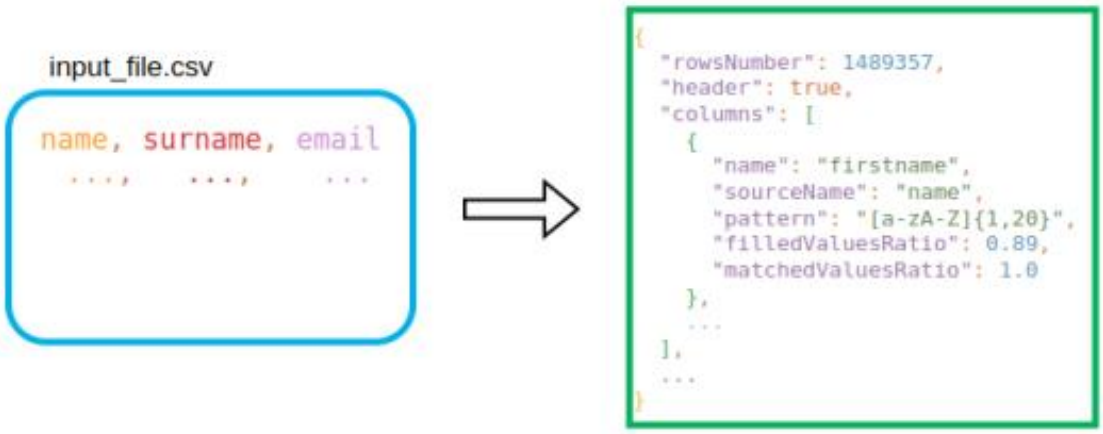


Figure 1. The result of document schema analysis

The result of the analysis will be a recipe that consists of two parts: a description of the schema of the input file, which ensures the preservation of the lineage of the data, and recommendations for transformation. It can be used to make decisions and further process data to align it with the needs of the organization. Automating this process reduces the time required to integrate input data sources and the human risks associated with manual analysis.

The proposed approach has a list of advantages over the analogs [3, 4]. Firstly, it can be easily customized according to the customer's needs and the peculiarities of the business domain. Secondly, it is transparent enough to exclude possible cyber threats connected with data processing in business analytics-oriented applications by large proprietary platforms, where data wrangling and processing is hidden from the customer and can not be traced appropriately. Besides, the developed pipeline imposes no restriction on external datastore connectivity

Possible applications of the developed pipeline range from scientific databases and digital libraries to on-line documentations and electronic commerce. Adoption of the proposed approach will significantly improve the subsequent data validation and querying as well as reduce security and privacy risks, enabling opportunities for efficient backup and disaster recovery.

References:

1. DAMA International. (2017). DAMA-DMBOK: Data Management Body of Knowledge (2nd ed.). Technics Publications.
2. S. Abiteboul (2009). Semi-Structured Data. In: Liu L., Özsu M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_799
3. Altair Monarch. URL: https://altair.com/monarch/, accessed 1 May 2024.
4. Alteryx. URL: https://www.alteryx.com/products/alteryx-platform, accessed 1 May 2024.

Додаток Б

Код порівняння схожості текстових полів за допомогою відстані Левенштейна

```
def levenshtein_distance(s1, s2):
    if len(s1) < len(s2):
        return levenshtein_distance(s2, s1)

    if len(s2) == 0:
        return len(s1)

    previous_row = range(len(s2) + 1)
    for i, c1 in enumerate(s1):
        current_row = [i + 1]
        for j, c2 in enumerate(s2):
            insertions = previous_row[j + 1] + 1
            deletions = current_row[j] + 1
            substitutions = previous_row[j] + (c1 != c2)
            current_row.append(min(insertions, deletions,
substitutions))
        previous_row = current_row

    return previous_row[-1]


def text_similarity(s1, s2):
    distance = levenshtein_distance(s1, s2)
    max_len = max(len(s1), len(s2))
    similarity = (max_len - distance) / max_len
    return similarity
```