ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика синтезу шрифту за зразком на основі методів розпізнавання зображень»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Денис ЖУЖКОВ
(підпис)

Виконав: здобувач вищої освіти група ПДМ-62
Денис ЖУЖКОВ

Керівник: Андрій БОНДАРЧУК
д.т.н., професор

Рецензент: Наталія ЛАЩЕВСЬКА
к.т.н. доцент

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Жужкову Денису Ігоровичу

1. Тема кваліфікаційної роботи: «Методика синтезу шрифту за зразком на основі методів розпізнавання зображень»

керівник кваліфікаційної роботи Андрій БОНДАРЧУК, д.т.н., професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк подання кваліфікаційної роботи «17» грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
3.1. Програми для розробки: Visual Studio Code, Figma, Chrome;
3.2. Вхідні поняття: OCR-Система, Синтез;
3.3. Інформаційні ресурси: tesseract, nodejs.org Npm.com.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1 Аналіз існуючих додатків на наявність помилок.
4.2 Аналіз OCR системи для покращення синтезу.

4.3 Розробка інтерфейсу для редагування зображень використовуючи однорідний стиль.

5. Перелік ілюстративного матеріалу: *презентація*

1. Аналіз існуючих додатків.
2. Характерні ознаки для пошуку відмінностей.
3. Методика OCR-системи та вдосконалення.
4. Методика класифікації на розбивку даних шрифтів за окремими характеристиками
5. Вдосконалення постобробки за рахунок додаткової сегментації окремих елементів на основі вказаних значень
6. Принцип забезпечення однорідності при редагуванні документу

6. Дата видачі завдання «16» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз існуючих додатків для розпізнавання шрифтів	16.10-23.10.2024	
2	Аналіз можливих покращень	24.10-30.10.2024	
3	Розробка покращень синтезу	31.10-05.11.2024	
4	Розробка покращень знаходження релевантних шрифтів за рахунок класифікації баз даних	06.11-15.11.2024	
5	Розробка інтерфейсу редагування зображень	16.11-20.11.2024	
6	Розробка редагування тексту в зображенні за рахунок синтезованих елементів	21.11-25.11.2024	
7	Оформлення роботи: вступ, висновки, реферат	26.11-27.11.2024	
8	Розробка демонстраційних матеріалів	28.11-29.11.2024	
9	Попередній захист роботи	30.11.2024	

Здобувач вищої освіти _____ Денис ЖУЖКОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис)

РЕФЕРАТ

Текстова частина бакалаврської роботи 67 стор., 10 рис., 45 джерел.

Об'єкт дослідження (розробки) – процес підбору релевантних шрифтів за зразком

Предмет дослідження (розробки) – методи аналізу елементів тексту.

Мета роботи – удосконалення використання релевантних варіантів шрифтів за рахунок методів підбору шрифту та розпізнавання тексту.

Методи дослідження(Розробки) – включають аналіз існуючих додатків, знаходження проблем, оптимізація роботи, аналіз уже наявних рішень порівнянням їхньої ефективності під час розпізнавання шрифтів. Експерименти з різними типами вхідних зображень, щоб з'ясувати вплив шумів та артефактів на точність сегментації. Перевірка різних підходів до відновлення літерних контурів: від покрокової сегментації до глибинних нейронних мереж. Результати цієї перевірки стали основою для створення синтезованих елементів, які зберігають унікальні особливості оригінальних символів. Тестування інтеграції отриманих контурів у веб-середовище, аби оцінити їх сумісність з різними платформами та форматами.

Галузь використання – Обробка зображення, електронний документообіг аналіз текс.

КЛЮЧОВІ СЛОВА: ФОРМУВАННЯ ШРИФТІВ; СІМЕЙСТВА ШРИФТІВ; ГЕНЕРАЦІЯ ШРИФТІВ; РОЗПІЗНАВАННЯ ТЕКСТ; РОЗПІЗНАВАННЯ ШРИФТІВ.

ABSTRACT

Text part of the bachelor's thesis 67 p., 10 fig., 45 sources.

Object of research (development) – the process of selecting relevant fonts by sample.

Subject of research (development) – methods of analyzing text elements.

Purpose of the work – is to improve the use of relevant font options through font selection and text recognition methods.

Research (Development) methods – include analysis of existing applications, finding problems, optimizing work, analysis of existing solutions by comparing their effectiveness during font recognition. Experiments with different types of input images to find out the impact of noise and artifacts on the accuracy of segmentation. Testing of different approaches to restoring letter contours: from step-by-step segmentation to deep neural networks. The results of this test became the basis for creating synthesized elements that retain the unique features of the original characters. Testing the integration of the resulting contours into the web environment to assess their compatibility with different platforms and formats.

Field of application – Image processing, electronic document management, text analysis.

KEYWORDS: FONT FORMATION; FONT FAMILIES; FONT GENERATION; TEXT RECOGNITION; FONT RECOGNITION.

ЗМІСТ

ВСТУП

У сучасному дизайні та друкарстві шрифти відіграють одну з ключових ролей. Вони не тільки відображають зміст написаного, а й формують певний настрій, стиль, зручність читання й навіть ідентичність бренду. Саме тому фахівці візуальних комунікацій часто приділяють величезну увагу вибору чи розробленню шрифтових рішень. Коли зображення містить унікальні або рідкісні шрифти, виникає потреба ідентифікувати їх, відтворити чи згенерувати схожі, щоб застосувати ці шрифти повторно або ж адаптувати під нові формати.

Актуальність дослідження зумовлена кількома чинниками. По-перше, шрифт може бути частиною фірмового стилю компанії: всі рекламні матеріали, веб-сайти, друкована продукція тощо мають виглядати узгоджено, й однакове шрифтове рішення цьому неабияк сприяє. По-друге, в окремих випадках, особливо в художньо-оформлювальній діяльності, потрібні унікальні шрифти для передачі емоцій, створення особливого образу, незабутньої атмосфери чи оригінального дизайну. По-третє, в епоху діджиталізації надзвичайно важливо, аби тексти залишалися читабельними й приємними, чи то на папері, чи на мобільному пристрої з невеликим екраном.

Об’єкт дослідження: процес підбору релевантних шрифтів за зразком.

Предмет дослідження: методи аналізу елементів тексту.

Мета роботи: забезпечення однорідності тексту при редагуванні зображення

Завдання розробки: в процесі підбору та синтезу тексту вирішувалися такі завдання:

1. Аналіз головних проблем для підбору шрифту
2. Розробка панелі керування, в якій є можливість завантажити зображення
3. Синтез елементів на основі тексту в зображенні
4. Підбір шрифтів для синтезу відсутніх елементів

Ідея створення шрифтів дало свій початок ще з розвитком писемності де кожна літера мала свої певні характеристики. Їх почали створювати з кількох дуже важливих причин, таких як:

- Стандартизація письма
- Необхідність збереження знань
- Естетика та дизайн
- Розвиток друкарства

Стандартизація письма - є основною необхідністю для розуміння та сприйняття інформації, оскільки з давніх часів людям необхідно було вести облік та якісь розрахунки для торгівлі різними видами товарів.

І щоб можна було дохідливо передати інформацію іншим - треба було щоб всі мали розуміння того що було написано на дошці, папірусі, листку паперу чи вже в наш час на екрані монітору. Якщо б кожен описував свої розрахунки по своєму це б спричинило повний хаос, не було б розуміння що означає кожен символ. Гарним прикладом буде ведення обліку якихось окремих зернових культур або іншої їжі які мають різний термін придатності. Якщо б не було чіткого розуміння скільки і яких ресурсів вистачить на певний термін часу, це б могло спричинити катастрофічні значення. Клинопис і єгипетські ієрогліфи були першими стандартизованими формами письма, що давали можливість передавати інформацію у вигляді різних символів та давали розуміння для широкого кола людей. Давали можливість ввести статистику та введення торгівлі та збереження релігійних текстів та законів.

Необхідність збереження знань є основою розвитку і тому дуже важливо було передавати інформацію новому поколінню. Завдяки розширенню письмових культур у середньовічній Європі виникла потреба зберігати знання через книги. Ченці у монастирях копіювали релігійні тексти, тому стандартизація форм літер допомагала зберігати консистентність у рукописах. Каролінгський мінускул – перший стандартизований шрифт у Європі, був створений для того, щоб зробити тексти більш читабельними і єдиними.

Згодом шрифти стали не тільки функціональним елементом, але й інструментом дизайну. У Ренесансі та пізніше друкарі і художники почали створювати нові шрифти, щоб відобразити естетичні смаки своєї епохи.

З розвитком друкарства ключовою причиною створення шрифтів став винахід друкарського верстата Йоганном Гутенбергом у середині XV століття.

Оскільки з часом написання текстів вручну було складною задачею і вимагає багато часу і окремо навчаних на це писарів де додатково кожен мав свій певний почерк.

Це дуже суттєво змінило ситуацію з переданням інформації і дала значний розвиток в суспільстві що повпливало на всі технології.

Щоб друкувати тексти швидко та масово, потрібно було створити набір типографських літер, що дозволяло б механічно відтворювати тексти. Так само кожна літера мала свої певні характеристики. Винахід рухомих літер дозволив друкарям використовувати різні шрифти для кожної сторінки, комбінуючи їх у різних послідовностях.

Це був перший крок у стандартизації шрифтів і типографії. Технологія полягала в тому, що літери виготовлялися з металу, найчастіше свинцю, і могли використовуватися багаторазово. Це дозволило створювати шрифти різних розмірів і стилів для книг, що заклало основу для сучасної типографії.

Кожен друкар мав свої набори шрифтів різних стилів, таких як антиква, готика та інші, які використовувалися в залежності від типу видання. Це були ранні зразки типографської естетики. Типографія почала розвиватися як мистецтво, і друкарі стали приділяти більше уваги гармонійному поєднанню шрифтів на сторінці, створенню візуального контрасту та читабельності. Шрифти, такі як Гарамонд, Бодоні та Каслон, стали зразками художнього оформлення, які не тільки поліпшували читабельність, але й надавали текстам певного стилю і витонченості.

З появою парових і ротаційних друкарських верстатів у 19 столітті виникла потреба у стандартизації шрифтів, щоб вони відповідали вимогам масового друку. Популярними стали такі шрифти, як Times New Roman, який розроблявся для

газетного друку. Великий тираж книг і газет вимагав чітких, простих шрифтів, які легко читаються при швидкому перегляді сторінок.

Це сприяло поширенню таких шрифтів, як антиква, що мають чіткі лінії та легко читаються навіть при невеликому кеглі. З розвитком друкарства і друкарень шрифти дозволили швидко і дешево поширювати інформацію серед великих груп людей.

Це було важливим фактором у розвитку таких явищ, як Реформація, Просвітництво та наукові відкриття. Офсетний друк дозволив використовувати більшу кількість шрифтів різних стилів, розмірів та ваги без втрати якості. Це дало новий поштовх для творчих експериментів з типографією.

Такі верстати дозволяли відтворювати більш складні шрифтові композиції, а також використовувати кольоровий друк, що відкрило нові можливості для типографічного дизайну в рекламних матеріалах і журналах.

З появою комп’ютерних технологій шрифти стали не лише інструментом для друку, а й важливим елементом цифрових інтерфейсів, веб-сайтів та мобільних додатків. Потреба у швидкому завантаженні текстів, адаптація до різних розмірів екранів і різних мов стимулювала створення нових шрифтів.

Безсерифні шрифти, такі як Arial і Helvetica, стали популярними через їх простоту і читабельність на екранах. Шрифти почали створювати через прагнення до стандартизації письма, поліпшення читабельності та ефективної передачі інформації.

З розвитком технологій і культури шрифти перетворилися на потужний інструмент як для поширення знань, так і для вираження стилю та дизайну.

Це дало змогу використовувати комп’ютерні шрифти без обмежень у кількості та стилях. Друкарі й дизайнери можуть вибирати будь-який шрифт для створення макетів, працюючи з високою точністю до кожної літери.

Завдяки цифровим технологіям стало можливим контролювати деталі типографії на дуже тонкому рівні, включно з кернингом тобто відстанню між буквами, відстеженням відстанню між словами та іншими параметрами, що дозволяють створювати ідеально збалансований текст.

З появою PDF-документів документи з будь то якісь підручники статті чи журнали зберігалися умовно як частини зображення і не було можливості їх редагування. Але в майбутньому може з’явитися потреба в самому редагуванні таких текстів з урахуванням однорідності, а значить і використанням однакових шрифтів.

Зараз вже є багато ресурсів які дають змогу редагування, аналізу та розпізнавання тексту. Але самі шрифти підбираються дуже погано. Оскільки досконало підібрати підходящий шрифт стало дуже проблематично у зв’язку з великою кількістю створених.

Проаналізувавши існуючі додатки було виявлено що вони дають зовсім не підходящі результати. Причому це явно видно по декількох основних ознак за якими можна відрізнити літери.

1 АНАЛОГИ ТА ПРИНЦИПИ ЇХ ДІЇ

1.1 Огляд сучасних аналогів

Проаналізувавши найпопулярніші додатки можна сказати що вони в принципі мають не досконалий підбір правильних значень. Причому головні характеристики за якими можна знайти щось спільне не використовуються. За основу можна взяти рейтинг найпопулярніших додатків які зараз існують, по кожному був проведений детальний аналіз їх роботи та додаткові можливості. Основні додатки для дослідження такі:

1. WhatTheFont - популярний сервіс для визначення шрифтів який використовує технології розпізнавання, щоб допомогти визначити шрифт, використаний на зображенні. Він працює по такому принципу, що користувач завантажує зображення, на якому є текст. Це може бути фотографія, скріншот або будь-яке інше зображення.

Додаток використовує технологію оптичного розпізнавання символів (OCR), щоб аналізувати текст на зображенні. Програма розпізнає окремі літери та символи на зображенні.

Після розпізнавання символів, сервіс порівнює отриманий текст із великою базою даних шрифтів. База включає сотні шрифтів із різними варіаціями стилів. Далі надає список шрифтів, схожих на ті, що розпізнані на зображенні, з посиланнями на сайти, де їх можна придбати або завантажити.

2. WhatFontIs - теж онлайн сервіс який так само працює за технологією для розпізнавання зображень і базою даних. Початковий принцип такий самий і використовує технологією (OCR) але він додатково дає можливість вручну вказати декілька символів на зображенні, якщо текст дуже складний для автоматичного

розпізнавання. На основі порівняння з базою даних, сервіс генерує список шрифтів, які найбільше відповідають розпізнаному тексту. Для кожного шрифту надається інформація про його назву, посилання для завантаження або покупки, а також варіанти стилів, якщо такі є.

3. FontSquirrel - надає велику бібліотеку шрифтів, серед яких більшість є безкоштовними для комерційного використання. Він надає можливість переглядати шрифти за категоріями наприклад засічкові, беззасічкові, рукописні, декоративні. Шукати за параметрами: можна фільтрувати шрифти за стилем, популярністю, призначенням тощо. Оцінювати ліцензію: на сайті чітко вказано, чи можна використовувати шрифт для комерційних цілей. Він має вбудований інструмент для розпізнавання шрифтів, який має назву Font Identifier, який працює за схожим принципом до WhatTheFont.

Також пропонує інструмент Webfont Generator, який дозволяє конвертувати шрифти в різні формати для веб-дизайну:

Користувач може завантажити свої шрифти (в форматах, таких як .otf або .ttf)

Інструмент автоматично конвертує ці шрифти в необхідні формати для вебу, такі як .woff, .woff2, .eot, .svg, а також генерує CSS код для використання шрифтів на веб-сторінках.

Система перевіряє, чи є у шрифта ліцензія для комерційного використання, і попереджає користувача, якщо ліцензія відсутня або обмежена.

4. Fontspring - інструмент застосовує технології розпізнавання зображень та аналізує шрифт на зображенні, щоб виявити індивідуальні символи. Після розпізнавання тексту система автоматично шукає співвідношення між розпізнаними буквами і своїми шрифтами в базі даних. Fontspring порівнює отримані дані з великою бібліотекою шрифтів, щоб знайти точний або подібний шрифт. Система надає список найбільш відповідних шрифтів, з можливістю перегляду їх варіантів наприклад, різних стилів і ваг.

Якщо система не може точно визначити шрифт, користувач може вручну підказати систему, виділивши правильні літери на зображенні. В результатах пошуку можна відфільтрувати шрифти за стилями, такими як беззасічкові, засічкові, декоративні тощо.

5. AdobeFonts - дає можливість завантажити зображення також ввести назву шрифта або описову характеристику. Також можна додати певні уточнення такі як:

Категорії шрифтів: засічкові, беззасічкові, рукописні, декоративні та інші.

Мови: можна обирати шрифти, що підтримують певні мови чи алфавіти.

Товщина та стилі: Пошук шрифтів за товщиною, стилем, накресленням (наприклад Regular, Bold, Italic).

Вага шрифта: Light, Medium, Bold.

Параметри використання: Пошук за вимогами, чи підходить шрифт для веб-дизайну, друку, мобільних додатків тощо.

Він надає можливість додатково мати ще розширений пошук за допомогою додаткових фільтрацій пов’язаних із сімейством шрифтів та стилем типу рукописного, геометричного чи класичного. Додатково в Adobe вбудована інтеграція з сервісами такими як Photoshop, Illustrator, InDesign, Premiere Pro, та іншими програмами Creative Cloud.

Adobe Fonts також використовує технології для розпізнавання шрифтів із зображень. Однак цей функціонал, на відміну від WhatTheFont, більше спрямований на порівняння стилістичних схожостей, а не точне розпізнавання шрифта. Тобто можна завантажити зображення або використати текст на зображенні для пошуку шрифтів, схожих за стилем.

Adobe Fonts дозволяє активувати шрифти для подальшого використання в програмах, але на відміну від традиційних шрифтів, їх не можна завантажити на комп’ютер у вигляді файлів .otf або .ttf. Замість цього шрифти можна синхронізувати безпосередньо з Adobe Creative Cloud, і вони стають доступними в додатках на платформі.

Пошук шрифтів в Adobe Fonts орієнтований на інтеграцію з продуктами Adobe і надання користувачам максимально зручного інструменту для швидкого пошуку, тестування та використання шрифтів у професійних проектах.

Таким чином WhatTheFont і WhatFontIs орієнтуються для швидкого розпізнавання шрифтів із зображень.

FontSquirrel корисний для тих, хто шукає безкоштовні шрифти з можливістю конвертації для вебу.

Fontspring і Adobe Fonts орієнтовані на професіоналів, пропонуючи ліцензовані шрифти для різних типів проектів з високою зручністю інтеграції.

1.2 Спосіб дослідження ефективності

Для дослідження ефективності будуть підібрані скріншоти із популярними шрифтами які мають різні типи складності. Основними типами складності будуть враховуватися популярність шрифту, чіткість зображення та можливий колір, різна жирність. Оскільки враховується потреба тестуванні реальних кейсах дані зображення не будуть в собі містити якихось графічних навантажень таких як перекреслення тексту чи можлива кривизна строки.

Головними критеріями для підтвердження правильності знайденого варіанту буде запропоновано в першу чергу орієнтуватися на головні розпізнавальні характеристики певних літер оскільки деякі літери мають дуже значні розбіжності і особливо це чітко видно в дизайні літері. Для прикладу будуть використані: літера «g» як зображено на рисунку 1.1 де вона внизу має досить скруглений хвостик. Вона відноситься до сімейства - serif. Це тип шрифтів, у яких символи мають невеликі штрихи або «зарубки» на кінцях основних ліній. Ці зарубки можуть бути як більш різкими, так і м’якими, залежно від стилю шрифту. Вони надають тексту більш класичний, формальний вигляд і часто використовуються у друкованих матеріалах, таких як книги, газети, журнали, а також у дизайні веб-сайтів, де потрібна висока читабельність.

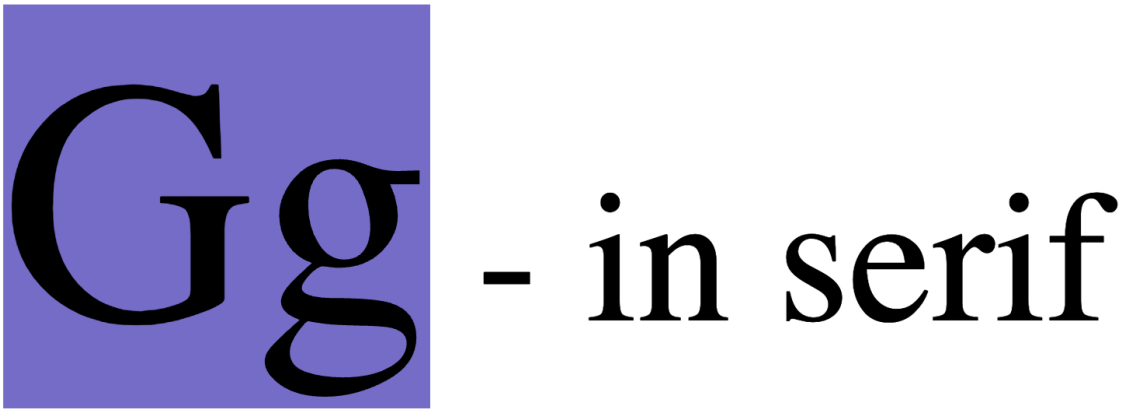


Рис 1.1 Приклад літери «g» в сімействі serif

Основними популярними шрифтами для такого сімейства є:

Times New Roman – класичний шрифт, часто використовуваний у друкованих та веб текстах.

Georgia – популярний для веб сайтів через гарну читабельність на екранах.

Garamond – елегантний і більш витончений шрифт з класичними формами.

Palatino – розроблений для використання у книгодрукуванні, має широкі й відкриті форми.

Baskerville – відомий своїми контрастами між тонкими і товстими лініями, що додає тексту вишуканості.

Шрифти цього сімейства зазвичай використовуються в довгих текстах, оскільки зарубки полегшують сприйняття і ведення рядка оком читача.

Сімейство шрифтів Sans-serif - це шрифти без зарубок як на прикладі з літерою “g” які зображені на рисунку 1.2 (serifs), тобто без маленьких декоративних штрихів на кінцях ліній. Ці шрифти мають чисті, прості лінії, що робить їх сучасними та мінімалістичними. Вони часто використовуються у веб дизайні, інтерфейсах програм та мобільних додатках, а також у друкованих матеріалах, де важлива чіткість і простота.

Рис. 1.2 Зображення літери «g» у sans-serif

Шрифти Sans-serif зазвичай використовуються в заголовках, інтерфейсах, логотипах, презентаціях і в текстах, що мають відображатися на екранах, оскільки вони забезпечують високу читабельність і простоту сприйняття. Основними представниками даного шрифту будуть:

Arial – універсальний шрифт, один із найпоширеніших у комп'ютерних системах та вебдизайні.

Helvetica – один із найвідоміших sans-serif шрифтів, відомий своєю нейтральністю і балансом.

Roboto – часто використовується в інтерфейсах Android, має сучасний вигляд і добре читається на екрані.

Open Sans - популярний вебшрифт, відзначається гарною читабельністю навіть при малих розмірах.

Futura – геометричний шрифт з простими й чіткими формами, популярний у графічному дизайні та рекламі.

Lato — елегантний шрифт з сучасним дизайном, також часто використовується в інтерфейсах та веб сайтах.

Але так само важливо додати що кожна літера має свої додаткові характеристики, такі як засєчки чи радіус похилу елементу літери.

1.3 Оцінка дослідження

Для оцінки дослідження будуть використані рівні складності де буде враховуватися популярність шрифту. Його легкість бо якщо взяти рукописний шрифт то тут його аналіз потребує набагато більше обчислень для будь якого сервісу. І його чіткість зображення, оскільки якщо взяти до прикладу якесь старе зображення популярного підручника з англійської мови то зазвичай в них зображення має певні розмитості що в свою чергу ускладнює процес розпізнавання.

Ці оцінки будуть класифікуватися по рівнях де найпростіший рівень буде потребувати найменш складні зображення без додаткових вад чи різкості.

В наведеному нижче зображенні буде запропоновано такий формат тексту в якому будуть використовуватися такі шрифти як Times New Roman, Arial, Comic Sans та Pacifico вони матимуть найнижчій рівень складності оскільки буде використано зображення тексту на білому фоні де контрастність кольорів дуже помітна на рисунку 1.3. Також рівнем ускладнення буде підтримка кирилиці, та оцінка шрифтів, для чистоти оцінювання буде використаний текст з достатньою кількістю літер для оцінки.

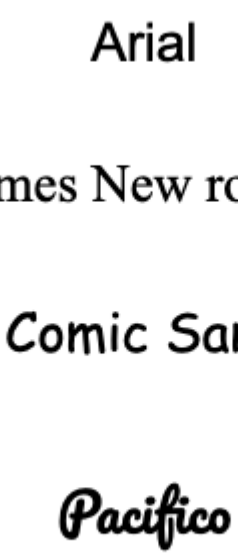


Рис. 1.3 Зображення шрифтів першого рівня складності

В наступному рівні будуть використанні скріншоти тексту з популярних сервісів, які можуть мати додаткові властивості такі як стилі кольору розміру чи жирності. Також буде враховуватись що популярність шрифту буде не така висока, де в даному випадку буде додатково оцінена схожість та відношення співпадінь.

Для третього рівня складності будуть використанні зображення фотографій тексту, де зазвичай різкість дуже втрачається або зображення може бути не чітки, як зображено на рисунку 1.4. Важливо щоб шрифт хоча б міг бути наближеним до того який є в зображенні, міг підібрати потрібне сімейство та візуальні ознаки.

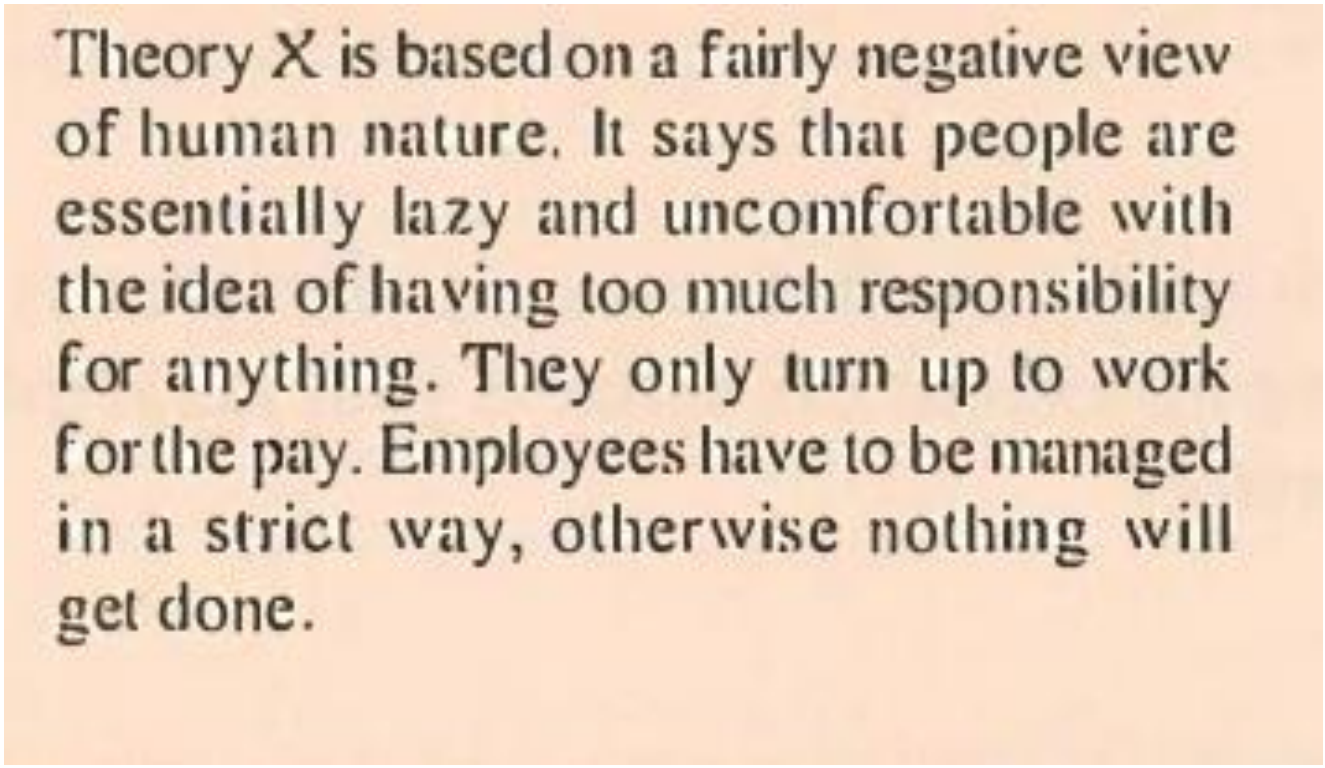


Рис.1.4 Зображення шрифтів 3 рівня складності

Для чистоти оцінювання буде використаний однаковий текст, який складається з літер вистроєних в алфавітному порядку де всі літери написані з маленької літери: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

На рисунку 1.5 в червоних колах зображено основні відмінності один від одного.

По цим зображенням можна буде розрізнити на око, і побачити основні характеристики. В яких зазвичай є основні літери які мають головні розбіжності – це літери такі як: a, g, k, y – в яких чіткіше всього можна їх побачити.

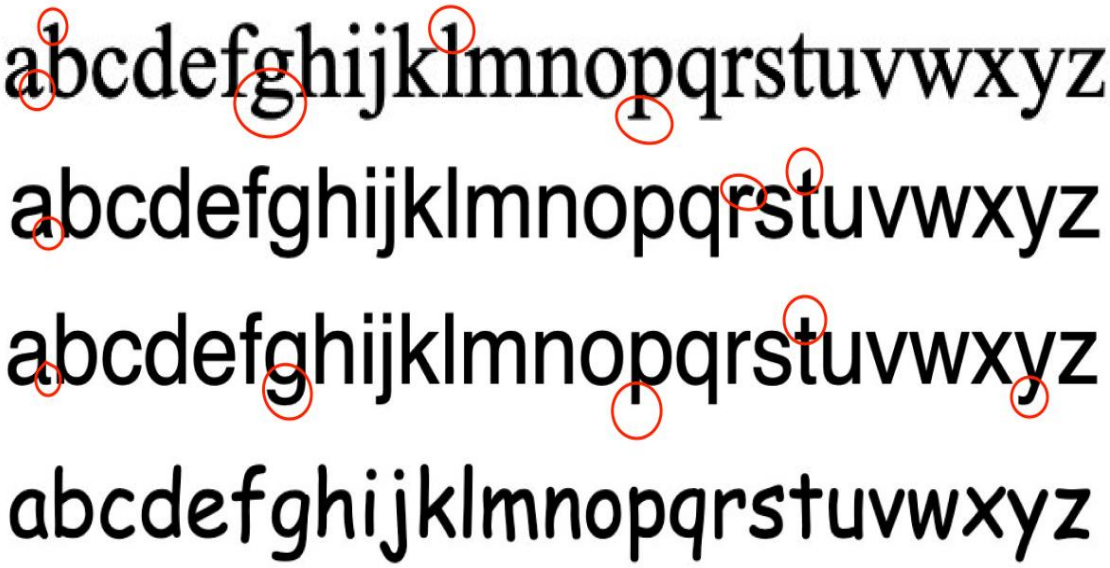


Рис.1.5 Зображення різних сімейств шрифтів

Ці шрифти були вибрані за аналізом на 2023 рік від ресурсу від linearity де чітко визначається що шрифти із засічками використовуються приблизно в 70% друкованих газет для кращої читабельності, а також приблизно 80% наукових журналів використовують традиційні шрифти із засічками, сам рейтинг можна побачити на рисуноку 1.6.

Без засічок використовуються приблизно 60% сучасних кіноафіш.

Понад 85% публічних вивісок використовують шрифти без засічок для кращої видимості.

Назва шрифту	Відсоток використання	Популярний у промисловості	Рік створення
Helvetica	25%	Графічний дизайн	1957 рік
Times New Roman	20%	Видавництво	1932 рік
Arial	15%	Бізнес	1982 рік
Comic Sans	10%	Повсякденний	1994 рік
Робото	10%	Веб-дизайн	2011 рік
Кур'єр Новий	5%	Сценарна робота	1955 рік
Грузія	5%	Цифрові медіа	1993 рік
Калібрі	5%	Офісне використання	2007 рік
Вердана	5%	Інтернет-контент	1996 рік

Рис.1.6 Зображення статистики популярності шрифтів від linearity

Далі будуть показані результати дослідження різних додатків для з’ясування основних проблем і неточностей для подальшої роботи.

1.4 Дослідження додатків

Аналіз шрифтів за допомогою додатків для пошуку шрифтів є важливим етапом для забезпечення точності та відповідності еталонному зразку. Під час використання різних веб додатків для автоматизованого підбору шрифтів, таких як: myfonts, fontsquirrel, fonts.adobe, fontspring, whatfontis.

Необхідно проводити детальну оцінку результатів, щоб знайти найбільш відповідний варіант. У цьому випадку було проведено аналіз шрифтів за допомогою шести різних додатків, кожен з яких запропонував близько 60 шрифтів, що теоретично могли відповідати оригінальному зразку. У процесі дослідження було знайдено варіанти, які мали від 0% до 75% подібності із заданим шрифтом, однак процес оцінювання передбачав ретельне порівняння різних параметрів для того, щоб визначити найбільш підходящі варіанти.

WhatFontIs — один із провідних додатків, що дозволяє ідентифікувати шрифти на основі завантаженого зображення або сканованого зразка. У процесі пошуку та аналізу шрифтів за допомогою цього інструменту алгоритм опрацьовує завантажене зображення, визначає ключові особливості шрифту, такі як форма літер, пропорції, міжрядковий інтервал та інші типографічні особливості, а потім пропонує кілька варіантів шрифтів з бази даних, які можуть відповідати зразку.

У цьому дослідженні було підготовлено чіткий зразок шрифту у форматі зображення, достатньо великого для того, щоб система могла коректно розпізнати всі деталі символів. Після завантаження зразка у WhatFontIs було отримано список з 60 можливих варіантів шрифтів, які могли б відповідати еталону. Додаток забезпечив кожен варіант шрифту невеликим попереднім переглядом, що дало можливість одразу ж оцінити зовнішній вигляд кожного з варіантів.

Основним критерієм для оцінювання була точність відповідності знайдених шрифтів оригінальному зразку. Візуальна подібність шрифту включає такі аспекти, як форма букв, пропорції великих і малих літер, товщина штрихів, загальна структура і стиль. Під час первинного огляду знайдених варіантів було приділено особливу увагу таким ключовим деталям:

Одним із найважливіших елементів був аналіз форми літер. Серед 60 запропонованих варіантів приблизно третина шрифтів мала схожу форму символів до зразка. У деяких випадках різниця була помітною в таких елементах, як заокруглення або загостреність країв букв. Наприклад, літери "g", "a" і "e" є типовими для порівняння через їхню унікальну форму в більшості шрифтів.

Наступним кроком було порівняння товщини штрихів між зразком і знайденими варіантами. У шрифтах із високим контрастом між тонкими і товстими штрихами цей параметр був ключовим для розпізнавання схожості. Близько 50% запропонованих варіантів мали точну або дуже близьку до зразка товщину штрихів. Інші шрифти відрізнялися, оскільки вони мали або занадто тонкі, або занадто грубі лінії.

Третім важливим фактором для порівняння була відстань між символами. Кернінг у зразку був специфічним і створював враження рівномірності та зручності для читання, що є ключовим для шрифту, який використовується в текстових блоках. Кожен варіант шрифту оцінювався за тим, як рівномірно літери розташовуються одна від одної, і приблизно 40% знайдених шрифтів мали схожий або майже ідентичний кернінг до зразка.

Ще одним важливим критерієм для аналізу був розмір великих літер щодо малих, а також співвідношення висоти шрифту. Наприклад, у шрифтах типу "сериф" зазвичай є чіткі правила пропорцій, де великі літери повинні виділятися, але не домінувати над малими літерами. Близько 60% знайдених шрифтів на WhatFontIs відповідали за пропорціями зразку.

Останнім фактором був аналіз інтерліньяжу тобто відстані між рядками, а також загальна гармонія шрифту. Це важливо для тексту, що використовується у великих блоках, таких як веб сторінки або друковані видання. Тут приблизно 25% шрифтів забезпечували адекватну читабельність та схожість із зразком.

1.5 Порівняння результатів між шістьма додатками

Під час використання шести різних додатків для аналізу, таких як MyFonts, FontSquirrel, Fonts.Adobe, FontSpring, WhatFontIs та інші, було отримано загалом близько 989 можливих варіантів шрифтів. Кожен додаток мав свої особливості у пошуку шрифтів і пропонував різні варіанти на основі своїх алгоритмів і баз даних.

Однак, серед усіх отриманих результатів найкраім показав себе WhatFontIs який використовував штучний інтелект.

Оцінка схожості використовувалась за формулою співвідношення де обчислювався кожен результат окремого шрифту окремо. Також додатково було визначено кількість шрифтів які стовідсотково відповідають зразку.

При оцінки таких шрифтів як Helvetica, Times New Roman, Arial, Comic Sans було проаналізовано що:

MyFonts запропонував в середньому від 57 до 77 варіантів релевантних шрифтів з яких Helvetica показав найкращий результат 63.07% тобто він запропонував 65 варіантів з яких 41 був практично ідентичним на око. І також запропонував з цих варіантів 2 які відповідають назві. Але найгірше він оцінив Comic Sans де із 77 варантів доден не відповідав високій точності оскільки це рукописний шрифт, де для оцінки потрібно дуже багато ресурсів, в якому кожен елемент мав дуже унікальні характеристики.

FontSquirrel показав себе краще у визначенні шрифту Times New Roman оскільки він знайшов 13 варіантів і із них 53.84% були схожими.

Але кращі результати у WhatFontIs де він запропонував в кожному по 60 варіантів і з них мінімум 18% були схожими і на додаток в усіх шрифтах він зміг знайти за назвою. Детально по кожному з них буде зображено на рисуку 1.7 в якому будуть зображені всі результати досліджень.

Назва шрифтів	Всього запропоновано подібних шрифтів	Процент ефективності з експериментальних даних запропонованих подібних шрифтів	Кількість співпадінь за назвою	Існуючі найпопулярніші Веб-додатки
Helvetica	65	63.07%	2	www.myfonts.com
Times New Roman	57	57.89%	1	
Arial	61	31.14%	4	
Comic Sans	77	0%	0	
Helvetica	22	31.81%	0	www.fontsquirrel.com
Times New Roman	13	53.84%	0	
Arial	19	31.57%	0	
Comic Sans	43	4.65%	0	
Helvetica	20	5%	0	fonts.adobe.com
Times New Roman	20	20%	0	
Arial	20	0%	0	
Comic Sans	20	0%	0	
Helvetica	16	12.5%	2	www.fontspring.com
Times New Roman	5	20%	0	
Arial	14	14.28%	0	
Comic Sans	37	0%	0	
Helvetica	60	73.33%	10	www.whatfontis.com
Times New Roman	60	38.33%	5	
Arial	60	55%	11	
Comic Sans	60	18.33%	5	

Рис.1.7 Зображення ефективності підбору шрифтів

Після проведеного аналізу і порівняння результатів між шістьма додатками було можливим звузити вибір до кількох варіантів шрифтів, що найкраще відповідали оригінальному зразку. Хоча кожен з додатків пропонував цікаві та релевантні варіанти, найкращі результати були отримані завдяки WhatFontIs. Серед знайдених варіантів близько 60% шрифтів мали майже ідеальну відповідність зразку. Важливим було те, що алгоритм цього додатку дозволяє не лише знайти візуально схожі варіанти, але й оцінити їх за технічними характеристиками, такими як розмір файлу, підтримка веб-форматів та наявність додаткових символів.

У підсумку, аналіз показав, що для ефективної оцінки та підбору шрифту важливо використовувати кілька додатків, кожен з яких може дати свої унікальні результати. WhatFontIs, зокрема, показав себе як ефективний інструмент для пошуку шрифтів з високою точністю відповідності.

2 ОПИС УДОСКОНАЛЕННЯ МЕТОДИКИ СИНТЕЗУ ШРИФТІВ ЗА ЗРАЗКОМ

2.1 Визначення основних проблем на основі аналізу

Проаналізувавши найпопулярніші додатки можна сказати що вони в принципі мають не досконалий підбір правильних значень. У цьому контексті було проведено детальний аналіз результатів роботи шести додатків для пошуку шрифтів. Було проаналізовано кілька ключових аспектів, таких як точність відповідності шрифтів, швидкість обробки запитів та зручність використання кожного з додатків. На основі цих факторів було виявлено низку основних проблем, що вплинули на кінцевий результат та ефективність підбору шрифтів.

Однією з найважливіших проблем, яка була виявлена під час аналізу, є неточна відповідність шрифтів оригінальному зразку. Хоча кожен з додатків надавав список шрифтів, що, за їхньою оцінкою, могли відповідати еталону, жоден із запропонованих варіантів не був ідеальною копією зразка. Навіть ті шрифти, що були визначені як найкращі, мали відхилення у вигляді різниці в товщині штрихів, формі окремих символів або міжсимвольних інтервалах.

Ця проблема має кілька причин: алгоритми, що використовуються у додатках для пошуку шрифтів, базуються на певних наборах даних та правилах, які можуть не враховувати всі дрібні деталі оригінального зразка. Наприклад, у випадку з WhatFontIs деякі шрифти мали дуже схожу загальну структуру, але відрізнялися за тонкістю штрихів або іншими дрібними характеристиками. Алгоритм не завжди може правильно розпізнати такі нюанси, що призводить до неточних результатів. Також кожен додаток має свою базу даних шрифтів, і, хоча вона може бути дуже великою, не всі шрифти доступні в таких базах. Відсутність рідкісних або специфічних шрифтів призводить до того, що додатки часто пропонують найбільш

схожі варіанти з наявних шрифтів, навіть якщо вони не є точними відповідниками оригіналу. Це обмежує можливості користувача знайти ідеальний варіант.

Ще однією важливою проблемою, виявленою на основі аналізу, є нестабільна точність результатів між різними додатками. У той час як один додаток може запропонувати шрифти з високим рівнем відповідності, інший може дати зовсім інші результати, навіть якщо використовувався той самий зразок. Наприклад, WhatFontIs надав варіанти шрифтів із приблизно 46% точністю відповідності, тоді як MyFonts мав лише близько 38% відповідності по всім варіантам. Це свідчить про те, що різні алгоритми працюють по-різному, і користувач може отримати значно відмінні результати залежно від обраного інструменту.

Причини такої проблеми можуть включати різні алгоритми обробки зображень коли додатки для пошуку шрифтів використовують різні методи обробки зображень та розпізнавання символів. Це призводить до того, що деякі алгоритми можуть бути більш точними у розпізнаванні певних характеристик шрифту, таких як форма літер або міжрядковий інтервал, тоді як інші можуть мати труднощі з тими самими завданнями. А також різні бази даних як було зазначено раніше, кожен додаток використовує свою унікальну базу даних шрифтів, і обсяг та якість цієї бази можуть значно відрізнятися. Наприклад, один додаток може мати велику кількість сучасних шрифтів, але не мати старих або спеціалізованих шрифтів, що обмежує точність пошуку.

Під час аналізу також було виявлено проблему з обмеженою підтримкою спеціальних символів та стилів. Багато шрифтів мають унікальні символи, декоративні елементи або специфічні стилістичні деталі, які можуть не бути підтримані або не відображатися належним чином у базах даних додатків для пошуку шрифтів. Це особливо помітно у випадках, коли потрібно знайти шрифти зі специфічними гліфами, такими як діакритичні знаки, спеціальні літери або символи для певних мов. У багатьох випадках бази даних додатків містять шрифти лише з основними наборами символів, без підтримки спеціальних символів або гліфів. Це ускладнює пошук шрифтів, які використовуються в багатомовних текстах або для створення специфічних декоративних елементів. Додатки для

пошуку шрифтів зазвичай орієнтуються на загальну структуру шрифту, але не враховують стилістичні елементи, такі як декоративні штрихи, які можуть бути важливими для точного відповідності. Це призводить до того, що шрифти, які виглядають схожими на перший погляд, можуть значно відрізнятися в деталях.

Деякі додатки мають проблеми зі швидкістю обробки запитів, особливо коли йдеться про великі зображення або складні шрифти. Наприклад, у деяких випадках обробка одного зразка шрифту могла займати до кількох хвилин, що значно уповільнює процес пошуку відповідного варіанту. Такі додатки використовують складні алгоритми для розпізнавання шрифтів, що потребує значної обчислювальної потужності. Це може призводити до повільної обробки зображень, особливо якщо вони мають високу роздільну здатність або містять складні елементи. У випадку з онлайн-сервісами, такими як WhatFontIs, швидкість обробки може залежати від завантаженості сервера або інтернет-з’єднання. Це може призвести до тимчасових затримок, особливо в періоди високого навантаження.

Ще однією проблемою, яка була виявлена під час аналізу, є невідповідність результатів з очікуваннями користувача. Хоча додатки для пошуку шрифтів часто обіцяють знайти "ідеальний" або "найбільш точний" варіант, на практиці результати можуть значно відрізнятися від очікувань користувача. Це може бути пов’язано як із технічними обмеженнями самих додатків, так і з обмеженими можливостями користувачів у налаштуванні пошукових параметрів.

Багато додатків пропонують лише базові параметри для налаштування пошуку шрифтів, що може не відповідати потребам користувачів.

2.2 Принцип роботи OCR-системи

Принцип дії систем оптичного розпізнавання символів (OCR) полягає у перетворенні зображень, що містять текст, у читабельный формат, який можна редагувати, поширювати або аналізувати. Цей процес є складним і включає кілька етапів, кожен з яких спрямований на розпізнавання текстових символів з

максимальною точністю. OCR-системи використовуються в різних сферах від автоматизації документообігу до цифровізації бібліотек та архівів. У процесі OCR бере участь комбінація методів комп’ютерного зору, штучного інтелекту та обробки зображень, що дозволяє витягувати текст із друкованих, рукописних або сканованих документів.

Першим етапом роботи OCR-системи є підготовка зображення для подальшого аналізу. Це критично важливий крок, оскільки якість зображення безпосередньо впливає на точність розпізнавання символів. На цьому етапі виконується декілька операцій:

Сканування документа коли система отримує вхідне зображення через сканер або цифрову камеру. Цей етап залежить від якості вихідного документа та налаштувань сканування, таких як роздільна здатність (DPI) та глибина кольору.

Конвертація у градації сірого щоб зменшити складність зображення, його часто перетворюють з кольорового формату в градації сірого або бінарне зображення (чорне-біле). Це дозволяє спростити подальшу обробку та аналіз тексту. У градаціях сірого легко відокремити текстові символи від фону.

Нормалізація зображення: Зображення може бути спотворене через нахил, шум або нечіткість. OCR-системи використовують алгоритми для вирівнювання зображення та видалення небажаних артефактів. Наприклад, може бути застосований процес дескеювання, щоб виправити нахилені або нерівні документи. Окрім цього, фільтри видаляють "шум" тобто зайві пікселі, що не містять корисної інформації.

Після підготовки зображення система переходить до етапу сегментації, який передбачає розділення зображення на окремі компоненти для подальшого аналізу. Це означає, що система повинна визначити області, що містять текст, і відокремити їх від інших елементів зображення, таких як зображення, графіки, або таблиці.

Вона починає з визначення текстових блоків на зображенні. Це можуть бути окремі абзаци або блоки тексту на сторінці. Система аналізує структуру зображення, шукаючи області, де присутні регулярні символи чи літери. Після визначення текстових блоків система розбиває їх на рядки та слова.

Для цього можуть використовуватися різні методи, зокрема аналіз проміжків між символами та рядками. Проміжки між рядками зазвичай більші, ніж між словами, що дозволяє алгоритмам точно визначити, де закінчується один рядок і починається інший. Наступний крок полягає у розбитті кожного слова на окремі символи. Це складний процес, оскільки деякі шрифти можуть мати дуже близькі за виглядом символи, а рукописні тексти можуть бути розмитими або об’єднаними.

Після сегментації система приступає до основного етапу — розпізнавання символів. На цьому етапі OCR використовує алгоритми для зіставлення виділених символів з базою даних шрифтів або шаблонів. Один із традиційних методів розпізнавання символів полягає у порівнянні виділених фрагментів із заздалегідь підготовленими шаблонами символів. Це працює добре для друкованих текстів із чіткими та стандартними шрифтами. Система порівнює кожен символ зі своїм набором шаблонів і визначає, який із них найбільш схожий.

Також сучасні OCR-системи все частіше використовують алгоритми машинного навчання та нейронні мережі.

Ці системи проходять навчання на великих наборах даних, що дозволяє їм розпізнавати символи навіть у складних умовах, таких як нерівний текст або нестандартні шрифти. Нейронні мережі можуть аналізувати контекст і розпізнавати символи, які важко відрізнити за традиційними методами. системи можуть використовувати контекстуальний аналіз для підвищення точності розпізнавання. Наприклад, після початкового розпізнавання тексту система може враховувати граматичні та лексичні правила мови для коригування результатів. Це допомагає уникати помилок, коли деякі символи виглядають схоже, але у контексті слова або речення мають різне значення.

Після завершення процесу розпізнавання тексту, OCR-система виконує етап постобробки для корекції можливих помилок та підвищення точності результатів. Сучасні OCR-системи часто включають модулі для автоматичної перевірки орфографії та граматики. Після первинного розпізнавання тексту система може аналізувати його на предмет поширених помилок, таких як неправильне написання слів або помилки у структурі речення.

У багатьох випадках OCR-системи повинні не тільки розпізнавати текст, але й зберігати структуру документа. Це особливо важливо для таблиць, списків, заголовків та інших елементів форматування. Система намагається відтворити ці елементи у кінцевому результаті, щоб він був максимально наближений до оригіналу. Для перевірки точності розпізнаного тексту OCR-системи можуть використовувати додаткові алгоритми. Це може включати повторний аналіз символів, що мають сумнівне розпізнавання, або зіставлення результатів із іншими джерелами даних. Остаточним етапом роботи OCR-системи є експорт розпізнаного тексту у зручний формат. Залежно від вимог користувача або системи, текст може бути збережений у форматах

Хоча сучасні OCR-системи можуть досягати високого рівня точності, вони все ще стикаються з певними проблемами та обмеженнями, особливо при роботі з нестандартними або складними документами.

Багато OCR-систем мають труднощі з розпізнаванням текстів, написаних нестандартними шрифтами або рукописом. Це особливо актуально для історичних документів або рукописних записів. Нечіткі, з низькою роздільною здатністю або пошкоджені документи можуть ускладнити роботу OCR. Навіть найкращі системи можуть давати неправильні результати при роботі з низькоякісними зображеннями. Хоча багато сучасних OCR-систем підтримують розпізнавання різних мов, мови з не латинськими символами (наприклад, китайська, японська, арабська) можуть створювати додаткові виклики для коректного розпізнавання тексту.

2.3 Вирішення проблеми сегментації OCR-системи

Оскільки є проблема в сегментації де елементи можуть бути злитими які зображенні на рисунку 2.1 зправа пропонується повторно їх сегментувати для подальшого аналізу символів.

Рис.2.1 Зображення помилок сегментації шрифтів

Таке вдосконалення є складним і багатоступеневим процесом, спрямованим на усунення помилок, які виникають під час початкової сегментації. Коли OCR-система неправильно розділяє символи, об’єднуючи два або більше символів в один, або, навпаки, розбиваючи один символ на кілька частин, необхідно застосувати підхід до уточнення сегментації. Це включає виявлення проблемних областей, аналіз характеристик злитих або розділених символів, застосування методів поділу та перевірки результатів.

Перший етап повторної сегментації полягає у виявленні помилкових областей. Для цього OCR-система аналізує розмір, форму та проміжки між символами в текстовому блоці. Наприклад, якщо один із символів значно більший за інші, це може свідчити про об’єднання кількох символів в один. Аналогічно, відсутність частини символу або появу окремих фрагментів, що не складають єдину форму, можна розглядати як помилку розділення. Аномальні проміжки між елементами також можуть сигналізувати про необхідність повторної сегментації. Ці дані збираються і використовуються для фокусування системи на конкретній проблемній області, що потребує додаткового аналізу.

На другому етапі, після локалізації проблемної області, запускається процедура повторної сегментації. Спочатку зображення цієї області піддається морфологічному аналізу. Використовуються операції ерозії та розширення, які дозволяють уточнити контури символів. Ерозія допомагає зменшити кількість

зайвих пікселів і виявити межі між злитими символами, тоді як розширення відновлює форму символів, щоб запобігти втраті важливих деталей. Після цього застосовується контурний аналіз для визначення чітких меж між символами. Алгоритми, такі як метод Канне, дозволяють виділити контури злитих символів і поділити їх на окремі частини.

Додатково, у складних випадках, коли межі між символами нечіткі або складні, застосовуються алгоритми активних контурів. Цей метод дозволяє адаптувати контур до форми символів, знаходячи критичні точки, які визначають межі між ними. Також можливе використання алгоритмів кластеризації, таких як k-means, які аналізують щільність пікселів у проблемній області. Якщо пікселі формують кілька щільних кластерів, це може бути індикатором злитих символів, які необхідно розділити.

У контексті OCR-систем вони використовуються для аналізу пікселів або фрагментів зображення, щоб виявити структуру символів, злитих або розділених у процесі початкової сегментації. Кластеризація дозволяє визначити, які частини зображення належать до одного символу, а які варто розділити або об’єднати. Цей підхід базується на пошуку схожостей між даними, такими як розташування, інтенсивність або форма пікселів.

Іншим методом є DBSCAN який кластеризує дані на основі щільності. Цей алгоритм виявляє області з високою щільністю пікселів, які утворюють кластери, і відокремлює області низької щільності як "шум". У випадку OCR DBSCAN особливо корисний для роботи з нерівномірними символами або складними рукописними текстами, де традиційні методи можуть не працювати через неоднорідність розподілу пікселів. DBSCAN дозволяє ефективно знаходити злиті символи, навіть якщо їх межі нечіткі або перекриваються.

Ще одним популярним підходом є алгоритм ієрархічної кластеризації, який створює дерево кластерів, об’єднуючи найближчі пари даних або розділяючи їх на основі схожості. У контексті OCR цей метод може використовуватися для обробки складних зображень, де символи мають кілька рівнів деталей. Наприклад, він може

спочатку об’єднати всі пікселі в групи, що відповідають загальній структурі символів, а потім уточнити ці групи, розділяючи їх на основі дрібніших деталей.

Важливо, що вибір алгоритму кластеризації залежить від конкретної проблеми. Для чітких друкованих текстів алгоритм k-means може бути більш ефективним, тоді як для рукописних текстів або складних зображень DBSCAN та ієрархічна кластеризація часто демонструють кращі результати. Після кластеризації OCR-система проводить аналіз отриманих кластерів, зіставляючи їх із шаблонами символів або оцінюючи контекст тексту для підтвердження правильності сегментації.

Після поділу символів система перевіряє результати, використовуючи шаблони символів із бази даних або застосовуючи контекстуальний аналіз. Це дозволяє переконатися, що розділені символи відповідають вимогам розпізнавання та правильно інтегруються у загальний текст.

На заключному етапі повторна сегментація завершується перевіркою результатів у контексті тексту. Якщо розділені символи формують логічні слова або фрази, це підтверджує успішність сегментації. У разі виявлення залишкових помилок система може ініціювати додаткову ітерацію сегментації або коригування. Завдяки такому підходу OCR-система здатна забезпечити високу точність розпізнавання тексту навіть у випадках, коли початкова сегментація виявилася недостатньо точною.

2.4 Аналіз постобробки так виклик повторної сегментації

Після детального опису повторної сегментації пропонується змінити порядок дій в OCR-системі. Для цього необхідно змінити послідовність і додати додаткову постобробку для розділення символів. До буде доступно ввести символи які можуть бути на зображенні а також вибрати ті які потребують повторного розділення як зображено на рисунку 2.2.

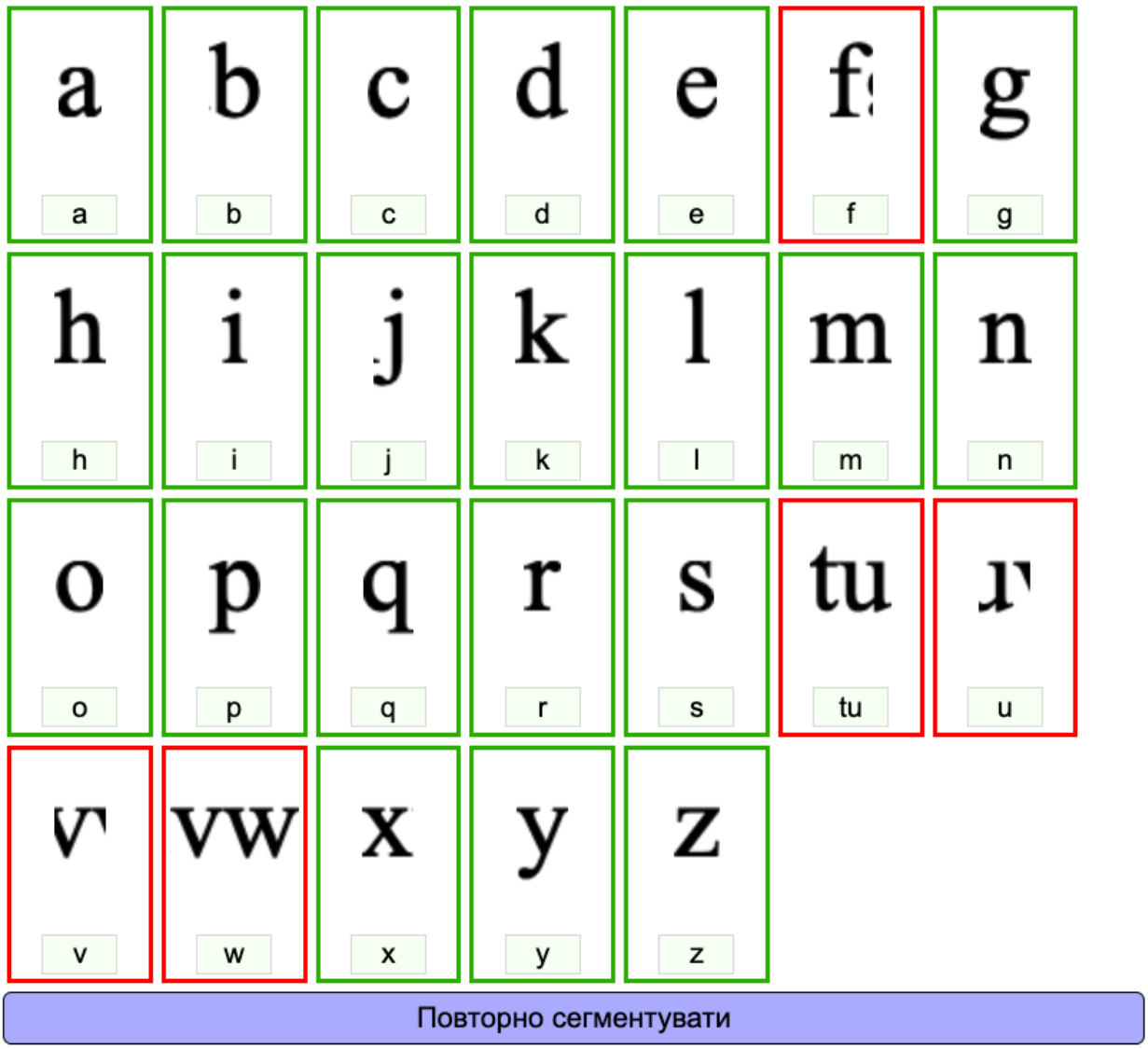


Рис.2.2 Зображення постобробки для повторного розділення

Таким чином це забезпечить правильну обробку і дасть можливість використовувати елементи в майбутньому. А такі елементи які підібрані правильно можуть використовуватися вже для редагування самого документу.

2.5 Вирішення проблеми підбору за допомогою класифікації баз даних

Класифікація бази даних шрифтів за окремими характеристиками є ключовим процесом для забезпечення точного і швидкого підбору відповідних

варіантів. Ця методика спрямована на впорядкування даних таким чином, щоб шрифти з подібними ознаками були згруповані разом, створюючи логічну структуру для пошуку. Основою такого підходу є визначення типографічних характеристик, які найбільше впливають на вибір шрифту.

До таких характеристик належать сімейство шрифту наприклад:

- серифний
- санс-серифний
- рукописний

вага шрифту, стиль такий як:

- курсив,
- прямий
- капітель

рівень контрастності між товстими і тонкими штрихами, а також призначення шрифту для друку, веб застосунків, заголовків або декоративного використання.

Окрім цього, важливу роль відіграють технічні параметри, такі як формат файлу TTF, OTF, WOFF і підтримка специфічних символів чи мов.

На початковому етапі шрифти розподіляються в загальні категорії за сімействами. Наприклад, всі серифні шрифти поміщаються в одну групу, санс-серифні – в іншу, а декоративні чи рукописні формують окремі категорії. Далі в межах кожного сімейства здійснюється подальший поділ на підгрупи, враховуючи такі параметри, як вага, стиль або контрастність. Це створює багаторівневу ієрархію, яка дозволяє швидко звузити пошук до конкретних типів шрифтів. Для ефективного управління цією структурою використовується багатовимірний векторний підхід, де кожен шрифт представляється у вигляді вектора характеристик. Наприклад, сімейство може бути закодоване числовим значенням, вага є представленою шкалою від легкого до жирного, а контрастність - числом від 0 до 1, що відповідає рівню різниці між штрихами.

Після класифікації всі шрифти отримують анотації, які включають опис їхніх характеристик. Наприклад, для шрифту може бути зазначено, що це sans-serif

шрифт із вагою 400, низькою контрастністю і призначенням для веб застосунків. Такі метадані дозволяють ефективно фільтрувати шрифти під час пошуку і легко ідентифікувати їх. Коли користувач вводить запит, система звертається до відповідного сегмента бази даних і виконує пошук лише серед шрифтів, які відповідають заданим характеристикам. Наприклад, якщо необхідно знайти sans-serif шрифт із високою контрастністю для заголовків, система обирає лише ті групи, які відповідають цим параметрам, що значно скорочує час обробки запиту.

Такий підхід має кілька важливих переваг. Він дозволяє не тільки швидко знаходити відповідні шрифти, а й забезпечує точність результатів завдяки зосередженню пошуку на релевантних групах. Окрім того, база даних залишається масштабованою: нові шрифти легко додаються до відповідних категорій без необхідності перегляду всієї структури. Це створює динамічну і гнучку систему, яка відповідає сучасним вимогам до пошуку та вибору шрифтів у цифровому середовищі.

2.6 Використання сегментованих елементів в документі

Підхід використання сегментованих елементів як окремих зображень полягає у створенні набору зображень, кожне з яких відповідає окремому символу, гліфу або елементу тексту. Цей підхід забезпечує високу гнучкість у подальшому використанні текстових елементів, зокрема шляхом прив’язки кожного зображення до певної клавіші або комбінації клавіш для швидкого вставлення в документ.

Основна ідея методу полягає в тому, що під час обробки тексту OCR-система або інший механізм розпізнавання розбиває текст на сегменти, які відповідають окремим символам. Кожен символ зберігається як окремий графічний файл у зручному форматі, наприклад PNG чи SVG. Ці зображення стають готовими для використання як незалежні елементи, які можна вставляти в текстові або графічні документи.

Для реалізації такого підходу першим етапом є сегментація тексту на окремі символи або елементи. Система виділяє контури кожного символу та ізолює його від інших частин тексту. Після цього кожен сегмент зберігається у вигляді окремого зображення. Доцільно забезпечити стандартизацію розмірів і пропорцій зображень, щоб усі елементи були уніфіковані, що спростить їх використання. Наприклад, усі символи можуть бути поміщені в рамку однакового розміру, навіть якщо деякі з них менші за інші, забезпечуючи узгодженість при вставці в документ.

Ключовою частиною цього підходу є система прив’язки кожного зображення до певної клавіші або комбінації клавіш. Для цього створюється таблиця відповідності, яка зіставляє кожен символ із його кодом клавіатури або Unicode. Наприклад, зображення символу "А" може бути прив’язане до клавіші "А", а спеціальний символ, такий як "Є", – до комбінації клавіш "Alt+E". Ця таблиця дозволяє автоматизувати процес вставлення зображень у документ, оскільки натискання відповідної клавіші або комбінації клавіш автоматично викликає потрібне зображення.

Для забезпечення точності та зручності використання сегментованих зображень у документі важливо враховувати прив’язку до сітки або координатної системи.

Наприклад, під час вставлення символи повинні автоматично вирівнюватися по базовій лінії тексту, щоб створити гармонійний і професійний вигляд. Для цього кожне зображення може містити метадані, які визначають його точку прив’язки до базової лінії, ширину та інтервал між символами.

Цей підхід відкриває широкі можливості для нестандартного використання текстових елементів. Наприклад, його можна застосувати для створення кастомних шрифтів, де кожен символ є зображенням, або для роботи зі спеціальними символами, які не підтримуються стандартними шрифтами.

Це також дозволяє легко інтегрувати графічні елементи в текстовий контент, зберігаючи точність розташування та масштабування.

Крім того, такий метод зручний для мов, де один символ може мати складну графічну структуру, або для історичних текстів, де стандартні шрифти не здатні точно відобразити оригінальні символи. Використання сегментованих зображень як окремих елементів дозволяє зберегти оригінальну естетику тексту, роблячи його доступним для цифрового використання та редагування.

Загалом, цей підхід забезпечує високу гнучкість, універсальність і точність, що робить його корисним для різних завдань у галузі роботи з текстами та графікою.

3 ПРОГРАМНА РОЗРОБКА ВИКОРИСТАННЯ СЕГМЕНТОВАНИХ ЕЛЕМЕНТІВ

3.1 Основні етапи реалізації

Головними етапами буде створення веб додатку, який використовує завантажене зображення для аналізу і сегментації елементів, які будуть використовуватися для редагування цих зображень.

Для цього буде використано node.js для бекенду та html, css, javascript для фронтенду. Для аналізу зображень і фрагментації будуть використанні такі модулі як:

- expres
- multer
- tesseract.js

Tesseract.js - забезпечує можливість аналізу тексту та сегментації. Це JavaScript-бібліотека для оптичного розпізнавання тексту (OCR) з використанням ядра Tesseract, портованого на WebAssembly. Вона дозволяє аналізувати зображення та витягувати текст. Для вирішення задачі сегментації зображення з отриманням окремих елементів тексту, таких як букви, у вигляді окремих canavas-об’єктів.

Основними етапами буде:

Створення UI-UX дизайну

Розробка функціоналу для завантаження зображення да відображення в canvas

Розробка функціоналу для отримання сегментів тексту

Панель відображення елементів у вигляді списку та можливої повторної сегментації

Використання сегментів в редагуванні даного зображення

Створення допоміжного вибору мови для аналізу елементів тексту.

Створення можливості завантаження відредагованого зображення

Таким чином реалізовано, основну панель для завантаження, сегментації та редагування зображень.

3.2 Розробка UI частни

За допомогою мов html і css розробленно основну панель керування, яка вміщує в себе:

кнопки завантаження зображення;

відображення зображення в основній частині;

відображення панелі елементів сегментації;

відображення вибору мови для кращого аналізу.

Таким чином буде відображено всі ключові елементи в межах одного екрану який можна побачити на рисунку 3.1

Рис. 3.1 Зображення основної панелі керування

Програмний код буде включати себе частково сам html а частина регенерованих елементів буде в собі мати вбудовану структуру де додатково імпортується tesseract

```
<script src="https://cdn.jsdelivr.net/npm/tesseract.js"></script>
<div style="display: flex; flex-direction: column; gap: 8px;padding-bottom: 12px;">
    <h1>Завантажте зображення для розпізнавання тексту</h1>
    <select style="max-width: 200px;" name="" id="lang">
      <option value="eng">English</option>
      <option value="ukr">Українська</option>
    </select>
</div>
  <form id="uploadForm">
    <input type="file" id="fileInput" accept="image/*" required>
    <button type="submit">Завантажити</button>
  </form>
  <h2>Ваше зображення:</h2>
  <pre id="output"></pre>
  <div class="canvas-wrapp">
    <canvas id="canvas"></canvas>
  </div>
  <button id="downloadBtn">Завантажити</button>
  <div id="canvases-container"></div>
  <script type="module" src="/script.js"></script>
```

За допомогою script.js буде підключено функціональну частину та за допомогою css розроблено основну стилістику для зручного користування програмою.

```
body {
      background-color: #b0afeb;
}
```

Для body додан колір основного фону сторінки

```
row  {
        display: flex;
      }
 .cell {
        display: flex;
        flex-direction: column;
}
```

Створено сутності row, cell які допомагають зручно розміщувати елементи наче конструктор

```
      .canvas-wrapp {
        margin-top: 12px;
        border: 2px solid #d1d1d1;
        width: fit-content;
        height: fit-content;
      }
```

Для правої панелі додано стилі які будуть розміщувати вікно постобробки, де будуть зображени сегментовані літери

```
      .letter-canvas {
         margin: 5px;
         display: inline-block;
      }
      #canvases-container {
         position: fixed;
         right: 0;
         top: 3px;
         height: calc(100% - 100px);
         overflow: auto;
         padding: 12px;
         border: 2px solid;
         max-width: 461px;
         display: flex;
         flex-wrap: wrap;
         justify-content: flex-start;
         background-color: #fff;
      }
      #canvases-container div {
         display: inline-flex;
         justify-content: center;
         flex-direction: column;
         align-items: center;
         border: 1px solid black;
         margin: 2px;
         flex: 1 1 13%;
```

```
    justify-content: flex-start;
    max-width: 13%;
    background-color: #fff;
}
#canvases-container input {
    width: 30px;
    text-align: center;
    background-color: #f6fff4;
    border: 1px solid #ddd;
}
```

3.3 Розробка функціоналу використання сегментації

Функціонал, який опрацьовує зображення з текстом і перетворює його на набір рядків і символів, полягає в послідовному застосуванні кількох кроків. Спочатку йде процес розпізнавання зображення за допомогою Tesseract.js. Цей інструмент аналізує пікселі на зображенні та виводить набір символів разом із координатами для кожного з них. Координати зберігаються у вигляді чотирьох величин: x0, y0, x1, y1. Перші дві означають верхній лівий кут кожного символа (x0 – це координата по горизонталі, а y0 – по вертикалі), а наступні дві – нижній правий кут (x1, y1). Іншими словами, Tesseract.js визначає прямокутну зону на зображенні, в якій розташовано певний символ, і повідомляє, де саме ця зона починається та де закінчується. Завдяки цьому можна не лише отримати сам текст у вигляді рядка, а й точно визначити, де кожна літера знаходиться на початковому зображенні.

Після розпізнавання тексту та отримання координат символів постає завдання належним чином відтворити те, що було на зображенні, використовуючи веб-технології, зокрема <canvas> у браузері. Щоб зберегти структуру тексту, спершу потрібно розподілити всі символи по рядках, адже око людини звикло, що текст не суцільний, а розбитий на лінії. Логічно припустити, що символи, які мають близькі значення координати y0, належать одному рядку, тоді як символи, які різко «стрибнули» вище чи нижче (відмінність у декілька десятків пікселів), повинні формувати наступний рядок. Тож у коді зазвичай зберігається попереднє значення

координати y0, яке береться з попереднього символа, і якщо різниця між поточним символом і попереднім перевищує певний поріг, вважається, що ми перейшли на інший рядок. У результаті такої ітерації формується структура даних, де кожен елемент – це «рядок», який містить масив символів, що дійсно розташовані приблизно на одній горизонтальній лінії в оригінальному зображенні.

Далі виникає питання відтворення реального вигляду кожного рядка у браузері. Для цього використовується <canvas>: під кожен символ створюється окремий канвас, на який «вмонтовується» відповідний фрагмент вихідного зображення. Якщо цього не робити, і малювати все на одному канвасі, керувати позиціями кожного символа може бути складніше, особливо коли потрібно робити подальше опрацювання або розмітку. Натомість, коли ісгнує окремий канвас на кожен символ, з ним легко експериментувати: його можна пересунути, масштабувати чи застосувати додаткові ефекти. Проте, щоб зберегти єдиний стиль відтворення тексту і зробити його візуально приємним, потрібно дотримуватися однакової висоти всіх канвасів у межах одного рядка.

Одним із ключових моментів є визначення висоти рядка. Зазвичай у справжніх шрифтах деякі літери мають верхні елементи (наприклад, «f» або «б» у кирилиці), які піднімаються вище за основний рівень, а інші літери мають елементи, що опускаються нижче базової лінії (скажімо, «g», «р» чи «у» в латиниці). Якщо взяти кілька літер із різними верхніми та нижніми хвостиками, стає очевидно, що висота рядка має покривати усі крайні точки – від найвищої до найнижчої.

Щоб з’ясувати ці точки, код аналізує всі символи, які належать конкретному рядку, і шукає мінімальне та максимальне значення координат y: мінімальна координата y0 серед символів визначить найвищу точку (з погляду візуалізації це фактично верх рядка), а максимальна координата y1 покаже, де закінчуються найглибші елементи літер. Різниця між цими двома значеннями дає загальну висоту рядка, до якої зазвичай додають кілька пікселів зверху та знизу, щоби літери не «прилипали» до краю канваса.

Коли визначено висоту рядка, можна обчислити, як саме зміщувати кожен символ всередині власного канваса. Адже, з одного боку, усі канваси в рядку повинні мати однакову висоту, а з іншого – кожна літера в оригіналі могла бути вище або нижче відносно базової лінії. Щоб це реалізувати, обчислюється відносне зміщення літери: якщо символ починається на координаті y0, а рядок починається з minY, то відстань (y0 - minY) задасть, наскільки зверху треба відступити. Якщо уявити, що мінімальне значення y0 серед усіх літер у рядку є умовною «вершиною» рядка, то літери, які в оригіналі починалися точно на цьому рівні, взагалі не отримають додаткового відступу. Навпаки, літера, що розташована нижче, відсунеться всередині канваса на певну кількість пікселів униз. Завдяки цьому літери з верхніми елементами не «обрізатимуться», а ті, що мають нижні елементи, теж будуть там, де потрібно, хоч і всі канваси мають однакову висоту.

Для того, щоб зобразити ту чи іншу літеру на відповідному канвасі, використовується метод *drowImage()*. Перші чотири аргументи дозволяють вирізати з оригінального зображення саме ту частину, де розміщена ця літера (x0, y0, letterWidth, letterHeight), а наступні чотири вказують, у якій точці канваса та з якими розмірами потрібно її намалювати (починаючи, наприклад, з нуля по осі X і topOffset по осі Y, і зберігаючи оригінальну ширину та висоту). У цьому разі topOffset – це наш вертикальний відступ, що компенсує різницю між найвищою точкою рядка і конкретним символом. Така схема дозволяє зберігати унікальне розташування літер, роблячи їхнє відображення максимально близьким до того, що було на оригінальному зображенні.

Цей підхід водночас дає змогу втілити ефект рівномірного «line-height». Кожен канвас у рядку матиме однакову висоту, яка відповідає найвищій/найнижчій точці серед усіх літер. Якщо якась літера не потребує стільки простору, вона малюється ближче до базової лінії; літери з підйомами чи спусками, відповідно, виходять за межі базової лінії, але не «вилазять» за канвас, оскільки його висота спеціально збільшена настільки, щоб вмістити всі можливі варіації. Усе це дуже схоже на базові принципи відображення шрифтів у реальних текстових системах,

де ми маємо поняття надрядкових і підрядкових виносів, що забезпечують приємний для читання, естетично вирівняний текст.

Таким чином, кожен символ у результаті опиняється в окремому блоці, але ці блоки узгоджені з точки зору вертикальних розмірів і позицій. Якщо уявити собі, що потрібно, наприклад, «захопити» одну конкретну літеру для подальшого дослідження чи інтерактивної взаємодії, то такий метод є дуже зручним. Адже для кожної літери не потрібно рахувати взаємні зміщення в межах одного канваса: вона вже має свій маленький канвас, де все чітко обмежено прямокутником, знайденим Tesseract.js. Крім того, якщо буде необхідно зробити якусь анімацію, зміну кольору чи будь-яку іншу обробку для літери, це значно спроститься, оскільки досить унести зміни всередині єдиного <canvas> елемента.

Також важливо зазначити, що якщо певний символ мав короткі вертикальні відстані (наприклад, «а» чи «е», які не дуже виступають догори чи донизу), він малюється, не використовуючи додатковий простір канваса, тобто в нього менший відступ зверху чи знизу.

Своєю чергою, «f» чи «у» можуть отримати значний відступ, бо мають більш виражені підйомні чи спускні елементи. Усе це відбувається автоматично завдяки тому, що при обчисленні висоти рядка беруться крайні координати, а при малюванні – коригуємо позицію кожної літери відносно minY.

В остаточному підсумку робота цього функціонала полягає у відтворенні «макету» тексту з оригінального зображення: спершу зображення зчитується Tesseract.js, далі символи поділяються на рядки, по кожному рядку визначаються глобальні вертикальні межі, і, нарешті, кожна літера потрапляє в окремий канвас, де вона намальована точнісінько так, як була на зображенні, але з урахуванням однакової загальної висоти для всіх літер цього рядка. При цьому зберігаються характеристики типу верхніх і нижніх виносів, а саме відображення має вигляд, максимально наближений до оригіналу.

Це дозволяє маніпулювати, скажімо, кольором кожної літери, а також розташуванням цих канвасів на сторінці, без ризику спотворити взаємне вирівнювання. Тож, коли виникає потреба «витягнути» текст із зображення та

перенести його у веб-середовище, а заодно зберегти всі відмінності високих і низьких частин символів, вищеописаний підхід стає зручним і доволі елегантним рішенням.

Для реалізації зміни кольору щоб елемент можна всюди використати створено для кожного елементу і він працює по такому принципу, що він відтворює тільки ту частину що має кольора тексту інші координати ігнорує

```
for (let i = 0; i < canvas.height; i++) {
for (let j = 0; j < canvas.width; j++) {
const [ r, g, b, a ] = ctx.getImageData( j, i, 1, 1 ).data;
            ctx2.fillStyle = `rgba(${r} ,${g}, ${b}, ${a})`
            if((a < 5) || (r > 240 && g > 240 && b > 240) ) {
              ctx2.fillStyle = `rgba(${255} ,${255}, ${255}, ${1})`
}
               if((r > 70 && g > 70 && b > 70) ) {
                     ctx2.fillStyle = `rgba(${r} ,${g}, ${b}, ${1})`
}
ctx2.fillRect(j,i,1,1);
}
}
```

Він ігнорує точки в яких колір найближчий до білого і додає прозорість, там де кольори наближені до темного таким чином елемент зберігає лише контури і при наступному використанні в буквах немає ефекту лишньої обводки.

```
let lines = [];
let elemText = [];
let currentLine = [];
let previousY = symbols[0].bbox.y0
symbols.forEach((symbol) => {
const { bbox } = symbol;
    const { y0 } = bbox;
    if (Math.abs(y0 - previousY) > 20) {
```

```
        lines.push(currentLine);
            currentLine = [];
        }
        currentLine.push(symbol);
        previousY = y0;
});
```

Цей код обробляє масив символів, визначаючи, які з них належать до однієї лінії тексту на основі їхнього вертикального розташування. Для кожного символу отримується його координата по вертикалі, яка порівнюється з координатою попереднього символу. Якщо різниця між ними перевищує заданий поріг 20 пікселів, це означає, що символ знаходиться на новій лінії. У такому випадку поточна лінія тексту додається до загального масиву ліній, і починається формування нової лінії. Після цього символ додається до поточної лінії, а вертикальна координата оновлюється для подальших порівнянь. У результаті всі символи групуються у лінії залежно від їхнього розташування на полотні.

3.4 Ефективність синтезу шрифтів на основі існуючих елементів

Там де потрібен унікальний шрифт, або ж потрібна однорідна стилістика, часто в таких випадках постає питання, що краще шукати в базах даних схожий шрифт чи спробувати самостійно синтезувати схожий набір символів, використовуючи можливості OCR та алгоритмів генерування. На перший погляд може здаватися, що простіше знайти готовий варіант: мовляв, велика ймовірність, що хтось уже створив щось подібне, особливо тепер, коли у світі існує безліч бібліотек шрифтів. Проте на практиці виявляється, що метод синтезу або реставрації, якщо йдеться про відновлення старовинного набору літер може бути значно ефективнішим і дає низку переваг.

Насамперед, синтезовані шрифти дозволяють максимально точно зберегти автентичний стиль, яким вирізнялися оригінальні літери. Коли ми шукаємо інший, уже існуючий шрифт, навіть той, який називається схожим, ми майже завжди втрачаємо певні нюанси форми. Адже схожість це поняття суб'єктивне, і часто професійне око дизайнера чи бренд-менеджера помітить різницю в товщині штрихів, формах засічок, куті нахилу кривих. Якщо ж у нас є навіть невеликий набір літер із оригінального зображення, OCR-алгоритм з подальшою обробкою може навчитися на цих кількох прикладах, а потім шляхом машинного навчання спробувати згенерувати решту. Завдяки цьому ми отримуємо гармонійну систему символів, яка збігається за стилем із вихідним зразком набагато краще, ніж будь-який сторонній схожий шрифт.

По-друге, синтез шрифтів уможливлює більшу гнучкість: якщо ми хочемо розширити цей набір символів додати латиницю, кирилицю, ще якісь спеціальні знаки чи діакритичні символи, маючи алгоритм, тренований на певному стилі, можемо досить швидко домалювати відсутні елементи. Якщо ж ми шукаємо готовий шрифт, то навряд чи знайдемо ту саму стилістику одночасно в різних розкладках або ж у повному діапазоні символів. Навіть коли знайдено «дуже близький» варіант, найімовірніше, що в ньому бракуватиме, наприклад, якоїсь специфічної букви, потрібної для іншої мови, а власнику шрифту треба буде додатково заплатити чи звернутися до розробника, щоб отримати потрібний знак.

Ще один важливий момент: у проєктах, де корпоративний стиль визначається історичною тяглістю чи унікальним фірмовим дизайном, шрифт може бути частиною бренд-айдентики. Якщо компанія була заснована, скажімо, сто років тому й використовувала оригінальний логотип чи оформлення документів, тоді відтворення цього стилю стає пріоритетним завданням. Пошук подібного готового шрифту ризикує спотворити суть старого дизайну.

Натомість метод синтезу дасть змогу взяти оригінальний плакат чи документ (навіть якщо це пожовклий від часу папір), пропустити його через OCR і комп'ютерний зір, а потім перетворити інформацію про форму літер на сучасний формат шрифту наприклад - OpenType.

У результаті компанія одержує власний візуальний кодекс, що не залежить від сторонніх авторських розробок, одночасно зберігаючи повну відповідність фірмовим традиціям.

Певна річ, синтезовані шрифти вимагають складнішої початкової підготовки й опрацювання. Потрібні системи OCR, машинне навчання чи ручне ретельне «шліфування» символів у векторних редакторах. Утім, порівняно з нескінченним пошуком у каталогах та постійними компромісами на кшталт «майже підходить, але ніби щось не те», підхід із генеруванням чи відтворенням потрібного набору літер часто швидший і результативніший. До того ж, шрифт, отриманий у такий спосіб, можна пізніше модифікувати: змінювати товщину, додавати альтернативні варіанти лігатур чи навіть моделювати стилі наприклад, устав, курсив, жирний.

Для деяких дизайнерів і компаній надзвичайно важливо мати повний контроль над своїм шрифтовим рішенням. Якщо ж взяти готовий шрифт, то для його використання можуть існувати ліцензійні обмеження: платні, обмежені за кількістю комп’ютерів чи форматів, або ж зовсім не доступні для комерційних проєктів.

Така залежність від стороннього правовласника іноді стає критичним фактором, особливо для великих корпорацій. А от синтезований шрифт, зроблений на основі власних розробок до яких ви маєте всі права чи історичних зразків якщо авторські обмеження вже не діють, повністю перебуває під вашим контролем. Можна вільно трансформувати його й застосовувати на будь-яких майданчиках.

Окрім того, при синтезі простіше заховати фірмові деталі всередині шрифту, створюючи унікальні особливості, невидимі на перший погляд. Така практика іноді використовується, аби попередити несанкціоноване копіювання: якщо хтось вирішить підробити бренд, у тексті з’являться дефекти, які відразу виявить оригінальний власник. Знайдені ж у відкритих каталогах шрифти не мають подібних індивідуальних налаштувань і, відповідно, захистити свій дизайн буде складніше.

Отже, синтезовані шрифти часто виявляються ефективнішими за пошук і використання існуючих, коли мова йде про точне відтворення оригінального стилю, гнучке доповнення новими символами, повний контроль над правами й можливість формувати унікальну айдентику. Звісно, створити такий шрифт не завжди просто залучаються алгоритми OCR, методи машинного навчання та вручну доопрацьовуються дрібні деталі. Проте за підсумком це дає фахівцям набагато більше свободи, ніж нескінченний пошук чогось більш-менш схожого, що може лише поверхово імітувати справжній характер літер. У результаті компанія чи дизайнер отримують шрифт, який не лише візуально нагадує оригінал, а насправді відтворює всі суттєві елементи його форми й дає змогу користуватися ним без обмежень.

ВИСНОВКИ

В ході дослідження процесу розпізнавання шрифтів за зразком та синтезу існуючих елементів було виконано та задокументовано такі завдання:

1. Проведений аналіз наявних програм для розпізнавання шрифтів продемонстрував, що автоматичні механізми визначення часто працюють неідеально. Зокрема, алгоритми можуть неправильно інтерпретувати особливості контурів літер, виявляючи шрифти, які насправді не належать до сімейства, заданого зразком. Крім того, недосконаліть у розпізнаванні призводить до неточних порівнянь зі схожими шрифтами, що лише виглядають подібними, але відрізняються в ключових елементах товщині штрихів, пропорціях, формах засічок. Як наслідок, користувач отримує результат, далекий від бажаного, і витрачає більше часу на ручний пошук та виправлення помилок у встановленні потрібного шрифтового сімейства.

2. Щоб підвищити точність і швидкість розпізнавання, було запропоновано низку поліпшень. Насамперед, це механізм повторної сегментації, що дозволяє вирішувати проблему злипання літер, коли кілька символів сприймаються алгоритмом як один. По-друге, у систему додано розширену класифікацію та деталізовану структуру бази даних шрифтів, здатну зберігати більше ознак наприклад, геометричні, метриковані й стилістичні. Завдяки цьому алгоритм може швидше відкидати невідповідні варіанти й коректно визначати потрібне сімейство, враховуючи особливості, притаманні кожній літері.

3. Останнім важливим рішенням стало впровадження функціоналу синтезу шрифтів, що базується на виокремлених сегментованих елементах літер. Ключова перевага в тому, що окремі символи не тільки розпізнаються, а й реконструюються, даючи змогу відтворити оригінальний вигляд кожної літери та, за необхідності, доповнити або змінити її. Цей підхід дає широкі можливості від склеювання розпізнаних сегментів у цілісні шрифтові файли до розробки нових символів у тому

ж стилі. Такий синтез особливо корисний, коли потрібно зберегти автентичну візуальну естетику або відродити унікальний дизайн.

Розробка даного додатку забезпечує однорідність при редагування зображень, і дає можливість створювати однаково стилістичний текст на основі сегментів кожного.

Результати дослідження апробовано та опубліковано у наступній тезі доповіді:

1. Жужков Д.І., Бондарчук А.П. Методика синтезу шрифту за зразком на основі методів розпізнавання зображень. // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії", 26 листопада 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.32-36

ПЕРЕЛІК ПОСИЛАНЬ

1. Smith R. (2007). «Огляд рушія Tesseract OCR». *9-та Міжнародна конференція з аналізу та розпізнавання документів (ICDAR)*, С. 629–633.

2. Rice S. V., Nagy G., & Nartker T. A. (1999). *Оптичне розпізнавання тексту: Ілюстрований путівник по передовому досвіду*. Springer.

3. Hull J. J. (1998). «Виявлення нахилу зображення документа: огляд і анотована бібліографія». *Серія з машинного сприйняття та штучного інтелекту*, World Scientific.

4. Shapiro L. G., & Stockman G. C. (2001). *Комп'ютерний зір*. Prentice Hall.

5. Gonzalez R. C., & Woods R. E. (2018). *Цифрова обробка зображень* (4-те вид.). Pearson.

6. Mori S., Suen C. Y., & Yamamoto K. (1992). «Історичний огляд досліджень та розробок OCR». *Праці IEEE*, 80(7), С. 1029–1058.

7. Otsu N. (1979). «Метод вибору порогу з гістограм відтінків сірого». *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), С. 62–66.

8. LeCun Y., Bengio Y., & Hinton G. (2015). «Глибоке навчання». *Nature*, 521, С. 436–444.

9. Breuel T. M. (2008). «Відкрита OCR-система OCRopus». *IS&T/SPIE Electronic Imaging Conference*, 6815.

10. Wu S., Manmatha R., & Riseman E. M. (1997). «Пошук тексту на зображеннях». *Праці Другої міжнародної конференції ACM з цифрових бібліотек*, 3–12.

11. Viard-Gaudin C., Kim H. S., & Choisy C. (2005). «Стратегії сегментації та розпізнавання для рукописних слів». *International Journal on Document Analysis and Recognition*, 7(1), С. 8–16.

12. Liang J., Doermann D., & Li H. (2005). «Аналіз тексту і документів на основі камер: огляд». *International Journal of Document Analysis and Recognition*, 7(2–3), С.84–104.

13. ABBYY. (2020). *Документація для розробників ABBYY FineReader Engine*. ABBYY.

14. Likforman-Sulem L., Zahour A., & Taconet B. (2007). «Сегментація рядків тексту в історичних документах: огляд». *International Journal on Document Analysis and Recognition*, 9(2–4), С.123–138.

15. Papert S. (1980). *Mindstorms: Діти, комп'ютери та потужні ідеї*. Basic Books (розділ про перші експерименти з розпізнавання тексту та ідейною сегментацією).

16. Garris M. D., & Wilkinson R. A. (1992). «Методи оцінювання ефективності систем OCR для рукописних документів». NIST Special Publication, 500-211.

17. Milyaev S., Bunke H., & Ogier J.-M. (2014). «Нові тенденції у методах розпізнавання документів і графічних структур». Pattern Recognition Letters, 38, С. 125–135.

18. Jain A. K., Duin R. P. W., & Mao J. (2000). «Статистичне розпізнавання образів: огляд». IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1), С. 4–37.

19. Srihari S. N. (2002). «Handwritten Document Image Segmentation and Recognition». Encyclopedia of Computer Science and Technology, 47, С.337–358.

20. Uchida S. (2017). «Text Recognition in Images and Video: Current Problems and Solutions». International Journal of Document Analysis and Recognition, 20(1), С.1–23.

21. Cheriet M., Kharma N., Liu C.-L., & Suen C. Y. (2008). Character Recognition Systems: A Guide for Students and Practitioners. John Wiley & Sons.

22. Smith L. I. (2002). «A Tutorial on Principal Component Analysis (PCA)». Cornell University ArXiv, cs/0205007 (акцент на попередню обробку зображень у OCR).

23. Deng L., & Yu D. (2014). «Глибинні навчальні методи для розпізнавання мовлення та зображень». Foundations and Trends in Signal Processing, 7(3–4), С. 197–387.

24. Kopec G. E., & Chou P. A. (1994). «Document Image Decoding Using Markov Source Models». IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(6), С. 602–617.

25. Casey R. G., & Lecolinet E. (1996). «Стратегії сегментації та розпізнавання рядків для машинного читання поштових адрес». Computer, 29(7), С.25–33.

26. WhatFontIs [Електронний ресурс]. – 2024. – URL: https://www.whatfontis.com/

27. MyFonts [Електронний ресурс]. – 2024. – URL: https://www.myfonts.com/

28. DaFont [Електронний ресурс]. – 2024. – URL: https://www.dafont.com/

29. Google Fonts [Електронний ресурс]. – 2024. – URL: https://fonts.google.com/

30. Font Squirrel [Електронний ресурс]. – 2024. – URL: https://www.fontsquirrel.com/

31. Tesseract OCR [Електронний ресурс]. – 2024. – URL: https://tesseract-ocr.github.io/

32. OCRmyPDF [Електронний ресурс]. – 2024. – URL: https://ocrmypdf.readthedocs.io/

33. Online OCR [Електронний ресурс]. – 2024. – URL: https://www.onlineocr.net/

34. ABBYY FineReader Online [Електронний ресурс]. – 2024. – URL: https://finereaderonline.com/

35. Font Identifier (Matcherator) [Електронний ресурс]. – 2024. – URL: https://www.fontspring.com/matcherator

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО -КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

МЕТОДИКА СИНТЕЗУ ШРИФТУ ЗА ЗРАЗКОМ НА ОСНОВІ МЕТОДІВ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

Виконав: студент групи ПДМ -62 Жужков Д.І.

Керівник: д.т.н., проф., професор кафедри ІПЗАС Бондарчук А.П.

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета дослідження: удосконалення використання релевантних варіантів шрифтів за рахунок методів підбору шрифту та розпізнавання тексту.

Об'єкт дослідження: процес підбору релевантних шрифтів за зразком.

Предмет дослідження: методи аналізу елементів тексту.

Назва шрифтів	Всього запропоновано подібних шрифтів	Процент ефективності з експериментальних даних запропонованих подібних шрифтів	Кількість співпадінь за назвою	Існуючі найпопулярніші Веб-додатки
Helvetica	65	63.07%	2	www.myfonts.com
Times New Roman	57	57.89%	1	
Arial	61	31.14%	4	
Comic Sans	77	0%	0	
Helvetica	22	31.81%	0	www.fontsquirrel.com
Times New Roman	13	53.84%	0	
Arial	19	31.57%	0	
Comic Sans	43	4.65%	0	
Helvetica	20	5%	0	fonts.adobe.com
Times New Roman	20	20%	0	
Arial	20	0%	0	
Comic Sans	20	0%	0	
Helvetica	16	12.5%	2	www.fontspring.com
Times New Roman	5	20%	0	
Arial	14	14.28%	0	
Comic Sans	37	0%	0	
Helvetica	60	73.33%	10	www.whatfontis.com
Times New Roman	60	38.33%	5	
Arial	60	55%	11	
Comic Sans	60	18.33%	5	

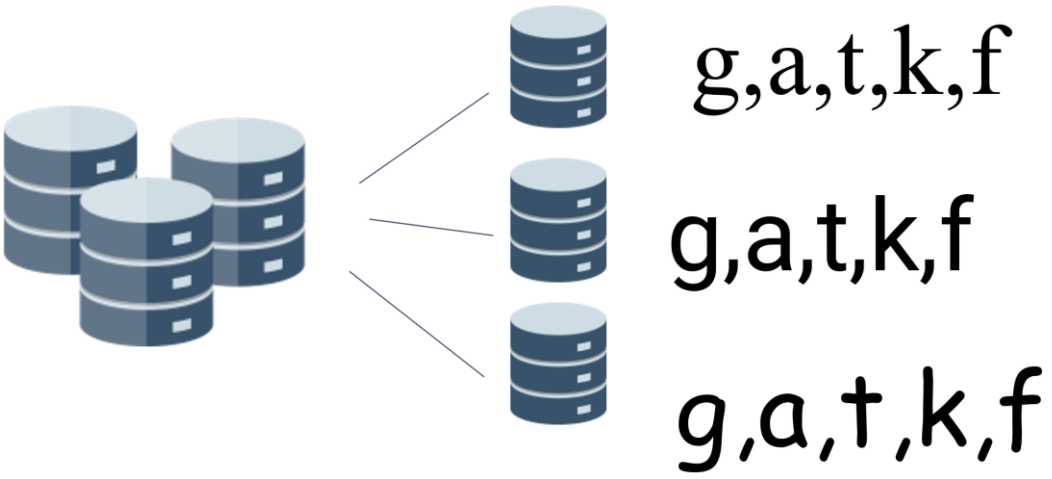
Характерні ознаки для пошуку відмінностей

Методика OCR-системи та вдосконалення:

1. Отримання зображення
2. Попередня обробка зображення
    1. Нормалізація яскравості та контрасту
    2. Бінаризація
    3. Шумозаглушення
    4. Вирівнювання
3. Сегментація
    1. Повторна сегментація об'єднаних елементів (вдосконалення)
4. Постобробка
5. Додавання можливості повторної сегментації для об'єднаних елементів (вдосконалення)
6. Розпізнавання символів
    1. Зменшення кількості НЕ релевантних шрифтів з рахунок класифікації (вдосконалення)

5

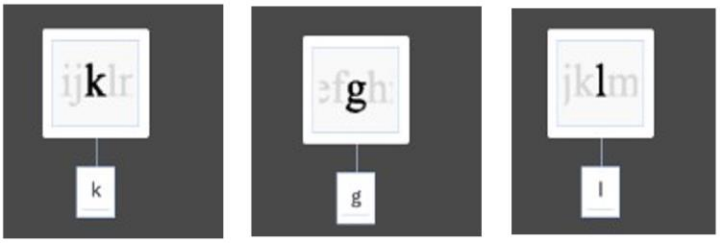
Методика класифікації на розбивку даних шрифтів за окремими характеристиками



6

Вдосконалення постобробки за рахунок додаткової сегментації окремих елементів на основі вказаних значень

Корректно сигментовані елементи:



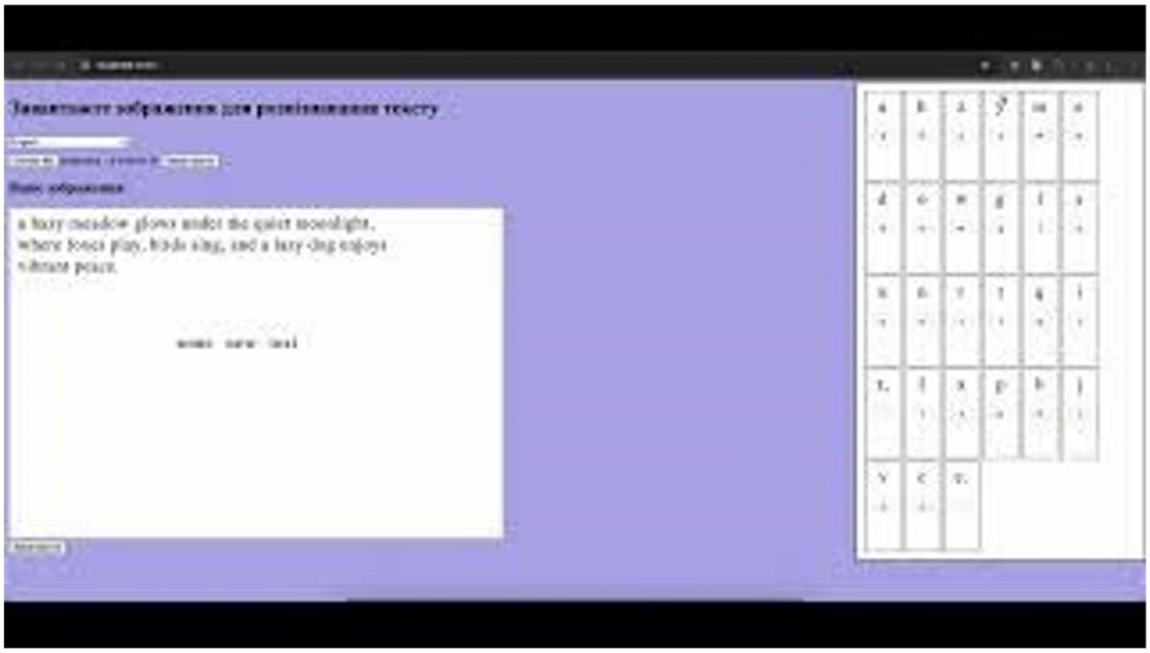
Помилки в сегментації:

7

Принцип забезпечення однорідності при редагуванні документу

8

Реалізація синтезу шрифту і редагування елементу



ВИСНОВКИ

1. На основі аналізу роботи існуючих додатків виявлено основні проблеми підбору релевантних шрифтів.

1. Розроблено методику класифікації шрифтів за індивідуальними ознаками родин для забезпечення більшої релевантності.

1. Покращено роботу сегментації шляхом вказання стиснутих символів для повторної сегментації окремих частин.

1. За допомогою сегментації існуючих елементів та генерування бракуючих через покращену OCR систему забезпечено однорідність символів при редагуванні.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Жужков Д.І.; д.т.н., проф. Бондарчук А.П., Методика синтезу шрифіту за зразком на основі методів розпізнавання зображень. // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії".

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
const express = require('express');
const multer = require('multer');
const tesseract = require('tesseract.js');
const path = require('path');

// Ініціалізація Express
const app = express();
const port = 3000;

// Налаштування Multer для завантаження файлів
const upload = multer({ dest: 'uploads/' });

// Сервірування статичних файлів
app.use(express.static('public'));

// Маршрут для завантаження файлу
app.post('/upload', upload.single('file'), (req, res) => {
    const filePath = path.join(__dirname, req.file.path);

    // Обробка зображення за допомогою tesseract.js
    tesseract.recognize(filePath, 'eng')
        .then(result => {
            res.json({ text: result.data.text });
        })
        .catch(err => {
            console.error(err);
            res.status(500).json({ error: 'Помилка під час розпізнавання тексту' });
        });
});

// Запуск сервера
app.listen(port, () => {
    console.log(`Сервер працює на http://localhost:${port}`);
});
```