Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Дослідження ефективності використання алгоритмів аналізу тексту для оптимізації роботи HR-спеціаліста у сфері IT-рекрутингу

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань

Студентка гр. ІПЗ-24м

_____/ В.В. Швець /

Керівник кваліфікаційної роботи _____ / Н.Н. Шаповалова /

Економіко-організаційна частина _____ /_____ /

Нормоконтроль _____ /_____ /

Завідувач кафедри _____ /А.М. Стрюк/

Кривий Ріг

2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М. Стрюк

«___»_____20__р.

ЗАВДАННЯ

на кваліфікаційну роботу

студентці групи ІПЗ-24м Швець Валерії Володимирівні

1. Тема: Дослідження ефективності використання алгоритмів аналізу тексту для оптимізації роботи HR-спеціаліста у сфері IT-рекрутингу

Затверджено наказом по КНУ № ___ від «__»_____ 20__ р.

2. Термін подання студентом закінченої роботи: «__» _____ 20__ р.

3. Вихідні дані по роботі: аналітичні матеріали щодо застосування алгоритмів обробки природної мови у задачах IT-рекрутингу; система для автоматичного оцінювання відповідності кандидатів вакансії.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): дослідити специфіку та виклики IT-рекрутингу для HR-спеціаліста; проаналізувати сучасні методи NLP; визначити технологічні, функціональні та якісні вимоги до дослідження; обґрунтувати вибір методів текстової векторизації для порівняння; розробити систему аналізу відповідності резюме вакансії.

5. Перелік ілюстративного матеріалу: структурна схема системи, функціональна схема, блок-схеми основних алгоритмів, схема бази даних, макети інтерфейсу користувача.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури за тематикою та збір даних	
2	Аналіз предметної області та наявних аналогів розроблюваної системи	
3	Підготовка матеріалів першого розділу роботи	
4	Розробка алгоритмів та схем програмного комплексу, вибір технологій реалізації	
5	Підготовка матеріалів другого розділу роботи	
6	Розробка програмного забезпечення	
7	Підготовка матеріалів третього розділу роботи	
8	Проведення експерименту та порівняння методів векторизації тексту	
9	Підготовка матеріалів четвертого розділу роботи	
10	Підготовка матеріалів п’ятого розділу роботи	
11	Оформлення пояснювальної записки	

Дата видачі завдання: «__» _____ 20__ р.

Студентка _____ / В.В. Швець /

Керівник роботи _____ / Н.Н. Шаповалова /

РЕФЕРАТ

МЕТОДИ ВЕКТОРИЗАЦІЇ ТЕКСТУ, NLP, TF-IDF, WORD2VEC, ОБРОБКА ТЕКСТУ, ВІДБІР КАНДИДАТІВ, IT-РЕКРУТИНГ, РЕЗЮМЕ, ОПТИМІЗАЦІЯ ПРОЦЕСІВ

Пояснювальна записка: 64с., 10 табл., 21 рис., 1 дод., 20 джерел.

Кваліфікаційна робота присвячена дослідженню ефективності методів векторизації тексту для задачі відбору кандидатів у IT-сфері у процесі рекрутингу.

У першому розділі роботи здійснено аналіз предметної області, розглянуто аналоги розроблюваної системи, проаналізовано наявні дослідження за даною тематикою, сформовано вимоги дослідження та здійснено постановку задачі.

У другому розділі обґрунтовано вибір методів векторизації тексту, розроблено структурну та функціональну схеми програмного комплексу, розроблено блок-схеми основних алгоритмів, обрано технології реалізації системи.

У третьому розділі роботи висвітлено основні етапи реалізації системи: реалізація модулів обробки тексту, векторизації, ранжування, генерації звіту; реалізація бази даних, API та інтерфейсу користувача.

У четвертому розділі роботи проведено порівняльний аналіз використання методів векторизації тексту TF-IDF та Word2Vec; здійснено інтерпретацію результатів їх використання.

У п’ятому розділі роботи проведено аналіз економічної ефективності розробки. Розраховано загальну собівартість впровадження розробки, проаналізовано вигоду та можливі ризики впровадження даного ПЗ.

Результатом роботи стала система аналізу резюме, що дозволить скоротити час на виконання роботи HR-спеціалістом та підвищити ефективність відбору кваліфікованих кандидатів на посаду.

ABSTRACT

TEXT VECTORIZATION METHODS, NLP, TF-IDF, WORD2VEC, TEXT PROCESSING, CANDIDATE SELECTION, IT RECRUITMENT, RESUME, PROCESS OPTIMIZATION

Thesis in 64p., 10 tables, 21 fig., 1 app., 20 references.

The qualification work is devoted to the study of the effectiveness of text vectorization methods for the task of selecting candidates in the IT sphere in the recruitment process.

In the first section of the work, an analysis of the subject area is carried out, analogues of the developed system are considered, existing research on this topic is analyzed, research requirements are formed and the task is formulated.

In the second section, the choice of text vectorization methods is justified, a structural and functional scheme of the software complex is developed, flowcharts of the main algorithms are developed, and system implementation technologies are selected.

In the third section, the main stages of system implementation are highlighted: implementation of text processing modules, vectorization, ranking, report generation; implementation of the database, API and user interface.

In the fourth section, a comparative analysis of the use of the TF-IDF and Word2Vec text vectorization methods is carried out; the results of their use are interpreted.

In the fifth section, an analysis of the economic efficiency of the development is carried out. The total cost of implementing the development is calculated, the benefits and possible risks of implementing software are analyzed.

The result of the work was a resume analysis system that will reduce the time required for the HR specialist to perform the work and increase the efficiency of selecting qualified candidates for the position.

ЗМІСТ

ВСТУП

Проблема найму кваліфікованих працівників завжди була актуальною й досі лишається такою. IT-сектор зараз як ніколи потерпає від нестачі вакантних місць. Проглядається невтішна динаміка для програмістів-початківців та тих, хто шукає роботу: кандидатів стає все більше, вакансій все менше.

Однак, компаніям властиво розширюватись. У такому разі виникає потреба в збільшенні штату працівників. А отже, чим швидше буде вирішено питання найму кадрів, тим краще для компанії. Наразі процес відбору кандидатів стає дедалі складнішим, оскільки обсяг роботи, яку має виконати HR-спеціаліст збільшилася в рази. Традиційний ручний аналіз займає значний час, доволі часто є суб’єктивним та не гарантує ефективного ранжування кандидатів. Рішенням даної проблеми може стати автоматизація процесу підбору працівників.

Стрімкий розвиток методів аналізу тексту, машинного навчання й штучного інтелекту в цілому, відкриває нові можливості в автоматизації даного процесу. Завдяки обчисленню міри схожості між текстами резюме та опису вакансії можливо створити програмне забезпечення, здатне розраховувати коефіцієнт відповідності кандидата вакансії та сприяти ефективнішому прийняттю рішень у сфері IT-рекрутингу.

Таким чином, метою даної роботи є аналіз сфери IT-рекрутингу та дослідження ефективності використання методів аналізу тексту для підвищення роботи HR-фахівців.

1 АНАЛІЗ ПРОБЛЕМИ І ПОСТАНОВКА ЗАДАЧІ РОБОТИ

1.1 Аналіз професійної області проблеми

Перед будь-якою компанією, великою чи малою, завжди постає одне й те саме питання – як за найменший час підібрати кваліфікованих працівників й більш того, знання й навички яких відповідали б максимальній кількості вимог компанії? Раніше такої професії як HR-фахівець, або ж рекрутер, просто не існувало. Рішення щодо персоналу ухвалювали безпосередньо самі керівники, функції ж управління персоналом виконували адміністративні або бухгалтерські служби. З розвитком ринку праці проблема відсутності систематизації у питанні найму, почала набирати великого значення. Через конкуренцію виникає потреба у ефективному пошуку працівників, розвитку персоналу й корпоративної культури. Таким чином, ускладнення економіки, конкуренція на ринку праці, зростання цінності людського капіталу – все це призвело до появи такої професії як HR-менеджер.

На відміну від інших сфер людської діяльності, IT-сфера у другій половині XX століття починає розвиватися за геометричною прогресією. Збільшуються обсяги даних та обчислювальних потужностей, з’являються нові технології та спеціальності. З кожним днем з’являється все більше компаній, продуктів та проектів, а з ними розширюється й штат працівників. Проте, так само актуальною лишається проблема швидкого пошуку та найму кваліфікованих кадрів, вирішити яку має HR-фахівець.

HR-спеціаліст – це фахівець, який відповідає за пошук, відбір та найм працівників. На початку, саме такими були основні задачі та обов’язки представників даної професії. Проте, з часом обов’язки HR-фахівця розширилися і стали включати наступні функції:

- допомога у адаптації та орієнтації співробітників;
- допомога в управлінні пакетами компенсацій та пільг;

- організація тренінгів, вебінарів, воркшопів та курсів для покращення hard- та soft-skills співробітників;
- створення мотиваційної системи;
- організація teambuilding активностей;
- відстеження стану співробітників;
- управління конфліктами;
- підтримка здорової робочої атмосфери.

Збільшення вимог до HR-працівників зумовило розширення інструментів для ефективного виконання їх основних задач. З'являються системи для роботи з персоналом, так звані HRM-системи (Human Resourse Managment), інструменти для пошуку кандидатів, інструменти для роботи з резюме, системи управління кандидатами (Applicant Tracking Systems), інструменти оцінки навичок та відбору, інструменти аналітики та прогнозування. Усі ці інструменти покликані полегшити роботу HR-спеціалістів. Проте, не варто забувати про людський фактор, який може вплинути як на якість роботи HR-працівника, так і на бізнес-процеси всієї компанії. Адже саме від цієї людини залежить наскільки кваліфіковані кадри потраплять у компанію.

Кожного дня HR-фахівці здійснюють пошук кандидатів, переглядають по кілька десятків резюме, беруть участь у співбесідах. Постійна робота з людьми може виснажувати як морально, так і фізично. На тлі втоми та постійного емоційного напруження, може збільшуватися й кількість помилок на роботі, що може призвести до негативних або іноді й критичних наслідків. Компанія може не знайти потрібного кандидата або ж бути незадоволеною роботою найнятого працівника. У результаті чого компанія зазнає фінансових та часових втрат, а у найгірших випадках – удару зазнає репутація бренду роботодавця.

Таким чином, з'являється необхідність у створенні програмного забезпечення, яке дозволило б зменшити ризики впливу людського фактору

на бізнес-процеси компанії. Таким може стати програмне забезпечення створене на основі алгоритмів аналізу тексту.

1.2 Дослідження особливості предметної галузі

Однією з основних задач HR-спеціаліста є робота з текстовою інформацією. Переглянути резюме кандидата, порівняти його на відповідність вимогам вакансії, вивчити супровідний лист кандидата (якщо такий надається). Головною проблемою при ручній обробці таких даних є їх неструктурованість або ж слабка структурованість. Це ускладнює роботу HR-фахівця, особливо при великому обсязі заявок. Однак, у такому випадку стає доцільним створення та впровадження програмного забезпечення для автоматизації процесів на основі методів аналізу тексту.

За допомогою методів аналізу тексту можуть бути автоматизовані наступні процеси:

- витяг структурованої інформації з резюме;
- порівняння кандидата з вимогами вакансії;
- визначення коефіцієнта відповідності кандидата вимогам вакансії;
- ранжування кандидатів за ступенем відповідності вакансії;
- класифікація резюме за спеціальностями.

Текстові дані у IT-сфері мають свої особливості – наявність галузевого жаргону, велика кількість скорочень та англіцизмів. Також, не менш вагомою специфікою є варіативність у подачі однієї і тієї ж за змістом інформації. Так, наприклад, один кандидат може вказати свій рівень знань англійської як B2, інший як Upper-Intermediate. Вигляд такі дані мають різний, проте зміст – однаковий. Таким чином, ускладнюється просте ключове пошукове порівняння. До того ж, деякі кандидати у своїх резюме використовують декілька мов, що у свою чергу, вимагає мультимовної обробки. Такі нюанси зумовлюють потребу у гнучких та інтелектуальних підходах до аналізу

тексту, що забезпечуються інструментами обробки природної мови, або ж Natural Language Processing (NLP).

NLP являє собою розділ штучного інтелекту, який займається взаємодією між комп'ютерами та природною мовою. Цей напрям поєднує в собі методи лінгвістики, інформатики та машинного навчання для забезпечення ефективної взаємодії між людиною та комп'ютером через мову. Тобто, за допомогою NLP машини починають «розуміти» створений людиною текст та осмислено на нього реагувати й коректно інтерпретувати. Основними завданнями NLP є розбиття тексту на окремі слова, визначення частин мови, приведення слів до їх базової форми, виявлення ключових об'єктів у тексті, аналіз емоційної складової повідомлення. Крім того, NLP дозволяє оцінювати ступінь релевантності одного тексту відносно іншого, що є вагомою перевагою у контексті створення програмного забезпечення для автоматизації підбору кандидатів HR-фахівцем.

Отже, враховуючи велику відповідальність покладену на HR-спеціалістів за результат найму працівників, автоматизація процесу ранжування та відбору найкращих кандидатів на вакансію за допомогою алгоритмів аналізу тексту, здатна зменшити вплив людського фактора, підвищити ефективність роботи HR-спеціаліста та зменшити час виконання повторюваних рутинних задач.

Таким чином, застосування алгоритмів аналізу тексту в рекрутингових процесах у IT-сфері є не лише доцільним, а й критично важливим для забезпечення ефективності роботи HR-фахівців.

1.3 Аналіз існуючих аналогів

Наразі більшість компаній задля економії власного часу та ресурсів, затрачених на пошук та підбір кандидатів, використовують вже готові рішення інших компаній. У сфері рекрутингу таких рішень більш ніж достатньо, розглянемо кілька з них.

Першим прикладом рішення для пошуку та підбору кандидатів, який ми розглянемо, є TextKernel Parser [1], головну сторінку якого можна побачити на рисунку 1.1. TextKernel Parser є потужним інструментом, що широко використовується HR-спеціалістами у сфері рекрутингу. Він автоматично аналізує резюме та вакансії з метою оптимізувати процес найму нових співробітників. Основною його функцією є швидке та точне «витягування» структурованої інформації з неструктурованих документів (наприклад, описи вакансій та резюме).

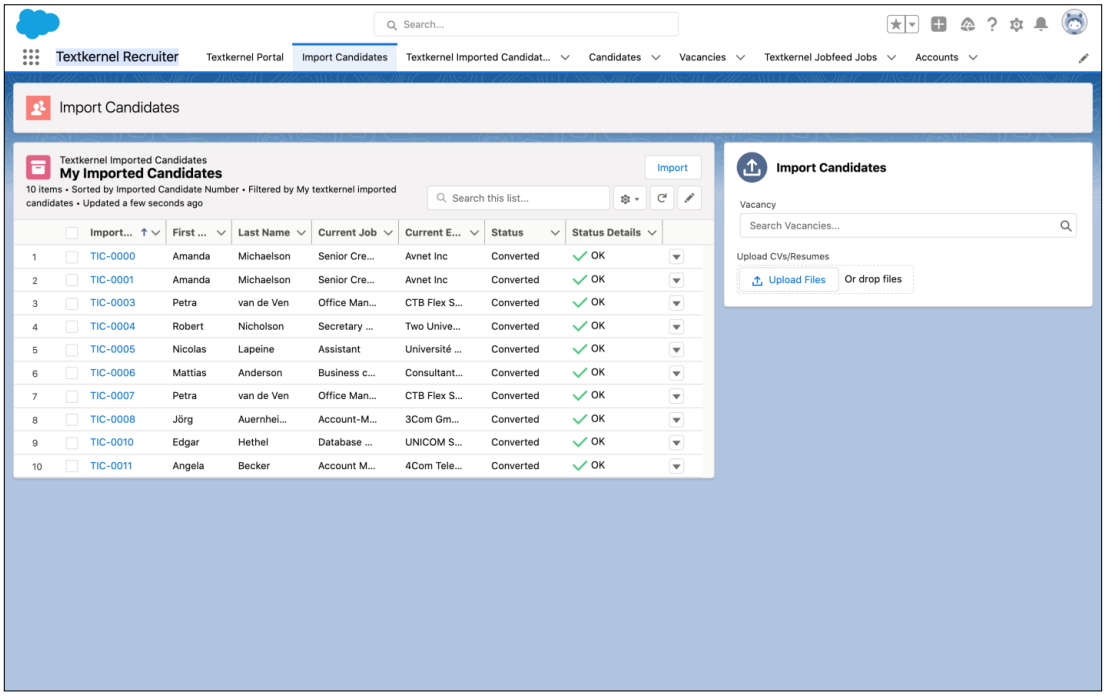


Рисунок 1.1 – Головна сторінка TextKernel Parser

Даний інструмент є дуже гнучким, так як здатен обробляти документи у різних форматах, таких як DOC, PDF, HTML та інших. Вагомою перевагою парсера є мультимовна підтримка. Він підтримує аналіз резюме 29 мовами та описів вакансій 9 мовами. Це значно розширює горизонти компанії, надаючи можливість найняти співробітників з різних куточків світу. TextKernel Parser можна легко інтегрувати з системами управління персоналом, що дозволить автоматизувати процеси рекрутингу та зменшити ручну роботу. Також, за

інформацією яку надала сама компанія, TextKernel впровадив LLM у Parser з можливостями GPT-3.5 для покращення точності та глибини аналізу резюме.

Однак, даний інструмент має й свої недоліки. По-перше, налаштування парсеру може вимагати часу та технічних знань працівників. По-друге, точність роботи парсеру напряму залежить від якості резюме. Неструктуровані або ж погано оформлені документи можуть створити перешкоди для коректного «витягування» даних. По-третє, хоча інструмент і підтримує роботу з багатьма мовами, менш поширені мови можуть не мати повної підтримки. Це, у свою чергу, обмежує його застосування у певних регіонах.

Наступним інструментом від компанії HireAbility є Resume Parser [2], сторінку якого можна побачити на рисунку 1.2. Аналогічно з попереднім інструментом, Resume Parser здатний обробляти документи у форматах DOC, PDF, HTML та інших. Проте, на відміну від TextKernel Parser, Resume Parser підтримує аналіз резюме понад 40 мовами. Також, компанія впровадила технології штучного інтелекту, які поєднують власні технології з можливостями NLP.

Рисунок 1.2 – сторінка HireAbility Resume Parser

Головними недоліками даного інструменту можуть бути низька якість обробки резюме при неструктурованих або погано структурованих вхідних даних та час й складність його налаштування.

Останній аналог, який ми розглянемо, це Eightfold.ai [3], сторінку якого можна побачити на рисунку 1.3. Eightfold.ai являється повноцінною платформою, яка використовує штучний інтелект для оптимізації процесу управління талантами. Її основними функціями виступають підбір працівників, планування робочої сили, розвиток працівників та утримання кадрів. На відміну від переглянутих раніше інструментів, Eightfold.ai орієнтована на великі підприємства.

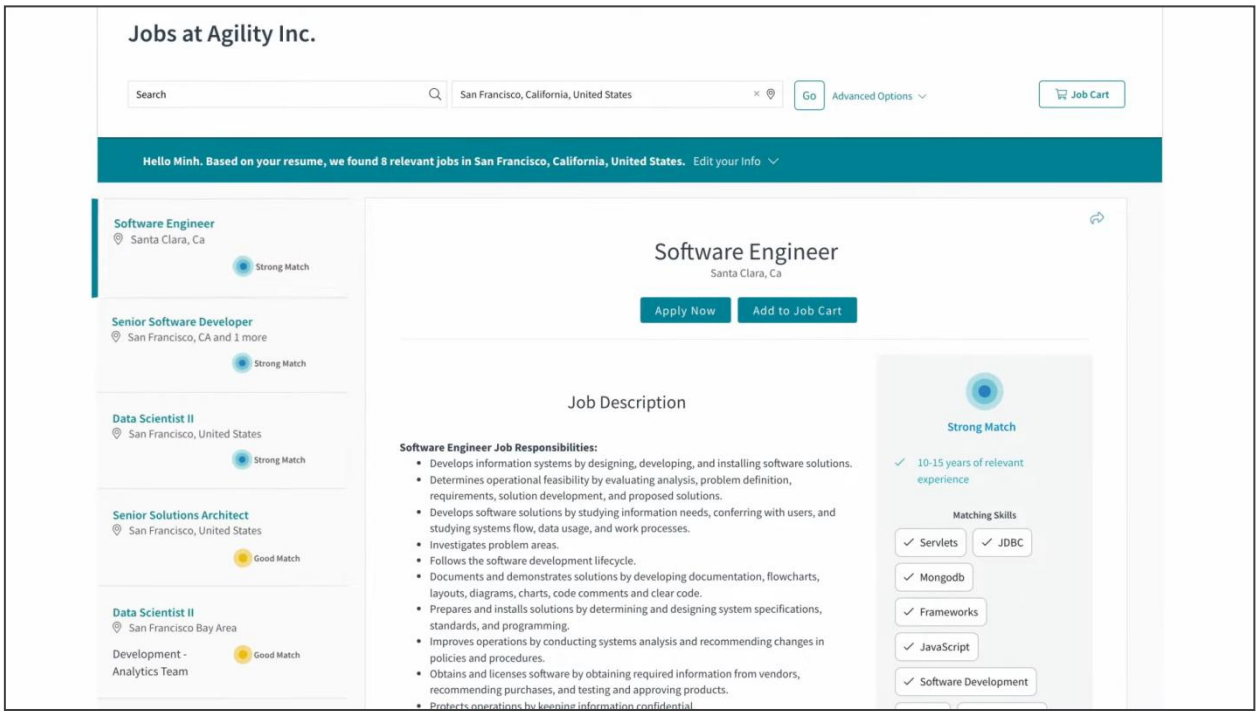


Рисунок 1.3 – сторінка Eightfold.ai

Для забезпечення аналізу понад мільярда кар’єрних траєкторій платформа Eightfold.ai використовує глибоке машинне навчання. Це дозволяє точно зіставляти кандидатів з вакансіями на основі їхніх навичок та досвіду, навіть якщо їхні резюме не містять точного відповідника вимогам вакансії.

Також, платформа надає можливість HR-спеціалістам прогнозувати майбутні потреби у працівниках, ідентифікувати прогалини в навичках та розробляти стратегії розвитку.

Проте, масштабність такої платформи має свої недоліки. При інтеграції з іншими HR-системами можуть виникати труднощі, що може призводити до тривалих затримок та додаткових витрат. Особливо, якщо підприємство-клієнт має комплексну IT-інфраструктуру. Також, широкий набір функцій може призводити до необхідності додаткового навчання HR-команд, оскільки інтерфейс користувача може здаватися менш інтуїтивним.

Таким чином, провівши аналіз аналогів інструментів аналізу резюме можна дійти висновку, що створення інструменту з великою кількістю функцій та охоплення великої сфери діяльності може мати свою ціну у вигляді низької продуктивності та більших затрат часу та фінансів на інтегрування даного інструменту.

1.4 Аналіз досліджень використання методів аналізу тексту у роботі HR-спеціаліста у сфері IT-рекрутингу

На тему використання алгоритмів природної мови для аналізу тексту у сфері рекрутингу є доволі широкий спектр статей та досліджень. Проте, звернемо увагу на дві статті з цієї теми.

Перша стаття має назву «Natural Language Processing (NLP) in AI-Driven Recruitment Systems» автором якої є С. Девараджу [4]. У ній розглядають кілька процесів притаманних сфері рекрутингу, у яких, для автоматизації процесів та підвищення ефективності роботи команди HR-фахівців, можна застосувати алгоритми природної мови: аналіз резюме, зіставлення кандидатів з вакансіями та проведення попередньої оцінки кандидата, генерація стандартних питань на співбесіду (відповідно до вакансії), безпосереднє проведення співбесіди та оцінювання відповідей кандидата, ранжування відповідності кандидатів. Метою роботи є підвищення ефективності процесів добору працівників шляхом використання

сучасних мовних моделей, таких як GPT і BERT, для аналізу резюме, описів вакансій та взаємодії кандидатів.

У роботі також виділено основні переваги використання мовних моделей у сфері рекрутингу. Штучний інтелект значно спростив процес пошуку, відбору та найму працівників. Системи відстеження кандидатів дозволяють перевіряти резюме, зіставляти вакансії з кандидатами та відстежувати найм, що пришвидшує даний процес та зменшує витрати на найм та час його проведення. Згідно зі статтею, 67% респондентів заявили, що ШІ у рекрутингу економить час, автоматизуючи початкову обробку резюме та спілкування з кандидатами, звільняючи час для HR-команд; 43% заявили, що штучний інтелект зменшує упередженість, стандартизуючи відповіді кандидатів за заздалегідь визначеними критеріями, щоб зробити процес найму більш справедливим; 31% респондентів переконані, що ШІ покращує підбір кандидатів на роботу, щоб найкращі кандидати знаходилися коректніше; 30% вважають, що використання рішень для рекрутингу, розроблених на основі штучного інтелекту, знижує витрати, мінімізуючи потребу у великій кількості ручної праці. Також, використання даних методів покращує досвід кандидатів.

Проте, у статті виділено й головні недоліки у використанні ШІ у сфері рекрутингу. Головним недоліком, за який використання штучного інтелекту критикують у цій сфері, є недостатня прозорість. Більшість компаній застосовує алгоритми так званої «чорної скриньки». Це, у свою чергу, може призвести до відсутності підзвітності, якщо алгоритми не дотримуються протоколів зменшення упередженості.

У висновку зазначається, що інтеграція NLP у процеси рекрутингу сприяє підвищенню ефективності, точності та об’єктивності прийняття рішень, а також покращує досвід кандидатів під час взаємодії з системами найму.

Друга розглянута стаття має назву «Challenges and Opportunities of NLP for HR Applications: A Discussion Paper», авторами якої є Йохен Лейднер та

Марк Стівенсон [5]. У даній статті більш широко розглядається процес рекрутингу, аналізу підпадає кожен етап роботи HR-працівника для дослідження доцільності впровадження на даному етапі NLP. Робиться акцент як на уже реалізовані, так і на потенційні варіанти використання природної мови в сфері HR, включаючи інтерактивні чат-боти для самообслуговування працівників; генерацію персоналізованих HR-документів (наприклад, відповіді на часті запитання); автоматичний аналіз ринку праці для виявлення тенденцій у попиті та пропозиції; генерація статистики заробітних плат; асистенти з найму, які допомагають фільтрувати та ранжувати резюме кандидатів; системи рекомендацій, що поєднують шукачів роботи з відповідними вакансіями; інструменти для виявлення прогалин у навичках та планування кар'єрного розвитку.

Не меншого значення у статті набуває питання можливих ризиків та викликів, пов'язаних з впровадженням NLP у бізнес-процеси. Наголошується на можливих упередженнях в алгоритмах, що можуть призвести до дискримінації; на питаннях конфіденційності та персональних даних працівників та кандидатів; на необхідності прозорості та етичності в автоматизованих рішеннях.

Також, у статті коротко викладається зміст суміжних до даної теми робіт інших дослідників, викладається їх головна ідея, обґрунтовуються слабкі та сильні сторони цих досліджень.

У підсумку автори зазначають, що більшість процесів у роботі HR-спеціалістів можуть бути автоматизовані за допомогою алгоритмів природної мови. Однак, слід враховувати етичні, правові та соціальні аспекти при їх впровадженні. Також, не менш важливим моментом є можливість використовувати NLP самими кандидатами для кращого представлення своєї кандидатури на посаду, шляхом формування структурованого та добре стилізованого резюме.

Отже, можна дійти висновку, що використання алгоритмів природної мови для автоматизації бізнес-процесів у сфері рекрутингу не є чимось

цілковито новим. Тим не менш, зацікавленість у розвитку даних алгоритмів саме у напрямі рекрутингу таких світових компаній як Google, Microsoft, IBM робить обрану тему дослідження ще більш цікавою та необхідною для опрацювання.

1.5 Формулювання актуальності дослідження

Розглянувши інструменти, що використовують методи аналізу тексту для автоматизації роботи HR-фахівців та проаналізувавши дослідження з даної теми вже можна дійти висновку, що ця тема досі лишається актуальною. Проте, слід розглянути питання з точки зору працедавців та працівників в Україні.

На тлі подій, що відбуваються у нашій країні, український IT-ринок потерпає від кризи вакантних місць у компаніях. Кількість вакансій зменшується, кількість кандидатів на вакансію – збільшується. За результатами дослідження Work.ua на початку повномасштабного вторгнення ринок праці в категорії «IT, комп'ютери, інтернет» скоротився у 7 разів – з 9887 до 1363 вакансій (Рисунок 1.4 – Динаміка кількості вакансій у категорії з січня 2021 р.).

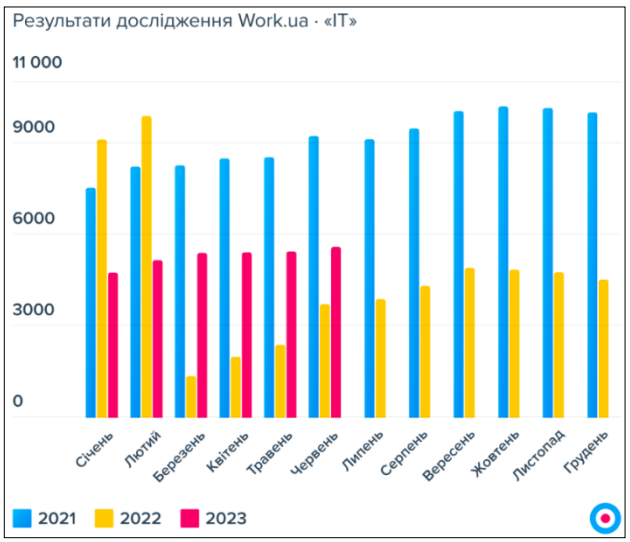


Рисунок 1.4 – Динаміка кількості вакансій у категорії з січня 2021 р. до червня 2023 р.

Таким чином, якщо у довоєнний час кандидат обирав компанію у якій хоче працювати, то на даний момент, компанія обирає кандидата. У результаті цього, збільшується навантаження на HR-працівників. Якщо раніше у середньому HR мав переглянути близько 30 резюме (2021 рік), то зараз ця цифра зросла до 150. З таким об'ємом роботи очевидно, що всі резюме не будуть переглянуті, якщо, наприклад, серед 20 переглянутих кандидатів HR відібрав 5 чи 10 підхожих. Проте, немає гарантії, що відібрані кандидати є «найкращими» зі 150 заявлених.

За допомогою методів штучного інтелекту, а саме алгоритмам обробки природної мови, обсяг роботи HR-працівника можна зменшити в рази і гарантувати відбір «найкращих» кандидатів.

На даний момент існує безліч методів для розпізнавання тексту, завдяки яким стало можливим розрахувати коефіцієнт, що показує, наскільки кандидат відповідає вимогам вакансії. Розрахунок даних коефіцієнтів надає можливість здійснити ранжування кандидатів і таким чином, переглянути резюме від найбільш до найменш підхожих кандидатів.

Крім того, використання методів розпізнавання тексту у роботі HR-працівників є великою перевагою для IT-компаній на сьогоднішній день. Адже, збільшення навантаження на роботі, яка напряму пов'язана з людським фактором, може вести до збільшення помилок через втому та неуважність. У сфері рекрутингу від такої помилки можуть постраждати не лише фінанси компанії, а й кандидати, резюме яких так і не було розглянуто.

Усі вищеперераховані фактори говорять за впровадження інструментів, які на основі методів штучного інтелекту здійснюють ранжування кандидатів на основі вмінь та навичок останніх.

1.6 Визначення вимог дослідження

Так як дослідження ефективності використання алгоритмів аналізу тексту для оптимізації роботи HR-спеціаліста у сфері IT-рекрутингу є доволі

комплексним, стало доцільним визначення основних напрямів, критеріїв та обмежень дослідницької діяльності. Таким чином, вимоги було розбито на кілька рівнів: загальні, функціональні, технічні та науково-дослідні.

До загальних вимог слід віднести вимогу про відповідність сучасному стану розвитку IT-рекрутингу та NLP. Наступною вимогою є необхідність поєднати діяльність HR-фахівців зі штучним інтелектом та алгоритмами аналізу тексту.

Функціональні вимоги до програмного забезпечення включають обробку тексту у найпоширеніших форматах (DOC, PDF), автоматичне розпізнавання ключових слів у текстах резюме та описах вакансій, зіставлення резюме кандидата із описом вакансії, розрахунок коефіцієнту відповідності кандидата вакансії, ранжування кандидатів відповідно до величини коефіцієнта відповідності та формування звітів для подальшої обробки HR-фахівцем.

До технічної складової дослідження висуваються наступні вимоги: використання сучасних NLP-бібліотек, підтримка розпізнавання тексту англійською мовою (так як більшість вакансій та резюме у IT-сфері наразі складаються саме англійською мовою), сумісність із базою даних для збереження результатів аналізу.

Науково-дослідні вимоги – використання різних підходів до аналізу тексту для подальшого проведення порівняльного аналізу, визначення критеріїв оцінки розробленого програмного забезпечення, дослідження впливу використання методів аналізу тексту на ефективність роботи HR-фахівця, формулювання висновків.

Отже, сформовані вимоги дозволяють розглянути дану тему не тільки з погляду теорії, але й реалізувати та дослідити її практичну складову.

1.7 Постановка задачі дослідження

На цьому етапі розглянуто усю теоретичну частину, що стосується даного дослідження, а також сформовано вимоги дослідження. Отже, лишилося сформулювати задачі дослідження.

Таким чином, метою даної роботи являється дослідження впливу алгоритмів аналізу тексту (або ж природної мови, NLP) на ефективність роботи HR-спеціаліста у сфері IT-рекрутингу. Передбачається, що впровадження таких алгоритмів дозволить суттєво скоротити час на первинний відбір кандидатів, підвищити точність відповідності навичок кандидатів до вимог вакансій та знизити кількість помилок у процесі прийняття рішень.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Дослідити та порівняти сучасні алгоритми обробки природної мови, які можуть бути застосовані до задач регресії та порівняння кандидатів на відповідність вакансії;
2. Розробити математичні моделі програмного забезпечення;
3. Розробити структурну та функціональну схеми програмного комплексу;
4. Реалізувати основні алгоритми аналізу тексту з використанням NLP;
5. Провести порівняльний аналіз ефективності використання обраних методів за ключовими метриками.

Очікуваним результатом дослідження є реалізація інструменту, здатного частково автоматизувати процеси попереднього аналізу кандидатів на вакансії у сфері IT, що дозволить HR-фахівцям ефективніше та з меншими витратами часу виконувати рутинну роботу.

2 РОЗРОБКА АЛГОРИТМІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Попередня обробка та формалізація текстових даних

Оскільки HR-спеціаліст працює в основному з неструктурованими документами у різних форматах, виникає необхідність побудови математичної моделі, що дозволить формалізувати вхідні документи.

Вхідними даними у даній задачі є резюме та опис вакансії. Так як, обидва документи являються неструктурованими текстами, їх необхідно представити у вигляді придатному для машинної обробки. Для цього необхідно здійснити попередню обробку тексту – розбити текст на слова (виконати токенізацію), виділити основу слів (стемінг), видалити стоп-слова та здійснити векторизацію тексту.

2.2 Вибір методів векторизації тексту

У даній роботі векторизація тексту грає ключову роль. Завдяки перетворенню текстової інформації у числову форму стає можливим використання алгоритмів машинного навчання. Оскільки тексти резюме та опису вакансії мають різну структуру, обсяг та лексику, безпосереднє порівняння їх у символьному вигляді є неефективним. У цьому випадку, векторизація дозволяє представити резюме та опис вакансії у вигляді векторів у багатовимірному просторі, де значення координат відповідають певним характеристикам слів або термінів. Це забезпечує можливість подальшого обчислення міри схожості між текстами.

Для даного дослідження було обрано два підходи для векторизації тексту – TF-IDF (Term Frequency – Inverse Document Frequency) та Word2Vec. Перший метод забезпечує просте та інтерпретоване частотне представлення документів, тоді як другий дозволяє моделі враховувати семантичні зв'язки між словами.

Вибір саме цих методів векторизації обумовлений необхідністю оцінити ефективність як статистичного, так і семантичного підходів до векторизації тексту. Застосування кожного з них окремо дозволяє виконати порівняльний аналіз точності, швидкодії та здатності виявляти приховані зв’язки між текстами.

2.2.1 Метод векторизації TF-IDF

Одним із найпоширеніших статистичних способів представлення тексту у вигляді векторів є метод TF-IDF [6]. Головною ідеєю цього методу є визначення важливості кожного слова (або ж терміна) для конкретного документа відносно всієї колекції текстів. Вага терміна визначається добутком частоти його появи у документі та оберненої частоти появи у всіх документах:

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D) \tag{2.1}$$

де:

– TF(t,d) – частота терміну у документі, що дорівнює відношенню частоти появи терміну t у документі d;

– IDF(t,D) – рідкість терміну в колекції документів, що дорівнює логарифму відношення загальної кількості документів до кількості документів, що містять термін.

Даний метод векторизації добре працює для коротких та структурованих текстів, таких як резюме або опис вакансії, оскільки дозволяє виділити найбільш значущі терміни для кожного документа.

Його головними перевагами є простота реалізації та інтерпретації, висока швидкодія, мінімальні вимоги до обчислювальних ресурсів. До недоліків TF-IDF слід віднести відсутність врахування семантики та контексту, неможливість розпізнавання синонімів.

Критерій оцінки

Для методу TF-IDF критерій оцінки семантичної близькості між текстом вакансії та резюме визначається за допомогою косинусної подібності.

Позначимо вхідний документ резюме як R, опис вакансії – як V. За допомогою методу TF-IDF обидва документи перетворюються у векторну форму – $\vec{R}$, $\vec{V}$. Таким чином, кожен документ постає у вигляді багатовимірного вектору у просторі ознак.

Основою математичної моделі являється функція відповідності Sim(R, V) (2.2), яка обчислює ступінь релевантності між резюме та вакансією:

$$\boldsymbol{Sim(R,V) = \cos(\vec{R},\vec{V}) = \frac{\vec{R}\cdot\vec{V}}{\|\vec{R}\|\cdot\|\vec{V}\|}} \tag{2.2}$$

Після розрахунку Sim(R, V) формується рейтинг кандидатів, де більше значення свідчить про більшу відповідність кандидата вакансії. Це дасть змогу відібрати найкращих кандидатів та встановити пороги «відсікання».

Можливим покращенням даного рішення може виступати введення вагового коефіцієнта значущості того чи іншого навику кандидата. Дане покращення стане у нагоді у випадку, якщо, критерій досвід роботи, наприклад, є більш вагомим фактором відповідності кандидата вакансії, аніж якісь технічні навички. Таким чином, формула відповідності видозміниться наступним чином:

$$Sim(R,V) = \sum_{i=1}^{n} w_i \cdot \cos(\vec{R}_i,\vec{V}_i) \tag{2.3}$$

де:

- w_i – ваговий коефіцієнт і-ї складової;
- n – загальна кількість складових.

2.2.2 Метод векторизації Word2Vec

Семантичну векторизацію тексту представляє метод Word2Vec [7]. Кожне слово перетворюється у вектор дійсних чисел, який відображає як статистичні закономірності, так і смислові зв’язки між словами. Таким чином, слова, що вживаються у схожих контекстах, мають близькі векторні представлення.

Даний метод базується на навчанні невеликої нейронної мережі, яка вчиться передбачати слово за його контекстом або ж контекст за словом. У результаті формується простір векторів, де семантично подібні слова розташовані поруч.

Перевагами методу Word2Vec є здатність виявляти семантичні зв’язки між словами та робота з контекстом. Головними недоліками є складність інтерпретації результатів та не завжди стабільна якість для коротких документів.

Модель оцінювання

Оскільки дані представлені методом Word2Vec є високо розмірними та містять складні латентні семантичні ознаки (Word2Vec представляє кожне резюме та опис вакансії у вигляді вектора розмірністю близько 100-300 чисел), оптимальним вибором моделі оцінювання якості передбачення відповідності кандидата вакансії є CatBoost [8]. На відміну від лінійних моделей CatBoost чудово працює з семантичними ознаками, а його дерева рішень та їх бустинг здатні моделювати складні нелінійності.

У задачі прогнозування коефіцієнта відповідності кандидата вакансії, CatBoost показує оптимальний баланс між точністю та стабільністю, оскільки розроблений для роботи у ситуаціях коли ознаки високорозмірні, семантика складна, а ціллю виступає прогнозування регресійного значення.

2.3 Розробка структурної і функціональної схеми програмного комплексу

Для розуміння складових та функціональності програмного комплексу було розроблено структурну та функціональну схеми.

На рисунку 2.1 зображено структурну схему програмного забезпечення. Виділено наступні компоненти:

- UI – відповідає за взаємодію з кінцевим користувачем;
- API – взаємодія між сервером та клієнтською частиною;
- Модуль обробки резюме – відповідає за попередню обробку тексту (токенізація, стемінг, видалення стоп-слів);
- Модуль векторизації – відповідає за вибір методу векторизації та його застосування до обробленого тексту;
- Модуль ранжування – відповідає за розрахунок коефіцієнта відповідності резюме опису вакансії та ранжування кандидатів;
- Генерація звіту – відповідає за створення кінцевого звіту на основі даних, отриманих в результаті аналізу резюме та опису вакансії.

Рисунок 2.1 – Структурна схема програмного комплексу

На рисунку 2.2 представлено контекстну функціональну діаграму. У якості вхідних даних виступають файли резюме та опис вакансії, вихідні дані – згенерований звіт.

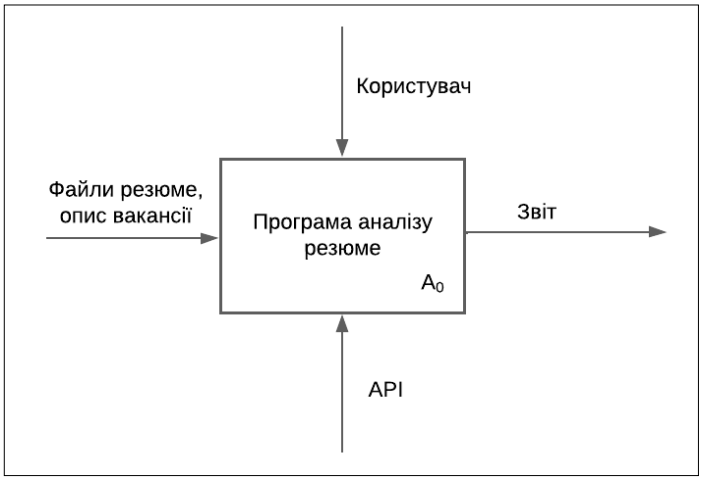


Рисунок 2.2 – Контекстна діаграма

На рисунку 2.3 зображено діаграму першого рівня, яка складається з 5 блоків:

1. Завантаження файлів;
2. Попередня обробка тексту;
3. Векторизація тексту;
4. Розрахунок коефіцієнта відповідності;
5. Генерація звіту.

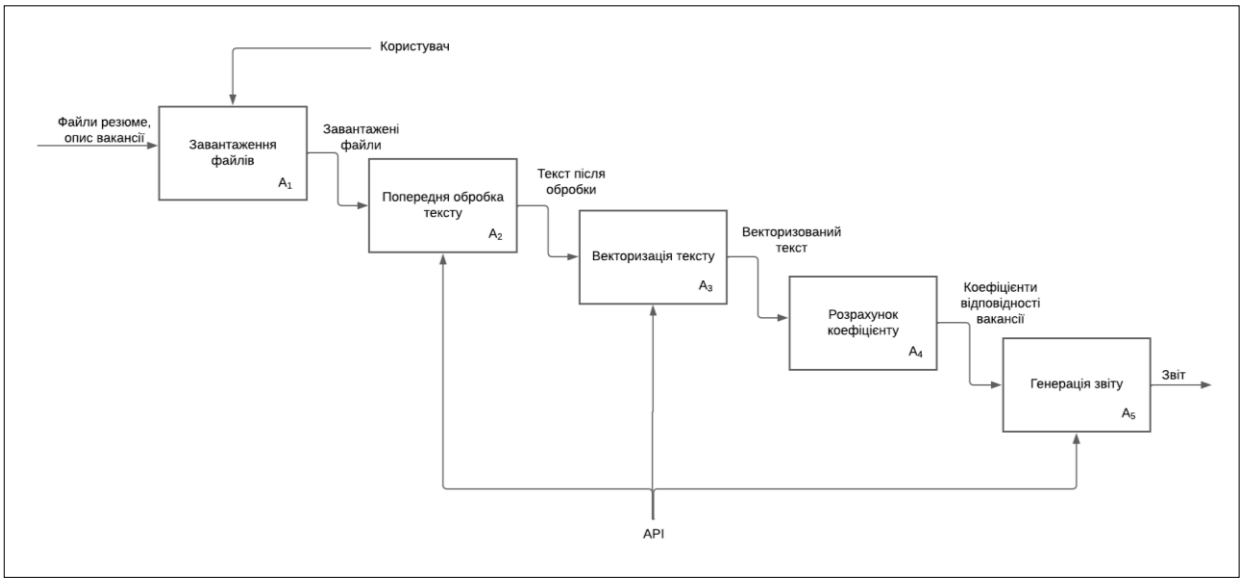


Рисунок 2.3 – Діаграма першого рівня

Для кращого розуміння процесів, що відбуваються на етапі A_2 (попередня обробка тексту), було здійснено його декомпозицію (Рисунок 2.4 – Декомпозиція блоку A_2). У процесі декомпозиції було виділено такі блоки: токенізація тексту, виділення основи слів, видалення стоп-слів.

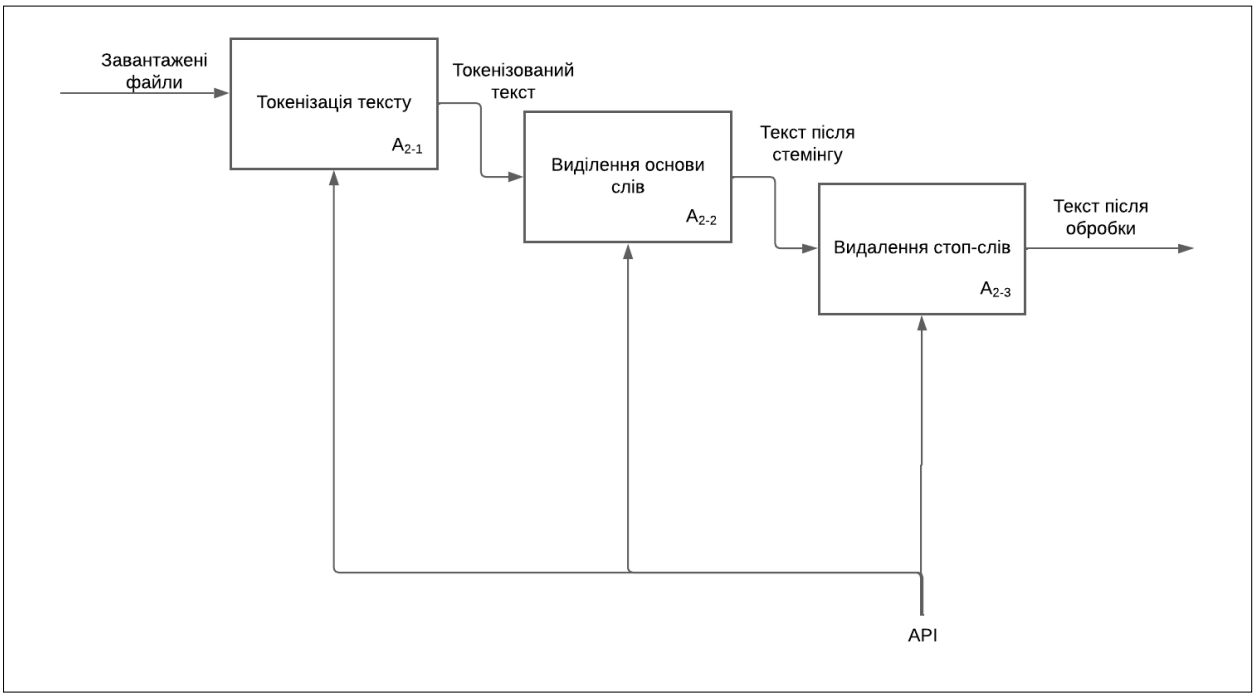


Рисунок 2.4 – Декомпозиція блоку A_2

Також, для розкриття деталей сутності етапу векторизації тексту (блок A_3) було здійснено його декомпозицію. Таким чином, виділено 3 блоки: вибір методу векторизації, перетворення тексту в вектори, нормалізація векторів.

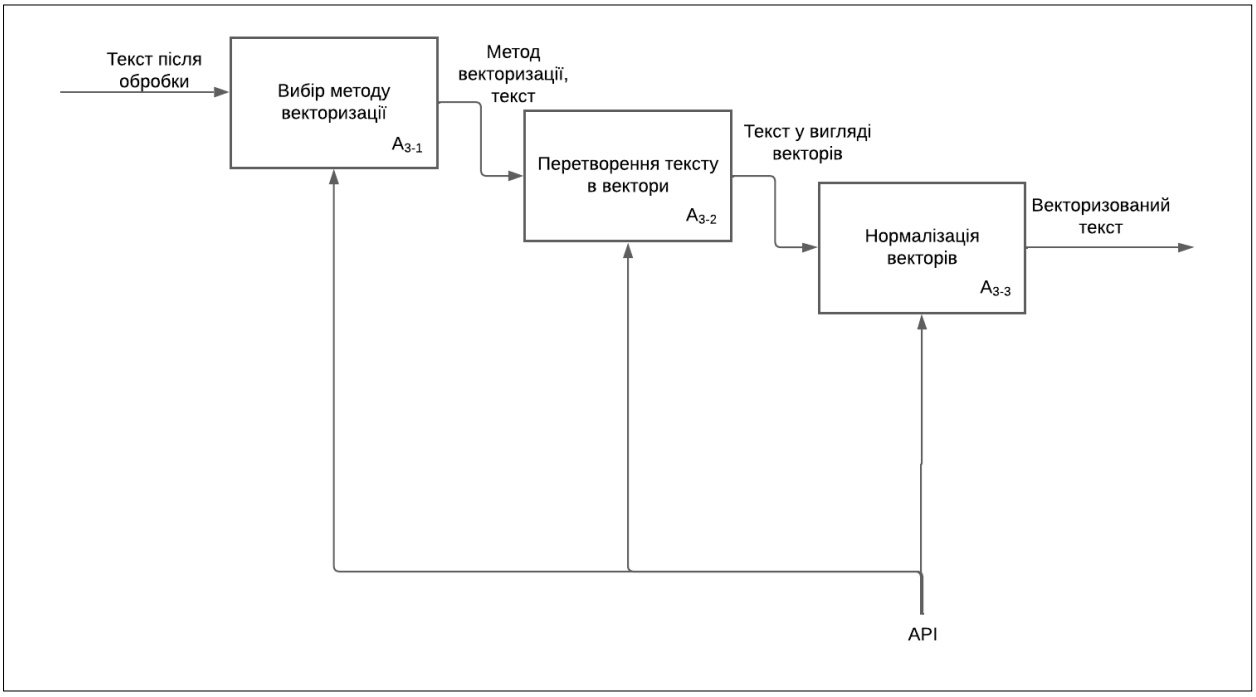


Рисунок 2.5 – Декомпозиція блоку A_3

2.4 Розробка основних алгоритмів програмного комплексу

У процесі порівняння подібності тексту резюме до опису вакансії використовується косинусна відстань. Оскільки, у розрахунок косинусної відстані (2.2) входять довжини векторів резюме та вакансії, виникає потреба у нормалізації векторів з метою уникнення впливу різної довжини векторів на результат.

На рисунку 2.6 зображено блок-схему алгоритму розрахунку Евклідової або ж L2- норми.

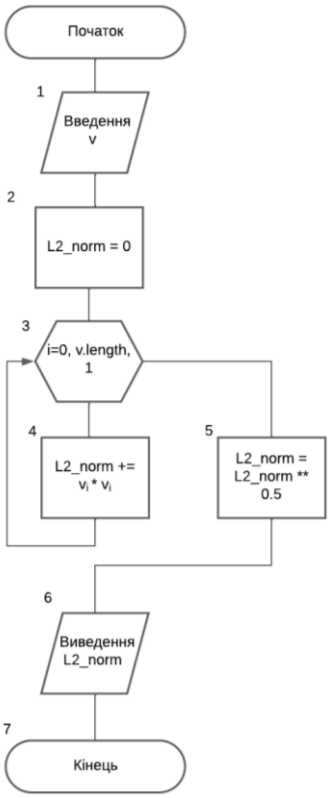


Рисунок 2.6 – Блок-схема алгоритму розрахунку L2-норми

На рисунку 2.7 зображено блок-схему алгоритму розрахунку коефіцієнта відповідності (косинусної відстані).

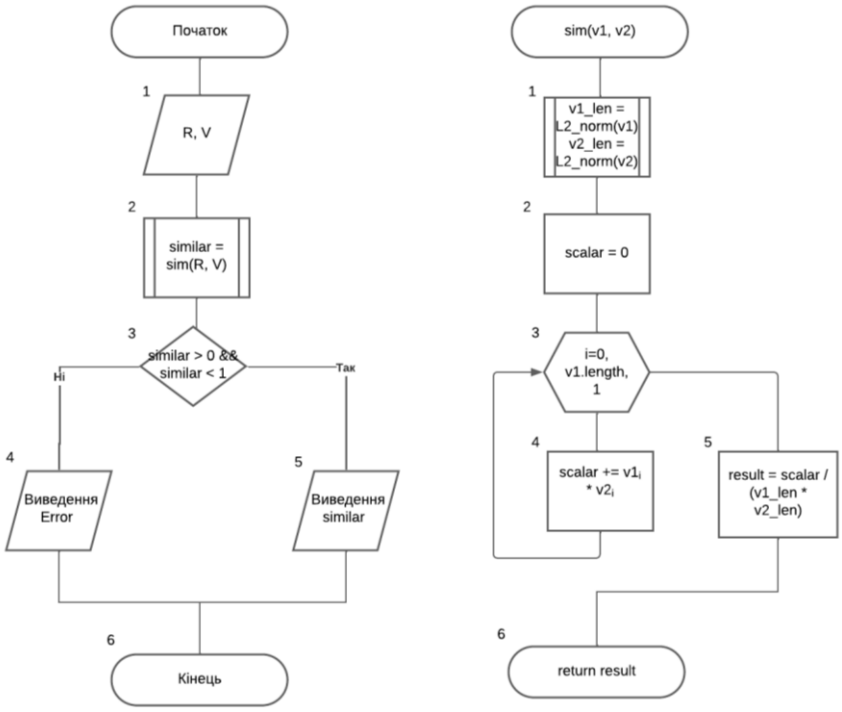


Рисунок 2.7 – Блок-схема алгоритму розрахунку коефіцієнта відповідності

2.5 Вибір технологій реалізації

2.5.3 Технології розробки клієнтської частини

Дане програмне забезпечення задумане у вигляді односторінкового веб-додатку. Таким чином, немає необхідності використовувати для реалізації клієнтської частини надто складних та трудомістких технологій. Проте, ніколи не слід забувати про можливість подальшого розширення програмного продукту, доповнення новим функціоналом. Отже, для розробки клієнтської частини додатку слід обрати технологію, яка дозволить на даному етапі реалізувати весь задуманий функціонал й у подальшому з легкістю його доповнити новими функціями.

Відштовхуючись від вищевикладеного, для розробки клієнтської частини програмного продукту було обрано сучасну JavaScript-бібліотеку – React [9]. Дана бібліотека дозволяє просто та швидко створювати динамічні веб-додатки.

Однією з головних концепцій React є компонентний підхід. Він базується на розбитті інтерфейсу на окремі незалежні компоненти, що таким чином спрощує побудову інтерфейсу, забезпечує його повторне використання та легке масштабування.

React працює з віртуальним DOM, завдяки чому забезпечується плавність і швидкість навіть при складних взаємодіях шляхом мінімальної кількості оновлень реального DOM.

Односторінковий інтерфейс програмного забезпечення ідеально узгоджується з архітектурою React, так як користувач взаємодіє з усіма елементами без переходу між сторінками.

Не менш вагомою перевагою використання React у даному проекті є легка взаємодія з серверною частиною. Це допоможе відокремити логіку обробки резюме від візуалізації результатів.

2.5.4 Технології розробки серверної частини

Якість даного програмного забезпечення напряму залежить від якості та швидкості обробки та аналізу тексту. Враховуючі дані фактори, доцільним буде використання Python-фреймворку FastAPI [10].

Завдяки асинхронній архітектурі FastAPI забезпечує високу продуктивність, що особливо корисно при виконанні ресурсно-затратних обчислень. Даний фреймворк дозволяє досить швидко створювати REST API, які легко інтегруються з клієнтською частиною реалізованою бібліотекою React.

Python містить велику кількість бібліотек для обробки природної мови, машинного навчання та аналізу даних. До них можна віднести scikit-learn, nltk, sentence-transformers, pandas та numpy. Так як, FastAPI добре сумісний з усіма цими бібліотеками це дає змогу ефективно реалізовувати основні алгоритми даного програмного забезпечення без необхідності додаткових надбудов.

Таким чином, використання FastAPI у даному проекті є технічно обґрунтованим рішенням. Він забезпечує баланс між продуктивністю, зручністю розробки, гнучкістю архітектури та можливістю масштабування у майбутньому.

2.6 Прототипування інтерфейсу користувача

При створенні будь-якого програмного забезпечення з часом постає питання розробки інтерфейсу користувача. Зазвичай, до інтерфейсу висуваються наступні вимоги:

- простота використання;
- інтуїтивність;
- мінімалізм;

Враховуючи вищеперераховані вимоги до інтерфейсу користувача, було розроблено прототип односторінкового інтерфейсу зображений на

рисунку 2.8, що реалізує основну функціональність програмного продукту – аналіз відповідності резюме вимогам вакансії. Візуально інтерфейс розділено на два блоки – блок вхідних даних та блок виводу результатів аналізу.

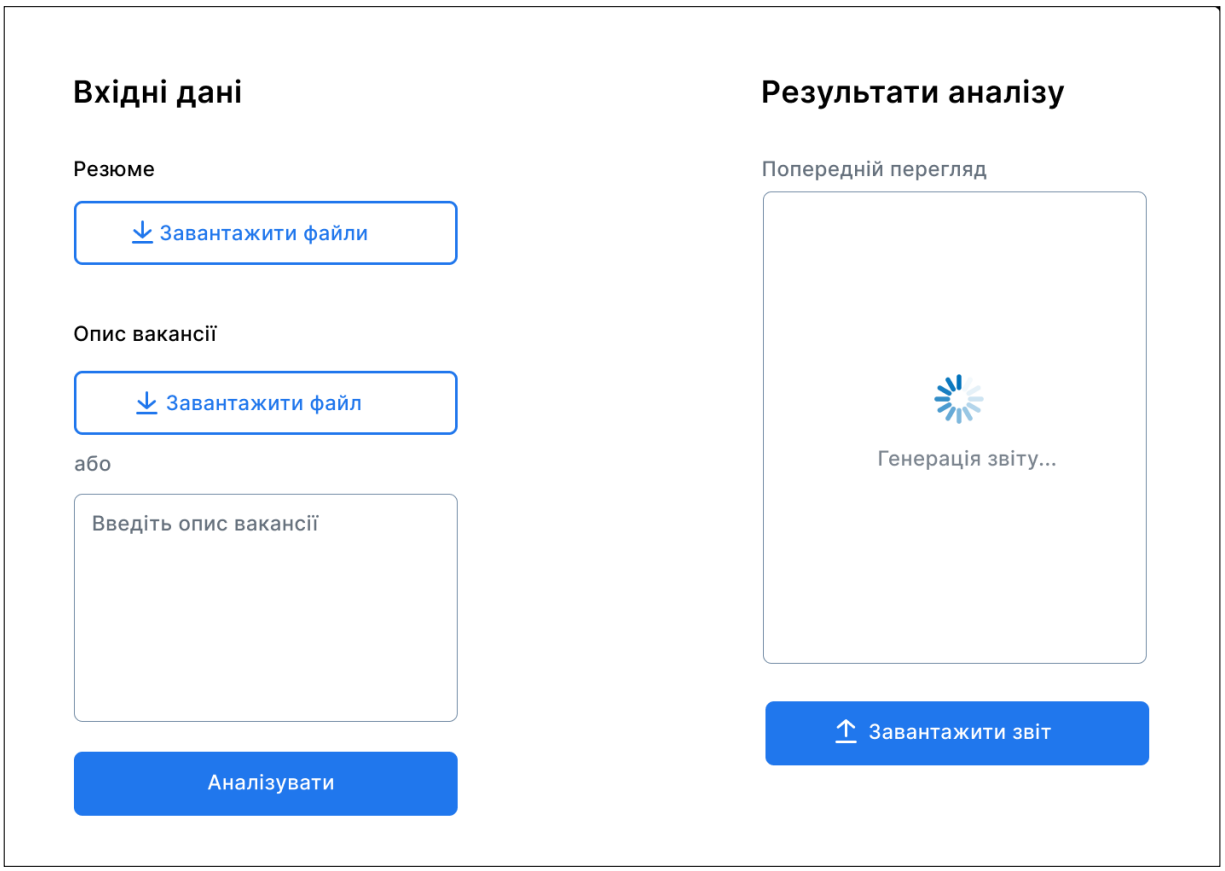


Рисунок 2.8 – Прототип інтерфейсу користувача

Блок вхідних даних дозволяє користувачу завантажити файли резюме, завантажити файл або ввести в ручну опис вакансії, ініціювати процес аналізу резюме натисканням кнопки «Аналізувати».

Блок виводу результатів аналізу надає можливість попереднього перегляду сформованого звіту та можливість завантаження звіту.

Інтерфейс користувача реалізовано з використанням системних шрифтів з гарною читабельністю, а кольорова схема відповідає принципам доступності. Таким чином, інтерфейс відповідає всім зазначеним вимогам.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація модуля обробки тексту

Одним із ключових компонентів розроблюваного програмного комплексу виступає модуль обробки тексту. Даний модуль забезпечує попередню підготовку тексту до подальшого аналізу.

Основною метою модуля обробки тексту є перетворення неструктурованих текстів резюме та опису вакансії у очищений набір лексем, придатний для подальшої математичної обробки.

Для реалізації модуля було використано Python-бібліотеки pdfplumber [11], python-docx [12] та nltk [13]. У функціонал модуля включено наступні етапи:

1. Зчитування тексту з файлів формату .pdf та .docx:

```
def read_pdf(self, file_path):
        text = ""
        with pdfplumber.open(file_path) as pdf:
            for page in pdf.pages:
                page_text = page.extract_text()
                if page_text:
                    text += page_text + "\n"
        return text

def read_docx(self, file_path):
        doc = docx.Document(file_path)
        return "\n".join([pair.text for pair in
doc.paragraphs])
```

2. Попередня очистка тексту, що включає видалення зайвих символів, приведення тексту до нижнього регістру, усунення не буквених та нецифрових символів:

```
 def clean_text(self, text):
         text = text.lower()
         text = re.sub(r'\s+', ' ', text)
         text = re.sub(r'[^a-zA-Za-яA-Я0-9 ]', '', text)
         return text
```

3. Токенізація:

```
def tokenize(self, text):
        return word_tokenize(text)
```

4. Видалення стоп-слів, що включає видалення найпоширеніших службових слів, які не несуть змістового навантаження:

```
def remove_stopwords(self, tokens):
        return [word for word in tokens if word not in
self.stop_words]
```

5. Стемінг, що включає зведення різних форм одного слова до спільного кореня:

```
def stem(self, tokens):
        return [self.stemmer.stem(word) for word in
tokens]
```

У результаті використання модуля обробки тексту отримуємо словник, де ключами виступають назви файлів, а значеннями – списки токенів, отриманих після обробки. Таким чином, формат отриманих даних є зручним для подальшого використання у модулі векторизації.

3.2 Реалізація модуля векторизації

Модуль векторизації у даному програмному комплексі забезпечує перетворення попередньо обробленого тексту у числові вектори. Таким чином, тексти резюме та опису вакансії стають придатними для машинного навчання.

У даному модулі використано два методи векторизації – TF-IDF та Word2Wec. Для реалізації модуля було використано Python-бібліотеки scikit-learn [14], gensim та numpy [15].

При реалізації методу векторизації TF-IDF усі токенізовані тексти об’єднуються у єдиний набір документів у вигляді рядків. Далі виконується

побудова словника та обчислення ваг для кожного слова. На останньому етапі отримані вектори мають форму масиву чисел, що відображають відносну важливість слів у документі:

```
def fit_tfidf(self, tokenized_texts):
        documents = [" ".join(tokens) for tokens in
tokenized_texts]
        self.tfidf_vectorizer = TfidfVectorizer()
        self.tfidf_vectorizer.fit(documents)

def vectorize_tfidf_list(self, tokenized_texts):
        documents = [" ".join(tokens) for tokens in
tokenized_texts]
        return
self.tfidf_vectorizer.transform(documents).toarray()
```

При реалізації методу Word2Wec, модель навчається на основі токенізованих текстів з параметрами (розмір векторного простору, розмір контекстного вікна, мінімальна кількість появ слова для врахування в моделі) й вектор документа формується як усереднення векторів слів, які входять до нього:

```
def train_word2vec(self, tokenized_texts, vector_size=100,
window=5, min_count=1):
        self.word2vec_model = Word2Vec(
            sentences=tokenized_texts,
            vector_size=vector_size,
            window=window,
            min_count=min_count
        )

def transform_word2vec(self, tokens):
        vectors = []
        for word in tokens:
            if word in self.word2vec_model.wv:

vectors.append(self.word2vec_model.wv[word])

        if len(vectors) == 0:
            return
np.zeros(self.word2vec_model.vector_size)

        return np.mean(vectors, axis=0)
```

3.3 Реалізація модуля ранжування

Основною задачею модуля ранжування є порівняння вектора вакансії з векторами резюме та формування відсортованого за спаданням списку кандидатів на посаду відповідно до отриманих показників подібності.

Модуль містить два основні методи – метод розрахунку міри відповідності кожного кандидата вакансії та метод, що сортує кандидатів за отриманими балами та формує фінальний рейтинг.

Робота методу розрахунку міри відповідності кандидата вакансії відрізняється в залежності від застосованого методу векторизації тексту (TF-IDF та Word2Vec). При використанні методу векторизації TF-IDF здійснюється розрахунок косинусної подібності вектора вакансії та резюме, при використанні методу Word2Vec застосовується семантична модельна оцінка:

```
def ranking_scores(self, vacancy_vector,
resume_vectors_dict):

        scores = {}

        if self.mode == "tfidf":
            v = vacancy_vector.reshape(1, -1)

            for candidate_id, r in
resume_vectors_dict.items():

                r_vec = r.reshape(1, -1)
                score = cosine_similarity(v, r_vec)[0][0]
                scores[candidate_id] = score

        elif self.mode == "word2vec":

            evaluator = CandidateEvaluator()

            for candidate_id, resume_vec in
resume_vectors_dict.items():
                score = evaluator.evaluate(resume_vec,
vacancy_vector)
                scores[candidate_id] = score

        return scores
```

Метод формування фінального рейтингу кандидатів викликає функцію обчислення оцінок відповідності, сортує отримані бали у порядку спадання і формує рейтинг із номером кожного кандидата в підсумковій таблиці:

```
def rank_candidates(self, vacancy_vector,
resume_vectors_dict):

        scores = self.ranking_scores(vacancy_vector,
resume_vectors_dict)

        ranking = sorted(scores.items(), key=lambda x:
x[1], reverse=True)

        return [(candidate_id, score, rank + 1)
                for rank, (candidate_id, score) in
enumerate(ranking)]
```

3.4 Реалізація модуля генерації звіту

Модуль генерації звіту виконує функцію підсумкового представлення результатів ранжування кандидатів у зручному та придатному для подальшої роботи форматі. Він дозволяє експортувати результати обробки резюме до табличного файлу, що може бути використаний HR-фахівцем для аналізу, фільтрації та прийняття кадрових рішень.

Створений звіт фіксує ступінь відповідності кожного кандидата вимогам вакансії, виражений у відсотках, що значно спрощує інтерпретацію результатів автоматизованого оцінювання.

Модуль реалізовано з використанням Python-бібліотеки pandas [16]. Він містить два методи – метод для формування структурованих табличних даних на основі результатів ранжування та метод для збереження даних у форматі Excel.

```
def create_dataframe(self, ranking):

        data = []
        for name, score, _ in ranking:
            percent = round(score * 100, 2
            data.append([name, percent])
```

```
        df = pd.DataFrame(data, columns=["Кандидат",
"Відсоток відповідності (%)"])
        return df

def save_to_excel(self, ranking,
output_path="report.xlsx"):

        df = self.create_dataframe(ranking)
        df.to_excel(output_path, index=False)
```

3.5 Реалізація бази даних

База даних системи призначена для зберігання інформації про кандидатів, їхні резюме, вакансії та результати аналізу відповідності між ними. Модель даних побудована за реляційним підходом і складається з чотирьох основних сутностей: Candidate, Resume, Vacancy та FitAnalysis (Рисунок 3.1 – Схема бази даних).

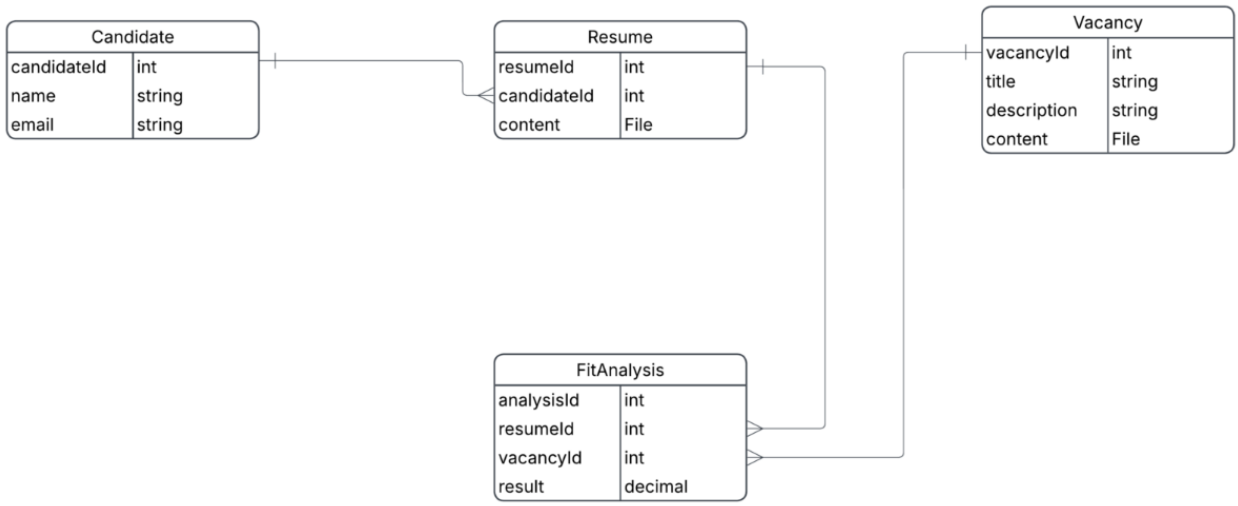


Рисунок 3.1 – Схема бази даних

Таблиця Candidates зберігає основні відомості про кандидатів. Кожен кандидат може мати одне або декілька резюме, що реалізує зв’язок один-до-багатьох із таблицею Resumes. Resumes пов’язана з таблицею Candidates через поле candidateId. Один кандидат може мати кілька резюме, але кожне резюме належить лише одному кандидату.

Таблиця Vacancies містить відомості про відкриті вакансії, кожна вакансія може бути використана для порівняння з багатьма резюме.

Таблиця FitAnalysis використовується для збереження результатів автоматизованого аналізу відповідності резюме певній вакансії. FitAnalysis реалізує зв’язок багато-до-багатьох між резюме та вакансіями. Одне резюме може бути оцінене для різних вакансій, так само одна вакансія може мати результати оцінювання для багатьох резюме.

Розроблена структура забезпечує централізоване зберігання всіх необхідних даних для системи оцінювання відповідності кандидатів вакансіям. Вона допускає підключення додаткових сутностей і може бути розширена без порушення існуючих зв’язків. Модель є придатною для інтеграції з алгоритмами автоматичного аналізу або машинного навчання, оскільки у сховищі вже передбачено окрему таблицю для збереження результатів оцінки.

3.6 Реалізація API

При розробці комплексних систем важливим кроком стає документування етапів розробки, тому в цьому підрозділі розглянуто документування API з бекенду, створене для взаємодії бекенду та фронтенду.

POST /api/analysis

Ендпоінт призначений для запуску аналізу відповідності між резюме та вакансією. Підтримує декілька сценаріїв використання, що дозволяє як завантажувати нові файли, так і використовувати наявні записи. Бекенд самостійно визначає кандидата, якому належить резюме, шляхом аналізу вмісту файлу.

Запит може виконуватися у форматі multipart/form-data або application/json залежно від обраного сценарію.

Параметри запиту та їх опис представлені у таблиці 3.1.

Таблиця 3.1 – Параметри запиту для запуску аналізу відповідності між резюме та вакансією

Назва	Тип	Опис
resumeFile	File	Нове резюме у форматі PDF, DOCX. При передачі цього поля система створює новий запис Resume і автоматично визначає відповідного кандидата.
vacancyFile	File	Файл із описом вакансії. При передачі цього поля система створює новий запис Vacancy.
resumeId	int	Ідентифікатор вже існуючого резюме. Якщо передано – resumeFile не потрібен.
vacancyId	int	Ідентифікатор існуючої вакансії. Якщо передано – vacancyFile не потрібен.

Підтримувані сценарії:

1. Передано resumeFile та vacancyFile. Система створює нове резюме, визначає кандидата, створює вакансію та проводить аналіз між новими записами.

2. Передано resumeFile та vacancyId. Створюється нове резюме, визначається кандидат, але використовується існуюча вакансія.

3. Передано resumeId та vacancyFile. Використовується існуюче резюме, але створюється нова вакансія.

4. Передано resumeId та vacancyId. Обидва записи вже існують, виконується лише аналіз.

5. Передано лише resumeFile (без вакансії). Система створює резюме та кандидата, але повертає помилку, якщо вакансія не передана у будь-якому вигляді.

6. Передано лише vacancyFile (без резюме). Створюється вакансія, але аналіз виконати неможливо – повертається помилка.

7. Нічого не передано. Повертається помилка з описом необхідних параметрів.

Формат відповіді:

```
{
   "analysisId": 142,
   "resumeId": 21,
   "vacancyId": 7,
   "candidateId": 4,
   "result": 0.82,
   "date": "2025-11-20T15:12:00"
}
```

GET /api/analysis

Повертає історію виконаних аналізів. Параметри фільтрації необов’язкові та можуть комбінуватися за правилом логічного І. Якщо параметри не задано – повертається повна історія. Параметри запиту наведено у таблиці 3.2.

Таблиця 3.2 – Параметри запиту на отримання історії виконаних аналізів

Назва	Тип	Опис
candidateId	int	Повертає всі аналізи, що стосуються заданого кандидата.
resumeId	int	Повертає всі аналізи для конкретного резюме.
vacancyId	int	Повертає всі аналізи для вказаної вакансії.
from	date	Початкова дата фільтрації (ISO формат).
to	date	Кінцева дата фільтрації (ISO формат).

Приклад відповіді:

```
[
 {
   "analysisId": 142,
   "resumeId": 21,
   "candidateId": 4,
   "vacancyId": 7,
   "result": 0.82,
   "date": "2025-11-20T15:12:00"
 }
]
```

GET /api/resumes

Повертає список резюме з мінімальними метаданими. Файли не передаються напряму.

Приклад відповіді:

```
[
 {
   "resumeId": 21,
   "candidateId": 4,
   "fileName": "resume_john_smith.pdf",
   "uploadDate": "2025-10-11T13:00:00"
 }
]
```

GET /api/resumes/{resumeId}

Повертає файл резюме.

GET /api/vacancies

Повертає список усіх вакансій.

Приклад відповіді:

```
[
 {
   "vacancyId": 7,
   "title": "Backend Developer",
   "created": "2025-09-05T14:21:00"
 }
]
```

GET /api/vacancies/{vacancyId}
Повертає файл з описом вакансії

3.7 Реалізація інтерфейсу користувача

На рисунку 3.1 зображено інтерфейс користувача, реалізований за допомогою JavaScript-бібліотеки React. Основною концепцією бібліотеки є компонентний підхід. Завдяки такому підходу стає можливим швидке розширення та масштабування проекту, повторне використання компонентів, чітке розділення логіки та відображення.

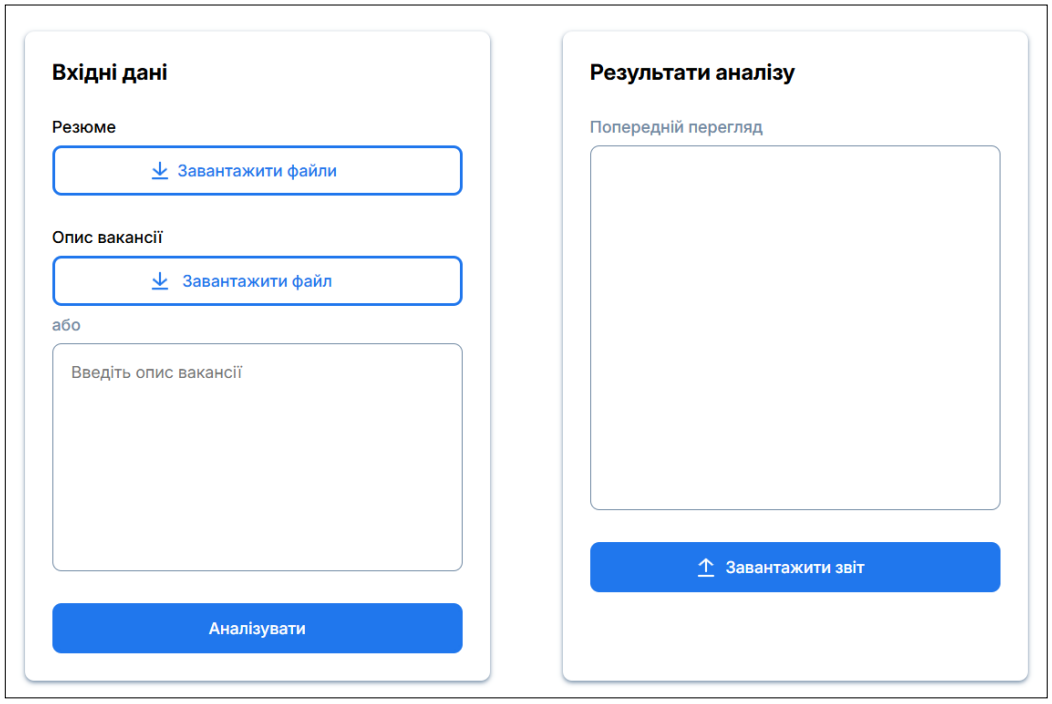


Рисунок 3.2 – Інтерфейс користувача

Взаємодія з бекендом реалізована через REST API, розроблений на основі FastAPI. Передача даних здійснюється за допомогою HTTP-методу POST. Користувач відправляє на сервер опис вакансії та набори резюме, а у відповідь отримує структуровані результати ранжування.

4 АНАЛІЗ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ МЕТОДІВ ВЕКТОРИЗАЦІЇ ШЛЯХОМ ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ

4.1 Формування експериментальних умов та метрик для моделей векторизації

4.1.1 Вхідні дані

У роботі використовувалися текстові документи двох типів: опис вакансії та резюме кандидатів у форматі PDF. Перед застосуванням методів векторизації до текстів було виконано однаковий набір етапів нормалізації, що включали токенізацію, стемінг та видалення стоп-слів.

Для навчання моделі CatBoostRegressor [17], яка використовувалася для оцінювання релевантності кандидатів, застосовувалися навчальні приклади , що містили текстові векторні подання та відповідні мічені значення, загальною кількістю 800 зразків.

Тестування роботи методів TF-IDF та Word2Vec здійснювалося на наборі із 60 файлів резюме кандидатів.

4.1.2 Параметри моделей TF-IDF та Word2Vec

Для векторизації текстів методом TF-IDF було використано клас TfidfVectorizer [18]. Було використано стандартні параметри, головними з яких виступають:

1. lowercase = true – приведення усіх символів до нижнього регістру перед токенізацією;

2. ngram_range = (1,1) – використовується лише уніграмний аналізатор;

3. min_df = 1 – під час побудови словника враховується слово, навіть якщо воно зустрічається лише в одному документі;

4. max_features = None – використовуються усі ознаки;

5. smooth_idf = true – використання згладжування, що запобігає діленню на нуль.

Використання даних параметрів дозволяє зберегти всю інформативність та покращити стійкість моделі при малих наборах даних.

Для векторизації текстів методом Word2Vec було використано клас Word2Vec бібліотеки gensim [19]. Було використано наступні параметри:

1. vector_size = 200 – розмірність векторів слів;

2. window = 5 – максимальна відстань між словами в контексті;

3. min_count = 1 – під час побудови словника враховуються навіть рідкісні слова.

Використання даних параметрів є оптимальним варіантом для невеликих наборів текстів – розмірність векторів слів забезпечує достатню інформативність та водночас запобігає перенавчанню моделі; максимальна відстань між словами у тексті дозволяє моделі бачити відносно широкий контекст; параметр min_count забезпечує включення всіх термінів (якими можуть бути фреймворки або ж рідкісні технології).

4.1.3 Метрики оцінки ефективності

Для порівняльного аналізу методів TF-IDF та Word2Vec у задачі визначення відповідності резюме опису вакансії використовувалися метрики подібності та оцінки точності моделей.

Основною метрикою оцінки ефективності при використанні методу TF-IDF була косинусна подібність, що вимірює кут між векторами документів. Цей критерій дозволяє оцінити семантичну близькість між резюме та описом вакансії.

Для моделі CatBoostRegressor, яка приймає векторизовані подання текстів Word2Vec як вхідні дані, використовувалися регресійні метрики, а саме MAE (Mean Absolute Error), MSE (Mean Squared Error) та R^2 (коефіцієнт детермінації) [20]. У результаті навчання моделі було отримано наступні значення метрик:

— MAE = 6,78 – у середньому модель помиляється на 6,78 пунктів під час прогнозування відповідності кандидата вакансії;
— MSE = 45,12 – вказує на стабільність моделі, оскільки сильні відхилення від реальності рідкі;
— R^2 = 0,84 – модель пояснює 84% залежності між резюме та відсотком відповідності вакансії.

Значення даних метрик дозволяють стверджувати, що модель стабільна, не перенавчається і добре працює з текстовими векторами.

4.2 Постановка експерименту

Мета: дослідити та оцінити ефективність використання методів текстової векторизації TF-IDF та Word2Vec у задачі аналізу резюме кандидатів для оптимізації відбору в IT-рекрутингу, визначивши, який з методів забезпечує вищу якість представлення текстових даних для подальшої автоматизованої обробки HR-спеціалістом.

Гіпотеза: метод Word2Vec забезпечує статистично значуще вищу якість аналізу текстів резюме, ніж TF-IDF, завдяки врахуванню семантичних зв’язків між словами.

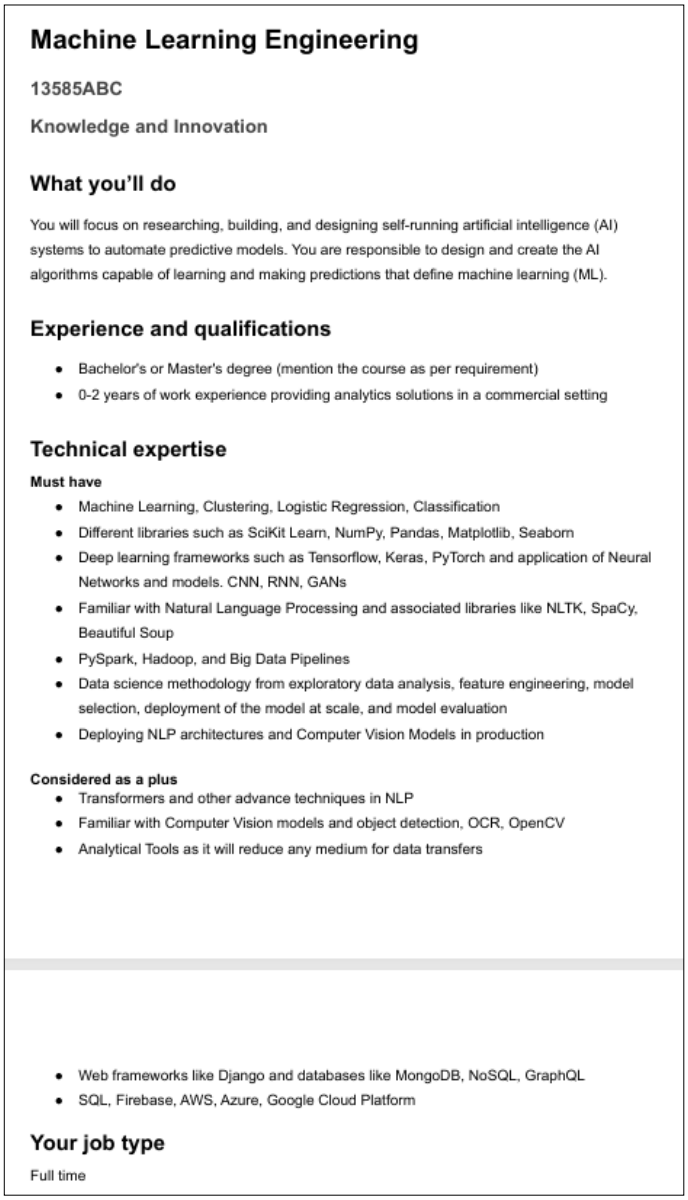
Хід роботи

Для порівняння результатів роботи методів TF-IDF та Word2Vec було використано однаковий набір вхідних даних: один опис вакансії та 60 резюме кандидатів на дану вакансію.

Для методу TF-IDF оцінка відповідності здійснювалася на основі косинусної подібності між векторами документів.

Для методу Word2Vec було попередньо навчено модель CatBoostRegressor, яка прогнозувала оцінку відповідності на основі векторизованих представлень резюме та вакансії.

Текст опису вакансії представлено на рисунку 4.1.



Machine Learning Engineering

13585ABC

Knowledge and Innovation

What you'll do

You will focus on researching, building, and designing self-running artificial intelligence (AI) systems to automate predictive models. You are responsible to design and create the AI algorithms capable of learning and making predictions that define machine learning (ML).

Experience and qualifications

- Bachelor's or Master's degree (mention the course as per requirement)
- 0-2 years of work experience providing analytics solutions in a commercial setting

Technical expertise

Must have

- Machine Learning, Clustering, Logistic Regression, Classification
- Different libraries such as SciKit Learn, NumPy, Pandas, Matplotlib, Seaborn
- Deep learning frameworks such as Tensorflow, Keras, PyTorch and application of Neural Networks and models. CNN, RNN, GANs
- Familiar with Natural Language Processing and associated libraries like NLTK, SpaCy, Beautiful Soup
- PySpark, Hadoop, and Big Data Pipelines
- Data science methodology from exploratory data analysis, feature engineering, model selection, deployment of the model at scale, and model evaluation
- Deploying NLP architectures and Computer Vision Models in production

Considered as a plus

- Transformers and other advance techniques in NLP
- Familiar with Computer Vision models and object detection, OCR, OpenCV
- Analytical Tools as it will reduce any medium for data transfers

- Web frameworks like Django and databases like MongoDB, NoSQL, GraphQL
- SQL, Firebase, AWS, Azure, Google Cloud Platform

Your job type

Full time

Рисунок 4.1 – Файл вакансії

Результати ранжування кандидатів та визначення п’яти найбільш релевантних резюме для кожного метода наведено у таблицях 4.1 та 4.2

Таблиця 4.1 – Топ-5 кандидатів за методом TF-IDF

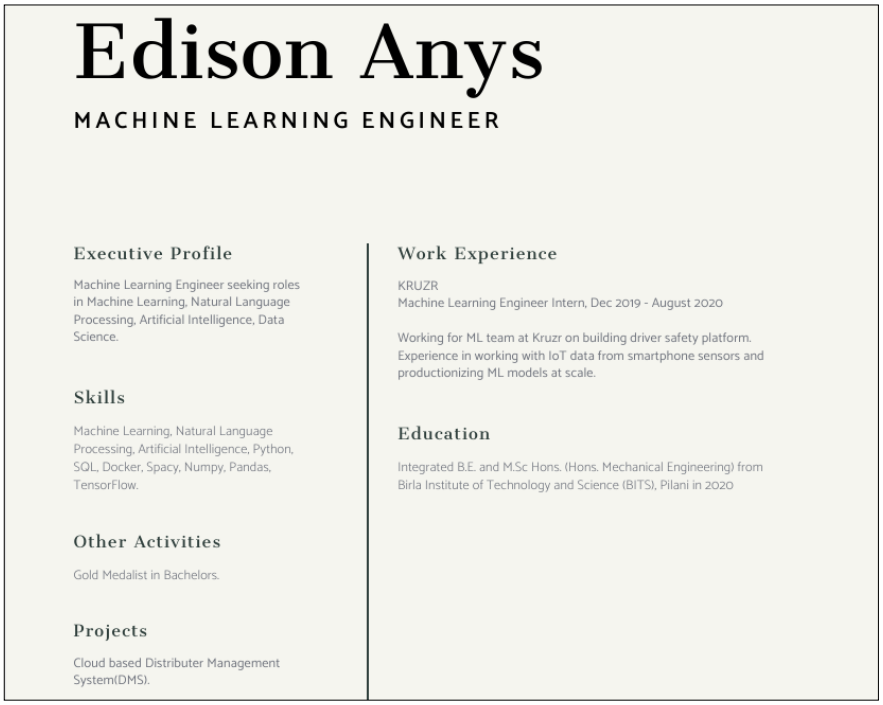
Ранг	Кандидат	Відсоток відповідності (%)
1	Candidate_069	51,12
2	Candidate_019	49,76
3	Candidate_099	45,32

Ранг	Кандидат	Відсоток відповідності (%)
4	Candidate_107	30,12
5	Candidate_017	21,22

Таблиця 4.2 – Топ-5 кандидатів за методом Word2Vec

Ранг	Кандидат	Відсоток відповідності (%)
1	Candidate_107	58,43
2	Candidate_051	43,97
3	Candidate_031	38,54
4	Candidate_019	33,11
5	Candidate_017	29,37

Тексти резюме отриманих кандидатів зображено на рисунках 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 та 4.8.



Edison Anys

MACHINE LEARNING ENGINEER

Executive Profile

Machine Learning Engineer seeking roles in Machine Learning, Natural Language Processing, Artificial Intelligence, Data Science.

Skills

Machine Learning, Natural Language Processing, Artificial Intelligence, Python, SQL, Docker, Spacy, Numpy, Pandas, TensorFlow.

Other Activities

Gold Medalist in Bachelors.

Projects

Cloud based Distributer Management System(DMS).

Work Experience

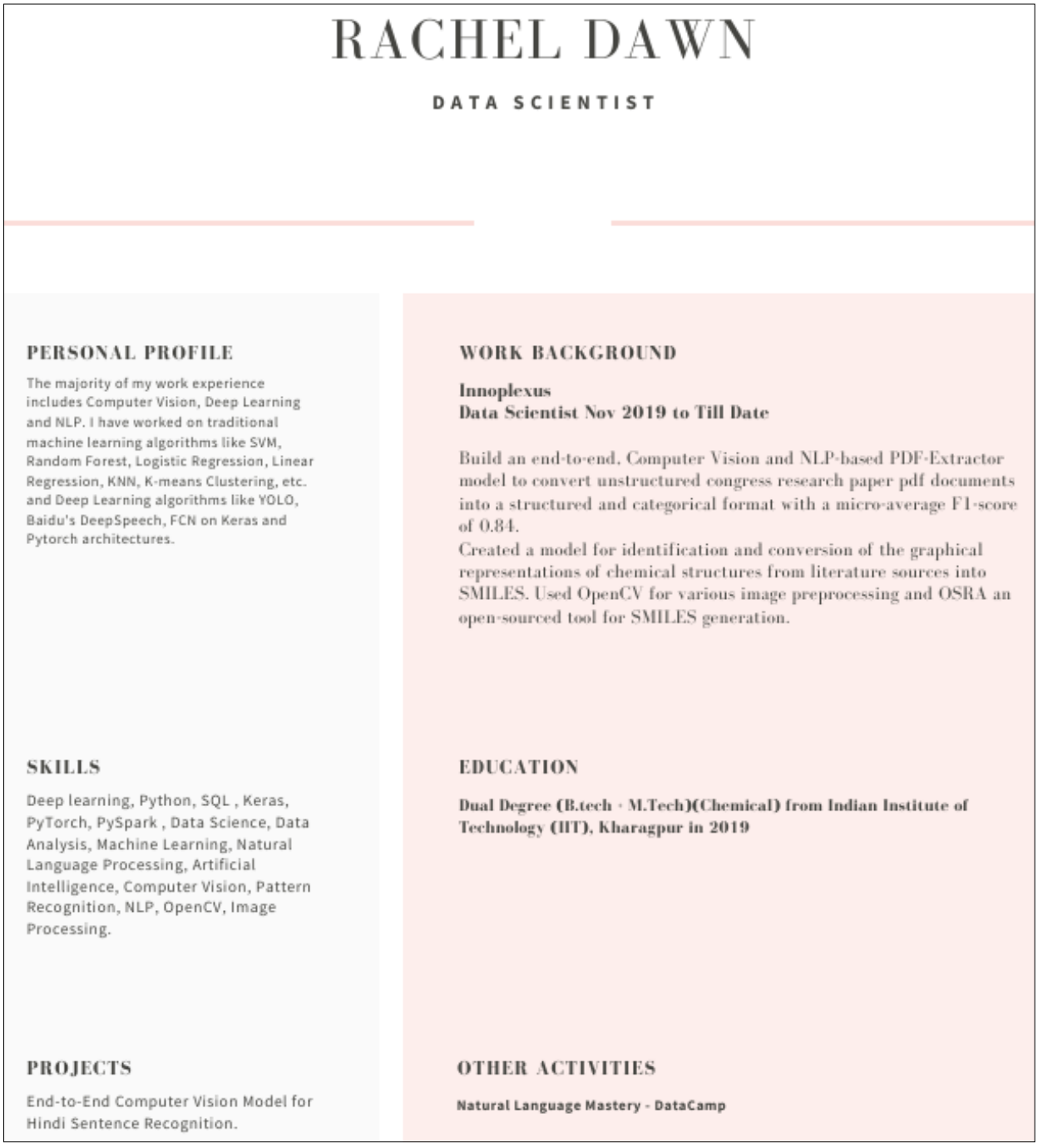
KRUZR
Machine Learning Engineer Intern, Dec 2019 - August 2020

Working for ML team at Kruzr on building driver safety platform. Experience in working with IoT data from smartphone sensors and productionizing ML models at scale.

Education

Integrated B.E. and M.Sc Hons. (Hons. Mechanical Engineering) from Birla Institute of Technology and Science (BITS), Pilani in 2020

Рисунок 4.2 – текст резюме Candidate_099



RACHEL DAWN

DATA SCIENTIST

PERSONAL PROFILE

The majority of my work experience includes Computer Vision, Deep Learning and NLP. I have worked on traditional machine learning algorithms like SVM, Random Forest, Logistic Regression, Linear Regression, KNN, K-means Clustering, etc. and Deep Learning algorithms like YOLO, Baidu's DeepSpeech, FCN on Keras and Pytorch architectures.

SKILLS

Deep learning, Python, SQL , Keras, PyTorch, PySpark , Data Science, Data Analysis, Machine Learning, Natural Language Processing, Artificial Intelligence, Computer Vision, Pattern Recognition, NLP, OpenCV, Image Processing.

PROJECTS

End-to-End Computer Vision Model for Hindi Sentence Recognition.

WORK BACKGROUND

Innoplexus
Data Scientist Nov 2019 to Till Date

Build an end-to-end, Computer Vision and NLP-based PDF-Extractor model to convert unstructured congress research paper pdf documents into a structured and categorical format with a micro-average F1-score of 0.84.
Created a model for identification and conversion of the graphical representations of chemical structures from literature sources into SMILES. Used OpenCV for various image preprocessing and OSRA an open-sourced tool for SMILES generation.

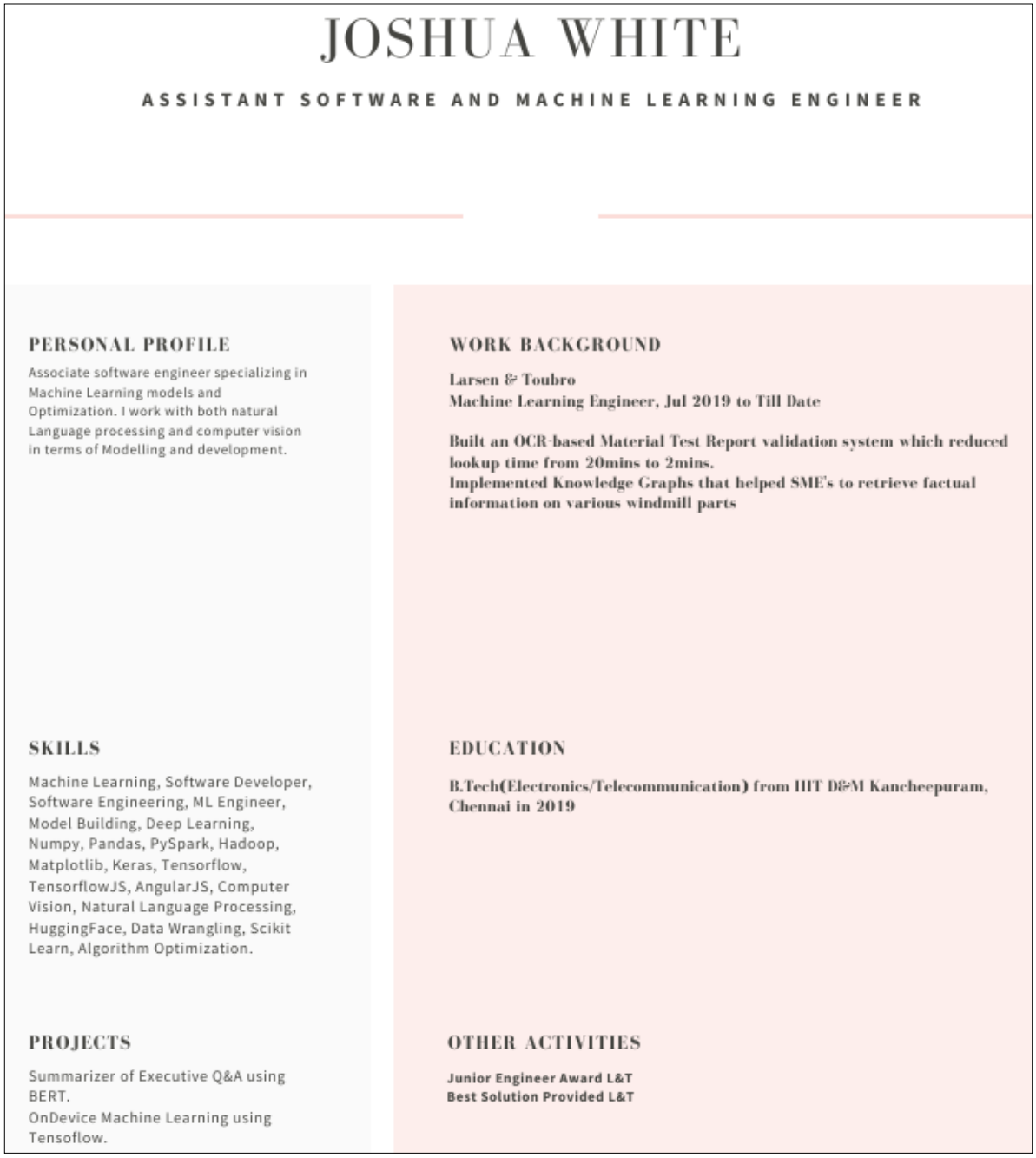
EDUCATION

Dual Degree (B.tech · M.Tech)(Chemical) from Indian Institute of Technology (IIT), Kharagpur in 2019

OTHER ACTIVITIES

Natural Language Mastery - DataCamp

Рисунок 4.3 – текст резюме Candidate_069



JOSHUA WHITE

ASSISTANT SOFTWARE AND MACHINE LEARNING ENGINEER

PERSONAL PROFILE

Associate software engineer specializing in Machine Learning models and Optimization. I work with both natural Language processing and computer vision in terms of Modelling and development.

SKILLS

Machine Learning, Software Developer, Software Engineering, ML Engineer, Model Building, Deep Learning, Numpy, Pandas, PySpark, Hadoop, Matplotlib, Keras, Tensorflow, TensorflowJS, AngularJS, Computer Vision, Natural Language Processing, HuggingFace, Data Wrangling, Scikit Learn, Algorithm Optimization.

PROJECTS

Summarizer of Executive Q&A using BERT.
OnDevice Machine Learning using Tensoflow.

WORK BACKGROUND

Larsen & Toubro
Machine Learning Engineer, Jul 2019 to Till Date

Built an OCR-based Material Test Report validation system which reduced lookup time from 20mins to 2mins.
Implemented Knowledge Graphs that helped SME's to retrieve factual information on various windmill parts

EDUCATION

B.Tech(Electronics/Telecommunication) from HIT D&M Kancheepuram, Chennai in 2019

OTHER ACTIVITIES

Junior Engineer Award L&T
Best Solution Provided L&T

Рисунок 4.4 – текст резюме Candidate_019

ANDORA MCCONNEL

DATA SCIENTIST

EXECUTIVE SUMMARY

Skilled in creating data regression models using predictive data modeling, and data mining algorithms to deliver insights and implemented action-oriented solutions to complex business problems by using Classification Algorithms and Regression Algorithms, Natural Language Processing (NLP), Object Recognition in Python.

PERSONAL SKILLS

Data Science, Data Analysis, Machine Learning, Deep Learning, Statistical Analysis, Data Mining, Computer Vision, Algorithm Development, Linear Regression, Market Basket Analysis, Python, R, SQL, NoSQL.

PROJECTS

OCR Model From Scratch.

EXTRA-CURRICULARS

Data Analyst Certification.

WORK EXPERIENCE

HP
Lead Data Scientist, Feb 2020 to till date

Predicting anomalies in the frequency of transactions on real-time data. Used python and prophet for the same.

Adnet Global
Data Scientist, Jan 2019 to Dec 2019

Built face recognition for identifying various Hollywood celebrities for a UK-based fashion magazine. Used Python, OpenCV, dlib. The model could identify 90 % of the celebrities correctly.

ACADEMIC PROFILE

B.Com (Commerce) from Delhi College of Art in 2017
PG Diploma (Data Science and Business Analytics) from Aegis School of Business & Data Science in 2019

Рисунок 4.5 – текст резюме Candidate_107

Рисунок 4.6 – текст резюме Candidate_017

SOPHIA FLORES

MACHINE LEARNING DEVELOPER(NLP)

EXECUTIVE SUMMARY

Recently graduated fresher experienced in Machine Learning and Natural Language Processing looking for roles that involve projects in the same.

PERSONAL SKILLS

Python Programming, Machine Learning, Natural language Processing, Numpy, NLTK, Text Analysis, Transformers, Series Analysis, Conversational AI.

PROJECTS

Movie Recommendation System
Natural Language Processing
Loan Approval System

EXTRA-CURRICULARS

Machine Learning - Py-Thonist
Big Data Analytics

WORK EXPERIENCE

eLevaTe LtD.
Machine Learning Developer(NLP), Apr 2019 - till date

Modelled an analytics-based recommender engine that used user text along with search and suggested the best possible products that they might be interested in.

ACADEMIC PROFILE

B.Tech(Computers) from KCCE University in 2019

Рисунок 4.7 – текст резюме Candidate_051

KIMBERLY DAVIS

DATA ANALYST AND ENGINEER

SKILLS

Data Analytics, Natural Language Processing, Production Engineering, Production Analytics, Tableau, Machine Learning, Python, NLTK.

PROJECTS

Price Forecasting using Time Series Analysis
Major Crapes exporting Dealer Classification using Logistic Regression

PROFILE

An avid explorer of analytics in multiple facets pertaining to production industry and have applied various Machine Learning Algorithms. I am also very well acquainted with Natural Language Processing using Python.

EXPERIENCE

MAHINDRA LTD

ASSISTANT ANALYST, 2019 - TILL DATE
Analyzed various production components' data and built models that are focused on production equipments and their refinement.

EDUCATION

B.TECH(ELECTRICAL AND ELECTRONICS) FROM SMV INSTITUTE OF TECHNOLOGY, 2019

ACTIVITIES AND AWARDS

GOOGLE ANALYTICS FOR BEGINNERS

Рисунок 4.8 – текст резюме Candidate_031

Результати

Результати ранжування двох методів векторизації тексту частково збігаються. Проте, різниця у вибраних кандидатах пояснюється принципово різними підходами до формування векторних представлень тексту.

Метод TF-IDF базується на статистичній важливості термінів у документі, тому надає перевагу резюме з високою частотою ключових слів.

Word2Vec, навпаки, відображає семантичну близькість слів, що дозволяє моделі враховувати контекст, синонімічні відповідності та загальну змістовність тексту, навіть якщо ключові слова зустрічаються рідше.

Порівняльний аналіз демонструє, що TF-IDF є ефективним у задачах пошуку прямих збігів термінів, тоді як Word2Vec забезпечує більш глибоке семантичне розуміння тексту, що може бути корисним для HR-процесів, де важливо враховувати не лише ключові слова, а й змістовний контекст резюме.

4.3 Інтерпретація результатів

Отримані результати показують, що списки найбільш релевантних кандидатів, сформовані методами TF-IDF та Word2Vec, різняться, що пояснюється принциповою відмінністю у способах представлення текстових даних.

Метод TF-IDF базується на статистичній важливості термінів у документі, тому надає перевагу резюме, у яких найчастіше зустрічаються ключові слова з опису вакансії. Таким чином, цей підхід добре працює у випадках, коли задачі підбору вимагають точного збігу термінології, але може ігнорувати змістовно релевантні резюме, де використані синоніми або інші мовні конструкції.

На відміну від TF-IDF, метод Word2Vec у поєднанні з моделлю CatBoostRegressor враховує семантичну подібність між словами. Це дозволяє моделі виявляти більш глибокі контекстні зв’язки між досвідом кандидата та вимогами вакансії, навіть якщо відповідні терміни не збігаються дослівно. Саме тому Word2Vec може показувати вищі оцінки для резюме з релевантним досвідом, але меншою кількістю ключових слів.

Порівняння топ-кандидатів за двома методами показує, що TF-IDF акцентує увагу на точних текстових збігах, а Word2Vec – на загальному

змістовному наповненні. Це робить Word2Vec більш гнучким у випадках реальних HR-процесів, де важливо враховувати не лише наявність ключових слів, а й релевантність досвіду в широкому контексті.

Таким чином, результати демонструють, що кожен метод має свої переваги: TF-IDF показує хороший результат у задачах суворого пошуку за ключовими словами, Word2Vec – у задачах семантичного аналізу та комплексної оцінки резюме.

5 АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ РОЗРОБКИ

5.1 Розрахунок собівартості розробки програмного забезпечення

Впровадження модуля текстового аналізу в систему IT-рекрутингу є стратегічною інвестицією у напрямку автоматизації та підвищення ефективності процесів відбору персоналу. Сучасний рекрутинг дедалі більше ґрунтується на обробці великих обсягів неструктурованих текстових даних – резюме, мотиваційних листів, описів вакансій тощо. Застосування методів аналізу тексту за допомогою штучного інтелекту дозволяє автоматично виділяти ключові навички, визначати рівень досвіду кандидата та оцінювати відповідність вимогам вакансії.

Нижче наведено економічний аналіз основних компонентів, необхідних для впровадження такої системи: розробка інтерфейсу та серверної частини, отримання та підготовка даних, створення та навчання моделі, а також витрати на навчання персоналу та технічну підтримку.

Інтерфейс системи забезпечує зручну взаємодію HR-спеціалістів із аналітичним модулем – завантаження документів кандидатів, перегляд результатів аналізу та візуалізацію знайдених навичок. Якість інтерфейсу безпосередньо впливає на ефективність роботи користувачів, оскільки більшість майбутніх користувачів модуля не мають технічної підготовки в галузі аналізу даних (Таблиця 5.1 – Вартість розробки інтерфейсу користувача).

Таблиця 5.1 – Вартість розробки інтерфейсу користувача

Компонент	Опис	Орієнтовна трудомісткість	Приблизна вартість, $
Проектування макетів	Розробка макетів та логіки взаємодії	40 год	1 000

Компонент	Опис	Орієнтовна трудомісткість	Приблизна вартість, $
Реалізація інтерфейсу	Розробка на React, інтеграція з API	120 год	3 000
Тестування та вдосконалення	Перевірка функціоналу та юзабіліті	20 год	500
Разом		**180 год**	**4 500**

Серверна частина відповідає за обробку запитів між користувачем і аналітичною моделлю. Вона приймає тексти для аналізу, взаємодіє з базою даних та надає результати через API. Крім того, вона має забезпечувати відповідність вимогам конфіденційності, адже у процесі обробляються персональні дані кандидатів. Також, необхідно врахувати подальшу інтеграцію кандидата в робочий процес. Тому, цей модуль мусить бути інтегрований з внутрішньою HR-системою, що додає додаткового рівня складності отриманої системи (Таблиця 5.2 – Вартість розробки серверної частини системи).

Таблиця 5.2 – Вартість розробки серверної частини системи

Компонент	Опис	Орієнтовна трудомісткість	Приблизна вартість, $
Проектування API	Визначення форматів запитів і відповідей	20 год	500
Реалізація логіки	Розробка на одному з популярних фреймворків	80 год	2 000
Інтеграція з базою	Додавання структур	20 год	500

Компонент	Опис	Орієнтовна трудомісткість	Приблизна вартість, $
даних	для збереження результатів		
Безпека та логування	Налаштування автентифікації та контролю доступу	20 год	500
Тестування та оптимізація	Перевірка стабільності роботи	20 год	500
Разом		**160 год**	**4 000**

Якість моделі напряму залежить від релевантності та різноманітності даних. У процессі навчання потрібна велика кількість резюме, описів вакансій та позначених прикладів відповідності навичок. Хоча існують відкриті набори даних, більшість із них потребують очищення, стандартизації або додаткової розмітки.

Для позначення даних що будуть використані у навчанні моделі, можна використати існуючих HR-спеціалістів та наявну базу резюме компанії. Однак, цього може виявитись недостатньо і тоді доведеться використати платні джерела даних (Таблиця 5.3 – Вартість даних для навчання моделі).

Таблиця 5.3 – Вартість даних для навчання моделі

Компонент	Опис	Приблизна вартість, $
Відкриті набори даних	Безкоштовні, але потребують адаптації	0
Комерційні набори вакансій	Одноразова плата за доступ або ліцензію	800 – 1500
Розмітка та підготовка	Ручна розмітка 5 000 – 10 000	1500 – 2500

Компонент	Опис	Приблизна вартість, $
даних	прикладів	
Разом		**2300 – 4000**

На етапі навчання моделі залучається фахівець із даних, який займається побудовою, навчанням і оптимізацією моделі. Залежно від обсягу проекту, це може бути як проста модель на основі TF-IDF, так і сучасна трансформерна архітектура, наприклад, BERT, RoBERTa, DistilBERT тощо (Таблиця 5.4 – Вартість навчання моделі).

Таблиця 5.4 – Вартість навчання моделі

Компонент	Опис	Орієнтовна трудомісткість	Приблизна вартість, $
Дослідження та прототипування	Вибір архітектури та створення базової моделі	80 год	2 800
Попередня обробка даних	Очищення та токенізація текстів	40 год	1 400
Навчання та оптимізація	Тренування, тестування, порівняння моделей	80 год	2 800
Інтеграція з системою	Співпраця з командою розробки серверної частини	20 год	700
Разом		**220 год**	**7 700**

Впровадження нового аналітичного модуля вимагає підготовки користувачів, які мають навчитися інтерпретувати результати аналізу,

розуміти рейтингові оцінки та правильно застосовувати їх у роботі з кандидатами (Таблиця 5.5 – Вартість навчання персоналу).

Таблиця 5.5 – Вартість навчання персоналу

Компонент	Опис	Приблизна вартість, $
Навчання персоналу	Проведення внутрішніх тренінгів (8 – 10 годин)	300
Хмарна інфраструктура	GPU-обчислення для тренування та розгортання	700 – 1200
Разом		**1000 – 1500**

Загальна оцінка витрат може варіюватись від багатьох факторів, таких як: терміновість виконання проектів, форс-мажорні обставини, прийняття конкретних рішень, рівень залученості в проект компанії, покупка готового рішення чи самостійна розробка, тощо.

Підсумки наведено у Таблиця 5.6 – Загальна вартість розробки системи.

Таблиця 5.6 – Загальна вартість розробки системи

Категорія	Орієнтовна вартість, $
Розробка інтерфейсу	4 500
Розробка серверної частини	4 000
Отримання та підготовка даних	2 300 – 4 000
Розробка та навчання моделі	7 700
Навчання персоналу та експлуатація	1 000 – 1 500
Загалом	**19 500 – 21 700**

Сумарна вартість впровадження модуля текстового аналізу в систему IT-рекрутингу становить близько $20 000, що є економічно доцільним в

контексті європейського або американського IT-ринку. Для української компанії це може бути важкопідйомна сума, проте враховуючи надлишок пропозиції молодших IT-спеціалістів це все ще може бути стратегічно вигідним рішенням.

5.2 Оцінка економічної вигоди від впровадження

Основною метою впровадження модуля текстового аналізу є підвищення ефективності рекрутингових процесів. У традиційній моделі добору IT-фахівців HR-спеціалісти витрачають значну частину робочого часу на перегляд резюме, визначення ключових навичок і попередню оцінку кандидатів. Автоматизація цього етапу дозволяє суттєво знизити витрати часу на одного кандидата та підвищити пропускну здатність відділу HR-спеціалістів без збільшення штату. Це особливо важливо в контексті українського оподаткування заробітних плат для роботодавців що робить зменшення обсягів штату ще більш вигідним ніж в класичних економічних моделях які були розроблені під американське або європейське законодавство.

Для оцінки ефекту було розглянуто типовий середній рекрутинговий процес у компанії, що займається пошуком IT-фахівців середнього рівня. В середньому один HR-спеціаліст обробляє близько 150 резюме на тиждень, витрачаючи приблизно 10 хвилин на одне резюме.

Після впровадження модуля, що здійснює автоматичне витягування навичок і попередню оцінку відповідності вакансії, час на одне резюме скорочується до 5-6 хвилин, що в свою чергу підвищує ефективність роботи спеціаліста.

При відборі кандидатів класичною є ситуація коли спочатку HR-спеціаліст проводить скрінінг кандидата на окремій співбесіді, а вже потім відбувається технічна співбесіда на якій і буде перевірено навички. Використання модуля підвищує якість відбору спеціалістів і допомагає зменшити кількість як і скрінінгів, так і технічних співбесід що має

прихований економічний ефект за рахунок того, що цінний спеціаліст який проводить технічні співбесіди менше відволікається від основної роботи на проведення співбесід. Також, зменшуються витрати на онбординг тих кандидатів, які пізніше не пройдуть випробувальний термін.

Окрім того, підвищується якість відбору кандидатів, що має непрямий економічний ефект, зменшуючи необхідність повторних наймів на ту ж позицію або повторне розширення штату. Так само можна сказати і про вірогідність відхилення кандидатів з гарними навичками, що також зменшує подальші витрати.

Сукупно, ці поліпшення роботи HR-відділу будуть означати підвищення його ефективності на 55-60% протягом першого року використання системи.

Для оцінки матеріального економічного ефекту зробимо опору на дані. Якщо взяти середній розмір HR-відділу в не досить великій компанії, то це буде приблизно 5-6 спеціалістів, одного з яких можна буде або скоротити, перевести на іншу посаду або відкласти розширення відділу. У такому випадку, будемо мати економію на заробітній платні такого спеціаліста, тобто близько $1200 – 1500, або $14 400 – 18 000 на рік.

Усі непрямі економічні ефекти не мають конкретної суми в яку їх можна оцінити, проте орієнтовно вони можуть складати від $2 000 до $10 000, візьмемо в середньому $6 000, що доводить загальну економічну вигоду до $24 000.

Ці цифри та вартість розробки й інтеграції дають нам можливість розрахувати окупність (ROI) та термін окупності такого нововведення (формули (5.1(5.2).

$$ROI = \frac{\text{Чистий прибуток}}{\text{Інвестиції}} \cdot 100\% = \frac{18000 + 6000 - 20000}{20000} \cdot 100\% = 25\% \quad (5.1)$$

$$\text{Термін окупності} = \frac{\text{Інвестиції}}{\text{Щорічна економія}} = \frac{20000}{24000} \approx 0.83 \quad (5.2)$$

Отже, система окупається приблизно за 10 місяців, після чого починає приносити чистий прибуток. Проте варто зазначити, що розрахунки велись для досить великого штату HR-спеціалістів й окупність та її терміни напряму залежать від його розміру, тому рішення не є універсальним, незважаючи на неймовірно гарні показники в розрахунках.

5.3 Аналіз ризиків впровадження

Впровадження інтелектуальної системи аналізу тексту у сферу IT-рекрутингу, попри очевидні переваги, супроводжується низкою ризиків технічного, економічного та організаційного характеру. Їх розуміння є необхідним для оцінки повної вартості проекту та розробки стратегії управління можливими проблемами.

Основним технічним ризиком є неточність самої моделі. Якщо навчальні дані будуть недостатніми або нерепрезентативними, система може некоректно оцінювати досвід чи навички кандидатів, що призведе до хибних результатів відбору. Такі похибки знижують довіру користувачів і потребують додаткового донавчання, що збільшує витрати на підтримку системи.

Також, можливі проблеми інтеграції нового модуля з існуючими HRM- або CRM-системами компанії. Неповна сумісність API або відмінності у форматах даних можуть призвести до затримок на етапі впровадження та збільшення вартості проекту приблизно на 10%.

Ще одним технічним ризиком є залежність від сторонніх бібліотек і сервісів штучного інтелекту. У разі зміни політики доступу або вартості використання таких платформ (наприклад, OpenAI або Azure AI) компанія

може зазнати додаткових витрат на перенесення рішення на іншу технологічну основу.

Економічні ризики полягають передусім у можливому перевищенні бюджету проекту. Це може статися через недооцінку складності моделі, обсягів необхідних даних або часу на розробку. Перевищення на 10-20 % бюджету є типовим для подібних проектів.

Ще один ризик – недостатня окупність інвестицій. Якщо очікуваний економічний ефект, наприклад скорочення часу на пошук кандидатів або зменшення навантаження на HR-спеціалістів, виявиться меншим, ніж прогнозувалося, строк окупності може збільшитися до трьох чи навіть чотирьох років. Важливо також враховувати витрати на підтримку системи після запуску – оновлення моделей, адаптацію до змін ринку праці та регулярне тестування якості роботи. Зазвичай ці витрати становлять 5-10 % початкового бюджету щороку.

До організаційних ризиків належать людський фактор і правові аспекти. Часто персонал може з настороженістю ставитися до автоматизації, побоюючись, що система замінить частину їхніх функцій. Це може призвести до опору використанню нових інструментів або свідомого ігнорування рекомендацій системи. Щоб цього уникнути, необхідно проводити навчання та демонструвати переваги інструменту як допоміжного засобу, а не заміни спеціаліста.

Ще один ризик полягає у нестачі фахівців здатних провести таку роботу з моделлю, що може змусити компанію залучати зовнішніх консультантів і збільшити витрати. Важливим є й питання конфіденційності: система працює з персональними даними кандидатів, тому вона має повністю відповідати вимогам національного законодавства. Порушення цих норм може спричинити не лише фінансові штрафи, але й репутаційні втрати для компанії.

Загалом, основними ризиками є технічна невизначеність результатів моделі, перевищення бюджету, затримка з окупністю інвестицій та можливі

проблеми з дотриманням вимог захисту даних. Зменшити їх вплив можна шляхом поетапного впровадження системи, проведення пілотного тестування, контролю якості роботи моделі та навчання персоналу. Також рекомендується створити резерв фінансування у розмірі близько 10-15 % бюджету для покриття непередбачених витрат і залучити юридичних консультантів для перевірки відповідності вимогам національного законодавства.

5.4 Висновки щодо економічної доцільності впровадження

Проведений аналіз витрат, економічних вигод і ризиків свідчить, що впровадження модуля аналізу тексту в систему IT-рекрутингу є економічно доцільним за умови належного планування й управління ризиками. Загальні початкові інвестиції на реалізацію проекту оцінювалися приблизно в $19 500 – $21 700 (у середньому ≈ $20 000). Цей бюджет охоплює розробку інтерфейсу, серверної частини, отримання та підготовку даних, розробку й навчання моделі та початкові витрати на навчання персоналу й експлуатацію.

Основним джерелом економічної вигоди є скорочення робочого часу HR-спеціалістів за рахунок автоматизації попереднього відбору резюме. У розрахунках було прийнято, що автоматизація дозволяє зменшити навантаження на HR-відділ таким чином, що місячна економія на заробітних платах складає до $1500 ($18 000 на рік). До цього додаються інші вигоди (скорочення повторних наймів, зменшення часу закриття вакансій тощо), які в сукупності оцінені орієнтовно ще в $6 000 на рік, тобто сумарна очікувана річна вигода була прийнята як $24 000.

На основі цих оцінок розрахунок ROI та терміну окупності має такий вигляд:

- Чиста річна вигода (приблизно): $24 000.
- Інвестиції: ≈ $20 000.
- ROI = (24 000 – 20 000) / 20 000 × 100% = 25%.

- Термін окупності = 20 000 / 24 000 ≈ 0,83 року (менше 11 місяців).

Отже, при збереженні допущень, використаних у розрахунках попередніх розділів, система окупається швидше одного року і дає позитивний економічний ефект вже в перший рік експлуатації.

Водночас результати чутливі до кількох ключових факторів, визначених у розділі ризиків. Насамперед це якість і репрезентативність навчальних даних, точність моделі, можливі технічні ускладнення при інтеграції, а також питання дотримання вимог захисту персональних даних. Вплив цих ризиків може зменшити або відтермінувати досягнення очікуваної економії; у гіршому випадку це призведе до збільшення витрат на донавчання моделі або на подолання технічних проблем.

Підсумовуючи, за збереження вихідних припущень і при впровадженні рекомендованих заходів з управління ризиками, інвестиція в модуль аналізу тексту має позитивне економічне підґрунтя: вона швидко починає приносити прибуток, створює стабільну річну економію та додає довгострокову аналітичну вартість для процесів рекрутингу.

ВИСНОВКИ

У магістерській роботі було досліджено ефективність застосування методів векторизації тексту – TF-IDF та Word2Vec – у задачі оцінювання відповідності кандидата вакансії у IT-сфері.

У теоретичній частині було розглянуто предметну область дослідження, виділено ключові моменти у роботі HR-фахівця, які можуть бути оптимізовані шляхом впровадження розробленого програмного забезпечення, проаналізовано аналоги даної системи, виділено їх основні переваги та недоліки.

У практичній частині розроблено модулі обробки тесту, векторизації, ранжування та генерації звіту. Модуль попередньої обробки тексту включає токенізацію, стемінг, видалення стоп-слів, що забезпечило коректність і об’єктивність подальшого порівняння моделей.

Проведене експериментальне порівняння показало, що обидві моделі дають доволі непоганий результат. Проте, модель Word2Vec дає змогу отримувати вектори, які краще відображають професійний контекст і семантичні зв’язки між термінами, ніж розріджені частотні вектори TF-IDF. Ці результати підтверджують перспективність практичного впровадження даної системи для семантичного аналізу та відбору кандидатів.

Отримані результати можуть стати основою для подальшого розвитку підходів до автоматизованого IT-рекрутингу з використанням машинного навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. TextKernel Parser [Електронний ресурс] / Режим доступу: https://www.textkernel.com/products-solutions/parser/

2. HireAbility Resume Parser [Електронний ресурс] / Режим доступу : https://www.hireability.com/trial-hireability/

3. Eightfold.ai [Електронний ресурс] / Режим доступу : https://eightfold.ai/products/resource-management/

4. Natural Language Processing (NLP) in AI-Driven Recruitment Systems [Електронний ресурс] / Sudheer Devaraju // International Journal of Scientific Research in Computer Science, Engineering and Information Technology. – 2022. – №8(3). – С. 555-566. – Режим доступу: https://figshare.com/articles/journal_contribution/Natural_Language_Processing_NLP_in_AI-Driven_Recruitment_Systems/28095221?file=51382214

5. Challenges and Opportunities of NLP for HR Applications: A Discussion Paper [Електронний ресурс] / Jochen L. Leidner, Mark Stevenson. – 2024. – Режим доступу: https://www.researchgate.net/publication/380791999_Challenges_and_Opportunities_of_NLP_for_HR_Applications_A_Discussion_Paper

6. Comparative analysis of text vectorization methods [Електронний ресурс] / Victor Sineglazov, Illia Savenko // Electronics and Control Systems. – 2023. – Vol. 2 No. 76. – Режим доступу: https://jrnl.nau.edu.ua/index.php/ESU/article/view17663

7. TensorFlow. Tutorials. Word2Vec [Електронний ресурс] / Режим доступу: https://www.tensorflow.org/text/tutorials/word2vec

8. CatBoost. Documentation [Електронний ресурс] / Режим доступу: https://catboost.ai/docs/en/

9. React Documentation [Електронний ресурс] / Режим доступу: https://react.dev/learn

10. FastAPI Documentation [Електронний ресурс] / Режим доступу : https://fastapi.tiangolo.com/

11. PDFPlumber [Електронний ресурс] / Режим доступу: https://www.pdfplumber.com/

12. Python-docx Documentation [Електронний ресурс] / Режим доступу : https://python-docx.readthedocs.io/en/latest/

13. Natural Language Toolkit. Documentation [Електронний ресурс] / Режим доступу: https://www.nltk.org/

14. Scikit-learn Documentation [Електронний ресурс] / Режим доступу: https://scikit-learn.org/stable/supervised_learning.html

15. Numpy Documentation [Електронний ресурс] / Режим доступу: https://numpy.org/doc/

16. Pandas Documentation [Електронний ресурс] / Режим доступу: https://pandas.pydata.org/docs/

17. CatBoost Regressor. Documentation [Електронний ресурс] / Режим доступу: https://catboost.ai/docs/en/concepts/python-reference_catboostregressor

18. TfidfVectorizer Documentation [Електронний ресурс] / Режим доступу: https://surl.lt/dwldee

19. Gensim Documentation [Електронний ресурс] / Режим доступу : https://www.nltk.org/howto/gensim.html

20. Metrics and scoring: quantifying the quality of predictions [Електронний ресурс] / Режим доступу: https://scikit-learn.org/stable/modules/model_evaluation.html

ДОДАТОК А – КОД ЗАСТОСУНКУ

text_processor.py

```
import os
import io
import pdfplumber
import docx
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')


class TextProcessor:
    def __init__(self, language="english"):
        self.language = language
        self.stop_words = set(stopwords.words(language))
        self.stemmer = SnowballStemmer(language)

    def read_pdf(self, file_bytes: bytes):
        text = ""
        with pdfplumber.open(io.BytesIO(file_bytes)) as pdf:
            for page in pdf.pages:
                page_text = page.extract_text()
                if page_text:
                    text += page_text + "\n"
        return text

    def read_docx(self, file_bytes: bytes):
        doc = docx.Document(io.BytesIO(file_bytes))
        return "\n".join([p.text for p in doc.paragraphs])

    def extract_text(self, upload_file):
        file_bytes = upload_file.file.read()

        if upload_file.filename.endswith(".pdf"):
            return self.read_pdf(file_bytes)
        elif upload_file.filename.endswith(".docx"):
            return self.read_docx(file_bytes)
        else:
            raise ValueError("Підтримуються тільки PDF або DOCX
файли.")

    def clean_text(self, text):
        text = text.lower()
        text = re.sub(r'\s+', ' ', text)
```

```
        text = re.sub(r'[^a-zA-Za-яА-Я0-9 ]', '', text)
        return text

    def tokenize(self, text):
        return word_tokenize(text)

    def remove_stopwords(self, tokens):
        return [word for word in tokens if word not in
self.stop_words]

    def stem(self, tokens):
        return [self.stemmer.stem(word) for word in tokens]

    def process_text(self, upload_file):
        raw = self.extract_text(upload_file)
        cleaned = self.clean_text(raw)
        tokens = self.tokenize(cleaned)
        tokens = self.remove_stopwords(tokens)
        tokens = self.stem(tokens)
        return tokens

    def process_directory(self, uploaded_files):
        results = {}
        for f in uploaded_files:
            try:
                tokens = self.process_text(f)
                results[f.filename] = tokens
            except Exception as e:
                print(f"Не вдалося обробити {f.filename}: {e}")
        return results
```

text_vectorizer.py

```
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
import numpy as np


class TextVectorizer:
    def __init__(self):
        self.tfidf_vectorizer = None
        self.word2vec_model = None

    def fit_tfidf(self, tokenized_texts):
        documents = [" ".join(tokens) for tokens in
tokenized_texts]
        self.tfidf_vectorizer = TfidfVectorizer()
        self.tfidf_vectorizer.fit(documents)

    def transform_tfidf(self, tokens):
        document = " ".join(tokens)
```

```
        return
self.tfidf_vectorizer.transform([document]).toarray()[0]

    def vectorize_tfidf_list(self, tokenized_texts):

        documents = [" ".join(tokens) for tokens in
tokenized_texts]
        return
self.tfidf_vectorizer.transform(documents).toarray()

    def train_word2vec(self, tokenized_texts, vector_size=200,
window=5, min_count=1):

        self.word2vec_model = Word2Vec(
            sentences=tokenized_texts,
            vector_size=vector_size,
            window=window,
            min_count=min_count
        )

    def transform_word2vec(self, tokens):

        vectors = []
        for word in tokens:
            if word in self.word2vec_model.wv:
                vectors.append(self.word2vec_model.wv[word])

        if len(vectors) == 0:
            return np.zeros(self.word2vec_model.vector_size)

        return np.mean(vectors, axis=0)

    def vectorize_word2vec_list(self, tokenized_texts):
        return np.array([self.transform_word2vec(tokens) for
tokens in tokenized_texts])
```

candidate_evaluator.py

```
import joblib
import numpy as np

class CandidateEvaluator:
    def __init__(self):
        self.model = joblib.load("trained_model_word2vec.pkl")
        self.scaler = joblib.load("scaler.pkl")
        self.pca = joblib.load("pca.pkl")

    def evaluate(self, resume_vector, job_vector):

        x = np.concatenate((resume_vector,
job_vector)).reshape(1, -1)
```

```
        x_scaled = self.scaler.transform(x)
        x_reduced = self.pca.transform(x_scaled)

        predicted_score = self.model.predict(x_reduced)[0]
        return round(predicted_score, 2)
```

ranker.py

```
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

class CandidateRanker:
    def __init__(self, mode="tfidf"):
        self.mode = mode

    def ranking_scores(self, vacancy_vector,
resume_vectors_dict):

        scores = {}

        if self.mode == "tfidf":
            v = vacancy_vector.reshape(1, -1)

            for candidate_id, r in resume_vectors_dict.items():
                r_vec = r.reshape(1, -1)
                score = cosine_similarity(v, r_vec)[0][0]
                scores[candidate_id] = score

        elif self.mode == "word2vec":

            evaluator = CandidateEvaluator()

            for candidate_id, resume_vec in
resume_vectors_dict.items():
                score = evaluator.evaluate(resume_vec,
vacancy_vector)
                scores[candidate_id] = score

        return scores

    def rank_candidates(self, vacancy_vector,
resume_vectors_dict):

        scores = self.ranking_scores(vacancy_vector,
resume_vectors_dict)
        ranking = sorted(scores.items(), key=lambda x: x[1],
reverse=True)

        return [(candidate_id, score, rank + 1)
                for rank, (candidate_id, score) in
enumerate(ranking)]
```

report_generator.py

```
import pandas as pd
from docx import Document

class ReportGenerator:
    def __init__(self):
        pass

    def create_dataframe(self, ranking):
        data = []
        for name, score, _ in ranking:
            percent = round(score * 100, 2)
            data.append([name, percent])

        df = pd.DataFrame(data, columns=["Кандидат", "Відсоток
відповідності (%)"])
        return df

    def save_to_excel(self, ranking, output_path="report.xlsx"):
        df = self.create_dataframe(ranking)
        df.to_excel(output_path, index=False)
```