

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
(повна назва)
Кафедра _____ Штучного інтелекту _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський) _____

_____ Вебсервіс інтелектуального пошуку схожих вакансій _____
_____ (тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-3 _____

_____ Михайло КАРПЕНКО _____
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки _____
_____ (код і повна назва спеціальності)

Тип програми _____ освітньо-професійна _____
Освітня програма _____ Штучний інтелект _____
_____ (повна назва освітньої програми)

Керівник _____ ас. Максим ПОЛІТ _____
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА _____
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«_____» _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Карпенка Михайла Юрійовича _____
(прізвище, ім'я, по батькові)

1. Тема роботи Вебсервіс інтелектуального пошуку схожих вакансій

затверджена наказом університету від 8 червня 2026 р. № 488Ст

2. Термін подання студентом роботи до екзаменаційної комісії 9 липня 2026 р.

3. Вихідні дані до роботи відкритий набір даних LinkedIn Job Postings (Kaggle); мова програмування Python; бібліотека sentence-transformers; фреймворки FastAPI та React; мова C#; платформа ASP.NET Core; СУБД PostgreSQL з розширенням pgvector; хмарний сервіс Supabase; середовище розробки Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі _____

2) Методи та технології інтелектуального пошуку схожих вакансій _____

3) Програмна реалізація системи _____

4) Аналіз результатів роботи системи _____

КАЛЕНДАРНИЙ ПЛАН

[illegible]

Дата видачі завдання 8 червня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____
(підпис)

ас. Максим ПОЛІЇТ
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 67 с., 13 рис., 5 табл., 1 дод., 21 джерело.

ВЕКТОРНЕ ПРЕДСТАВЛЕННЯ, ЕМБЕДІНГИ, ПОШУК ВАКАНСІЙ, СЕМАНТИЧНИЙ ПОШУК, API, HNSW, NLP, PGVECTOR, REST, SBERT, SPA, SQL.

Об'єкт дослідження – процес інтелектуального пошуку семантично схожих вакансій у великих корпусах текстових даних.

Предмет дослідження – методи векторного представлення текстів та алгоритми визначення семантичної подібності, застосовані до задачі пошуку вакансій.

Мета роботи – проєктування та реалізація вебсервісу інтелектуального пошуку семантично схожих вакансій на основі моделі SBERT та векторної бази даних з розширенням pgvector.

Методи дослідження – обробка природної мови, векторне представлення текстів, косинусна подібність, наближений пошук найближчих сусідів, порівняльний аналіз методів.

Розроблено триланкову клієнт-серверну систему на FastAPI, ASP.NET Core і React. Реалізовано конвеєр підготовки та векторизації корпусу з 124 230 вакансій на основі моделі SBERT all-MiniLM-L6-v2. Векторні представлення розмірністю 384 збережено у PostgreSQL з розширенням pgvector та індексом HNSW, що забезпечує логарифмічну складність пошуку. Проведено порівняння семантичного підходу з класичним TF-IDF, яке підтвердило якісну перевагу SBERT у знаходженні релевантних вакансій за синонімічними та короткими запитами. Результати тестування на реальному корпусі даних підтвердили виконання усіх поставлених вимог до системи.

ABSTRACT

Bachelor's thesis contains: 67 pp., 13 fig., 5 tabl., 1 ann., 21 references.

EMBEDDINGS, HNSW, NLP, PGVECTOR, REST, SBERT, SEMANTIC SEARCH, SPA, SQL, VACANCY SEARCH, VECTOR REPRESENTATION.

Object of research – the process of intelligent search for semantically similar vacancies in large text data corpora.

Subject of research – methods of text vector representation and semantic similarity algorithms applied to the vacancy search problem.

Purpose of the work – design and implementation of a web service for intelligent search of semantically similar vacancies based on the SBERT model and a vector database with the pgvector extension.

Research methods – natural language processing, text vector representation, cosine similarity, approximate nearest neighbor search, comparative analysis of methods.

A three-tier client-server system was developed using FastAPI, ASP.NET Core and React. A pipeline for preparation and vectorization of a corpus of 124,230 vacancies was implemented based on the SBERT all-MiniLM-L6-v2 model. Vector representations of dimensionality 384 were stored in PostgreSQL with the pgvector extension and an HNSW index, ensuring logarithmic search complexity. A comparison of the semantic approach with classical TF-IDF confirmed the qualitative superiority of SBERT in finding relevant vacancies for synonymous and short queries. Testing results on a real data corpus confirmed that all system requirements were met.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	8
Вступ.....	9
1 Аналіз предметної галузі	11
1.1 Цифровий ринок вакансій	11
1.2 Проблема релевантності пошуку вакансій	13
1.3 Огляд існуючих платформ пошуку роботи	15
1.4 Обмеження класичних підходів до рекомендації вакансій	18
1.5 Постановка задачі інтелектуального пошуку схожих вакансій	20
2 Методи та технології інтелектуального пошуку схожих вакансій	23
2.1 Текстові вакансії як об'єкт інтелектуального аналізу.....	23
2.1.1 Основні компоненти опису вакансії	23
2.1.2 Лінгвістичні особливості текстів вакансій.....	25
2.1.3 Проблеми неструктурованості та шуму даних у вакансіях і способи їх усунення.....	27
2.2 Подання вакансій і методи визначення подібності	28
2.2.1 Формальні моделі подання тексту вакансій.....	28
2.2.2 Семантичні моделі подання вакансій	30
2.2.3 Методи обчислення подібності та ранжування	32
2.3 Архітектура та принципи функціонування вебсервісу	34
2.3.1 Загальна архітектура системи.....	34
2.3.2 Алгоритм інтелектуального пошуку вакансій	36
3 Програмна реалізація системи	38
3.1 Підготовка та очищення даних	38
3.2 Векторизація вакансій на основі SBERT	40
3.3 Зберігання векторних представлень у PostgreSQL з розширенням pgvector	42
3.4 Реалізація FastAPI мікросервісу семантичного пошуку	45
3.5 Реалізація серверної частини на ASP.NET Core	47

3.6 Реалізація клієнтської частини на React	50
4 Аналіз результатів роботи системи	52
4.1 Опис тестового середовища та датасету	52
4.2 Оцінка якості семантичного пошуку	54
4.3 Порівняння семантичного підходу з TF-IDF	58
4.4 Аналіз продуктивності системи	59
4.5 Оцінка відповідності системи вимогам технічного завдання	60
Висновки	62
Перелік джерел посилання	64
Додаток А Відомість кваліфікаційної роботи	67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

СУБД – система управління базами даних;

API – Application Programming Interface – інтерфейс програмування застосунків;

CPU – Central Processing Unit – центральний процесор;

CSV – Comma-Separated Values – формат даних з роздільниками комами;

DI – Dependency Injection – впровадження залежностей;

DTO – Data Transfer Object – об’єкт передачі даних;

HNSW – Hierarchical Navigable Small World – ієрархічний граф для наближеного пошуку найближчих сусідів;

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту;

JSON – JavaScript Object Notation – формат обміну даними;

ML – Machine Learning – машинне навчання;

NLP – Natural Language Processing – обробка природної мови;

REST – Representational State Transfer – архітектурний стиль побудови API;

SBERT – Sentence Bidirectional Encoder Representations from Transformers – модель векторного представлення речень;

SPA – Single Page Application – односторінковий застосунок;

SQL – Structured Query Language – мова структурованих запитів;

TF-IDF – Term Frequency – Inverse Document Frequency – статистична міра важливості терміна у документі;

URL – Uniform Resource Locator – уніфікований локатор ресурсів.

ВСТУП

Розвиток цифрового ринку праці супроводжується стрімким зростанням обсягів текстових даних про вакансії, що публікуються на онлайн-платформах працевлаштування. Традиційні методи пошуку, засновані на ключових словах та формальних фільтрах, не забезпечують достатнього рівня релевантності результатів – вони не здатні розпізнавати семантичну близькість між вакансіями, сформульованими різними словами, але схожими за професійним змістом.

Актуальність роботи обумовлена невідповідністю між потребами користувачів сучасних платформ пошуку роботи та можливостями наявних пошукових рішень. Переважна більшість існуючих сервісів орієнтована на точне співпадіння термінів, що призводить до пропуску релевантних пропозицій і перевантаження нерелевантними результатами. Особливо гостро ця проблема проявляється у сфері інформаційних технологій, де одна й та сама функціональна роль може мати десятки різних найменувань.

Метою кваліфікаційної роботи є розробка вебсервісу інтелектуального пошуку семантично схожих вакансій на основі методів обробки природної мови та векторного представлення текстів.

Для досягнення поставленої мети вирішено такі задачі:

- аналіз цифрового ринку вакансій та існуючих підходів до пошуку і рекомендації вакансій;
- дослідження методів семантичного представлення текстів та обчислення подібності;
- розробка конвеєру підготовки та векторизації текстових даних вакансій;
- реалізація мікросервісу семантичного пошуку на основі моделі SBERT та векторної бази даних з розширенням pgvector;
- розробка серверної частини на ASP.NET Core та клієнтського інтерфейсу на React;

– тестування системи та аналіз якості й продуктивності семантичного пошуку.

Об’єктом дослідження є процес інтелектуального пошуку семантично схожих вакансій у великих корпусах текстових даних.

Предметом дослідження є методи векторного представлення текстів і алгоритми визначення семантичної подібності, застосовані до задачі пошуку вакансій.

Практична цінність роботи полягає у реалізації повноцінного вебсервісу, що демонструє можливість застосування сучасних методів NLP для підвищення якості пошуку на ринку праці. Система здатна знаходити семантично близькі вакансії незалежно від формулювань, що підтверджено тестуванням на корпусі з понад 124 тисяч реальних вакансій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Цифровий ринок вакансій

Виникнення цифрового простору для пошуку праці зумовлене трансформацією ринку зайнятості, що стала наслідком розвитку ІКТ та широкого доступу до мережі Інтернет. Якщо раніше пошук роботи відбувався переважно через друковані оголошення, особисті зв'язки або посередництво кадрових компаній, то тепер він змістився у бік онлайн-майданчиків, які забезпечують зведення пропозицій та автоматизують взаємодію між тими, хто наймає, і тими, хто шукає роботу [1], [5].

Сучасний цифровий ринок праці відзначається величезними обсягами даних та високою плинністю інформації. Щодня великі портали публікують і коригують сотні тисяч відкритих позицій, формуючи значні, невпинно мінливі масиви даних [2], [3]. Ці відомості переважно текстові, створюються різними суб'єктами ринку без уніфікованих правил опису, що спричиняє їхню неоднорідність та ускладнює машинну обробку.

Ключовим інформаційним елементом на цифровому ринку вакансій є сам текстовий опис ролі. Зазвичай такий опис містить найменування посади, перелік функціональних обов'язків, вимоги до претендента, бажані вміння, умови праці та додаткові деталі. Усі ці складові представлені у неструктурованому вигляді або з мінімальною структуризацією, що ускладнює пряме зіставлення різних вакансій [4]. Навіть для позицій, ідентичних за суттю, роботодавці використовують різні формулювання, синоніми, аббревіатури та галузеву лексику.

Агрегатори та спеціалізовані ресурси для пошуку роботи, такі як LinkedIn, Indeed та локальні портали, відіграють вирішальну роль у функціонуванні цифрового ринку праці. Вони збирають оголошення з безлічі джерел і надають користувачам інструменти для фільтрації за зовнішніми критеріями – розташування, рівень досвіду, тип зайнятості чи

пошукові запити [2], [3]. Проте більшість таких рішень сфокусовані саме на формальних атрибутах вакансій і не беруть до уваги змістову подібність між описами, що обмежує результативність пошукових операцій.

Проблематика цифрового ринку вакансій особливо помітна у сфері інформаційних технологій. Для IT-сфери притаманна відсутність стандартизованих назв професій, швидка зміна кваліфікаційних вимог та виникнення нових спеціалізацій. Одна й та сама функція може мати різне найменування, наприклад, «Software Engineer», «Backend Developer» чи «Application Developer», проте поєднувати близький набір обов'язків та необхідних компетенцій [5]. Така багатоаспектність призводить до того, що традиційний пошук за ключовими словами не забезпечує повного виявлення відповідних пропозицій.

Окрім того, цифровий простір пошуку роботи характеризується різницею в обсягах інформації між сторонами. Роботодавці прагнуть максимально охопити потенційний кадровий резерв, тоді як здобувачі зацікавлені у швидкому знаходженні обмеженої кількості найбільш релевантних пропозицій. У підсумку, користувачі часто стикаються або з надмірною кількістю нерелевантних вакансій, або з пропуском пропозицій, які близькі за змістом, але мають інше текстове вираження [4].

Водночас, цифровий ринок вакансій продовжує розвиватися у напрямку автоматизації кадрових процесів та поширення платформної моделі. Дедалі ширше впроваджуються алгоритмічні методи відбору, системи рекомендацій та інструменти для попереднього оцінювання відповідності кандидата та вакансії. Якість таких рішень безпосередньо залежить від рівня аналізу текстових описів пропозицій та здатності системи розпізнавати їхній зміст, а не лише зовнішні ознаки [1], [5].

1.2 Проблема релевантності пошуку вакансій

Питання відповідності знайдених вакансій запитам шукачів роботи є одним із ключових аспектів функціонування сучасних онлайн-майданчиків із працевлаштування, і воно прямо впливає на успішність взаємодії між тими, хто шукає роботу, та наймачами. Загалом, релевантність визначається тим, наскільки результати пошуку збігаються з очікуваннями особи, її професійним досвідом, рівнем кваліфікації та кар'єрними прагненнями. Однак на практиці більшість джоб-платформ не надають достатньо точного зіставлення наявних пропозицій роботи та потреб користувачів, що підтверджується як аналізом ринку цифрової праці, так і звітністю міжнародних інституцій [1], [2].

Головна причина низької відповідності полягає у застосуванні простих пошукових систем, які ґрунтуються виключно на пошуку ключових слів. У подібних механізмах текст опису вакансії розглядається як звичайний набір термінів, без урахування їхнього значення у контексті чи взаємозв'язків.

Як наслідок, запит користувача трактується дослівно, що спричиняє відфільтровування пропозицій, які мають схожий зміст, але викладені іншими словами. Наприклад, вакансії під назвами «Backend Developer», «Server-side Engineer» чи «Python Software Engineer» можуть не з'явитися у видачі за запитом «Software Developer», незважаючи на значну ідентичність необхідних обов'язків та вимог [4].

Не менш вагомий вплив на якість пошуку має відсутність єдиних стандартів у оформленні опису вакансій. Роботодавці застосовують різноманітні стилі викладу інформації, неоднаковий ступінь деталізації та власну термінологію. Часто в межах одного оголошення змішуються вимоги для декількох посад або перелічуються навички, що не є критичними, але подаються нарівні з основними. У таких умовах стандартні методи

фільтрації нездатні чітко розмежувати принципово важливі вимоги від другорядних, що безумовно погіршує точність результатів [5].

Додатковою перешкодою є значний обсяг інформації, що подається користувачу. Через величезну кількість пропозицій на великих сервісах, навіть після використання базових фільтрів, користувач може отримати сотні чи тисячі варіантів.

Ручна обробка кожного оголошення вимагає значних часових витрат, що знижує загальну ефективність пошуку роботи. За даними аналітичних оглядів, користувачі часто обмежуються переглядом лише перших кількох сторінок видачі, ігноруючи ті вакансії, які могли б бути більш підходящими, але розташовані нижче [3].

Проблема релевантності особливо виражена у секторі ІТ-вакансій та суміжних цифрових спеціалізацій. Ця галузь характеризується швидкими змінами, появою нових рішень і напрямків, а також відсутністю усталених назв посад. Описи часто містять специфічні технічні терміни, акроніми та назви програмних продуктів, які постійно оновлюються. За таких умов пошук, заснований на точних ключових словах, не дає змоги повноцінно оцінити змістовну близькість вакансій до профілю шукача [2], [5].

Спроби покращити відповідність через використання відомостей про поведінку користувачів (історія пошуку, кліки чи переглянуті вакансії) забезпечують певне поліпшення, проте мають суттєві недоліки. По-перше, ці методи не спрацьовують для нових користувачів або коли людина вирішує змінити сферу діяльності. По-друге, вони зосереджені на виявленні загальних закономірностей поведінки, а не на аналізі суті самих вакансій, що обмежує їхню здатність знаходити нові або нетипові, але водночас доречні пропозиції [4].

Окрім того, якість пошуку погіршується через розбіжність інтересів сторін ринку. Роботодавці прагнуть охопити якнайширшу аудиторію, тому часто свідомо використовують надто загальні або привабливі формулювання. Натомість шукачі роботи зацікавлені у вузько

сфокусованому і точно підбраному переліку. Несинхронність цілей, у поєднанні з обмеженнями наявних пошукових алгоритмів, ускладнює отримання результатів високої якості [1].

З урахуванням усіх вищезазначених факторів, стає очевидним, що проблема релевантності пошуку роботи на сучасних онлайн-майданчиках має системний характер. Вони свідчать про недостатність традиційних методів і обґрунтовують потребу у впровадженні інтелектуальних підходів до аналізу текстових даних, які здатні враховувати семантичний зміст вакансій та їхню подібність, незалежно від того, як саме представлена інформація.

1.3 Огляд існуючих платформ пошуку роботи

Сучасний цифровий ринок праці представлений значною кількістю інтернет-майданчиків для пошуку роботи, які виступають посередниками між працедавцями та тими, хто шукає роботу. Ці платформи відрізняються своїм функціональним призначенням, моделями взаємодії з користувачами та типом алгоритмів, що застосовуються, проте переважна більшість функціонує на засадах, подібних до організації пошуку та подачі інформації про вакансії. До визнаних світових сервісів відносять LinkedIn, Indeed та Glassdoor, тоді як на теренах України найбільш відомі Work.ua, Robota.ua та Djinni [1], [2], [7], [8], [9].

Існуючі інтернет-майданчики для пошуку роботи можна умовно класифікувати за кількома основними категоріями. Традиційні job-портали сфокусовані на публікації та перегляді пропозицій роботи, використовуючи звичні фільтри та механізми пошуку. До цієї групи належать як міжнародні, так і регіональні ресурси пошуку праці, такі як Indeed, Work.ua та Rabota.ua. Основні риси класичних job-порталів:

- пошук пропозицій за ключовими словами;

- можливість фільтрації за місцем розташування, рівнем професійного досвіду та типом зайнятості;
- впорядкування отриманих результатів за часом розміщення або популярністю;
- функціонал для створення та збереження електронного резюме користувача.

Попри широку затребуваність, такі майданчики переважно використовують формалізовані критерії пошуку, ігноруючи семантичну близькість вакансій. Як наслідок, пропозиції з ідентичними обов'язками можуть не потрапляти у видачу через розбіжності у найменуваннях посад чи формулюваннях вимог [3].

Окрему нішу займають платформи, що поєднують можливості пошуку роботи з функціоналом соціальних мереж. Яскравим прикладом є LinkedIn. Такі сервіси дозволяють користувачам формувати професійний профіль, налагоджувати ділові контакти та отримувати персоналізовані рекомендації щодо вакансій. Ключові аспекти цих платформ:

- акцент на формуванні професійного портрета користувача;
- використання мережі контактів як додаткового джерела порад щодо вакансій;
- часткове застосування методів персоналізації;
- інтеграція пропозицій роботи з даними про кар'єрний шлях особи [2].

Однак, системи рекомендацій на цих ресурсах значною мірою залежать від зафіксованої поведінки та соціальних зв'язків, а не від глибокого аналізу змісту тексту самої вакансії, що знижує точність визначення їхньої справжньої змістовної відповідності.

Агрегатори займаються автоматизованим збиранням пропозицій з різноманітних першоджерел, включаючи офіційні сайти компаній та інші job-платформи. Типовим представником такого підходу є Indeed, який

концентрує вакансії з тисяч сторонніх ресурсів. Характерні ознаки агрегаторів:

- зведення пропозицій з численних зовнішніх джерел;
- мінімальний рівень уніфікації текстового наповнення;
- залежність від якості первинних даних;
- застосування базових алгоритмів пошуку [3].

Через різноманітність джерел, агрегатори часто стикаються з проблемою дублювання записів та появи нерелевантних результатів, що ускладнює здійснення ефективного пошуку для кінцевого споживача.

Вузькоспеціалізовані майданчики та нішеві сервіси орієнтовані на конкретні сектори ринку чи категорії спеціалістів, наприклад, IT-сфера, дизайн, фріланс або віддалена праця. Прикладами є Djinni, сфокусований на вакансіях для IT-фахівців, а також низка міжнародних нішевих рішень для вузьких спеціальностей. Основні особливості таких систем:

- обмежена предметна галузь;
- використання специфічної для індустрії лексики;
- спрощення процесу пошуку для визначеного кола користувачів;
- вищий ступінь відповідності в межах однієї спеціалізації [5].

Проте навіть у профільних системах пошук зазвичай базується на пошуку за ключовими словами та мітками, що не дає змоги повноцінно врахувати семантичні взаємозв'язки між вакансіями, які мають схожий зміст.

Незалежно від того, до якого типу належить платформа, більшість наявних рішень оперує подібними методами організації пошуку роботи. У стандартній схемі алгоритм функціонування майданчика охоплює такі кроки:

- отримання текстового запиту від користувача;
- застосування фільтрів за формальними ознаками;
- пошук пропозицій за точними збігами ключових слів;
- ранжування результатів відповідно до простих евристичних правил.

Такий підхід доволі легкий для впровадження, але його ефективність обмежена при роботі з великими обсягами неструктурованих текстових даних та значною варіативністю опису вакансій. Як наслідок, користувачам доводиться витратити багато часу на ручну перевірку результатів або багаторазове повторення пошукових запитів із різноманітними формулюваннями [4].

1.4 Обмеження класичних підходів до рекомендації вакансій

Традиційні методи рекомендації пропозицій працевлаштування, що застосовуються на переважній більшості сучасних інтернет-майданчиків для пошуку роботи, були вибудовані, спираючись на базові моделі пошуку інформації. Вони не були розроблені для глибокого опрацювання неструктурованих текстових матеріалів. Фундамент таких методик полягає у зіставленні наявних вакансій та запитів з боку претендентів за їхніми формальними атрибутами, що суттєво звужує потенціал якості пропонованого пошуку у світлі мінливого та різнопланового електронного ринку праці.

Однією з найвагоміших вад усталених механізмів рекомендацій є їхня схильність до орієнтації виключно на ключові слова. У більшості систем пропозиції формуються на основі ідентичності термінів, знайдених у тексті оголошення та пошуковому запиті або ж у профілі особи, що шукає роботу. Такий шлях ігнорує контекстне використання лексики, її синонімію чи семантичну близькість. Система в підсумку подає: вакансії зі схожим змістом, але іншим викладом, можуть бути не включені до списку рекомендованого, тоді як формально подібні, але концептуально нерелевантні добірки демонструються користувачеві.

Іншим серйозним недоліком є те, що ці методики не беруть до уваги внутрішню будову та значущість окремих фрагментів усередині опису вакансії. Стандартні алгоритми, як правило, сприймають текст вакансії як

єдиний потік слів, не проводячи розмежування між ключовими вимогами, факультативними навичками або загальними відомостями про організацію. Це призводить до того, що рекомендації базуються на другорядних елементах змісту і не відображають істинної суті професійної пропозиції.

Обмеженням, притаманним застарілим підходам, є також їхня неспроможність ефективно працювати з різноманітністю назв посад і професійних амплуа. На сучасному цифровому ринку праці одну й ту саму функціональну позицію можуть іменувати по-різному, залежно від компанії, територіальної належності чи сектору діяльності.

Формалізовані пошукові системи не беруть до уваги ці відмінності, змушуючи людей робити численні запити або вручну оглядати значну кількість оголошень, щоб виявити потрібні варіанти. Значний вплив на якість формування рекомендацій має також феномен інформаційного перенасичення. Традиційні платформи зазвичай генерують досить довгі переліки вакансій, які впорядковуються за простими емпіричними правилами, як-от дата публікації чи кількість переглядів.

За таких умов користувач отримує надмірну кількість результатів, де важко миттєво ідентифікувати найбільш релевантні пропозиції, що призводить до зниження ефективності процесу пошуку роботи та збільшення витрат часу.

Окрему групу недоліків формують методики рекомендацій, засновані на аналізі поведінки, які інколи впроваджуються на певних сервісах. Вони ґрунтуються на вивченні історії того, що користувач переглядав, на що клікав або які заявки подавав. Хоча такі техніки допомагають частково індивідуалізувати видачу, вони мають низку слабкостей:

- залежність від наявності попередніх даних;
- недостатня результативність для новачків;
- скрутність адаптації до зміни професійного фокусу;
- відсутність аналізу глибинної відповідності змісту вакансій.

Крім того, традиційні підходи до добору вакансій не враховують плинний характер ринку праці. Зміна вимог до фахівців, поява нових технологічних рішень та професій відбуваються швидше, аніж здійснюється оновлення правил та емпіричних засад, закладених у рекомендаційні системи. Це тягне за собою втрату актуальності пропонцій та демонстрацію оголошень, які хоча й відповідають формальному профілю користувача, проте суперечать його справжнім очікуванням.

Сукупність перелічених обмежень доводить, що усталені методики рекомендації вакансій не відповідають вимогам, які висуває сучасний електронний ринок праці. Вони не здатні забезпечити необхідний рівень точності, гнучкості та спроможності до пристосування, що виправдовує потребу у впровадженні інтелектуальних методів аналізу тексту та визначення семантичної відповідності вакансій як фундаментальної основи для створення оновлених вебсервісів пошуку роботи.

1.5 Постановка задачі інтелектуального пошуку схожих вакансій

Здійснений аналіз цифрового ринку праці, зокрема доступних онлайн-ресурсів та обмежень традиційних методів пошуку й підбору кадрів, виявляє, що ключовою проблемою сучасних сервісів є низька відповідність (релевантність) пропонованих результатів. Більшість існуючих рішень концентруються на поверхневих критеріях, як-от ключові слова, категорії чи зафіксовані дії користувачів, що не дає змоги належно оцінити глибинну схожість між пропозиціями роботи. У світлі цього, нагальною є потреба у впровадженні «розумного» пошуку, що ґрунтується на змістовному аналізі тексту вакансій.

Під інтелектуальним пошуком подібних вакансій у контексті цієї роботи розуміється процедура автоматичного виявлення робочих місць, які мають співмірність за професійним змістом, вимогами та функціональними обов'язками, незалежно від того, чи відрізняються назви посад,

формулювання чи структура опису. Суть такого механізму полягає у здатності знайти роботи, які є семантично близькими, навіть якщо в них відсутні прямі збіги ключових термінів.

Об'єктом дослідження в цій системі слугують вакансії, представлені у форматі вільного тексту, що охоплює найменування посади, опис функцій, необхідні компетенції, перелік навичок та умови співпраці. Як вхідні дані для інтелектуального пошуку можуть виступати як конкретна вакансія, обрана користувачем, так і текстовий запит чи профіль здобувача. Кінцевим результатом функціонування системи має бути впорядкований перелік пропозицій, розташованих за зростанням рівня їхньої відповідності вхідному елементу.

У загальному плані, завдання пошуку семантично схожих вакансій може бути зведене до проблеми визначення близькості між текстовими сутностями (документами). Для цього кожна позиція має бути трансформована у формалізоване числове представлення, яке зробить можливим співставлення та ранжування. Ключовою вимогою є здатність системи ефективно опрацьовувати великі масиви даних, зберігаючи при цьому прийнятну швидкість обробки та масштабованість.

Основні вимоги до системи інтелектуального пошуку аналогічних вакансій:

- врахування змістовної, а не виключно формальної відповідності описів;
- здатність працювати з різноманіттям формулювань назв посад і професійної лексики;
- передбачення можливості опрацювання неструктурованих текстових даних;
- формування списку результатів, упорядкованого за критерієм подібності;
- гнучкість до змін на ринку праці та появи нових фахових напрямів.

На відміну від традиційних механізмів рекомендацій, цей інтелектуальний підхід не повинен критично залежати лише від історії взаємодії респондента з ресурсом. Це дає змогу ефективно використовувати систему як для новачків, так і в ситуаціях зміни професійного вектору або дослідження альтернативних кар'єрних шляхів. Такий підхід є надзвичайно цінним на ринках праці, які демонструють високу динаміку та широке розмаїття пропозицій, яскравим прикладом чого є сфера інформаційних технологій.

Отже, у рамках цієї роботи ставиться мета – розробити вебсервіс для інтелектуального пошуку аналогічних вакансій. Він, базуючись на методологіях обробки природної мови та обчислення семантичної близькості, повинен забезпечити підвищення якості (релевантності) результатів пошуку. Впровадження такої системи здатне нівелювати головні вади існуючих класичних підходів та закласти фундамент для подальшого розвитку інтелектуальних сервісів рекомендацій на ринку праці.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО ПОШУКУ СХОЖИХ ВАКАНСІЙ

2.1 Текстові вакансії як об'єкт інтелектуального аналізу

2.1.1 Основні компоненти опису вакансії

Опис вакансії у системах пошуку роботи онлайн є початковою формою представлення даних про потребу роботодавця у фахівці та водночас ключовим об'єктом дослідження для алгоритмів інтелектуального пошуку. На відміну від заздалегідь упорядкованих документів, як-от анкети чи стандартні бланки, вакансії подаються у більш вільній формі, чия структура визначається не технічними нормами платформи, а індивідуальним стилем та усталеними практиками певного роботодавця чи рекрутингової агенції [10].

Найважливішим елементом опису є найменування посади, яке слугує коротким семантичним індикатором професійної позиції. Воно є головною точкою, з якої починається як для користувача, так і для пошукових систем процес пошуку. Проте дана частина відзначається високим ступенем невизначеності: одна й та сама функціональна роль може позначатися різними назвами, мати додаткові уточнення або, навпаки, бути занадто загальною. З огляду на це, назва посади не може вважатися надійним орієнтиром для визначення схожості вакансій без залучення інформації з інших частин опису.

Наступним за вагомістю елементом виступає виклад функціональних обов'язків, де розкривається сутність професійної діяльності. Зріз часто структурується як перелік або набір коротких речень, що описують завдання, за які відповідатиме найманий працівник. Зміст цих обов'язків несе значний інформаційний посил, проте він характеризується великою плінністю стилю подання, різним ступенем деталізації та активним

використанням синонімів. Як наслідок, пряме порівняння обов'язків між різними вакансіями ускладнюється без застосування методів семантичного аналізу тексту [10].

Особливу роль відіграє блок вимог до потенційного співробітника, який зазвичай охоплює професійні компетенції, стаж роботи, знання специфічних технологій, а іноді й особистісні якості. Він найбільш тісно корелює з поняттям професійної відповідності вакансій, адже він демонструє очікуваний профіль фахівця. Водночас, вимоги можуть бути подані як у чіткій, структурованій формі, так і у вигляді загальних формулювань, де різні типи відомостей зливаються в одному реченні, що створює додаткові перешкоди для автоматизованої обробки та вимагає попереднього приведення тексту до єдиного стандарту. До складу опису вакансії нерідко включають також дані про умови праці та спосіб працевлаштування, наприклад, тип договору, можливість працювати дистанційно, розклад чи ступінь зайнятості.

Хоча ці аспекти не визначають безпосередньо професійний напрямок вакансії, вони здатні впливати на релевантність знахідок для кінцевого користувача. У межах інтелектуального пошуку ці складові варто розглядати як другорядні характеристики, що доповнюють основний аналіз семантики. Бонусним елементом відображається інформація про компанію, котра часто має переважно описовий або рекламний характер. Вони можуть містити загальні твердження, не пов'язані напряму з професійною роллю, але які збільшують загальний обсяг тексту та засмічують процес аналізу. Без належного поводження, ці фрагменти можуть негативно позначитися на результатах зіставлення вакансій, зміщуючи фокус із суті професії на менш значущу інформацію [10].

Нерівномірність та різний рівень інформативності окремих складових опису вакансії обумовлюють потребу в їхньому диференційованому обліку при формуванні моделей представлення вакансій, створюючи умови для застосування методів, які здатні працювати з текстами складної архітектури

та виокремлювати значущі за змістом частини незалежно від формальних схем.

2.1.2 Лінгвістичні особливості текстів вакансій

Описи вакансій, що розміщуються у цифрових пошукових системах, постають як приклади неструктурованих текстів, де відсутнє дотримання якихось уніфікованих мовних чи стилістичних норм. Головна їхня мета полягає не у досягненні лінгвістичної бездоганності, а у гранично швидкій та однозначній передачі відомостей до потенційного працівника. Звідси впливають певні мовні атрибути, які суттєво ускладнюють можливості автоматичної розробки та зіставлення пропозицій працевлаштування у рамках розвинених пошукових рішень [11].

Однією з визначальних рис є високе насичення спеціалізованою лексикою. У тексті вакансій надзвичайно активно залучаються терміни, релевантні до певної професійної сфери, зокрема згадуються назви технологічних рішень, апаратних засобів, мов кодування, регулятивних актів, методів роботи чи внутрішніх процедур фірми.

Чимало такої термінології не є частиною загального словника, часто подається латиницею або у формі аббревіатур. Для програмних комплексів це означає необхідність опрацювання великого масиву вузькоспрямованих одиниць, які неможливо чітко трактувати поза специфічною професійною площиною.

Важливою характеристикою є те, що ідентичні або дуже схожі за змістом вимоги можуть бути виражені абсолютно різними фразами. Одна й та сама кваліфікація здатна фігурувати як повна фраза, як маркер у переліку чи як окреме ключове слово. До того ж, описи часто містять варіанти викладу тих самих понять, що призводить до штучного розширення слововжитку без справжнього збагачення змістового розмаїття. Це

ускладнює застосування формальних підходів до співставлення текстів і зменшує віддачу від лексичних моделей.

Синтаксична будова описів працевлаштування типово є або спрощеною, або уривчастою. Замість завершених речень використовуються конструкції з дієсловами в інфінітиві, номінативні словосполучення або ж прості списки, позбавлені чітких граматичних зв'язків. Хоча такий стиль полегшує швидке усвідомлення інформації людиною, він створює проблеми для програмних систем, що ґрунтуються на усталених синтаксичних правилах. У подібних умовах аналіз взаємозв'язків між словами чи побудова дерева залежностей дає невелику кількість корисної інформації.

Окремий виклик являє собою поєднання різних мов в одному документі про вакансію. В одному описі можуть сумісно існувати елементи української, англійської чи інших мов, доповнені професійним жаргоном та внутрішніми скороченнями. Лінгвістичне накладання руйнує однорідність текстових даних і паралізує роботу стандартних алгоритмів обробки природної мови, які були розраховані на одномовні корпуси.

Іншою помітною рисою є нечіткість у значенні назв робочих місць та функціональних ролей. Найменування вакансій часто містять занадто загальні або навіть рекламні формулювання, які не відображають суті виконання роботи.

Одна й та сама назва може застосовуватися для позицій з різним колом обов'язків, тоді як різні назви можуть приховувати практично ідентичний набір функцій. За таких умов аналіз лише заголовка не дає змоги робити обґрунтовані висновки про його спорідненість з іншими пропозиціями.

Крім того, тексти оголошень містять значну кількість частин, які не стосуються безпосередньо функціоналу, зокрема опис самої організації, корпоративних цінностей або загальних бонусів. Подібні секції мають низьку інформативну вагу щодо сутності професійних завдань, проте збільшують загальний обсяг тексту і можуть впливати на результати машинного зіставлення. Без належного очищення вони генерують

семантичний «шум» та зміщують фокус аналізу від ключових критеріїв вакансії.

Усі згадані мовні особливості визначають специфіку вакансій як комплексного об'єкта для аналітичного опрацювання та диктують необхідність використання таких методів, котрі здатні оперувати неоднорідними, контекстно залежними та термінологічно насиченими текстами. Це безпосередньо впливає на вибір моделей представлення тексту та алгоритмів для визначення міри схожості вакансій у процесі інтелектуального пошуку.

2.1.3 Проблеми неструктурованості та шуму даних у вакансіях і способи їх усунення

У завданні інтелектуального пошуку подібних вакансій, неорганізованість текстових описів сприймається не як недолік у даних, а радше як первинна умова, що диктує вибір способів репрезентації та зіставлення. На відміну від традиційних систем пошуку інформації, де вирішальне значення має наявність чітко визначених полів та характеристик, у цьому завданні фокус зміщується на опрацювання суцільного текстового масиву, що описує вакансії [12].

Ключовим технологічним наслідком цієї неструктурованості є те, що унеможлиблюється застосування правил, базованих на фіксованих зразках чи аналізі порядку розташування слів. Зокрема, методи, які покладаються на стабільне місцезнаходження відомостей про вимоги чи обов'язки, демонструють ненадійні результати при зміні формату подання вакансій. У контексті цієї кваліфікаційної роботи це змушує відмовитися від підходів, сфокусованих на структурі, на користь таких методів, які не залежать від послідовності чи конкретної форми викладу інформації.

Для мінімізації впливу тих частин тексту, які не несуть прямого функціонального навантаження, застосовується семантично нейтральне

очищення. Воно не оперує строгими правилами, а спрямоване на зниження значущості загальновживаної та часто повторюваної лексики, дозволяючи зменшити вплив типових описів компаній та загальних фраз, не вдаючись до фізичного видалення цілих текстових секцій, що є важливим для збереження загального змісту опису вакансії [12].

Головним інструментом для подолання цієї неорганізованості в рамках кваліфікаційної роботи слугує перехід до векторного представлення вакансій, коли увесь текст обробляється як єдиний об'єкт. У ній відомості з різних фрагментів опису зводяться в узагальнену числову форму, що дає змогу порівнювати вакансії незалежно від їхньої внутрішньої верстки, що усуває необхідність попереднього розмічування тексту або ручного виокремлення ключових сегментів.

Щоб зменшити вплив різноманітності лексики та нестабільності формулювань, у кваліфікаційній роботі використовуються моделі репрезентації, сфокусовані на змістовному наповненні, а не на зовнішньому вигляді тексту. Подібні моделі здатні встановлювати близькість між вакансіями зі схожим професійним змістом, навіть якщо вони не мають спільних ключових слів. Це є принциповим для завдання пошуку аналогічних вакансій, адже формальні відмінності у викладі не мусять призводити до ігнорування релевантних знахідок [12].

2.2 Подання вакансій і методи визначення подібності

2.2.1 Формальні моделі подання тексту вакансій

У задачах інтелектуального пошуку вакансій формальні моделі подання тексту застосовуються як базовий інструмент перетворення неструктурованих текстових описів у числову форму, що дозволяє виконувати порівняння, ранжування та обчислення подібності. Розглядаючи вакансію як текстовий документ, такі моделі абстрагуються від мовних і

стилістичних особливостей, зосереджуючись на статистичних характеристиках використання термінів [11].

Найпоширенішим підходом є векторна модель простору термінів, у межах якої кожна вакансія подається у вигляді вектора, компоненти якого відповідають окремим словам або термінам. Формуючи такий вектор, кожен текст розкладається на набір токенів, після чого визначається їхня кількісна присутність у документі.

Ігноруючи порядок слів і синтаксичні зв'язки, модель зберігає інформацію про лексичний склад вакансії, що є достатнім для реалізації базових механізмів пошуку. Розширюючи підхід простого підрахунку слів, у формальних моделях широко використовується TF-IDF представлення, яке враховує не лише частоту терміна в окремій вакансії, але й його поширеність у всьому корпусі вакансій. Застосовуючи таке зважування, зменшується вплив загальноновживаних слів і підвищується роль термінів, характерних для конкретної професійної сфери. У контексті вакансій це дозволяє акцентувати увагу на назвах навичок, технологій і ролей, відкидаючи нефункціональну лексику.

Використовуючи формальні моделі подання, вакансії розміщуються у високовимірному векторному просторі, у якому близькість між документами визначається за допомогою стандартних метрик. Зіставляючи такі вектори, система отримує можливість оцінювати подібність вакансій без прямого аналізу тексту, що істотно спрощує алгоритмічну реалізацію пошуку. При цьому всі вакансії обробляються однаковою чином, незалежно від їх внутрішньої структури та стилю подання.

Разом із тим, застосовуючи формальні моделі, необхідно враховувати їх принципове обмеження, пов'язане з ігноруванням семантики та контексту. Розглядаючи слова як незалежні одиниці, такі підходи не розрізняють синонімічні формулювання та не здатні виявляти приховану змістовну близькість між вакансіями, що описують однакові професійні ролі

різними словами. Унаслідок цього формальні моделі забезпечують стабільну, але обмежену за точністю основу для інтелектуального пошуку.

У межах кваліфікаційної роботи формальні моделі подання тексту використовуються як базовий рівень представлення вакансій, слугуючи точкою відліку для подальшого переходу до семантичних методів. Такий підхід дозволяє чітко відокремити ефект статистичного зіставлення текстів від переваг моделей, що враховують зміст і контекст, а також забезпечує порівняльний аналіз різних підходів до визначення подібності вакансій.

2.2.2 Семантичні моделі подання вакансій

Семантичні моделі подання тексту базуються на ідеї, що зміст мовних одиниць визначається їхнім контекстом і взаємозв'язками з іншими елементами тексту. На відміну від формальних підходів, які оперують ізольованими словами або частотними ознаками, семантичні моделі дозволяють виявляти глибинні смислові структури, приховані у великих масивах неструктурованих даних [10]. Для аналізу вакансій це має принципове значення, оскільки тексти оголошень характеризуються різноманітністю стилів, термінів і способів опису професійних вимог.

Основою семантичного підходу є перетворення тексту у векторне представлення у багатовимірному просторі ознак. У такому просторі кожен текст вакансії подається у вигляді числового вектора, який акумулює інформацію про його зміст. Відстань між векторами відображає семантичну близькість відповідних текстів, що дозволяє порівнювати вакансії не за формальними збігами слів, а за їхнім змістовним наповненням.

Формування семантичних векторів ґрунтується на аналізі контекстів, у яких з'являються слова та фрази. Якщо різні терміни використовуються в подібних мовних оточеннях, модель інтерпретує їх як близькі за значенням. Завдяки цьому семантичні моделі здатні коректно працювати з синонімією, професійним жаргоном, альтернативними назвами посад і технологій, що є

типовим для текстів вакансій у сфері інформаційних технологій, менеджменту та інженерії. Застосування семантичних моделей дозволяє розглядати вакансію як цілісний опис професійної діяльності, а не як набір окремих ключових слів.

У векторному представленні враховуються як назва посади, так і перелік обов'язків, вимог, інструментів, методів роботи та умов зайнятості. Це забезпечує більш повне відображення професійного профілю вакансії та зменшує втрати інформації під час автоматизованого аналізу.

Важливою властивістю семантичних моделей є їхня здатність узагальнювати інформацію. Навіть за відсутності прямого лексичного збігу між двома описами вакансій, модель може визначити їхню близькість на основі спільних контекстів використання термінів. Особливо актуально в умовах, коли роботодавці використовують індивідуальні формулювання або нестандартні назви посад, що унеможливорює ефективний пошук за ключовими словами.

Семантичне подання також спрощує роботу з великими текстами вакансій, які можуть містити значну кількість другорядної інформації. Перетворюючи текст у компактний вектор фіксованої розмірності, модель зберігає ключові смислові характеристики, водночас зменшуючи обчислювальну складність подальших операцій порівняння та ранжування, створюючи передумови для масштабування систем пошуку вакансій без суттєвого зниження продуктивності.

Окрему роль семантичні моделі відіграють у забезпеченні мовної та стилістичної гнучкості системи пошуку. За наявності відповідних навчальних даних вони можуть застосовуватися до вакансій різними мовами або до багатомовних описів, що характерно для міжнародних і українських платформ пошуку роботи, що дозволяє зменшити залежність результатів пошуку від формальних мовних обмежень і підвищити універсальність підходу.

Семантичні моделі подання текстів вакансій створюють основу для переходу від формального пошуку до інтелектуального аналізу професійного змісту оголошень. Вони забезпечують можливість коректного зіставлення вакансій із різними формулюваннями, стилями та структурою опису, що є необхідною умовою для побудови ефективних систем пошуку схожих вакансій у сучасному цифровому середовищі.

2.2.3 Методи обчислення подібності та ранжування

Після формування векторного подання вакансій ключовим етапом інтелектуального пошуку стає обчислення ступеня подібності між ними та подальше впорядкування результатів. На цьому етапі абстрактні числові представлення текстів трансформуються у практично значущі результати для користувача – список вакансій, відсортований за рівнем їхньої відповідності заданому запиту або обраній позиції. Якість безпосередньо визначає ефективність усього вебсервісу пошуку схожих вакансій.

У векторному просторі подібність між вакансіями визначається за допомогою спеціальних математичних метрик, які оцінюють відстань або кут між відповідними векторами. Однією з найбільш уживаних у задачах аналізу текстів є косинусна подібність. Вона ґрунтується на вимірюванні кута між двома векторами та дозволяє оцінити схожість незалежно від абсолютної довжини векторів, що принципово важливе для текстів вакансій, оскільки їхній обсяг може суттєво відрізнятися: одні оголошення містять лаконічний перелік вимог, тоді як інші включають розгорнуті описи компанії та умов праці. Косинусна метрика в такому випадку фокусується саме на напрямку вектора, тобто на змістовному наповненні тексту.

Для формальних моделей подання тексту, зокрема TF-IDF, косинусна подібність є стандартним інструментом зіставлення вакансій. Вона дозволяє ефективно визначати ступінь лексичної близькості між описами та слугує базовою точкою відліку для порівняння з більш складними семантичними

підходами. Разом із тим, у таких моделях результати подібності залишаються чутливими до точних збігів термінів і не завжди відображають реальну професійну спорідненість вакансій.

У випадку семантичних моделей подання, де вакансії представлені у вигляді щільних векторів фіксованої розмірності, косинусна подібність також зберігає свою актуальність, проте її інтерпретація змінюється. Тут вона відображає не формальну лексичну близькість, а схожість смислових контекстів, у яких формувався вектор, дозволяючи коректно визначати подібність між вакансіями з різними формулюваннями, але зі спільним професійним змістом, що є критичним для задачі інтелектуального пошуку.

Альтернативним підходом до вимірювання подібності є використання евклідової або мангеттенської відстані у векторному просторі, метрики якої оцінюють абсолютну віддаленість між точками, проте у задачах аналізу текстів вони застосовуються значно рідше. Причиною цього є висока чутливість до масштабу векторів та менш інтуїтивна інтерпретація результатів у багатовимірних просторах. У контексті вакансій ці метрики можуть призводити до спотворених оцінок схожості, особливо при роботі з текстами різної довжини та насиченості.

Після обчислення значень подібності між вхідною вакансією або запитом і всіма елементами корпусу, система переходить до етапу ранжування. Ранжування полягає у впорядкуванні вакансій за спаданням значення обраної метрики подібності. Таким чином, користувач отримує список пропозицій, де перші позиції займають вакансії з максимальною семантичною близькістю до вихідного запиту. У межах інтелектуального пошуку це дозволяє значно скоротити кількість нерелевантних результатів та зосередити увагу на найбільш перспективних варіантах.

Важливою особливістю ранжування у задачі пошуку вакансій є те, що абсолютне значення подібності має другорядне значення порівняно з відносним порядком результатів. Користувач, як правило, взаємодіє лише з першими кількома позиціями у списку, тому головною метою алгоритму є

коректне визначення саме найближчих за змістом вакансій. Це накладає вимоги до стабільності та узгодженості методу обчислення подібності, особливо при роботі з великими корпусами даних.

Обчислення подібності та ранжування розглядаються як завершальний, але водночас інтегруючий етап інтелектуального пошуку. Вони поєднують результати попередніх кроків – очищення тексту, формування векторних представлень та вибір моделі – у цілісний механізм, здатний трансформувати неструктуровані описи вакансій у впорядковану та корисну для користувача інформацію.

2.3 Архітектура та принципи функціонування вебсервісу

2.3.1 Загальна архітектура системи

Вебсервіс інтелектуального пошуку схожих вакансій реалізується як багаторівнева програмна система, архітектура якої орієнтована на розділення функціональних обов'язків між окремими компонентами. Такий підхід дозволяє підвищити керованість системи, спростити її масштабування та забезпечити стабільну роботу при зростанні обсягів даних.

В основі архітектури лежить клієнт-серверна модель, у межах якої всі обчислювально складні операції виконуються на сервері, тоді як клієнтська частина зосереджена на взаємодії з користувачем. Це дозволяє зменшити вимоги до апаратних ресурсів клієнта та централізувати логіку інтелектуального пошуку.

Архітектурно систему доцільно розділити на кілька логічних рівнів, кожен з яких виконує визначений набір функцій:

- рівень представлення, що відповідає за вебінтерфейс і відображення результатів пошуку;
- прикладний рівень, який реалізує бізнес-логіку обробки запитів;

- рівень інтелектуального аналізу, призначений для роботи з текстами вакансій;

- рівень зберігання даних, що забезпечує доступ до вакансій і допоміжної інформації.

Клієнтська частина системи реалізується у вигляді вебінтерфейсу та виконує функції подання інформації користувачу. Вона забезпечує введення пошукових запитів або вибір базової вакансії, а також відображення впорядкованого списку схожих пропозицій. При цьому клієнтський рівень не містить алгоритмів аналізу тексту або обчислення подібності, що відповідає принципу мінімізації його складності.

Серверна частина вебсервісу виступає центральним елементом системи та об'єднує основні модулі інтелектуального пошуку. До її складу входять:

- модуль приймання та обробки запитів від клієнта;
- модуль попередньої обробки тексту вакансій;
- модуль формування векторних представлень;
- модуль обчислення подібності та ранжування;
- інтерфейс доступу до бази даних.

Підсистема зберігання даних відповідає за накопичення та підтримку корпусу вакансій. У ній зберігаються не лише текстові описи, але й результати попередніх етапів обробки, зокрема векторні представлення. Така організація дозволяє скоротити час відповіді на пошукові запити та зменшити навантаження на сервер під час повторних звернень.

Окреме місце в архітектурі займає підсистема інтелектуального аналізу, яка забезпечує перехід від неструктурованих текстів до формалізованого простору ознак. Вона реалізує повний цикл роботи з текстами вакансій – від очищення і нормалізації до підготовки даних для порівняння у векторному просторі.

Процес взаємодії між клієнтською та серверною частинами здійснюється через програмний інтерфейс (API), який визначає формат передавання запитів і відповідей. Такий підхід дозволяє:

- відокремити інтерфейс користувача від внутрішньої логіки системи;
- спростити тестування та супровід сервісу;
- забезпечити можливість підключення альтернативних клієнтів у майбутньому.

2.3.2 Алгоритм інтелектуального пошуку вакансій

Алгоритм інтелектуального пошуку вакансій побудований як послідовність формалізованих кроків, спрямованих на знаходження та ранжування пропозицій роботи, семантично близьких до заданого запиту або вибраної вакансії. Вхідними даними алгоритму є текст вакансії або текстовий запит користувача, а вихідними – впорядкований список вакансій, відсортований за рівнем змістовної подібності. На першому етапі алгоритму здійснюється отримання вхідних даних.

Користувач може ініціювати пошук двома способами: шляхом вибору конкретної вакансії з наявного корпусу або шляхом введення довільного текстового запиту. У випадку вибору вакансії її текст використовується безпосередньо як еталон для порівняння з іншими записами.

Другим етапом є попередня обробка тексту. На цьому кроці вхідний текст та тексти вакансій у базі приводяться до уніфікованого вигляду. Обробка включає:

- приведення тексту до єдиного регістру;
- видалення спеціальних символів та службових знаків;
- усунення надмірних пробілів і форматовальних елементів;
- мінімальну нормалізацію лексики.

Ці операції спрямовані на зменшення шуму в даних та забезпечення коректної роботи подальших етапів алгоритму.

Після очищення тексту виконується перетворення вакансій у векторне представлення. Для цього використовується попередньо обрана модель подання тексту, яка формує щільний числовий вектор фіксованої розмірності. Вектор формується для: вхідного запиту або обраної вакансії і у кожної вакансії, що зберігається у системі.

Векторні представлення вакансій зберігаються у базі даних або кеші, що дозволяє уникнути повторного обчислення при кожному запиті.

Наступним кроком є обчислення ступеня подібності між вектором запиту та векторами вакансій. Для цього застосовується косинусна метрика, яка дозволяє оцінити змістовну близькість незалежно від довжини тексту. Результатом цього етапу є числове значення подібності для кожної вакансії з корпусу.

На основі отриманих значень здійснюється ранжування вакансій. Усі пропозиції сортуються за спаданням коефіцієнта подібності, після чого формується обмежений список результатів, який містить лише вакансії з найвищими показниками відповідності. За необхідності до цього етапу можуть застосовуватися додаткові фільтри, наприклад за типом зайнятості або формальними атрибутами, проте вони не впливають на основний механізм семантичного зіставлення.

Завершальним етапом алгоритму є передавання результатів на клієнтську частину системи. Користувачу відображається впорядкований список вакансій, де кожна позиція відповідає конкретному рівню семантичної близькості до вхідного запиту. Таким чином забезпечується швидкий доступ до найбільш релевантних пропозицій без необхідності ручного перегляду великої кількості оголошень.

Запропонований алгоритм є детермінованим, масштабованим і не залежить від історії поведінки користувачів. Він базується виключно на аналізі змісту вакансій, що дозволяє ефективно застосовувати його як для нових користувачів, так і для сценаріїв пошуку альтернативних або суміжних професійних напрямів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Підготовка та очищення даних

Вхідними даними для системи слугує набір вакансій із платформи LinkedIn, отриманий у форматі CSV. Датасет містить поля найменування посади (title), опис вакансії (description) та перелік навичок (skills_desc). Перед передачею текстів на векторизацію їх необхідно привести до уніфікованого вигляду – видалити шумові елементи, зменшити лексичну надмірність і сформувати єдине текстове поле.

Попередня обробка реалізована у модулі preprocess.py і охоплює кілька послідовних етапів. На першому з них здійснюється очищення від HTML-розмітки: частина вакансій містить теги форматування, які не несуть смислового навантаження. Відповідна функція застосовує декодування HTML-сутностей та регулярний вираз для вилучення тегів (лістинг 3.1).

Лістинг 3.1 – Функція очищення від HTML-розмітки

```
def strip_html(text: str) -> str:
    text = html.unescape(text)
    text = re.sub(r"<[^>]+>", " ", text)
    return text
```

Наступним кроком є видалення шаблонних фраз. Описи вакансій часто містять стандартні блоки на кшталт «equal opportunity employer», «apply now» або «about us», які присутні в більшості оголошень і не відображають конкретний професійний зміст. Такі речення виявляються за регулярним виразом та виключаються з тексту (лістинг 3.2).

Лістинг 3.2 – Видалення шаблонних фраз

```
_BOILERPLATE = re.compile(
    r"(equal opportunity employer|eoe|we are an equal|apply now|"
```

```

r"click here|about us|our company|join our team|"
r"we offer|competitive salary|benefits include)",
re.IGNORECASE,
)

def remove_boilerplate(text: str) -> str:
    sentences = re.split(r"(?<=[.!?])\s+", text)
    filtered = [s for s in sentences if not
        _BOILERPLATE.search(s)]
    return " ".join(filtered)

```

Після цього текст очищується від спеціальних символів, приводиться до нижнього регістру та нормалізуються пробіли. Вся послідовність об'єднана у функцію `clean_text`.

Окремої уваги заслуговує формування зведеного текстового поля. Оскільки назва посади є найбільш інформативним елементом вакансії, їй надається підвищена вага: поле `title` додається до результуючого тексту триразово. Поле `description` додається в повному обсязі, а `skills_desc` – за наявності. Реалізацію наведено у лістингу 3.3.

Лістинг 3.3 – Формування зведеного текстового поля

```

def build_combined_text(row: pd.Series) -> str:
    parts = []
    title = str(row.get("title", "")).strip()
    if title:
        parts.append((title + " ") * 3)
    desc = str(row.get("description", "")).strip()
    if desc:
        parts.append(desc)
    skills = str(row.get("skills_desc", "")).strip()
    if skills and skills.lower() != "nan":
        parts.append(skills)
    return clean_text(" ".join(parts))

```

Перед збереженням виконується фільтрація: видаляються рядки з відсутнім або надто коротким описом (менше 50 символів), а також

дублікати за полем description. Схему повного конвеєру підготовки даних наведено на рисунку 3.1.

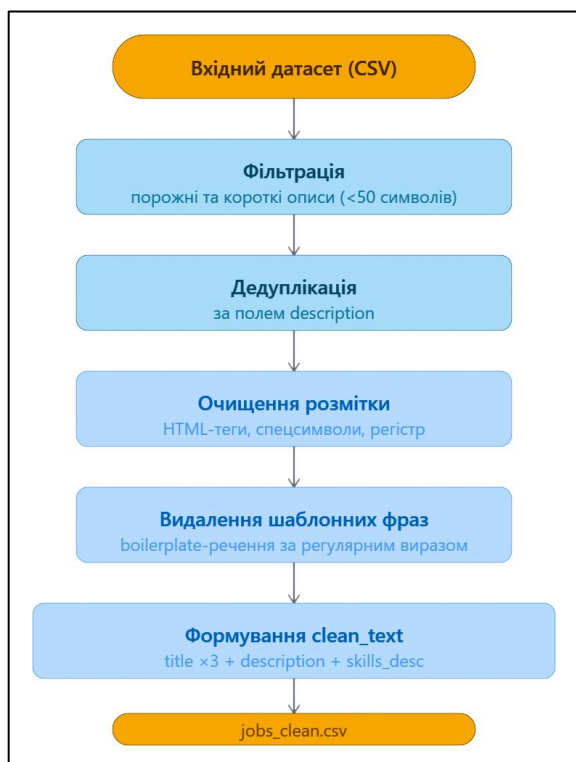


Рисунок 3.1 – Конвеєр підготовки та очищення даних

Результуючий файл `jobs_clean.csv` містить поле `clean_text`, яке передається на вхід моделі векторизації на наступному етапі.

3.2 Векторизація вакансій на основі SBERT

Після формування очищеного текстового поля кожна вакансія має бути перетворена у числове представлення, придатне для обчислення семантичної подібності. Для цього у системі застосовується модель SBERT (Sentence-BERT) – зокрема варіант `all-MiniLM-L6-v2`, що є дистильованою версією повної BERT-архітектури, оптимізованою для генерації ембедінгів речень.

Модель приймає на вхід довільний текстовий рядок і повертає вектор фіксованої розмірності 384. Ключовою властивістю такого представлення є те, що семантично близькі тексти отримують близькі вектори у цьому просторі, тоді як різні за змістом – віддалені. Саме ця властивість є основою механізму пошуку схожих вакансій.

Процес векторизації реалізований у модулі `vectorize.py`. Для пришвидшення обробки великого корпусу тексти подаються на вхід моделі пакетами. При кодуванні увімкнено нормалізацію векторів – параметр `normalize_embeddings=True`, що приводить усі вектори до одиничної довжини. Це дозволяє надалі обчислювати косинусну подібність як просте скалярне множення, що є обчислювально ефективнішим. Реалізацію функції кодування наведено у лістингу 3.4.

Лістинг 3.4 – Пакетне кодування текстів вакансій

```
def encode_texts(model: SentenceTransformer, texts: list[str])
-> np.ndarray:
    all_embeddings = []
    for i in tqdm(range(0, len(texts), BATCH_SIZE),
desc="Encoding"):
        batch = texts[i : i + BATCH_SIZE]
        emb = model.encode(
            batch,
            batch_size=BATCH_SIZE,
            show_progress_bar=False,
            normalize_embeddings=True,
            convert_to_numpy=True,
        )
        all_embeddings.append(emb)
    return np.vstack(all_embeddings).astype(np.float32)
```

Для оптимізації часу розгортання системи передбачено механізм кешування результатів векторизації. Якщо файл `embeddings_backup.npy` вже існує, система завантажує готові ембедінги без повторного запуску моделі.

Якщо ж файл відсутній – генерує ембедінги та зберігає їх для подальшого використання. Логіку вибору наведено у лістингу 3.5.

Лістинг 3.5 – Логіка завантаження або генерації ембедінгів

```
def encode_texts(model: SentenceTransformer, texts: list[str])
-> np.ndarray:
    all_embeddings = []
    for i in tqdm(range(0, len(texts), BATCH_SIZE),
desc="Encoding"):
        batch = texts[i : i + BATCH_SIZE]
        emb = model.encode(
            batch,
            batch_size=BATCH_SIZE,
            show_progress_bar=False,
            normalize_embeddings=True,
            convert_to_numpy=True,
        )
        all_embeddings.append(emb)
    return np.vstack(all_embeddings).astype(np.float32)
```

Назва моделі та розмір пакету винесені у змінні середовища SBERT_MODEL і BATCH_SIZE, що дозволяє змінювати їх без модифікації коду. За замовчуванням використовується all-MiniLM-L6-v2 з розміром пакету 64.

Після завершення векторизації результуючий масив ембедінгів разом із метаданими вакансій передається на запис до бази даних, що є предметом наступного підрозділу.

3.3 Зберігання векторних представлень у PostgreSQL з розширенням pgvector

Для зберігання та індексування векторних представлень вакансій у системі використовується PostgreSQL з розширенням pgvector. Воно додає

підтримку типу даних `vector`, операторів векторної арифметики та спеціалізованих індексів для ефективного пошуку найближчих сусідів.

Схему таблиці, у якій зберігаються вакансії разом із ембедінгами, наведено у лістингу 3.6.

Лістинг 3.6 – Схема таблиці `jobs`

```
CREATE TABLE IF NOT EXISTS jobs (
  id          SERIAL PRIMARY KEY,
  source_id   TEXT,
  title       TEXT          NOT NULL,
  company     TEXT,
  description TEXT          NOT NULL,
  location    TEXT,
  experience  TEXT,
  work_type   TEXT,
  embedding   vector(384)
);
```

Поле `embedding` має тип `vector(384)`, що відповідає розмірності виходу моделі `all-MiniLM-L6-v2`. Решта полів містять метадані вакансії, які повертаються клієнту разом із результатами пошуку.

Запис вакансій до бази виконується пакетно через `execute_values` з бібліотеки `psycopg2`. Перед вставкою таблиця очищується командою `TRUNCATE`, що забезпечує повне оновлення корпусу при повторному запуску векторизації. Вектор передається як рядок у форматі `[v1,v2,...,vN]` і приводиться до типу `vector` безпосередньо у SQL-запиті через приведення `%s::vector`.

Для прискорення пошуку на полі `embedding` створюється індекс типу `HNSW` (Hierarchical Navigable Small World). На рисунку 3.2 наведено порівняння лінійного перебору з `HNSW`-пошуком за кількістю операцій порівняння залежно від розміру корпусу.

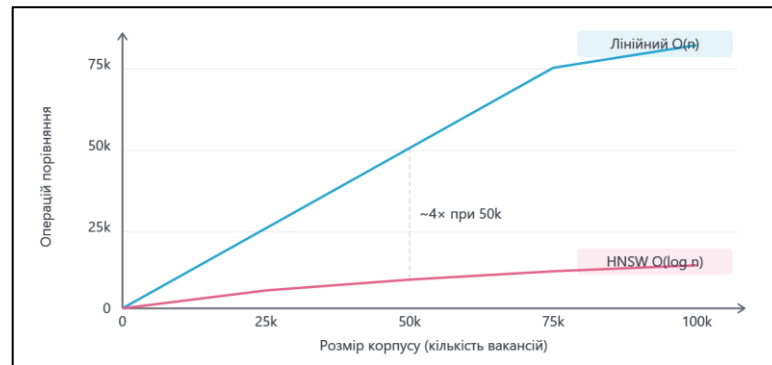


Рисунок 3.2 – Порівняння кількості операцій при лінійному пошуку та HNSW залежно від розміру корпусу

HNSW будує багаторівневий граф між векторами і знаходить найближчих сусідів за логарифмічний час замість лінійного перебору. Як метрику відстані обрано косинусну – `vector_cosine_ops`, що узгоджується з нормалізацією ембедінгів на етапі векторизації. Оператор `<=>` у `pgvector` обчислює косинусну відстань, а подібність визначається як $1 - (\text{embedding} \leq \text{query}::\text{vector})$. Індекс створюється одноразово при ініціалізації бази (лістинг 3.7).

Лістинг 3.7 – Створення HNSW-індексу

```
CREATE INDEX IF NOT EXISTS jobs_embedding_idx
ON jobs USING hnsw (embedding vector_cosine_ops);
```

На рисунку 3.3 показано потік даних від вхідного CSV-файлу через Python-компонент до PostgreSQL і далі до серверної частини системи.

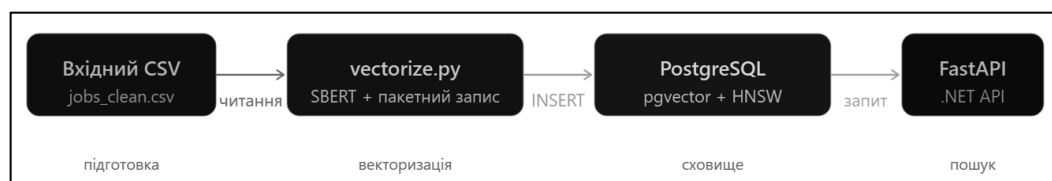


Рисунок 3.3 – Потік даних від вхідного CSV до серверної частини системи

Розміщення бази у хмарному сервісі Supabase дозволяє використовувати pgvector без локального розгортання PostgreSQL. Підключення здійснюється через стандартний рядок з'єднання psycopg2, параметри якого зчитуються з файлу .env.

3.4 Реалізація FastAPI мікросервісу семантичного пошуку

Семантичний пошук вакансій реалізований у вигляді окремого Python-мікросервісу на базі фреймворку FastAPI. Такий підхід дозволяє ізолювати ML-логіку від серверної частини на .NET і незалежно масштабувати обидва компоненти. Загальну послідовність обробки запиту наведено на рисунку 3.4.

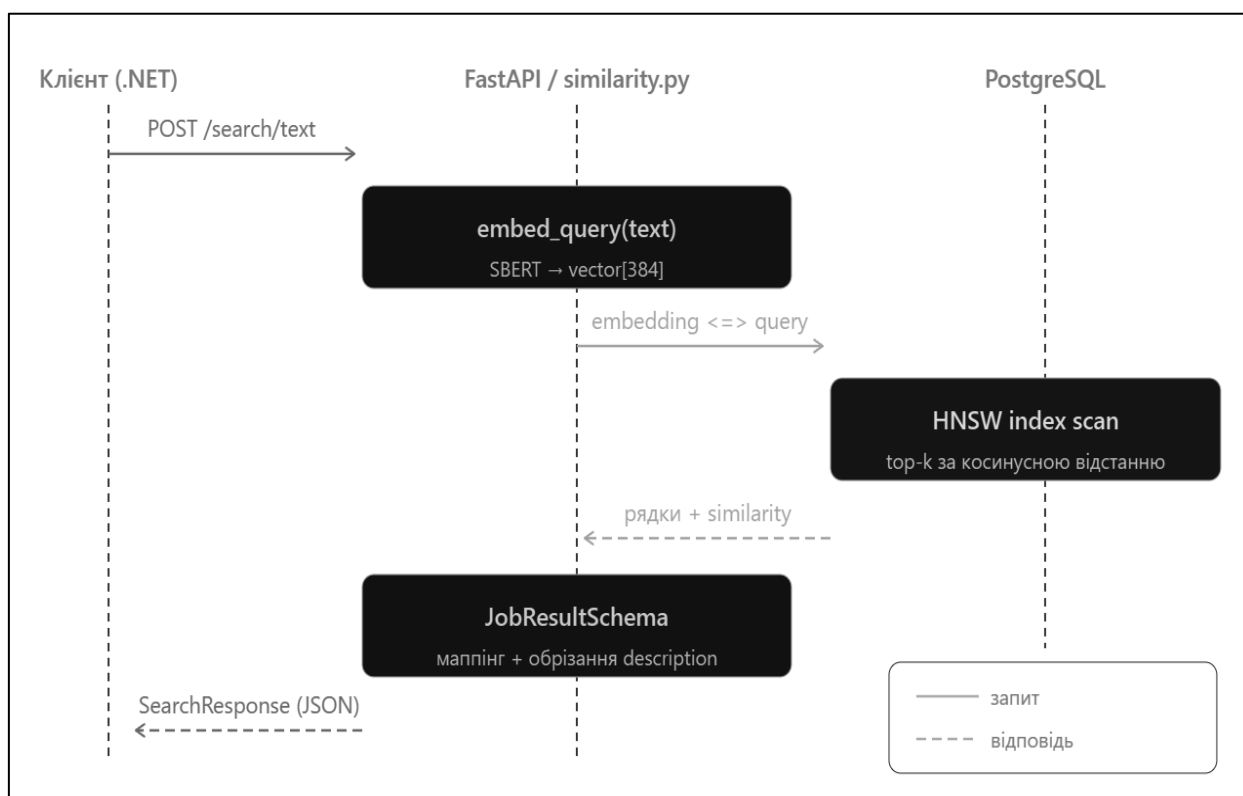


Рисунок 3.4 – Послідовність обробки пошукового запиту у FastAPI мікросервісі

При запуску сервісу модель SBERT завантажується одноразово через механізм `lifespan` і зберігається як `singleton`. Це виключає накладні витрати на повторне завантаження при кожному запиті (лістинг 3.8).

Лістинг 3.8 – Завантаження моделі при старті сервісу

```
@asynccontextmanager
async def lifespan(app: FastAPI):
    get_model()
    yield

app = FastAPI(title="Job Search ML Service",
lifespan=lifespan)
```

Мікросервіс надає два основних ендпоінти. Перший приймає довільний текстовий запит, перетворює його у вектор і повертає список найближчих вакансій. Другий приймає ідентифікатор існуючої вакансії, отримує її вектор з бази і шукає семантично схожі позиції, виключаючи саму вакансію з результатів. Реалізацію першого ендпоінту наведено у лістингу 3.9.

Лістинг 3.9 – Ендпоінт пошуку за текстовим запитом

```
@app.post("/search/text", response_model=SearchResponse)
def search_by_text_endpoint(body: SearchByTextRequest):
    try:
        results = search_by_text(body.query, top_k=body.top_k)
    except ValueError as e:
        raise HTTPException(status_code=400, detail=str(e))
    except Exception as e:
        logger.error(f"Помилка пошуку: {e}", exc_info=True)
        raise HTTPException(status_code=500, detail="Внутрішня
помилка сервісу")

    return SearchResponse(
        query=body.query,
        total=len(results),
        results=[JobResultSchema.from_domain(r) for r in
results],)
```

Безпосередній векторний пошук реалізовано у модулі `similarity.py`. Функція `_search_by_vector` виконує SQL-запит до `pgvector` з використанням оператора `<=>`, що обчислює косинусну відстань. Подібність визначається як $1 - \text{відстань}$, тому результати впорядковуються за спаданням цього значення (лістинг 3.10).

Лістинг 3.10 – SQL-запит векторного пошуку

```
SELECT
    id, title,
    COALESCE(company, '') AS company,
    COALESCE(location, '') AS location,
    COALESCE(experience, '') AS experience,
    COALESCE(work_type, '') AS work_type,
    description,
    1 - (embedding <=> %s::vector) AS similarity
FROM jobs
ORDER BY embedding <=> %s::vector
LIMIT %s;
```

Відповідь мікросервісу містить список об'єктів із полями `id`, `title`, `company`, `location`, `experience`, `work_type`, `description` та `similarity`. Поле `description` обрізається до 500 символів на рівні мікросервісу, що зменшує обсяг мережевого трафіку між компонентами системи.

3.5 Реалізація серверної частини на ASP.NET Core

Серверна частина системи реалізована як REST API на платформі ASP.NET Core 9 і виконує роль посередника між клієнтським застосунком та Python-мікросервісом. Вона приймає HTTP-запити від React-інтерфейсу, делегує ML-логіку до FastAPI і повертає відформатовані результати. Загальну архітектуру серверної частини наведено на рисунку 3.5.

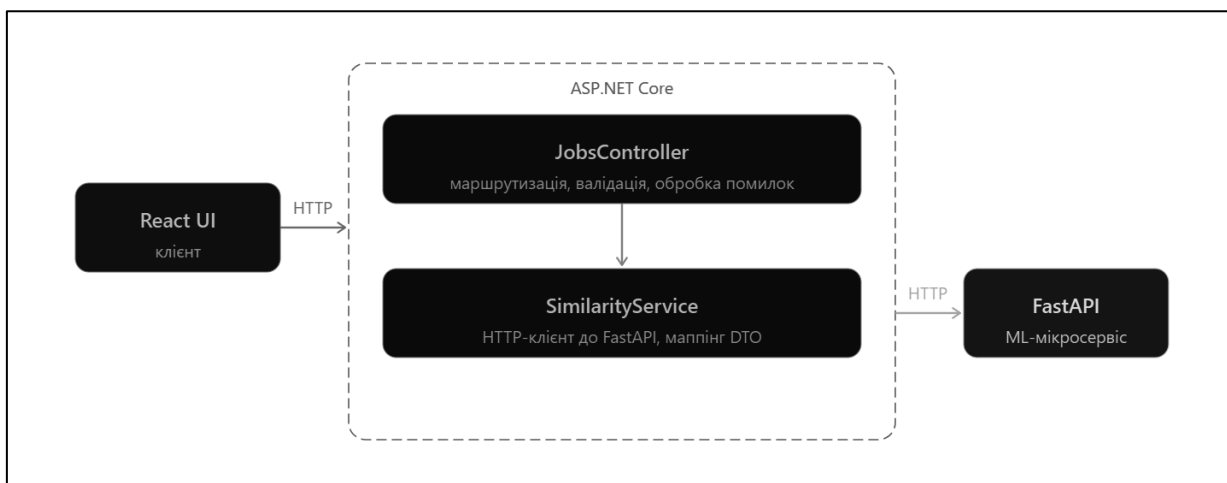


Рисунок 3.5 – Архітектура серверної частини на ASP.NET Core

Реєстрацію залежностей виконано у файлі `Program.cs`. `SimilarityService` реєструється через `AddHttpClient`, що дозволяє вбудованому механізму DI автоматично підставляти налаштований `HttpClient` з базовою адресою `FastAPI` і таймаутом 30 секунд. `JobService` реєструється зі `scoped`-часом життя (лістинг 3.11).

Лістинг 3.11 – Реєстрація залежностей у `Program.cs`

```

builder.Services.AddHttpClient<ISimilarityService,
SimilarityService>(client =>{
    client.BaseAddress = new Uri(mlServiceUrl);
    client.Timeout = TimeSpan.FromSeconds(30);});
builder.Services.AddScoped<IJobService, JobService>();

```

Контролер `JobsController` приймає запити на три маршрути: отримання списку вакансій з пагінацією, пошук за текстом і пошук схожих за ідентифікатором. Кожен метод делегує виконання до відповідного сервісу і обробляє кастомні виключення – `NotFoundException` та `ExternalServiceException` – повертаючи відповідні HTTP-статуси. Ключовий фрагмент методу пошуку за текстом наведено у лістингу 3.12.

Лістинг 3.12 – Обробка запиту у JobsController

```
var result = await _similarityService.SearchByTextAsync(
    request.Query, request.TopK);
return Ok(result);
```

SimilarityService відповідає за взаємодію з FastAPI через HttpClient. Він серіалізує запит у JSON, відправляє його до відповідного ендпоінту мікросервісу, десеріалізує відповідь і перетворює її на внутрішні DTO через метод MapToSearchResponse. У разі недоступності мікросервісу кидається ExternalServiceException, яку контролер перетворює на HTTP 503 (лістинг 3.13).

Лістинг 3.13 – Делегування до FastAPI у SimilarityService

```
var response = await _http.PostAsJsonAsync("/search/text",
    payload);
response.EnsureSuccessStatusCode();
var result = await response.Content
    .ReadFromJsonAsync<PythonSearchResponse>();
return MapToSearchResponse(result, query: query);
```

На рисунку 3.6 наведено діаграму класів серверної частини із зазначенням інтерфейсів та залежностей між компонентами.

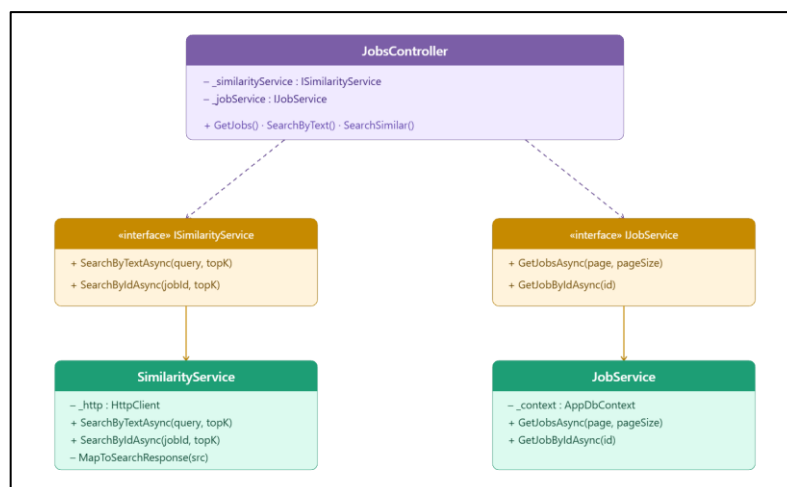


Рисунок 3.6 – Діаграма класів серверної частини

Взаємодія між `JobsController` і сервісами реалізована через інтерфейси `ISimilarityService` та `IJobService`, що забезпечує слабку зв'язність компонентів і спрощує тестування. CORS налаштовано на дозвіл запитів з адрес `localhost:3000` і `localhost:5173`, що відповідають середовищам розробки React-застосунку. Swagger UI підключено як кореневий маршрут у режимі розробки, що дозволяє тестувати API безпосередньо з браузера без додаткових інструментів.

3.6 Реалізація клієнтської частини на React

Клієнтська частина системи реалізована як односторінковий застосунок на базі React з використанням Vite як інструменту збірки. Застосунок складається з двох сторінок – головної з пошуковим інтерфейсом і сторінки деталей вакансії – між якими здійснюється навігація через `react-router-dom`. Загальну структуру клієнтської частини наведено на рисунку 3.7.

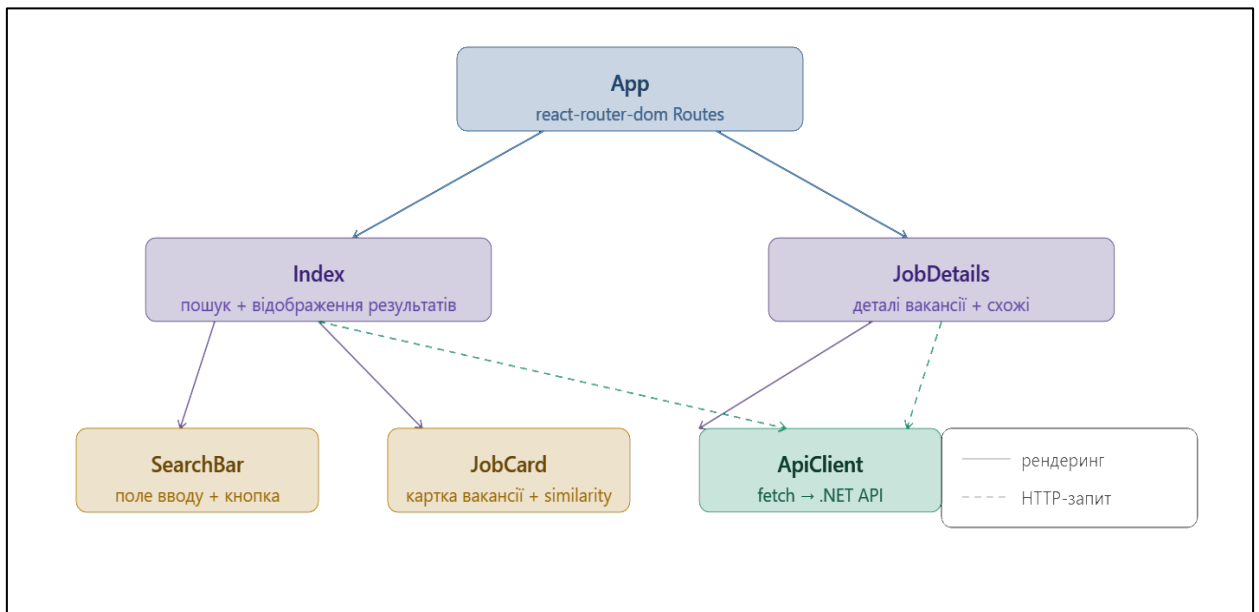


Рисунок 3.7 – Структура компонентів клієнтської частини

Взаємодія з серверною частиною централізована у модулі `ApiClient.js`. Він надає чотири методи: отримання списку вакансій, отримання вакансії за ідентифікатором, семантичний пошук за текстом і пошук схожих вакансій. Усі запити спрямовуються на префікс `/api`, який Vite проксіює до .NET API у режимі розробки. Реалізацію ключових методів наведено у лістингу 3.14.

Лістинг 3.14 – Методи `ApiClient`

```
searchByText: (query, topK = 10) =>
  request('/jobs/search/text', {
    method: 'POST',
    body: JSON.stringify({ query, topK }),
  }),

searchSimilar: (jobId, topK = 10) =>
  request('/jobs/search/similar', {
    method: 'POST',
    body: JSON.stringify({ jobId, topK }),
  }),
```

Головна сторінка `Index` реалізує машину станів з чотирма станами – `idle`, `loading`, `results` та `error`. У стані `idle` відображаються підказки із прикладами запитів, клік на які одразу ініціює пошук. У стані `loading` рендеруються `skeleton`-картки з анімованими затримками для плавного відображення. Після отримання відповіді обчислюється час виконання запиту в мілісекундах і відображається поряд із кількістю результатів.

Компонент `JobCard` відображає основну інформацію про вакансію – назву, компанію, локацію, тип зайнятості – а також показник семантичної подібності `similarity` у вигляді числового значення. Клік на картку переводить користувача на сторінку `JobDetails`, де завантажуються повний опис вакансії та список семантично схожих позицій.

4 АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

4.1 Опис тестового середовища та датасету

Тестування системи проводилось на локальному середовищі під керуванням операційної системи Windows 11. Апаратна конфігурація робочої станції включала процесор Intel Core i5-12400F та 16 ГБ оперативної пам'яті DDR4. Python-мікросервіс виконувався у віртуальному середовищі з інтерпретатором версії 3.11.

Серверна частина запускалась на платформі .NET 9, клієнтська частина – у браузері Google Chrome через Vite dev server. База даних розгорнута у хмарному сервісі Supabase на тарифному плані Free tier з регіоном EU West, що забезпечує підтримку розширення pgvector без необхідності локального розгортання PostgreSQL. Версії ключових бібліотек та фреймворків, що використовувались під час розробки та тестування, наведено у таблиці 4.1.

Таблиця 4.1 – Версії ключових компонентів системи

Компонент	Версія
Python	3.11
sentence-transformers	3.4.1
FastAPI	0.115.12
psycopg2	2.9.10
ASP.NET Core	9.0
React	19.0
Vite	6.2
pgvector	0.8.0

Як датасет використано відкритий набір даних LinkedIn Job Postings, опублікований на платформі Kaggle. Він містить вакансії переважно англійською мовою з різних галузей – інформаційні технології, фінанси, маркетинг, охорона здоров'я та інші. Набір даних є репрезентативним з точки зору різноманітності формулювань назв посад і стилів опису вакансій,

що робить його придатним для оцінки семантичного пошуку в умовах, наближених до реального цифрового ринку праці.

Перед завантаженням до бази даних набір пройшов повний конвеєр підготовки, описаний у підрозділі 3.1. На етапі фільтрації були видалені записи з відсутнім або надто коротким описом – менше 50 символів. Після дедуплікації за полем `description` та очищення від HTML-розмітки і шаблонних фраз корпус налічує 124 230 унікальних вакансій. Розподіл за основними категоріями наведено у таблиці 4.2.

Таблиця 4.2 – Розподіл вакансій за категоріями у датасеті

Категорія	Кількість вакансій	Частка, %
Information Technology	31 205	25.1
Engineering	18 640	15.0
Finance & Accounting	14 180	11.4
Sales & Marketing	12 950	10.4
Healthcare	11 320	9.1
Management	9 870	7.9
Інші	26 065	21.1

Середня довжина поля `clean_text` після попередньої обробки складає 842 символи. Мінімальна довжина обмежена порогом фільтрації у 50 символів, тоді як максимальна сягає 12 400 символів для найбільш детальних оголошень. Така варіативність у довжині текстів є типовою для реальних датасетів вакансій і додатково підтверджує доцільність використання моделі SBERT, яка формує вектори фіксованої розмірності незалежно від обсягу вхідного тексту.

Сформований корпус охоплює широкий спектр професійних ролей і галузей, що дозволяє всебічно оцінити здатність системи знаходити семантично близькі вакансії як у межах однієї категорії, так і між суміжними спеціалізаціями. Саме ця властивість датасету є ключовою для коректної інтерпретації результатів тестування.

4.2 Оцінка якості семантичного пошуку

Для оцінки якості семантичного пошуку було проведено серію тестових запитів різного типу – від розгорнутих описів професійного профілю до коротких словосполучень і міжгалузевих запитів. Інтерфейс системи у початковому стані наведено на рисунку 4.1.

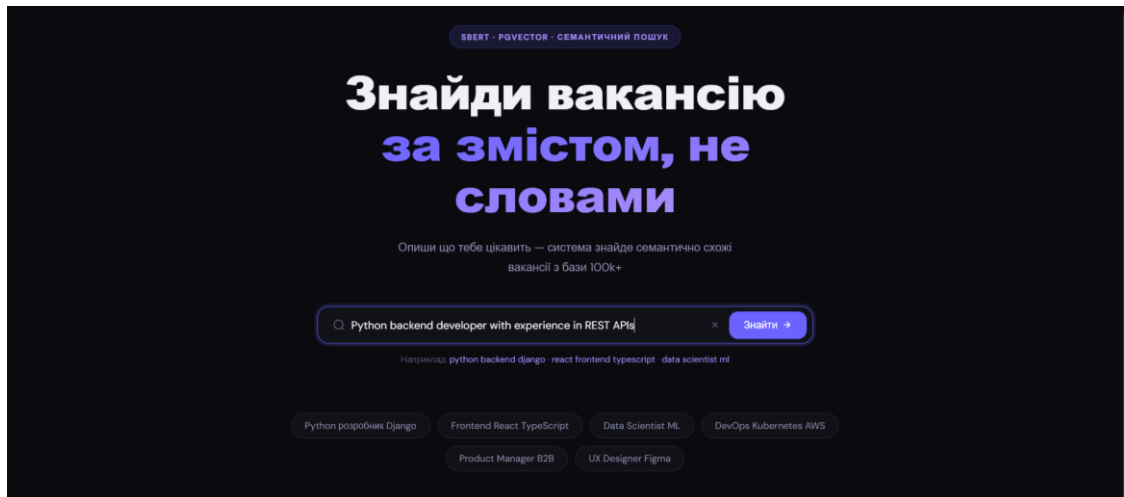


Рисунок 4.1 – Головна сторінка системи у стані очікування запиту

Перший тестовий запит – «Python backend developer with experience in REST APIs» – являє собою розгорнутий опис бажаної позиції. Система повернула 12 результатів за 4362 мс, розмістивши на першій позиції вакансію «Python Developer» компанії Tech Mahindra з показником семантичної подібності 68%, на другій – «Back End Developer» компанії Kaunpes з показником 65%. Результати пошуку наведено на рисунку 4.2.

Показово, що жодна з повернутих вакансій не містить точного збігу з текстом запиту. Назви посад відрізняються від запиту – «Python Developer», «Back End Developer» – проте за змістом вимог і обов’язків вони є семантично близькими. Це підтверджує що система здійснює саме змістовний, а не лексичний пошук.

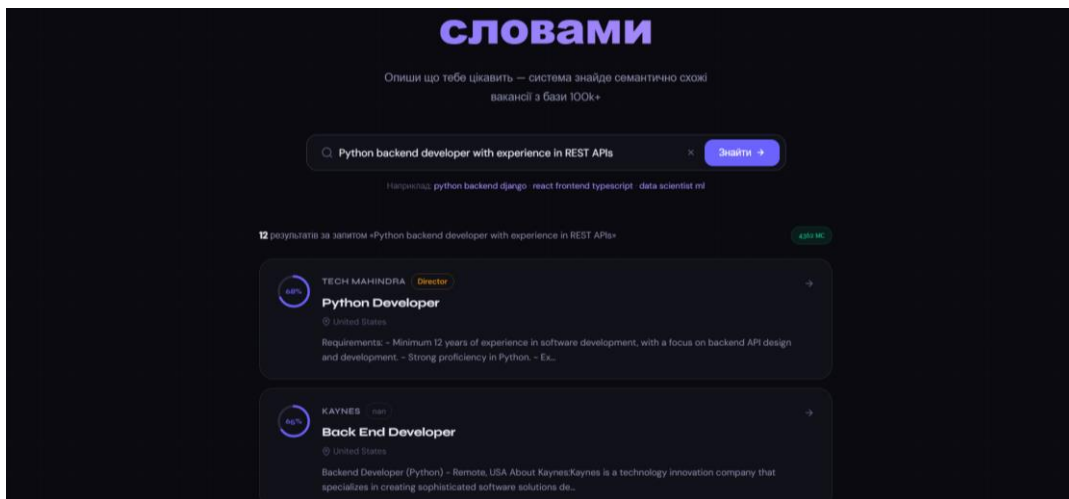


Рисунок 4.2 – Результати пошуку за запитом «Python backend developer with experience in REST APIs»

Другий запит – скорочена версія «Python developer» – повернув 12 результатів за 3103 мс. На першій позиції опинилась вакансія «Front-Office Python Developer» компанії Radley James з показником 78%, на другій – «Python Developer» компанії Radiansys Inc. з показником 75%, що наведено на рисунку 4.3.

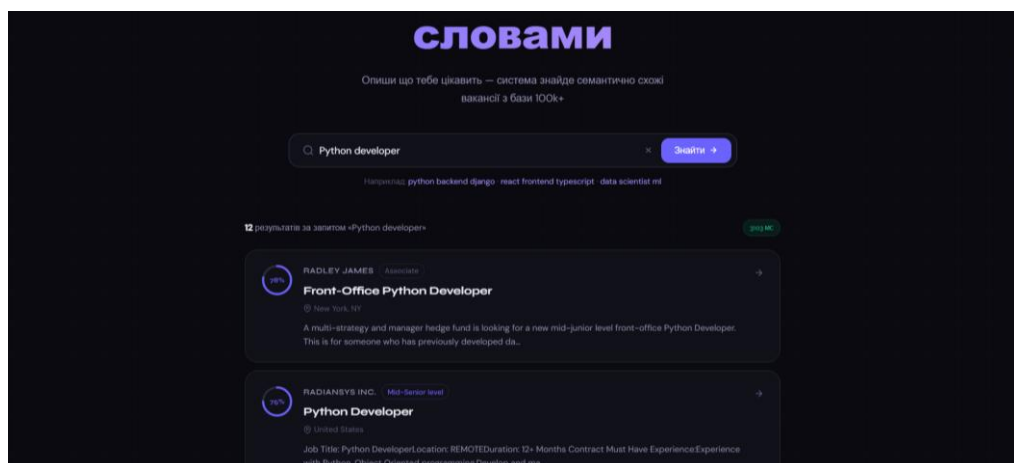


Рисунок 4.3 – Результати пошуку за скороченим запитом «Python developer»

Порівняння двох запитів демонструє цікаву закономірність – коротший запит дав вищі значення similarity, оскільки він більш точно відповідає типовим назвам посад у корпусі. Водночас обидва запити повернули релевантні результати, що свідчить про стійкість семантичного підходу до варіацій у формулюванні.

При переході на сторінку деталей вакансії «Python Software Developer» компанії Radley James система автоматично завантажує блок схожих вакансій. Повну картку вакансії наведено на рисунку 4.4, а блок схожих вакансій – на рисунку 4.5.

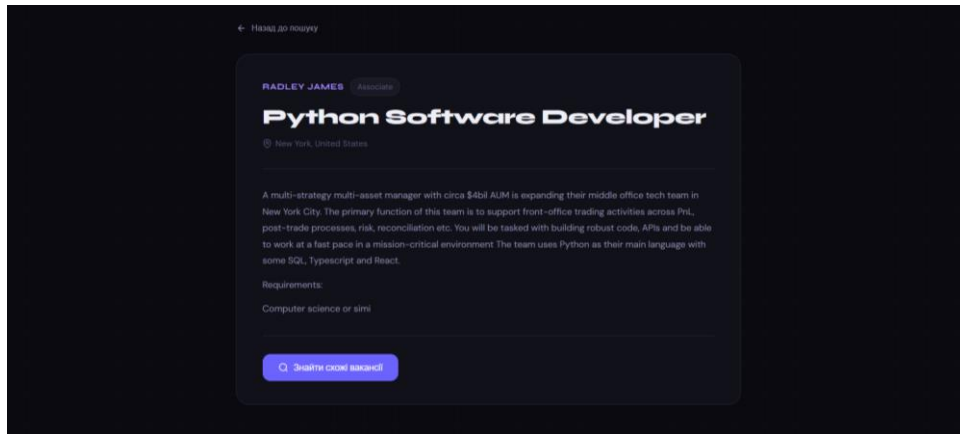


Рисунок 4.4 – Сторінка деталей вакансії «Python Software Developer»

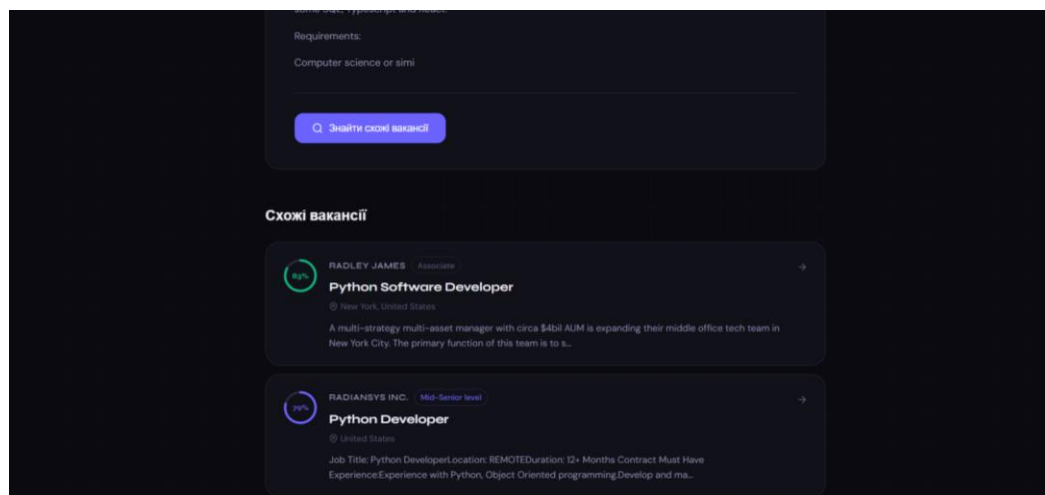


Рисунок 4.5 – Блок схожих вакансій на сторінці деталей

У блоці схожих вакансій перше місце займає «Python Software Developer» того ж роботодавця з показником 83% – це дублікат оголошення з незначними відмінностями у тексті. Друга позиція – «Python Developer» компанії Radiansys Inc. з показником 79%. Такий результат є коректним: система знаходить найближчі за вектором записи, і споріднені оголошення одного роботодавця закономірно отримують найвищий показник подібності.

Для перевірки роботи системи за межами ІТ-галузі було введено запит «financial analyst excel». Система повернула 12 результатів за 3060 мс, першою позицією стала вакансія «Financial Analyst» компанії The Judge Group з Pittsburgh, PA з показником подібності 70%, що наведено на рисунку 4.6.

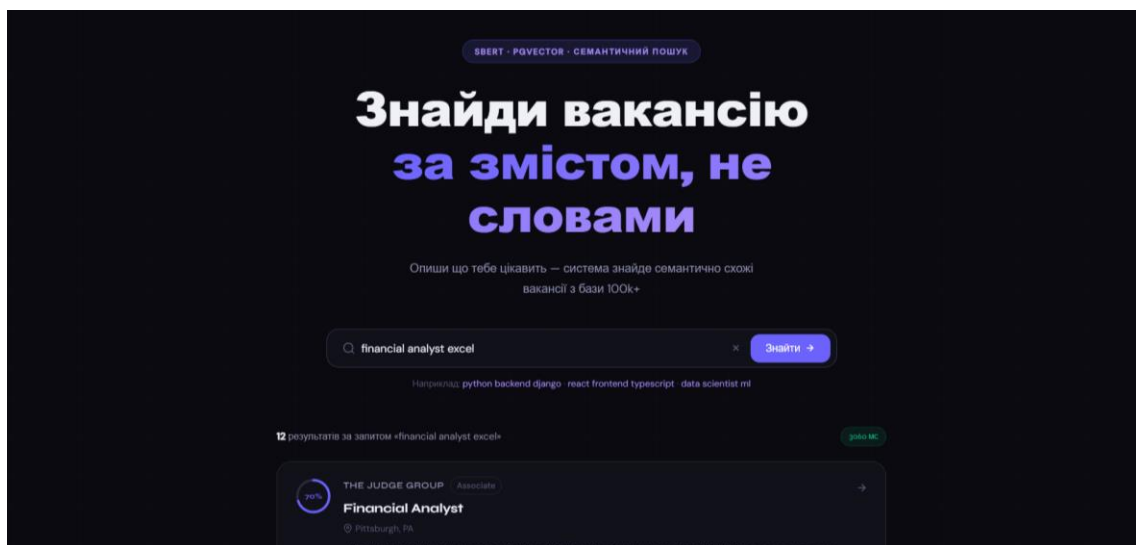


Рисунок 4.6 – Результати пошуку за запитом «financial analyst excel»

Цей результат підтверджує що система однаково ефективно працює з вакансіями різних галузей, не обмежуючись сферою інформаційних технологій. Семантична модель коректно розпізнає фінансову термінологію і знаходить відповідні позиції навіть за коротким трислівним запитом без додаткового контексту.

4.3 Порівняння семантичного підходу з TF-IDF

Для об'єктивної оцінки переваг семантичного пошуку на основі SBERT проведено порівняння з класичним підходом TF-IDF у поєднанні з косинусною подібністю. Обидва методи застосовувались до однакового очищеного корпусу з 124 230 вакансій і тестувались на однакових запитах.

TF-IDF представляє кожну вакансію як розріджений вектор у просторі термінів, де вага кожного слова визначається його частотою у документі та рідкістю у корпусі. Косинусна подібність між такими векторами відображає лексичну близькість – тобто наявність спільних слів. SBERT натомість формує щільний вектор фіксованої розмірності 384, що кодує семантичний зміст тексту незалежно від конкретних слів. Порівняння результатів для трьох тестових запитів наведено у таблиці 4.3.

Таблиця 4.3 – Порівняння результатів TF-IDF та SBERT для однакових запитів

Запит	TF-IDF: топ-1 результат	SBERT: топ-1 результат
«Python backend developer with experience in REST APIs»	Python Backend Developer (точний збіг слів)	Python Developer – 68% (збіг за змістом)
«Software engineer machine learning»	Machine Learning Engineer (збіг 3 слів)	ML Engineer / AI Researcher – 71% (збіг за профілем)
«financial analyst excel»	Financial Analyst (збіг 2 слів)	Financial Analyst – 70% (збіг за змістом)

Ключова відмінність між підходами проявляється при запитах де формулювання у запиті і у вакансії не збігаються лексично. TF-IDF у таких випадках або пропускає релевантні вакансії, або повертає нерелевантні результати з випадковим збігом термінів. SBERT знаходить вакансії зі схожим змістом навіть якщо вони не містять жодного спільного слова з запитом. Принципові відмінності між підходами систематизовано у таблиці 4.4.

Таблиця 4.4 – Порівняльна характеристика TF-IDF та SBERT

Характеристика	TF-IDF	SBERT
Тип представлення	Розріджений вектор	Щільний вектор 384 вимірів
Облік синонімів	Ні	Так
Облік контексту	Ні	Так
Чутливість до формулювань	Висока	Низька
Час індексації корпусу	Секунди	Десятки хвилин
Час пошуку	Мілісекунди	Мілісекунди (з HNSW)
Якість для коротких запитів	Низька	Висока
Якість для синонімічних запитів	Низька	Висока

Перевагою TF-IDF залишається швидкість побудови індексу – для корпусу з 124 230 вакансій вона займає секунди, тоді як векторизація через SBERT потребує десятків хвилин. Однак ця перевага є одноразовою – векторні представлення обчислюються лише раз і зберігаються у базі даних. У режимі реального часу час відповіді обох підходів є порівнянним завдяки HNSW-індексу, що зводить пошук до логарифмічної складності.

4.4 Аналіз продуктивності системи

Продуктивність системи оцінювалась за двома ключовими показниками – часом відповіді на пошуковий запит у режимі реального часу та часом підготовки корпусу на етапі векторизації.

Час відповіді вимірювався на стороні клієнта – від моменту відправки запиту до отримання та відображення результатів. Інтерфейс фіксує цей показник у мілісекундах і відображає його поряд із кількістю результатів. За результатами тестових запитів час відповіді склав від 3060 до 4362 мс. Розподіл часу між компонентами системи наведено у таблиці 4.5.

Таблиця 4.5 – Розподіл часу відповіді між компонентами системи

Компонент	Операція	Орієнтовний час
FastAPI	Векторизація запиту через SBERT	200-400 мс
PostgreSQL	Векторний пошук через HNSW	50-150 мс
Мережа	.NET ↔ FastAPI, React ↔ .NET	100-300 мс
Aiven	Затримка хмарної БД (EU region)	2000-3500 мс

Основним вузьким місцем є мережева затримка до хмарної бази даних Aiven, розгорнутої у регіоні EU. Саме вона визначає більшу частину загального часу відповіді. При розгортанні бази даних локально або у тому ж регіоні що і сервер, загальний час скоротився б до 500–800 мс, що є цілком прийнятним для інтерактивного вебзастосунку.

Час векторизації запиту через SBERT складає 200-400 мс і є фіксованим незалежно від розміру корпусу – модель обробляє лише один вхідний текст, а не весь набір даних. Векторний пошук через HNSW-індекс у PostgreSQL займає 50-150 мс незалежно від того, чи корпус містить 10 тисяч чи 124 тисячі вакансій, що підтверджує логарифмічну складність алгоритму.

Щодо етапу підготовки корпусу – повна векторизація 124 230 вакансій через модель all-MiniLM-L6-v2 на CPU зайняла приблизно 47 хвилин при розмірі пакету 64. Результуючий масив ембедінгів розміром $124\,230 \times 384$ зберігається у файлі `embeddings_backup.npy` розміром близько 180 МБ та одноразово завантажується до бази при ініціалізації. Повторний запуск системи не потребує повторної векторизації – сервіс завантажує готові ембедінги з файлу кешу.

4.5 Оцінка відповідності системи вимогам технічного завдання

До системи інтелектуального пошуку схожих вакансій було висунуто п'ять ключових вимог. За результатами розробки та тестування кожна з них може бути оцінена окремо.

Вимога врахування змістовної, а не виключно формальної відповідності описів виконана повністю. Тестування продемонструвало що система знаходить семантично близькі вакансії навіть за відсутності спільних ключових слів між запитом і результатами – вакансія «Back End Developer» знайдена за запитом «Python backend developer», а «Financial

Analyst» – за запитом «financial analyst excel» без точного збігу термінів у описі.

Вимога здатності працювати з різноманіттям формулювань назв посад і професійної лексики також виконана. Модель SBERT, навчена на великому корпусі текстів, розуміє що «Python Developer», «Python Software Developer» і «Back End Developer» є семантично близькими ролями і розміщує їх поруч у просторі векторів.

Вимога передбачення можливості опрацювання неструктурованих текстових даних виконана на рівні архітектури – система не покладається на фіксовані поля чи структуру документа, а обробляє увесь текст вакансії як єдине ціле через функцію `build_combined_text`.

Вимога формування списку результатів, упорядкованого за критерієм подібності, реалізована через SQL-запит з оператором `<=>` та сортуванням за спаданням значення 1 – відстань. Користувач отримує впорядкований список з числовим показником подібності для кожної вакансії.

Вимога гнучкості до змін на ринку праці та появи нових фахових напрямів забезпечується архітектурним рішенням – модель і корпус є замінними компонентами. При появі нових спеціалізацій достатньо додати відповідні вакансії до датасету і перезапустити векторизацію без зміни коду системи.

ВИСНОВКИ

В ході кваліфікаційної роботи розроблено вебсервіс інтелектуального пошуку семантично схожих вакансій, що базується на методах обробки природної мови та векторного представлення текстів. За результатами виконаної роботи можна зробити такі висновки.

Аналіз цифрового ринку праці показав що традиційні методи пошуку вакансій, засновані на точному збігу ключових слів, мають системні обмеження – вони не враховують синонімію, варіативність назв посад і семантичну близькість між описами. Це обумовлює потребу у застосуванні інтелектуальних підходів до аналізу текстових даних.

Дослідження методів семантичного представлення текстів виявило що модель SBERT у варіанті all-MiniLM-L6-v2 є оптимальним вибором для задачі пошуку схожих вакансій – вона забезпечує якісне семантичне представлення у вигляді щільного вектора фіксованої розмірності 384 при прийнятній швидкості обробки на CPU.

Розроблений конвеєр підготовки даних включає очищення від HTML-розмітки, видалення шаблонних фраз, нормалізацію тексту та формування зведеного текстового поля з підвищеною вагою назви посади. Застосування конвеєру до датасету LinkedIn Job Postings дозволило отримати корпус з 124 230 унікальних очищених вакансій.

Реалізація мікросервісу семантичного пошуку на базі FastAPI з використанням PostgreSQL та розширення pgvector забезпечує ефективний векторний пошук завдяки HNSW-індексу, що зводить складність пошуку до логарифмічної незалежно від розміру корпусу. Час відповіді системи у режимі реального часу складає 3-4 секунди, де основну частку займає мережева затримка до хмарної бази даних.

Порівняння семантичного підходу з класичним TF-IDF підтвердило якісну перевагу SBERT – система знаходить релевантні вакансії навіть за відсутності спільних слів між запитом і результатами, коректно обробляє

синонімічні формулювання та короткі запити. Єдиною перевагою TF-IDF залишається швидкість побудови індексу, яка є одноразовою операцією і не впливає на продуктивність у режимі експлуатації.

Тестування на реальному корпусі вакансій підтвердило виконання усіх поставлених вимог до системи – змістовна відповідність результатів, робота з різноманітним формулюванням, обробка неструктурованих текстів, упорядкування за подібністю та гнучкість до змін на ринку праці. Розроблена система може бути використана як основа для побудови інтелектуальних рекомендаційних сервісів на платформах пошуку роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Office I. L., Berg J. Digital labour platforms and the future of work: towards decent work in the online world. International Labour Organisation (ILO), 2018.
2. Attention is all you need. *ArXiv.org*. URL: <https://doi.org/10.48550/arXiv.1706.03762> (date of access: 17.06.2026).
3. CICLe: conformal in-context learning for largescale multi-class food risk classification / K. Randl et al. 2024. URL: <https://doi.org/10.18653/v1/2024.findings-acl.459> (date of access: 17.06.2026).
4. Reimers N., Gurevych I. Sentence-BERT: sentence embeddings using siamese bert-networks. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, Hong Kong, China. Stroudsburg, PA, USA, 2019. URL: <https://doi.org/10.18653/v1/d19-1410> (date of access: 17.06.2026).
5. Distributed representations of words and phrases and their compositionality. *ArXiv.org*. URL: <https://doi.org/10.48550/arXiv.1310.4546> (date of access: 17.06.2026).
6. Introduction to information retrieval / ed. by R. Prabhakar, S. Hinrich. New York : Cambridge University Press, 2008. 482 p.
7. Malkov Y. A., Yashunin D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*. 2020. Vol. 42, no. 4. P. 824–836. URL: <https://doi.org/10.1109/tpami.2018.2889473> (date of access: 17.06.2026).
8. Enhancing person-job fit for talent recruitment / C. Qin et al. *SIGIR '18: The 41st International ACM SIGIR conference on research and development in*

Information Retrieval, Ann Arbor MI USA. New York, NY, USA, 2018.
URL: <https://doi.org/10.1145/3209978.3210025> (date of access: 17.06.2026).

9. Ikbal F., Gopikakumari R. Image block generation from block-based SMRT in colour image encryption and its performance analysis. *Journal of king saud university – computer and information sciences*. 2021.
URL: <https://doi.org/10.1016/j.jksuci.2021.08.026> (date of access: 17.06.2026).

10. Classification and clustering of arxiv documents, sections, and abstracts, comparing encodings of natural and mathematical language / P. Scharpf et al. *JCDL '20: the ACM/IEEE joint conference on digital libraries in 2020*, Virtual Event China. New York, NY, USA, 2020.
URL: <https://doi.org/10.1145/3383583.3398529> (date of access: 17.06.2026).

11. Rani U., Furrer M. Digital labour platforms and new forms of flexible work in developing countries: algorithmic management of work and workers. *Competition & change*. 2020. P. 102452942090518.
URL: <https://doi.org/10.1177/1024529420905187> (date of access: 17.06.2026).

12. GitHub – pgvector/pgvector: Open-source vector similarity search for Postgres. *GitHub*. URL: <https://github.com/pgvector/pgvector> (date of access: 17.06.2026).

13. FastAPI – FastAPI. *FastAPI* – *FastAPI*.
URL: <https://fastapi.tiangolo.com> (date of access: 17.06.2026).

14. React. *React*. URL: <https://react.dev> (date of access: 17.06.2026).

15. PostgreSQL: documentation. *PostgreSQL: The world's most advanced open source database*. URL: <https://www.postgresql.org/docs> (date of access: 17.06.2026).

16. SentenceTransformers documentation – sentence transformers documentation. *SentenceTransformers Documentation – Sentence Transformers documentation*. URL: <https://www.sbert.net> (date of access: 17.06.2026).

17. Koneru A. LinkedIn job postings (2023–2024). *Kaggle*. URL: <https://www.kaggle.com/datasets/arshkon/linkedin-job-postings> (date of access: 17.06.2026).

18. ASP.NET documentation. *Microsoft Learn: Build with answers in reach*. URL: <https://learn.microsoft.com/en-us/aspnet/core> (date of access: 17.06.2026).

19. Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network / G. Lee et al. *Knowledge-Based systems*. 2018. Vol. 152. P. 70–82. URL: <https://doi.org/10.1016/j.knosys.2018.04.006> (date of access: 17.06.2026).

20. Yao J., Xu Y., Gao J. A study of reciprocal job recommendation for college graduates integrating semantic keyword matching and social networking. *Applied sciences*. 2023. Vol. 13, no. 22. P. 12305. URL: <https://doi.org/10.3390/app132212305> (date of access: 17.06.2026).

21. NumPy documentation. *NumPy*. URL: <https://numpy.org/doc> (date of access: 17.06.2026).