Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ЗАСТОСУВАННЯ НЕЙРОМЕРЕЖЕВИХ ЗАСОБІВ ДЛЯ АНАЛІЗУ ТЕКСТУ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-23-2
Дебре В.С.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ___________ Кобилін О.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри ______________
(підпис)
«____»______________ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Дебре Віктору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження технології застосування нейромережевих засобів для аналізу тексту

затверджена наказом по університету від 25 листопада 2024 року № 1246Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 2024 р.

3. Вихідні дані до роботи теоретичні відомості про методи оцінки навичок кандидатів з використанням семантичних моделей, вибірка транскрипцій інтерв'ю для навчання та валідації результатів, перелік використаних програмних засобів, огляд сучасних методіваналізу тексте, механізми уваги та адаптивна комбінація даних.

4. Перелік питань, що потрібно опрацювати в роботі

1. Моделі інтеграції додаткових даних у процес аналізу тексту.
2. Математичні моделі комбінування різних аспектів текстового аналізу.
3. Можливості комбінування бази знань та текстових відповідей для покращення точності аналізу.
4. Огляд методів аналізу тексту з використанням мовних моделей.
5. Методи генерації семантичних ознак для текстового аналізу.
6. Огляд методів роботи з текстовими даними у контексті автоматизованої оцінки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) перелік моделей оброблення, схем, таблиць, діаграм, комп’ютерних ілюстрацій: актуальність проблеми автоматизованого аналізу тексту, постановка задачі, схеми роботи алгоритмів обробки тексту, вхідні тестові дані, порівняння результатів автоматизованої оцінки з експертними оцінками, діаграми кореляції між оцінками, таблиці з результатами тестування.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	25.11.2024	
2	Аналіз завдання, підбір літератури	26.11.24-02.12.24	
3	Аналіз літератури з досліджуваної проблеми	06.12.24-10.12.24	
4	Аналіз технічних засобів	11.12.24-14.12.24	
5	Розробка комп’ютерної моделі	14.12.24-17.12.24	
6	Програмна реалізація	16.12.24-19.12.24	
7	Оформлення пояснювальної записки	20.12.24-29.12.24	
8	Перевірка на плагіат	1.01.2024	
9	Рецензування	05.01.2024	
10	Підготовка презентації та доповіді	10.01.2024	
11	Занесення роботи в електронний архів	12.01.2025	
12	Попередній захист кваліфікаційної роботи	13.01.2025	

Дата видачі завдання 25 листопада 2024 р.

Студент ____________________________________
(підпис)

Керівник роботи ___________________________ проф. Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 73 с., 2 табл., 13 рис., 0 дод., 44 джерела.

МОВНІ МОДЕЛІ, СЕМАНТИЧНИЙ АНАЛІЗ, РОЗПІЗНАВАННЯ МОВИ, ТРАНСКРИПЦІЯ АУДІО, ТОКЕНІЗАЦІЯ, АЛГОРИТМИ ОБРОБКИ ТЕКСТУ.

Об'єктом дослідження є аналіз транскрипцій інтерв'ю з використанням сучасних мовних моделей та локальних баз знань.

Метою дослідження є розробка методів та алгоритмів, що базуються на використанні великих мовних моделей і векторного представлення даних для автоматичної оцінки відповідності кандидатів заданим критеріям.

У дослідженні використано методи аналізу мовних моделей, оптимізації векторних представлень, а також обчислювальні методи для порівняння результатів оцінки з навчальними вибірками. Розроблено підхід до інтеграції мовних моделей із локальними базами знань для покращення точності оцінювання кандидатів.

У результаті створено та досліджено алгоритм оброблення текстових транскрипцій, що включає токенізацію, векторизацію, контекстний аналіз, а також реалізовано програмну модель для оцінки компетенцій кандидатів.

LANGUAGE MODELS, SEMANTIC ANALYSIS, LANGUAGE RECOGNITION, AUDIO TRANSCRIPTION, TOKENIZATION, TEXT PROCESSING ALGORITHMS.

The object of the research is the analysis of interview transcriptions using modern language models and local knowledge bases.

The purpose of the research is the development of methods and algorithms based on the use of large language models and vector representation of data for automatic assessment of candidates' compliance with given criteria.

The research uses methods of analyzing language models, optimization of vector representations, as well as computational methods for comparing the evaluation results with training samples. An approach has been developed to integrate language models with local knowledge bases to improve the accuracy of candidate evaluation.

As a result, an algorithm for processing text transcriptions was created and researched, including tokenization, vectorization, contextual analysis, and a software model was implemented for evaluating the candidates' competencies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP – Natural Language Processing (обробка природної мови): галузь штучного інтелекту, яка займається взаємодією між комп’ютерами та людською мовою

LLM – Large Language Models (великі мовні моделі): нейромережеві моделі, здатні обробляти, аналізувати та генерувати текст на основі великих корпусів даних

GPT – Generative Pre-trained Transformer: трансформерна мовна модель, яка використовує механізми самоуваги для генерації тексту

BERT – Bidirectional Encoder Representations from Transformers: трансформерна модель, яка враховує контекст слова як зліва, так і справа для аналізу тексту

RLHF – Reinforcement Learning from Human Feedback: метод навчання мовних моделей із використанням зворотного зв’язку від людей для покращення якості відповідей

Embeddings (векторне представлення) – математичні вектори, які використовуються для кодування семантичної інформації про слова чи фрази

Трансформер – архітектура нейронної мережі, що використовує механізм самоуваги для аналізу залежностей між словами в тексті

Мультиголовна самоувага – механізм, який дозволяє моделі одночасно аналізувати різні аспекти тексту через паралельні обчислення уваги

Feed-forward (об’ємні шари) – шари нейронної мережі, які застосовують нелінійні перетворення для покращення обробки тексту

Fine-tuning (тонке налаштування) – додаткове навчання попередньо тренованої моделі на спеціалізованих наборах даних для виконання конкретних завдань

Самоувага (Self-attention) – механізм у трансформерах, що обчислює важливість кожного слова в тексті відносно інших

Синтаксичний аналіз – процес виявлення граматичних відношень між словами у реченні

Семантичний аналіз – метод аналізу тексту для визначення значення слів у контексті

Лексичний аналіз – процес розбиття тексту на окремі слова чи фрази для подальшої обробки

Моделювання тем – автоматизований метод, який виділяє основні теми тексту шляхом групування схожих за змістом слів чи фраз

Порівняльний аналіз – метод оцінки текстів шляхом порівняння відповіді кандидата з еталонними зразками

GPT-4 – четверта версія моделі GPT, яка відрізняється вищою продуктивністю, підтримкою більших контекстів та кращим розумінням тексту

Контекст – інформація, яка враховується для правильного аналізу чи генерації тексту, включаючи попередній текст

Тональність – емоційне забарвлення відповіді, яке визначається через аналіз тексту

Інтерв'ю-транскрипція – текстовий формат усних відповідей, отриманий через автоматичне розпізнавання мови

ВСТУП

Аналіз тексту з використанням сучасних мовних моделей набуває дедалі більшої популярності у різних сферах, включаючи рекрутинг, освіту, маркетинг і науку. Основними напрямами в цьому процесі є синтаксичний, семантичний аналіз та моделювання тем.

Синтаксичний аналіз полягає у виявленні граматичних відношень між словами в тексті. Це дозволяє зрозуміти структуру речення, логічність викладу думок та якість аргументації. Семантичний аналіз спрямований на визначення контекстуальних значень слів, що особливо важливо для розуміння технічних термінів або слів із багатозначними значеннями.

Моделювання тем дозволяє виділити основні концепції тексту, спрощуючи оцінювання його змісту та релевантності.

Особливу роль у цьому процесі відіграють трансформерні моделі, такі як GPT або BERT, які завдяки своїй архітектурі здатні обробляти великі обсяги тексту, зберігати контекст та аналізувати довгострокові залежності між словами [1-3]. Вони дозволяють автоматизувати процеси оцінки навичок кандидатів, проводити аналіз транскрипцій інтерв'ю та виявляти ключові компетенції.

Актуальність дослідження зумовлена потребою в автоматизації аналізу текстових даних, зокрема у сфері підбору персоналу. Використання нейромережевих моделей значно спрощує обробку транскрипцій інтерв'ю, роблячи процес оцінювання швидшим, точнішим і менш суб'єктивним. Це відкриває нові можливості для масштабування таких процесів у великих організаціях, де аналіз текстових даних є критично важливим.

1 АНАЛІЗ СУЧАСНИХ НЕЙРОМЕРЕЖЕВИХ ЗАСОБІВ ДЛЯ ОБРОБКИ ТЕКСТІВ

1.1 Підходи до автоматизованого оброблення тексту

Аналіз тексту за допомогою сучасних мовних моделей стає надзвичайно ефективним у сфері оцінки кандидатів, особливо коли йдеться про транскрипції інтерв'ю. Насамперед, для обробки природної мови (NLP) використовується лексичний та семантичний аналіз [4]. Лексичний аналіз дозволяє розбивати текст на окремі слова та фрази, що дає можливість швидко виявити специфічні навички кандидата, наприклад, знання мов програмування або певних технологій. Семантичний аналіз, у свою чергу, допомагає зрозуміти контекст цих слів у конкретній ситуації, забезпечуючи правильне трактування технічних термінів та дозволяючи краще оцінити розуміння кандидатом спеціалізованих понять [5].

Ще один ефективний підхід – це аналіз емоцій, який дозволяє моделі визначати тональність відповідей кандидата, оцінюючи емоційний настрій його відповідей. Цей метод особливо корисний для виявлення реакції кандидата на запитання про його мотивацію, стресостійкість або готовність до складних завдань. На основі цього аналізу можна скласти глибший портрет кандидата, оцінюючи його позитивність, впевненість чи сумнівність у відповідях.

Для узагальнення та структурування інформації широко використовується моделювання тем, яке дозволяє визначити основні теми, що обговорюються під час інтерв'ю. Це полегшує виявлення ключових компетенцій, необхідних для позиції, таких як технічні знання, знання інструментів або процесів, без необхідності проходити весь текст вручну.

Високоефективними також є трансформерні моделі, такі як BERT або GPT, які спеціалізуються на аналізі великих обсягів тексту та складних зв'язків між словами. Вони дозволяють зберегти контекст відповідей кандидата на кожному етапі розмови, що особливо важливо для оцінки його глибоких знань, а також для виявлення потенційних невідповідностей або суперечностей у відповідях.

Порівняння текстів є ще одним корисним інструментом, за допомогою якого можна оцінити рівень знань кандидата, порівнюючи його відповіді з еталонними зразками або відповідями фахівців. Такий аналіз дозволяє встановити, наскільки відповіді кандидата відповідають вимогам до певної позиції.

Нарешті, важливим інструментом є аналіз синтаксичних структур, що дозволяє розглядати взаємозв'язки між словами в кожній відповіді. Це допомагає оцінити логічність і послідовність думок кандидата, що вказує на його вміння аргументувати свої відповіді й формулювати чіткі та послідовні думки.

Розвиток мовних моделей значно розширює можливості автоматизованої обробки тексту, зокрема у сфері рекрутингу та управління персоналом. На відміну від традиційних методів, які часто обмежуються лише перевіркою ключових слів або простих відповідей, сучасні моделі, такі як GPT [5], здатні забезпечувати глибокий аналіз тексту, враховуючи лексичні, синтаксичні та семантичні аспекти відповідей кандидатів. Ця здатність моделей «розуміти» значення слів у контексті сприяє кращій оцінці навичок, враховуючи особливості мови кандидата, рівень його компетенції у відповідній сфері та його особистісні якості [6].

Крім того, автоматизація аналізу інтерв'ю відкриває можливість масштабованої оцінки кандидатів за допомогою транскрипцій їхніх відповідей, що знижує навантаження на HR-відділи. Використання великих мовних моделей у цьому процесі дозволяє швидко та об'єктивно оцінювати не лише технічні знання, але й міжособистісні навички, а також здатність до вирішення проблем та адаптивність. Таким чином, впровадження таких інструментів у процес підбору персоналу стає ефективною стратегією для компаній, які прагнуть швидко та якісно знаходити таланти відповідно до високих сучасних вимог.

Зауважимо, що комбінація розглянутих підходів дає можливість створити детальну та об'єктивну оцінку навичок і компетенцій кандидата на основі його відповідей в інтерв'ю.

1.2 Особливості архітектури та навчання великої мовної моделі

Моделі трансформерів LLM (Large Language Models) – це вид архітектури глибокого навчання, яка використовує механізм самоуваги (self-attention) для аналізу тексту та генерації відповідей. Вперше їх запропонували дослідники Google в 2017 році у статті «Attention Is All You Need». Трансформери стали основою для багатьох сучасних моделей обробки природної мови завдяки їхній здатності ефективно працювати з великими обсягами текстових даних [7].

Архітектура трансформера складаються з двох основних компонентів: енкодера та декодера. Енкодер відповідає за обробку вхідних даних та витяг інформації з них, тоді як декодер використовується для генерації вихідних даних на основі контексту, отриманого з енкодера. У випадку LLM зазвичай використовується тільки енкодер, який складається з багатьох шарів самоуваги та об'ємних шарів feed-forward [8]. Кожен шар застосовує нормалізацію, механізм самоуваги та нелінійні перетворення для збагачення представлення вхідного тексту.

Механізм самоуваги дозволяє моделі оцінювати важливість кожного слова у тексті щодо інших слів. Це досягається за допомогою обчислення ваг між словами на основі їх контексту. У процесі самоуваги кожне слово представляється у вигляді векторів ключів (keys), запитів (queries) і значень (values). Модель обчислює матрицю уваги, яка вказує, наскільки кожне слово повинно враховувати інші слова при генерації відповіді. Це дозволяє трансформерам ефективно вловлювати довгострокові залежності в тексті.

Мультиголовна самоувага, щоб модель могла захоплювати різні аспекти тексту одночасно, використовують механізм мультиголовної самоуваги. Це означає, що процес самоуваги виконується паралельно кілька разів, кожного разу з різними параметрами для ключів, запитів і значень. Результати цих множинних обчислень об'єднуються, що дозволяє моделі мати більш глибоке розуміння тексту [8].

Нормалізація та об'ємні шари покращує стабільність і швидкість навчання, зменшуючи проблему затухання або вибуху градієнтів. Об'ємні шари використовують двошарову нейронну мережу для обробки виходу після самоуваги, додаючи більше нелінійності та потужності моделі. Це дозволяє збагачувати представлення тексту на кожному шарі.

Навчання великих мовних моделей для навчання LLM використовуються величезні корпуси тексту, що містять мільярди прикладів. Модель навчається прогнозувати наступне слово або заповнювати пропуски в тексті, використовуючи контекст попередніх слів [9]. Навчання проходить на графічних процесорах з використанням методів оптимізації, таких як Adam, що допомагають зменшувати похибки моделі.

Інструменти генерації тексту після навчання LLM може використовуватися для різних задач, таких як автоматична генерація тексту, відповідь на питання, переклад, тощо. Для генерації тексту модель вибирає наступне слово на основі ймовірностей, розрахованих під час проходження через всі шари. Вибір може включати температурне регулювання або семплювання для додавання варіативності в відповіді.

Регуляризація та перенавчання: LLM зазвичай попередньо навчають на загальних текстах, а потім адаптуються для конкретних задач шляхом додаткового навчання на спеціалізованих датасетах.

Регуляризація, така як дропаут або техніка зупинки навчання, застосовується для уникнення перенавчання.

Моделі LLM демонструють відмінні результати в багатьох задачах обробки тексту, але мають обмеження. Вони можуть генерувати невірну або упереджену інформацію, особливо якщо навчальні дані були недосконалими. Однак, їх можливість розуміти контекст та адаптуватись до різних завдань робить їх ключовими інструментами для сучасної обробки природної мови.

Враховуючи складність мовних моделей, важливим аспектом їх використання є розуміння того, як вони обробляють інформацію. Для забезпечення надійності, моделі потребують ретельного навчання на великих і якісних наборах даних, що включають різні стилі, теми та контексти.

Це дозволяє їм покращувати точність аналізу та адаптацію до складних завдань, таких як обробка транскрипцій інтерв'ю, де важливими є не лише зміст, але й тональність та емоційне забарвлення відповідей. Застосовуючи багаторівневий підхід до аналізу, моделі можуть глибше розуміти різні аспекти мови, що підвищує їхню ефективність у реальних умовах.

Однією з найпопулярніших і широко використовуваних моделей на сьогодні є ChatGPT, яка стала відомою завдяки своїй здатності вести природні та інформативні діалоги з користувачами. Ця модель, заснована на архітектурі GPT-4, демонструє високий рівень розуміння контексту, що дозволяє їй надавати детальні відповіді на запитання, вирішувати різноманітні завдання з обробки тексту та аналізу інформації, а також адаптуватися до потреб користувача [10]. Завдяки цьому ChatGPT активно використовується в багатьох сферах, включно з освітою, бізнесом та дослідженнями.

Модель GPT-4 (Generative Pre-trained Transformer 4) є одним із найбільш розвинених прикладів великих мовних моделей (LLM), побудованих на архітектурі трансформера. Вона використовує всі основні принципи архітектури трансформера, описані вище, але з кількома важливими вдосконаленнями та оптимізаціями, як вказано на (рис. 1.1).

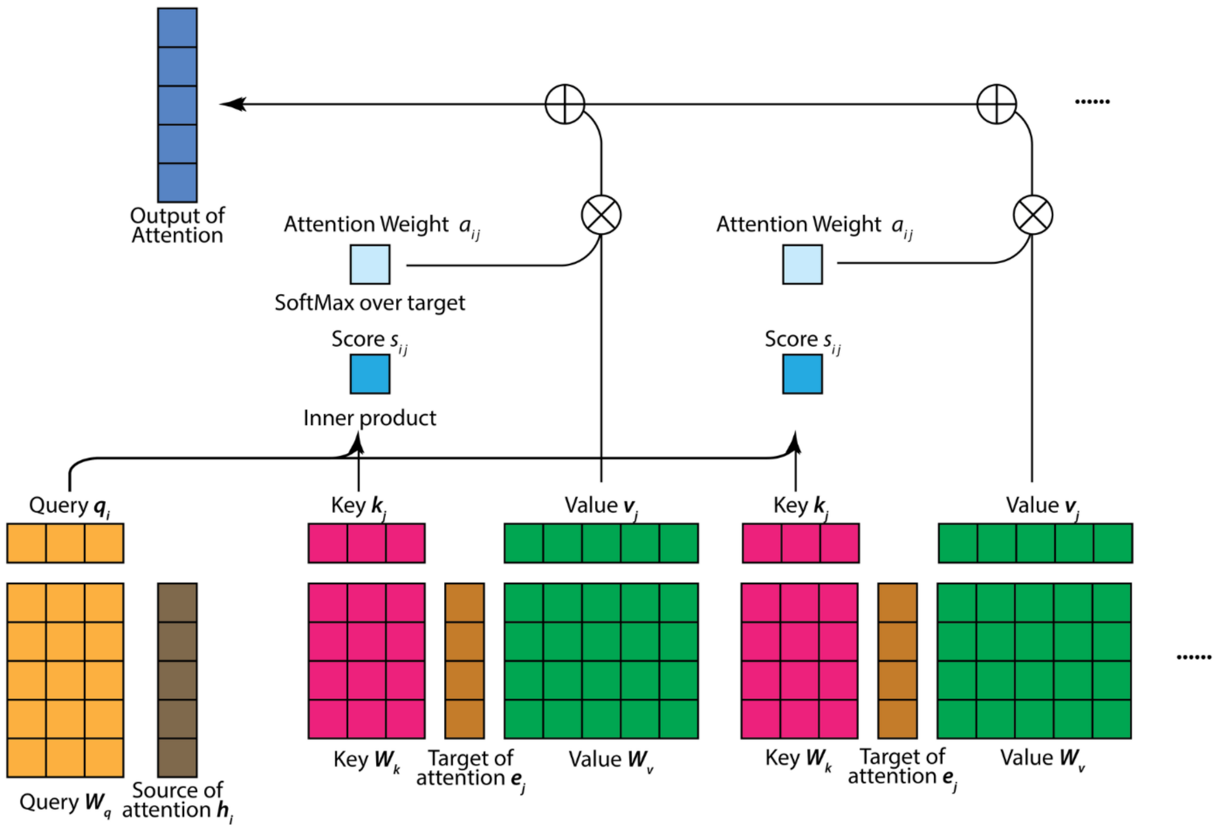

Рисунок 1.1 – Архітектура моделі трансформера

GPT-4 є декодерною мовною моделлю, що означає, що вона використовує лише частину архітектури трансформера, яка відповідає за генерацію тексту. Вона складається з багатьох шарів самоуваги та об'ємних шарів feed-forward. Кожен шар виконує обчислення самоуваги та обробляє результати за допомогою нелінійних перетворень, що забезпечує більш складні представлення вхідних даних [11]. GPT-4 значно більша за своїх попередників (GPT-3 і GPT-3.5), з більшою кількістю параметрів (ймовірно, понад 100 мільярдів), що підвищує її здатність до генерації складних і точних текстів.

Самоувага в GPT-4 працює так само, як і в класичних трансформерах, але з деякими оптимізаціями для збільшення ефективності обчислень. GPT-4 може ефективніше обробляти великі послідовності тексту завдяки техніці сегментованої самоуваги, що дозволяє моделі працювати з великими контекстами, розділяючи текст на частини та враховуючи попередній контекст при генерації наступних слів. Це дозволяє GPT-4 працювати з набагато довшими текстами, ніж попередні версії, без втрати продуктивності.

GPT-4 використовує мультиголовну самоувагу для захоплення різноманітних аспектів тексту. Кожна «голова» самоуваги аналізує різні семантичні аспекти тексту, дозволяючи моделі зберігати більш детальну інформацію про різні значення та контексти слів.

У GPT-4 використовуються вдосконалені об’ємні шари feed-forward, які є ширшими та глибшими, ніж у попередніх моделях. Це забезпечує вищу нелінійність та підвищує здатність моделі до обробки складних контекстів. Нормалізація у GPT-4 виконується після кожного шару самоуваги, що сприяє більш стабільному навчальному процесу та швидшій конвергенції. Всі ці шари та модулі вказані на (рис.1.2).

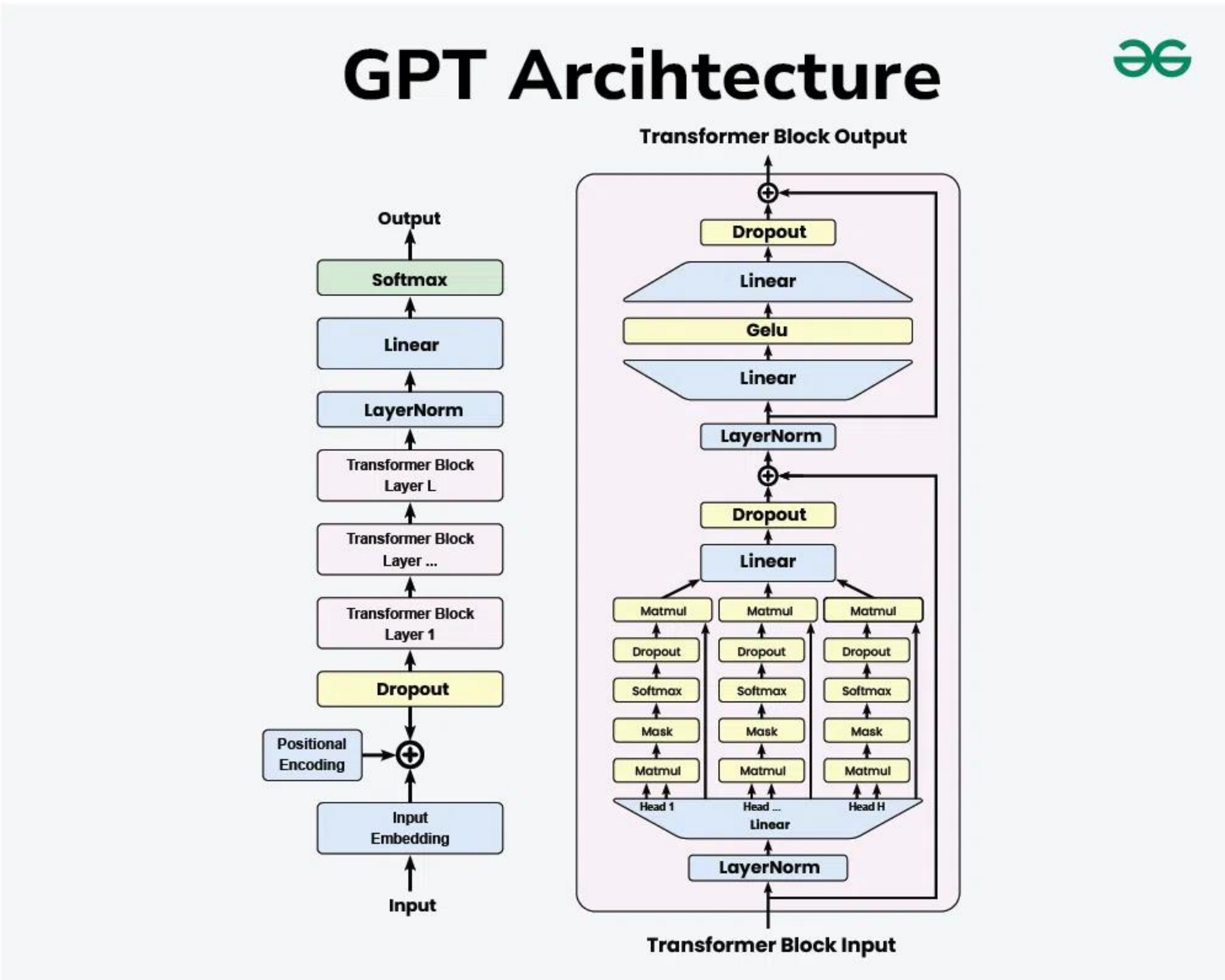

Рисунок 1.2 – Архітектура GPT моделі

Навчання GPT-4 проходить у два етапи: перенавчання та fine-tuning (тонке налаштування). На етапі перенавчання модель навчається на великих корпусах тексту з різних джерел, таких як книги, статті, веб-сторінки тощо. Це допомагає моделі розуміти широкий спектр тем і контекстів. Після цього модель проходить процес тонкого налаштування, який включає використання більш специфічних наборів даних і технік навчання, таких як RLHF (Reinforcement Learning from Human Feedback) [12], що допомагає покращити якість відповідей, підлаштовуючи їх під очікування користувачів.

Однією з ключових особливостей GPT-4 є використання RLHF для покращення відповідей моделі. Це передбачає навчання моделі за допомогою людських оцінок. Люди оцінюють якість згенерованих відповідей, що дозволяє моделі навчитися генерувати більш відповідні та зрозумілі відповіді. Модель використовує ці оцінки для корекції своїх параметрів і покращення результатів на майбутніх задачах.

GPT-4 підтримує різні методи керування генерацією тексту, такі як регулювання температури, обмеження довжини відповіді, семплювання з вершини (top-p sampling), що дозволяє балансувати між креативністю та точністю відповідей. Це забезпечує користувачам гнучкість у виборі стилю відповіді: від формальних і точних до більш розмовних та творчих.

GPT-4 може працювати з великими контекстами, дозволяючи враховувати попередній текст на декілька тисяч токенів, що робить його придатним для аналізу документів, генерації довгих текстів або діалогів із збереженням контексту.

Для представлення слів використовуються вбудовані вектори (embeddings), які кодують семантичну інформацію про кожне слово чи фразу, дозволяючи моделі працювати з абстрактними поняттями та складними структурами, процес творення векторів представленно по принципу, як вказано на рис.1.3.

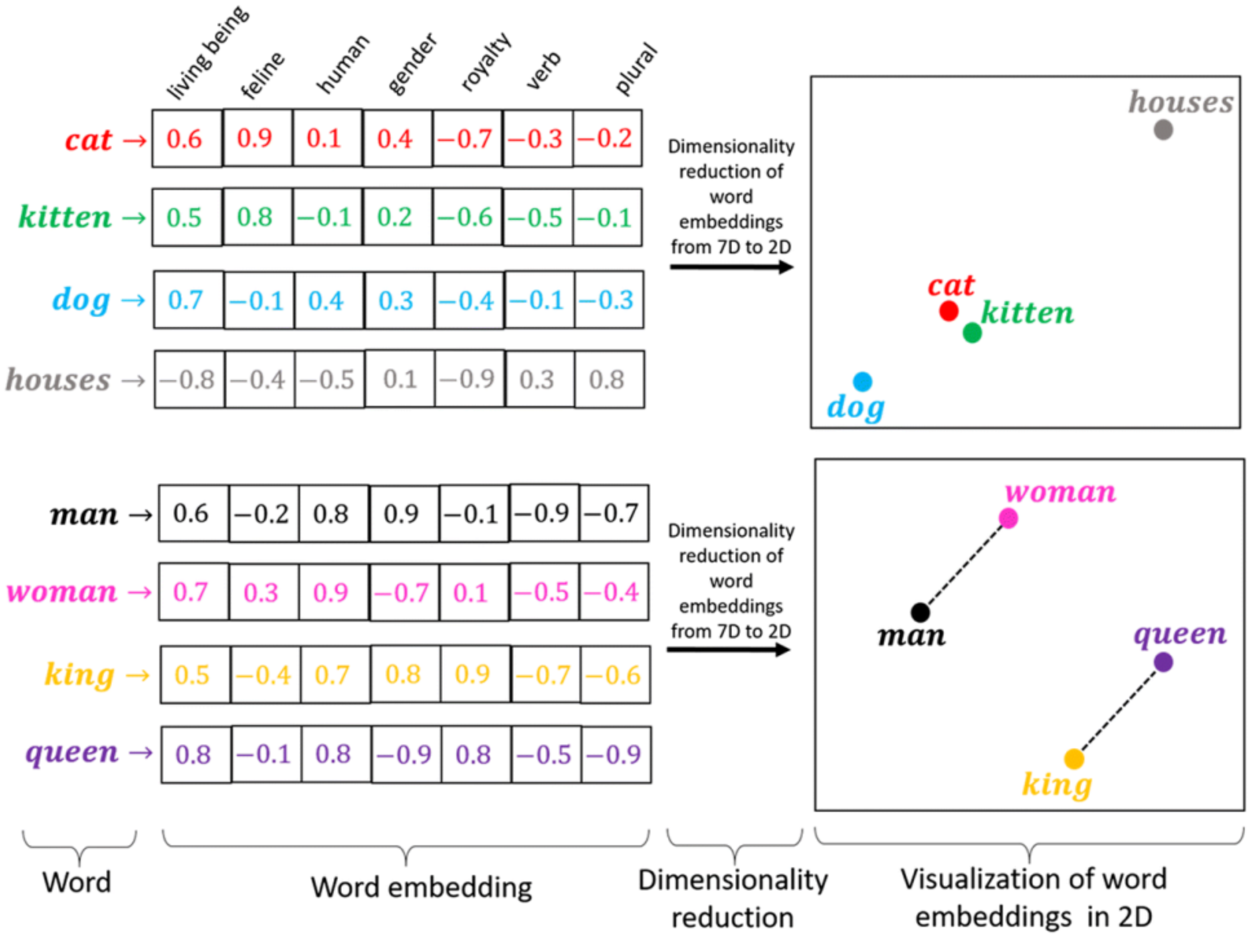

Рисунок 1.3 – Приклад візуалізації створення векторного представлення

Попри значні вдосконалення, GPT-4 все ще має обмеження. Вона може генерувати упереджені або некоректні відповіді, особливо якщо навчальні дані містили такі упередження. GPT-4 іноді не в змозі виконувати задачі, що вимагають глибокого розуміння специфічних фактів, або може бути схильна до «галюцинацій» – генерації відповідей, які виглядають правдоподібно, але є вигаданими [13]. Тому важливо комбінувати цю модель з іншими системами контролю якості та перевірки інформації.

GPT-4 демонструє виняткову продуктивність в широкому спектрі завдань обробки тексту. Вона придатна для автоматизації багатьох процесів, таких як створення контенту, переклад, відповідь на запити та інтелектуальні системи підтримки користувачів. Потенціал моделі виходить далеко за межі традиційних застосувань штучного інтелекту, відкриваючи можливості для творчих і технічних інновацій.

1.3 Принципи та етапи аналізу інтерв'ю мовними моделями

Аналіз інтерв'ю та оцінка навичок кандидатів за допомогою мовних моделей, таких як GPT, відкривають нові можливості для автоматизації процесів найму. Перший етап починається з транскрибування тексту: голосова інформація кандидата перетворюється на текстовий формат, зазвичай за допомогою спеціалізованих програм або API для розпізнавання голосу. Це дозволяє мовній моделі працювати з текстом, аналізуючи не лише зміст, а й тональність відповідей кандидата. З транскрибованим текстом модель може надалі проводити глибокий аналіз навичок та якостей кандидата.

У процесі оцінки відповіді кандидата проходять через кілька послідовних викликів мовних моделей, кожен з яких відповідає за конкретну задачу. Спершу GPT-моделі дозволяють перевірити відповідність базовим вимогам до посади, визначаючи основні компетенції, такі як знання технологій або методів, необхідних для позиції. На цьому етапі текст часто структуровано за тематикою: технічні питання, поведінкові питання і загальні компетенції, що полегшує подальший аналіз. Завдяки цьому підходу стає можливим швидко визначити відповідність кандидата до основних вимог, не переглядаючи всю інформацію вручну.

Важливим елементом є також оцінка логічності та послідовності відповідей. GPT-4 може виявляти структурність думок, визначати головні тези, підкріплюючі аргументи та їхню логічну послідовність, за рахунок збереження та передачі історії між викликами [14]. Такий підхід дозволяє краще оцінити когнітивні навички кандидата і його здатність чітко формулювати думки, що є важливим для ролей, які вимагають критичного мислення та вирішення проблем.

На завершальному етапі відповіді кандидата можна порівняти з еталонними відповідями, наданими фахівцями у цій галузі, за допомогою методів оцінки кореляцій чисельних значень, таких як метод метрика Ейлера.

Це допомагає оцінити точність і глибину знань кандидата у порівнянні зі стандартними відповідями, що особливо важливо для технічних ролей, де потрібна висока точність. Така послідовність викликів GPT дозволяє структурувати і глибоко аналізувати кожен аспект відповіді, роблячи процес відбору кандидатів більш об’єктивним, швидким і ефективним.

1.4 Постановка задачі дослідження

Використання нових, точних і автоматизованих методів аналізу тексту є актуальним завданням у галузі обробки природної мови. Ставиться завдання розроблення алгоритму для реалізації послідовної оцінки навичок кандидатів за транскрипціями інтерв’ю, використовуючи мовні моделі типу GPT, що дозволяє ефективно оцінювати компетенції кандидатів. На практиці необхідно здійснювати аналіз відповідей кандидата та зіставляти їх з еталонними відповідями на основі знань, що містяться в додаткових матеріалах.

Об’єктом роботи є методи аналізу тексту за допомогою послідовних викликів мовних моделей GPT для детального оцінювання навичок кандидатів.

Метою роботи є створення моделі автоматизованого аналізу навичок кандидатів, яка надасть можливість швидкої та точної оцінки за текстовими даними інтерв’ю.

Для досягнення мети необхідно вирішити такі завдання:

- провести аналіз недоліків існуючих методів оцінки навичок на основі текстових даних;
- розробити послідовну модель аналізу компетенцій кандидата за допомогою викликів GPT;
- застосувати розроблену модель для аналізу транскрипцій інтерв’ю та оцінки навичок кандидатів.

2 ОРГАНІЗАЦІЯ ВИКЛИКУ НЕЙРОМЕРЕЖЕВИХ ЗАСОБІВ ДЛЯ АНАЛІЗУ ТЕКСТУ

2.1 Схема послідовності виклику мовних моделей

З огляду на великі обсяги текстових даних, що аналізуються і обробляються сучасними мовними моделями, постає задача в збереженні контексту та досягненні високої точності результатів. Для цього необхідно ефективно використовувати обчислювальні ресурси та забезпечити роботу моделей з локальними даними, зберігаючи при цьому відповідність між вхідними даними та їхнім контекстом. Нові підходи до обробки тексту зазвичай спираються на глибокі нейронні мережі, які навчаються на величезних масивах тексту, що дозволяє їм зберігати складні мовні структури та контексти. Однак виникає необхідність зменшення обчислювальних витрат, оскільки процес обробки вимагає великої кількості ресурсів, особливо для завдань контекстуалізації та збереження зв’язків між різними елементами тексту [15-16].

У процесі вирішення проблеми аналізу текстової транскрипції інтерв’ю для створення автофільтрів відбору кандидатів ефективним методом є правильний виклик послідовності звернень до мовних моделей та локальних баз знань, що зберігаються у векторному представленні. Такий підхід дозволяє гнучко адаптувати аналіз тексту до конкретних вимог, спочатку використовуючи великі мовні моделі для контекстного аналізу відповідей, а потім звертаючись до локальних баз знань, представлених у векторній формі. І після цього робити окремі виклики на аналіз окремих частин тексту зв’язаних між собою контекстом, а не розбитих випадковим чином. Це дає змогу покращити точність фільтрації даних та забезпечити релевантність даних, не вимагаючи великих обчислювальних ресурсів для повного навчання моделі з нуля.

В якості даних для обробки та аналізу використовується транскрипція інтерв'ю, що проставляє собою текст розпізнаний Speech To Text (STT) моделлю з аудіофайлу запису інтерв'ю між рекрутером та кандидатом на позицію .NET розробника.

Спочатку відбувається запис розмови рекрутера з кандидатом. Ця розмова включає питання з боку рекрутера та відповіді кандидата на тему його технічних знань, навичок у .NET, C#, ASP.NET Core, Entity Framework та інші релевантні технології.

Аудіофайл передається в STT-модель, яка автоматично розпізнає мовлення, розбиває його на слова та речення, формуючи текстову транскрипцію. Якість цього перетворення значною мірою залежить від двох основних факторів: чіткості аудіозапису та ефективності моделі розпізнавання мови. Чим краща якість запису, тим точніше модель розпізнає слова й структуру речень.

Після створення текстової транскрипції необхідно провести її попередню обробку. Цей етап включає видалення зайвих пауз, мовних заповнювачів та виправлення граматичних помилок, щоб зробити текст більш читабельним і структурованим.

Далі виконується маркування тексту: кожен фрагмент транскрипції позначається відповідно до того, хто говорить у цей момент – рекрутер чи кандидат. Це дозволяє чітко розмежувати репліки для подальшого аналізу.

Остаточна транскрипція стає основою для проведення семантичного аналізу та інших методів текстової аналітики, які використовують мовні моделі. У результаті текст перетворюється на структуроване й зручне представлення, придатне для отримання ключових інсайтів і глибшого аналізу [17-20].

Для створення набору оцінок результатів аналізу тексту для інтерв'ю та використання запропонованого підходу до виклику мовних моделей наведено наступні кроки у схемі (рис. 2.1).

Для вирішення задачі аналізу текстової транскрипції вхідний текст обробляється та передається через послідовність кроків та певних модулів обробки.

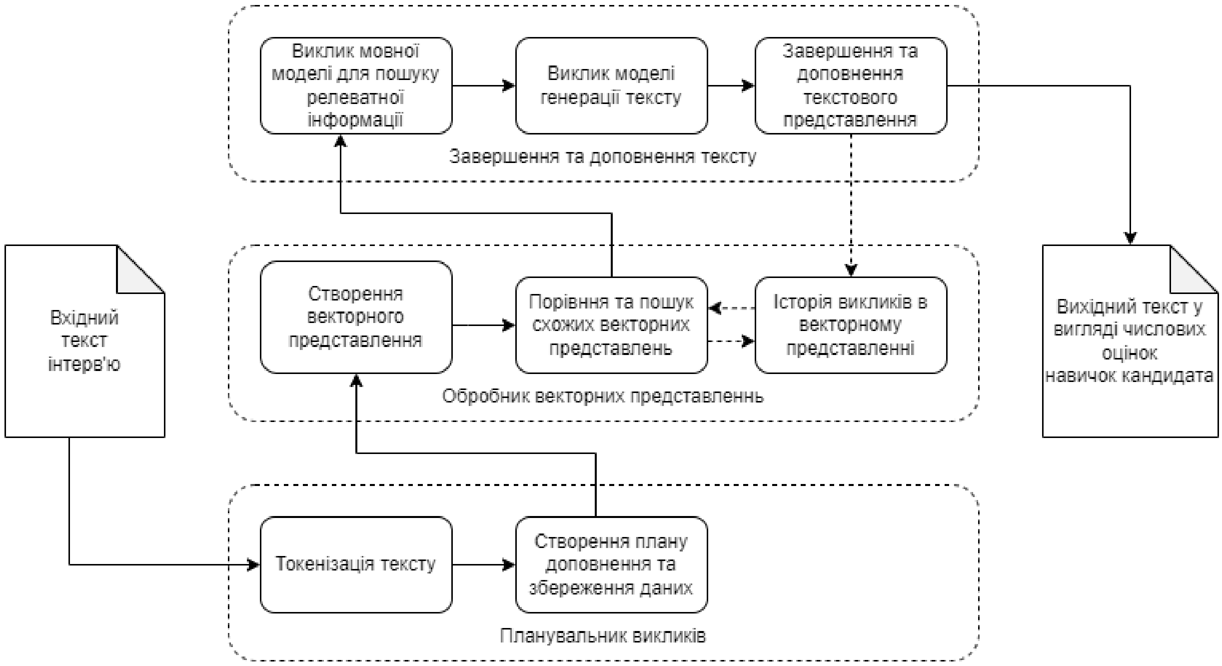

Рисунок 2.1 – Алгоритм виконання виклику модулів оцінки

На схемі зображена послідовність викликів, що демонструє процес аналізу текстової транскрипції інтерв’ю з використанням мовних моделей та векторного представлення баз знань [21]. Ось опис кожного етапу:

Крок 1. Вхідний текст – початковий текст транскрипції інтерв’ю, який буде оброблятись.

Крок 2. Токенізація тексту – На першому етапі текст ділиться на окремі елементи (токени), що дозволяє моделі працювати з різними частинами тексту окремо.

Крок 3. Виклик мовної моделі для пошуку релевантної інформації. Перший етап виклику мовної моделі, що аналізує текст для виділення основних сутностей та ключових концепцій.

Крок 4. Створення векторного представлення. Результати аналізу тексту переводяться у векторне представлення для подальшої роботи з локальними базами знань.

Крок 5. Виклик моделі генерації тексту. На основі отриманої інформації та векторного представлення виконується наступний виклик моделі для додаткового аналізу тексту та генерування нових висновків.

Крок 6. Порівняння та пошук схожих векторних представлень. Векторне представлення тексту порівнюється з базою знань для пошуку релевантних паттернів або критеріїв.

Крок 7. Історія викликів у векторному представленні. Для покращення точності аналізу зберігається історія попередніх викликів і отриманих результатів.

Крок 8. Створення плану доповнення та збереження даних. Після отримання попередніх результатів створюється план подальших викликів для завершення аналізу тексту.

Крок 9. Завершення та доповнення текстового представлення. Останні моделі завершують обробку тексту та формують підсумковий результат.

Крок 10. Вихідний текст у вигляді числових оцінок навичок кандидата. Фінальним результатом є оцінки кандидата у вигляді числових показників.

Кожен етап схеми демонструє послідовність викликів, де аналіз тексту виконується через мовні моделі, векторні представлення та збереження отриманих результатів [22]. Для формалізації задачі можна представити її у вигляді математичної моделі. Тоді є такі основні елементи:

– $T = \{t_1, t_2, ..., t_n\}$ – множина текстових транскрипцій інтерв’ю, де кожен t_i є текстом, отриманим від окремого кандидата;

– $Q = \{q_1, q_2, ..., q_m\}$ – множина запитань, на які відповідав кандидат під час інтерв’ю;

– $A = \{a_1, a_2, ..., a_k\}$ – множина зафіксованих відповідей кандидатів;

– $L = \{l_1, l_2, ..., l_p\}$ – множина локальних баз знань, що містять релевантні критерії відбору кандидатів, збережених у векторному представленні;

– $M(T)$ – велика мовна модель, яка аналізує транскрипції та забезпечує контекстуальну обробку тексту, зокрема оцінку відповідей;

– $V(T)$ – векторне представлення баз знань, до яких звертається мовна модель для уточнення критеріїв відбору;

– $f(t_i, L)$ – функція, що визначає оцінку відповідності кандидата на основі аналізу транскрипції t_i та локальних баз знань L.

Задача полягає в тому, щоб для кожної транскрипції t_i знайти таку оцінку s_i, що відображає рівень відповідності кандидата критеріям відбору:

$$s_i = f(M(t_i), V(L)). \tag{2.1}$$

де i – номер транскрипції;

t_i – транскрипція інтерв'ю;

$M(t_i)$ – це виклик мовної моделі для аналізу тексту транскрипції (t_i), яка визначає ключові аспекти відповіді кандидата;

$V(L)$ – векторне представлення локальної бази знань.

Результатом виконання цієї функції стає числовий показник в рамках вибраних для оцінки напрклад від 0 до 5.

2.2 Перетворення тексту на векторне представлення

Векторне представлення тексту – це спосіб перетворення слів, речень або навіть цілих документів у числові вектори (набори чисел), які можуть бути оброблені комп'ютером для аналізу їх змісту. В основі цього підходу лежить ідея того, що значення слів та їх взаємодія можуть бути представлені математично у вигляді векторів у багатовимірному просторі.

У випадку моделей, таких як gpt4o, текст спочатку кодується у вигляді токенів, менших частин тексту. Токени є наборами із символів, які потім перетворюються на чисельні представлення у виді векторів. Ці вектори дозволяють нейронній мережі розуміти не лише окремі слова, але й їхні зв'язки в контексті. Моделі, зокрема трансформери, вивчають семантичні зв'язки між словами на основі величезних обсягів даних [23].

Вхідних набір даних у вигляді тексту може бути перетворений за допомогою конкретної моделі трансформера на представлення в виді токенів. Приклад такого перетворення можна знайти у документації Open AI API, де вказано як на прикладі (рис. 2.2) та (рис 2.3).

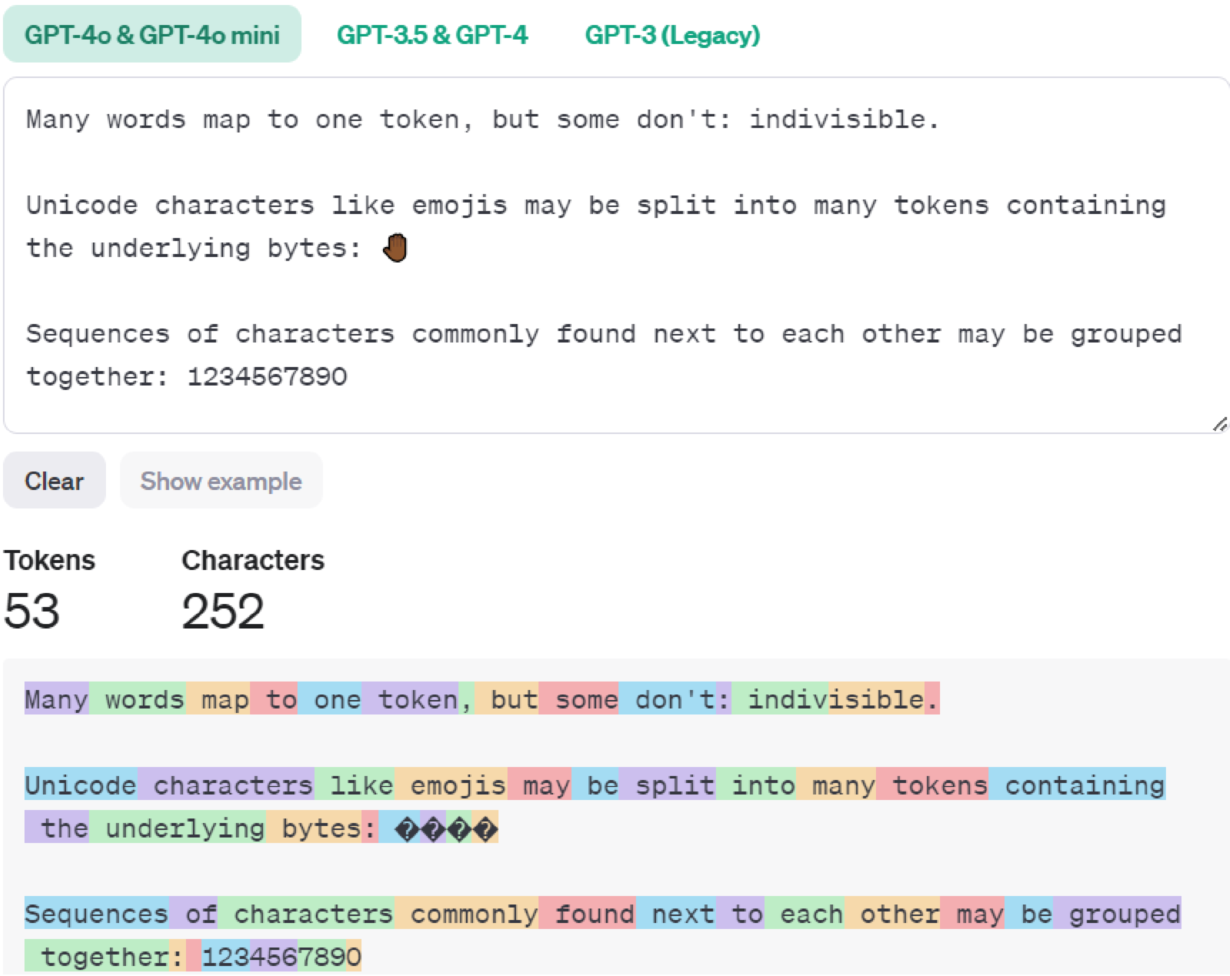

Рисунок 2.2 – Приклад перетворення тексту на токени

Модель обробляє вектори, визначаючи, які слова важливі в певному контексті, та знаходить зв’язки між ними. Вектори постійно оновлюються залежно від контексту, дозволяючи моделі розуміти значення кожного слова у контексті всієї фрази чи тексту.

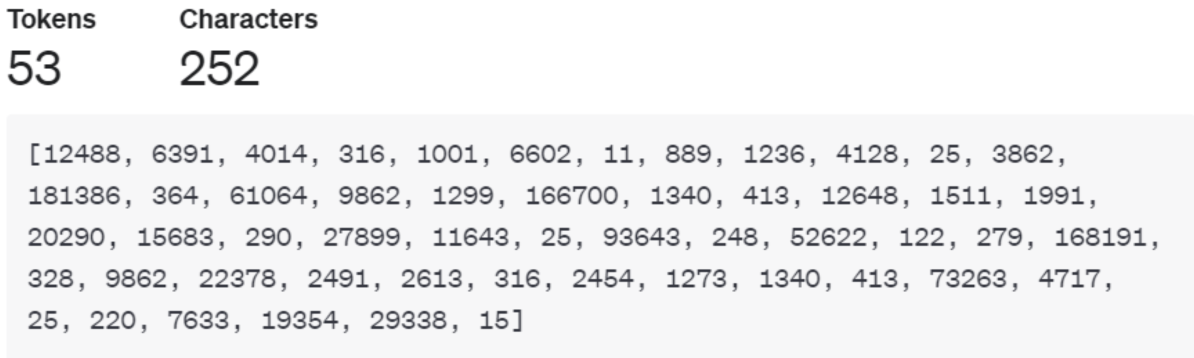

Рисунок 2.3 – Приклад перетворення тексту на токени у векторному представленні

Перетворення тексту на векторне представлення виконується за допомогою спеціальних алгоритмів, таких як Word2Vec, GloVe, FastText або складніших моделей, таких як BERT і GPT. В цьому дослідженні розглядалось виключно використання моделі gpt4o. Основні етапи цього процесу можна описати так наступним чином.

Спочатку текст розділяється на окремі елементи – токени. Це можуть бути як окремі слова, так і частини слів і набори символів. Наприклад, у випадку конкретного слова «перетворення», його можна розбити на декілька токенів, якщо воно є складним для обробки моделі.

Після цього кожен токен перетворюється на числовий вектор, який представляє його в певному багатовимірному просторі. Вектори утворюються на основі навчання моделі на величезних обсягах тексту, де вона вивчає зв’язки між словами. Чим ближче два вектори у просторі, тим більш схожими вони є за змістом [24].

Модель gpt4o, використовує контекст для того, щоб уточнити векторне представлення кожного слова в залежності від оточуючих його слів. Це означає, що одне і те ж слово в різних контекстах матиме різні вектори. Після перетворення тексту в вектори нейромережа може легко порівнювати тексти за змістом, знаходити схожі тексти або класифікувати їх за темою.

Цей процес дозволяє моделі краще розуміти та аналізувати текст, співставляти його з існуючими даними, і формувати більш обґрунтовані відповіді на основі векторних представлень.

В якості вхідних даних для аналізу в межах цього дослідження було обрано набір транскрипцій інтерв’ю на позицію Backend .NET розробника такі як на (рис. 2.4). Ці інтерв’ю є цінним джерелом інформації завдяки своїй спрямованості на перевірку глибоких технічних знань та навичок кандидатів. Питання, поставлені в рамках цього інтерв’ю, охоплюють такі теми, як архітектура високонавантажених систем, принципи роботи з ASP.NET Core, використання мови програмування C#, а також підходи до управління базами даних з використанням Entity Framework. Аналізуючи це інтерв’ю, важливо зберегти технічний контекст відповідей кандидата для точного визначення рівня його компетентностей.

У процесі збереження даних у локальній базі знань використовується модель Ada, яка виконує векторизацію текстових даних. Ця модель перетворює текст кожного фрагмента на векторні представлення, що дозволяє ефективно зберігати та аналізувати інформацію. Для більшої точності пошуку всі завантажені навчальні матеріали з мови програмування C# розбиваються на окремі сторінки. Однак кожна сторінка не існує ізольовано – система враховує попередній і наступний фрагменти, щоб зберегти логічний контекст тексту.

Завдяки векторним представленням для кожної сторінки формується унікальний вектор, який описує її зміст і використовується для подальшого порівняння з іншими сторінками або фрагментами [25-29]. Врахування сусідніх сторінок дозволяє уникнути втрати важливих деталей або розриву логічного ланцюжка, що може статися при обмеженій обробці лише однієї сторінки. Таким чином, векторизація забезпечує цілісність інформації й точність у пошуку релевантних матеріалів.

```
Recruiter: Hi Alex, thanks for taking the time to speak with me today.

Candidate: Hi, it's my pleasure. Thank you for considering me for the position.

Recruiter: Absolutely. To start, could you tell me a bit about your background and experience with .NET development?

Candidate: Sure. I have over six years of experience working with .NET technologies. I started my career focusing on
ASP.NET MVC applications and later transitioned to using ASP.NET Core. I've worked extensively with C#, Entity Framework,
and have experience with SQL Server and Azure services. In my most recent role, I was a senior developer leading a
team to develop microservices for a fintech application.

Recruiter: That's impressive. Can you share more about the fintech project you mentioned?

Candidate: Certainly. We were tasked with creating a secure, scalable platform for processing financial transactions.
I was responsible for designing the architecture using microservices with .NET Core. We implemented RESTful APIs,
utilized Azure Service Bus for messaging, and ensured compliance with industry security standards. The project
improved transaction processing speed by 40% and reduced downtime significantly.

Recruiter: Great to hear. Let's dive into some technical aspects. Can you explain the difference between an abstract
class and an interface in C#?

Candidate: Of course. An abstract class can have both abstract methods, which are declarations without implementations,
and concrete methods with implementations. It serves as a base class that other classes can inherit from. An
interface, on the other hand, can only contain declarations of methods, properties, events, or indexers without
implementations (though default implementations are allowed in newer versions). A class can implement multiple
interfaces but can inherit from only one abstract class.

Recruiter: Good explanation. How have you implemented dependency injection in your projects?

Candidate: Dependency injection is a core part of the architecture in my projects. In ASP.NET Core, I've used
the built-in IoC container to register services with different lifetimes like singleton, scoped, and transient.
This allows for loose coupling and makes unit testing easier. For example, in our fintech app, we injected
repository interfaces into our controllers to access data, which made it easy to swap out implementations for testing.

Recruiter: Excellent. Can you discuss how you've used LINQ in your applications?

Recruiter: That's great to hear. Well, I think we've covered everything for today. We'll be in touch soon
regarding the next steps.

Candidate: Thank you for your time. I'm looking forward to the possibility of working together.
```

Рисунок 2.4 – Текстова транскрипція інтерв’ю на .NET розробника

Цей підхід також підвищує ефективність пошуку, оскільки векторні представлення дозволяють швидко й точно знаходити найбільш релевантні частини тексту відповідно до запитання або теми, яку аналізує система. Завдяки цьому стає можливим швидке отримання необхідної інформації з великого обсягу навчальних матеріалів, що зберігаються в локальній базі знань.

Ключовим викликом при такому аналізі є забезпечення ефективного використання обчислювальних ресурсів. Оскільки обробка інтерв'ю включає аналіз великих обсягів тексту, необхідно оптимізувати процес для зменшення витрат часу та ресурсів. У цьому випадку нейромережі мають здатність ефективно працювати з локальними даними, що знижує навантаження на зовнішні обчислювальні ресурси та одночасно дозволяє зберігати контекстну точність.

При безпосередній оцінці також важливим фактором є мова якою передається транскрипція, та якою написані навчальні матеріали. І хоча мовні моделі підтримують українську мову та переклад на неї, в тому числі прямо під час виклику. Через відсутність достатньої кількості навчальних матеріалів для формування локальної бази знань та доступних інтерв'ю українською – було прийнято рішення проводити аналіз англіською мовою. При подальшому дослідженні цієї теми можна буде застосувати ці самі підходи і для інших мов.

2.3 Послідовний виклик модулів та локальної бази знань

Для більш точної обробки тексту інтерв'ю, метод використовує багаторівневий підхід, що включає послідовні виклики мовних моделей для розділення та аналізу даних, з метою покращення розуміння контексту та якості відповідей на різні теми, які були підняті в ході інтерв'ю.

Перший етап цього процесу полягає в попередній обробці тексту, який отриманий за допомогою моделі розпізнавання мовлення (STT). Цей текст ще не структурований і містить інформацію з усього інтерв'ю, тому потребує класифікації на основі тем, що обговорювались. Для цього застосовується набір модулів, які автоматично розподіляють текстові дані відповідно до релевантності щодо кожної з тем. Наприклад, якщо в інтерв'ю обговорювалися різні аспекти, такі як знання C#, досвід роботи з Entity Framework, або комунікаційні навички, ці модулі розпізнають ключові слова, вирази і контекст для поділу інформації за відповідними блоками.

Після цього для кожного з цих тематичних блоків проводиться окремий виклик мовної моделі. Це дозволяє виконати глибший аналіз конкретної навички або питання. Замість того, щоб аналізувати інтерв'ю цілісно, модель фокусується на одному аспекті – наприклад, оцінює знання кандидата в C# або його здатність працювати з .NET Core, відокремлюючи відповіді на ці теми від інших частин інтерв'ю. Таким чином, кожен набір даних, що стосується певної навички або технічної компетенції, обробляється окремо. Це підвищує точність розуміння контексту, дозволяючи моделі краще фокусуватися на відповідях кандидата саме по конкретній темі, що знижує ризик помилкових висновків через змішування контекстів або тем.

Коли аналіз по кожній темі завершений, результатом є окремі блоки інформації, які стосуються тільки конкретної навички або питання. Далі йде заключний етап – виклик мовної моделі для комплексного аналізу. На цьому етапі система об'єднує всі тематичні блоки та оцінює відповіді на структурованість, повноту, логіку та відповідність поставленим запитанням. Кожне запитання з інтерв'ю розглядається окремо, і відповіді кандидата оцінюються відповідно до заданої теми, але вже в контексті загальної бесіди [30]. Це дозволяє точно виявити, наскільки кандидат розуміє певні концепції та наскільки повно він відповідає на поставлені питання.

Ключовою перевагою такого підходу є те, що він дає змогу уникнути проблем із контекстом і багатозадачністю в рамках одного аналізу. Модель не намагається обробляти всі аспекти інтерв'ю одночасно, що може призводити до втрати деталей або неточностей. Замість цього інформація ретельно структурується, що дозволяє забезпечити високу точність у кожному окремому напрямку, підвищуючи якість оцінювання кандидатів і зменшуючи ймовірність помилкових висновків.

Після попереднього етапу аналізу, в процесі оцінки відповідей кандидата, відбувається інтеграція з локальною базою знань. Ця база містить релевантну інформацію, зокрема щодо мови програмування C# та інших тем, які можуть бути предметом інтерв'ю. Важливою особливістю є те, що інформація в базі зберігається у вигляді векторних представлень. Це дозволяє системі ефективно зіставляти введений текст із вже наявною інформацією.

Коли вхідний текст, тобто відповідь кандидата, токенізується (розбивається на менші компоненти – токени), система використовує ці токени для пошуку релевантних даних у базі знань. Це забезпечує доступ до конкретних фактів або концепцій, пов'язаних із темою запитання, наприклад, синтаксис або концепції C#, що були обговорені в інтерв'ю. Завдяки використанню векторних представлень, пошук інформації відбувається більш точно, оскільки векторні моделі здатні враховувати контекст і змістовну близькість між запитом і даними, зберігаючи релевантність.

Наступним етапом є виклик мовної моделі для аналізу відповідності між інформацією, отриманою з локальної бази знань, та відповіддю кандидата. На цьому етапі модель виконує базовий семантичний аналіз, тобто аналіз змісту та значення тексту, з акцентом на збереження контексту інтерв'ю. Це критично важливо, оскільки аналіз виконується в межах одного і того ж діалогу, і модель повинна оцінювати відповіді кандидата не лише на основі конкретних фрагментів інформації, але й у контексті попередніх запитань і відповідей. Підсумковий аналіз включає розгорнуту оцінку, що підсвічує сильні й слабкі сторони відповіді, наприклад, правильність базової інформації та відсутність технічних нюансів.

У локальній базі знань книги, завантажені для аналізу, розбиті на окремі частини у вигляді сторінок. Однак, щоб зберегти контекст і не втрачати важливу інформацію, кожна сторінка доповнена попередніми та наступними сторінками. Це дозволяє моделі зберігати зв’язність думок і логіку, навіть якщо частина інформації знаходиться на межі двох сторінок.

Таким чином, під час обробки запиту мовна модель отримує векторне представлення не тільки окремої сторінки, а й її найближчого контексту – це забезпечує глибше розуміння теми й підвищує точність вибору релевантних фрагментів з локальної бази. Цей підхід допомагає уникнути втрати критично важливих деталей, які можуть бути в попередніх або наступних частинах тексту, і дозволяє давати більш точні та зв’язні відповіді на запити.

Дані з локальної бази знань витягуються через ранжування їх за релевантністю, використовуючи векторні представлення вхідного тексту запитання та відповідних документів. Коли надходить запитання, мовна модель перетворює його на вектор, який порівнюється з векторами документів у базі знань, наприклад, з книгами про програмування на C#. Це дозволяє знайти найбільш релевантні фрагменти з локальної бази знань, які відповідають на запит. Вибрані фрагменти оцінюються та підтягуються в контекст для подальшої перевірки відповіді кандидата.

Після цього модель оцінює повноту відповіді кандидата, визначаючи, чи відповів кандидат на всі аспекти запитання, чи міг він пропустити певні важливі деталі. Також аналізується правильність відповіді: чи збігається те, що сказав кандидат, з очікуваними знаннями, отриманими з локальної бази знань. Мовна модель формує розгорнуту оцінку, вказуючи на сильні та слабкі сторони відповіді. Наприклад, модель може вказати, що кандидат дав правильну відповідь на базовий аспект запитання, але пропустив ключові технічні деталі або не врахував важливі концепції.

Після оцінки всіх відповідей на конкретну навичку збираються висновки. Інформація, що була отримана з локальної бази знань, разом із висновками про якість відповідей кандидата, передається в контекст для наступного виклику мовної моделі. Цей контекст використовується для подальшого аналізу й оцінки навичок кандидата, зокрема для визначення того, наскільки добре кандидат володіє конкретною технічною темою.

На завершальному етапі мовна модель викликається для комплексної оцінки за шкалою знань, що визначає рівень кандидата в конкретній навичці. Ця шкала дозволяє встановити, наскільки добре кандидат відповів на запитання, і відобразити рівень його знань: від базового розуміння до просунутого володіння технологією або інструментом.

Всі описані кроки продемонстровано на (рис. 2.5), де показано розбиття на різні теми в інтерв’ю такі як: ASP.NET, Entity Framework, синтаксис C#. Для кожної теми також приведено по 2 приклади типових запитань з цих тем та з яких даних проводиться оцінка результатів знань з конкретної теми.

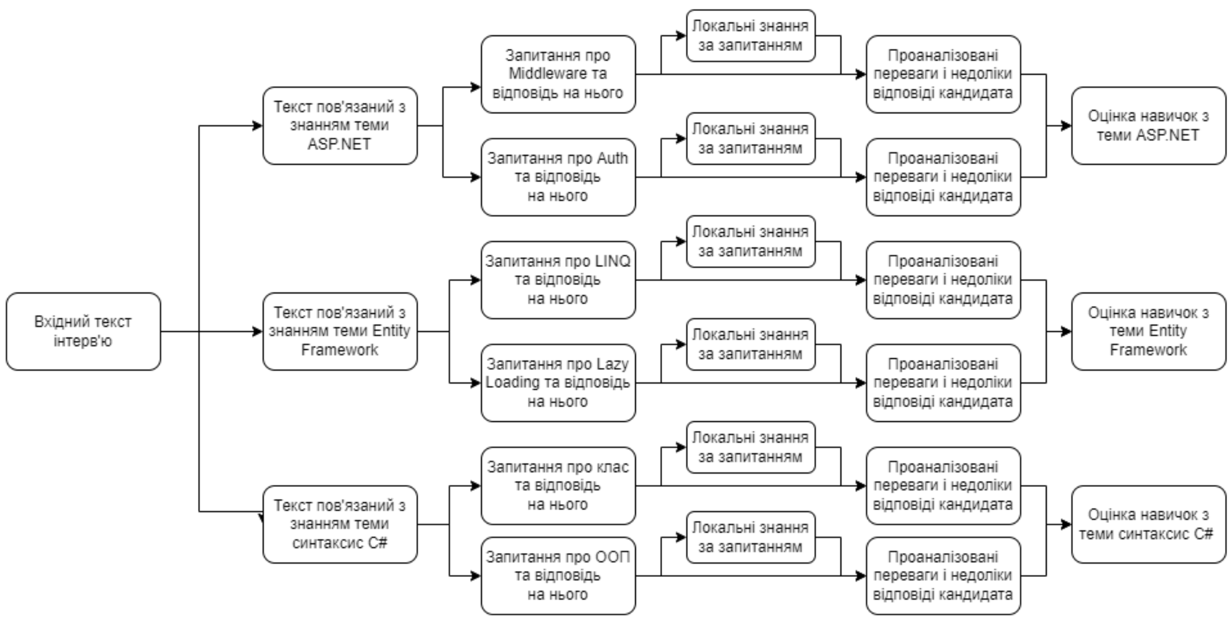

Рисунок 2.5 – Приклад кроків переходу даних після викликів мовних моделей

На завершальному етапі всі відповіді кандидата об’єднуються, і модель надає підсумкову оцінку його знань на основі повного аналізу.

2.4 Обґрунтування оцінювання якості проведеного аналізу

Для проведення аналізу порівняння результатів інтерв'ю спершу отримуються дані у вигляді оцінок кандидатів за ключовими навичками. Ці навички включають такі категорії:

– загальні комунікативні навички (Soft skills). Це оцінка здатності кандидата ефективно комунікувати, включаючи вміння розмовляти з командою, передавати інформацію і розуміти співрозмовників. Такі навички важливі для будь якої ролі в команді, оскільки від комунікації залежить якість співпраці;

– знання мови програмування C# (C# skills): Це базові знання мови програмування C#, включаючи її синтаксис та основні принципи роботи. Важливо перевірити, чи кандидат володіє всіма необхідними аспектами мови, які застосовуються у реальних проектах;

– CLR (Common Language Runtime): Це платформа виконання для додатків .NET. Оцінка знань CLR включає розуміння процесу збірки коду та управління пам'яттю в .NET, що є основоположним для роботи з платформою;

– LINQ (Language Integrated Query): Це технологія для маніпуляції даними, що дозволяє писати запити безпосередньо в коді. Важливо оцінити, наскільки кандидат розуміє цю бібліотеку для роботи з колекціями даних;

– Entity Framework: Це бібліотека для взаємодії з базами даних, яка спрощує доступ та роботу з даними. Оцінка цього компонента допомагає визначити рівень компетенції кандидата у розробці систем з великими обсягами даних;

– Azure: Це платформа хмарних рішень. Оцінка знань Azure показує, наскільки кандидат здатний працювати з хмарними сервісами для створення, деплою і управління додатками.

Усі наведені навички детально описані в різних джерелах, що зберігаються у локальній базі знань. Ця база знань слугує основою для проведення більш точного аналізу. Вона дозволяє співставляти результати, отримані від кандидатів, з перевіреними матеріалами, що описують кожну навичку. Це підвищує якість порівняння та дає змогу системі точніше оцінювати рівень знань кандидатів, використовуючи релевантну інформацію для конкретної навички.

Оцінка навичок в системі буде відбуватися в діапазоні від 0 до 5 для всіх навичок. Це забезпечить узгодженість та уніфікованість оцінювання, що дозволить більш легко порівнювати результати між кандидатами або між результатами системи та еталонними даними. Кожна навичка, незалежно від її специфіки, оцінюється в межах цього діапазону, що полегшує інтерпретацію результатів та їх подальший аналіз.

Після збору цих даних вони представлені у вигляді числового вектора. Кожна навичка має своє числове значення, яке в подальшому може бути використане для порівняння результатів між кандидатами та для аналізу системи. Основна ідея полягає в тому, щоб використовувати цей вектор для оцінки різних моделей роботи системи і коригування процесу виклику моделей з метою підвищення точності [31].

Крім того, було створено тренувальну вибірку даних. Ця вибірка складається з результатів оцінки професіонала в області .NET, який мав досвід проведення співбесід. Професіонал вручну оцінив транскрипції розмов кандидатів за тими ж критеріями, що й система, використовуючи однакову шкалу оцінок. Цей процес дозволяє порівняти роботу автоматичної системи з людською оцінкою.

Для порівняння результатів використовується метрика Ейлера, яка дозволяє вимірювати відстань між числовими векторами результатів системи та тренувальних даних. Метрика Ейлера краще підходить для цього завдання, оскільки вона більш чутлива до реальних розбіжностей між значеннями оцінок. Наприклад, якщо кандидат отримав оцінки, що близькі між собою, евклідова відстань буде меншою, що вказує на високий рівень відповідності результатів. У той же час більша розбіжність між оцінками окремих навичок призводить до значно більшої евклідової відстані, показуючи чіткі відмінності між кандидатами або системою та еталонною оцінкою. Це дозволяє виявити дрібні відмінності, які можуть бути непомітними при використанні інших метрик, таких як Манхетенська відстань, яка просто сумує абсолютні значення різниць.

Для опису метрики Ейлера (евклідової відстані) та її нормалізації спочатку представимо базову формулу для розрахунку евклідової відстані між двома векторами оцінок навичок [32].

Нехай у нас є два вектори оцінок для двох кандидатів або для системи та тренувальної вибірки: $\vec{X} = (x_1, x_2, ..., x_n)$ та $\vec{Y} = (y_1, y_2, ..., y_n)$

Тоді евклідова відстань d між цими двома векторами обчислюється за формулою:

$$d(\vec{X}, \vec{Y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (2.2)$$

де x_i та y_i – це оцінки навичок для кожного кандидата або для порівнюваних результатів;

n – кількість навичок, що оцінюються; у цьому прикладі використано 6 навичок: Soft skills, C#, CLR, LINQ, Entity Framework, Azure.

Це означає, що система порівнює оцінки навичок, які знаходяться в однаковому діапазоні, дозволяючи більш зручно та інтуїтивно оцінювати відповідність між результатами [33]. Нормалізація також допомагає забезпечити рівнозначний внесок кожної навички в підсумковий результат, що робить порівняння результатів системи більш зрозумілим та простим для інтерпретації.

Нормалізація метрики Ейлера може відбуватися шляхом поділу отриманого значення відстані на максимальну можливу відстань у цьому діапазоні. У випадку наявності оцінок кожен елемент яких може варіюватися в межах певного діапазону, максимальна відстань між двома векторами $\vec{X}$ та $\vec{Y}$ буде наступною:

$$d_{\max} = \sqrt{n \times (a_{\max} - a_{\min})^2} \tag{2.3}$$

де n – кількість навичок, що оцінюються;

$a_{\max}$ – максимальний бал при оцінюванні;

$a_{\min}$ – мінімальний бал при оцінюванні.

Тобто, максимальна евклідова відстань для оцінок, що можуть змінюватися між 0 і 5 та , становить $d_{\max} = \sqrt{6 \times (5-0)^2} = 5 \times \sqrt{6} \approx 12.247$.

Щоб нормалізувати евклідову відстань $d(\vec{X}, \vec{Y})$, отриману за попередньою формулою, можна поділити її на максимальну відстань:

$$d_{normalized} = \frac{d(\vec{X}, \vec{Y})}{d_{\max}}. \tag{2.4}$$

Нормалізована відстань буде лежати в діапазоні від 0 до 1, що дозволяє легше інтерпретувати відповідність результатів – чим ближче до 0, тим більш схожі оцінки, а чим ближче до 1, тим більше відмінностей між ними.

3 ДОСЛІДЖЕННЯ ПРОГРАМНИХ МЕТОДІВ ОБРОБКИ ПРОМПТІВ ТА КОНТЕКСТУ

3.1 Обґрунтування вибору засобів виклику мовних моделей

У рамках кваліфікаційної роботи був розроблений алгоритм для автоматичної обробки текстових транскрипцій інтерв’ю, що дозволяє об’єктивно оцінювати компетенції кандидатів. Цей алгоритм поєднує сучасні підходи до аналізу тексту, включаючи використання великих мовних моделей (LLMs) та інтеграцію з локальними базами знань, представленими у векторному форматі. Метою розробки є автоматизація процесу оцінки, підвищення точності аналізу та забезпечення релевантності результатів.

Основна ідея алгоритму полягає у структурованому аналізі відповідей кандидата на основі контексту бесіди. Текст транскрипції попередньо обробляється: маркується, очищується від зайвих елементів і розподіляється на тематичні блоки. Потім алгоритм використовує мовну модель для визначення ключових аспектів відповідей, а також порівнює їх із релевантними матеріалами з локальної бази знань. Завдяки цьому забезпечується глибоке розуміння контексту та відповідність відповідей кандидата заданим критеріям.

Результатом роботи алгоритму є числові оцінки знань та навичок кандидата за ключовими категоріями, такими як знання C#, ASP.NET Core, Entity Framework, Azure, а також комунікативні здібності. Це дозволяє формувати об’єктивні висновки про рівень кваліфікації кандидата, підвищуючи ефективність та прозорість процесу відбору.

У сучасних умовах автоматизації аналізу текстових даних особливої уваги заслуговують технології, які поєднують потужність великих мовних моделей із гнучкістю інтеграції локальних джерел інформації. Для забезпечення контекстної обробки текстів транскрипцій інтерв’ю та оцінки компетенцій кандидатів у цій роботі використовується Semantic Kernel –

інноваційний інструмент, що дозволяє зберігати та використовувати контекст, інтегрувати мовні моделі з локальними базами знань та виконувати складні сценарії обробки даних. Цей підхід дає змогу не лише підвищити точність аналізу, але й забезпечити об’єктивність та прозорість оцінювання, що робить його ефективним рішенням для задач автоматизації у сфері підбору персоналу.

Semantic Kernel (SK) – це інструмент, розроблений для створення інтелектуальних додатків, які об’єднують можливості великих мовних моделей (LLMs) із традиційними обчислювальними процесами. Його основна мета це створення систем, здатних виконувати складні завдання шляхом інтеграції знань, контексту і дій у єдину платформу.

Semantic Kernel працює як модульний каркас, який дозволяє розробникам комбінувати мовні моделі з власними функціями, сценаріями та локальними базами знань, зберігаючи при цьому контекст завдань. SK забезпечує ефективний спосіб обробки тексту, виконання запитів і створення висновків, враховуючи всю історію взаємодії [34].

Semantic Kernel працює за основними принципами, які забезпечують ефективну інтеграцію штучного інтелекту в різні сценарії. По-перше, він використовує великі мовні моделі (LLMs), такі як OpenAI GPT чи Azure AI, для виконання завдань, що потребують розуміння тексту, створення відповідей, аналізу даних чи проведення складних обчислень. Це дає змогу адаптувати сценарії до запитів і контексту користувачів. Важливою особливістю є збереження контексту: SK може враховувати історію взаємодії, забезпечуючи узгодженість і послідовність у відповідях.

Semantic Kernel базується на кількох ключових компонентах. Skills – це набір функцій для виконання конкретних завдань, таких як пошук інформації, аналіз тексту або обчислення. Memory забезпечує збереження історії взаємодії, що допомагає враховувати попередні дані. Planner створює послідовність дій для реалізації складних сценаріїв. Connectors інтегрують

SK із зовнішніми системами, такими як API, локальні бази знань або хмарні сервіси.

Модульний підхід дозволяє поєднувати можливості великих мовних моделей із зовнішніми функціями: викликом API, роботою з базами даних, обробкою векторних представлень тощо. Завдяки цьому інтеграція AI з існуючими системами значно спрощується. Semantic Kernel також підтримує роботу з текстом у векторному форматі, що корисно для задач із глибокого аналізу семантики чи обробки великих обсягів даних .

Як приклад, можна розглянути додаток для аналізу транскрипцій інтерв'ю. Semantic Kernel обробляє текстову транскрипцію, використовуючи LLM для аналізу відповідей, виділення ключових ідей і оцінки знань кандидата. Завдяки пам'яті SK враховує попередні відповіді, що робить аналіз послідовним. На виході генерується структурований звіт з оцінками навичок, порівнянням із локальною базою знань і рекомендаціями.

Semantic Kernel ідеально підходить для розробки систем, які потребують інтеграції великих мовних моделей із зовнішніми джерелами даних, збереженням контексту й виконанням складних сценаріїв.

3.2 Застосування принципів семантичних плагінів Semantic Kernel

Семантичний плагін – це концептуальний модуль у Semantic Kernel, який виконує специфічні завдання, орієнтовані на семантичну обробку даних [35-36]. Плагін використовує можливості мовних моделей для виконання функцій, пов'язаних із текстовим аналізом, генерацією або взаємодією з іншими системами. Основна цінність плагіна полягає у його здатності спрощувати інтеграцію LLM у різноманітні сценарії завдяки:

Інкапсуляції логіки: Усі функції плагіна зібрані в одному місці, що дозволяє легко змінювати або розширювати їх.

Модульності: Плагіни можна створювати, видаляти або замінювати без значного впливу на роботу інших компонентів.

Семантичної адаптації: Вони дозволяють налаштовувати відповіді моделі відповідно до домену знань (наприклад, медицина, право, IT).

Наприклад, якщо потрібно створити плагін для аналізу тексту резюме, функція плагіна може автоматично витягувати ключову інформацію: ім’я, контактні дані, досвід роботи, навички тощо.

В Semantic Kernel функції плагінів бувають двох типів:

Семантичні функції: задаються у вигляді шаблонів текстових запитів до LLM. Наприклад, функція, яка витягує ключові слова (рис. 3.1) та має окремий файл з конфігурацією де вказуються важливі показники для виклику плагіну (рис. 3.2).

```
Given a conversation transcription between a interviewer and a candidate,
Indentify the candidate's {{$skill}} skills level.
Skill basic description: {{$skill_description}}
Book knowledges regarding skill: {{$skill_info}}

Describe the reasoning why you giving the candidate some point for
knowing the theme and describe possible gaps in the candidate's knowledge.
Also, provide deep insights into the candidate's knowledge and skills.

Output Template:
The candidate {{$skill}} skills level is great, because he knows the basic,
he knows the advanced and he knows the expert info about {{$skill}}.
His possible gaps are the missing knowledge about compiler work.

Conversation Transcription:
{{$input}}
```

Рисунок 3.1 – Приклад тіла промпту для семантичного плагіну

Кодовані функції: реалізуються програмістом мовою програмування, яка використовується в Semantic Kernel (наприклад, C# або Python). Вони дозволяють виконувати складні логічні або математичні операції.

```json
{
  "schema": 1,
  "description": "Evaluate the knowledge and practical experience of candidate regarding the parrticullar skill",
  "type": "completion",
  "completion": {
    "max_tokens": 1000,
    "temperature": 0.5,
    "top_p": 1.0,
    "presence_penalty": 0.0,
    "frequency_penalty": 0.0,
    "model_id": "gpt-4",
    "service_id": "gpt-4"
  }
}
```

Рисунок 3.2 – Приклад файлу конфігурації для семантичного плагіну

Файл конфігурації визначає, як плагін Semantic Kernel працює з мовною моделлю для виконання завдання.

Параметр "schema": 1 визначає версію формату файлу, що забезпечує сумісність із Semantic Kernel. "description" описує мету плагіна – у цьому випадку оцінку знань і практичного досвіду кандидата з певної навички. "type": "completion" означає, що модель буде виконувати текстову генерацію.

У блоці "completion" параметр "max_tokens": 1000 обмежує довжину тексту відповіді, наприклад, якщо відповідь на завдання не повинна перевищувати 1000 токенів (що включає як вхідний текст, так і згенеровану відповідь). "temperature": 0.5 регулює креативність відповіді: модель генеруватиме збалансовані відповіді, не надто передбачувані, але й не хаотичні. Наприклад, якщо кандидату задається питання про технічну концепцію, відповідь буде структурованою, без зайвих імпровізацій.

Параметр "top_p": 1.0 дозволяє враховувати всі можливі варіанти вибору токенів під час генерації, забезпечуючи повну варіативність. Це корисно, якщо потрібна повноцінна відповідь, без обмеження на вибір лише найбільш імовірних слів. "presence_penalty": 0.0 і "frequency_penalty": 0.0 не штрафують модель за повторення токенів, що підходить для завдань, де можлива часта згадка певних ключових термінів (наприклад, технічних термінів у відповіді).

Параметри "model_id": "gpt-4" і "service_id": "gpt-4" визначають використання конкретної моделі GPT-4. Це забезпечує високу точність і якість відповіді, що критично для завдань, пов’язаних із комплексною аналітикою або оцінкою знань.

Для конфігурації та налаштування семантичних плагінів необхідно виконати наступні етапи:

Створення функцій плагіна. Семантичний плагін зазвичай складається з функцій, які виконують чітко визначені завдання.

Реєстрація плагіна. Для інтеграції плагіна в SK його необхідно зареєструвати в внутрішньому окнтейнері реалізції, наприкалд Dependency Injection контейнері, для подальшого використання через рефлексію. Це дозволяє ядру знати про плагін і забезпечити доступ до його функцій.

Налаштування доступу до ресурсів. Якщо плагін взаємодіє з зовнішніми джерелами (наприклад, базами даних, вебсервісами), необхідно налаштувати доступ. Зазвичай це передбачає вказівку API-ключів або токенів авторизації, конфігурацію URL для підключення до API, Налаштування тайм-аутів та інших параметрів.

Семантичні плагіни в Semantic Kernel викликаються через стандартний інтерфейс, що дозволяє працювати з їхніми функціями. Є кілька підходів до виклику:

– прямий виклик функції. Цей підхід підходить для виконання конкретної функції плагіна і в цьому сценарії функція з плагіном напряму викликається в виконуваному коді;

– інтеграція у складні сценарії. Функції плагінів можуть викликатися як частина більших процесів. Наприклад, сценарій обробки заявки кандидата може включати. Витяг ключових моментів резюме. Порівняння отриманих даних із вимогами вакансії. Оцінку відповідності. В ході виклику різних плагінів, контекст передаванних зміних буде залишатись одним, що дозволить налаштовувати правильну його обробку всередині логіки самих цих плагінів;

– автоматичний запуск плагіна. Semantic Kernel дозволяє запускати плагіни автоматично на основі тригерів чи умов, заданих у коді. Та автоматично визначає який плагін має бути викликаним на основі їх опису та поточної збереженої інформації в середині контексту виконання коду.

Семантичні плагіни мають широкий спектр застосувань, спрямованих на автоматизацію завдань, інтеграцію з бізнес-процесами, розширення можливостей Semantic Kernel і покращення точності обробки даних. Вони дозволяють автоматизувати роботу з текстами, наприклад, генерувати звіти, відповідати на запити, перекладати тексти на інші мови або витягувати ключову інформацію з великих текстових масивів, таких як важливі дати чи імена [37-39]. Це особливо корисно у сферах, де обробка великих обсягів даних вручну є надто тривалою або ресурсозатратною.

Плагіни також інтегруються в бізнес-процеси, допомагаючи з автоматизацією, наприклад, заповнення даних у CRM-системах або створення чат-ботів, які поєднують відповіді мовних моделей із базами знань. Вони можуть бути корисними у сфері аналітики для аналізу відгуків клієнтів, що дозволяє оперативно реагувати на потреби користувачів і покращувати сервіс.

Завдяки кастомним плагінам Semantic Kernel можна адаптувати для вирішення специфічних завдань, таких як аналіз фінансових документів, оцінка ризиків у медичних чи юридичних звітах або навіть створення навчальних матеріалів. Це забезпечує високу гнучкість системи та дозволяє застосовувати її в різних галузях.

Окрім того, семантичні плагіни покращують точність обробки даних, використовуючи контекстно-специфічну логіку. Це робить їх ефективнішими порівняно зі стандартними алгоритмами, особливо в завданнях, де важлива глибока семантична обробка інформації або унікальні доменні знання.

Було реалізувавано підхід, який дозволяє Semantic Kernel визначати намір (intent) кожного окремого виклику на основі вхідних даних і автоматично підбирати відповідний семантичний плагін для виконання завдання. Це стало можливим завдяки використанню мовної моделі, яка аналізує текст, визначаючи, яке саме завдання має бути виконане: оцінка знань кандидата, витяг ключових даних, пошук додаткової інформації тощо. На основі цього наміру викликається потрібний плагін із чітко заданими параметрами.

Ключовим моментом є те, що цей підхід автоматизує процес, позбавляючи необхідності вручну задавати плагін або визначати його функції. Модель сама розуміє, що від неї потрібно, і відповідно адаптує свою поведінку. Наприклад, для аналізу відповіді кандидата вона може одразу витягнути ключову інформацію, перевірити її на відповідність вимогам вакансії та створити структурований звіт.

Одним із ключових досягнень у реалізації системи стало впровадження механізму послідовного виклику плагінів, що дозволяє значно покращити якість аналізу. Завдяки цьому підходу обробка даних стає не лише автоматизованою, але й структурованою, що забезпечує глибоке розуміння кожного етапу процесу. На початковому етапі система звертається до хмарного сховища для завантаження релевантної інформації, яка може включати вимоги до конкретної вакансії, дані про компанію чи результати попередніх етапів відбору. Ці знання є критично важливими, оскільки вони створюють контекст, без якого аналіз відповідей кандидата може бути неповним або поверховим.

Контекст із хмарного сховища інтегрується в подальші кроки, де плагіни виконують строго визначені завдання. Спочатку аналізується загальна відповідь кандидата, оцінюються її ключові аспекти, такі як логіка викладу, відповідність поставленому питанню та структура аргументації.

Далі залучаються моделі, які спеціалізуються на технічному аналізі. Вони порівнюють отримані дані з конкретними вимогами до навичок чи знань, що є важливими для успішного виконання посадових обов'язків. На завершальному етапі вся отримана інформація об'єднується, і формується комплексний звіт, який не лише відображає сильні та слабкі сторони кандидата, але й надає рекомендації щодо його потенційної придатності до ролі.

Послідовний виклик плагінів дозволяє системі враховувати як локальний контекст кожної відповіді, так і глобальні фактори, що стосуються специфіки вакансії чи очікувань компанії. Це суттєво підвищує точність оцінки, оскільки враховується не тільки текстова відповідь, але й взаємозв'язок між різними аспектами оцінювання. Крім того, такий підхід мінімізує ризик помилок через недостатню або нерелевантну інформацію, оскільки кожен плагін працює з максимально адаптованим до нього контекстом.

У реалізованій системі для аналізу інтерв'ю було впроваджено комплексний підхід завдяки інтеграції семантичних плагінів, які виконують різні аспекти оцінювання відповідей кандидатів. Кожен із плагінів відіграє важливу роль у загальному процесі, забезпечуючи детальний аналіз різних параметрів і допомагаючи отримати повну картину компетенцій кандидата.

CandidateConfidence – цей плагін оцінює рівень впевненості кандидата у своїх відповідях. Його основна мета – визначити, наскільки впевнено кандидат відповідає на запитання, базуючись на структурі, стилі й послідовності його мови. Плагін може бути корисним для виявлення слабких місць, де кандидат демонструє невпевненість, що може вказувати на прогалини у знаннях чи досвіді. Цей параметр є особливо важливим для оцінки комунікативних навичок і здатності кандидата пояснювати свої думки.

EvaluateSkillByGrade – цей плагін виконує оцінку навичок кандидата за попередньо заданою шкалою. Він аналізує відповідь кандидата й порівнює її з еталонними відповідями або матеріалами з бази знань, присвоюючи відповідну оцінку. Цей плагін корисний для стандартизації процесу оцінки, забезпечуючи об’єктивність і можливість порівнювати результати між різними кандидатами. Наприклад, він може оцінити рівень знань у таких областях, як Dependency Injection або асинхронне програмування в C#, і надати оцінку за шкалою (наприклад, від 1 до 10).

EvaluateSkillConclusion – цей плагін формує загальний висновок щодо оцінюваних навичок кандидата. Він агрегує результати інших плагінів і створює підсумковий звіт, який може включати основні сильні та слабкі сторони кандидата, а також його відповідність вимогам вакансії. Цей плагін є корисним для інтерв’юера, оскільки дозволяє отримати готові рекомендації на основі всебічного аналізу.

SkillDeepKnowledge – плагін відповідає за глибокий аналіз специфічних навичок кандидата. Він перевіряє знання кандидата в деталях, наприклад, розуміння технічних концепцій або специфіки реалізації певних алгоритмів. Основна цінність цього плагіна полягає в його здатності виявляти рівень професійної компетенції в конкретних технічних аспектах, що є ключовим для оцінки глибини знань.

SkillUnderstanding – цей плагін оцінює рівень розуміння кандидатом основних концепцій і принципів, пов’язаних із запитанням. Наприклад, якщо питання стосується Dependency Injection, плагін перевіряє, чи кандидат розуміє основні принципи роботи цього патерну, такі як інверсія управління та контейнер залежностей. Плагін є корисним для визначення базового рівня знань, що допомагає інтерв’юеру зрозуміти, чи володіє кандидат фундаментальними знаннями в предметній області.

Результати аналізу стають максимально релевантними, оскільки система не лише оцінює відповіді кандидата в ізоляції, а й інтегрує їх у загальну картину, враховуючи всі доступні дані. Це дозволяє приймати більш обґрунтовані рішення, які базуються на комплексному розумінні сильних і слабких сторін кандидата, його відповідності посаді та можливостях для подальшого розвитку [40-41]. Така архітектура не просто виконує завдання, а й робить процес оцінки кандидата максимально прозорим і осмисленим.

Усі ці плагіни працюють разом, створюючи комплексний процес оцінки інтерв'ю. Спочатку система збирає дані про відповідь кандидата, аналізуючи його впевненість, розуміння, глибину знань і відповідність навичок еталонним стандартам. Потім результати всіх плагінів об'єднуються у підсумковий звіт, який допомагає інтерв'юеру отримати повну картину про компетенції кандидата. Завдяки такому підходу система забезпечує об'єктивність, структурованість і точність оцінки, що значно спрощує процес прийняття рішень під час найму.

Окрім цифрової оцінки, система формує розширений профіль кандидата, який надає детальну характеристику його компетенцій. Цей профіль дозволяє інтерв'юеру глибше зрозуміти як технічні навички, так і загальні особливості кандидата. Одним із ключових елементів цього профілю є опис сильних і слабких сторін кандидата у різних технічних аспектах. Наприклад, система може зафіксувати, що кандидат добре володіє базовими концепціями Dependency Injection, включаючи інверсію управління та механізм контейнерів залежностей, але водночас виявляє поверхове розуміння таких складніших аспектів, як Service Lifetime або налаштування залежностей у динамічному середовищі. Ця інформація допомагає виявити, на яких саме аспектах варто зосередити подальшу увагу під час інтерв'ю чи навчання.

Додатково система проводить аналіз стилю відповідей кандидата, враховуючи такі фактори, як упевненість, чіткість та логічність викладу думок. Завдяки цьому формується уявлення про те, наскільки кандидат упевнено презентує свої знання. Наприклад, якщо плагін CandidateConfidence виявив низький рівень впевненості у відповідях, це може свідчити про недостатню підготовку, нерішучість чи навіть незнання ключових аспектів теми. Такий результат може бути корисним сигналом для інтерв'юера, який може уточнити окремі моменти чи поставити більш деталізовані запитання, щоб зрозуміти причини невпевненості.

Крім цього, система здатна пропонувати рекомендації для подальшого оцінювання. Якщо, наприклад, плагін SkillDeepKnowledge виявив, що кандидат має обмежене розуміння складних технічних концепцій, система може згенерувати додаткові запитання, які допоможуть глибше дослідити певні аспекти знань. Такі рекомендації дозволяють інтерв’юеру ефективніше використовувати час інтерв’ю, зосереджуючись на тих областях, які потребують додаткової уваги.

Ще одним важливим елементом профілю є оцінка відповідності кандидата вимогам вакансії. Система автоматично аналізує відповіді кандидата у співвідношенні з ключовими навичками, зазначеними в описі вакансії, і вказує, наскільки вони співпадають. Наприклад, вона може показати, що кандидат найкраще відповідає вимогам у таких аспектах, як розуміння базових принципів програмування, але має недостатній досвід роботи з конкретними інструментами чи технологіями, зазначеними в описі ролі.

Підсумковий звіт для інтерв’юера включає як цифровий бал, що дозволяє легко порівнювати кандидатів між собою, так і розширену характеристику, яка додає контекст і допомагає глибше зрозуміти сильні та слабкі сторони кожного кандидата. Це також включає конкретні рекомендації щодо подальшого оцінювання чи розвитку кандидата.

3.3 Архітектура модулів для реалізації програмного застосунку

У процесі розробки системи було впроваджено принципи Clean Architecture, що дозволило створити гнучку, модульну та стійку до змін структуру проєкту. Основна ідея полягала в розподілі логіки на чітко визначені рівні: доменний, інфраструктурний, прикладний і презентаційний.

Це забезпечило ізоляцію бізнес-логіки від деталей реалізації, таких як робота з базою даних чи інтеграція семантичних плагінів. Завдяки такому підходу система стала легшою у підтримці та масштабуванні, оскільки будь-які зміни в окремих модулях не впливають на інші.

Для обробки даних і забезпечення ефективності процесів було впроваджено низку модулів, які підтримують асинхронну обробку (рис. 3.3). Це дозволяє системі обробляти великі обсяги даних одночасно, не блокуючи основний потік виконання завдань.

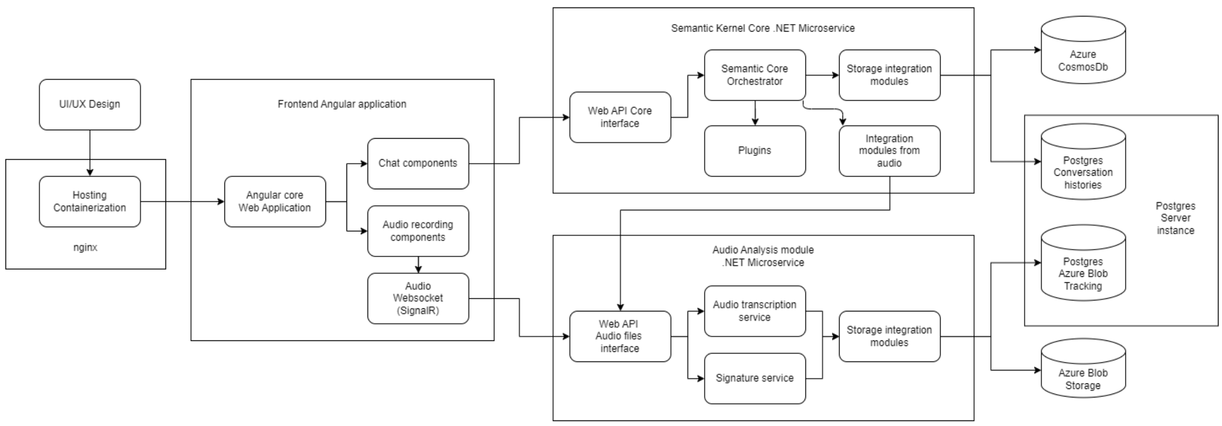

Рисунок 3.3 – Архітектура програмної реалізації модулів обробки інтерв’ю

Наприклад, обробка даних, отриманих від плагінів, збереження результатів аналізу в базі даних та формування підсумкових звітів виконуються в асинхронному режимі. Це забезпечує високу продуктивність навіть за умов, коли аналізу підлягає кілька відповідей одночасно або відбувається обробка складних задач із залученням великої кількості зовнішніх ресурсів.

Обробка даних у системі організована таким чином, щоб максимально використовувати можливості асинхронної архітектури, що значно підвищує її продуктивність та масштабованість. Наприклад, аналіз даних, отриманих від семантичних плагінів, збереження результатів у базі даних і формування підсумкових звітів відбувається в асинхронному режимі. Це забезпечує одночасну обробку кількох запитів без блокування ресурсів, що особливо важливо у випадках, коли потрібно працювати з великим обсягом інформації або виконувати складні завдання, такі як оцінка відповідей кількох кандидатів одночасно.

Асинхронна архітектура реалізована за рахунок використання спеціалізованих модулів, які дозволяють керувати чергами завдань, моніторити їх виконання та обробляти результати у паралельному режимі. Це дозволило уникнути проблем із блокуванням ресурсів, які зазвичай виникають у синхронних системах, та значно скоротити час виконання кожного запиту. Завдяки цьому система здатна виконувати аналіз без помітного зниження продуктивності, навіть за умови високого навантаження або роботи з великою кількістю одночасних запитів.

Однією з ключових особливостей цієї архітектури є використання HTTP-клієнта та зовнішнього API для взаємодії з мовною моделлю. Замість локального запуску моделі, система звертається до потужних зовнішніх ресурсів через API, що забезпечує високу точність і гнучкість аналізу. Цей підхід дозволяє масштабувати систему без необхідності витрачати значні ресурси на локальне розгортання великих мовних моделей. Наприклад, під час аналізу відповіді кандидата система може виконати кілька послідовних чи паралельних викликів до зовнішнього API для різних аспектів оцінки, таких як аналіз впевненості у відповіді, технічних знань або формування підсумкового звіту [42].

Завдяки такому підходу система може обробляти одночасно десятки чи навіть сотні запитів, що робить її особливо ефективною для використання у великих компаніях чи рекрутингових агентствах, які щодня працюють із багатьма кандидатами. Наприклад, під час великого відбору система може одночасно аналізувати відповіді кількох кандидатів, формуючи їхні профілі й зберігаючи результати в базі даних. Це забезпечує не лише швидкість, а й надійність, оскільки всі процеси чітко синхронізовані та не впливають один на одного.

Для збереження результатів аналізу використовується база даних, яка була спроєктована спеціально для підтримки швидкого доступу та ефективної обробки великих обсягів інформації. У цій базі даних зберігаються всі ключові аспекти оцінки: від основних показників, таких як цифрові оцінки навичок, до детальної інформації, отриманої від кожного плагіна. Крім того, зберігаються порівняння відповідей із вимогами вакансії та підсумкові звіти. Така структура дозволяє вести історію кожного інтерв'ю, що забезпечує прозорість процесу оцінки й дозволяє використовувати результати повторно. Наприклад, результати можуть бути використані для аналізу динаміки розвитку кандидата, якщо він проходить кілька етапів відбору, або для порівняння між кількома кандидатами, які претендують на одну й ту саму посаду.

Ця архітектура також легко масштабується для роботи з численними вакансіями й великою кількістю кандидатів. Асинхронний підхід і використання зовнішніх API у поєднанні зі структурованою базою даних гарантують, що навіть у випадках інтенсивного навантаження система залишається стабільною, швидкою й ефективною. Це робить її універсальним рішенням для сучасних процесів аналізу відповідей на інтерв'ю та автоматизації рекрутингу.

Для забезпечення надійного збереження даних, пов'язаних із практичною частиною роботи, а також для управління та збереження результатів аналізу відповідей кандидатів, була використана база даних із визначеною структурою. Схема бази даних була спроєктована таким чином, щоб забезпечити зручне зберігання, швидкий доступ та ефективну обробку інформації. У цій базі даних зберігаються результати роботи семантичних плагінів, зокрема ключові аспекти оцінок, витягнута інформація, порівняння з вимогами вакансії та підсумкові звіти.

Завдяки такій організації даних система отримала можливість вести історію кожного інтерв'ю, забезпечуючи прозорість процесу оцінки. Це не лише полегшує доступ до результатів, але й дозволяє використовувати їх повторно для аналізу динаміки кандидата на наступних етапах або для порівняння з іншими претендентами. Додатково, структура бази даних була адаптована для роботи з великими обсягами інформації, що є важливим для масштабування системи в умовах обробки даних для численних вакансій та кандидатів.

3.4 Хмарні сховища та методи семантичного пошуку в них

У рамках свого дослідження я зібрав і систематизував навчальні матеріали, які включали книги по C#, а також ресурси з порадами та типовими запитаннями для інтерв'ю. Ці матеріали стали основою для побудови бази знань, яка використовувалася в системі аналізу кандидатів. Основна ідея полягала в тому, щоб забезпечити доступ до високоякісної та релевантної інформації, яка могла б підтримати процес оцінки відповідей кандидатів під час інтерв'ю.

Навчальні книги по C# були вибрані для надання глибокого технічного контексту. Вони охоплювали ключові аспекти мови програмування, включаючи основи синтаксису, об'єктно-орієнтоване програмування, роботу з базами даних, асинхронність, оптимізацію коду та інші важливі теми. Ці матеріали забезпечили систему знаннями, необхідними для оцінки технічних відповідей кандидатів, їхнього розуміння теоретичних концепцій і вміння застосовувати знання на практиці.

Окрім технічних матеріалів, я зібрав ресурси з порадами щодо підготовки до інтерв'ю та типових запитань. Це включало як загальні рекомендації, так і конкретні приклади завдань або ситуаційних питань, які часто задаються на співбесідах для розробників. Ці дані дозволили системі аналізувати відповіді не лише з технічної точки зору, але й з точки зору комунікаційних навичок, логіки мислення та здатності кандидата відповідати на нестандартні запитання.

На початковому етапі роботи з матеріалами книги по C# та ресурси з порадами для інтерв'ю були завантажені в Azure Blob Storage. Це рішення забезпечило зручне та безпечне зберігання великих текстових файлів у хмарі, створивши централізоване сховище для обробки та подальшого аналізу. Azure Blob Storage був обраний через його інтеграцію з іншими інструментами Microsoft Azure, зокрема Azure AI Search, а також через його високу продуктивність і масштабованість (рис. 3.4). Завантаження книг у це сховище стало першим кроком у створенні індексу для ефективного пошуку.

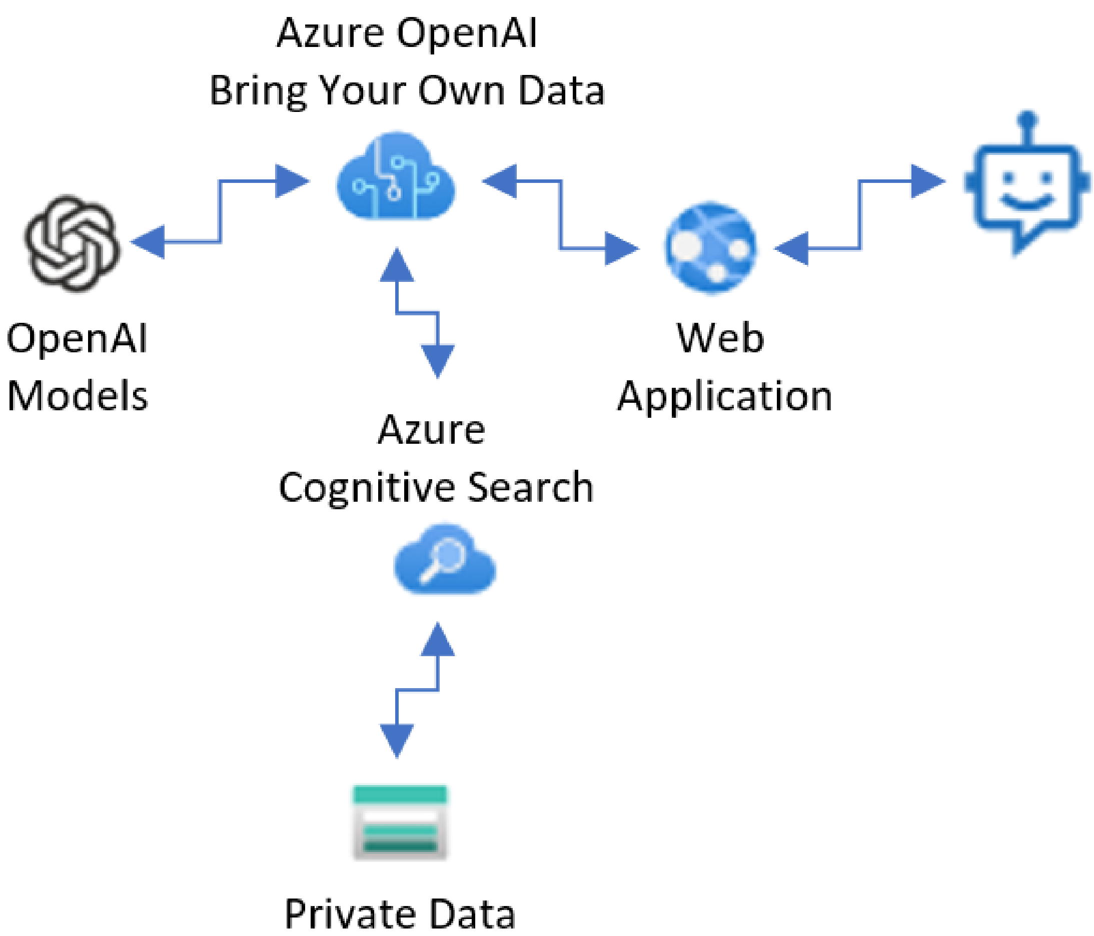

Рисунок 3.4 – Інфраструктура Azure AI Search

Зберігання даних у Blob Storage дозволило організувати зручну структуру файлів, де кожна книга чи матеріал представляли окремий об'єкт. Завдяки цьому було легко ідентифікувати кожен ресурс за метаданими, такими як назва, автор чи категорія. Це забезпечило простоту інтеграції з

Azure AI Search, оскільки дані зі сховища могли бути безпосередньо проіндексовані. Крім того, використання Blob Storage гарантувало, що навіть великі текстові файли могли бути швидко оброблені та підготовлені до подальшого розбиття на менші фрагменти для індексації [43].

Після завантаження файлів у Blob Storage була налагоджена інтеграція з Azure AI Search, яка дозволила автоматизувати процес створення індексу. Сховище виступало джерелом для отримання даних, які потім були оброблені інструментами Azure AI Search для побудови векторного індексу. Завдяки цьому вдалося забезпечити не лише зручний доступ до матеріалів, але й гнучкий механізм пошуку, що враховував семантичну релевантність інформації та зберігав контекст між окремими частинами текстів.

Зібрані матеріали були завантажені до бази знань через Azure AI Search. Книги та ресурси були спочатку розділені на логічні фрагменти, які векторизувалися для забезпечення семантичного пошуку. Наприклад, книги по C# були поділені на глави та ключові розділи, щоб кожна частина могла бути швидко знайдена за відповідним запитом. Поради та запитання для інтерв'ю були структуровані у вигляді набору тематичних блоків, що дозволяло знаходити рекомендації чи приклади завдань, релевантних до запитань кандидата.

Розбиття цих матеріалів через інструмент Skills забезпечило збереження контексту між фрагментами. Це гарантувало, що під час пошуку відповіді система могла знайти не лише конкретну сторінку, але й усі пов'язані фрагменти, що стосувалися того ж питання. Наприклад, якщо у відповіді кандидата згадувалася асинхронність у C#, система могла надати розділи книги, що описували цю тему, разом із прикладами коду та поясненнями.

Цей підхід дозволив мені створити ефективну базу знань, яка інтегрувалася в систему аналізу інтерв'ю, забезпечуючи релевантність, точність і глибину аналізу відповідей кандидатів.

У процесі роботи над локальною базою знань я реалізував Indexer та Index, які стали центральними елементами для налаштування ефективного пошуку релевантної інформації. Indexer був створений для автоматичної обробки зібраних матеріалів, таких як книги по C# та поради щодо інтерв'ю. Приклад яким чином релізовний Index приведено на (рис. 3.5).

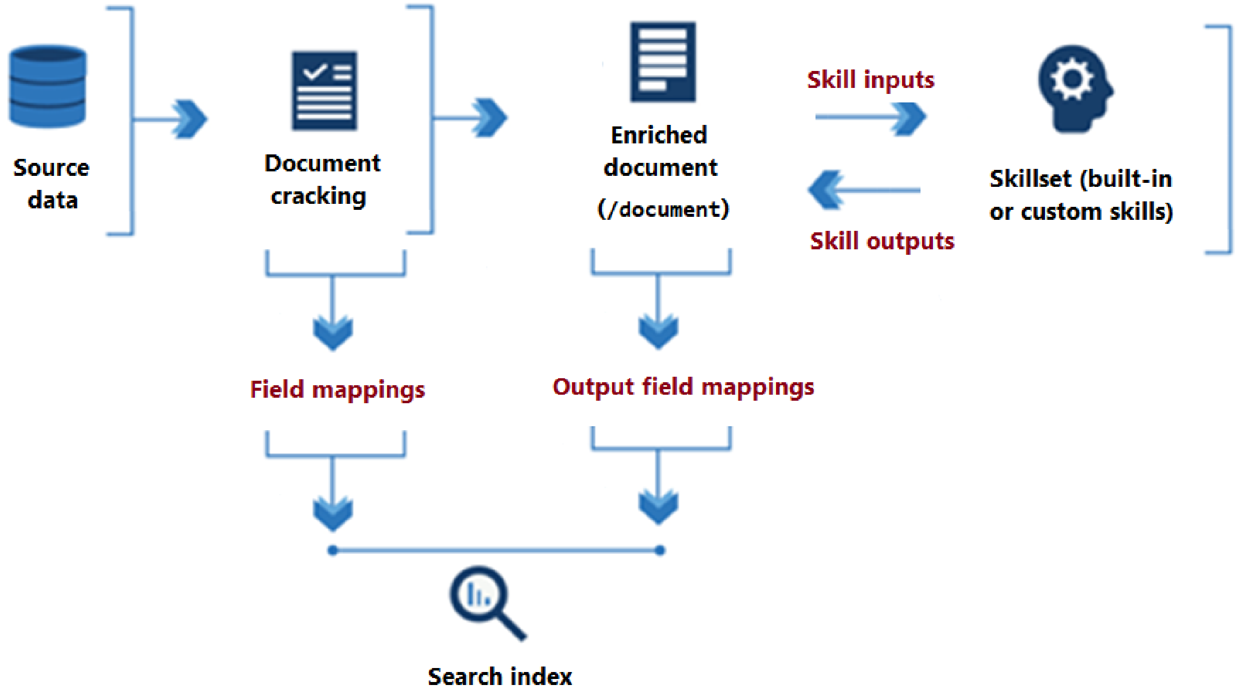

Рисунок 3.5 – Принцип створення індексу на основі існуючого контейнера

Він забезпечував розбиття великих текстових обсягів на логічно структуровані частини, наприклад, сторінки чи тематичні розділи, і перетворював їх у векторні представлення. Ці вектори слугували для семантичного пошуку, дозволяючи знаходити інформацію не тільки за ключовими словами, а й за її змістом і контекстом.

Index, у свою чергу, зберігав проіндексовані дані у формі, яка забезпечувала швидкий і точний доступ до них. Кожен фрагмент тексту, доданий до індексу, містив не лише свій оригінальний вміст, а й унікальний ідентифікатор, метадані (такі як назва книги, номер сторінки чи розділ) і векторне представлення. Це дозволяло системі оперативно знаходити необхідні дані, зберігаючи контекст між різними частинами тексту. Наприклад, під час пошуку інформації про асинхронність у C#, система могла знайти як окремі сторінки з поясненнями, так і всі пов’язані фрагменти з іншими деталями.

Механізм пошуку був реалізований на базі Azure AI Search, що забезпечило гнучкість і точність запитів. Завдяки цьому система могла враховувати семантичну релевантність інформації, розуміючи зміст запиту, а не лише шукаючи точні збіги слів. Окрім того, пошук підтримував фільтрацію за певними параметрами, такими як тип матеріалу або тематичний розділ, що робило його ще більш ефективним.

Ця реалізація значно покращила швидкість і якість роботи з великими обсягами текстових даних. Використання Indexer та Index дозволило забезпечити миттєвий доступ до знань, які зберігалися в локальній базі, і гарантувало, що навіть за розбиття тексту на дрібні фрагменти контекст залишався цілісним. У підсумку, система могла не лише швидко знаходити релевантну інформацію, але й використовувати її у правильному контексті для підтримки процесу аналізу відповідей кандидатів.

Практичне застосування створених ресурсів на базі Azure демонструє, як система може ефективно використовувати локальну базу знань для підтримки процесу аналізу кандидатів під час інтерв’ю. Наприклад, коли кандидат дає відповідь на питання, пов’язане з певною навичкою, наприклад Dependency Injection у C#, система автоматично звертається до бази знань для пошуку релевантної інформації.

При аналізі відповіді система надсилає запит до Azure AI Search, який виконує пошук у проіндексованих даних із завантажених книг. В результаті база знань повертає всі сторінки з матеріалів, які містять інформацію про Dependency Injection, включаючи фрагменти з різних книг. Наприклад, це можуть бути глави, які описують принципи роботи Dependency Injection, приклади його використання та рекомендації щодо оптимізації цього патерну у C#. Azure AI Search додатково оцінює релевантність кожного знайденого фрагмента, присвоюючи йому числовий індекс релевантності.

Для забезпечення точності оцінки відповіді кандидата система відбирає лише ті матеріали, які мають індекс релевантності 8 або вище. Це гарантує, що використовується лише найважливіша й найбільш відповідна інформація. Крім того, щоб уникнути перевантаження аналізатора великою кількістю тексту, до обробки передаються лише перші три найбільш релевантні матеріали. Наприклад, це можуть бути:

– глави з однієї книги, що детально описують теоретичні основи Dependency Injection;

– практичні приклади його впровадження з іншого джерела;

– короткий посібник із порадами щодо найкращих практик.

Знайдені матеріали стають контекстом для подальшого аналізу. Наприклад, відповіді кандидата можуть бути порівняні з ключовими аспектами, які викладаються в цих книгах. Якщо кандидат згадує, як використовував Dependency Injection у своєму проєкті, система перевіряє його відповідь на основі знайдених даних, підтверджуючи, чи враховані важливі деталі, такі як правильне використання контейнерів залежностей або дотримання принципів інверсії управління.

Цей підхід гарантує, що аналіз відповіді кандидата базується на високоякісних, релевантних даних. Система забезпечує глибокий контекст для оцінки, дозволяючи інтерв’юерам отримувати точну й обґрунтовану інформацію про рівень знань кандидата. Завдяки обмеженню на релевантність (індекс $\geq$ 8) та вибору лише перших трьох матеріалів аналіз стає точним, швидким і зрозумілим для кінцевого користувача.

Після того, як система отримує три найбільш релевантні матеріали з бази знань, вони інтегруються у процес аналізу, створюючи повноцінний контекст для оцінки відповіді кандидата. Ці матеріали можуть бути використані як частина кількох етапів оцінювання: від первинної перевірки відповідності відповіді кандидата до створення детального звіту для інтерв’юера.

На першому етапі система порівнює відповідь кандидата з основними поняттями й підходами, представленими у знайдених матеріалах. Наприклад, якщо кандидат пояснює принципи роботи Dependency Injection, система перевіряє, чи згадуються ключові концепції, такі як інверсія управління, реєстрація залежностей у контейнері та впровадження залежностей через конструктори чи методи. У разі відповідності система може автоматично позначити, що кандидат добре орієнтується в основах цієї теми.

На другому етапі аналізуються приклади використання. Якщо знайдені матеріали містять приклади коду або рекомендації щодо реалізації Dependency Injection, система може перевірити, чи відповідає відповідь кандидата найкращим практикам. Наприклад, у випадку, якщо кандидат наводить власний приклад, система порівнює його з теоретичними рекомендаціями або реальними прикладами, знайденими у книгах. Це дозволяє оцінити, наскільки відповідь кандидата є не лише правильною, але й практично застосовною.

Третій етап передбачає створення підсумкового звіту. На основі знайдених матеріалів і аналізу відповіді система формує рекомендації щодо подальшої оцінки кандидата. Наприклад, якщо відповідь була неповною або містила неточності, система може запропонувати додаткові запитання, які допоможуть глибше зрозуміти рівень знань кандидата. У разі, якщо відповідь була повною й обґрунтованою, звіт може містити висновок про відповідність кандидата вимогам до вакансії щодо знань Dependency Injection.

Особливо важливим аспектом цього підходу є те, що контекст, отриманий із бази знань, не втрачається. Навіть якщо матеріали були розбиті на окремі сторінки чи розділи, система забезпечує збереження їхнього взаємозв’язку. Це дозволяє аналізувати відповіді кандидата з урахуванням усіх аспектів теми, включно з деталі, які можуть бути розподілені між різними джерелами. Таким чином, інтерв’юер отримує доступ до повноцінної картини, яка допомагає приймати обґрунтовані рішення щодо кожного кандидата.

Оскільки запитання для інтерв'ю також були проіндексовані та збережені у вигляді векторної бази, система Semantic Kernel отримала можливість легко передавати контекст даних для їх обробки. Це стало можливим завдяки тому, що і запитання, і інші матеріали бази знань (книги, поради, приклади) були представлені в однаковому форматі векторного представлення (*embeddings*), що дозволяє системі ефективно працювати з усією інформацією.

Використання єдиної моделі для створення embedding забезпечило уніфікованість даних. Коли кандидат дає відповідь під час інтерв'ю, текст цієї відповіді також обробляється через ту саму модель для векторизації. Завдяки цьому система може швидко порівнювати векторні представлення відповіді з векторними представленнями запитань і матеріалів у базі знань. Такий підхід дозволяє зберігати контекст у процесі аналізу та уникати втрати семантичної інформації.

Зокрема, це забезпечує можливість точної ідентифікації запитань або частин бази знань, які найбільш відповідають змісту відповіді кандидата. Наприклад, якщо кандидат говорить про Dependency Injection, Semantic Kernel зіставляє вектор його відповіді з векторами запитань про цю тему. Це дозволяє швидко визначити, які запитання найкраще співвідносяться з відповіддю, а також знайти матеріали з бази знань, які підтримують або уточнюють цю тему.

Ключова перевага цього підходу полягає у тому, що весь контекст обробляється на одному рівні – у вигляді векторних представлень, що є природним для системи. Це означає, що немає потреби додатково трансформувати або адаптувати дані, оскільки і текст відповідей, і матеріали бази знань вже мають спільну семантичну основу. Завдяки цьому система може не лише швидко знаходити релевантні матеріали, але й забезпечувати їхню інтеграцію у процес аналізу відповіді в режимі реального часу.

Таке узгодження формату embedding також сприяє тому, що будь-яка нова інформація, додана до бази знань (наприклад, нові запитання або додаткові матеріали), одразу стає доступною для семантичного пошуку й аналізу. Це значно спрощує масштабування системи та забезпечує її гнучкість при роботі з динамічно змінюваними даними. У підсумку, система не лише ефективно обробляє текст інтерв'ю, але й робить це з урахуванням усіх релевантних аспектів контексту, представлених у базі знань.

Однією з ключових переваг використання єдиного формату векторних представлень є здатність системи Semantic Kernel підтримувати високий рівень інтеграції між усіма елементами бази знань та відповідями користувачів. Це не лише забезпечує більш швидкий і точний пошук інформації, але й дозволяє глибше аналізувати контекст кожного інтерв'ю в реальному часі. Коли відповідь кандидата перетворюється у векторне представлення, система може безпосередньо порівнювати її не тільки з базою знань, але й із попередніми відповідями кандидата чи інших кандидатів на аналогічні запитання. Це дозволяє визначати не лише правильність і повноту відповіді, а й її унікальність і якість відносно загальної бази даних.

Ще одним важливим аспектом є можливість зберігання зв'язків між відповідями, запитаннями та матеріалами бази знань. Завдяки тому, що кожен елемент індексується у вигляді векторів, система може не лише знаходити релевантні сторінки чи фрагменти тексту, але й будувати семантичні зв'язки між ними. Наприклад, якщо відповідь кандидата на запитання про Dependency Injection містить згадку інших концепцій, таких як Service Lifetime або Singleton Pattern, система може автоматично знайти відповідні матеріали з бази знань, навіть якщо ці терміни не були явно згадані в оригінальному запитанні.

Це створює ефект семантичної мережі, де кожна відповідь, запитання чи матеріал бази знань є вузлом, який пов'язаний із іншими вузлами на основі їхнього контекстуального змісту. Це не лише підвищує точність аналізу, але й дозволяє інтерв'юеру отримати додатковий контекст або уточнення без необхідності вручну шукати інформацію. Система автоматично інтегрує ці дані в процес аналізу, забезпечуючи глибокий і повноцінний розгляд відповіді кандидата.

Крім того, така архітектура значно покращує масштабованість системи. Додавання нових матеріалів до бази знань або оновлення існуючих не вимагає складних маніпуляцій, оскільки всі дані автоматично трансформуються у векторні представлення за допомогою тієї ж моделі embedding. Це дозволяє легко інтегрувати нові джерела знань, наприклад, додаткові книги, статті чи приклади, і робить систему адаптивною до змін у технологічному чи професійному середовищі.

У підсумку, використання єдиної моделі embedding для всіх елементів системи забезпечує не лише ефективність і точність роботи, але й глибоке розуміння контексту. Це дозволяє системі Semantic Kernel бути не просто інструментом для автоматизації, а повноцінним асистентом, здатним інтелектуально підтримувати процес оцінки кандидатів у реальному часі.

3.5 Аналіз результатів оцінок інтерв’ю в дослідженні

Аналіз інтерв’ю є важливим етапом дослідження, адже він дозволяє оцінити не лише якість відповідей кандидатів, але й роботу системи Semantic Kernel у контексті автоматизованої обробки даних, релевантності бази знань і здатності семантичних плагінів коректно інтерпретувати контекст.

Особливу увагу приділено тому, як система допомагає оцінювати різні аспекти компетенцій кандидатів: від технічних знань до розуміння концепцій та здатності мислити критично. Під час аналізу враховувалася релевантність знайдених матеріалів, точність відповідей кандидатів і відповідність їхніх знань вимогам вакансії. Завдяки структурованому підходу до обробки інтерв’ю були отримані дані, які дають змогу оцінити ефективність розробленої системи та запропонувати напрямки для її подальшого вдосконалення.

Для оцінки ефективності роботи системи Semantic Kernel було створено контрольну вибірку, яка включала транскрибовані інтерв'ю кандидатів. Ця вибірка попередньо була оцінена експертом із розробки на C# за допомогою тієї ж цифрової шкали, яка використовується системою для оцінки. Експерт не лише виставляв оцінки за ключовими технічними навичками кандидатів, але й надавав рецензії, де детально описував сильні та слабкі сторони відповідей, що наведено в (табл. 3.1). Це дало змогу створити еталонні дані, які використовувалися для порівняння з автоматичними результатами системи.

Таблиця 3.1 – Приклад результатів методу голосування

Навичка	C#	LINQ	DI	Azure	JS	Communication
Реальна оцінка	5	4	3	4	5	2
Отримана оцінка	4	4	2	5	5	0

Ті самі транскрипти інтерв'ю були оброблені системою Semantic Kernel за допомогою реалізованих семантичних плагінів. Система аналізувала відповіді кандидатів у кількох вимірах: технічні знання, розуміння концепцій, впевненість і здатність критично мислити. Для кожного транскрипту було сформовано оцінку на основі запрограмованих метрик і створено деталізовані звіти, які включали не лише цифрові оцінки, але й додаткові висновки, сформовані на основі релевантних матеріалів із бази знань.

Результати роботи системи були порівняні з оцінками та рецензіями, наданими експертом із розробки на C#, які охоплювали ключові навички, такі як комунікативні здібності, знання мови C#, розуміння платформи CLR, робота з LINQ, Entity Framework та Azure. Для проведення порівняння використовувалася метрика Ейлера, яка дозволила врахувати багатовимірний характер даних, зокрема різні аспекти компетенцій, оцінені як системою, так і експертом. Аналіз порівняння між оцінками системи та експерта показав, наскільки автоматизований підхід здатний точно відтворити людську оцінку, відображаючи як загальні тенденції, так і тонкі відмінності в оцінці конкретних навичок (табл. 3.2).

В якості результатів було представленно евклідову відстань між оцінками вибірок. На основі цієї відстані було вирахуванно відсоток схожості який вираховується як різниця відношення евклідової відстані до її максимального значення у процентному представленні.

Таблиця 3.2 – Порівняння результатів оцінки інтерв'ю між вхідною та контрольною вибірками

Транскрибції	Евклідова відстань	Відсоток схожості	Кількість навичок	6
Інтерв'ю 1	2,6458	78,40%	Максимальний розбіг між оцінками	5
Інтерв'ю 2	2,2361	81,74%	Максимальна евклідова відстань	12,2474
Інтерв'ю 3	0	100,00%		
Інтерв'ю 4	1	91,84%		
Інтерв'ю 5	2	83,67%		

Особливий акцент було зроблено на тому, як система обробляє контекст відповідей. Наприклад, аналіз релевантності матеріалів, знайдених у базі знань, дозволив оцінити, чи допомагає система правильно інтерпретувати відповіді кандидатів, навіть якщо вони були неповними або неконкретними.

Також враховувалася здатність системи фокусуватися на ключових аспектах, які відзначив експерт у своїй рецензії, таких як розуміння специфічних концепцій (наприклад, Dependency Injection у C#).

Результати порівняння дозволили оцінити сильні сторони та обмеження системи. У тих випадках, де кореляція з експертними оцінками була високою, це свідчило про ефективність семантичних плагінів і бази знань для аналізу технічних відповідей. Натомість у випадках розбіжностей було виявлено аспекти, які потребують доопрацювання, наприклад, поліпшення обробки складних відповідей із прихованим контекстом або адаптація системи до більш вузькоспеціалізованих запитів.

Такий підхід до аналізу результатів не лише підтвердив ефективність системи, але й вказав на можливості для її подальшого вдосконалення. Завдяки цьому вдалося створити не лише інструмент для автоматизації аналізу інтерв'ю, але й гнучку платформу, яка може адаптуватися до потреб різних користувачів і завдань.

На основі отриманих результатів, представлених у таблицях, можна зробити наступні висновки щодо роботи системи Semantic Kernel та її порівняння з експертними оцінками. Результати показують, що система в цілому здатна відтворювати оцінки експерта з високим рівнем точності. Наприклад, у таблиці з транскрипціями видно, що найвища схожість між системою і експертом становить 100% (Інтерв'ю 3), що свідчить про ідеальне співпадіння оцінок. У більшості випадків рівень схожості перевищує 80%, що підтверджує ефективність системи у відтворенні людської оцінки за визначеними навичками.

Є випадки, коли система демонструє певні розбіжності з експертними оцінками. Наприклад, для Інтерв'ю 1 евклідова відстань становить 2,6458, що відповідає схожості 78,40%, а для Інтерв'ю 2 схожість сягає 81,74% з евклідовою відстанню 2,2361. Це показує, що система іноді має труднощі з коректною оцінкою певних аспектів навичок, таких як Communication (де оцінка системи становить 0, тоді як експерт оцінив її у 2). Це може свідчити про необхідність доопрацювання алгоритмів оцінки soft skills, які складніше виміряти через їх суб'єктивність.

Таблиця вказує на максимальний розбіг між оцінками в 5 балів для деяких навичок, що може бути спричинено різницею в інтерпретації системи та експерта певних відповідей або відсутністю чіткого контексту для оцінки. Наприклад, для навички Communication система не змогла адекватно оцінити рівень кандидата, що вказує на необхідність покращення методів аналізу невербальних аспектів і контекстуальних відповідей.

У середньому, система демонструє високий рівень відповідності експертним оцінкам. Більшість транскрипцій мають схожість понад 80%, що є прийнятним показником для автоматизованої системи оцінки. Середні значення евклідової відстані знаходяться в межах 1-2 балів, що свідчить про стабільність системи у більшості випадків.

Система показує високу точність в оцінці технічних навичок, таких як C#, LINQ, DI, Azure і JS. Оцінки системи здебільшого відповідають оцінкам експерта, що підтверджує релевантність бази знань і ефективність роботи семантичних плагінів для аналізу технічних аспектів. Додатковою перевагою є масштабованість системи: завдяки асинхронній архітектурі та автоматизованій обробці можливо аналізувати кілька інтерв'ю одночасно без втрати точності. Високий рівень відповідності результатів експертним оцінкам також підтверджується метрикою евклідової відстані, що свідчить про чітку кореляцію між автоматизованим аналізом і людською оцінкою.

Водночас, аналіз виявив і певні слабкі сторони системи, які потребують вдосконалення. Найбільші розбіжності спостерігаються в оцінці комунікативних навичок (soft skills). Це може бути пов'язано з недостатнім врахуванням контексту або відсутністю спеціалізованих моделей для аналізу цих аспектів. У складних чи багатозначних відповідях система іноді недооцінює важливість контексту, що призводить до розбіжностей у баллах між експертною та автоматизованою оцінками. Окрім того, для уникнення значного розбігу між оцінками різних навичок, доцільно оптимізувати вагові коефіцієнти для кожної категорії навичок, враховуючи їхню важливість для конкретної ролі.

Система Semantic Kernel демонструє високу точність та ефективність у більшості випадків, що підтверджується результатами порівняння з експертними оцінками. Водночас, слабкі сторони в аналізі soft skills і контекстуальних відповідей вказують на можливі напрямки вдосконалення. Загалом, підхід довів свою ефективність і є придатним для використання в автоматизованій оцінці інтерв'ю, особливо для технічних спеціалістів, із потенціалом для подальшого розвитку.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений метод аналізу інтерв'ю, який базується на використанні сучасних мовних моделей та алгоритмів текстової обробки. Метод включає автоматичну трансформацію аудіозаписів у текст, попередню обробку даних, маркування ролей спікерів (рекрутер, кандидат) та проведення семантичного аналізу.

Результати дослідження підтвердили ефективність запропонованого підходу у покращенні якості аналізу інтерв'ю. Система дозволяє автоматизувати процеси, що раніше виконувалися вручну, значно скорочуючи час і ресурси, необхідні для обробки даних. Зокрема, впроваджені алгоритми показали високу точність у визначенні ключових аспектів спілкування, таких як аналіз відповідності відповідей кандидата вимогам посади та виявлення емоційних тональностей.

Запропонований метод забезпечує можливість адаптації до різних типів інтерв'ю, завдяки чому його можна застосовувати в різноманітних сферах, таких як рекрутинг, навчання персоналу та дослідження поведінкових патернів. Отримані результати відкривають перспективи для подальшого вдосконалення системи, зокрема, інтеграції з іншими інструментами для аналітики великих даних.

Результати роботи апробовано у вигляді тез доповідей під час XXVIII Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [44].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Shamshiri, A., Ryu, K. R., & Park, J. Y. (2024). Text mining and natural language processing in construction. *Automation in Construction*, *158*, 105200.
2. Chen, S., Zhang, Y., & Yang, Q. (2024). Multi-task learning in natural language processing: An overview. *ACM Computing Surveys*, *56*(12), 1-32.
3. Верколаб, Г., & Кузьомін, О. (2023). ДОСЛІДЖЕННЯ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ МОЖЛИВОСТЕЙ ІНФЕРЕНСА ДЛЯ LLM ЗА ДОПОМОГОЮ NVIDIA TRITON. *UNIVERSUM*, (3), 148-156.
4. Iosifov, I., & Sokolov, V. Y. (2024). Методи аналізу природної мови та застосування нейронних мереж в кібербезпеці. *Кібербезпека: освіта, наука, техніка*, *4*(24), 398-414.
5. Alto, V. (2023). *Modern Generative AI with ChatGPT and OpenAI Models: Leverage the capabilities of OpenAI's LLM for productivity and innovation with GPT3 and GPT4*. Packt Publishing Ltd. pp. 11-12
6. De Paoli, S. (2024). Performing an inductive thematic analysis of semi-structured interviews with a large language model: An exploration and provocation on the limits of the approach. *Social Science Computer Review*, *42*(4), 997-1019.
7. Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., & Chao, L. S. (2019). Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*.
8. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
9. Baktash, J. A., & Dawodi, M. (2023). Gpt-4: A review on advancements and opportunities in natural language processing. *arXiv preprint arXiv:2305.03195*.
10. Wang, C., Xu, Y., Peng, Z., Zhang, C., Chen, B., Wang, X., ... & An, B. (2023). keqing: knowledge-based question answering is a nature chain-of-thought mentor of LLM. *arXiv preprint arXiv:2401.00426*.

11. Liu, J., Yu, C., Gao, J., Xie, Y., Liao, Q., Wu, Y., & Wang, Y. (2023). Llm-powered hierarchical language agent for real-time human-ai coordination. *arXiv preprint arXiv:2312.15224*.

12. Wang, C., Tang, Y., Ma, X., Wu, A., Popuri, S., Okhonko, D., & Pino, J. (2020). Fairseq S2T: Fast speech-to-text modeling with fairseq. *arXiv preprint arXiv:2010.05171*.

13. Borg, M., Alégroth, E., & Runeson, P. (2017, May). Software engineers' information seeking behavior in change impact analysis-an interview study. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)* (pp. 12-22). IEEE.

14. Johanssen, J. O., Kleebaum, A., Paech, B., & Bruegge, B. (2018, May). Practitioners' eye on continuous software engineering: an interview study. In *Proceedings of the 2018 international conference on software and system process* (pp. 41-50).

15. Pan, J. J., Wang, J., & Li, G. (2024). Survey of vector database management systems. *The VLDB Journal*, *33*(5), 1591-1615.

16. Gorokhovatskyi V., Gadetska S., Stiahlyk N. (2020) Image structural classification technologies based on statistical analysis of descriptions in the form of bit descriptor set. In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020), 2608, pp. 1027-1039.

17. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, Computers, Materials & Continua, vol. 80, no. 2, pp. 3085-3106.

18. Gorokhovatskyi, V., Stiahlyk, N., Mazur, Y., Vechirska, A. (2024) Способи метричної грануляції для опису зображень у задачі класифікації. Системи управління, навігації та зв'язку. Збірник наукових праць, 3(77), 106-112.

19. Maddy, A. (2024). Integrating AI Services into Semantic Kernel: A Case Study on Enhancing Functionality with Google PaLM and Large Language Models. *Transactions on Open Source Software Projects*, *1*(1).

20. Soh, J., & Singh, P. (2024). Semantic Kernel, Plugins, and Function Calling. In *Data Science Solutions on Azure: The Rise of Generative AI and Applied AI* (pp. 191-221). Berkeley, CA: Apress.

21. Sun, Z., Jiang, H., Wang, D., Li, X., & Cao, J. (2023). Safl-net: Semantic-agnostic feature learning network with auxiliary plugins for image manipulation detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 22424-22433).

22. Han, Y., Liu, C., & Wang, P. (2023). A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*.

23. Zhao, X., Zhou, X., & Li, G. (2024). Chat2Data: An Interactive Data Analysis System with RAG, Vector Databases and LLMs. *Proceedings of the VLDB Endowment*, *17*(12), 4481-4484.

24. Pan, J. J., Wang, J., & Li, G. (2024, June). Vector Database Management Techniques and Systems. In *Companion of the 2024 International Conference on Management of Data* (pp. 597-604). Gadetska, S.V., Gorokhovatskyi, V. O., Stiahlyk, N. I., Vlasenko, N.V. Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points. Radio Electronics, Computer Science, Control, 2021, №4, pp. 58-68.

25. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. Advanced Information Systems, 5(4), pp. 10-16.

26. Tvoroshenko I.S., and Gorokhovatsky V.O. (2019) Intelligent classification of biophysical system states using fuzzy interval logic, Telecommunications and Radio Engineering, 78(14), pp. 1303–1315.

27. Gorokhovatsky, V. (2014), Structural Analysis and Intellectual Data Processing in Computer Vision, SMIT, Kharkiv.

28. V. Gorokhovatsky, Y. Putyatin and V. Stolyarov (2017) Research of Effectivenss of Structural Image Classification Methods using Cluster Data Model, Radio Electronics Computer Science Control, vol. 3, no. 42, pp. 78-85.

29. Poppe, O., Castro, P., Lang, W., & Leeka, J. (2023). Proactive Resource Allocation Policy for Microsoft Azure Cognitive Search. *ACM SIGMOD Record*, *52*(3), 41-48.

30. Machiraju, S., Modi, R., Machiraju, S., & Modi, R. (2018). Azure cognitive services. *Developing Bots with Microsoft Bots Framework: Create Intelligent Bots using MS Bot Framework and Azure Cognitive Services*, 233-260.

31. Salvaris, M., Dean, D., & Tok, W. H. (2018). Deep learning with azure. *Building and Deploying Artificial Intelligence Solutions on Microsoft AI Platform, Apress*.

32. Dewan, H., & Hansdah, R. C. (2011, July). A survey of cloud storage facilities. In *2011 IEEE World Congress on Services* (pp. 224-231). IEEE.

33. Aman, S. S., N'guessan, B. G., Agbo, D. D. A., & Tiemoman, K. O. N. E. (2023). Search engine Performance optimization: methods and techniques. *F1000Research*, *12*.

34. Kale, V. (2019). *Parallel computing architectures and APIs: IoT big data stream processing*. Chapman and Hall/CRC.

35. Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, IEEE Access, vol. 12, pp. 73376-73385.

36. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, International Journal of Academic Engineering Research, 7(9), pp. 64-72.

37. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, 33 (1), 113-125.

38. Gorokhovatskyi, V., Gadetska, S., & Stiahlyk, N. (2023). Accelerating Image Classification based on a Model for Estimating Descriptor-to-Class Distance. International Journal of Computing, 22(4), 485-492.

39. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, 11, 126938-126949.

40. Tvoroshenko, I., & Zarivchatskyi, R. (2020). Analysis of existing methods for searching object in the video stream, in Proc. VI Int. Sci. Practic. Conf. «About the problems of science and practice, tasks and ways to solve them», Milan, pp. 500-505.

41. Gorokhovatskyi V.A., Zamula A.A. (2016) Employment of Intelligent Technologies in Multiparametric Control Systems. Telecommunications and Radio Engineering. Vol. 75, No 19, p. 1775–1785.

42. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., Vlasenko, N. (2023) Explanation of CNN Image Classifiers with Hiding Parts. In: J. Benois-Pineau, R. Bourqui, D. Petkovic, G. Quenot (eds), Explainable Deep Learning Artificial Intelligence, pp. 125-146, Academic Press, 346 p.

43. Gorokhovatskyi V., Gadetska S., Stiahlyk N. (2020) Image structural classification technologies based on statistical analysis of descriptions in the form of bit descriptor set. In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020), 2608, pp. 1027-1039.

44. Дебре В. (2024) Сучасні мовні моделі для аналізу тексту. 28-ий міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті», С. 39-40.