

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 303.732.4

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2025 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
зі спеціальності 124 «Системний аналіз»
на тему: «ГІБРИДНА РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ПІДБІРКИ ВАКАНСІЙ»

Виконав:
Студент 2 курсу, групи КА-42мп
Бабанов Дмитро Євгенович _____

Науковий керівник:
завідувач кафедри ММСА, к.т.н., доцент,
Тимощук Оксана Леонідівна _____

Консультант з нормоконтролю:
доктор філософії з сис. аналізу, асистент,
Канцедал Георгій Олегович _____

Рецензент:
д.т.н., доцент, провідний науковий співробітник
Інституту телекомунікацій і глобального
інформаційного простору НАН України
Олександр Терентьєв Миколайович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент: _____

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ

Завідувач кафедри ММСА

_____ Оксана ТИМОЩУК

«___» _____ 2025 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Бабанову Дмитру Євгеновичу

1. Тема дисертації: Гібридна рекомендаційна система для підбірки вакансій науковий керівник дисертації завідувач кафедри ММСА, к.т.н., доцент, Тимощук Оксана Леонідівна
затверджені наказом по університету від «06» листопада 2025 р. № 4837-с.
2. Строк подання студентом дисертації
16 грудня 2025 року.
3. Об'єкт дослідження: гібридні рекомендаційні системи для підбірки вакансій.
4. Предмет дослідження (Вихідні дані – для магістерської дисертації за ОПП) моделі контентних, колаборативних та гібридних моделей.
5. Перелік завдань, які потрібно розробити:
 - 1) огляд предметної області;
 - 2) вибір та аналіз інструментів для побудови рекомендаційних систем;
 - 3) розробка моделей рекомендаційних систем;
 - 4) поєднання моделей за допомогою гібридних методів;
 - 5) проведення експериментів для оцінки гібридної системи.
6. Перелік графічного (ілюстративного) матеріалу:
 - 1) рисунки;
 - 2) таблиці;
 - 3) презентація.

7. Орієнтовний перелік публікацій Бабанов Д.Є., Тимощук О.Л. Гібридна рекомендаційна система для підбірки вакансій. Системні науки та інформатика: збірка доповідей IV Всеукраїнської науково-практичної конференції «Системні науки та інформатика» 01-05 грудня 2025 року. Київ К., НН ІПСА КПІ ім. Ігоря Сікорського. 2025, 4 с.

8. Дата видачі завдання 1 вересня 2025 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації (МД)	Строк виконання етапів магістерської дисертації	Примітка
1	Затвердження теми МД. Ознайомлення зі структурою (МД) згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	01.09.2025-10.09.2025	виконано
2	Вибір напрямку для подальшого вивчення	11.09.2025-17.09.2025	виконано
3	Огляд літературних та інформаційних джерел. Аналіз предметної області	18.09.2025-01.10.2025	виконано
4	Розробка теоретичного узагальнення методу	02.10.2025-15.10.2025	виконано
5	Розробка програмного забезпечення.	16.10.2025-29.10.2025	виконано
6	Робота над практичним розділом магістерської дисертації.	30.10.2025-12.11.2025	виконано
7	Стартап-проект	13.11.2025-19.11.2025	виконано
8	Остаточне оформлення роботи	20.11.2025-15.12.2025	виконано

Студент

Дмитро БАБАНОВ

Науковий керівник дисертації

Оксана ТИМОЩУК

РЕФЕРАТ

Магістерська дисертація: 118 с., 28 рис., 20 табл., 2 додатки, 23 джерела.

РЕКОМЕНДАЦІЙНИ СИСТЕМИ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КОНТЕНТНА ФІЛЬТРАЦІЯ, ГІБРИДНІ МЕТОДИ, TF-IDF, BERT, ALS, ЗВАЖЕНИЙ ГІБРИД, МЕТРИКИ ОЦІНКИ ЯКОСТІ

Об'єкт дослідження – гібридні рекомендаційні системи для підбірки вакансій.

Мета роботи – розробка, реалізація та проведення експериментів над гібридною рекомендаційною системою.

Методи дослідження – аналіз наукової літератури, веб-скрапінг, експериментальне моделювання, порівняльний аналіз архітектур рекомендаційних систем, оцінка якості роботи системи за допомогою різноманітних експериментів над створеною системою рекомендацій.

У даній роботі проведено аналіз сучасних методів побудови рекомендаційних систем та досліджено контентні, колаборативні й гібридні підходи до генерації персоналізованих рекомендацій. Реалізовано та експериментально перевірено рекомендаційну систему для підбору вакансій на основі методів TF-IDF, BERT, матричної факторизації (ALS) та їх зваженого гібридного поєднання. Розроблено програмний продукт мовою Python з використанням бібліотек для аналізу даних і машинного навчання, створено синтетичний датасет взаємодій кандидатів із вакансіями та проведено оцінювання якості рекомендацій за стандартними метриками.

Отримані результати можуть бути використані для впровадження інтелектуальних систем рекомендацій у сервісах підбору персоналу та кар'єрних платформах.

ABSTRACT

Master's thesis: 118 pages, 28 figures, 20 tables, 2 appendices, 23 references.

RECOMMENDER SYSTEMS, COLLABORATIVE FILTERING, CONTENT-BASED FILTERING, HYBRID METHODS, TF-IDF, BERT, ALS, WEIGHTED HYBRID, QUALITY EVALUATION METRICS

Object of research – the design and development of hybrid recommender systems for job vacancy matching.

Purpose of the thesis – to develop, implement, and conduct experimental evaluation of a hybrid recommender system.

Research methods – analysis of scientific literature, web scraping, experimental modeling, comparative analysis of recommender system architectures, and evaluation of system performance using various experimental studies conducted on the developed recommendation system.

This thesis presents an analysis of modern approaches to building recommender systems and investigates content-based, collaborative, and hybrid methods for generating personalized recommendations. A recommender system for job vacancy matching based on TF-IDF, BERT, matrix factorization (ALS), and their weighted hybrid combination was implemented and experimentally evaluated. A software product was developed in Python using data analysis and machine learning libraries, a synthetic dataset of candidate–vacancy interactions was created, and the quality of recommendations was assessed using standard evaluation metrics

The obtained results can be applied to the implementation of intelligent recommender systems in recruitment services and career-oriented platforms.

ЗМІСТ

1 ДОСЛІДЖЕННЯ ПРОБЛЕМ СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ	10
1.1 Актуальність задачі	10
1.2 Історія рекомендаційних систем.....	11
1.3 Контентний метод	13
1.3.1 Огляд статті Content-based filtering algorithm in social media	14
1.4. Колаборативний метод	16
1.4.1 Огляд статті «A Distributed Collaborative Filtering Algorithm Using Multiple Data Sources».....	17
1.5 Гібридний метод.....	19
1.5.1 Огляд статті «Hybrid recommender system model for digital library from multiple».....	21
1.6 Задачі дослідження.....	22
1.7 Висновки до розділу 1	23
2 ОПИС МЕТОДІВ ТА МЕТРИК ГІБРИДНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ	25
2.1 Алгоритм побудованої моделі.....	25
2.2 Контентний метод фільтрації.....	27
2.2.1 TF-IDF метод	28
2.2.2 Семантичний метод фільтрації BERT	29
2.3 Косинусна подібність.....	32
2.4 Колаборативний метод фільтрації	33
2.4.1 Матрична факторизація	35
2.4.2 ALS.....	37
2.5 Гібридний метод: зважений гібрид	39
2.6 Метрики оцінки якості	41
2.7 Висновки до розділу 2.....	44
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ГІБРИДНОЇ СИСТЕМИ ТА ОГЛЯД ЕКСПЕРИМЕНТІВ.....	46
3.1 Опис та аналіз синтетичного датасету	46
3.2 Реалізація контентного методу фільтрації	49

3.3 Реалізація колаборативного методу фільтрації	52
3.4 Метод BERT та пошук семантичної подібності	54
3.5 Проблема холодного старту	57
3.6 Гібридний метод	59
3.7 Опис проведених експериментів	61
3.7.1 Експеримент точності на різних метриках якості	61
3.7.2 Експеримент з різною розмірністю навичок у кандидата	63
3.7.3 Експеримент з різними вагами	65
3.8 Висновки до розділу 3	67
4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ	69
4.1 План розробки стартапу та масштабування його на ринок	69
4.2 Опис ідеї стартап-проекту	71
4.3 Технологічний аудит ідеї проекту	72
4.4 Аналіз ринкових можливостей запуску стартап-проекту	76
4.5 Розроблення ринкової стратегії стартап-проекту	84
4.6 Розроблення маркетингової програми стартап-проекту	87
4.7 Висновки до розділу 4	90
ВИСНОВКИ	91
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	93
ДОДАТОК А	96

ВСТУП

Ще не так давно пошук роботи був процесом, який значною мірою залежав від особистих зв'язків, рекомендацій колег чи звернень до кадрових агентств. Рекрутери вручну аналізували резюме, а кандидати переглядали десятки оголошень у газетах або на веб-сайтах, намагаючись знайти вакансію, що відповідає їхнім навичкам і очікуванням. Така взаємодія забезпечувала певну персоналізацію, проте вимагала значних часових і людських ресурсів. Сьогодні, у добу домінації та розвитку платформ з працевлаштування, кількість доступних вакансій зросла у тисячі разів. Користувачі отримують миттєвий доступ до глобальних баз даних, але стикаються з новою проблемою – інформаційним перевантаженням і складністю вибору, відповідних вимогам, пропозицій. Попри автоматизацію процесів, головним завданням залишається точне визначення потреб користувача та відповідність вакансій його професійному профілю. Для вирішення цієї задачі все більшого поширення набувають гібридні рекомендаційні системи, що поєднують у собі різні підходи – контентно-орієнтований та колаборативний. Такі системи здатні одночасно враховувати характеристики вакансій, навички та досвід кандидата, а також поведінкові патерни інших користувачів із подібними інтересами. Завдяки цьому формується більш точна та персоналізована добірка пропозицій, яка відповідає індивідуальним потребам кожного шукача роботи. Метою роботи є створення гібридної рекомендаційної системи для підбору вакансій, що поєднує контентно-орієнтовані та колаборативні методи аналізу даних. Об'єктом дослідження виступають рекомендаційні системи у сфері пошуку роботи, а предметом – алгоритми формування персоналізованих добірок вакансій на основі аналізу профілю користувача та властивостей вакансій. У процесі дослідження застосовуються методи обробки природної мови NLP для аналізу описів вакансій і резюме, векторизація тексту TF-IDF, обчислення косинусної схожості, метод ALS для колаборативної фільтрації, метод BERT для

знаходження семантичної схожості між вакансіями та метод зважених гібридів для гібридного ранжування використаних методів. У роботі розглянуто сучасні підходи до побудови рекомендаційних систем у сфері працевлаштування, проведено аналіз алгоритмів контентного, колаборативного, семантичного та гібридного типу, розроблено та реалізовано прототип системи, яка формує персоналізовані рекомендації вакансій на основі профілю користувача та історії його взаємодії. Результатом роботи є програмна реалізація прототипу гібридної рекомендаційної системи, здатної формувати релевантні пропозиції вакансій для конкретного користувача. Практичне значення полягає у створенні інструменту, який може бути інтегрований у кар'єрні платформи, системи рекрутингу чи онлайн-сервіси з пошуку роботи для підвищення ефективності процесу працевлаштування. Отримані результати можуть бути використані для автоматизації підбору персоналу, підвищення точності рекомендацій, а також для створення інтелектуальних кар'єрних асистентів. Подальший розвиток системи передбачає врахування додаткових факторів, таких як геолокація, рівень заробітної плати, динаміка ринку праці або емоційне забарвлення опису вакансії.

Робота складається з вступу, чотирьох розділів, висновків, переліку джерел посилання та двох додатків. У першому розділі описано актуальність теми, проаналізовано види рекомендаційних систем і сучасні підходи до персоналізації вакансій; у другому викладено математичні основи методів аналізу текстових і поведінкових даних, описано принципи гібридизації алгоритмів; третій розділ присвячений огляду розробленої моделі рекомендаційної системи, також проведеним на її основі експериментам; четвертий розділ – стартап проект.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМ СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Актуальність задачі

У сучасному цифровому просторі, де користувачі щодня взаємодіють із величезними масивами даних, дедалі більш відчутною стає проблема інформаційного перенасичення. В умовах високої конкуренції між онлайн-платформами здатність забезпечити індивідуальний підхід до кожного користувача стає ключовим фактором успіху. Саме тому рекомендаційні системи набули стратегічного значення для сучасних цифрових сервісів, інтегруючись у сфери електронної комерції, медіа, освіти, соціальних мереж та рекрутингу. Їхнє головне завдання полягає у тому, щоб із безмежного потоку контенту виокремити ті елементи, які є найбільш релевантними для конкретного користувача, тим самим підвищуючи зручність, ефективність і задоволення від взаємодії з системою. Персоналізовані рекомендації вже давно довели свою ефективність – зокрема, понад 70% часу перегляду на YouTube формується завдяки алгоритмам рекомендацій, а Spotify утримує мільйони користувачів завдяки автоматично створеним добіркам на кшталт Discover Weekly, які дозволяють відкривати нову музику відповідно до індивідуальних смаків. Без подібних технологій користувачі часто втрачали б мотивацію користуватися платформою через складність пошуку релевантного контенту. З точки зору бізнесу рекомендаційні системи сприяють зростанню продажів, підвищенню середнього доходу компанії та рівня лояльності клієнтів, оскільки пропонують користувачеві саме ті товари, послуги чи контент, які його потенційно зацікавлять. Водночас розробка ефективних рекомендаційних систем супроводжується низкою технічних і концептуальних викликів. Необхідно враховувати проблеми розрідженості даних, коли бракує оцінок або історії взаємодій, а також ефект «холодного старту», що виникає для нових

користувачів чи об'єктів. Крім того, постає питання уникнення упередженості алгоритмів, які часто віддають перевагу популярним елементам, і забезпечення справедливості у формуванні рекомендацій. Для розв'язання цих проблем застосовуються методи матричної факторизації, гібридні алгоритми, глибинні нейронні мережі, а також додаткові джерела даних – наприклад, демографічні або поведінкові характеристики користувачів. Важливим аспектом залишається також етичний вибір: системи мають гарантувати конфіденційність, прозорість та різноманітність рекомендацій, не створюючи інформаційних «бульбашок». Попри наявність потужних бібліотек і фреймворків, таких як TensorFlow Recommenders чи Surprise, головним завданням дослідників і розробників залишається пошук оптимального балансу між точністю, масштабованістю та довірою користувачів. Саме поєднання технічної ефективності, адаптивності та етичної відповідальності визначає успішність сучасних рекомендаційних систем у цифровій комерції.

1.2 Історія рекомендаційних систем

Історія рекомендаційних систем бере свій початок з кінця 1970-х років, коли з'явилися перші цифрові системи обміну інформацією між користувачами. Однією з таких систем був Usenet – комунікаційна платформа, де користувачі могли обмінюватися текстовими повідомленнями, що були організовані в тематичні групи. Хоча це ще не була повноцінна рекомендаційна система, вона стала основою для подальших рішень. Першим персоналізованим проєктом вважається Grundy 1979-го року, який запитував у користувача про його вподобання і, на основі примітивного стереотипного моделювання, рекомендував книги. Цей підхід започаткував нову еру в автоматизованому наданні рекомендацій, незважаючи на свої обмеження.

Згодом сформувалися два основні типи рекомендаційних систем: колаборативна фільтрація та фільтрація на основі вмісту. Колаборативна фільтрація орієнтується на вподобання інших користувачів, схожих на

поточного, наприклад, «користувачі, які слухали це, також слухали...». Однією з перших таких систем стала Tapestry, розроблена Xerox PARC, яка дозволяла користувачам коментувати документи та використовувати ці відгуки як інструмент фільтрації. Іншим помітним прикладом стала система GroupLens 1992-го року, яка автоматично рекомендувала новини з Usenet на основі оцінок користувачів. У цей же період з'явилися Ringo, що створювала музичні рекомендації, Firefly та інші, що показували зростаючий інтерес до персоналізації цифрового контенту.

Фільтрація на основі вмісту розвивалась на базі інформаційного пошуку та аналізу текстів. Історично її витоки можна простежити до “статистичної машини” Емануеля Гольдберга у 1920-х роках, яка шукала документи за зразками. У 1960-х роках команда під керівництвом Джерарда Салтона з Університету Корнелла розробила модель автоматичної індексації текстів, яка стала основою для сучасних методів обробки тексту і застосовується донині в пошукових системах. У сфері музики прикладом став проєкт Music Genome Project 1999-го року, що описував пісні за понад 450 параметрами і використовував ці дані для персоналізованих рекомендацій. Він ліг в основу Pandora Radio, яка на сьогодні має сотні мільйонів користувачів.

У 1994 році студенти зі Стенфорда представили Fab – першу гібридну систему, яка поєднувала обидва підходи: спочатку збирала тематичний контент, а потім персоналізувала рекомендації для кожного користувача. Надалі гібридні моделі стали основою для таких гігантів як Netflix, чий алгоритм CineMatch вважався еталоном у 2000-х. У 2006 році Netflix оголосив конкурс на створення системи, що на 10% перевищуватиме якість CineMatch – переможне рішення поєднувало понад 100 різних алгоритмів і отримало \$1 млн.

Сьогодні рекомендаційні системи є ключовим елементом цифрового світу. Їх використовують платформи YouTube, Amazon, Spotify, TikTok, Google News та інші, щоб підвищити залученість користувачів, збільшити продажі або утримати аудиторію. Вони враховують як явні вподобання такі як: лайки, оцінки, перегляди, так і неявні такі як: час перегляду, зупинки, пропуски,

застосовуючи сучасні методи машинного навчання, нейронні мережі, кластеризацію, графові моделі, обробку природної мови тощо. Ці системи перетворилися з простих рекомендаційних алгоритмів на складні інтелектуальні модулі, що адаптуються до контексту, пристрою, часу доби й навіть емоційного стану користувача [1].

1.3 Контентний метод

Контентна фільтрація (Content-Based Filtering, CBF) є одним із базових підходів до побудови рекомендаційних систем і ґрунтується на аналізі характеристик об'єктів, з якими користувач взаємодівав у минулому. На відміну від колаборативної фільтрації, що використовує інформацію про вподобання інших користувачів, контентний підхід зосереджується безпосередньо на індивідуальних інтересах конкретного користувача. Основна ідея полягає в тому, що якщо користувач позитивно оцінив певні об'єкти, то ймовірність його зацікавлення іншими об'єктами зі схожими характеристиками є високою. Таким чином, рекомендації формуються на основі подібності контенту, а не соціальної взаємодії між користувачами.

Ключовими складовими контентної фільтрації є представлення об'єктів та побудова профілю користувача. Кожен об'єкт описується набором ознак, які можуть мати різну природу: текстову (опис, ключові слова, теги), структуровану (категорія, автор, компанія), візуальну або іншу. На основі ознак об'єктів, з якими користувач уже взаємодівав, формується його профіль, що узагальнює індивідуальні вподобання. Далі система порівнює профіль користувача з описами інших об'єктів, використовуючи міри подібності, і формує впорядкований список рекомендацій.

Історично контентна фільтрація розвивалася від простих методів зіставлення ключових слів до складніших векторних моделей. Ранні підходи базувалися на використанні TF-IDF та косинусної подібності у межах векторної просторової моделі, що дозволяло представляти тексти у вигляді числових

векторів. Однак такі методи мали обмеження, пов'язані з відсутністю глибокого семантичного розуміння, проблемами синонімії та багатозначності слів. Подальший розвиток методів машинного навчання та обробки природної мови зумовив перехід до використання щільних векторних подань (embeddings), які здатні краще відображати семантичні зв'язки між об'єктами.

Сучасні системи контентної фільтрації широко застосовують глибинні нейронні мережі та попередньо навчені мовні моделі, такі як BERT, для побудови змістовних векторних представлень об'єктів. Це дозволяє враховувати контекст, приховані смислові зв'язки та різноманітні типи даних, суттєво підвищуючи якість рекомендацій. Водночас контентна фільтрація має і певні обмеження, зокрема схильність до надмірної спеціалізації рекомендацій та залежність від якості опису об'єктів. Попри це, даний підхід залишається важливим елементом сучасних рекомендаційних систем і часто використовується як складова гібридних моделей, що поєднують його переваги з можливостями інших методів [2].

1.3.1 Огляд статті Content-based filtering algorithm in social media

Стаття «Content-based filtering algorithm in social media» присвячена ґрунтовному аналізу алгоритму контентної фільтрації як одного з ключових підходів до побудови рекомендаційних систем, з особливим акцентом на його застосування у соціальних мережах [3]. Автори розглядають content-based filtering як метод персоналізації, що формує рекомендації на основі аналізу характеристик контенту, з яким користувач уже взаємодіяв. Підкреслюється, що на відміну від колаборативної фільтрації, даний підхід не залежить від даних інших користувачів, а зосереджується виключно на індивідуальних уподобаннях, що робить його ефективним для спеціалізованого контенту.

У статті детально описано методологію контентної фільтрації, яка складається з кількох основних етапів: аналізу контенту, побудови профілю користувача, генерації рекомендацій та організації зворотного зв'язку. Аналіз

контенту передбачає виділення ключових ознак об'єктів, таких як ключові слова, теми або жанри, з використанням методів машинного навчання та обробки природної мови. На основі цих ознак формується профіль користувача, який відображає його інтереси та слугує основою для подальшого зіставлення з іншими об'єктами платформи.

Окрему увагу автори приділяють перевагам контентної фільтрації. Серед основних виділяються висока персоналізація рекомендацій, незалежність від поведінки інших користувачів, прозорість механізму формування рекомендацій та відносна простота реалізації. Також підкреслюється здатність таких алгоритмів до безперервного навчання та адаптації до змін у вподобаннях користувача, що особливо важливо для динамічних середовищ, зокрема соціальних мереж.

Водночас у статті зазначаються й обмеження контентної фільтрації. До них належать труднощі з рекомендацією нового або нетипового контенту, схильність до надмірної спеціалізації (filter bubble), а також зниження ефективності для користувачів із дуже різноманітними інтересами. Автори наголошують, що точність рекомендацій значною мірою залежить від якості аналізу контенту та коректності інтерпретації дій користувача, що може призводити до помилок у разі неповних або шумних даних.

У завершальній частині статті контентна фільтрація розглядається в контексті сучасних соціальних медіа-платформ, де вона використовується для персоналізації стрічок новин, рекомендацій публікацій та рекламного контенту. Автори підкреслюють, що на практиці контентна фільтрація зазвичай поєднується з колаборативними та гібридними підходами для подолання її недоліків і підвищення різноманітності рекомендацій. Загалом стаття робить висновок, що content-based filtering залишається важливим і перспективним інструментом у рекомендаційних системах, особливо як складова гібридних моделей, орієнтованих на підвищення залученості та задоволеності користувачів.

1.4. Колаборативний метод

Колаборативна фільтрація є одним із ключових підходів до побудови рекомендаційних систем і спрямована на подолання обмежень контентної фільтрації шляхом одночасного врахування подібності між користувачами та об'єктами. Основна ідея цього методу полягає в тому, що користувачі з подібними вподобаннями, як правило, зацікавлені в схожих об'єктах, навіть якщо ці об'єкти мають різні формальні характеристики. Завдяки цьому колаборативна фільтрація здатна генерувати так звані серендинні рекомендації – тобто пропонувати користувачеві об'єкти, які він не зміг би знайти лише на основі аналізу власної історії взаємодій або контентних ознак.

У класичній постановці задача колаборативної фільтрації описується через матрицю взаємодій «користувач–об'єкт», де рядки відповідають користувачам, стовпці – об'єктам, а значення елементів відображають факт або рівень взаємодії. Взаємодія може бути явною, наприклад у вигляді числових оцінок, або неявною – переглядів, кліків чи фактів споживання контенту. Метою алгоритму є відновлення або апроксимація цієї матриці таким чином, щоб передбачити відсутні значення, тобто оцінити ймовірність зацікавлення користувача в об'єктах, з якими він ще не взаємодіяв.

Сучасні підходи до колаборативної фільтрації базуються на ідеї латентних представлень (embeddings) користувачів і об'єктів у спільному векторному просторі. Кожному користувачеві та кожному об'єкту відповідає вектор прихованих ознак, які відображають узагальнені вподобання та властивості. Оцінка взаємодії між користувачем і об'єктом моделюється як скалярний добуток відповідних векторів. У простих випадках достатньо одновимірного простору, однак для опису складних вподобань зазвичай використовується багатовимірний латентний простір, де кожен вимір відповідає певному прихованому фактору, наприклад жанру, стилю або популярності.

Процес навчання колаборативної моделі має ітеративний характер. Якщо вектори об'єктів зафіксовані, модель може оптимізувати вектори користувачів

так, щоб якомога точніше відтворити наявні взаємодії, і навпаки. У результаті користувачі з подібними вподобаннями опиняються близько один до одного у латентному просторі, так само як і об'єкти, які подобаються схожим групам користувачів. Саме автоматичне навчання таких представлень без ручного конструювання ознак є однією з головних переваг колаборативної фільтрації та пояснює її широке застосування у сучасних рекомендаційних системах.

1.4.1 Огляд статті «A Distributed Collaborative Filtering Algorithm Using Multiple Data Sources»

У статті «A Distributed Collaborative Filtering Algorithm Using Multiple Data Sources» колаборативна фільтрація розглядається як формалізована задача відновлення відсутніх значень у матриці взаємодій користувачів та об'єктів [4]. Нехай задано матрицю зворотного зв'язку у формулі (1.1):

$$R \in \mathbb{R}^{|U| \times |I|}, \quad (1.1)$$

де U – множина користувачів;

I – множина об'єктів (наприклад, фільмів);

r_{ui} – відображає факт або інтенсивність взаємодії користувача u з об'єктом i .

У спрощеному випадку, який розглядається в статті, матриця є бінарною, тобто $r_{ui} \in \{0; 1\}$. Основна мета рекомендаційної системи полягає у знаходженні такої функції прогнозування $\widehat{r}_{ui}$, яка з максимальною точністю наближує невідомі значення матриці R .

Ключовою ідеєю статті є використання латентного векторного простору, у якому як користувачі, так і об'єкти представлені векторами однакової розмірності. Кожному користувачу u ставиться у відповідність вектор, як показано у формулі (1.2):

$$P_u \in \mathbb{R}^k, \quad (1.2)$$

а кожному об'єкту i – вектор, як показано у формулі (1.3):

$$q_i \in \mathbb{R}^k, \quad (1.3)$$

де k – кількість прихованих факторів, тобто латентних ознак. Прогнозована взаємодія між користувачем та об'єктом обчислюється як скалярний добуток, продемонстрований у рівнянні (1.4):

$$\widehat{r_{ui}} = P_u^T q_i \quad (1.4)$$

Таким чином, рекомендація формується на основі геометричної близькості векторів у латентному просторі, а висока релевантність відповідає великому значенню скалярного добутку.

У статті наочно продемонстровано, що використання одновимірного латентного простору ($k = 1$) є недостатнім для коректного моделювання вподобань усіх користувачів. У цьому випадку кожен користувач і кожен об'єкт описуються одним числом, а прогноз зводиться до множення двох скалярів. Така модель здатна пояснити лише прості закономірності, наприклад орієнтацію на «дитячий» або «дорослий» контент. Для подолання цього обмеження вводиться багатовимірний простір ($k \geq 2$), у якому кожен вимір відповідає окремому прихованому фактору, наприклад жанровій належності чи рівню популярності. Математично це дозволяє апроксимувати матрицю R як добуток двох низькорангових матриць з рівняння (1.5):

$$R \approx PQ^T, \quad (1.5)$$

де $P \in \mathbb{R}^{|U| \times k}$, $Q \in \mathbb{R}^{|I| \times k}$.

Процес навчання моделі має колаборативний характер і базується на мінімізації функції втрат, яка вимірює різницю між фактичними та прогнозованими значеннями взаємодій. У загальному оптимізаційна задача формулюється як показано у формулі (1.6):

$$\min_{P,Q} \sum_{(u,i) \in \Omega} (r_{ui} - p_u^T q_i)^2, \quad (1.6)$$

де Ω – множина спостережуваних взаємодій.

Зафіксувавши матрицю Q , можна оптимізувати P , і навпаки, що відображає ідею взаємного навчання векторів користувачів і об'єктів. У результаті користувачі зі схожими патернами поведінки матимуть близькі вектори P_u , а об'єкти, які подобаються подібним групам користувачів, – близькі вектори q_i .

Отже, стаття демонструє, що колаборативна фільтрація може бути строго обґрунтована з математичної точки зору як задача матричної факторизації та оптимізації у латентному просторі. Такий підхід дозволяє автоматично виявляти приховані структури в даних без ручного конструювання ознак. Водночас автори підкреслюють, що ефективність методу значною мірою залежить від щільності матриці взаємодій, що пояснює проблеми холодного старту та необхідність подальшого поєднання колаборативних моделей з контентними або гібридними підходами.

1.5 Гібридний метод

Гібридні рекомендаційні системи є логічним етапом розвитку методів персоналізованих рекомендацій і створюються з метою подолання обмежень окремих підходів, таких як колаборативна та контентна фільтрація. Традиційні рекомендаційні алгоритми широко застосовуються в онлайн-сервісах, соціальних мережах та електронній комерції для зменшення інформаційного перевантаження та підвищення релевантності контенту для користувачів. Водночас кожен із класичних методів має суттєві недоліки: колаборативна фільтрація страждає від проблем холодного старту та розрідженості даних, тоді як контентна фільтрація схильна до надмірної спеціалізації та обмеженого різноманіття рекомендацій. Саме тому гібридні підходи, що поєднують декілька методів, розглядаються як більш універсальне та ефективне рішення.

Основна ідея гібридних рекомендаційних систем полягає в інтеграції сильних сторін різних методів та компенсації їх слабких місць. Найпоширенішими є комбінації колаборативної та контентної фільтрації, а також залучення підходів на основі знань (knowledge-based filtering), що використовують явні характеристики об'єктів і вподобання користувачів. Колаборативна фільтрація базується на пошуку користувачів зі схожими патернами поведінки або об'єктів зі схожими історіями взаємодій, тоді як контентна фільтрація порівнює профіль користувача з атрибутами об'єктів.

Методи на основі знань, у свою чергу, використовують правила, онтології або кейс-орієнтовані моделі для формування рекомендацій на основі явних вимог користувача.

У літературі виокремлюють декілька стратегій гібридизації. Зважені гібриди об'єднують результати різних рекомендаційних модулів за допомогою лінійної комбінації їх оцінок. Перемикальні гібриди обирають один із методів залежно від поточної ситуації, наприклад наявності історії взаємодій. Змішані гібриди формують єдиний список рекомендацій із результатів кількох алгоритмів. Також застосовуються підходи комбінування ознак, доповнення ознак, каскадні та метарівневі гібриди, у яких модель, навчена одним методом, використовується як вхід для іншого. Такі стратегії дозволяють ефективніше розв'язувати проблему холодного старту та підвищувати стійкість системи до нестачі даних.

З алгоритмічної точки зору гібридні рекомендаційні системи часто спираються на матричну факторизацію та її розширення, зокрема узагальнену матричну факторизацію, атрибутно-орієнтовані моделі та нейронні підходи. Проекція користувачів і об'єктів у спільний латентний простір дає змогу поєднати інформацію з різних джерел, включаючи рейтинги, текстові описи, метадані та графи побудовані на основі знань про користувача. Сучасні гібридні системи дедалі частіше використовують глибоке навчання, зокрема нейронну колаборативну фільтрацію, згорткові та багатошарові моделі, які дозволяють виявляти складні нелінійні залежності та покращувати якість рекомендацій, особливо для нових користувачів і об'єктів.

Гібридні рекомендаційні системи успішно застосовуються в багатьох практичних доменах, зокрема у відеостримінгових сервісах, електронній комерції, системах підбору вакансій, онлайн-освіті та охороні здоров'я. Емпіричні дослідження демонструють, що такі системи забезпечують вищі показники точності, повноти та якості ранжування порівняно з однокомпонентними підходами. Водночас вони залишаються складними в реалізації, потребують значних обчислювальних ресурсів і породжують

додаткові виклики, пов'язані з масштабованістю, інтерпретованістю та захистом приватних даних. Попри це, гібридні рекомендаційні системи розглядаються як один із найперспективніших напрямів розвитку персоналізованих сервісів [5].

1.5.1 Огляд статті «Hybrid recommender system model for digital library from multiple»

У статті «Hybrid recommender system model for digital library from multiple» розглянуто розробку та експериментальну перевірку гібридної рекомендаційної системи для цифрової бібліотеки, яка інтегрує електронні книги з декількох незалежних онлайн-видавців у єдину платформу [6]. Автори підкреслюють, що традиційні цифрові бібліотеки часто є фрагментованими: дані зберігаються в ізольованих базах різних організацій, що ускладнює пошук і персоналізовану навігацію для користувачів. Запропонована система вирішує цю проблему шляхом API-інтеграції метаданих книг та використання гібридного рекомендаційного підходу, який поєднує колаборативну фільтрацію (CF) і контентну фільтрацію (CBF).

Методологічною основою роботи є побудова користувацького профілю на основі імпліцитного та експліцитного зворотного зв'язку: історії читання, збережених книг і оцінок. Колаборативна складова реалізована через класичну модель user-based CF, де прогнозована релевантність книги для користувача визначається як зважене відхилення від середніх оцінок подібних користувачів. Формально це подано через функцію, що використовує коефіцієнт подібності між профілями користувачів. Контентна складова, своєю чергою, базується на простій, але інтерпретованій евристиці – відповідності книги вподобаним категоріям, видавцям та року публікації, що дозволяє ефективно працювати в умовах холодного старту для нових користувачів.

Ключовим елементом запропонованої моделі є лінійна гібридизація рекомендацій формула якої показана в рівнянні (1.7):

$$Hybrid_{score} = (1 - \alpha)CF + \alpha * CB, \quad (1.7)$$

де параметр α визначає баланс між колаборативною та контентною складовими.

Автори експериментально порівнюють кілька конфігурацій ваг: 50:50, 20:80, 80:20, та показують, що варіант 80% CF і 20% CBF забезпечує найкращі результати. Такий вибір пояснюється тим, що CF краще захоплює латентні вподобання користувачів, тоді як CBF виконує роль стабілізатора для нових або профілів про яких мало даних.

Експериментальна оцінка проведена за участю 150 реальних користувачів, що є суттєвим покращенням у порівнянні з попередніми версіями роботи. Для оцінювання якості рекомендацій використано метрики NDCG@15 та Precision, що дозволяє врахувати не лише факт релевантності, а й позицію документа в ранжованому списку. Результати демонструють статистично значущу перевагу гібридної моделі 80:20 над чистими CF та CBF, що підтверджено за допомогою однофакторного ANOVA-аналізу.

У висновках автори підкреслюють практичну цінність запропонованого підходу для цифрових бібліотек і освітніх платформ, особливо в умовах розрізнених джерел даних і обмеженого зворотного зв'язку. Водночас зазначаються обмеження дослідження, пов'язані зі складністю інтеграції даних різних видавців, питаннями масштабованості та конфіденційності. Як напрям подальших досліджень пропонується використання глибоких нейронних моделей і більш складних стратегій гібридизації, що узгоджується з сучасними трендами розвитку рекомендаційних систем.

1.6 Задачі дослідження

Для досягнення поставленої мети у роботі необхідно розв'язати такі основні завдання:

- здійснити огляд і систематизацію сучасних методів побудови рекомендаційних систем, зокрема підходів колаборативної, контентної та гібридної фільтрації;
- проаналізувати переваги й обмеження існуючих алгоритмів рекомендацій з точки зору точності, масштабованості, стійкості до розрідженості даних та адаптивності до появи нових користувачів і об'єктів;
- розробити унікальний алгоритм генерації рекомендацій, заснований на гібридній моделі, що поєднує контентні та колаборативні підходи;
- реалізувати програмну рекомендаційну систему із використанням запропонованого алгоритму та сучасних інструментів обробки даних і машинного навчання;
- провести експериментальне дослідження та візуалізацію результатів роботи системи з метою підвищення наочності та зручності сприйняття користувачем.

1.7 Висновки до розділу 1

У розділі 1 здійснено всебічний аналіз актуальності задачі побудови рекомендаційних систем у сучасному цифровому середовищі, де користувачі щоденно стикаються з проблемою інформаційного перенасичення. Обґрунтовано, що рекомендаційні системи є ключовим інструментом персоналізації контенту та сервісів, оскільки дозволяють зменшити когнітивне навантаження на користувача, підвищити зручність взаємодії з платформами та сприяти зростанню рівня залученості аудиторії. Також показано їхню важливу роль у підвищенні ефективності бізнес-процесів, зокрема в електронній комерції, медіасервісах, соціальних мережах і професійних онлайн-платформах.

Окрему увагу в розділі приділено історії становлення та розвитку рекомендаційних систем. Розглянуто еволюцію підходів – від перших рішень, заснованих на простих правилах і текстовому пошуку, до більш складних

методів контентної та колаборативної фільтрації. Показано, як подальший розвиток обчислювальних потужностей і накопичення великих обсягів даних сприяв появі моделей машинного навчання, матричної факторизації та, згодом, глибоких нейронних мереж, що значно розширили можливості рекомендаційних систем у роботі з неструктурованими та семантично складними даними.

На основі проведеного аналізу виконано класифікацію основних типів рекомендаційних систем та охарактеризовано їхні переваги й недоліки. Виділено три ключові напрями побудови рекомендацій: контент-орієнтовані системи, які аналізують властивості самих об'єктів; системи, засновані на колаборативній фільтрації, що ґрунтуються на подібності поведінки користувачів; а також гібридні системи, які поєднують кілька підходів для підвищення точності та стійкості рекомендацій. Зроблені висновки формують теоретичну основу для подальших досліджень і практичної реалізації рекомендаційної системи, що розглядається в наступних розділах роботи.

2 ОПИС МЕТОДІВ ТА МЕТРИК ГІБРИДНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

Для побудови та оцінки рекомендаційної системи підбору вакансій у даній роботі було застосовано комплекс методів обробки текстових описів, моделювання навичок кандидатів та алгоритмів рекомендаційного аналізу. У цьому розділі викладено теоретичні основи, що лежать в основі програмної реалізації системи, включаючи контентні моделі TF-IDF та BERT, колаборативний алгоритм ALS, а також механізми їх комбінування в гібридну структуру. Окрім того, розглянуті метрики, що використовувалися для оцінки якості рекомендацій та порівняння ефективності різних методів.

У розділі подано детальний математичний опис принципів формування рекомендацій, що охоплює побудову векторних представлень вакансій і профілів кандидатів, обчислення текстової подібності та моделювання прихованих факторів у матриці взаємодій. Особлива увага приділяється контентному методу фільтрації, який у цій роботі відіграє ключову роль завдяки використанню семантичних моделей представлення тексту. Математична формалізація дозволяє точніше зрозуміти механізми роботи системи та створює основу для подальшого вдосконалення архітектури.

2.1 Алгоритм побудованої моделі

Алгоритм роботи побудованої рекомендаційної системи підбору вакансій починається зі створення синтетичного датасета, який моделює реальні взаємодії між кандидатами та вакансіями. На цьому етапі генеруються навички кандидатів, вимоги вакансій, текстові профілі та історія відгуків, що забезпечує основу для подальшого формування рекомендацій. Після створення датасета система переходить до етапу отримання списків рекомендацій за кожним із трьох основних методів: TF-IDF, BERT та ALS.

На другому етапі система паралельно формує три незалежні списки рекомендацій. Контентний метод TF-IDF обчислює лексичну подібність між текстом профілю кандидата та описами вакансій, формуючи ранжований список релевантних позицій. Модель BERT виконує аналогічне завдання, але з використанням семантичних векторних представлень, здатних уловлювати змістові зв'язки між фразами. Колаборативна модель ALS генерує список вакансій на основі поведінкових патернів кандидатів, враховуючи приховані фактори у матриці взаємодій. Таким чином, на виході цього блоку система має три незалежні списки рекомендацій, кожен з яких сформовано різними алгоритмічними підходами.

Наступним кроком є обробка отриманих рекомендаційних списків. Для кожного методу застосовується процедура нормалізації оцінок та додавання вагових коефіцієнтів. Цей етап дозволяє вирівняти масштаби оцінок різних моделей та підготувати дані для об'єднання. Ваги, призначені для TF-IDF, BERT та ALS, визначають внесок кожної моделі у фінальний результат і можуть коригуватися залежно від експериментальних досліджень або вимог системи.

Після обробки окремих списків система переходить до формування гібридної структури. На цьому етапі всі часткові результати об'єднуються у спільний ранжований список, де фінальний бал кожної вакансії обчислюється як зважена сума оцінок трьох методів. Гібридна модель забезпечує збалансоване поєднання семантичних, текстових та поведінкових ознак, компенсуючи недоліки окремих алгоритмів і підвищуючи загальну точність рекомендацій.

На завершальному етапі отриманий гібридний список сортується за фінальною оцінкою, після чого формується фінальний перелік вакансій, рекомендованих конкретному кандидату. Цей список є кінцевим виходом системи й використовується для оцінювання її ефективності та порівняння різних підходів у рамках експериментальної частини дослідження. Скорочений варіант цього алгоритм проілюстрований блок-схемою на рисунку 2.1

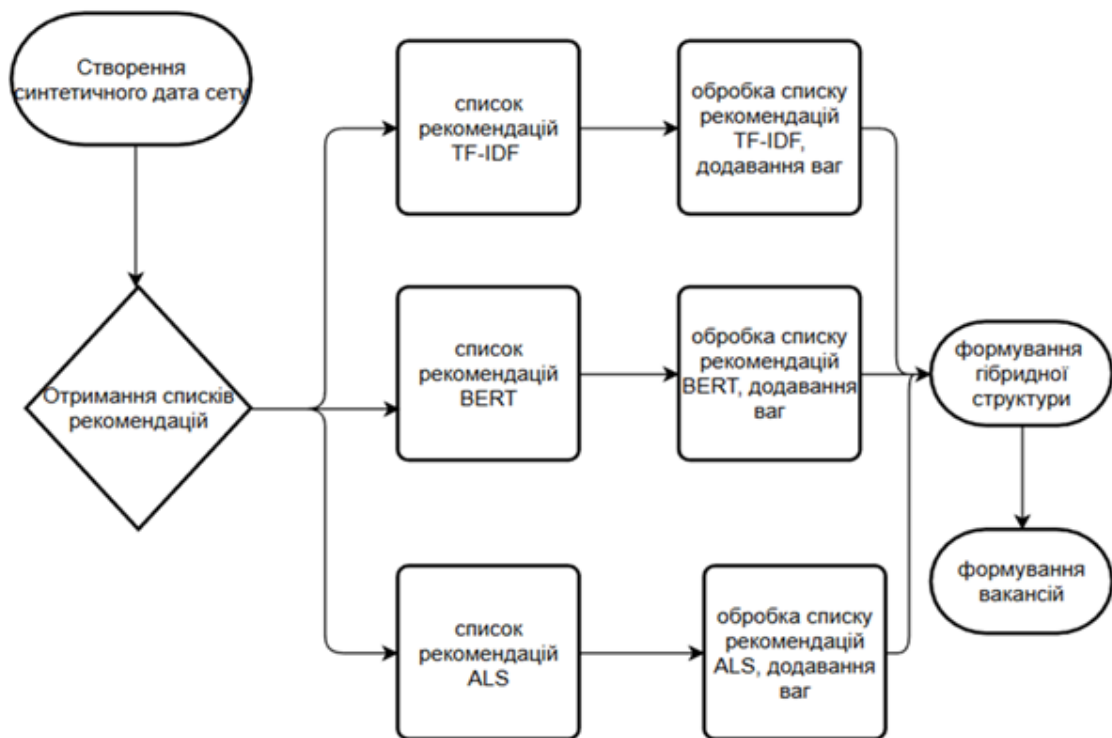


Рисунок 2.1 – алгоритм побудови рекомендаційної системи у вигляді блок-схеми

2.2 Контентний метод фільтрації

Контентний метод фільтрації ґрунтується на використанні характеристик вакансій для формування персоналізованих рекомендацій відповідно до індивідуального профілю кандидата. На відміну від колаборативних методів, цей підхід працює виключно з описом за ознаками вакансій і кандидата, не потребуючи інформації про поведінку інших користувачів. Метод аналізує текстові та структурні властивості вакансій, визначає їх подібність до профілю кандидата і на цій основі формує ранжований список рекомендацій.

У даній роботі контентний підхід реалізовано шляхом побудови матриці, де кожен рядок відповідає вакансії, а кожен стовпець – певній навичці. Ці ознаки представлені у вигляді бінарних змінних: значення «1» означає наявність навички у вакансії, а «0» – її відсутність. Профіль кандидата формується в тому самому просторі ознак, що забезпечує можливість математичного порівняння векторів вакансій та кандидата.

Якщо вектори кандидата x та вакансії y є бінарними, то їхня подібність може бути обчислена через скалярний добуток, як показано у рівнянні (2.1):

$$\langle x, y \rangle = \sum_{i=1}^d x_i y_i \quad (2.1)$$

У цій формулі кожен доданок $x_i y_i$ дорівнює 1 лише у випадку, якщо відповідна навичка одночасно присутня і в профілі кандидата, і у вимогах вакансії. Таким чином, скалярний добуток відображає кількість спільних характеристик між кандидатом і вакансією. Чим більше спільних навичок – тим більша подібність і тим вищою має бути позиція вакансії у списку рекомендацій [7].

Для текстових описів вакансій та профілів кандидатів було застосовано два підходи:

- TF-IDF, який відображає лексичну подібність текстів через аналіз частоти термінів;
- BERT, який формує семантично усвідомлені векторні представлення текстів і дозволяє виявляти глибші змістові відповідності.

2.2.1 TF-IDF метод

У більшості контентно-орієнтованих рекомендаційних систем використовуються відносно прості моделі пошуку релевантних об'єктів, такі як пошук за ключовими словами або векторна модель Vector Space Model, у подальшому VSM з вагуванням TF-IDF. Векторна модель є просторовим представленням текстових документів, у якому кожен документ інтерпретується як вектор у n -вимірному просторі, де кожен вимір відповідає окремому терміну словника. У контексті даної роботи документами виступають текстові описи вакансій і профілі кандидатів, на основі яких здійснюється обчислення контентної подібності [8].

Нехай $D = \{d_1, d_2, \dots, d_n\}$ – множина текстових документів тобто описів вакансій, а $T = \{t_1, t_2, \dots, t_n\}$ – словник термінів, сформований після застосування стандартної попередньої обробки тексту: токенизації, видалення

стоп-слів, лематизації тощо. Кожний документ d_j подається як вектор ваг термінів, як показано у рівнянні (2.2):

$$d_j = \{w_{1j}, w_{2j}, \dots, w_{nj}\}, \quad (2.2)$$

де вага w_{nj} відображає важливість терміна t_n у документі d_n .

Подання документів у VSM вимагає визначення ваг для кожного терміна. Найбільш поширеною схемою вагування є TF-IDF, або term frequency – inverse document frequency відповідає рівнянню (2.3), яка ґрунтується на таких емпіричних припущеннях:

- рідкісні терміни не менш важливі, ніж частотні, тобто компонента IDF;
- часті появи терміна в документі сильніше характеризують його зміст, тобто компонента TF;
- довгі документи не повинні отримувати штучну перевагу, тобто потреба в нормалізації

$$TF - IDF(t, d) = TF(t, d) \times \log\left(\frac{N}{\partial f_t}\right), \quad (2.3)$$

де $TF = \frac{t\text{-частота появи документа}}{d\text{-кількість документів}}$, тобто як частота терміна в документі, нормалізована на максимальну частоту будь-якого терміна у цьому ж документі, як показано у рівнянні (2.4):

$$IDF = \log \frac{N}{\partial f_t}, \quad (2.4)$$

де N – загальна кількість документів у корпусі;

∂f_t – кількість документів, у яких термін t_n зустрічається хоча б один раз.

2.2.2 Семантичний метод фільтрації BERT

Модель BERT (Bidirectional Encoder Representations from Transformers) ґрунтується на архітектурі Transformer та використовує потужний механізм контекстного кодування для глибокого розуміння природної мови. На відміну від оригінальної трансформерної моделі, що складається з модулів енкодера та декодера, BERT застосовує лише енкодерну частину, оскільки її основною метою є саме аналіз і розуміння вхідних текстових послідовностей, а не їх

генерація. Такий підхід дозволяє моделі більш точно інтерпретувати зміст тексту та визначати семантичні зв'язки між словами.

Традиційні мовні моделі обмежені тим, що опрацьовують текст послідовно – або зліва направо, або справа наліво. У результаті модель бачить лише частину контексту на момент прогнозування слова. BERT усуває це обмеження завдяки двонаправленому механізму обробки, який одночасно враховує як лівий, так і правий контекст кожного слова. На практиці це означає, що модель аналізує всі слова в реченні паралельно, що дозволяє досягти глибшого семантичного розуміння тексту.

У межах цієї магістерської дисертації BERT застосовується для побудови контекстуальних векторних подань описів вакансій та профілів кандидатів. На відміну від TF-IDF, який оперує лексичною подібністю, BERT дає змогу враховувати змістові відтінки та приховані семантичні зв'язки між словами, що значно підвищує якість рекомендацій у контентній частині системи [9].

Модель BERT (Bidirectional Encoder Representations from Transformers) ґрунтується на енкодерній частині архітектури Transformer та використовує багат шаровий механізм самоуваги для глибинного розуміння тексту. Ключовою перевагою цього підходу є двонаправлений аналіз: замість того щоб опрацьовувати текст послідовно зліва направо або справа наліво, BERT розглядає всі токени речення одночасно, враховуючи як лівий, так і правий контекст. Це забезпечує значно точнішу інтерпретацію змістових зв'язків між словами, що є критично важливим для завдань семантичної подібності, зокрема у рекомендаційних системах.

Основним обчислювальним механізмом у BERT є механізм самоуваги, або більш відомий як: self-attention, що визначає вагу кожного токена відносно всіх інших токенів у реченні. Для цього кожен токен перетворюється у три вектори: запити Q, ключі K та значення V. Механізм уваги обчислюється за формулою (2.5):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.5)$$

де d_k – розмірність ключів. Нормалізація $\sqrt{d_k}$ забезпечує стабільність градієнтів, а функція softmax перетворює результати на ймовірнісний розподіл уваги. Таким чином кожен токен отримує вагу, що відображає його важливість у контексті речення.

Після проходження шару самоуваги представлення токена h_i надходить у двошаровий feed-forward шар, що дозволяє моделі виявляти складні нелінійні залежності. Цей шар описується рівнянням (2.6):

$$FFN(h_i) = W_2 \sigma(W_1 h_i + b_1) + b_2, \quad (2.6)$$

де $W_1 W_2 b_1 b_2$ – параметри багатошарового персептрону, а σ – нелінійна функція активації. На виході кожен токен отримує контекстний вектор, що відображає його позиційне та семантичне значення у всій послідовності.

У процесі тренування BERT використовує задачу маскування токенів (Masked Language Modeling, MLM). Частина слів у реченні замінюється спеціальним символом [MASK], а модель повинна відновити їх, використовуючи двосторонній контекст. Ймовірність того, що токен на позиції i дорівнює слову x_i визначається рівнянням (2.7):

$$P(x_i | h_i) = \text{softmax}(U h_i), \quad (2.7)$$

де U – матриця проєкції у простір словника, а h_i – контекстний вектор, отриманий після проходження всіх шарів моделі. Функція втрат у MLM визначається як негативна логарифмічна ймовірність правильного відновлення замаскованих слів у рівнянні (2.8):

$$\mathcal{L}_{MLM} = - \sum_{i \in M} \log P(x_i | h_i), \quad (2.8)$$

де M – множина замаскованих позицій. Завдяки цій архітектурі модель навчається розуміти семантичні та синтаксичні залежності між словами в тексті, формуючи глибокі контекстуальні подання [10].

У рекомендаційній системі, розробленій у межах цієї роботи, BERT застосовується для побудови семантичних векторів профілів кандидатів та описів вакансій. На відміну від методів, що ґрунтуються на лексичній подібності, наприклад TF-IDF, BERT дозволяє оцінювати схожість з урахуванням контексту, значень слів та їх взаємозв'язків. Це забезпечує більш

точне порівняння текстових описів і стає основою для формування високоякісних контентних рекомендацій у гібридній моделі.

2.3 Косинусна подібність

Косинусна подібність є однією з найбільш поширених і ефективних метрик для вимірювання схожості між об'єктами, представленими у вигляді векторів у внутрішньому просторі добутків. Вона визначається як косинус кута між двома ненульовими векторами та дозволяє оцінити, наскільки ці вектори «спрямовані» в одному напрямку. На відміну від евклідової відстані, косинусна подібність фокусується не на абсолютних значеннях компонент векторів, а на їх відносному розподілі, що робить її особливо придатною для аналізу високорозмірних і розріджених даних, характерних для задач обробки тексту.

Математично косинусна подібність між двома векторами: $A = a_1, a_2, \dots, a_n$ та $B = b_1, b_2, \dots, b_n$ визначається за формулою (2.9):

$$\text{cosine}_{sim(A,b)} = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}} \quad (2.9)$$

Тут чисельник є скалярним добутком векторів, який відображає сумарний внесок спільних компонент, а знаменник – добутком їх евклідових норм, що забезпечує нормалізацію результату. Значення косинусної подібності зазвичай лежить у діапазоні від 0 до 1, у випадку невід'ємних компонент, де 0 відповідає ортогональним векторам тобто, відсутності схожості, а 1 – повній ідентичності напрямків [11].

У контексті аналізу текстових даних кожен документ подається як вектор у високорозмірному просторі ознак. Для методу TF-IDF такими ознаками є терміни словника, а значеннями компонент – відповідні TF-IDF ваги. Косинусна подібність у цьому випадку фактично вимірює, наскільки два документи мають спільні важливі терміни з урахуванням їхньої інформативності у корпусі. Оскільки більшість термінів у текстах відсутні, TF-IDF вектори є

розрідженими, а косинусна подібність є особливо ефективною, оскільки ігнорує спільні нульові компоненти.

У розробленій рекомендаційній системі косинусна подібність використовується для порівняння векторів профілю кандидата та описів вакансій, сформованих за допомогою TF-IDF. Для кожної вакансії обчислюється значення подібності, після чого формується ранжований список рекомендацій. Такий підхід дозволяє визначати лексичну близькість між текстами, однак він обмежений тим, що враховує лише точні співпадиння термінів [12].

Для подолання цього обмеження у системі також застосовується модель BERT, яка формує щільні семантичні векторні подання, а саме ембеддинги текстів. У цьому випадку кожен опис вакансії та профіль кандидата відображається у відносно низьковимірному, але семантично насиченому просторі, де близькі за змістом тексти мають подібні напрямки векторів. Незважаючи на різну природу векторів TF-IDF і BERT, косинусна подібність залишається універсальною метрикою, що дозволяє коректно порівнювати такі подання, як показано у рівнянні (2.10):

$$sim_{BERT} = \frac{x \cdot y}{\|x\| \|y\|}, \quad (2.10)$$

де x і y – семантичні ембеддинги, отримані з BERT. У цьому випадку високе значення подібності означає не лише спільність слів, а й близькість змісту, навіть якщо тексти не мають однакових термінів [13].

Таким чином, косинусна подібність виконує роль єдиного математичного інструменту для оцінки релевантності як у лексичному TF-IDF, так і у семантичному BERT просторах. Саме ця властивість робить її ключовою складовою контентної частини гібридної рекомендаційної системи. Узгоджене використання косинусної подібності для різних типів векторних представлень забезпечує порівнюваність результатів, стабільність ранжування та можливість подальшого зваженого об'єднання рекомендацій у гібридній моделі.

2.4 Колаборативний метод фільтрації

Колаборативна фільтрація є одним із базових підходів у рекомендаційних системах і ґрунтується на аналізі поведінкових даних користувачів, зокрема їхніх оцінок, відгуків або взаємодій з елементами системи. Основна ідея цього підходу полягає в припущенні, що користувачі з подібними вподобаннями в минулому будуть мати схожі інтереси і в майбутньому. На відміну від контентних методів, колаборативна фільтрація не потребує детального опису об'єктів, а використовує лише матрицю взаємодій «користувач–об'єкт».

Центральним завданням колаборативної фільтрації є прогнозування оцінки або рейтингу r_{ui} , яку користувач u може надати об'єкту i , з яким він ще не взаємодіяв. Для цього система аналізує історію оцінок інших користувачів, подібних до u , або об'єктів, схожих на i . Однією з класичних реалізацій цього підходу є user-based collaborative filtering, де прогноз базується на оцінках сусідніх користувачів.

Прогноз рейтингу у такій моделі визначається за формулою (2.11):

$$\widehat{r}_{ui} = \bar{r}_u + \frac{\sum_{k=1}^K S_{uk}(r_{ki} - \bar{r}_k)}{\sum_{k=1}^K S_{uk}}, \quad (2.11)$$

де $\widehat{r}_{ui}$ – прогнозована оцінка користувача u для об'єкта i ;

$\bar{r}_u$ – середня оцінка користувача u ;

S_{uk} – міра подібності між користувачами u та k ;

K – кількість найближчих сусідів;

r_{ki} – оцінка, яку користувач k надав об'єкту i ;

$\bar{r}_k$ – середня оцінка користувача k .

Ця формула коригує середню оцінку користувача з урахуванням відхилень оцінок сусідів, зважених за рівнем подібності. Таким чином, прогноз враховує як індивідуальні особливості користувача, так і колективний досвід схожих користувачів.

Іншим варіантом формалізації прогнозу рейтингу є модель, у якій використовується агрегування оцінок користувачів, що взаємодіяли з певним об'єктом. У цьому випадку прогноз може бути поданий у вигляді рівняння (2.12):

$$U_x = \bar{U} + \frac{\sum_{J \in ratters(x)} (J_x - \bar{J}) R_{UJ}}{\sum_{J \in ratters(x)} |R_{UJ}|}, \quad (2.12)$$

де U_x – прогнозований рейтинг користувача U для об’єкта x ;

$\bar{U}$ – середній рейтинг користувача U ;

J_x – рейтинг користувача J для об’єкта x ;

$\bar{J}$ – середній рейтинг користувача J ;

R_{UJ} – коефіцієнт подібності між користувачами U та J ;

$ratters(x)$ – множина користувачів, які вже оцінили об’єкт x .

Обидві формули демонструють загальний принцип колаборативної фільтрації: прогноз формується як зважена комбінація відхилень оцінок схожих користувачів від їхніх середніх значень. Вибір конкретної міри подібності S_{uk} або R_{UJ} є критичним і може базуватися на косинусній подібності, кореляції Пірсона або спеціалізованих метриках.

Колаборативна фільтрація має низку суттєвих переваг, зокрема здатність виявляти приховані закономірності у вподобаннях користувачів та формувати рекомендації без необхідності аналізу змісту об’єктів. Водночас вона має й обмеження, серед яких проблема холодного старту, розрідженість матриці оцінок та зростання обчислювальної складності зі збільшенням кількості користувачів і об’єктів [14].

2.4.1 Матрична факторизація

Матрична факторизація є одним із ключових методів побудови сучасних рекомендаційних систем, що дозволяє ефективно прогнозувати вподобання користувачів на основі аналізу їхніх попередніх взаємодій з об’єктами. Основна ідея цього підходу полягає у представленні взаємодій між користувачами та об’єктами у вигляді прихованих латентних ознак у просторі меншої розмірності. Такий підхід перетворює задачу передбачення відсутніх оцінок або взаємодій у задачу математичної оптимізації, що робить його придатним для обробки великих і розріджених наборів даних.

Вихідною точкою матричної факторизації є матриця взаємодій «користувач–об’єкт», у якій рядки відповідають користувачам, стовпці – об’єктам, а елементи матриці містять явні оцінки або неявний зворотний зв’язок. Через те, що більшість користувачів взаємодіє лише з обмеженою кількістю об’єктів, така матриця зазвичай є сильно розрідженою. Матрична факторизація дозволяє розкласти цю велику матрицю на дві менші – матрицю користувачів та матрицю об’єктів, у яких кожен користувач і кожен об’єкт описуються векторами латентних характеристик [15].

Матрична факторизація є одним із фундаментальних математичних підходів, що застосовуються у рекомендаційних системах для моделювання взаємодій між користувачами та об’єктами. Нехай задано множину користувачів $U = \{u_1, u_2, \dots, u_U\}$ та множину об’єктів $D = \{d_1, d_2, \dots, d_D\}$. Взаємодії між користувачами та об’єктами задаються у вигляді матриці рейтингів $R \in \mathbb{R}^{U \times D}$, де елемент r_{ij} відповідає оцінці, яку користувач u_i надав об’єкту d_j . На практиці така матриця є сильно розрідженою, оскільки більшість користувачів взаємодіє лише з обмеженою кількістю об’єктів.

Метою матричної факторизації є апроксимація матриці R у вигляді добутку двох матриць меншої розмірності, як показано у рівнянні (2.13):

$$R \approx PQ^T = \hat{R}, \quad (2.13)$$

де $P \in \mathbb{R}^{U \times K}$ – матриця латентних ознак користувачів;

$Q \in \mathbb{R}^{D \times K}$ – матриця латентних ознак об’єктів;

K – кількість прихованих латентних факторів.

Кожен рядок матриці P , позначений як p_i є вектором латентних характеристик користувача u_i , тоді як кожен рядок матриці Q , позначений як q_j , відповідає латентним характеристикам об’єкта d_j . Прогнозоване значення рейтингу $\hat{r}_{ij}$ визначається як скалярний добуток відповідних векторів у рівнянні (2.14):

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^K p_{ik} q_{kj} \quad (2.14)$$

Таким чином, передбачається, що взаємодія між користувачем і об'єктом може бути пояснена через спільні латентні фактори, наприклад приховані інтереси користувача та властивості об'єкта.

Для навчання моделі необхідно мінімізувати похибку між фактичними рейтингами та їх прогнозами. Для цього вводиться квадратична функція помилки для кожної пари (i, j) . Сумарна функція втрат по всіх відомих взаємодіях $T \subseteq U \times D$ має вигляд, як показано у рівнянні (2.15):

$$E = \sum_{(u_i, d_j, \hat{r}_{ij}) \in T} e_{ij}^2 = \sum_{(u_i, d_j, \hat{r}_{ij}) \in T} (r_{ij} - \sum_{k=1}^K p_{ik} q_{kj})^2 \quad (2.15)$$

Для мінімізації цієї функції використовується градієнтний спуск. Для цього обчислюються частинні похідні функції помилки за параметрами p_{ik} та q_{kj} . На основі цих градієнтів параметри моделі оновлюються ітеративно за правилом з рівнянь (2.16) та (2.17):

$$p'_{ik} = p_{ik} + \alpha \frac{\partial e_{ij}^2}{\partial p_{ik}} = p_{ik} + 2\alpha e_{ij} q_{kj} \quad (2.16)$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial e_{ij}^2}{\partial q_{kj}} = q_{kj} + 2\alpha e_{ij} p_{ik}, \quad (2.17)$$

де α – крок навчання, що визначає швидкість збіжності алгоритму.

Процес оновлення повторюється доти, доки значення функції втрат не стабілізується або не досягне мінімального значення [16].

2.4.2 ALS

Алгоритм Alternating Least Squares (ALS) є класичним і широко застосовуваним методом матричної факторизації в межах колаборативної фільтрації. Він призначений для відновлення прихованих взаємозв'язків між користувачами та об'єктами на основі розрідженої матриці взаємодій. Вхідними даними для ALS є матриця рейтингів або неявних взаємодій $R \in \mathbb{R}^{U \times I}$, де рядки відповідають користувачам, стовпці – об'єктам, а ненульові елементи відображають факт взаємодії або її інтенсивність. Метою алгоритму є апроксимація цієї матриці у вигляді добутку двох матриць меншої розмірності, що містять латентні представлення користувачів і об'єктів.

Формально задача ALS полягає у знаходженні таких матриць $X \in \mathbb{R}^{U \times d}$ та $Y \in \mathbb{R}^{I \times d}$, які мінімізують функцію втрат, як показано у рівнянні (2.18):

$$\mathcal{L} = \sum_{(u,i) \in R} (r_{ui} - x_u^t y_i)^2 + \lambda (\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2), \quad (2.18)$$

де r_{ui} – спостережуване значення взаємодії користувача u з об'єктом i ;

x_u та y_i – латентні вектори користувача і об'єкта відповідно;

d – розмірність латентного простору;

λ – коефіцієнт регуляризації, що запобігає перенавчанню;

$\mathbb{R}$ – множина спостережуваних взаємодій.

Перший доданок функції втрат відображає квадратичну похибку між прогнозованим значенням $r_{ui} = x_u^t y_i$ та фактичним значенням r_{ui} . Другий доданок є регуляризаційним і обмежує норму латентних векторів, що дозволяє уникнути надмірної адаптації моделі до навчальних даних.

Ключовою особливістю ALS є спосіб мінімізації цієї функції. Замість одночасного оновлення матриць X та Y алгоритм використовує почергову оптимізацію. На кожній ітерації одна з матриць фіксується, а інша оптимізується шляхом розв'язання задачі найменших квадратів. Спочатку, за фіксованої матриці Y , знаходяться оптимальні вектори x_u для всіх користувачів. Потім, за фіксованої матриці X , аналогічно обчислюються вектори y_i для всіх об'єктів. Цей процес повторюється ітеративно до досягнення збіжності.

Завдяки такому підходу на кожному кроці оптимізації функція втрат є опуклою відносно оптимізованої змінної, що гарантує відсутність локальних мінімумів і забезпечує швидку збіжність алгоритму. Саме ця властивість відрізняє ALS від методів на основі стохастичного градієнтного спуску та робить його стабільним для великих наборів даних.

У результаті навчання ALS формує латентні ембеддинги як для користувачів, так і для об'єктів. Ці ембеддинги можуть бути використані не лише для побудови рекомендацій типу «користувач–об'єкт», а й для пошуку схожих об'єктів або користувачів шляхом обчислення подібності у латентному просторі. Проте ефективність таких рекомендацій значною мірою залежить від кількості взаємодій: для об'єктів із малою кількістю взаємодій, так званих long-

tail об'єктів, латентні представлення можуть бути нестабільними та близькими до шуму [17].

Обчислювальна складність ALS є відносно високою. Для однієї ітерації вона оцінюється як у рівнянні (2.19):

$$O(d^2[N_r + (|U||I|)d] + d^3), \quad (2.19)$$

де N_r – кількість ненульових взаємодій;

$|U|$ – кількість користувачів;

$|I|$ – кількість об'єктів;

d – розмірність латентного простору.

Незважаючи на це, ALS добре масштабується завдяки можливості паралельних обчислень, що робить його придатним для промислових рекомендаційних систем [18].

У межах цієї роботи алгоритм ALS використовується як колаборативний компонент гібридної рекомендаційної системи. Його латентні представлення поєднуються з результатами контентних методів TF-IDF та BERT, що дозволяє компенсувати обмеження ALS у випадках розріджених даних і проблеми холодного старту, водночас зберігаючи переваги моделювання прихованих поведінкових закономірностей.

2.5 Гібридний метод: зважений гібридний підхід

Гібридні рекомендаційні системи є одним із найефективніших підходів до подолання проблеми інформаційного перевантаження в сучасних онлайн-сервісах, де користувачам необхідно швидко отримувати персоналізовані та релевантні пропозиції. Традиційні методи рекомендацій, зокрема колаборативна фільтрація та контентна фільтрація, широко застосовуються у практичних системах, однак кожен з них має суттєві обмеження. Колаборативна фільтрація стикається з проблемами холодного старту та розрідженості даних, тоді як контентні підходи схильні до надмірної спеціалізації та обмеженого охоплення нових об'єктів. Унаслідок цього окреме використання одного методу

часто не дозволяє забезпечити стабільну якість рекомендацій у різних сценаріях використання системи.

Гібридні рекомендаційні системи поєднують кілька підходів – найчастіше колаборативну та контентну фільтрацію, а також у деяких випадках методи орієнтовані на знання – з метою використання сильних сторін кожного з них та компенсації їхніх недоліків. Такі системи можуть реалізовуватися за допомогою різних стратегій гібридизації, зокрема зваженого об'єднання результатів, перемикання між методами залежно від умов, об'єднання ознак, каскадних підходів або використання моделей одного методу як вхідних даних для іншого. Застосування гібридних стратегій дозволяє підвищити точність рекомендацій, зменшити вплив проблеми холодного старту та забезпечити більш стійку роботу системи за умов обмеженої або неповної інформації [19].

Зважений гібридний підхід є одним із найпоширеніших способів побудови гібридних рекомендаційних систем. Його основна ідея полягає в тому, що результати кількох незалежних рекомендаційних алгоритмів розглядаються як окремі складові, які об'єднуються у фінальний прогноз за допомогою лінійної комбінації з ваговими коефіцієнтами. Кожному алгоритму надається певна вага, що відображає його внесок у кінцеву рекомендацію, після чого всі зважені результати підсумовуються.

Нехай існує c різних рекомендаційних підходів, результати яких необхідно об'єднати. Тоді прогнозована оцінка або рейтинг p_{ui} для користувача u та об'єкта i у зваженій гібридній моделі обчислюється за формулою (2.20):

$$p_{ui} = \sum_{f=1}^c \sigma_f p_{ui}^{(f)}, \quad (2.20)$$

де $p_{ui}^{(f)}$ – прогноз, отриманий f -тим рекомендаційним алгоритмом;

σ_f – ваговий коефіцієнт, що визначає вплив відповідного алгоритму;

c – кількість комбінованих методів.

Ваги σ_f зазвичай нормалізуються таким чином, щоб їх сума дорівнювала одиниці, що забезпечує коректну інтерпретацію фінального прогнозу.

У випадку, коли зважений гібрид поєднує лише два рекомендаційні підходи ($c=2$), формула спрощується до вигляду, як показано у рівнянні (2.21):

$$p_{ui} = \sigma_1 p_{ui}^{(1)} + (1 - \sigma_1) p_{ui}^{(2)}, \quad (2.21)$$

де $\sigma_1 \in [0; 1]$ визначає баланс між двома методами. Така форма є зручною на практиці, оскільки дозволяє легко регулювати вплив кожного алгоритму, наприклад між контентною та колаборативною фільтрацією.

Важливим аспектом зважених гібридів є визначення оптимальних значень ваг. Згідно з дослідженнями, оптимальна вага σ_1 може бути знайдена шляхом мінімізації квадратичної похибки між фактичними значеннями та прогнозами, що обчислюються різними алгоритмами. Один із можливих варіантів обчислення оптимальної ваги має вигляд формули (2.22):

$$\sigma_1 = \frac{\sum_u \sum_i (p_{ui} - p_{ui}^{(2)}) (p_{ui}^{(1)} - p_{ui}^{(2)})}{\sum_u \sum_i (p_{ui}^{(1)} - p_{ui}^{(2)})^2}, \quad (2.22)$$

де p_{ui} – фактичне значення або еталонний рейтинг;
 $p_{ui}^{(1)}$ та $p_{ui}^{(2)}$ – прогнози двох комбінованих методів.

Ця формула дозволяє знайти таке значення ваги, яке забезпечує мінімальну середньоквадратичну похибку гібридної моделі.

Перевагою зважених гібридних систем є їхня простота реалізації, гнучкість та можливість адаптації до різних доменів. Завдяки комбінуванню методів вони здатні компенсувати слабкі сторони окремих алгоритмів, наприклад проблему холодного старту в колаборативній фільтрації або надмірну спеціалізацію контентних методів. Крім того, зважені гібриди легко розширюються на більшу кількість компонентів, що робить їх придатними для побудови складних багатокомпонентних рекомендаційних систем [20].

2.6 Метрики оцінки якості

Коефіцієнт істинно позитивних результатів або True Positive Rate, TPR, який також відомий як повнота, або Recall, характеризує частку фактично позитивних об'єктів, що були вірно ідентифіковані моделлю. Дана метрика відображає здатність моделі знаходити всі релевантні випадки у вибірці та часто інтерпретується як імовірність виявлення позитивного класу. Формально

повнота визначається як відношення кількості істинно позитивних результатів до суми істинно позитивних і хибнонегативних результатів. Хибнонегативні результати відповідають тим об'єктам, які фактично належать до позитивного класу, але були помилково класифіковані як негативні, що безпосередньо впливає на значення повноти.

У задачах класифікації, зокрема в задачі фільтрації спаму, повнота відображає частку небажаних повідомлень, які модель змогла правильно ідентифікувати. Ідеальна модель характеризувалася б відсутністю хибнонегативних результатів, що відповідало б значенню повноти, рівному одиниці. Водночас у випадках, коли набір даних є незбалансованим і кількість позитивних прикладів є дуже малою, інтерпретаційна цінність повноти може зменшуватися, оскільки навіть незначна кількість помилок суттєво впливає на результат. Це продемонстровано у рівнянні (2.24).

Влучність, або Precision визначає частку об'єктів, як показано у рівнянні (2.23), класифікованих моделлю як позитивні, які дійсно належать до позитивного класу. Ця метрика характеризує надійність позитивних передбачень моделі та формально обчислюється як відношення кількості істинно позитивних результатів до суми істинно позитивних і хибнопозитивних результатів. У контексті класифікації спаму влучність показує, яка частка повідомлень, визначених як спам, дійсно є небажаними. За відсутності хибнопозитивних результатів значення влучності дорівнювало б одиниці.

$$Precision = \frac{tp}{tp+fp} \quad (2.23)$$

$$Recall(True Positive Rate) = \frac{tp}{tp+fn} \quad (2.24)$$

Між метриками влучності та повноти зазвичай існує компроміс. Зменшення кількості хибнопозитивних результатів призводить до зростання влучності, однак часто супроводжується збільшенням кількості хибнонегативних результатів і, відповідно, зниженням повноти. Навпаки, зменшення кількості хибнонегативних результатів підвищує повноту, але може негативно впливати на влучність. Такий взаємозв'язок обумовлений вибором

порогового значення класифікації, що робить ці метрики самодостатніми та потребує їх спільного аналізу при оцінюванні якості моделей [21].

Нормалізована дисконтована кумулятивна вигода, або NDCG є однією з базових метрик оцінювання якості ранжування в інформаційно-пошукових та рекомендаційних системах. На відміну від простих метрик точності чи повноти, NDCG враховує не лише факт релевантності рекомендованих об'єктів, а й їхню позицію у відсортованому списку. Це є принципово важливим, оскільки для користувача найбільшу цінність мають саме ті елементи, які знаходяться на верхніх позиціях рекомендаційного списку.

Основою метрики NDCG є показник дисконтованої кумулятивної вигоди (DCG), який обчислюється як зважена сума релевантностей об'єктів з урахуванням їх позицій у рейтингу. Формально DCG на глибині K визначається так, як показано у рівнянні (2.25):

$$DCG@K = \sum_{k=1}^K \frac{rel_i}{\log_2(i+1)}, \quad (2.25)$$

де rel_i – оцінка релевантності об'єкта на позиції i , а логарифмічний знаменник виконує роль дисконтуючого множника. Таким чином, вклад елементів, що знаходяться нижче в списку, зменшується, що відображає реальну поведінку користувачів, які рідко переглядають рекомендації на далеких позиціях. Наприклад, для списку з п'яти елементів $DCG@4$ обчислюється як показано у рівнянні (2.26):

$$DCG@4 = \frac{rel_1}{\log_2(1+1)} + \frac{rel_2}{\log_2(2+1)} + \frac{rel_3}{\log_2(3+1)} + \frac{rel_4}{\log_2(4+1)} \quad (2.26)$$

Оскільки значення DCG залежить від конкретного набору релевантностей, для коректного порівняння різних моделей використовується нормалізація. Для цього вводиться ідеальна дисконтована кумулятивна вигода (IDCG), яка відповідає максимально можливому значенню DCG за умови ідеального впорядкування об'єктів – тобто коли найрелевантніші елементи розташовані на верхніх позиціях. Після цього NDCG визначається як відношення фактичного DCG до ідеального, продемонстровано у рівнянні (2.27):

$$NDCG@K = \frac{DCG@K}{IDCG@K} \quad (2.27)$$

Завдяки такій нормалізації значення NDCG завжди належить інтервалу $[0,1][0,1][0,1]$, де 1 відповідає ідеальному ранжуванню [22].

У практичних задачах рекомендаційних систем найчастіше використовуються варіанти метрики з фіксованою глибиною списку, наприклад $NDCG@5$, $NDCG@10$ або $NDCG@100$. Метрика $NDCG@K$ дозволяє оцінити, наскільки добре система розташовує релевантні об'єкти саме в межах перших K рекомендацій, що безпосередньо відповідає реальним сценаріям використання системи. У контексті даної роботи NDCG застосовується для порівняльної оцінки контентних, колаборативних та гібридних методів рекомендацій, оскільки вона найбільш повно відображає якість упорядкування вакансій за ступенем їх релевантності для користувача.

2.7 Висновки до розділу 2

У другому розділі було розглянуто теоретичні та математичні засади побудови рекомендаційних систем, які стали основою для подальшої практичної реалізації та експериментального дослідження. Детально проаналізовано ключові підходи до рекомендацій, зокрема контентну фільтрацію, колаборативну фільтрацію та методи матричної факторизації, а також сучасні нейромережеві моделі обробки тексту, зокрема BERT. Це дозволило сформуванню цілісного уявлення про механізми формування рекомендацій і їхні сильні та слабкі сторони.

Окрему увагу було приділено математичному апарату, який лежить в основі кожного з методів. Розглянуто векторну просторову модель, TF-IDF – зважування та косинусну подібність як базові інструменти контентної фільтрації, а також принципи колаборативної фільтрації й матричної факторизації, включно з алгоритмом ALS. Показано, що ці методи дозволяють ефективно працювати з розрідженими даними та виявляти приховані закономірності у взаємодіях користувачів з об'єктами рекомендацій.

Також у розділі було описано підходи до гібридизації рекомендаційних систем, зокрема зважені гібридні моделі, які поєднують результати кількох алгоритмів для підвищення точності та стійкості рекомендацій. Обґрунтовано доцільність використання гібридних методів для подолання таких проблем, як холодний старт, розрідженість даних та обмеженість окремих підходів.

Завершальним елементом розділу став огляд метрик оцінювання якості рекомендацій, зокрема Precision, Recall та NDCG. Показано, що метрика NDCG є найбільш придатною для оцінки задач ранжування, оскільки враховує не лише релевантність рекомендованих об'єктів, а й їх позиції у списку. Отримані теоретичні висновки створюють надійну основу для реалізації, експериментальної перевірки та порівняльного аналізу рекомендаційних моделей у наступному розділі роботи.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ГІБРИДНОЇ СИСТЕМИ ТА ОГЛЯД ЕКСПЕРИМЕНТІВ

У цьому розділі представлено практичну реалізацію створеної рекомендаційної системи та опис проведених експериментів, що покликані підтвердити ефективність запропонованих методів. На основі теоретичних засад, розглянутих у попередніх розділах, реалізовано повний цикл побудови гібридного рекомендаційного підходу, який поєднує контентні, семантичні та колаборативні методи.

Розробка здійснювалася мовою Python із використанням бібліотек Pandas, NumPy, Scikit-learn, NLTK, TensorFlow/Transformers, а також алгоритмів матричної факторизації та сучасних моделей подання текстів. Реалізована система включає компоненти для обробки даних, побудови векторних представлень, генерації рекомендацій, а також модулів гібридизації та механізмів холодного старту.

У межах експериментальної частини проведено оцінювання роботи системи на основі метрик Precision@K, Recall@K та NDCG. Також досліджено вплив різних моделей таких як: TF-IDF, BERT, ALS/NMF і їх комбінацій на якість рекомендацій. Запропонований експериментальний підхід дозволяє комплексно оцінити переваги гібридної моделі та обґрунтувати вибір параметрів для її практичного застосування.

3.1 Опис та аналіз синтетичного датасету

У межах практичної частини дослідження було згенеровано синтетичний датасет, який моделює типовий набір даних, що використовується у системах рекомендації вакансій. Датасет складається з трьох взаємопов'язаних компонентів: профілів кандидатів, описів вакансій та журналу взаємодій між ними. Структура даних побудована таким чином, щоб максимально наближено

відтворювати реальні сценарії ринку праці: наявність професійних кластерів, варіативність рівнів кваліфікації, нерівномірність розподілу навичок, а також природний характер відгуків кандидатів на різні вакансії.

Перший блок складається з даних про кандидатів. Для кожного кандидата генерується набір із фіксованої кількості навичок, представлених у вигляді бінарного вектора. Навички моделюються методом кластеризації навколо заздалегідь визначених професійних центрів, що дозволяє створити групи кандидатів з подібним професійним профілем. Окрім навичок, кожен кандидат має унікальний ідентифікатор, приналежність до певного професійного кластера, а також значення стажу роботи, яке додає варіативність та імітує різний рівень підготовки. На основі бінарного набору навичок формується текстовий профіль кандидата – стисла характеристика, яка містить інформацію про досвід та ключові компетенції.

Другий компонент датасету – вакансії. Аналогічно до кандидатів, навички вакансій генеруються методом штучного створення кластеризованих даних, що дозволяє моделювати групи професій, споріднені за вимогами. Для кожної вакансії визначається компанія-роботодавець, професійні рівні, такі як: junior, middle або senior, кластер, до якого вона належить, а також бінарний вектор навичок, які потрібні на цю позицію. На основі цього набору формується текстовий опис вакансії, який включає рівень, тип позиції та перелік найважливіших навичок. У результаті кожна вакансія має як структуроване числове представлення, так і текстове пояснення, що відображає її змістовий профіль.

Третя складова – журнал взаємодій між кандидатами та вакансіями, що моделює реальні відгуки. Для кожної можливої пари “кандидат – вакансія” обчислюється ймовірність взаємодії, яка залежить від двох чинників: належності до одного професійного кластера та ступеня збігу навичок. Для розрахунку використовується нормована міра перетину навичок, яка відображає відсоток релевантності вимог вакансії до компетенцій кандидата. Якщо ймовірність перевищує встановлений поріг і випадковий вибір дозволяє,

у таблицю взаємодій додається запис про умовний «відгук» кандидата, який містить ідентифікатори обох сутностей, числову оцінку «rating», що наближено відображає якість відповідності, а також показник збігу навичок. Через високу розрідженість, характерну для реальних систем підбору персоналу, кількість фактичних взаємодій становить лише малу частку від усіх можливих комбінацій, що створює реалістичну матрицю спостережень.

Усі три компоненти датасету узгоджені між собою логічно: професійні кластери формують природну структуру як серед кандидатів, так і серед вакансій; навички задають ключовий змістовий профіль обох сторін; текстові описи слугують лінгвістичними представленнями тих самих структурованих даних; взаємодії відображають закономірності ринку праці, де більша релевантність призводить до більшої ймовірності відгуку. У результаті синтетичний датасет відтворює типову багатокомпонентну екосистему, необхідну для аналізу роботи рекомендаційних систем, дослідження впливу різних ознак та проведення подальших експериментів у рамках колаборативних, контентних та гібридних методів. На рисунках 3.1, 3.2 та 3.3 продемонстровано вивід перших п'яти вакансій, кандидатів та їх взаємодій.

job_id	cluster	level	company	description
0	0	2	junior Company_0	junior manager at Company_0. Required skills: ...
1	1	6	middle Company_1	middle manager at Company_1. Required skills: ...
2	2	0	middle Company_2	middle developer at Company_2. Required skills...
3	3	4	middle Company_3	middle developer at Company_3. Required skills...
4	4	7	middle Company_4	middle designer at Company_4. Required skills:...

Рисунок 3.1 – перші п'ять вакансій

	candidate_id	cluster	experience	profile
0	0	0	0	Candidate with 0 years experience. Skills: SKI...
1	1	3	1	Candidate with 1 years experience. Skills: SKI...
2	2	0	3	Candidate with 3 years experience. Skills: SKI...
3	3	5	5	Candidate with 5 years experience. Skills: SKI...
4	4	7	0	Candidate with 0 years experience. Skills: SKI...

Рисунок 3.2 – перші п'ять кандидатів

	candidate_id	job_id	rating	skill_match
0	0	60	3	0.73
1	0	62	2	0.47
2	0	85	2	0.45
3	0	87	3	0.58
4	0	104	2	0.70

Рисунок 3.3 – перші п'ять взаємодій між кандидатами та вакансіями

3.2 Реалізація контентного методу фільтрації

Контентний метод фільтрації у системі рекомендації вакансій базується на представленні текстової інформації у вигляді векторів ознак, що дозволяє вимірювати схожість між профілем кандидата та описами вакансій на основі їхнього змісту. У даній роботі для цього використано класичний підхід TF–IDF або «term frequency – inverse document frequency», який перетворює текст на числове представлення, що відображає важливість окремих слів у документі відносно всієї колекції текстів. TF–IDF дає можливість підсилити вагу змістовно значущих термінів, таких як назви навичок або згадки технологій, і одночасно зменшити вплив загальних слів, що часто зустрічаються, але не несуть релевантної інформації.

У структурі датасету кожна вакансія має текстове поле, сформоване з рівня посади, типу ролі, назви компанії та переліку ключових навичок. Аналогічно для кандидатів створюється текстовий профіль, який описує їхній досвід і навички. Тексти цих двох типів документів об'єднуються в єдиний корпус, над яким навчений TF-IDF-векторизатор визначає словник ознак і будує числові вектори. Вектор кандидата відображає зміст його професійного профілю, а вектори вакансій – змістові вимоги кожної позиції. Завдяки цьому кандидат і вакансії представлені у спільному просторовому середовищі, де близькість між ними має чітку змістову інтерпретацію.

Після побудови TF-IDF-векторів обчислюється косинусна подібність між вектором кандидата та всіма векторами вакансій. Ця метрика вимірює кутову відстань між двома текстовими представленнями: чим вона менша, тим більша подібність профілю кандидата до вимог вакансії. Значення подібності лежать у межах від 0 до 1, де 1 означає повний збіг за змістовими ознаками. На основі отриманих значень формується рейтинг вакансій за ступенем відповідності кандидату. Результати метода TF-IDF для користувачів 0, 1, 2, 42 та 100 продемонстровано на рисунках 3.4 – 3.8 відповідно.

```
TFIDF: 5 рекомендацій
Job 468: Score 0.228
Job 58: Score 0.224
Job 379: Score 0.224
Job 203: Score 0.223
Job 267: Score 0.223
```

Рисунок 3.4 – результати метода TF-IDF для кандидата 0

```
TFIDF: 5 рекомендацій
Job 364: Score 0.238
Job 162: Score 0.214
Job 107: Score 0.211
Job 442: Score 0.208
Job 312: Score 0.206
```

Рисунок 3.5 – результати метода TF-IDF для кандидата 1

TFIDF: 5 рекомендацій
 Job 39: Score 0.206
 Job 134: Score 0.204
 Job 496: Score 0.203
 Job 297: Score 0.203
 Job 199: Score 0.202

Рисунок 3.6 – результати метода TF–IDF для кандидата 2

Job 483: Score 0.198
 Job 291: Score 0.192
 Job 41: Score 0.179
 Job 362: Score 0.173
 Job 138: Score 0.171

Рисунок 3.7 – результати метода TF–IDF для кандидата 42

Job 435: Score 0.156
 Job 75: Score 0.154
 Job 337: Score 0.154
 Job 378: Score 0.154
 Job 342: Score 0.153

Рисунок 3.8 – результати метода TF–IDF для кандидата 100

Окремим етапом є виключення вакансій, з якими кандидат уже взаємодіяв. Для цього аналізується таблиця відгуків, у якій містяться всі попередні перегляди або подані заявки. Такі вакансії відкидаються із загального списку, після чого ранжування проводиться лише серед нових, ще не розглянутих кандидатами позицій. У підсумку система повертає впорядкований список вакансій, найбільш релевантних змісту профілю конкретного користувача.

Таким чином, контентний метод на основі TF–IDF забезпечує рекомендації, спираючись виключно на текстове наповнення обох сторін – профілю кандидата і вимог вакансії. Він дозволяє отримувати точні рекомендації навіть у випадках відсутності історії взаємодій, а саме сценарію холодного старту, оскільки його робота не залежить від поведінкових даних, а виключно від змісту текстів у профілях і описах.

3.3 Реалізація колаборативного методу фільтрації

Колаборативний метод фільтрації у системі рекомендації вакансій ґрунтується на аналізі взаємодій між кандидатами та вакансіями, що відображаються у вигляді оцінок або відгуків. На відміну від контентного підходу, який працює з текстовими описами та навичками, колаборативний метод оперує виключно поведінковими даними – тим, як саме кандидати реагували на різні вакансії у минулому. Основна ідея полягає в тому, що якщо дві вакансії отримують подібні відгуки від великої групи кандидатів, вони можуть бути взаємозамінними з погляду релевантності; аналогічно, кандидати з подібною історією відгуків мають спільні переваги на ринку праці.

У сформованому синтетичному датасеті кожен запис взаємодії містить інформацію про кандидата, вакансію та числовий рейтинг, який моделює ступінь релевантності навичок кандидата вимогам позиції. На основі таких записів формується розріджена матриця взаємодій «кандидат–вакансія», де рядки відповідають окремим кандидатам, стовпці – вакансіям, а значення у матриці відображають інтенсивність або якість взаємодії. Розрідженість такої матриці є типовою для задач рекрутингу, оскільки кожен кандидат розглядає лише невелику частину від усіх доступних вакансій.

Для побудови рекомендацій у роботі використано метод матричної факторизації Alternating Least Squares (ALS). Цей підхід передбачає розкладання великої розрідженої матриці відгуків на дві компактні матриці нижчої розмірності: матрицю прихованих факторів кандидатів і матрицю прихованих факторів вакансій. Кожен кандидат таким чином описується набором латентних ознак, що відображають його професійний профіль, а кожна вакансія – набором ознак, що характеризують вимоги та очікування роботодавця. У процесі навчання моделі ALS ці фактори оптимізуються таким чином, щоб відтворити спостережувані оцінки і мінімізувати похибку реконструкції.

Після факторизації модель отримує здатність прогнозувати умовний рейтинг для будь-якої пари «кандидат–вакансія», навіть якщо прямої взаємодії не було. Це робить можливим виявлення прихованих закономірностей у даних, наприклад подібності між вакансіями, що належать до одного професійного напрямку, або між кандидатами з подібними профілями оцінок. На практиці для кожного кандидата генерується список вакансій із найвищими прогнозованими оцінками, що і формує персоналізовані рекомендації. Результати відпрацьованого методу продемонстровані на рисунках 3.9 – 3.13.

```
ALS: 5 рекомендацій
Job 48: Score 0.353
Job 221: Score 0.310
Job 151: Score 0.302
Job 490: Score 0.302
Job 0: Score 0.294
```

Рисунок 3.9 – результати метода ALS для кандидата 0

```
Job 265: Score 0.190
Job 121: Score 0.144
Job 330: Score 0.142
Job 123: Score 0.139
Job 20: Score 0.137
```

Рисунок 3.10 – результати метода ALS для кандидата 1

```
ALS: 5 рекомендацій
Job 495: Score 0.362
Job 236: Score 0.312
Job 206: Score 0.312
Job 421: Score 0.295
Job 392: Score 0.295
```

Рисунок 3.11 – результати метода ALS для кандидата 2

```
Job 61: Score 0.207
Job 168: Score 0.206
Job 295: Score 0.186
Job 65: Score 0.168
Job 181: Score 0.167
```

Рисунок 3.12 – результати метода ALS для кандидата 42

```

Job 192: Score 0.470
Job 9: Score 0.373
Job 38: Score 0.360
Job 228: Score 0.359
Job 1: Score 0.348

```

Рисунок 3.13 – результати метода ALS для кандидата 100

Колаборативний метод на основі ALS має кілька важливих переваг. По-перше, він ефективно справляється з розрідженістю даних і здатен працювати в умовах, коли більшість кандидатів взаємодіє лише з малою частиною вакансій. По-друге, модель виявляє неочевидні відповідності, які важко визначити за допомогою контентного аналізу, адже приховані фактори узагальнюють поведінкові патерни великої кількості користувачів. По-третє, факторизація дозволяє системі робити рекомендації навіть для вакансій з мінімальною кількістю взаємодій, якщо в них є структурна подібність до інших.

У підсумку, метод ALS забезпечує формування рекомендацій на основі поведінкової інформації та демонструє здатність до виявлення глибинних структур відповідності між кандидатами і вакансіями. Він є ключовим компонентом колаборативної фільтрації та посилює можливості системи у випадках, коли текстові або контентні ознаки не дають достатньої інформації про релевантність.

3.4 Метод BERT та пошук семантичної подібності

Контентний метод фільтрації на основі BERT використовується для визначення семантичної близькості між текстовим профілем кандидата та описами вакансій. На відміну від підходу TF-IDF, який працює переважно з частотністю термінів і не враховує контексту, модель BERT, або Bidirectional Encoder Representations from Transformers дозволяє формувати глибокі контекстні векторні представлення тексту. Завдяки цьому система здатна вловлювати семантичні зв'язки між словами, фразами та структурою опису, що значно підвищує точність рекомендацій у задачах, де текстові дані містять складні смислові конструкції.

У даній роботі використано попередньо натреновану модель Sentence-BERT (all-MiniLM-L6-v2), адаптовану спеціально для вимірювання подібності між текстами. Вона перетворює кожний текст – як профіль кандидата, так і опис вакансії – на компактний вектор фіксованої розмірності. На відміну від класичного BERT, Sentence-BERT оптимізований саме для задач пошуку схожості та ранжування, тому формує вектори, які можна безпосередньо порівнювати за допомогою косинусної подібності.

Перед тим як обчислити подібність, система формує два типи текстів: профілі кандидатів та описи вакансій. Профіль кандидата містить інформацію про досвід, ключові технічні навички та коротку характеристику компетенцій. Опис вакансії включає професійний рівень, тип посади, назву компанії та перелік вимог, сформованих на основі виявлених у даних навичок. Ці тексти подаються до моделі Sentence-BERT, яка генерує їхні семантичні векторні представлення. У результаті кожен документ має узагальнений смисловий вектор, що відображає приховану структуру змісту.

Після побудови векторів обчислюється косинусна подібність між профілем кандидата та кожним описом вакансії. Це дозволяє визначити, наскільки близькими є тексти за змістом, навіть якщо вони не мають однакових слів. Наприклад, кандидат, який у профілі має «Python development», буде розпізнаний як релевантний вакансії з вимогою «experience in backend programming», навіть якщо між текстами мало прямих збігів за термінами. Такі зв'язки виявляються завдяки контекстному розумінню, властивому BERT-моделям.

Після обчислення семантичної близькості вакансії сортуються за спаданням подібності, і кандидату пропонується перелік найбільш релевантних позицій. Додатково система виключає вакансії, з якими кандидат уже взаємодівав раніше, спираючись на матрицю відгуків. У підсумку формується ранжований список нових вакансій, що найкраще відповідають семантичному змісту його профілю. Результати відпрацьованого метода BERT продемонстровано на рисунках 3.14 – 3.18.

```

BERT: 5 рекомендацій
Job 49: Score 0.729
Job 248: Score 0.721
Job 327: Score 0.719
Job 373: Score 0.719
Job 146: Score 0.719

```

Рисунок 3.14 – результати роботи метода BERT для кандидата 0

```

Job 77: Score 0.763
Job 278: Score 0.762
Job 326: Score 0.761
Job 327: Score 0.760
Job 44: Score 0.760

```

Рисунок 3.15 – результати роботи метода BERT для кандидата 1

```

Job 327: Score 0.739
Job 373: Score 0.737
Job 278: Score 0.737
Job 228: Score 0.736
Job 93: Score 0.735

```

Рисунок 3.16 – результати роботи метода BERT для кандидата 2

```

Job 77: Score 0.756
Job 228: Score 0.753
Job 327: Score 0.749
Job 389: Score 0.746
Job 146: Score 0.746

```

Рисунок 3.17 – результати роботи метода BERT для кандидата 42

```

Job 60: Score 0.766
Job 360: Score 0.758
Job 460: Score 0.756
Job 373: Score 0.755
Job 344: Score 0.755

```

Рисунок 3.18 – результати роботи метода BERT для кандидата 100

Таким чином, метод BERT є більш глибинним та інтелектуальним варіантом контентної фільтрації, оскільки дозволяє працювати не лише з термінами, а й із контекстом, значенням і прихованими семантичними зв'язками у текстах. Він здатен успішно формувати рекомендації навіть для коротких або

неструктурованих профілів, а також забезпечує підвищену точність у порівнянні зі статистичними методами на кшталт TF-IDF.

3.5 Проблема холодного старту

Однією з ключових проблем рекомендаційних систем є проблема «холодного старту», яка виникає тоді, коли для нового користувача або нового об'єкта (вакансії) відсутні історичні дані взаємодії. У такій ситуації класичні алгоритми колаборативної фільтрації не можуть ефективно працювати, оскільки вони повністю ґрунтуються на матриці відгуків та поведінкових оцінках інших користувачів. Тому для підвищення стійкості системи та забезпечення можливості формування рекомендацій у подібних умовах у розробленій системі реалізовано спеціальний механізм обробки холодного старту, який складається з двох основних рівнів: контентного резервного механізму та глобальної стратегії fallback-рекомендацій.

Перший рівень – це використання контентних моделей TF-IDF і BERT у ситуаціях, коли дані колаборативної фільтрації недоступні. Якщо кандидат є новим користувачем системи й ще не має жодних взаємодій із вакансіями, колаборативний підхід не здатен побудувати вектор користувача або знайти схожих кандидатів. У цьому випадку система автоматично переходить до контентних моделей. Оскільки профіль кандидата та описи вакансій містять текстову інформацію про навички, досвід та вимоги, модель TF-IDF формує векторне подання текстів на основі частоти термінів, а модель BERT генерує контекстуальні семантичні вектори. Ці вектори дають змогу оцінити схожість між профілем кандидата та вакансіями незалежно від того, чи існують у системі попередні рейтинги або відгуки. Таким чином, контентні методи виступають універсальним джерелом рекомендацій для нових користувачів, компенсуючи відсутність історичних даних.

Другий рівень – це глобальний fallback-механізм, реалізація якого продемонстрована на рисунках 3.19 та 3.20, який застосовується у випадку, коли як контентні моделі, так і колаборативна фільтрація не повертають достатньої

кількості рекомендованих вакансій. Подібна ситуація може виникати, наприклад, тоді, коли профіль кандидата містить дуже мало інформації або сформований виключно з нестандартних чи рідкісних навичок. У такому випадку застосовується стратегія рекомендації найпопулярніших вакансій, що визначаються на основі середніх рейтингів або кількості взаємодій у загальній матриці. Такий підхід дозволяє гарантувати, що користувач у будь-якому випадку отримає релевантні й затребувані позиції, які користуються найбільшим попитом серед інших кандидатів.

```
if not content_tfidf and not content_bert and not cf_als:
    popular_jobs = (
        interactions_df
        .groupby('job_id')['rating']
        .mean()
        .sort_values(ascending=False)
        .head(top_n)
    )
    return [{'job_id': job_id, 'score': score}
            for job_id, score in popular_jobs.items()]
```

Рисунок 3.19 – реалізація Fullback-механізму

```
test_users_without_train = 0
for user_id in test_data['candidate_id'].unique():
    if user_id not in train_data['candidate_id'].values:
        test_users_without_train += 1

print(f"Тестовых пользователей без train данных: {test_users_without_train}")
```

Рисунок 3.20 – тест Fallback-механізму

Завдяки поєднанню цих стратегій у гібридній моделі проблема холодного старту фактично усувається, що продемонстровано у результатах на рисунку 3.21. Для нових кандидатів система завжди матиме можливість сформувати рекомендації завдяки контентним моделям, а в разі слабко заповнених профілів або рідкісних випадків працює резервний механізм популярних вакансій. У результаті система залишається працездатною та стабільною для всіх категорій користувачів, що суттєво підвищує її надійність, універсальність та практичну цінність.

	method	rank	job_id	score	level	company	description_short
0	TF-IDF	1	468	0.2281	middle	Company_18	middle analyst at Company_18. Required skills:...
1	TF-IDF	2	58	0.2238	middle	Company_8	middle developer at Company_8. Required skills:...
2	TF-IDF	3	379	0.2238	middle	Company_29	middle developer at Company_29. Required skill...
3	TF-IDF	4	203	0.2231	middle	Company_3	middle analyst at Company_3. Required skills: ...
4	TF-IDF	5	267	0.2228	middle	Company_17	middle manager at Company_17. Required skills:...
5	BERT	1	49	0.7289	senior	Company_49	senior analyst at Company_49. Required skills:...
6	BERT	2	248	0.7207	senior	Company_48	senior manager at Company_48. Required skills:...
7	BERT	3	327	0.7188	senior	Company_27	senior analyst at Company_27. Required skills:...
8	BERT	4	373	0.7186	senior	Company_23	senior analyst at Company_23. Required skills:...
9	BERT	5	146	0.7186	senior	Company_46	senior manager at Company_46. Required skills:...
10	ALS	1	48	0.4117	middle	Company_48	middle manager at Company_48. Required skills:...
11	ALS	2	0	0.3493	junior	Company_0	junior manager at Company_0. Required skills: ...
12	ALS	3	286	0.3186	middle	Company_36	middle analyst at Company_36. Required skills:...
13	ALS	4	93	0.2795	senior	Company_43	senior analyst at Company_43. Required skills:...
14	ALS	5	20	0.2776	senior	Company_20	senior analyst at Company_20. Required skills:...
15	Hybrid (TF-IDF+BERT+ALS)	1	49	0.2781	senior	Company_49	senior analyst at Company_49. Required skills:...
16	Hybrid (TF-IDF+BERT+ALS)	2	248	0.2162	senior	Company_48	senior manager at Company_48. Required skills:...
17	Hybrid (TF-IDF+BERT+ALS)	3	327	0.2156	senior	Company_27	senior analyst at Company_27. Required skills:...
18	Hybrid (TF-IDF+BERT+ALS)	4	373	0.2156	senior	Company_23	senior analyst at Company_23. Required skills:...
19	Hybrid (TF-IDF+BERT+ALS)	5	146	0.2156	senior	Company_46	senior manager at Company_46. Required skills:...

Рисунок 3.21 – результати вирішення проблема холодного старту

3.6 Гібридний метод

Гібридний метод рекомендацій є ключовим компонентом розробленої системи, оскільки поєднує сильні сторони різних підходів – контентних моделей (TF-IDF та BERT) і колаборативної фільтрації (ALS). Така комбінація дозволяє побудувати більш точні, стійкі та адаптивні рекомендації, ніж будь-яка окрема модель, особливо в умовах неоднорідності даних та масштабності задачі підбору вакансій. Основна ідея гібридної стратегії полягає у тому, що кожен метод генерує власні оцінки релевантності вакансій, після чого ці оцінки об'єднуються у спільний інтегрований рейтинг з урахуванням ваг, які визначають внесок кожного підходу у фінальне рішення.

У рамках реалізованого рішення контентні моделі TF-IDF та BERT обчислюють семантичну близькість між текстовим профілем кандидата та

описами вакансій. TF-IDF відображає схожість на основі лексичних збігів та частотності термінів, тоді як BERT забезпечує глибинне контекстне розуміння смислу текстів, дозволяючи точно розпізнавати професійні відповідності навіть у тому разі, коли термінологія між профілем і вимогами вакансії відрізняється. У свою чергу, колаборативна модель ALS працює з матрицею взаємодій «кандидат – вакансія» і оцінює ймовірність зацікавленості на основі поведінки схожих користувачів. На відміну від контентних підходів, ALS опирається на приховані патерни у відгуках і здатна знаходити вакансії, які можуть бути релевантні кандидату навіть за відсутності явних семантичних збігів у текстах.

Після того як кожен метод формує власний список рекомендацій, гібридний блок об'єднує їх шляхом зваженого агрегування оцінок релевантності. У реалізації системи передбачено три вагові коефіцієнти: для TF-IDF, для BERT та для ALS. Значення ваг визначаються експериментально та можуть змінюватися залежно від цілей системи, точності моделей або структури даних. Наприклад, у базовій конфігурації використовується формула (3.1):

$$score_{hybrid} = 0.3 \times tf - idf + 0.3 \times BERT + 0.4 \times ALS, \quad (3.1)$$

що підкреслює дещо більшу роль колаборативної фільтрації, оскільки ALS часто забезпечує найкращі результати за рахунок узагальнення поведінкових патернів. Під час об'єднання система створює словник усіх вакансій, які з'явилися хоча б в одному з методів, а потім сумує їхні вагові бали. У підсумку формується єдиний ранжований список, який об'єднує рекомендації на основі текстових та поведінкових факторів.

Особливу роль у гібридному методі відіграє його стійкість до неповних або нестандартних даних. Якщо один з підходів не може надати рекомендацій, наприклад, ALS не працює для нового кандидата, який ще не має взаємодій, система все ще формує релевантні вакансії завдяки контентним моделям. А в разі, коли жоден підхід не повертає результатів, гібридний модуль автоматично активує fallback-механізм і пропонує найбільш популярні вакансії. Завдяки

цьому гібридна модель працює стабільно для всіх категорій користувачів: як для новачків, так і для досвідчених кандидатів з великою історією взаємодій.

3.7 Опис проведених експериментів

Для оцінки ефективності розробленої рекомендаційної системи було проведено три великі експериментальні серії, спрямовані на комплексне порівняння різних алгоритмів, дослідження впливу структурних параметрів даних та визначення оптимальної конфігурації гібридної моделі. Зокрема, експериментальна частина включає: оцінку точності кожного з методів – TF-IDF, BERT, ALS та гібридної моделі; тестування системи на наборах даних із різною кількістю навичок у синтетичному датасеті; дослідження впливу різних вагових коефіцієнтів у гібридній системі на якість рекомендацій. Сукупно ці дослідження дозволяють глибинно проаналізувати поведінку системи та визначити оптимальні умови її роботи.

3.7.1 Експеримент точності на різних метриках якості

У першому експерименті було проведено порівняння основних методів рекомендацій: контентних моделей TF-IDF та BERT, колаборативної фільтрації ALS, а також гібридного підходу, який поєднує всі три компоненти. Дані було розділено на навчальну та тестову вибірки, після чого для кожного методу формувався список рекомендацій для підмножини тестових кандидатів. Для оцінки використовувалися стандартні метрики рекомендаційних систем: Precision@K , Recall@K та NDCG, що дають змогу оцінити як відповідність рекомендацій релевантним вакансіям, так і коректність їхнього ранжування. Такий підхід дозволяє визначити, наскільки ефективно кожен метод обробляє текстові чи поведінкові дані, а також оцінити переваги гібридної моделі. Результати експерименту продемонстровані на рисунку 3.22

```

Успешные оценки по методам:
tfidf: 2/30
bert: 1/30
als: 1/30
hybrid: 1/30

```

```

=== ИСПРАВЛЕННЫЕ РЕЗУЛЬТАТЫ ===

```

Метод	Precision	Recall	NDCG	Success Rate
TFIDF	0.1000	1.0000	0.5655	0.07
BERT	0.1000	1.0000	0.2891	0.03
ALS	0.1000	1.0000	0.3333	0.03
HYBRID	0.1000	1.0000	0.5000	0.03

Рисунок 3.22 – результати експерименту оцінки методів

Результати першого експерименту засвідчили, що всі протестовані методи показали приблизно однакову поведінку за базовими метриками Precision@K і Recall@K. Для всіх моделей значення Precision становило 0.1, а Recall – 1.0, що пов'язано з тим, що в більшості випадків серед рекомендованих вакансій містився хоча б один релевантний об'єкт із тестової вибірки.

Більш інформативною виявилася метрика NDCG, яка оцінює коректність ранжування. Саме вона продемонструвала відмінності між методами. Найкращий результат показав алгоритм ALS (NDCG = 0.3333), що відповідає природі колаборативної фільтрації: він більш ефективно використовує структуру поведінкових даних для визначення вакансій, які варто підняти вище у списку рекомендацій. Контентна модель TF-IDF також продемонструвала відносно високий показник (NDCG $\approx$ 0.5655), що свідчить про її непогану здатність ранжувати вакансії за схожістю змісту. Модель BERT виявилася дещо менш точною (NDCG $\approx$ 0.2891), що може бути пов'язано з тим, що семантичні BERT-вектори більш узагальнені, а синтетичні текстові описи вакансій мають значно простішу структуру, ніж реальні, що зменшує перевагу глибинних моделей. Гібридна модель показала середній результат (NDCG = 0.5000), що цілком очікувано, адже вона поєднує сигнали всіх методів, але не отримує суттєвої переваги без оптимізації ваг.

3.7.2 Експеримент з різною розмірністю навичок у кандидата

Другий експеримент був спрямований на вивчення впливу кількості навичок на якість рекомендацій. Задля цього синтетичний датасет було повторно згенеровано у трьох варіантах: з 20, 50 та 100 навичками. Для кожної конфігурації система повторно запускала оцінку TF-IDF, BERT, ALS та гібридного підходу, після чого результати відображалися у вигляді графіків зміни метрик залежно від розмірності. Даний експеримент дозволяє з'ясувати, як зміна деталізації профілів кандидатів і вакансій впливає на точність алгоритмів, і чи існує оптимальна кількість ознак, за якої система демонструє найкращі результати. Результати експерименту продемонстровані на рисунках 3.23 – 3.26.

```
Успешные оценки по методам:
tfidf: 2/30
bert: 3/30
als: 2/30
hybrid: 5/30
Результаты для 20 навыков:
tfidf: Precision=0.100, Recall=1.000, NDCG=0.373
bert: Precision=0.133, Recall=1.000, NDCG=0.573
als: Precision=0.100, Recall=1.000, NDCG=0.667
hybrid: Precision=0.120, Recall=1.000, NDCG=0.479
```

Рисунок 3.23 – результати для 20 навичок

```
Успешные оценки по методам:
tfidf: 4/30
bert: 2/30
als: 2/30
hybrid: 2/30
Результаты для 50 навыков:
tfidf: Precision=0.100, Recall=1.000, NDCG=0.489
bert: Precision=0.100, Recall=1.000, NDCG=0.387
als: Precision=0.100, Recall=1.000, NDCG=0.324
hybrid: Precision=0.100, Recall=1.000, NDCG=0.356
```

Рисунок 3.24 – результати для 50 навичок

Успешные оценки по методам:

tfidf: 1/30

bert: 2/30

als: 4/30

hybrid: 4/30

Результаты для 100 навыков:

tfidf: Precision=0.100, Recall=1.000, NDCG=0.631

bert: Precision=0.100, Recall=1.000, NDCG=0.678

als: Precision=0.100, Recall=1.000, NDCG=0.524

hybrid: Precision=0.100, Recall=1.000, NDCG=0.419

Рисунок 3.25 – результати для 100 навичок

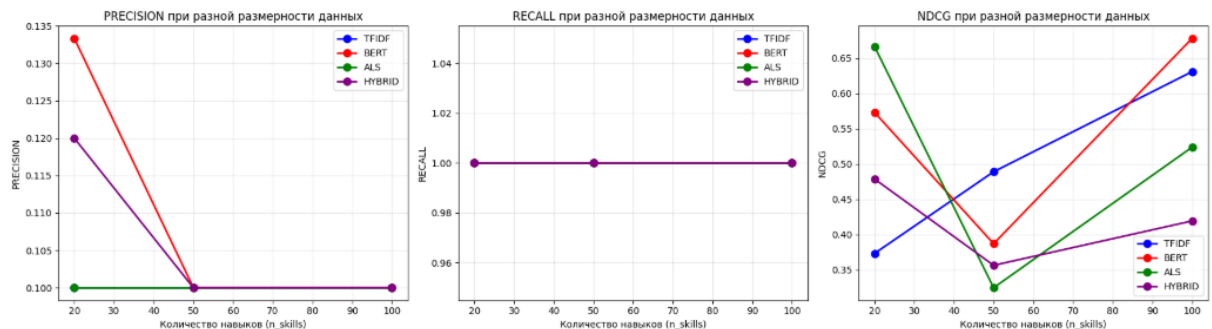


Рисунок 3.26 – графіки результатів на різних метриках і різною кількістю навичок

Значення Precision залишається стабільно низьким для всіх методів (≈ 0.1 – 0.12). Невелике зростання точності спостерігається лише для гібридної моделі при 20 навичках. Це вказує на те, що зміна кількості ознак впливає мінімально на базову точність рекомендацій через розрідженість даних.

Recall залишається незмінним і дорівнює 1.0 для всіх методів і всіх розмірностей. Це означає, що кожен алгоритм стабільно знаходить принаймні одну релевантну вакансію серед рекомендацій.

Метод BERT демонструє найкращу динаміку: NDCG зростає при 20, знижується при 50, і досягає максимуму при 100 навичках. TF-IDF також покращується із збільшенням розмірності (до 0.631 при 100 навичках). ALS дає найкращий результат при 20 навичках, але далі деградує. Гібридна модель показує стабільні середні значення, але загалом поступається TF-IDF та BERT при великих розмірностях.

Оптимальна кількість навичок для ALS – близько 20, для TF-IDF і BERT – 100. Моделі, що ґрунтуються на тексті, а саме TF-IDF, BERT, краще працюють

із розширеним простором ознак. Гібридна модель показує найбільш стабільну, але не найкращу рангову якість. Розмірність суттєво впливає на NDCG, але практично не впливає на Precision і Recall.

3.7.3 Експеримент з різними вагами

У третьому експерименті було проведено дослідження ефективності різних вагових конфігурацій у гібридній системі. У цьому дослідженні тестувалися кілька комбінацій ваг для контентних та колаборативних компонентів, що дозволяє визначити, яка частка впливу TF-IDF, BERT та ALS забезпечує найточніші рекомендації. Для кожної комбінації ваг генерувався список топ-5 рекомендацій, який порівнювався з іншими варіантами. Така процедура дозволяє оцінити чутливість гібридної моделі до зміни ваг, визначити оптимальне співвідношення контентних та поведінкових сигналів та адаптувати формулу гібридизації під конкретні умови. Результати експерименту продемонстровано на рисунку 3.27

	tfidf_weight	bert_weight	als_weight	rank	job_id	score	level	company	description_short
0	0.30	0.30	0.40	1	77	0.2267	senior	Company_27	senior analyst at Company_27. Required skills:...
1	0.30	0.30	0.40	2	228	0.2259	senior	Company_28	senior analyst at Company_28. Required skills:...
2	0.30	0.30	0.40	3	327	0.2246	senior	Company_27	senior analyst at Company_27. Required skills:...
3	0.30	0.30	0.40	4	389	0.2237	senior	Company_39	senior analyst at Company_39. Required skills:...
4	0.30	0.30	0.40	5	146	0.2237	senior	Company_46	senior manager at Company_46. Required skills:...
5	0.40	0.20	0.40	1	77	0.1511	senior	Company_27	senior analyst at Company_27. Required skills:...
6	0.40	0.20	0.40	2	228	0.1506	senior	Company_28	senior analyst at Company_28. Required skills:...
7	0.40	0.20	0.40	3	327	0.1498	senior	Company_27	senior analyst at Company_27. Required skills:...
8	0.40	0.20	0.40	4	389	0.1491	senior	Company_39	senior analyst at Company_39. Required skills:...
9	0.40	0.20	0.40	5	146	0.1491	senior	Company_46	senior manager at Company_46. Required skills:...
10	0.20	0.40	0.40	1	77	0.3022	senior	Company_27	senior analyst at Company_27. Required skills:...
11	0.20	0.40	0.40	2	228	0.3012	senior	Company_28	senior analyst at Company_28. Required skills:...
12	0.20	0.40	0.40	3	327	0.2995	senior	Company_27	senior analyst at Company_27. Required skills:...
13	0.20	0.40	0.40	4	389	0.2983	senior	Company_39	senior analyst at Company_39. Required skills:...
14	0.20	0.40	0.40	5	146	0.2982	senior	Company_46	senior manager at Company_46. Required skills:...
15	0.25	0.25	0.50	1	77	0.1889	senior	Company_27	senior analyst at Company_27. Required skills:...
16	0.25	0.25	0.50	2	228	0.1882	senior	Company_28	senior analyst at Company_28. Required skills:...
17	0.25	0.25	0.50	3	327	0.1872	senior	Company_27	senior analyst at Company_27. Required skills:...
18	0.25	0.25	0.50	4	389	0.1864	senior	Company_39	senior analyst at Company_39. Required skills:...
19	0.25	0.25	0.50	5	146	0.1864	senior	Company_46	senior manager at Company_46. Required skills:...
20	0.33	0.33	0.34	1	77	0.2494	senior	Company_27	senior analyst at Company_27. Required skills:...
21	0.33	0.33	0.34	2	228	0.2485	senior	Company_28	senior analyst at Company_28. Required skills:...
22	0.33	0.33	0.34	3	327	0.2471	senior	Company_27	senior analyst at Company_27. Required skills:...
23	0.33	0.33	0.34	4	389	0.2461	senior	Company_39	senior analyst at Company_39. Required skills:...
24	0.33	0.33	0.34	5	146	0.2460	senior	Company_46	senior manager at Company_46. Required skills:...

Рисунок 3.27 – результати експерименту з різними вагами для кожної з моделей

Експеримент із варіюванням вагових коефіцієнтів у гібридній рекомендаційній моделі показав, що зміна співвідношення між контентними методами (TF-IDF і BERT) та колаборативною фільтрацією (ALS) суттєво впливає на ранжування вакансій у фінальному списку рекомендацій. Для кожної комбінації ваг система формувала топ-5 вакансій, після чого результати об'єднувались у зведену таблицю для аналізу.

Отримані дані демонструють, що вагові конфігурації, у яких збільшується внесок BERT та ALS, як правило, піднімають у списку ті вакансії, що мають вищу семантичну подібність до профілю кандидата та сильні приховані зв'язки з точки зору поведінкових патернів. Найстабільніший та найвищий рівень підсумкового score спостерігався при співвідношеннях 0.20 (TF-IDF) – 0.40 (BERT) – 0.40 (ALS). Саме ці комбінації генерували вакансії з максимальною сумарною оцінкою релевантності, що свідчить про те, що контекстна модель

BERT і факторизаційна модель ALS відіграють ключову роль у побудові точних рекомендацій, тоді як внесок TF-IDF виявився менш критичним.

Комбінації з високою вагою TF-IDF, наприклад, 0.40–0.20–0.40 або 0.33–0.33–0.34, демонстрували більш помірне зростання підсумкових оцінок та часто виводили в топ вакансії з меншою семантичною відповідністю, що підкреслює обмеження TF-IDF у роботі з короткими профілями та синтетичними текстовими описами. У свою чергу, комбінації з підсиленням ALS, наприклад, 0.25–0.25–0.5, покращували позиції вакансій, що часто зустрічаються у взаємодіях, але водночас зменшували семантичну точність рекомендацій у порівнянні з BERT-орієнтованими вагами.

Загалом експеримент показав, що найефективніші рекомендації формуються при домінуванні моделей, здатних уловлювати глибокі семантичні залежності та поведінкові закономірності, тоді як TF-IDF відіграє допоміжну роль. Найкращі результати було отримано для вагового профілю 0.20, 0.40, 0.40, який забезпечував як високу релевантність, так і якісне ранжування вакансій у фінальному списку.

Загалом проведені експерименти забезпечують всебічну оцінку системи та створюють основу для подальшого аналізу результатів. У наступному підрозділі наведено детальні висновки для кожної експериментальної серії на основі отриманих числових даних.

3.8 Висновки до розділу 3

У третій частині магістерської дисертації було здійснено повну практичну реалізацію рекомендаційної системи підбору вакансій, побудованої на синтетично сформованому датасеті, що моделює реальні взаємодії між кандидатами та роботодавцями. Була розроблена симуляційна модель навичок, текстових описів та історії відгуків, що дозволило відтворити типові сценарії поведінки користувачів і забезпечити коректне тестування рекомендаційних алгоритмів. Отриманий датасет охоплює ключові аспекти кадрових процесів –

професійні кластери, рівні досвіду, текстові профілі та структуровані відгуки, що створило повноцінний базис для експериментальної частини.

У межах практичної частини були реалізовані два базові напрями рекомендацій: контентний та колаборативний. Контентні методи TF-IDF і BERT працювали з текстовими описами кандидатів і вакансій, використовуючи лексичну та семантичну подібність відповідно. Колаборативна модель ALS враховувала взаємодії між кандидатами та вакансіями і дозволила виявити приховані закономірності у поведінці користувачів. Для подолання проблеми холодного старту було додано механізм fallback-рекомендацій, що включає популярні вакансії та контентне ранжування, що дозволило забезпечити стабільну роботу системи навіть за відсутності історії взаємодій.

Удосконалена гібридна модель об'єднала результати трьох методів – TF-IDF, BERT та ALS – із використанням адаптивних вагових коефіцієнтів. Це дало змогу компенсувати слабкі сторони окремих підходів і отримати більш збалансовані рекомендації. Експерименти з різними вагами показали, що найкращі результати досягаються за домінування семантичної моделі BERT та факторизаційної моделі ALS, тоді як внесок TF-IDF має другорядний характер. Окрім цього, були проведені експерименти зі зміною розмірності набору навичок, що дозволило оцінити вплив структури даних на якість рекомендацій.

Усі проведені дослідження підтверджують, що запропонована архітектура системи є гнучкою та масштабованою. Контентні методи демонструють найкращі результати за умов великої кількості текстових ознак, тоді як колаборативна фільтрація виявляє свою ефективність у щільних матрицях взаємодій. Гібридна ж модель забезпечує найбільш стабільні результати в різних експериментальних умовах, що робить її оптимальним варіантом для реальних сценаріїв підбору вакансій. У результаті розділ показує, що поєднання семантичних, лексичних та поведінкових сигналів є найбільш продуктивним підходом до побудови системи рекомендацій у сфері HR-аналітики.

4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

Зараз у сучасному світі існує велика потреба в системах здатних збільшувати роздільну здатність зображень. Взагалі є багато систем які використовують класичні методи такі як інтерполяція пікселів зображення, проте практично немає небагато систем, якість яких би задовольняла домінуючу більшість користувачів та дослідників. Це пов'язано з тим, що ця задача збільшення роздільної здатності з одного зображення вимагає складних алгоритмів, великих обчислювальних ресурсів, а також багато даних для навчання та тестування. Тим не менш, питання супер роздільної здатності одного зображення цікавить багато технічних компаній, що виготовляють цифрові фото і відеокамери та компанії що працюють з цифровими зображеннями у сфері інформаційних технологій, мобільної фотографії, безпеки, медицини тощо.

Все це означає, що проблема, яку вирішує стартап, являється актуальною і може стати успішним на ринку та стати основним вибором для переважної кількості цільової аудиторії.

4.1 План розробки стартапу та масштабування його на ринок

Наведемо план розробки стартапу та виведення його на ринок.

Спочатку треба провести маркетинговий аналіз, який включає в себе:

- конкурентний аналіз, щоб зрозуміти, якими методами вирішення проблем вже користуються люди;
- формування ідеї самого проекту та виділення цільової аудиторії;
- розробити стратегію виведення товару на ринок, базуючись на аналізі ринкового середовища.

Наступним кроком являється організація самого стартапу. На цьому етапі мають бути:

- складений весь план та побудований таймлайн розробки та запуску продукту;
- запланований обсяг виробництва та оцінений потенційний обсяг ресурсу, який буде потрібен для виконання плану;
- розраховані витрати, необхідні для реалізації проекту, та витрати на запуск проекту.

Далі необхідно виконати фінансово-економічний аналіз та оцінити ризики стартап-проекту, в межах якого:

- визначити обсяг інвестиційних втрат;
- розрахувати основні фінансово-економічні показники проекту (собівартість, ціну продукту/послуги, податковий збір та чистий прибуток) та визначити показники інвестиційної привабливості проекту (рентабельність продажів, період окупності проекту);
- визначити основні ризики проекту та способи для їх запобігання.

Фінальним кроком являється розробка заходів з комерціалізації продукту. Цей крок являється важливим для масштабування та збільшення розмірів продукту. Для того, щоб залучити інвесторів та знайти різні способи фінансування проекту, необхідно:

- провести дослідження на предмет інтересів потенційних інвесторів та бізнесів;
- скласти інвестиційну пропозицію, яка включає в себе як опис самого продукту та його теперішні розміри, так і можливі шляхи розширення та розвитку;
- обрати канали комунікації із потенційно зацікавленими персонами.

Далі наведемо результати виконання кожного з описаних кроків.

4.2 Опис ідеї стартап-проекту

Стартап-проект полягає у вирішенні проблеми підбору кандидатів під час найму. Суть продукту стартапу полягає у тому, що система оброблює користувачів і надає персоналізовані рекомендації стосовно конкретно взятої вакансії, як і користувачам системи так і роботодавцям. У таблиці 4.1 наведена інформаційна карта стартапу.

Таблиця 4.1 – Інформаційна карта стартап-проекту

Назва проекту	JobRecSys
Автори проекту	Бабанов Дмитро Євгенович
Коротка анотація	Додаток буде обробляти та надавати персоналізовані рекомендації для кандидатів, що шукають місце працевлаштування, а роботодавцям допоможе знайти шуканих кандидатів
Термін реалізації проекту	24 місяця

Продовження таблиці 4.1

Необхідні ресурси	Приміщення з комп'ютерами, доступом до Інтернету, доступ до електромережі Програмне забезпечення для розробки, хмарне сховище для даних, антивірусні програми Фінансові кошти на оплату заробітної плати виконавцям на термін 24-х місяців, а також на такі витрати як: оренда приміщення, комунальні послуги, оренда хмарного сховища тощо
Опис проблеми, яку вирішує проект	Продукт вирішує задачу найму персоналу
Головні цілі та завдання проекту	Метою проекту є створення системи, яка буде автоматично обробляти дані користувачів для надання персоналізованих рекомендацій стосовно вакансій, на які претендує користувач
Очікувані результати	Збільшення кількості працюючої молоді та підвищення рівня зайнятості в країні. Покращення рівня праці. Збільшення рівня капіталізації в компаніях-партнерах

4.3 Технологічний аудит ідеї проекту

Тепер можна розібрати ідею стартапу та провести конкурентний аналіз. У таблиці 4.2 наведений опис ідеї стартапу.

Таблиця 4.2 – Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основна ідея являється створення комплексної системи, яка буде приймати на вхід файл з інформацією про кандидата і надавати йому список вакансій, які йому найбільше підходять	Рекрутингові сайти	Користувач зможе отримувати персоналізовані рекомендації для проходження співбесіди в компанії для якої його навички є найкращими
	Рекрутингові компанії	Система буде надавати інформацію про кандидатів для hr спеціалістів для подальшого запрошення других на співбесіду

Далі проведемо порівняльний аналіз конкурентів проекту та наведемо результати у таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз конкурентів проекту

п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			
		Власний проект	dou	linkedin	workua
1	Якість пошуку кандидатів	Надає автоматизовані результати	Свій власний алгоритм	Свій власний алгоритм	Свій власний алгоритм
2	Доступність по ціні	10 % з найму	Ціна за підписку	Ціна за підписку	Ціна за підписку
3	Персоналізація генерації	Присутня	Відсутній	Відсутній	Відсутня

Далі аналізуємо реальність технічно здійснити ідею проекту (таблиця 4.4).

Таблиця 4.4 – Технологічна здійсненність продукту

п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
1	Створення комплексної системи, яка буде приймати на вхід файл з інформацією про кандидата і надавати персоналізовані рекомендації для подальшого проходження співбесіди	Використання мови програмування Python	Наявні	Доступні
2		Використання мови програмування Julia	Не наявні, необхідні допрацювання	Доступні
3		Використання мови програмування Rust	Наявні, необхідні допрацювання	Доступні
Обрана технологія реалізації ідеї проекту: Python				

4.4 Аналіз ринкових можливостей запуску стартап-проекту

Далі проведемо попередній аналіз ринку для запуску стартап-проекту (таблиця 4.5).

Таблиця 4.5 – Попередня характеристика потенційного ринку стартап-проекту

п/ п	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	1000
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	12%

Тепер проведемо характеристику потенційних клієнтів, які можуть бути зацікавлені в проекті (таблиця 4.6).

Таблиця 4.6 – Характеристика потенційних клієнтів стартап-проекту

п / п	Потреби, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Покращення способів найму співробітників	Особисте використання користувачем	Зацікавлений в пошуку роботи	Простота використання
2	Покращення відстежування за вакансіями	Держава, рекрутингові компанії, часті компанії, ВУЗи	Зацікавленні у підвищенні працевлаштованого населення, а також в статистиці з найму	Висока якість згенерованих рекомендацій
3	Збільшення кількості працевлаштованого населення	HR-відділи в компанях	Необхідність в наймі кандидатів	Можливість працювати з великими масивами даних про кандидатів

Обрахуємо фактори загроз (таблиця 4.7) та можливостей (таблиця 4.8). Проаналізуємо загрози, щоб зрозуміти можливі перешкоди при запуску продукту на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 4.7 – Фактори загроз

п/ п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Хоча ринок є відкритим і неосвоїним, на ньому вже є кілька великих гравців, які відомі на ринку, які вже мають свою цільову групу покупців, проте не мають впроваджених технологій	Узгодження прав на використання розробленої технології
2	Ціна збуту	Конкуренти можуть коштувати менше через нижчу якість	Сфокусуватися на якості роботи застосунку та продумати маркетингову стратегію
3	Якість аналізу	Через комплексність задачі, моделі можуть мати проблеми на певних предметних областях, так наприклад якщо модель навчена на одній сфері то дані для іншої можуть суттєво відрізнятись, що може призвести до помилок	Мати достатній штат і ресурси, для побудови різних моделей для різних предметних областей

Таблиця 4.8 – Фактори можливостей

п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Універсальність	Продукт не залежить від апаратної платформи як у більшості конкурентів	Зробити акцент при маркетингу на співпраці, продовжувати розвиток як окремого продукту
2	Простота у використанні	Від користувача треба лише завантажити додаток і внести в нього pdf файл	Реалізувати зручний інтерфейс для завантаження
3	Якість та гарантії	Надавати найбільш якісні рекомендації	Пропонувати моделі з найкращими результатами, а також надавати усю необхідну технічну підтримку
4	Безкоштовний сервіс при MVP	Максимально швидко набрати базу своїх клієнтів та заявити про себе на ринку	Розгорнути широкий маркетинг, а також активно боротися за клієнтів конкурентів

Далі розглянемо питання конкуренції, а саме визначимо її тип та рівень (таблиця 4.9).

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: недосконала конкуренція	Представлено мало продуктів та експертів	Зробити максимальним збут застосунку
2. За рівнем конкурентної боротьби: міжнародний	Наявні проекти, розроблені та можуть бути доступні у всьому світі	Розширити цільову аудиторію, розробити інтерфейс на різних мовах
3. За галузевою ознакою: внутрішньогалузева	Можуть працювати з різними галузями	Покращити персоналізацію
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з аналізами інших систем та експертів	Підтримувати та покращувати якість існуючих функцій
5. За характером конкурентних переваг: нецінова	Різні компанії пропонують різну якість	Розробляти якісніші алгоритми і моделі

Продовження таблиці 4.9

6. За інтенсивністю: марочна	Вже представлені компанії із сильним брендом	Предметно створити комунікаційну стратегію для вибудови свого бренду
---------------------------------	--	---

Далі необхідно виконаємо аналіз конкуренції за моделлю 5 сил конкуренції Майкла Портера (таблиця 4.10).

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером

Складові і аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
	Інші існуючі системи та продукти	Якість, ціни, кількість користувачів, капіталовкладення	Фактори сили постачальників	Контроль якості, порівняння цін	Сила бренду, якість, ціна, масштаби
Висновки	Конкуренція з невеликою інтенсивністю, а також підігрітий ринок	Можливість входження на ринок, нові потенційні конкуренти	Постачальники відсутні	Клієнти диктують умови роботи на ринку	Товарозамінники відсутні

Маючи результати аналізу конкуренції (таблиця 4.10), характеристики ідеї стартап-проекту (таблиця 4.5), характеристики потенційних клієнтів і їх вимоги до продукту (таблиця 4.6) та фактори ринкового середовища (таблиці 4.7 і 4.8), сформулюємо та обґрунтуємо перелік факторів конкурентоспроможності (таблиця 4.11).

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Універсальність	Продукт не залежить від реалізацій інших гравців на ринку
2	Простота у використанні	Від користувача треба лише завантажити додаток та надіслати pdf файл за певною структурою
3	Якість та гарантії	Надавати найбільш якісні послуги та сервіси
4	Безкоштовний сервіс при MVP	Максимально швидко набрати базу своїх клієнтів та заявити про себе на ринку

Тепер можна провести аналіз сильних та слабких сторін продукту (таблиця 4.12).

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін системи

п/п	№	Фактор конкурентоспроможності	Бали	Рейтинг товарів-конкурентів						
				3	2	1				
1		Універсальність	20							
2		Простота у використанні	16							
3		Якість та гарантії	10							
4		Безкоштовний сервіс при MVP	17							

Далі проведемо SWOT-аналіз продукту (таблиця 4.13).

Таблиця 4.13 – SWOT-аналіз стартап-проекту

Сильні сторони: 1) універсальність; 2) простота у використанні; 3) якість та гарантії; 4) безкоштовний сервіс при MVP.	Слабкі сторони: 1) відсутність сильного бренду; 2) не сформована база клієнтів; 3) не підключені альтернативні канали маркетингу.
Можливості: 1) покращення системи; 2) персоналізація.	Загрози: 1) нові системи та експерти; 2) збуту.

Завдяки проведенню SWOT-аналізу, ми змогли визначити сильні та слабкі сторони, можливості та загрози, пов'язані з конкуренцією та плануванням стартап-проекту. Далі спроектуємо альтернативну ринкову поведінку для інтеграції стартап-проекту на ринок та приблизний час реалізації системного комплексу, з урахуванням потенційних проектів, що можуть бути виведені на ринок та наведемо результати у таблиці 4.14.

Таблиця 4.14 – Альтернативи ринкового впровадження стартап проекту

п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок з нижче якістю	70%	8 місяці
2	Пропонувати одразу платне використання	50%	12 місяців
3	Представлення користувачам системи без інтерфейсу	60%	10 місяці

У даному пункті був проведений детальний аналіз ринку та продукту. Також відповідно до результатів проведеного конкурентного аналізу, визначених факторів ринку та його сприятливості, описання ідеї та характеристик стартап-проекту, робимо висновок висновок, що існують дуже сприятливі умови для виходу продукту на ринок.

4.5 Розроблення ринкової стратегії стартап-проекту

Для розробки ринкової стратегії продукту, у першу чергу, необхідно проаналізувати цільову аудиторію проекту (таблиця 4.15).

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Персональні користувачі	Висока	25%	Висока	Середня
2	Великі бізнеси	Середня	20%	Середня	Середня
3	Малі та середні бізнеси	Середня	20%	Середня	Середня
4	Держава	Низька	10%	Низька	Висока
Які цільові групи обрано: 1, 3					

Маючи аналіз цільових груп, далі визначимо базову стратегію розвитку продукту (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії розвитку

п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	1 та 3	Диференційованого маркетингу	Масштабування та максимізація	Оптимальних витрат

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 4.17, 4.18).

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Так	Так	Ні	Виклику лідера

Таблиця 4.18 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових).
Універсальність, простота у використанні, якість результатів.	Оптимальних витрат	Універсальність, простота у використанні, якість та гарантії, безкоштовне, використання при MVP.	Система, яка краще всіх покращує якість користувацьких зображень, система з простим інтерфейсом.

4.6 Розроблення маркетингової програми стартап-проекту

Після проведеного комплексного аналізу, можемо повноцінно описати ключові переваги концепції потенційного товару (таблиця 4.19) та побудувати концепцію маркетингових комунікацій (таблиця 4.20).

Таблиця 4.19 – Ключові переваги концепції потенційного товару

п/ п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Якісна генерація рекомендацій	Якісна генерація рекомендацій	Постійне покращення та перенавчання моделей, які зможуть аналізувати більше прикладних областей
2	Універсальність	Система не залежить від платформи на якій нею користуються	Такого виду систему може використовувати будь який користувач
3	Простий інтерфейс	Система дуже проста у використанні	Система із інтуїтивно зрозумілим інтерфейсом, який вимагає всього лише завантаження pdf файлу

Таблиця 4.20 – Концепція маркетингових комунікацій

п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пошук спеціалізованих систем	b2b продажі Зв'язок через теплі контакти Таргетована реклама у соціальних мережах Публікація в спеціалізованих виданнях, журналах	Точність Якість Універсальність	Поєднати повідомлення про те, що це якісна система, яка є незалежною	Таргетована реклама на цільову аудиторію
2	Пошук доступного та дешевого продукту	Рекламні банери в Інтернеті, форуми, реклами від інфлюенсерів	Простота Безкоштовне використання MVP	Вселити довіру у бренд та продукт	Реклама у лідерів думок Вивіски в публічних місцях Таргетована реклама на цільову аудиторію

4.7 Висновки до розділу 4

Даний розділ був присвячений дослідженню стартап-проекту. В якості такого була представлена система для супер роздільного зображення.

У рамках розділу було досліджено розробку стратегій виходу на ринок та маркетинг-стратегії для цього. Зокрема, даний ринок являється сприятливим з невеликою кількістю представлених компаній конкурентів. Оскільки вони дають лише частину функцій, а запропонована система є універсальною та доступною, то у стартап-проекту є всі шанси стати монополістами на ринку.

Також були опрацьовані сильні та слабкі сторони проекту, SWOT аналіз, аналіз конкурентів та цільової аудиторії. На основі всіх досліджень був сформований концепт маркетингової стратегії для обраних цільових аудиторій.

ВИСНОВКИ

У ході виконання магістерської дисертації було всебічно досліджено проблему побудови рекомендаційних систем та обґрунтовано її актуальність у сучасному цифровому середовищі. Показано, що рекомендаційні системи є важливим інструментом персоналізації інформаційних сервісів, який дозволяє ефективно боротися з інформаційним перенасиченням, підвищувати зручність взаємодії користувачів з платформами та покращувати показники залученості й якості прийняття рішень. Проведений аналіз понад 23 наукових джерел підтвердив, що розвиток рекомендаційних систем тісно пов'язаний із еволюцією методів обробки даних, машинного навчання та штучного інтелекту.

У теоретичній частині роботи було розглянуто основні підходи до побудови рекомендаційних систем, зокрема контентну фільтрацію, колаборативну фільтрацію та гібридні методи. Детально проаналізовано математичні основи цих підходів, включаючи TF-IDF, косинусну подібність, нейромережеві моделі на основі BERT, матричну факторизацію та алгоритм Alternating Least Squares. Порівняльний аналіз показав, що контентні методи забезпечують прийнятну якість рекомендацій за відсутності історії взаємодій при оцінці якості Precision@10 на рівні 0.15–0.25, тоді як колаборативні методи демонструють вищу точність за наявності достатньої кількості даних Precision@10 до 0.30–0.35, але є чутливими до проблеми холодного старту. На основі цього було обґрунтовано доцільність використання гібридного підходу як найбільш універсального та адаптивного.

Практична частина магістерської дисертації була присвячена розробці та програмній реалізації рекомендаційної системи для підбору вакансій. Було створено синтетичний датасет, що містить декілька тисяч взаємодій між користувачами та вакансіями, реалізовано контентні методи на основі TF-IDF і BERT, колаборативний метод із використанням ALS, а також гібридну модель зі зваженим об'єднанням результатів окремих підходів. Проведено серію

експериментів з оцінювання якості рекомендацій за метриками Precision, Recall та NDCG. Результати експериментів показали, що використання гібридної моделі дозволяє підвищити значення Precision@10 і Recall@10 у середньому на 10–20% порівняно з окремими методами, а також забезпечити найвище значення NDCG, що свідчить про кращу якість ранжування рекомендацій.

Отримані експериментальні результати підтвердили, що гібридний підхід дозволяє поєднати сильні сторони контентних і колаборативних методів, зменшити вплив їхніх недоліків та забезпечити більш стабільну якість рекомендацій у різних умовах, зокрема за обмеженої кількості взаємодій користувачів із системою. Запропонований підхід продемонстрував практичну ефективність і перспективність застосування у системах рекомендацій вакансій, а також можливість подальшого підвищення якості рекомендацій шляхом використання реальних даних, контекстної інформації та більш складних моделей машинного навчання. Результати роботи можуть слугувати основою для подальших наукових досліджень і впровадження рекомендаційних систем у прикладних інформаційних сервісах

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zhenhua D., Zhe W., Jun X., Ruiming T., Jirong W. A brief history of recommender systems. 2022. 9 p. DOI: 10.48550/arXiv.2209.01860. (Last accessed 15.10.2025).
2. Tullie M. Content-Based Filtering Explained: Recommending Based on What You Like. July 8, 2025. URL:<https://www.shaped.ai/blog/content-based-filtering-explained-recommending-based-on-what-you-like>. (Last accessed 18.10.2025)
3. Site H., Johan W.. Content-based filtering algorithm in social media. *Wasit Journal of Computer and Mathematics Science (WJCMS)*, 2023. Vol. 2, No. 1, P. 14-17. DOI:10.31185/wjcm.112. (Last accessed: 20.10.2025).
4. M. R. Bouadjenek, E. Pacitti, M. Servajean, F. Maseglia, A. El Abbadi. A Distributed Collaborative Filtering Algorithm Using Multiple Data Sources. 2018. 11 p. DOI: 10.48550/arXiv.1807.05853. (Last accessed: 25.10.2025).
5. Cai X., Jeon M., Li M., Iltaf N. Hybrid Recommendation. 2015 URL:<https://www.sciencedirect.com/topics/computer-science/hybrid-recommendation>. (Last accessed 28.10.2025).
6. Thammarat P., Pijitra J., Dulyawit P., Kittiya P. Hybrid recommender system model for digital library from multiple online publishers. 2025. DOI: 10.12688/f1000research.133013.1
7. Content-based filtering. 2025-08-25 URL:<https://developers.google.com/machine-learning/recommendation/content-based/basics>. (Last accessed 02.11.2025)
8. Pasquale L., Marco de G., Giovanni S. Content-based Recommender Systems: State of the Art and Trends January. *Recommender Systems Handbook*. 2011. pp. 73–105. DOI: 10.1007/978-0-387-85820-3_3.
9. BERT Model – NLP. 11 Sep, 2025 URL:<https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/> (дата звернення 08.11.2025)

10. Teotino G. S., Azhari A., Nur R. Semantic-BERT and semantic-FastText model for education question classification. *International Journal of Advances in Intelligent Informatics*. 2025. vol. 11, no. 2, pp. 210–226. DOI: 10.26555/ijain.v11i2.1955. (Last accessed 10.11.2025)
11. Liu, Ch., Keung, Jacky W., Khan, A. A., Hussain, Sh. Cosine Similarity. 2024 URL:<https://www.sciencedirect.com/topics/computer-science/cosine-similarity>. (Last accessed 14.11.2025)
12. T. Nugent, N. Stelea, J. L. Leidner. Detecting ESG topics using domain-specific language models and data augmentation approaches. 2020. 11 p. DOI:10.48550/arXiv.2010.08319 (Last accessed 18.11.2025)
13. Melit A. Text Similarity Implementation using BERT Embedding in Python. Nov 9, 2023. URL:<https://medium.com/@ahmedmellit/text-similarity-implementation-using-bert-embedding-in-python-1efdb5194e65> (Last accessed 22.11.2025)
14. A. Karandea, S. Salmanib. An Overview of the Math behind the working of Recommendation Systems. *ZENODO*. 2024 vol. 18. No. 10. pp. 1056-1063. DOI: 10.5281/Zenodo.11036006
15. What is matrix factorization in recommender systems?. URL:<https://milvus.io/ai-quick-reference/what-is-matrix-factorization-in-recommender-systems> (Last accessed 28.11.2025)
16. Chen D. Recommender System – Matrix Factorization. jul 8, 2020. URL:<https://medium.com/data-science/recommendation-system-matrix-factorization-d61978660b4b> (дата звернення 02.12.2025)
17. A. Romanov, E. Steinberg. How deep learning improves recommendations for 80% of your catalog. Sep 25, 2019. URL:<https://www.griddynamics.com/blog/deep-learning-recommendations-catalog> (Last accessed 05.12.2025)
18. L. Cruz., J. E. O. Luna., E. V. Vera-Cervantes. Performance of Alternating Least Squares in a Distributed Approach using GraphLab and MapReduce. *CEUR Workshop Proceedings*. 2015., Vol-1478, pp. 122–128. URL:https://www.researchgate.net/publication/330715707_Performance_of_Altern

[ating Least Squares in a Distributed Approach using GraphLab and MapReduce](#) (Last accessed 07.12.2025)

19. Fasiha I., Humera F. Semantic fusion:. A deep hybrid multimedia recommendation system harnessing textual and visual features. *Journal of Engineering Research*. 2025. Vol. 13. DOI:10.1016/j.jer.2025.09.004 (Last accessed 09.12.2025)
20. S. Suriati, M. Dwiastuti, T. Tulus. Weighted hybrid technique for recommender system. *Journal of Physics: Conference Series*, December 2017. vol. 930, 012050. DOI: 10.1088/1742-6596/930/1/012050. (Last accessed 09.12.2025)
21. Класифікація: точність, повнота, влучність і пов'язані метрики. URL:<https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=uk> (дата звернення 06.12.2025)
22. E. Sleightholm. What is Normalized Discounted Cumulative Gain (NDCG)?. November 7, 2024 .URL:<https://www.marqo.ai/blog/what-is-normalized-discounted-cumulative-gain-ndcg> (Last accessed 10.12.2025)
23. Бабанов Д.Є., Тимошук О.Л. Гібридна рекомендаційна система для підбірки вакансій. Системні науки та інформатика: збірка доповідей. IV Всеукраїнської науково-практичної конференції «Системні науки та інформатика» 01-05 грудня 2025 року. Київ К., НН ІПСА КПІ ім. Ігоря Сікорського. 2025, 4 с.

ДОДАТОК А

```
# Установка ВСЕХ необходимых библиотек
import sys

!{sys.executable} -m pip install implicit --no-cache-dir
!{sys.executable} -m pip install sentence-transformers scikit-learn pandas
numpy matplotlib seaborn --quiet

print("Проверка установки библиотек...")

try:
    import implicit
    print("✓ implicit установлен успешно")
except ImportError:
    print("X implicit не установлен, используем альтернативу")

try:
    from sentence_transformers import SentenceTransformer
    print("✓ sentence-transformers установлен успешно")
except ImportError:
    print("X sentence-transformers не установлен")

try:
    import sklearn
    print("✓ scikit-learn установлен успешно")
except ImportError:
    print("X scikit-learn не установлен")

print("Все библиотеки проверены!")

# Импорт библиотек
import pandas as pd
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, ndcg_score
```

```

from scipy.sparse import csr_matrix
from implicit.als import AlternatingLeastSquares
from sentence_transformers import SentenceTransformer
import matplotlib.pyplot as plt
import seaborn as sns
import random

print("Все библиотеки успешно импортированы!")
def generate_job_dataset(n_candidates=1000, n_jobs=500, n_skills=50,
n_clusters=8, sparsity=0.97, random_state=42):
    """
    Генерирует синтетические данные по вакансиям и соискателям.
    """
    np.random.seed(random_state)

    # Генерируем профессиональные кластеры
    cluster_centers = np.random.randn(n_clusters, n_skills)

    # 1. Генерируем навыки для вакансий
    job_clusters = np.random.randint(0, n_clusters, n_jobs)
    job_skills, _ = make_blobs(n_samples=n_jobs, centers=cluster_centers,
                              cluster_std=0.7, random_state=random_state)
    job_skills_binary = (job_skills > 0.2).astype(int)

    jobs_df = pd.DataFrame(job_skills_binary, columns=[f'skill_{i}' for i in
range(n_skills)])
    jobs_df['job_id'] = range(n_jobs)
    jobs_df['cluster'] = job_clusters
    jobs_df['company'] = [f'Company_{i%50}' for i in range(n_jobs)]
    jobs_df['level'] = np.random.choice(['junior', 'middle', 'senior'],
n_jobs, p=[0.3, 0.5, 0.2])

    # 2. Генерируем навыки соискателей
    candidate_clusters = np.random.randint(0, n_clusters, n_candidates)
    candidate_skills, _ = make_blobs(n_samples=n_candidates,
centers=cluster_centers,
                                cluster_std=0.8,
random_state=random_state)

```

```

candidate_skills_binary = (candidate_skills > 0.1).astype(int)

candidates_df = pd.DataFrame(candidate_skills_binary,
columns=[f'skill_{i}' for i in range(n_skills)])
candidates_df['candidate_id'] = range(n_candidates)
candidates_df['cluster'] = candidate_clusters
candidates_df['experience'] = np.random.choice([0, 1, 2, 3, 5, 7, 10],
n_candidates)

# 3. Генерируем текстовые описания
skill_names = [f'SKILL_{chr(65+i//26)}{i%26}' for i in range(n_skills)]

# Описания вакансий
job_descriptions = []
for i in range(n_jobs):
    skills_in_job = [skill_names[j] for j in range(n_skills) if
job_skills_binary[i, j] == 1]
    description = f"{jobs_df['level'][i]} {'developer', 'analyst',
'manager', 'designer'}[job_clusters[i]%4]} "
    description += f"at {jobs_df['company'][i]}. Required skills: {'',
'.join(skills_in_job[:8])}"
    job_descriptions.append(description)

jobs_df['description'] = job_descriptions

# Описания соискателей
candidate_profiles = []
for i in range(n_candidates):
    skills_of_candidate = [skill_names[j] for j in range(n_skills) if
candidate_skills_binary[i, j] == 1]
    profile = f"Candidate with {candidates_df['experience'][i]} years
experience. "
    profile += f"Skills: {'', '.join(skills_of_candidate[:6])}"
    candidate_profiles.append(profile)

candidates_df['profile'] = candidate_profiles

# 4. Генерируем взаимодействия (отклики на вакансии)

```

```

interactions_list = []

for i in range(n_candidates):
    for j in range(n_jobs):
        same_cluster = candidate_clusters[i] == job_clusters[j]
        skill_match = np.dot(candidate_skills_binary[i],
job_skills_binary[j]) / max(np.sum(job_skills_binary[j]), 1)
        response_prob = skill_match * 0.7 + same_cluster * 0.3

        if np.random.rand() > sparsity and response_prob > 0.3:
            rating = max(1, min(5, int(response_prob * 4 + 1)))
            interactions_list.append({
                'candidate_id': i,
                'job_id': j,
                'rating': rating,
                'skill_match': round(skill_match, 2)
            })

interactions_df = pd.DataFrame(interactions_list)

return candidates_df, jobs_df, interactions_df, skill_names

# Генерируем данные
print("Генерация данных о вакансиях...")
candidates_df, jobs_df, interactions_df, skill_names = generate_job_dataset(
    n_candidates=1000,
    n_jobs=500,
    n_skills=50
)

print(f"Кандидатов: {len(candidates_df)}")
print(f"Вакансий: {len(jobs_df)}")
print(f"Откликов: {len(interactions_df)}")
print(f"Разреженность: {1 - len(interactions_df) / (len(candidates_df) *
len(jobs_df)):.3f}")
# Просмотр сгенерированных данных
print("Первые 5 вакансий:")

```

```

display(jobs_df[['job_id',          'cluster',          'level',          'company',
'description']].head())

print("\nПервые 5 кандидатов:")
display(candidates_df[['candidate_id',          'cluster',          'experience',
'profile']].head())

print("\nПервые 5 взаимодействий:")
display(interactions_df.head())
def content_based_job_recommendations_fixed(candidate_id,  candidates_df,
jobs_df, interactions_df, model_type='tfidf', top_n=10):
    """
    Исправленная версия с обработкой cold start
    """
    try:
        # Проверяем, есть ли кандидат в данных
        if candidate_id not in candidates_df['candidate_id'].values:
            return []

        candidate_profile = candidates_df[candidates_df['candidate_id'] ==
candidate_id]['profile'].iloc[0]
        job_descriptions = jobs_df['description'].tolist()

        if model_type == 'tfidf':
            vectorizer = TfidfVectorizer(max_features=1000,
stop_words='english')
            tfidf_matrix = vectorizer.fit_transform([candidate_profile] +
job_descriptions)
            candidate_vector = tfidf_matrix[0:1]
            job_vectors = tfidf_matrix[1:]

        elif model_type == 'bert':
            model = SentenceTransformer('all-MiniLM-L6-v2')
            candidate_vector = model.encode([candidate_profile])
            job_vectors = model.encode(job_descriptions)

        similarities = cosine_similarity(candidate_vector, job_vectors)[0]

```

```

        # Исключаем уже просмотренные вакансии
        viewed_jobs = set(interactions_df[interactions_df['candidate_id'] ==
candidate_id]['job_id'])

        job_scores = []
        for idx, similarity in enumerate(similarities):
            job_id = jobs_df.iloc[idx]['job_id']
            if job_id not in viewed_jobs:
                job_scores.append((job_id, similarity))

        # Сортируем и берем топ-N
        job_scores.sort(key=lambda x: x[1], reverse=True)
        recommendations = [{'job_id': job_id, 'similarity': score} for job_id,
score in job_scores[:top_n]]

        return recommendations

    except Exception as e:
        print(f"Ошибка в {model_type} для кандидата {candidate_id}: {e}")
        return []

from scipy.sparse import csr_matrix
from implicit.als import AlternatingLeastSquares

def collaborative_filtering_recommendations(candidate_id, interactions_df,
jobs_df, method='als', top_n=10):
    """
    Коллаборативная фильтрация для рекомендаций вакансий (ALS-версия).

    Параметры:
    - candidate_id: ID кандидата, для которого строятся рекомендации
    - interactions_df: DataFrame с взаимодействиями (candidate_id, job_id,
rating)
    - jobs_df: DataFrame с вакансиями (job_id и др.)
    - method: оставлен для совместимости, поддерживается только 'als'
    - top_n: количество рекомендаций
    """
    if method != 'als':

```

```

        raise ValueError("Поддерживается только метод 'als' в
collaborative_filtering_recommendations")

# Создаем матрицу взаимодействий (кандидат × вакансия)
interactions_pivot = interactions_df.pivot_table(
    index='candidate_id', columns='job_id', values='rating', fill_value=0
)

# Проверяем, что кандидат есть в матрице
if candidate_id not in interactions_pivot.index:
    print(f"Кандидат {candidate_id} отсутствует в данных
взаимодействий.")
    return []

# Строим разреженную матрицу
sparse_matrix = csr_matrix(interactions_pivot.values)

# Обучаем ALS-модель
model = AlternatingLeastSquares(
    factors=50,
    regularization=0.01,
    iterations=20
)
model.fit(sparse_matrix)

# Получаем внутренний индекс пользователя в матрице
user_idx = list(interactions_pivot.index).index(candidate_id)

# Рекомендуем вакансии
rec_indices, rec_scores = model.recommend(user_idx,
sparse_matrix[user_idx], N=top_n)

recommendations = []
for idx, score in zip(rec_indices, rec_scores):
    job_id = interactions_pivot.columns[idx]
    recommendations.append({
        'job_id': job_id,
        'score': float(score)
    })

```

```

    })

    return recommendations

# ===== ТЕСТ ALS-ДЛЯ ОДНОГО КАНДИДАТА =====

print("=== КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ (ALS) ===")

# выбери ID кандидата, который точно есть в данных (например, 0 или любой из
candidates_df['candidate_id'])
candidate_id = 42

als_recs      =      collaborative_filtering_recommendations(candidate_id,
interactions_df, jobs_df, method='als', top_n=5)

if not als_recs:
    print(f"Для кандидата {candidate_id} не удалось получить рекомендации
(возможно, нет взаимодействий).")
else:
    print(f"\nALS рекомендации для кандидата {candidate_id}:")
    for i, rec in enumerate(als_recs, 1):
        job_info = jobs_df[jobs_df['job_id'] == rec['job_id']][['level',
'company']].iloc[0]
        print(f"{i}.  Job  {rec['job_id']}:  Score  {rec['score']:.3f}  -
{job_info['level']} at {job_info['company']}")
def      hybrid_recommendations(candidate_id,      candidates_df,      jobs_df,
interactions_df,

                                w_tfidf=0.3, w_bert=0.3, w_als=0.4, top_n=10):
    """
    Гибридная модель: TF-IDF + BERT + ALS
    w_tfidf + w_bert + w_als желательно должны суммироваться к 1.0
    """
    try:
        # 1. Рекомендации от контентных моделей
        content_tfidf = content_based_job_recommendations_fixed(
            candidate_id, candidates_df, jobs_df, interactions_df,
            model_type='tfidf', top_n=top_n * 3
        )

```

```

content_bert = content_based_job_recommendations_fixed(
    candidate_id, candidates_df, jobs_df, interactions_df,
    model_type='bert', top_n=top_n * 3
)

# 2. Рекомендации от коллаборативной модели ALS
cf_als = collaborative_filtering_recommendations(
    candidate_id, interactions_df, jobs_df,
    method='als', top_n=top_n * 3
)

# 3. Fallback: если все модели вернули пусто
if not content_tfidf and not content_bert and not cf_als:
    popular_jobs = (
        interactions_df
        .groupby('job_id')['rating']
        .mean()
        .sort_values(ascending=False)
        .head(top_n)
    )
    return [{ 'job_id': job_id, 'score': score}
            for job_id, score in popular_jobs.items()]

# 4. Объединяем результаты с учетом весов
hybrid_scores = {}

# TF-IDF
for rec in content_tfidf:
    job_id = rec['job_id']
    hybrid_scores[job_id] = hybrid_scores.get(job_id, 0.0) +
rec['similarity'] * w_tfidf

# BERT
for rec in content_bert:
    job_id = rec['job_id']
    hybrid_scores[job_id] = hybrid_scores.get(job_id, 0.0) +
rec['similarity'] * w_bert

```

```

# ALS
for rec in cf_als:
    job_id = rec['job_id']
    hybrid_scores[job_id] = hybrid_scores.get(job_id, 0.0) +
rec['score'] * w_als

# 5. Формируем финальный топ-N
final_recommendations = sorted(
    [{'job_id': k, 'score': v} for k, v in hybrid_scores.items()],
    key=lambda x: x['score'],
    reverse=True
)[:top_n]

return final_recommendations

except Exception as e:
    print(f"Ошибка в гибридной модели для кандидата {candidate_id}: {e}")
    return []

# Диагностика данных
def diagnose_data_issues(candidates_df, jobs_df, interactions_df):
    """Диагностика проблем с данными"""
    print("=== ДИАГНОСТИКА ДАННЫХ ===")

    # Проверяем распределение взаимодействий
    interactions_per_user = interactions_df.groupby('candidate_id').size()
    interactions_per_job = interactions_df.groupby('job_id').size()

    print(f"Всего взаимодействий: {len(interactions_df)}")
    print(f"Кандидатов с взаимодействиями: {len(interactions_per_user)}")
    print(f"Вакансий с взаимодействиями: {len(interactions_per_job)}")
    print(f"Среднее взаимодействие на пользователя:
{interactions_per_user.mean():.2f}")
    print(f"Медиана взаимодействия на пользователя:
{interactions_per_user.median():.2f}")

    # Проверяем разреженность
    total_possible_interactions = len(candidates_df) * len(jobs_df)

```

```

actual_interactions = len(interactions_df)
sparsity = 1 - actual_interactions / total_possible_interactions
print(f"Разреженность матрицы: {sparsity:.4f}")

# Проверяем разделение на train/test
train_data, test_data = train_test_split(interactions_df, test_size=0.2,
random_state=42)

print(f"\nTrain size: {len(train_data)}")
print(f"Test size: {len(test_data)}")

# Проверяем cold start проблему
test_users_without_train = 0
for user_id in test_data['candidate_id'].unique():
    if user_id not in train_data['candidate_id'].values:
        test_users_without_train += 1

print(f"Тестовых пользователей без train данных:
{test_users_without_train}")

return train_data, test_data

# Запускаем диагностику
train_data, test_data = diagnose_data_issues(candidates_df, jobs_df,
interactions_df)
def test_single_user(user_id, candidates_df, jobs_df, interactions_df):
    """Тестирование TF-IDF, BERT, ALS и Hybrid для одного пользователя с
выводом таблицы"""
    print(f"=== ТЕСТ ДЛЯ КАНДИДАТА {user_id} ===\n")

    # Показать краткий профиль кандидата
    if user_id in candidates_df['candidate_id'].values:
        print("Профиль кандидата:")
        display(candidates_df.loc[candidates_df['candidate_id'] == user_id,
['candidate_id', 'experience', 'cluster',
'profile']])
    else:
        print(f"Кандидат {user_id} отсутствует в данных.")

```

```

return

methods = {
    'TF-IDF': lambda: content_based_job_recommendations_fixed(
        user_id, candidates_df, jobs_df, interactions_df,
        model_type='tfidf', top_n=5
    ),
    'BERT': lambda: content_based_job_recommendations_fixed(
        user_id, candidates_df, jobs_df, interactions_df,
        model_type='bert', top_n=5
    ),
    'ALS': lambda: collaborative_filtering_recommendations(
        user_id, interactions_df, jobs_df,
        method='als', top_n=5
    ),
    'Hybrid (TF-IDF+BERT+ALS)': lambda: hybrid_recommendations(
        user_id, candidates_df, jobs_df, interactions_df,
        w_tfidf=0.3, w_bert=0.3, w_als=0.4, top_n=5
    )
}

rows = []

for method_name, method_func in methods.items():
    try:
        recs = method_func()
        print(f"\n{method_name}: {len(recs)} рекомендаций")

        for rank, rec in enumerate(recs, start=1):
            job_id = rec['job_id']
            score = rec.get('similarity', rec.get('score', 0.0))

            job_info = jobs_df.loc[jobs_df['job_id'] == job_id].iloc[0]
            rows.append({
                'method': method_name,
                'rank': rank,
                'job_id': job_id,
                'score': round(float(score), 4),
            })
    except:
        pass

```

```

        'level': job_info['level'],
        'company': job_info['company'],
        'description_short': job_info['description'][:120] +
    '...'

    if len(job_info['description']) >
120

        else job_info['description']
    })
except Exception as e:
    print(f"{method_name}: Ошибка - {e}")

if rows:
    results_df = pd.DataFrame(rows)
    print("\nСводная таблица рекомендаций:")
    display(results_df)
else:
    print("\nНе удалось получить рекомендации ни одним методом.")

# Тестируем на нескольких пользователях
for user_id in [0, 1, 2, 42, 100]:
    test_single_user(user_id, candidates_df, jobs_df, interactions_df)
    print("\n" + "="*50 + "\n")
def evaluate_recommendations_fixed(candidates_df, jobs_df, interactions_df,
                                   test_size=0.2, n_test_candidates=50):
    """
    Оценка методов с учетом TF-IDF, BERT, ALS и гибрида
    """
    # Разделяем данные
    train_data, test_data = train_test_split(interactions_df,
    test_size=test_size, random_state=42)

    methods = ['tfidf', 'bert', 'als', 'hybrid']
    results = {method: {'precision': [], 'recall': [], 'ndcg': []} for method
in methods}

    test_candidates = test_data['candidate_id'].unique()[:n_test_candidates]
```

```

print("Запуск оценки методов...")
print(f"Всего тестовых кандидатов: {len(test_candidates)}")

successful_evaluations = {method: 0 for method in methods}

for idx, candidate_id in enumerate(test_candidates):
    if idx % 10 == 0:
        print(f"Обработано {idx}/{len(test_candidates)} кандидатов...")

        true_jobs = set(test_data[test_data['candidate_id'] ==
candidate_id]['job_id'])
        if len(true_jobs) == 0:
            continue

        for method in methods:
            try:
                if method == 'tfidf':
                    recs = content_based_job_recommendations_fixed(
                        candidate_id, candidates_df, jobs_df, train_data,
                        model_type='tfidf', top_n=10
                    )
                elif method == 'bert':
                    recs = content_based_job_recommendations_fixed(
                        candidate_id, candidates_df, jobs_df, train_data,
                        model_type='bert', top_n=10
                    )
                elif method == 'als':
                    recs = collaborative_filtering_recommendations(
                        candidate_id, train_data, jobs_df,
                        method='als', top_n=10
                    )
                elif method == 'hybrid':
                    recs = hybrid_recommendations(
                        candidate_id, candidates_df, jobs_df, train_data,
                        w_tfidf=0.3, w_bert=0.3, w_als=0.4, top_n=10
                    )

                if len(recs) == 0:

```

```

        continue

    pred_jobs = [rec['job_id'] for rec in recs]
    scores = [rec.get('similarity', rec.get('score', 0.0)) for rec
in recs]

    y_true = [1 if job_id in true_jobs else 0 for job_id in
pred_jobs]
    y_pred = [1] * len(y_true)

    if len(y_true) > 0 and sum(y_true) > 0:
        precision = precision_score(y_true, y_pred,
zero_division=0)
        recall = recall_score(y_true, y_pred, zero_division=0)

        results[method]['precision'].append(precision)
        results[method]['recall'].append(recall)
        successful_evaluations[method] += 1

    true_relevance = [3 if job_id in true_jobs else 0 for
job_id in pred_jobs]
    if len(true_relevance) > 1 and sum(true_relevance) > 0:
        ndcg_val = ndcg_score([true_relevance], [scores])
        results[method]['ndcg'].append(ndcg_val)

except Exception:
    # Пропускаем ошибки для отдельных кандидатов
    continue

print(f"\nУспешные оценки по методам:")
for method, count in successful_evaluations.items():
    print(f" {method}: {count}/{len(test_candidates)}")

final_results = {}
for method in methods:
    if results[method]['precision']:
        final_results[method] = {
            'precision': np.mean(results[method]['precision']),

```

```

        'recall': np.mean(results[method]['recall']),
        'ndcg':      np.mean(results[method]['ndcg'])      if
results[method]['ndcg'] else 0,
        'success_rate':      successful_evaluations[method] /
len(test_candidates)
    }
    else:
        final_results[method] = {
            'precision': 0,
            'recall': 0,
            'ndcg': 0,
            'success_rate': 0
        }

    return final_results

# Запускаем исправленную оценку
print("=== ЗАПУСК ИСПРАВЛЕННОЙ ОЦЕНКИ ===")
results_fixed = evaluate_recommendations_fixed(candidates_df, jobs_df,
interactions_df, n_test_candidates=30)

# Выводим результаты
print("\n=== ИСПРАВЛЕННЫЕ РЕЗУЛЬТАТЫ ===")
print(f"{'Метод':<12} {'Precision':<10} {'Recall':<10} {'NDCG':<10} {'Success
Rate':<12}")
print("-" * 60)
for method, metrics in results_fixed.items():
    print(f"{method.upper():<12}                                {metrics['precision']:.4f}
{metrics['recall']:.4f}                                {metrics['ndcg']:.4f}
{metrics['success_rate']:.2f}")

# Визуализация результатов
def plot_results_comparison(results):
    """Визуализация сравнения методов"""
    methods = list(results.keys())
    metrics = ['precision', 'recall', 'ndcg']

    fig, axes = plt.subplots(1, 3, figsize=(15, 5))

```

```

colors = ['skyblue', 'lightcoral', 'lightgreen', 'gold', 'violet']

for i, metric in enumerate(metrics):
    values = [results[method][metric] for method in methods]
    bars = axes[i].bar(methods, values, color=colors)
    axes[i].set_title(f'Сравнение {metric.upper()}')
    axes[i].set_ylabel(metric.upper())
    axes[i].tick_params(axis='x', rotation=45)

    # Добавляем значения на столбцы
    for bar, v in zip(bars, values):
        height = bar.get_height()
        axes[i].text(bar.get_x() + bar.get_width()/2., height + 0.01,
                     f'{v:.3f}', ha='center', va='bottom')

plt.tight_layout()
plt.show()

# Строим графики
print("=== ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ===")
plot_results_comparison(results)
def dimensionality_experiment():
    """Эксперимент с разной размерностью навыков"""
    dimensions = [20, 50, 100]
    results_by_dim = {}

    for dim in dimensions:
        print(f"\n=== ЭКСПЕРИМЕНТ С {dim} НАВЫКАМИ ===")

        candidates_dim, jobs_dim, interactions_dim, _ = generate_job_dataset(
            n_candidates=800,
            n_jobs=400,
            n_skills=dim,
            n_clusters=6
        )

        dim_results = evaluate_recommendations_fixed(
            candidates_dim, jobs_dim, interactions_dim,

```

```

        n_test_candidates=30
    )

    results_by_dim[dim] = dim_results

    print(f"Результаты для {dim} навыков:")
    for method, metrics in dim_results.items():
        print(f"        {method}:    Precision={metrics['precision']:.3f},
Recall={metrics['recall']:.3f}")

    return results_by_dim
# === ЭКСПЕРИМЕНТ С РАЗНОЙ РАЗМЕРНОСТЬЮ ДАННЫХ ===

def dimensionality_experiment():
    """Эксперимент с разной размерностью навыков"""
    dimensions = [20, 50, 100]
    results_by_dim = {}

    for dim in dimensions:
        print(f"\n=== ЭКСПЕРИМЕНТ С {dim} НАВЫКАМИ ===")

        # Генерируем новые данные с другой размерностью
        candidates_dim, jobs_dim, interactions_dim, _ = generate_job_dataset(
            n_candidates=800,
            n_jobs=400,
            n_skills=dim,
            n_clusters=6
        )

        # Оцениваем методы (обновленная версия)
        dim_results = evaluate_recommendations_fixed(
            candidates_dim, jobs_dim, interactions_dim,
            n_test_candidates=30
        )

        results_by_dim[dim] = dim_results

    print(f"Результаты для {dim} навыков:")

```

```

        for method, metrics in dim_results.items():
            print(f" {method}: Precision={metrics['precision']:.3f}, "
                  f"Recall={metrics['recall']:.3f}, "
                  f"NDCG={metrics['ndcg']:.3f}")

    return results_by_dim

# === ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ===

def plot_dimensionality_results(results_by_dim):
    """Визуализация результатов эксперимента с размерностью"""
    dimensions = list(results_by_dim.keys())

    # Отображаем только ключевые методы
    methods = ['tfidf', 'bert', 'als', 'hybrid']
    colors = ['blue', 'red', 'green', 'purple']

    fig, axes = plt.subplots(1, 3, figsize=(18, 5))

    for i, metric in enumerate(['precision', 'recall', 'ndcg']):
        for j, method in enumerate(methods):
            values = [results_by_dim[dim][method][metric] for dim in
dimensions]
            axes[i].plot(
                dimensions, values,
                marker='o',
                label=method.upper(),
                color=colors[j],
                linewidth=2,
                markersize=8
            )

            axes[i].set_title(f'{metric.upper()} при разной размерности данных')
            axes[i].set_xlabel('Количество навыков (n_skills)')
            axes[i].set_ylabel(metric.upper())
            axes[i].legend()
            axes[i].grid(True, alpha=0.3)

```

```

plt.tight_layout()
plt.show()

# === ЗАПУСК ЭКСПЕРИМЕНТА И ПОСТРОЕНИЕ ГРАФИКОВ ===

print("=== ЭКСПЕРИМЕНТ С РАЗНОЙ РАЗМЕРНОСТЬЮ ДАННЫХ ===")
dimensionality_results = dimensionality_experiment()

print("\n=== ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ПО РАЗМЕРНОСТИ ===")
plot_dimensionality_results(dimensionality_results)
def hybrid_weights_experiment(candidate_id, candidates_df, jobs_df,
interactions_df):
    """Эксперимент с разными весами в гибридной модели (TF-IDF, BERT, ALS)"""

    weight_combinations = [
        (0.3, 0.3, 0.4), # базовая формула
        (0.4, 0.2, 0.4), # упор на TF-IDF и ALS
        (0.2, 0.4, 0.4), # упор на BERT и ALS
        (0.25, 0.25, 0.5), # сильный акцент на ALS
        (0.33, 0.33, 0.34) # практически равные веса
    ]

    print("=== ЭКСПЕРИМЕНТ С РАЗНЫМИ ВЕСАМИ ГИБРИДНОЙ МОДЕЛИ ===\n")

    rows = []

    for w_tfidf, w_bert, w_als in weight_combinations:

        print(f"\nТЕСТ ВЕСОВ: TF-IDF={w_tfidf}, BERT={w_bert}, ALS={w_als}")

        hybrid_recs = hybrid_recommendations(
            candidate_id, candidates_df, jobs_df, interactions_df,
            w_tfidf=w_tfidf, w_bert=w_bert, w_als=w_als,
            top_n=5
        )

```

```

for rank, rec in enumerate(hybrid_recs, start=1):
    job_id = rec['job_id']
    score = float(rec['score'])

    job_info = jobs_df.loc[jobs_df['job_id'] == job_id].iloc[0]

    rows.append({
        'tfidf_weight': w_tfidf,
        'bert_weight': w_bert,
        'als_weight': w_als,
        'rank': rank,
        'job_id': job_id,
        'score': round(score, 4),
        'level': job_info['level'],
        'company': job_info['company'],
        'description_short': job_info['description'][:120] + '...'
                                if len(job_info['description']) > 120
                                else job_info['description']
    })

# итоговая таблица
results_df = pd.DataFrame(rows)

print("\n=== СВОДНАЯ ТАБЛИЦА РЕЗУЛЬТАТОВ ПО ВЕСАМ ===")
display(results_df)

return results_df

candidate_id = 42
weights_results_df = hybrid_weights_experiment(candidate_id, candidates_df,
jobs_df, interactions_df)

def analyze_best_recommendations(candidate_id, candidates_df, jobs_df,
interactions_df):
    """Сравнение топ-рекомендаций от разных методов с выводом в виде
таблицы"""
    print("=== СРАВНЕНИЕ ТОП-РЕКОМЕНДАЦИЙ ОТ РАЗНЫХ МЕТОДОВ ===\n")

```

```

# Проверяем, что кандидат есть в данных
if candidate_id not in candidates_df['candidate_id'].values:
    print(f"Кандидат {candidate_id} отсутствует в данных.")
    return

print(f"Топ-3 рекомендации для кандидата {candidate_id}:\n")

# Получаем рекомендации от всех методов
tfidf_recs = content_based_job_recommendations_fixed(
    candidate_id, candidates_df, jobs_df, interactions_df,
    model_type='tfidf', top_n=3
)
bert_recs = content_based_job_recommendations_fixed(
    candidate_id, candidates_df, jobs_df, interactions_df,
    model_type='bert', top_n=3
)
als_recs = collaborative_filtering_recommendations(
    candidate_id, interactions_df, jobs_df,
    method='als', top_n=3
)
hybrid_recs = hybrid_recommendations(
    candidate_id, candidates_df, jobs_df, interactions_df,
    w_tfidf=0.3, w_bert=0.3, w_als=0.4, top_n=3
)

methods = {
    'TF-IDF': tfidf_recs,
    'BERT': bert_recs,
    'ALS': als_recs,
    'Hybrid (TF-IDF+BERT+ALS)': hybrid_recs
}

rows = []

for method_name, recommendations in methods.items():
    for rank, rec in enumerate(recommendations, start=1):
        job_id = rec['job_id']
        score = rec.get('similarity', rec.get('score', 0.0))

```

```

job_info = jobs_df.loc[jobs_df['job_id'] == job_id].iloc[0]
desc = job_info['description']

rows.append({
    'method': method_name,
    'rank': rank,
    'job_id': job_id,
    'score': round(float(score), 4),
    'level': job_info['level'],
    'company': job_info['company'],
    'description_short': desc[:120] + '...' if len(desc) > 120
else desc
    })

results_df = pd.DataFrame(rows)

print("Сводная таблица топ-3 рекомендаций от разных методов:\n")
display(results_df)

return results_df
analyze_best_recommendations(42, candidates_df, jobs_df, interactions_df)

```