Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів векторного пошуку при побудові персоналізованого переліку товарів в рекомендаційних системах
(тема)

Виконала:
студентка _2_ курсу, групи _ІУСТзм-22-1__
Зобова Аліса Марківна
(прізвище, ім'я, по батькові)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Інформаційні управляючі системи та технології
(повна назва освітньої програми)
Керівник Сергій ЧАЛИЙ
(Власне ім'я ПРІЗВИЩЕ)

Допускається до захисту

Зав. кафедри _____ Костянтин ПЕТРОВ
(підпис) (Власне ім'я ПРІЗВИЩЕ)

2024 р.

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні управляючі системи та технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_04_» __грудня__ 20 23_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Зобовій Алісі Марківні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів векторного пошуку при побудові персоналізованого переліку товарів в рекомендаційних системах
затверджена наказом університету від _01_ __грудня__2023_ р. № _259Стз_
2. Термін подання студентом роботи до екзаменаційної комісії _20___ __01__ 2024 р.
3. Вихідні дані до роботи наукова література, публікації та інтернет-джерела з досліджуваної проблеми, матеріали переддипломної практики.

4. Перелік питань, що потрібно опрацювати в роботі: аналіз рекомендаційних систем, аналіз методів до побудови рекомендацій та пошуку товарів в системах електронної комерції, дослідження методів створення векторних вбудовувань, постановка задачі дослідження, обґрунтування вибору моделі для створення векторних вбудовувань, обґрунтування вибору векторної бази даних, удосконалення методу створення персоналізованого переліку товарів на основі векторного пошуку, опис удосконаленого методу, імплементація модулю векторного пошуку, експериментальна перевірка отриманих теоретичних результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	04.12.2024	виконано
2	Аналіз рекомендаційних систем	04.12 – 6.12	виконано
3	Аналіз методів до побудови рекомендацій та пошуку товарів в системах електронної комерції	06.12 – 8.12	виконано
4	Дослідження методів створення векторних вбудовувань	08.12 – 10.12	виконано
5	Постановка задачі дослідження	11.12 – 12.12	виконано
6	Обґрунтування вибору моделі для створення векторних вбудовувань	13.12 – 14.12	виконано
7	Обґрунтування вибору векторної бази даних	14.12 – 15.12	виконано
8	Удосконалення методу створення персоналізованого переліку товарів на основі векторного пошуку	15.12 – 25.12	виконано
9	Опис удосконаленого методу	26.12 – 28.12	виконано
10	Імплементація модулю векторного пошуку	28.12 – 04.01	виконано
11	Експериментальна перевірка отриманих теоретичних результатів	04.01 – 10.01	виконано
12	Оформлення пояснювальної записки до кваліфікаційної роботи	10.01 – 16.01	виконано
13	Захист роботи	22.01.2024	виконано

Дата видачі завдання 04 грудня 2023 р.

Студент _____ Аліса ЗОБОВА
(підпис) (власне ім’я, прізвище)

Керівник роботи _____ д.т.н., проф. каф. ІУС Сергій ЧАЛИЙ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи містить: 105 с., 4 розділи, 23 рис., 3 табл., 1 додаток, 30 джерел.

ВЕКТОРНИЙ ПОШУК, ПЕРСОНАЛІЗАЦІЯ ПЕРЕЛІКУ ТОВАРІВ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, ТЕКСТОВІ ВБУДОВУВАННЯ.

У роботі виконано огляд методів векторного пошуку при побудові персоналізованого переліку товарів в рекомендаційних системах. Проаналізовано існуючі методи створення рекомендацій. На підставі проведеного аналізу запропоновано покращений метод векторного пошуку з використанням попередньої обробки вхідного тексту.

В ході дослідження отримані такі результати: визначені існуючі методи створення рекомендацій; проведено їхній порівняльний аналіз; визначені існуючі методи векторного пошуку; проаналізовані існуючі моделі для створення текстових вбудовувань та векторні бази даних; розроблено опис удосконаленого методу; проведена експериментальна перевірка по удосконаленому методу.

ABSTRACT

Explanatory note to the master's qualification work contains: 105 p., 4 chapters, 23 figures, 3 tables, 1 appendix, 30 sources.

PERSONALIZATION OF THE LIST OF PRODUCTS, RECOMMENDER SYSTEMS, TEXT EMBEDDINGS, VECTOR SEARCH.

The paper provides an overview of vector search methods for building a personalized list of products in recommender systems. The existing methods for creating recommendations are analyzed. Based on the analysis, an improved vector search method is proposed using the Elasticsearch vector database and OpenAI Embeddings text embeddings.

The study obtained the following results: existing methods for creating recommendations were identified; their comparative analysis was carried out; existing methods for vector search were identified; existing models for creating text embeddings and vector databases were analyzed; a description of the improved method was developed; and an experimental test of the improved method was carried out.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ВП – векторний пошук

РС – рекомендаційна система

ШІ – штучний інтелект

МН – машинне навчання

БД – база даних

ТЕ – text embeddings

ВСТУП

Сфера інтернет-торгівлі постійно зростає в обсягах та складності. Мільйони товарів доступні споживачам з усього світу, і знайти саме той, що відповідає їхнім потребам та уподобанням, може стати надто важкою задачею.

Звичайний текстовий пошук, яким користуються більшість інтернет-магазинів, часто обмежений тим, що він спирається на ключові слова та не завжди здатний забезпечити користувачів точними та релевантними результатами. Однак персоналізований підхід до рекомендацій може допомогти вирішити цю проблему.

Сучасні методи побудови рекомендацій орієнтовані на використання структурованих вхідних даних, що задають характеристики товарів. Однак, в більшості випадків інформація про товари, які використовує рекомендаційна система (РС), надаються у вигляді текстового опису. Перетворення такого текстового опису в структурований вигляд, який може бути збережений в базі даних (БД), вимагає значних витрат часу спеціалістів в даній предметній галузі. Тому удосконалення та подальше використання семантичних методів пошуку підходящих товарів в рекомендаційній системі є актуальною задачею.

Дослідження методів ВП в РС має велике практичне значення для покращення якості обслуговування користувачів в інтернет-магазинах та інших онлайн-платформах. Відомості, які будуть набуті в результаті цього дослідження, можуть стати основою для подальших покращень в сфері РС і сприяти покращенню користувацького досвіду від використання інтернет-сервісів.

Результатом виконання дослідження є вдосконалений метод створення персоналізованого переліку товарів з використанням попередньої обробки вхідного тексту, який сприятиме підвищенню ефективності рекомендаційних систем для користувачів у інтернет-магазинах та інших онлайн-платформах при цьому скоротивши витрати на створення векторних вбудовувань.

Отримані в ході магістерської кваліфікаційної роботи результати представлені у формі пояснювальної записки.

1 ДОСЛІДЖЕННЯ МЕТОДІВ ПОБУДОВИ ПЕРСОНАЛІЗОВАНОГО ПЕРЕЛІКУ ТОВАРІВ В РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ

1.1 Аналіз рекомендаційних систем

Зі збільшенням кількості покупок в Інтернеті зросла потреба в тому, щоб мати можливість знаходити релевантні товари з-поміж величезного асортименту. Для цього будуються РС.

РС використовують спеціалізовані алгоритми та рішення МН. Завдяки автоматизованій конфігурації, координації та управлінню алгоритмами прогнозної аналітики МН, РС може максимально точно вибрати, які фільтри застосувати до конкретної ситуації конкретного користувача. Це допомагає маркетологам максимізувати конверсії та середню вартість замовлення.

Загальна структура рекомендаційної системи наведена на рисунку 1.1.

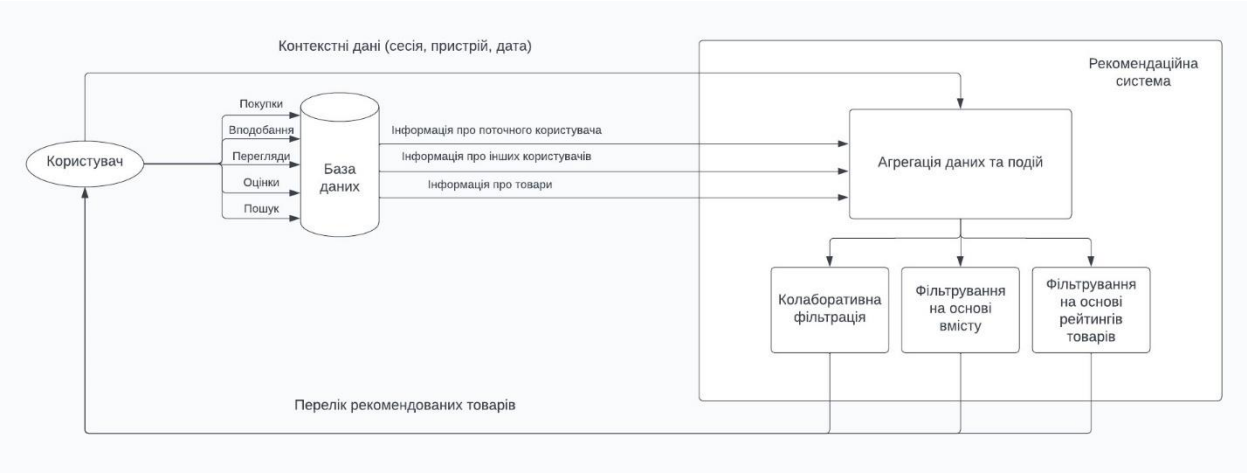


Рисунок 1.1 – Структура рекомендаційної системи

РС можуть прогнозувати оцінки користувачів навіть до того, як вони їх нададуть, що робить їх ефективним інструментом. В основному РС обробляє дані в чотири етапи.

1. Збір даних. На цьому етапі рекомендаційна система збирає різноманітні дані, пов'язані із здійсненням користувачами дій на платформі. Це може включати перегляди товарів, покупки, відгуки, оцінки інших користувачів та інші дії. Збір даних може також враховувати контекстні параметри, такі як час і місцезнаходження.

2. Зберігання даних. Отримані дані потрібно зберігати так, щоб вони були легко доступні для подальшого аналізу та обробки. Зазвичай використовуються бази даних, які дозволяють ефективно зберігати та організовувати великі обсяги інформації.

3. Аналіз. На етапі аналізу рекомендаційна система використовує різноманітні алгоритми та методи для вивчення патернів у зібраних даних. Можливі аспекти аналізу включають виявлення схожих користувачів чи товарів, визначення популярних категорій та інші характеристики, які можуть бути використані для створення ефективних рекомендацій.

4. Фільтрування. На останньому етапі застосовуються різні фільтри та моделі для визначення, які товари чи послуги найкраще підходять конкретному користувачеві.

1.2 Аналіз методів побудови рекомендацій та пошуку товарів в системах електронної комерції

Якщо розглядати традиційні методи рекомендацій та пошуку, то найбільш часто використовують аналіз текстового опису товарів та використання ключових слів для побудови рекомендацій. Такий підхід спирається на аналіз метаданих та текстового опису товарів. Він може бути ефективним для визначення загальної тематики товару та збігу ключових слів, але зазвичай недостатньо точний для персоналізованих рекомендацій.

Іншим традиційним методом є колаборативна фільтрація, яка аналізує взаємозв'язки між користувачами та товарами на підставі їхніх попередніх взаємодій, таких як покупки чи оцінки. Наприклад, якщо користувач А виражає зацікавленість у яблуках, бананах і манго, а користувач Б виражає зацікавленість у яблуках, бананах і грейпфрутах, то їхні інтереси вважаються схожими. Таким чином, ймовірно, що користувач А може зацікавитися грейпфрутом, а користувач Б – манго.

Перевагою цієї системи рекомендацій є здатність точно рекомендувати складні елементи без прямого розуміння самого об’єкта, немає потреби в аналізі контенту. Однак значним недоліком є потреба у великих обсягах даних, і цей підхід може бути обмеженим у випадках, коли недостатньо інформації про користувачів.

Також існують методи фільтрації на основі вмісту, які використовують опис продукту та профілі вибору користувача. У цій рекомендаційній системі продукти описуються за допомогою ключових слів, і профіль користувача створюється для визначення його вподобань щодо товарів.

Наприклад, якщо користувач зацікавлений у фільмах, таких як "Залізна людина", система рекомендацій може запропонувати інші фільми в жанрі супергероїв або ті, які мають схожий опис головного героя, такого як Тоні Старк.

Основною ідеєю фільтрації на основі вмісту є припущення, що якщо вам подобається певний елемент, вам ймовірно також сподобається елемент, який має схожі характеристики чи опис.

Деякі системи електронної комерції використовують методи, що базуються на рейтингах, які надають користувачі товарам. Ці підходи можуть бути ефективними для створення рекомендацій, проте вони потребують значної кількості рейтингів та можуть стикатися з "проблемою холодного старту" при введенні нових товарів.

Для підвищення ефективності рекомендацій використовують гібридні системи, які комбінують фільтрацію на основі вмісту і колаборативну фільтрацію. Ці системи надають користувачам більш різноманітний та точний перелік товарів, що робить їх перспективнішими порівняно з іншими системами рекомендацій.

Чудовим прикладом такої гібридної системи є Netflix, який використовує як фільтрацію на основі вмісту, так і колаборативну фільтрацію. Netflix аналізує звички перегляду та пошуку користувачів, знаходячи схожих користувачів на платформі. Використовуючи колаборативну фільтрацію, вони рекомендують шоу та фільми, які мають схожі риси з тими, що отримали високі оцінки. Також, застосовуючи фільтрацію на основі вмісту, Netflix враховує схожість характеристик для рекомендацій, запобігаючи загальним проблемам у системах рекомендацій, таким як проблеми холодного запуску і недостатність даних. [9]

У таблиці 1.1 наведено порівняння перерахованих вище методів.

Таблиця 1.1 — Методи побудови рекомендацій

Метод	Опис	Переваги	Недоліки
1	2	3	4
Метод пошуку за ключовими словами по текстовому опису товарів	Аналіз текстового опису товарів та використання ключових слів для рекомендацій. Спирається на аналіз метаданих та текстового опису товарів.	Ефективний для визначення загальної тематики та збігу ключових слів.	Недостатньо точний для персоналізованих рекомендацій
Метод колаборативної фільтрації	Аналізує зв'язки між користувачами та товарами на основі попередніх взаємодій.	Точно рекомендує складні елементи. Не потребує аналізу самого об'єкта.	Вимагає великої кількості структурованих даних. Обмежений у випадках з недостатньою інформацією про користувачів.

Кінець таблиці 1.1

1	2	3	4
Метод фільтрування на основі вмісту	Використання опису продукту та профілю користувача для рекомендацій на основі схожості.	Рекомендації на основі контенту та користувацьких вподобань.	Потрібна якісна структурована описова інформація про товари.
Рейтинговий метод	Базується на рейтингах, які надають користувачі.	Ефективність для рекомендацій на основі користувацьких відгуків.	Вимагають великої кількості рейтингів. Схильні до "проблеми холодного старту" для нових товарів.
Гібридний метод побудови рекомендацій	Комбінує фільтрацію на основі вмісту та колаборативну фільтрацію для ширшого асортименту продуктів.	Надає точніші рекомендації, уникнення загальних проблем.	Потребує комбінування структурованих даних про товари і користувачів.

Одним із підходів для створення рекомендації є використання векторних представлень товарів, які створюються за допомогою методів ШІ та МН. Товари представляються у векторному просторі, де кожен з них відображається

у векторі з числовими значеннями. Такий підхід дозволяє вимірювати схожість між товарами на основі відстаней векторів.

Різні методи створення векторних представлень для товарів та користувачів включають у себе як класичні підходи, такі як TF-IDF та Word2Vec, так і інші методи, які використовують нейронні мережі.

TF-IDF (частотно-інверсна частота документа) є методом аналізу тексту, який визначає важливість кожного терміну в документі. Загальний підхід TF-IDF поділяється на дві складові: TF (частота термінів) та IDF (інверсія частоти документа).

TF визначається як кількість разів, коли термін зустрічається в документі, поділена на загальну кількість термінів у документі. Однак, при розгляді TF можна виявити, що деякі загальні терміни, такі як "a" чи "the", отримують завищену важливість. Зазвичай такі слова просто видаляються з тексту. Однак може виникнути інша проблема, коли слова мають однакову частоту, але різну важливість для статті.

Тут вступає в силу IDF, яка враховує частоту термінів в усьому корпусі документів. IDF зменшує вагу термінів, що зустрічаються часто і збільшує вагу тих, що зустрічаються рідко. Інверсія частоти документа (IDF) визначається шляхом ділення кількості документів, що містять певний термін, на загальну кількість документів, після чого застосовується логарифмування. У випадку, коли термін часто входить в документи, значення IDF наближається до нуля. З іншого боку, якщо термін рідко зустрічається, значення IDF зростає.

Остаточна оцінка TF-IDF визначається перемноженням результатів TF і IDF. Вищий індекс TF-IDF вказує на більшу релевантність терміну в документах. Такий підхід дозволяє ефективно визначити важливість термінів в конкретних документах у контексті всього корпусу.

Word Embeddings (Word2Vec, GloVe) представляє собою метод, який використовує нейронні мережі для створення векторних представлень слів або об'єктів. Цей метод включає неглибоку двошарову нейронну мережу, яка використовується для створення вбудовувань слів. Основна ідея полягає в тому, щоб взяти обширний словник як вхідні дані, а потім створити векторний простір, де кожне слово відображається на унікальний вектор, що дозволяє представити зв'язок між словами. Word2Vec дозволяє захоплювати семантичну схожість між об'єктами, що робить його ефективним для рекомендацій на основі схожості. Важливою перевагою є швидкість навчання та здатність працювати з великими обсягами даних.

Існують дві основні архітектури моделі - CBOW (Continuous Bag of Words) і Skip-Gram. Обидві архітектури слугують для вивчення представлень слів, проте вони відрізняються за підходом і продуктивністю. Модель Skip-Gram передбачає слова, які оточують конкретне слово, з фокусом на контекстній схожості. З іншого боку, CBOW використовує підхід Bag of Words і прогнозує слово на основі контексту, що його оточує.

Важливо відзначити, що модель Skip-Gram є точнішою для рідковживаних слів та підходить для великих баз даних, але вимагає більше оперативної пам'яті. З іншого боку, CBOW є швидшою, менше вимогливою до оперативної пам'яті і підходить для менших баз даних.

FastText – це розширення Word2Vec, яке дозволяє представляти слова як комбінації їхніх підслів. Це дозволяє легше узгоджувати слова, що містять префікси або суфікси.

Doc2Vec є аналогом Word2Vec, але цей метод здатен створювати вектори для цілих документів або речень, замість окремих слів.

Transformer-based models (BERT, GPT) використовують методи, побудовані на трансформаторах, такі як BERT (Bidirectional Encoder

Representations from Transformers) та GPT (Generative Pre-trained Transformer), здатні генерувати контекстуалізовані вектори для слів або фраз.

Методи на основі векторних представлень мають кілька важливих переваг:

– точність — вони здатні до точних та персоналізованих рекомендацій, оскільки враховують схожість між об'єктами;

– швидкість навчання — деякі методи, такі як Word2Vec, навчаються дуже швидко, що дозволяє використовувати їх без зайвих витрат часу;

– здатність до роботи з різноманітними неструктурованими типами даних - вони можуть бути використані для аналізу текстів, зображень, відгуків користувачів, тощо.

Незважаючи на свої переваги, методи на основі векторних представлень також мають свої недоліки:

– вимоги до обчислювальних ресурсів — деякі методи можуть вимагати багато обчислювальних ресурсів для навчання та використання;

– потреба в великому обсязі даних — для досягнення високої якості рекомендацій часто потрібні великі обсяги даних;

– складність підтримки та налаштування — робота з методами на основі векторних представлень може бути складною в реалізації та налаштуванні.

У таблиці 1.2 наведено порівняння методів створення векторних представлень.

Таблиця 1.2 — Методи створення векторних представлень

Метод	Опис	Переваги	Недоліки
1	2	3	4
TF-IDF	Враховує важливість термінів у текстах за допомогою інвертованої частоти документів та частоти термінів	Ефективність врахування важливості термінів, простота в реалізації	Не враховує семантичні зв'язки між словами, не розуміє контексту
Word Embeddings (Word2Vec, Glove)	Представлення слів у просторі, де семантично схожі слова розташовані близько одне до одного	Враховує семантичні зв’язки між словами; здатність узагальнювати слова, яких не було в тренувальному датасеті	Вимагає довгого навчання та великої кількості тренувальних даних
FastText	Розширення Word2Vec, представляє слова як комбінації підслів	Враховує морфологічні особливості мови, що дозволяє легше узгоджувати слова, що містять префікси або суфікси	Вимагає більше обчислювальних ресурсів та часу через велику кількість параметрів для кожного слова

Кінець таблиці 1.2

1	2	3	4
Transformer-based models (BERT, GPT)	Методи, побудовані на трансформаторах, здатні генерувати контекстуалізовані вектори для слів або фраз	Здатність враховувати зв'язок між словами; висока точність	Вимагає великої кількості обчислювальних ресурсів для тренування та використання; потребує великої кількості даних для досягнення високої якості векторів
Doc2Vec	Метод, що базується на Word2Vec, але генерує вектори для цілих документів або речень	Здатність представляти семантику цілих документів або речень; може враховувати контекст слів у різних частинах тексту	Вимагає більше обчислювальних ресурсів та даних для тренування, особливо для великих корпусів текстів; може бути складно знаходити оптимальні гіперпараметри для конкретного завдання

Виходячи з результатів аналізу різних методів для створення рекомендацій, можна зробити висновок, що попри більші вимоги до обчислювальних ресурсів та довге навчання, методи на основі векторних представлень показують кращі результати при роботі з неструктурованими даними через врахування семантичних зв’язків між словами.

Набільшими перевагами пошуку на основі векторних представлень є:

– здатність до роботи з неструктурованими типами даних (текстом, зображеннями) – дозволяє зберегти час, що для традиційних методів був би використаний на створення бази характеристик, парсингу даних про товари та інше;

– семантичний пошук – враховує сенс і контекст запиту. Він здатен розуміти значення слів, їх взаємозв'язки та контекстуальні нюанси, що поліпшує точність результатів. Цей вид пошуку може враховувати схожі або пов'язані поняття, навіть якщо вони не містять точно ті ж самих ключових слів, що і в початковому запиті. Це дозволяє отримувати більше релевантних результатів.

1.3 Дослідження методів створення векторних вбудовувань

Text Embeddings (TE) – це потужний інструмент, який дозволяє перетворювати неструктуровані дані в структуровану форму, використовуючи вбудовування. Неструктуровані дані – це зображення, відео, аудіо, текст, молекулярні зображення та інші види даних, які не мають формальної структури [1].

Однією з ключових переваг є можливість порівняння фрагментів тексту, незалежно від їх довжини, будь то окремі слова, речення, абзаци чи навіть довші документи. Вбудовування тексту дозволяє взаємодіяти з неструктурованою текстовою інформацією за допомогою різних методів обробки та аналізу.

Схема створення векторного вбудовування наведена на рисунку 1.2.

Рисунок 1.2 – Схема створення векторного вбудовування

Вектор, отриманий в результаті обробки моделлю вбудовувань, складається з коефіцієнтів, які представляють собою ваги навченої моделі і відображають семантичні властивості тексту.

Одне із застосувань ТЕ – це семантичний пошук або визначення подібності. Замість обмеження результатів точним введеним запитом, вбудовування тексту дозволяє враховувати контекст і семантичне значення запиту, що полегшує пошук інформації у великих сховищах текстових даних.

Відображення семантичної схожості понять при векторному вбудовуванні наведено на рисунку 1.3.

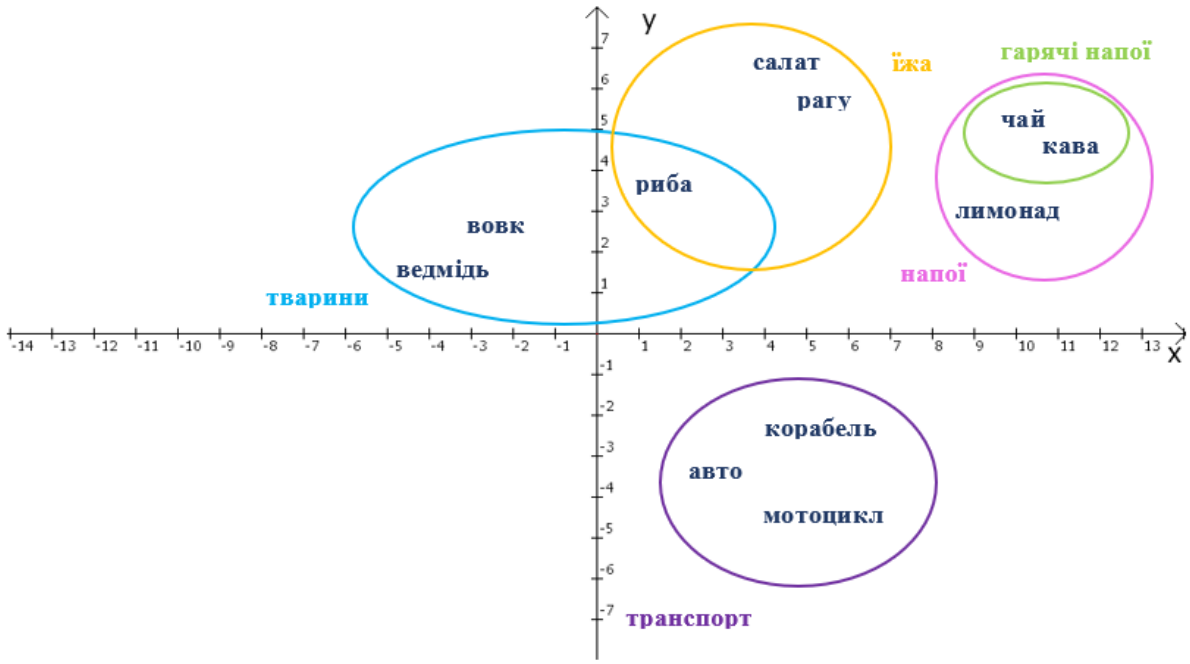


Рисунок 1.3 – Розташування векторних вбудовувань у просторі

Крім того, ТЕ допомагає в розв'язанні завдань кластеризації. Організації можуть використовувати цей метод для виявлення основних тем в колекції документів або для сегментації клієнтів на основі їхніх вподобань і діяльності. Алгоритми кластеризації, використовуючи вбудовання тексту, дозволяють легко виявляти нові закономірності в текстових даних.

Також ТЕ застосовується в класифікації тексту. Знаючи типи класів, до яких слід віднести дані, цей метод може бути використаний для автоматичного позначення токсичного вмісту, допомагаючи модераторам у виборі та обробці текстового матеріалу. Крім того, класифікація намірів за допомогою текстових вбудовувань може забезпечити ефективний спосіб обробки запитів клієнтів.

Існують два основних типи технік вбудовування для векторизації тексту: "розріджені" та "щільні". Обидва вони представлені у вигляді упорядкованих числових векторів, але відрізняються як методом генерації, так і здатністю кодувати властивості тексту.

Розріджені вбудовування використовують одногаряче кодування, де кожне унікальне слово пов'язане з певною позицією у векторі, що створює розріджений вектор. Такий метод можна легко застосувати до речень чи текстових колекцій, відображаючи не лише наявність слова, але й його частоту в тексті.

Для удосконалення розріджених вбудовувань існують методи, такі як tf-idf і BM25, які враховують значущість слів. Такі вбудовування швидко обчислюються, однак при цьому мають обмежені можливості: їх розмір залежить від обсягу словникового запасу, вони не представляють невідомих слів, не враховують контекст та не мають семантичного розуміння.

У порівнянні з цим, щільні вбудовування, такі як ті, що використовують моделі, засновані на нейронних мережах, наприклад BERT, розв'язують багато недоліків розріджених вбудовувань. Вони створюють вектори фіксованого розміру, здатні враховувати семантичні та синтаксичні зв'язки, а також представляти раніше невидимі слова.

Розміри цих щільних вбудовувань відображають різні властивості слів та текстів і залежать від процесу навчання моделі. Такі вбудовування надають абстрактне, але багатоаспектне відображення семантичних характеристик мови [5].

Процес навчання моделей для створення векторних вбудовувань — це ключовий етап для створення векторних представлень тексту та об'єктів, які можуть бути використані для пошуку, класифікації, рекомендацій та інших завдань обробки тексту.

Цей процес складається з п’яти етапів.

1. Збір та підготовка даних. Спочатку необхідно зібрати велику кількість навчальних даних, які відповідають задачам, які планується вирішувати з векторними представленнями. Після збору даних, їх необхідно підготувати, включаючи токенізацію тексту, видалення стоп-слів, лематизацію та інші операції обробки, які допомагають покращити якість векторних представлень.

2. Вибір моделі та архітектури. Після підготовки даних обирається модель та архітектура, яка найкраще підходить для задачі. Наприклад, для обробки тексту можуть бути вибрані моделі, які спеціалізуються на текстових даних, такі як OpenAI Embeddings, Word2Vec, BERT або Doc2Vec.

3. Навчання моделі. Навчання моделі включає в себе подачу навчальних даних моделі та оптимізацію параметрів, щоб отримати векторні представлення, які найкраще відображають семантику даних. Під час навчання моделі використовуються методи оптимізації, такі як стохастичний градієнтний спуск, для покращення векторних представлень, зменшення функції втрати та мінімізації помилок.

4. Оцінка моделі. Після завершення навчання моделі важливо провести оцінку її ефективності на тестових даних. Це допомагає визначити, наскільки добре векторні представлення відображають семантику та роблять прогнози для поставленого завдання. Метрики, такі як точність, відсоток правильних відповідей, коефіцієнти Кореляції Пірсона та Спірмена чи середні квадратичні помилки, можуть бути використані для оцінки моделі.

5. Налаштування та підтримка. Після навчання моделі може знадобитися налаштування параметрів, зокрема розміру векторів, швидкості навчання та інших гіперпараметрів, щоб покращити якість векторних

представлень. Модель також може вимагати періодичного перенавчання на нових даних, щоб залишатися актуальною та ефективною

1.4 Постановка задачі дослідження.

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є процес персоналізації переліку товарів для користувачів систем електронної комерції.

Предметом дослідження являються методи ВП в задачах формування персоналізованих рекомендацій.

Традиційні методи створення рекомендацій вже не є достатньо ефективними, тому не можуть повноцінно задовольнити потреби користувачів. До того ж вони мають безліч недоліків, таких як недостатня точність, необхідність великої кількості вхідної інформації, проблема «холодного старту» та інші. Векторний пошук допомагає значно полегшити створення персоналізованих переліків товарів для користувачів, при цьому дозволяючи оптимізувати процес зберігання описів товарів. Сучасні методи побудови рекомендацій орієнтовані на використання структурованих вхідних даних, що задають характеристики товарів. Однак, в більшості випадків інформація про товари, які використовує рекомендаційна система, надаються у вигляді текстового опису. Перетворення такого текстового опису в структурований вигляд, який може бути збережений в базі даних, вимагає значних витрат часу спеціалістів в даній предметній галузі. Тому удосконалення та подальше

використання семантичних методів пошуку підходящих товарів в рекомендаційній системі є актуальною задачею.

Метою даної роботи є дослідження методів ВП при побудові персоналізованого переліку товарів для користувачів систем електронної комерції.

Наукова новизна роботи полягає в удосконаленні методу векторного пошуку для побудови рекомендованого переліку товарів на основі їх текстового опису шляхом використання попередньої обробки вхідного тексту. Метод видаляє спеціальні символи, проводить токенізацію та стемінг, а також прибирає стоп-слова. Метод дозволяє зменшити фінансові витрати на створення векторного вбудовування за рахунок зменшення кількості токенів.

Задачами даної роботи є:

– дослідження особливостей рекомендаційної системи;

– дослідження традиційних методів та методів на основі векторних вбудовувань для створення рекомендацій;

– аналіз переваг та недоліків використання ВП в РС;

– удосконалення методу для створення персоналізованого переліку товарів на основі ВП за допомогою попередньої обробки вхідних даних;

– проведення експериментальної перевірки по удосконаленому методу.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ВЕКТОРНОГО ПОШУКУ

2.1 Обґрунтування вибору моделі для створення векторних вбудовувань

Нейронні мережі мають різні архітектури та навчаються на різних наборах даних, що робить векторне вбудовування кожної моделі унікальним. Ось чому робота з векторними вбудованнями є складною. [11]

На сьогодні існують такі моделі:

– OpenAI Embeddings. Вбудовування OpenAI представляє собою не традиційні вбудовування слів, а скоріше "текстові вбудовування" - абстракцію вищого рівня, що конвертує цілий фрагмент тексту у вектор чисел з плаваючою комою, набагато більше, ніж просто слово. Замість того, щоб кожне слово представлялося як фіксований вектор у високорозмірному просторі, вбудовування OpenAI є контекстуалізованими, що означає, що представлення кожного слова залежить від його оточення у тексті. Застосовуючи передові загальні мовні моделі, вбудовування OpenAI вражаюче точно визначає наміри та основні значення слів. Це ефективно навіть при порівнянні вбудовувань, написаних різними мовами, і дозволяє визначити їхню концептуальну взаємодію. Це математичне представлення тексту дозволяє виконувати корисні операції, такі як обчислення евклідової відстані або косинусної подібності між векторами, для визначення їхнього ступеня взаємозв'язку в задачах класифікації, пошуку, створення рекомендацій та ін.;

– Word2Vec — це модель, яка навчається векторним представленням слів на основі їхнього контексту в тексті. Вона дозволяє векторизувати слова, забезпечуючи їхнім семантичним значенням векторні представлення. Ця модель дозволяє визначати семантичну схожість слів на основі контексту та

використовувати ці вектори для різних завдань, таких як пошук синонімів, аналогій та класифікації тексту;

– GloVe — це інструмент для глобального векторного представлення слів. У порівнянні з Word2Vec, який фіксує лише локальний контекст слів, GloVe розглядає весь корпус і формує велику матрицю, що відображає взаємодію слів у всьому корпусі. Метод GloVe об'єднує переваги двослівних методів навчання, таких як матрична факторизація (наприклад, латентний семантичний аналіз або LSA) та метод локального контекстуального вікна (такий як Skip-gram). Його функція найменших квадратичних вартостей спрощує обчислювальні витрати навчання моделі, і вставки слів, отримані за допомогою GloVe, є відмінними та вдосконаленими. GloVe демонструє значні покращення у вирішенні завдань розпізнавання аналогій слів та іменованих об'єктів. У деяких завданнях він виявляється ефективнішим за Word2Vec, а в інших він конкурує з ним. Важливо відзначити, що обидва методи ефективно фіксують семантичну інформацію у текстовому корпусі;

– BERT (Bidirectional Encoder Representations from Transformers). Є однією з найбільш впливових моделей ВП. Вона здатна розуміти контекст тексту з урахуванням лівого та правого контексту, що допомагає вирішувати завдання, де важливе контекстуальне розуміння;

– Doc2Vec — це модель, аналогічна Word2Vec, але здатна створювати векторні представлення цілих документів замість окремих слів. Ця модель може бути використана для пошуку схожих документів на основі їхнього вмісту;

– FastText — це модель ВП, розроблена Facebook. Вона поєднує векторні представлення слів і здатність враховувати субслова. Це дозволяє ефективно векторизувати текст навіть для слів, які модель не бачила під час навчання.

Якщо брати до уваги аналіз існуючих моделей та обирати з доступних API для отримання вбудовування текстових вбудовувань, одним з найкращих варіантів є OpenAI GPT-3 Embedding API.

Зараз є декілька доступних моделей вбудовувань для вибору, проте важливо уникати старіших версій, особливо версії 1, які не рекомендовано використовувати. Для роботи було обрано text-embedding-ada-002, яка, за інформацією від OpenAI, володіє найкращою продуктивністю на датасеті BEIR (Benchmarking-IR) порівняно з попередніми моделями (рисунок 2.1).

MODEL	ROUGH PAGES PER DOLLAR	EXAMPLE PERFORMANCE ON BEIR SEARCH EVAL
text-embedding-ada-002	3000	53.9
-davinci--001	6	52.8
-curie--001	60	50.9
-babbage--001	240	50.4
-ada--001	300	49.0

Рисунок 2.1 – Продуктивність різних моделей OpenAI [2]

На ресурсі Hugging Face існує MTEB Leader Board (Massive Text Embedding Benchmark), який призначений для оцінки ефективності різних мовних моделей в завданнях природної мови.

Станом на січень 2023 року, згідно із зібраними даними з [3] text-embedding-ada-002 посідає 23 зі 128 місць, що свідчить про її високу якість та ефективність у різноманітних випробуваннях.

Зазначаючи text-embedding-ada-002 у цьому контексті, можна побачити діаграму на рисунку 2.2, де її продуктивність представлена в порівнянні з першою у рейтингу моделлю e5-mistral-7b-instruct на різних наборах даних і завданнях тестування.

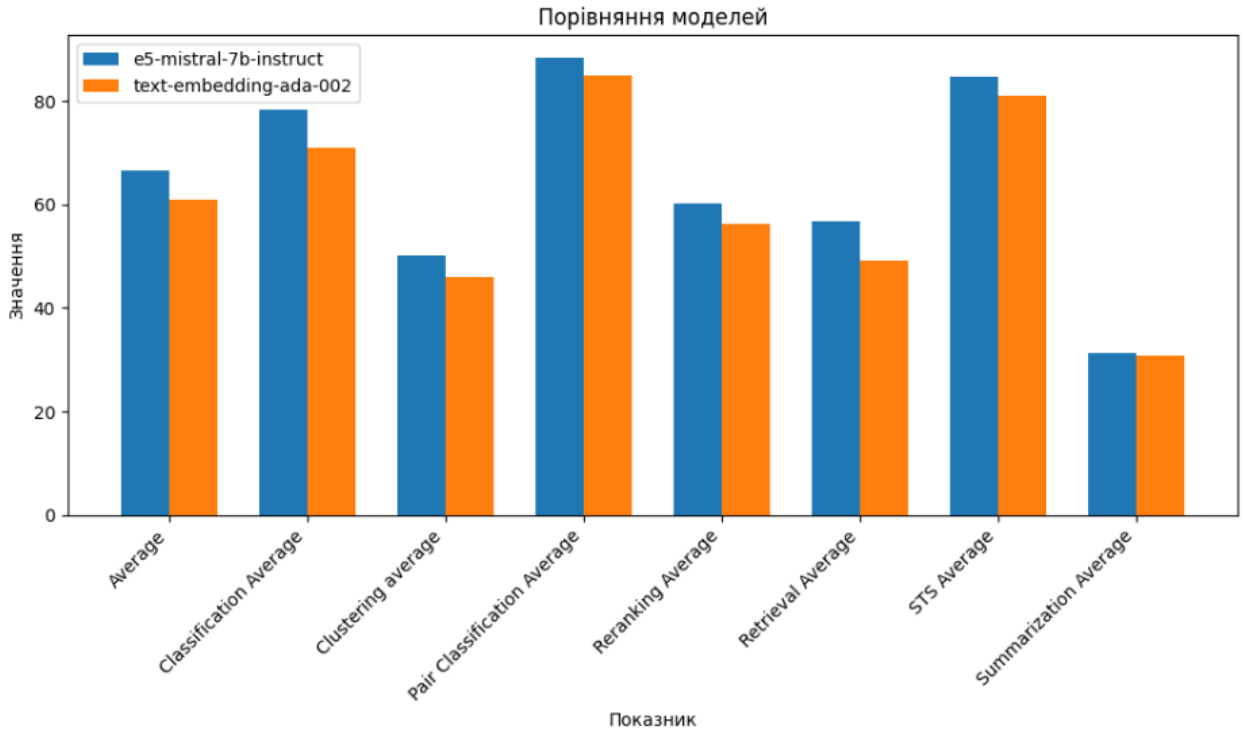


Рисунок 2.2 – Продуктивність моделі text-embedding-ada-002 у порівнянні з e5-mistral-7b-instruct (першою моделлю за рейтингом MTEB Leader Board)

Використання даної моделі також не є занадто дорогим. Кожен раз, коли викликається OpenAI Embedding API, обчислюється вартість, яка визначається за кількістю токенів у запиті та використовуваною моделлю. У випадку text-embedding-ada-002 це складає 0,01 цента за кожні 1000 токенів, що приблизно відповідає 4000 символам.

При роботі з даною моделлю варто враховувати те, що кожна модель OpenAI має обмеження за кількістю токенів. Для text-embedding-ada-002 це 8191 токен. Проте, цей недолік навряд створить проблеми, адже товари зазвичай мають доволі стислі описи.

2.2 Обґрунтування вибору векторної бази даних

Вбудовування представляють собою вектори, тобто набори чисел, виражені в комп'ютерній системі як числа з плаваючою точкою, зазвичай розміром 4 байти для одинарної точності. Вектор, як тип даних, може бути збережений та оброблений більшістю сучасних баз даних. Однак для оптимізації пошуку векторів важливо мати певні можливості, і саме для цього на сучасному ринку існують векторні бази даних [4].

Такі спеціалізовані бази даних, спроектовані для обробки векторних даних, стають невід'ємною частиною розвитку сучасних технологій, особливо в галузі штучного інтелекту та аналізу великих обсягів даних.

Традиційні скалярні бази даних виявляються неефективними при роботі з векторними даними через їхню складність та масштаб. Векторні бази даних вирішують цю проблему, надаючи продуктивність, масштабованість і гнучкість, необхідні для ефективного використання векторних вбудовувань. Їхні можливості поєднують традиційні функції баз даних з спеціалізацією на роботі з векторними даними.

Використання векторних баз даних у сфері штучного інтелекту дозволяє додати розширені функції, такі як пошук семантичної інформації та довготривала пам'ять. Це відкриває нові можливості для розвитку додатків, які вимагають обробки великих обсягів складних даних в реальному часі. Діаграма на рисунку 2.3 дає краще розуміння векторних баз даних у додатках такого типу.

Рисунок 2.3 – Діаграма зв’язку векторної БД з додатком

Етапи побудови рекомендацій на основі векторного пошуку.

1. Створення структури для векторної бази даних і вибір критерію для розрахунку подібності векторів.

2. Побудова векторних вбудовувань з вхідних даних (описів товарів, тощо).

3. Нормалізація отриманих векторів (Цей процес гарантує, що показники подібності зосереджуються на напрямку вектора, а не на величині, забезпечуючи більш точні порівняння.)

4. Отримані векторні вбудовування додаються до векторної бази даних із додатковим текстовим полем контенту, з якого вони були створені та полем часу створення вектора.

5. Отримання текстового запиту від користувача.

6. Створення векторного вбудовування із запиту та його нормалізація.

7. Надсилання запиту у векторну БД для пошуку в базі даних найбільш схожих векторів.

8. Отримання переліку найбільш схожих описів у первинній текстовій формі, що береться з поля вихідного контенту після сортування записів за схожістю векторів.

Автономні векторні індекси, представлені, наприклад, системою FAISS (Facebook AI Similarity Search), відзначаються високою ефективністю у векторному пошуку, проте вони обмежені в ряді можливостей, які визначаються наявністю спеціалізованих векторних баз даних. Векторні бази даних, спеціально розроблені для управління векторними вбудовуваннями, надають декілька вагомих переваг порівняно з використанням автономних векторних індексів.

По-перше, вони пропонують ефективне управління даними, забезпечуючи зручні функції для вставки, видалення та оновлення інформації. Це робить керування та підтримку векторних даних більш простими порівняно з автономними векторними індексами, які вимагають додаткових заходів для інтеграції з рішеннями зберігання. Векторні бази даних також відрізняються можливістю зберігання та фільтрації метаданих, що дозволяє користувачам проводити детальніші запити за допомогою додаткових фільтрів.

Крім того, векторні бази даних спеціально створені для масштабування та надають кращу підтримку розподіленої обробки даних та паралельних операцій. Вони також дозволяють оновлення в реальному часі, що дозволяє динамічно модифікувати дані, в той час як автономні векторні індекси можуть вимагати повного процесу повторного індексування. Резервне копіювання та колекції векторних баз даних забезпечують надійний захист даних та можливість вибору сегментарних резервних копій для ефективного використання.

Векторні бази даних представляють собою важливий етап еволюції систем управління даними, відрізняючись від традиційних баз даних. Замість

зберігання скалярних даних у вигляді рядків і стовпців, вони працюють з векторами, впроваджуючи новий підхід до оптимізації та виконання запитів. У традиційних базах даних запити зазвичай спираються на точну відповідність значень, тоді як векторні бази використовують метрику подібності для знаходження найближчого вектора до запиту.

Векторні бази даних використовують показники подібності для оцінки ступеня схожості між векторами у векторному просторі, що визначає релевантність результатів для конкретного запиту. Міри подібності визначають математичні методи для вимірювання схожості між двома векторами та грають важливу роль у роботі векторних баз даних.

Однією з основних мір подібності є косинусна подібність, яка визначає косинус кута між двома векторами у векторному просторі. Цей показник варіюється від -1 до 1, де значення 1 вказує на ідентичність векторів, 0 — на ортогональність, а -1 — на протилежність. Розраховується даний показник за формулою 2.1.

$$cos\,\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\cdot\|\vec{b}\|}, \tag{2.1}$$

де a та b – вектори, для яких визначається подібність.

Іншою мірою є евклідова відстань, яка вимірює відстань між векторами по прямій лінії і визначається від 0 до безкінечності. Для розрахунку використовують формулу 2.2:

$$\mathrm{d}(\vec{a},\vec{b}) = \sqrt{\sum_{i=1}^{n}\left(\vec{a}_i - \vec{b}_i\right)^2}, \tag{2.2}$$

де a та b – вектори, для яких визначається подібність;

n – розмірність векторів a і b.

Скалярний добуток є ще однією мірою, яка враховує добуток величин векторів та косинус кута між ними. Цей показник коливається від -∞ до ∞, і його додатні значення вказують на спрямованість векторів в одному напрямку, 0 — на ортогональність, а від'ємні значення — на напрямок в протилежні сторони. Розрахунки проводять за формулою 2.3:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} \vec{a}_i \vec{b}_i, \tag{2.3}$$

де a та b – вектори, для яких визначається подібність;

n – розмірність векторів a і b.

Використання цих мір подібності дозволяє ефективно порівнювати та визначати ступінь схожості векторів у векторних базах даних [6].

Векторних БД з інтегрованим модулем для створення векторних вбудовувань, що можна було б використати без залучення розробників, під час аналізу знайдено не було. Тим не менш, було знайдено декілька сервісів для роботи з векторними БД, які можна налаштувати, використовуючи програмування.

Перший — це Weaviate [12]. Це платний сервіс, що надає можливість спочатку створити векторні представлення даних за допомогою, наприклад, OpenAI Embeddings, а потім виконувати пошук за цими векторами. Також є

можливість додати додаткові фільтри та використовувати вектори, згенеровані за допомогою інших сервісів.

Технологією ВП також користується Google. Компанія запропонувала продукт Vertex AI Vector Search [13] - тут немає вбудованих генераторів векторів, але можна використати сервіс Generative AI [14], що також розроблений Google.

Ще зі світових гігантів Microsoft створив векторну БД Azure Cognitive Search [15]. Цей сервіс вони пропонують використовувати у комбінації із ще одним своїм продуктом Azure OpenAI Embeddings models [16].

Також, було знайдено декілька сервісів, що мають векторні бази даних з методами для пошуку найбільш схожих між собою векторів. Такі сервіси також інколи мають рекомендації для того, як використовувати їх в якості рекомендаційних систем:

Milvus [17] - сучасна безкоштовна хмарна векторна БД. Серед її переваг: висока продуктивність при проведенні ВП на великих наборах даних; спільнота розробників, яка пропонує багатомовну підтримку та набори інструментів; хмарна масштабованість і висока надійність навіть у разі збою; гібридний пошук досягається шляхом поєднання скалярної фільтрації з пошуком векторної подібності.

В якості сервісу для створення векторних вбудовувань в документації Milvus рекомендують використовувати Towhee [18].

ElasticSearch [19] - швидка та надійна векторна БД, що вже давно завоювала довіру користувачів. Є безкоштовною.

Amazon також створив векторну БД OpenSearch Service [20], але ця база є платною і не має рекомендацій щодо сервісів для створення векторних представлень.

В таблиці 2.1 наведена порівняльна характеристика знайдених сервісів.

Таблиця 2.1 - Порівняльна характеристика знайдених сервісів

Назва сервісу	Компанія-розробник	Наявність рекомендованих або вбудованих дод. сервісів	Особливості
1	2	3	4
Weaviate	B.V. Built та Docusaurus	Так, є інтеграція OpenAI Embeddings	- Платний - Надає можливість створення векторних представлень даних. - Дозволяє виконувати пошук за векторами та додавати фільтри. - Можливість використання векторів інших сервісів.
Vertex AI Vector Search	Google	Так, для генерації векторів пропонується використовувати Generative AI	- Платний - Використовує технологію ВП.
Azure Cognitive Search	Microsoft	Так, рекомендується використовувати в комбінації з Azure OpenAI Embeddings	- Платна векторна БД

Кінець таблиці 2.1

1	2	3	4
Milvus	LF AI & DATA	Так, рекомендують Towhee	- Безкоштовна хмарна векторна БД з високою продуктивністю та підтримкою спільноти розробників. - Хмарна масштабованість і висока надійність навіть у разі збою - Гібридний пошук досягається шляхом поєднання скалярної фільтрації з пошуком векторної подібності
ElasticSearch	Elastic	Ні	- Швидка та надійна векторна БД - Є безкоштовною - Має стек з додатковим функціоналом Elastic Stack.
OpenSearch Service	Amazon	Ні	- Платна векторна БД без рекомендацій щодо сервісів для створення векторних представлень

В якості векторної БД для роботи було обрано Elasticsearch, адже ця БД швидка, надійна та є безкоштовною.

Також, для полегшення використання Elasticsearch в різних сценаріях розробки та адміністрування є можливість використовувати Elastic Stack (Beats, Elasticsearch, Logstash, Kibana) [7], зв’язок яких наведено на рисунку 2.4. Logstash використовується для збору, обробки та направлення лог-даних в

Elasticsearch. Beats – легкі агенти для надсилання різних видів данних в Elasticsearch, таких як логи, метрики та інше. Kibana, в свою чергу, забезпечує візуалізацію та аналіз даних, збережених у Elasticsearch, надаючи інтерактивний інтерфейс для керування та моніторингу даних.

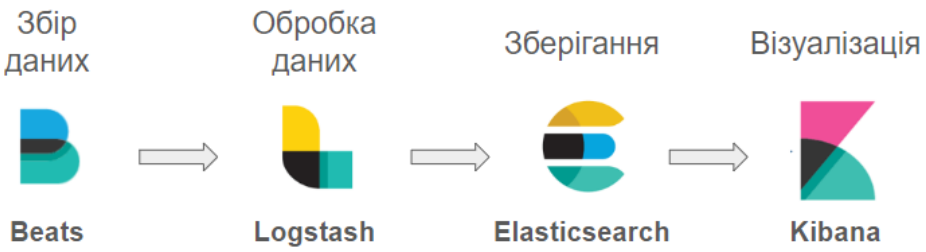


Рисунок 2.4 – Зв’язок між компонентами Elastic Stack

Elastic Stack стає важливим інструментарієм для комплексного розв'язання завдань збору, аналізу та візуалізації даних. Зручність налаштування, гнучкість та здатність обробляти різноманітні дані роблять Elasticsearch разом із компонентами Elastic Stack популярними в середовищі розробників та адміністраторів систем управління даними.

2.3 Удосконалення методу створення персоналізованого переліку товарів на основі векторного пошуку

Використання векторних представлень продуктів за допомогою методів ШІ та МН є перспективним підходом до створення рекомендацій. Цей метод

дозволяє представляти продукти у векторному просторі, вимірюючи подібність між ними на основі векторних відстаней.

Методи, засновані на векторних представленнях, мають численні переваги, включаючи можливість пошуку по неструктурованим даним, семантичність пошуку, високу точність та можливість працювати з малою кількістю вхідної інформації.

Результатом роботи є удосконалений метод створення персоналізованого переліку товарів на основі векторного пошуку, який використовує модель для створення вбудовувань OpenAI Embeddings та векторну БД ElasticSearch для їх зберігання.

До методу були додані такі етапи попередньої обробки тексту, як:

– очищення тексту від спеціальних символів;

– токенізація (процес розбиття тексту на окремі одиниці, які називаються токенами. Токени можуть бути словами, фразами, реченнями або навіть символами, залежно від завдання і вимог обробки тексту);

– приведення тексту до нижнього регістру;

– стемінг (процес зведення слова до його основи шляхом відкидання вторинних частин, таких як закінчення або суфікс).

Це має наступні переваги.

1. Очищення тексту від зайвих символів дозволяє зберегти лише значущі компоненти тексту. Це особливо корисно для створення векторів, оскільки усувається непотрібний шум, що може впливати на якість вбудовувань.

2. Токенізація, переведення тексту в набір окремих слів, дозволяє працювати з окремими одиницями тексту, полегшуючи процес обробки тексту.

3. Стемінг та вилучення стоп-слів сприяє зменшенню кількості токенів у кінцевому тексті. Це може призвести до прискорення процесу

створення векторів, оскільки менша кількість токенів означає менші вимоги щодо обчислювальних ресурсів.

4. Модель для створення вбудовувань має обмеження по кількості токенів за один запит, тому видалення спеціальних символів, стоп-слів та стемінг дають можливість брати на вхід описи значно більших обсягів. До того ж кількість токенів безпосередньо впливає на вартість запиту, чим менше токенів – тим менша вартість.

В цілому, ці операції попередньої обробки дозволяють усунути шум, забезпечуючи краще відображення сутності тексту та сприяючи зниженню вартості запитів за рахунок зменшення розміру запиту до моделі вбудовувань та оптимізації обробки текстових даних.

Вдосконалений метод використовує вбудовування OpenAI та алгоритми рекомендацій на основі косинусної подібності між вектором пошукового запиту та існуючими продуктами в базі даних Elasticsearch. Це передбачає створення системи рекомендацій для покупок, підкріпленої семантичним пошуком. Ключовими перевагами цієї системи є можливість працювати з неструктурованими описами товарів, а також відсутність необхідності ручного навчання моделі та швидкість впровадження такої системи рекомендацій у поєднанні з високою якістю отриманих рекомендацій.

Система рекомендацій для персоналізованого переліку товарів в основному включає два модулі. Перший слугує для створення векторних вбудовувань на основі даних про товар і тексту пошукового запиту за допомогою OpenAI Embeddings, а другий є модулем для збереження інформації в векторній базі даних Elasticsearch з подальшим використанням косинусної подібності векторного вбудовування у функції пошуку для отримання рекомендацій щодо продуктів.

Вхідними даними є описи товарів, пошуковий запит користувача, структура БД та критерій для розрахунку подібності векторів.

Процес створення рекомендацій складається з 10 етапів.

Етап 1. Отримання даних з описами товарів.

Етап 2. Попередня обробка описів товарів (видалення спеціальних символів, токенізація, стемінг, видалення стоп-слів). З точки зору стемінгу слово розглядається у вигляді двох частин: стему та афіксу. Формулою слова для стемінгу є:

$$W = S + A, \tag{2.4}$$

де W – слово;

S – стем;

A – афікс.

Словники стемів та афіксів формуються за формулами 2.5 та 2.26 відповідно.

$$Vs = \{S_1, S_2, S_3, \dots, S_n\}, \tag{2.5}$$

де Vs – словник стемів;

S_i – і-тий стем.

$$V_A = \{A, A_2, A_3, \dots, A_n\}, \tag{2.6}$$

де V_A – словник афіксів;

A_i – і-тий афікс.

Етап 3. Побудова векторних вбудовувань з описів товарів за допомогою OpenAI Embedding.

Етап 4. Нормалізація отриманих векторів (Цей процес гарантує, що показники подібності зосереджуються на напрямку вектора, а не на величині, забезпечуючи більш точні порівняння)

Етап 5. Отримані векторні вбудовування додаються до векторної бази даних із додатковим текстовим полем контенту, з якого вони були створені та полем часу створення вектора.

Етап 6. Отримання текстового запиту від користувача.

Етап 7. Попередня обробка тексту (видалення спеціальних символів, токенізація, стемінг, видалення стоп-слів).

Етап 8. Створення векторного вбудовування із запиту за допомогою OpenAI Embedding та його нормалізація.

Етап 9. Надсилання запиту у векторну БД Elasticsearch для пошуку в базі даних найбільш схожих векторів.

Етап 10. Отримання переліку описів рекомендованих товарів.

Для розрахунку нормалізованого вектора використовувались формули 2.7 та 2.8:

$$Д = \sqrt{\sum_{i=1}^{n} К_i^2}\,, \tag{2.7}$$

де Д – довжина вектора;

$К_i$ – і-тий компонент вхідного вектора.

$$НВ = \left[\frac{К_1}{Д}, \frac{К_2}{Д}, \cdots, \frac{К_n}{Д}\right], \quad (2.8)$$

де НВ – нормалізований вектор;

$К_i$ – і-тий компонент вхідного вектора;

Д – довжина вектора.

Зв'язок між модулями показаний на рисунку 2.5.

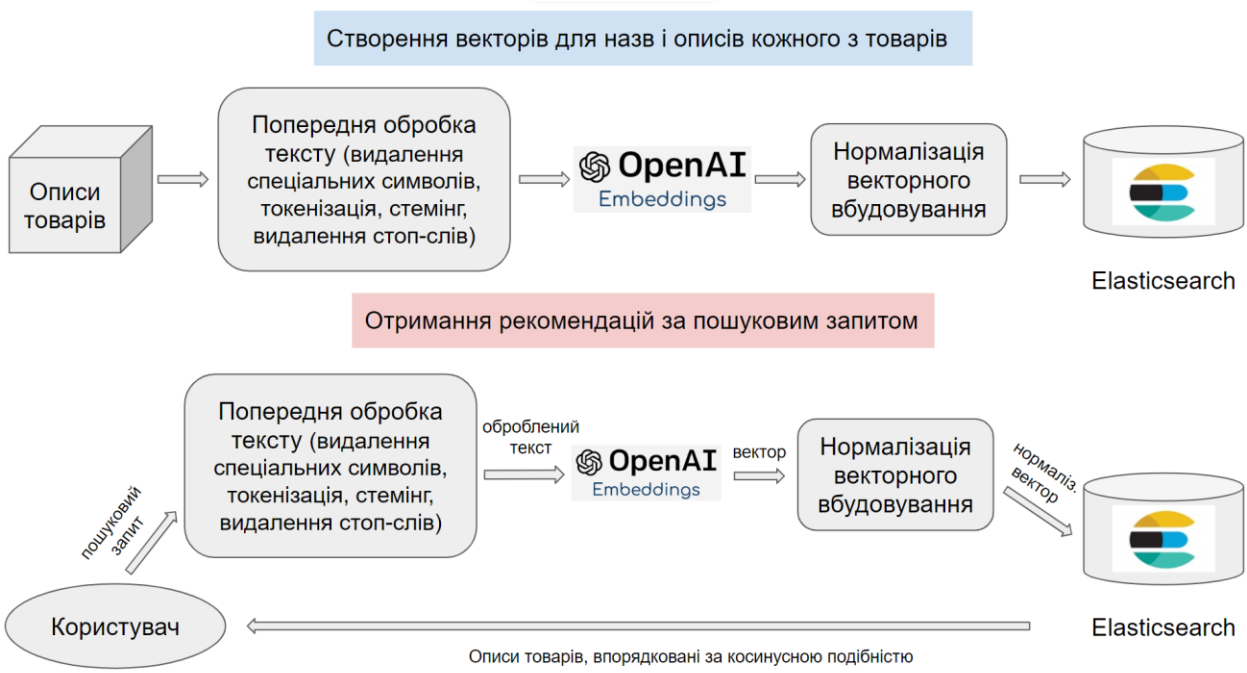


Рисунок 2.5 – Структура вдосконаленого методу

Отриманий рекомендований перелік товарів враховуватиме не ключові слова, не рейтинги і не історію пошуку, як в звичайних РС, а семантичну схожість запиту користувача і опису товарів. Це дозволить користувачу описувати свої побажання легше та зручніше, не спираючись на текстові співпадіння та заздалегідь сформований на основі історичних даних чи відгуків інших користувачів перелік товарів. Цей персоналізований підхід

дозволить отримати результати, що задовольняють потребам саме поточного запиту.

3 РОЗРОБКА ТЕХНОЛОГІЇ ПОБУДОВИ РЕКОМЕНДАЦІЙ З ВИКОРИСТАННЯМ ВЕКТОРНОГО ПОШУКУ

3.1 Технологія формування персоналізованого переліку товарів на основі векторного пошуку

Технологія використання удосконаленого методу векторного пошуку з попередньою обробкою вхідних даних складається з етапів, що наведені на рисунку 3.1.

Етапи створення структури для індексу векторної бази даних та вибору методу для розрахунку подібності векторів налаштовують БД для подальшого зберігання то пошуку векторів.

Етап попередньої обробки вхідних даних та створення векторних вбудовувань. За допомогою очищення тексту від спеціальних символів, токенізації, приведення тексту до нижнього регістру та стемінгу описи товарів готуються для подальшої обробки. Далі використовується модель для створення векторних вбудовувань OpenAI Embeddings.

Етап нормалізації векторів та збереження у БД для кожного вектора створює нормалізовану версію вектора. Після цього відбувається збереження нормалізованого вектора у векторну БД Elasticsearch.

На етапі отримання пошукового запиту та його попередньої обробки до методу надходить запит від користувача. Далі відбувається попередня обробка та створення векторного вбудовування для отриманого запиту.

На етапі нормалізації вектора та надсилання запиту у БД векторне вбудовування, що було отримане з пошукового запиту нормалізується та з ним

формується запит на пошук n-ої кількості рекомендованих товарів, що будуть відсортовані за показником косинусної подібності.

На останньому етапі з отриманої від БД відповіді формується персоналізованих перелік товарів для користувача та виводиться на екран.

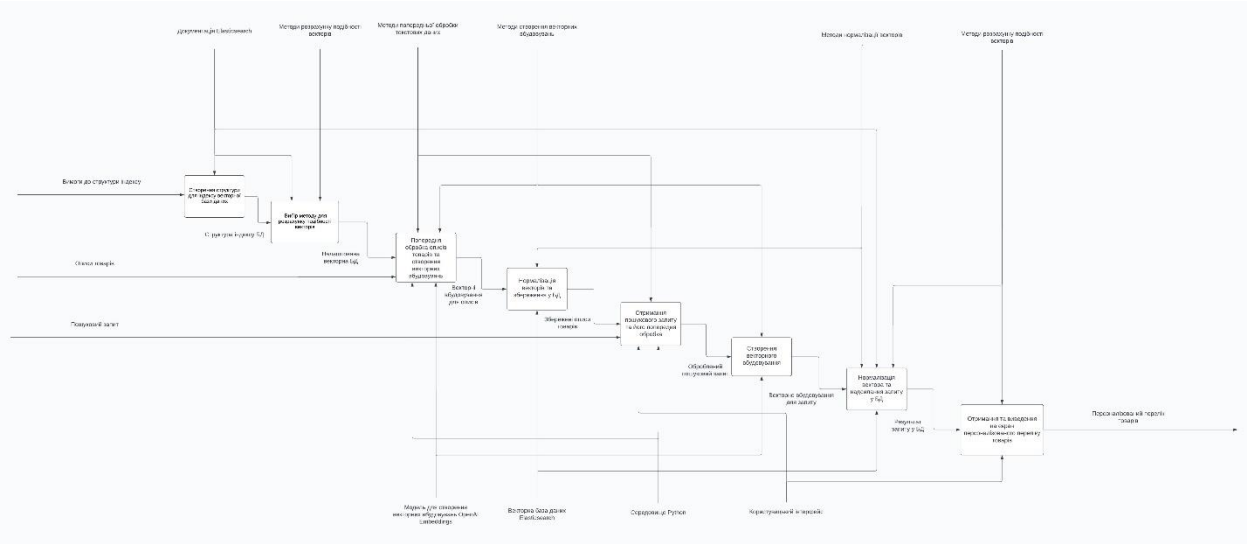


Рисунок 3.1 – Діаграма удосконаленого методу

3.2 Імплементація модулю векторного пошуку

Загальна структура модулю векторного пошуку показана на рисунку 3.2. Спочатку запит надходить від користувача, далі обробляється проксі-сервером Nginx і надсилається на сервер Laravel, після чого обробляється OpenAI Embeddings. Отриманий вектор передається як запит до Elasticsearch. Використовуючи косинусну подібність, Elasticsearch повертає попередньо визначену кількість товарів, відсортованих за найвищим рівнем подібності між вектором пошукового запиту та вектором, створеним з опису продукту. Усі

дані, що містяться в Elasticsearch, можна переглянути в панелі адміністратора Kibana.

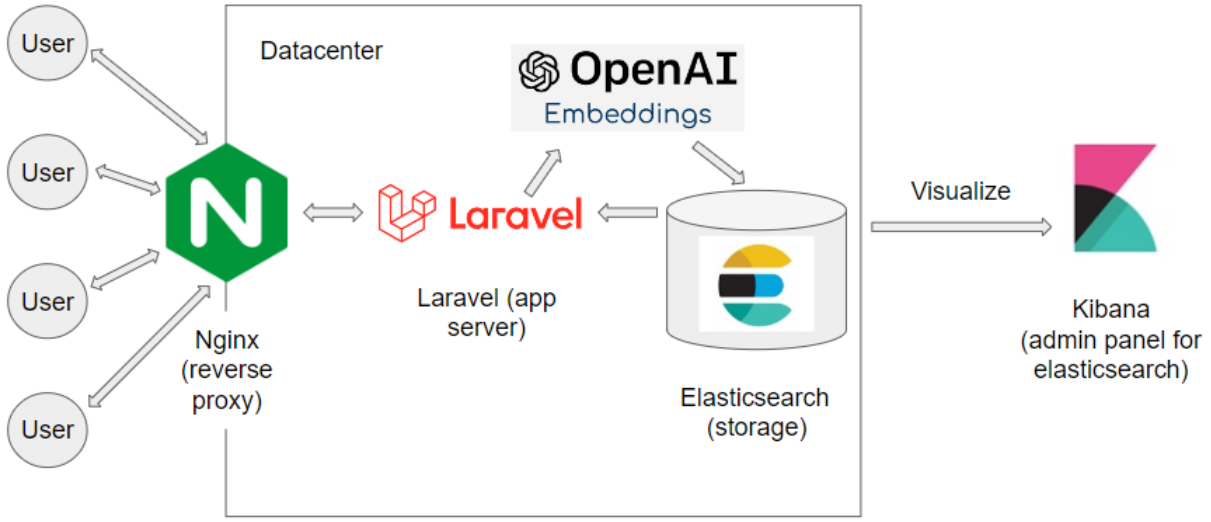


Рисунок 3.2 – Загальна структура системи

Для зберігання та порівняння векторів після створення векторних вбудовувань за допомогою OpenAI використовується нормалізація. Нормалізація векторів має важливе значення під час роботи з векторними вбудовуваннями в Elasticsearch, особливо для алгоритмів на основі косинусної подібності. Цей процес гарантує, що показники подібності зосереджуються на напрямку вектора, а не на величині, забезпечуючи точні та значущі порівняння. У проекті вектори масштабуються до одиничної довжини, роблячи їх незалежними від величини та покращуючи надійність моделі. У контексті Elasticsearch ця нормалізація підвищує ефективність і результативність пошуку, оскільки забезпечує послідовне та надійне порівняння векторів, зменшуючи вплив різної довжини вектора на результати пошуку.

Весь процес модулю векторного пошуку налаштований в контейнерах Docker. Для контейнера Elasticsearch та Kibana застосовувались існуючі образи docker.elastic.co/elasticsearch/elasticsearch:8.8.2 та docker.elastic.co/kibana/kibana:8.8.2 відповідно, для проксі-сервера Nginx застосовувався образ nginx:alpine.

Налаштування усіх сервісів у docker-compose наведені на рисунку 3.3 та 3.4.

```yaml
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:8.8.2
  ports:
    - ${FORWARD_ELASTICSEARCH_PORT:-9200}:9200
  environment:
    - node.name=elasticsearch
    - cluster.name=es-docker-cluster
    - discovery.type=single-node
    - bootstrap.memory_lock=true
    - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
  ulimits:
    memlock:
      soft: -1
      hard: -1
  volumes:
    - elasticsearch:/usr/share/elasticsearch/data
  networks:
    - ai-api

kibana:
  image: docker.elastic.co/kibana/kibana:8.8.2
  ports:
    - ${FORWARD_KIBANA_PORT:-5601}:5601
  environment:
    - ELASTICSEARCH_USERNAME=${ELASTICSEARCH_USERNAME}
    - ELASTICSEARCH_PASSWORD=${ELASTICSEARCH_PASSWORD}
  depends_on:
    - elasticsearch
  networks:
    - ai-api
```

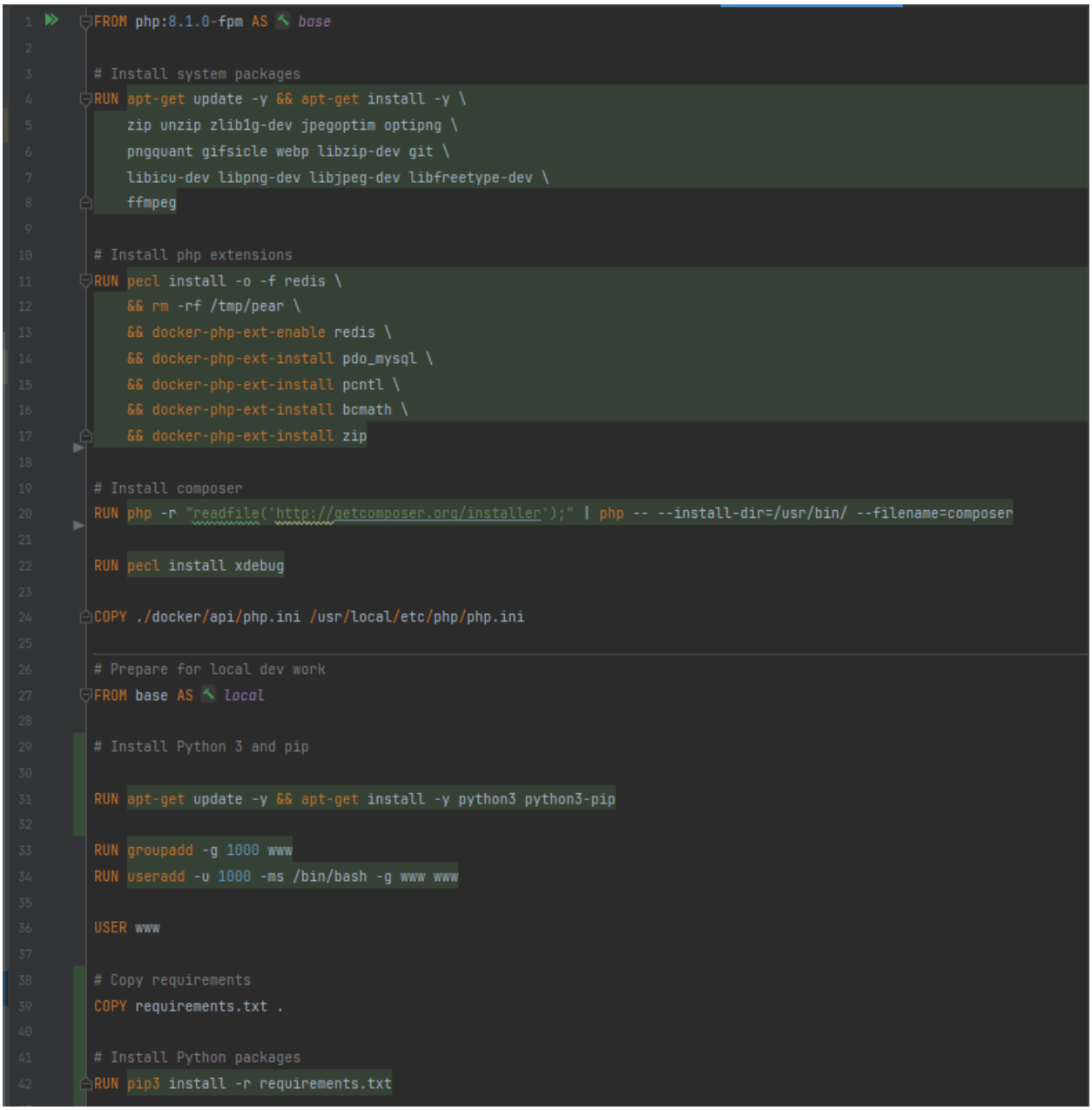
Рисунок 3.3 – Налаштування сервісів elasticsearch та kibana

```yaml
nginx:
    image: 'nginx:alpine'
    networks:
        - ai-api
    ports:
        - '${APP_PORT:-8000}:80'
    volumes:
        - '.:/var/www/html'
        - './docker/nginx/nginx.conf:/etc/nginx/nginx.conf'
        - './docker/nginx/conf.d:/etc/nginx/conf.d'
    depends_on:
        - api
    extra_hosts:
        - "host.docker.internal:host-gateway"
    environment:
        PHP_IDE_CONFIG: "serverName=${SERVER_NAME}"

api:
    build:
        context: .
        dockerfile: docker/api/Dockerfile
        target: 'local'
    networks:
        - ai-api
    volumes:
        - '.:/var/www/html'
        - './docker/api/php.ini:/usr/local/etc/php/php.ini'
        - './docker/ssh:/home/www/.ssh'
        - './docker/api/.bashrc:/home/www/.bashrc'
    depends_on:
        - elasticsearch
        - kibana
    extra_hosts:
        - "host.docker.internal:host-gateway"
    environment:
        PHP_IDE_CONFIG: "serverName=${SERVER_NAME}"
```

Рисунок 3.4 – Налаштування сервісів nginx та api

Для Laravel API використовувався самостійно створений образ, фрагмент якого наведено на рисунку 3.5.



```dockerfile
FROM php:8.1.0-fpm AS base

# Install system packages
RUN apt-get update -y && apt-get install -y \
    zip unzip zlib1g-dev jpegoptim optipng \
    pngquant gifsicle webp libzip-dev git \
    libicu-dev libpng-dev libjpeg-dev libfreetype-dev \
    ffmpeg

# Install php extensions
RUN pecl install -o -f redis \
    && rm -rf /tmp/pear \
    && docker-php-ext-enable redis \
    && docker-php-ext-install pdo_mysql \
    && docker-php-ext-install pcntl \
    && docker-php-ext-install bcmath \
    && docker-php-ext-install zip

# Install composer
RUN php -r "readfile('http://getcomposer.org/installer');" | php -- --install-dir=/usr/bin/ --filename=composer

RUN pecl install xdebug

COPY ./docker/api/php.ini /usr/local/etc/php/php.ini

# Prepare for local dev work
FROM base AS local

# Install Python 3 and pip

RUN apt-get update -y && apt-get install -y python3 python3-pip

RUN groupadd -g 1000 www
RUN useradd -u 1000 -ms /bin/bash -g www www

USER www

# Copy requirements
COPY requirements.txt .

# Install Python packages
RUN pip3 install -r requirements.txt
```

Рисунок 3.5 – Фрагмент Dockerfile для створення контейнера API

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ ДОСЛІДЖЕНИХ РЕЗУЛЬТАТІВ

4.1 Програмна реалізація удосконаленого методу

Для роботи з векторною базою даних необхідно створити індекс, який в подальшому зберігатиме документи з усіма векторними вбудовуванням описів товарів.

Структура індексу наведена на рисунку 4.1.

```php
$this->client->indices()->create([
    'index' => $this->indexName,
    'body' => [
        'mappings' => [
            '_source' => [
                'enabled' => true,
            ],
            'properties' => [
                'embedding' => [
                    'type' => 'dense_vector',
                    'index' => true,
                    'dims' => 1536,
                    'similarity' => 'cosine',
                ],
                'description' => [
                    'type' => 'keyword',
                ],
                'created_at' => [
                    'type' => 'date',
                ],
            ],
        ],
    ],
]);
```

Рисунок 4.1 – Створення індексу Elasticsearch з відповідною структурою документів

Параметр «index» вказує ім'я створюваного індексу, яке визначається значенням змінної $this->indexName. В тілі запиту («body») визначається структура індексу, включаючи параметри мапінгу.

Мапінг визначає, які поля будуть у документах і як будуть оброблятися. У даному випадку, для кожного документа в індексі передбачено поле «embedding», яке вказується як dense_vector (щільний вектор) [8] з розмірністю 1536 (розмірність вбудовування OpenAI Embedding). Параметр «index» встановлюється в «true», що дозволяє використовувати це поле для пошуку. Для «similarity» встановлено «cosine», що означає використання косинусної подібності для визначення схожості між векторами.

Також додано поля «description» типу «keyword» для оригінального текстового опису товару. Тип «keyword» використовується для зберігання текстових даних, які не потрібно аналізувати.

Поле «created_at» типу «date» використовується для зберігання дат створення документів.

Поле «_source» містить вихідне тіло документа JSON, передане під час індексування. Обробка «_source» увімкнена, що означає, що оригінальні документи будуть збережені в індексі для подальшого отримання.

У вдосконаленому методі була додана попередня обробка вхідного тексту, код для якої наведено на рисунку 4.2.

```python
def preprocess_text(text: str):
    # Step 1: Text Cleaning (Remove punctuation, retain hyphens)
    cleaned_text = re.sub(r'[^\w\s-]', '', text)

    # Step 2: Tokenization
    tokens = word_tokenize(cleaned_text)

    # Step 3: Lower-casing
    lowercase_tokens = [token.lower() for token in tokens]

    # Step 4: Stopword Removal
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [token for token in lowercase_tokens if token not in stop_words]

    # Step 5: Stemming
    stemmer = PorterStemmer()
    stemmed_tokens = ' '.join([stemmer.stem(token) for token in filtered_tokens])

    return stemmed_tokens
```

Рисунок 4.2 – Попередня обробка вхідного тексту

Метод «preprocess_text» виконує комплексну обробку тексту для подальшого створення вбудовувань. Він включає:

– регулярний вираз для видалення всіх символів, крім букв, цифр, пробілів і дефісів;

– word_tokenize з бібліотеки NLTK для розділення тексту на окремі слова або токени;

– за допомогою stopwords.words('english') з NLTK відбирається список стоп-слів для англійської мови, після чого всі токени, які є стоп-словами, вилучаються з тексту;

– застосовується стемінг до кожного токену за допомогою PorterStemmer з NLTK.

Для створення векторного вбудовування спочатку для описів товарів, а потім для пошукового запиту користувача застосовується метод «createEmbedding», код якого наведено на рисунку 4.3.

```php
public function createEmbedding(Enums\Model $model, string $input): Contracts\EmbeddingContract
{
    $response = $this->client->embeddings()->create([
        'model' => $model->value,
        'input' => $input,
    ]);

    return new Embedding(
        $response->embeddings[0]->embedding,
        $response->usage->promptTokens,
        $response->usage->totalTokens
    );
}
```

Рисунок 4.3 – Метод для створення векторного вбудовування

Для створення вбудовування необхідно вказати контент та назву моделі, в нашому випадку text-embedding-ada-002. Після цього з даних відповіді створюється вбудовування класу Embedding для подальшого використання.

Для збереження документу в індекс Elasticsearch використовується метод «createDocument», який наведено на рисунку 4.4.

```php
public function createDocument(array $embedding, string $originalText)
{
    $params = [
        'index' => $this->indexName,
        'body' => [
            'embedding' => $this->normalizeVector($embedding),
            'description' => $originalText,
            'created_at' => date( format: 'c', time()),
        ],
    ];

    try {
        return $this->client->index($params);
    } catch (ClientResponseException|ServerResponseException|Exception $e) {
        return $e->getMessage();
    }
}
```

Рисунок 4.4 – Створення документу в індексі Elasticsearch

Для цього необхідно вказати назву індексу, куди буде збережено документ, та передати нормалізований вектор, текстовий опис та час створення документу у «body».

Безпосередньо для пошуку на основі косинусної подібності по існуючим описам товарів використовується метод getRecommendations, код якого наведено на рисунку 4.5.

```php
public function getRecommendations(array $queryEmbedding, int $resultsCount = 10)
{
    $params = [
        'index' => $this->indexName,
        'body' => [
            'query' => [
                'script_score' => [
                    'query' => [
                        'match_all' => new \stdClass(),
                    ],

                    'script' => [
                        'source' => "cosineSimilarity(params.query_vector, 'embedding') + 1.0",
                        'params' => ['query_vector' => $this->normalizeVector($queryEmbedding)],
                    ],
                ],
            ],
            'size' => $resultsCount,
            '_source' => ['description'],
            'sort' => [
                '_score' => ['order' => 'desc'],
            ],
        ],
    ];

    $response = $this->client->search($params);

    return $response['hits']['hits'];
}
```

Рисунок 4.5 – Метод для отримання рекомендацій

Параметри методу:

– $queryEmbedding: вектор запиту, який буде порівнюватися з векторами в індексі Elasticsearch;

– $resultsCount: кількість рекомендацій, які потрібно повернути. За замовчуванням – 10.

Параметри запиту Elasticsearch.

1. Індекс, в якому буде проведений пошук ($this->indexName).
2. Тіло запиту, включаючи:

– «query»: запит на використання script_score, що дозволяє визначити релевантність на основі скрипту;

– «match_all»: пошук всіх документів (без обмежень);

– «script»: оголошення скрипту;

– «source»: скрипт для обрахування косинусної подібності;

– «params»: параметри, включаючи нормалізований вектор запиту;

– «size»: кількість результатів, які повертаються;

– «_source»: вказано, що потрібно повертати лише поле 'description' (оригінальний текстовий опис) з результатів;

– «sort»: вказано, що результати повинні бути відсортовані за зменшенням релевантності ('_score').

Далі запит надсилається до Elasticsearch, а отримані результати повертаються у вигляді масиву документів.

4.2 Експериментальна перевірка отриманих теоретичних результатів

Для порівняння ефективності пошуку було використано існуючий торгівельний майданчик Amazon, який займається продажом різноманітних

категорій товарів. Були порівняні результати пошуку з традиційними методами на Amazon та на модулі з удосконаленим векторним пошуком.

Пошук товарів для створення домашнього затишку з доволі детальним описом на Amazon повернув лише один товар – книгу (рисунок 4.6). Звісно, це не зовсім той результат, який на такий запит хотів би отримати користувач.

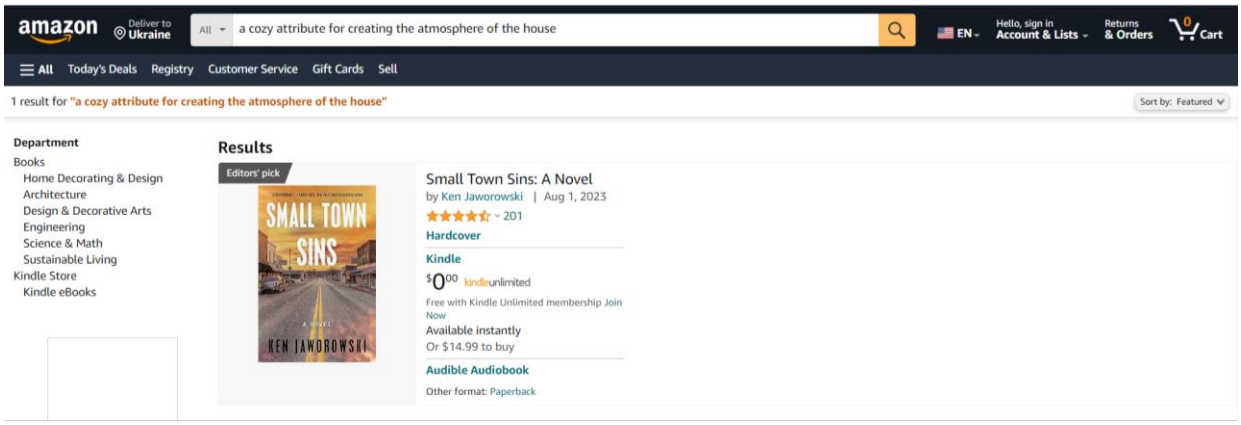


Рисунок 4.6 – Результати пошуку товарів для створення затишку на Amazon

В створеному модулі результат є значно релевантнішим (рисунок 4.7). Рекомендовані товари включають свічки, ковдри, ароматичні дифузори, подушки, светр та килимок.

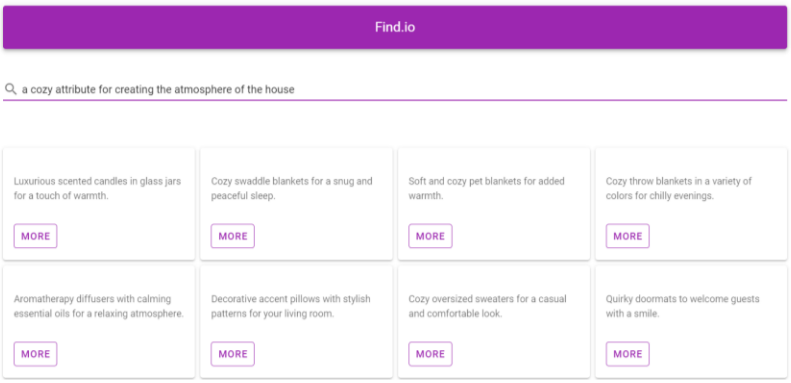


Рисунок 4.7 – Результати пошуку товарів для створення затишку в модулі векторного пошуку

Якщо замінити «cozy» на синонім «slug», то рекомендаційна система Amazon взагалі поверне порожній результат (рисунок 4.8) в той час, як результати векторного пошуку будуть ті ж самі за рахунок семантичної схожості цих слів (рисунок 4.9).

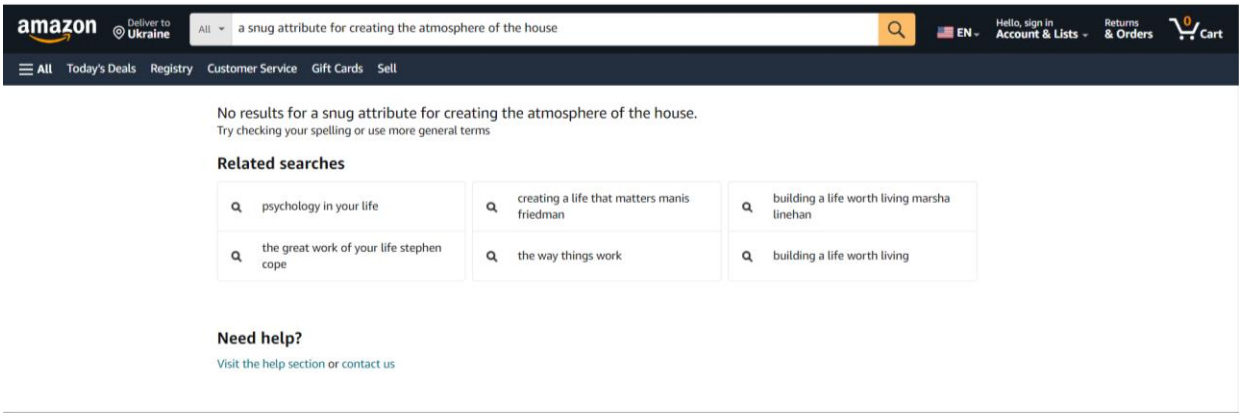


Рисунок 4.8 – Результати пошуку товарів із синонімом на Amazon

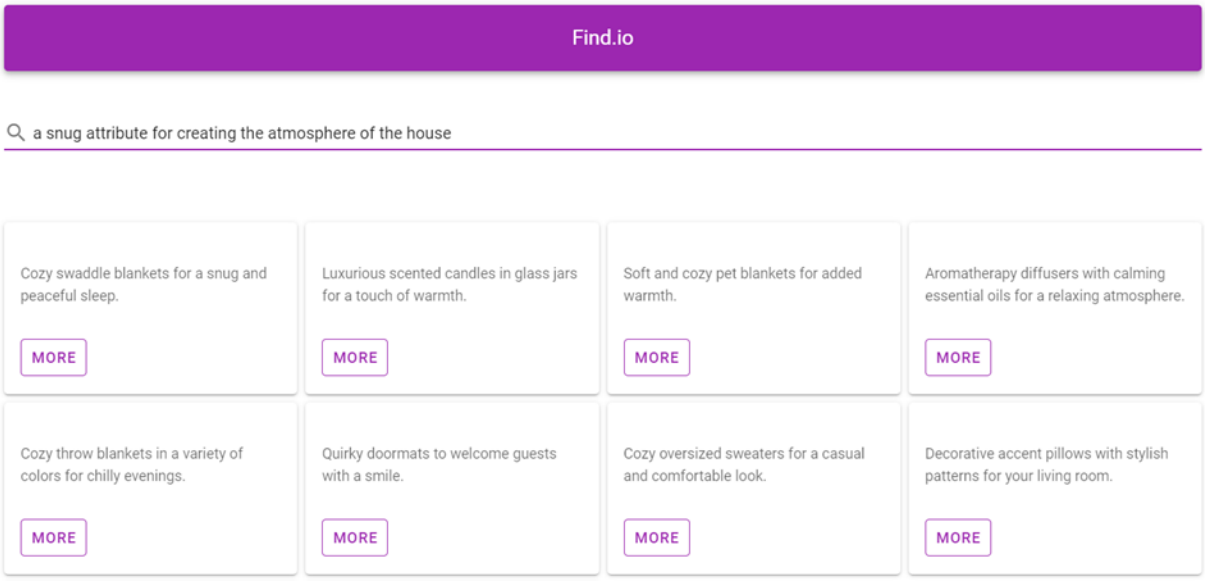


Рисунок 4.9 – Результати пошуку із синонімом в модулі векторного пошуку

Також було здійснено пошук комфортного взуття для ходіння та підкорення гір з амортизатором. Результат пошуку на Amazon при тому, що явно на майданчику доволі багато товарів, які могли б задовольнити споживача, повернув лише 3 товари, які наведено на рисунку 4.10. Скоріш за все, це не найкращі претенденти для купівлі користувачем.

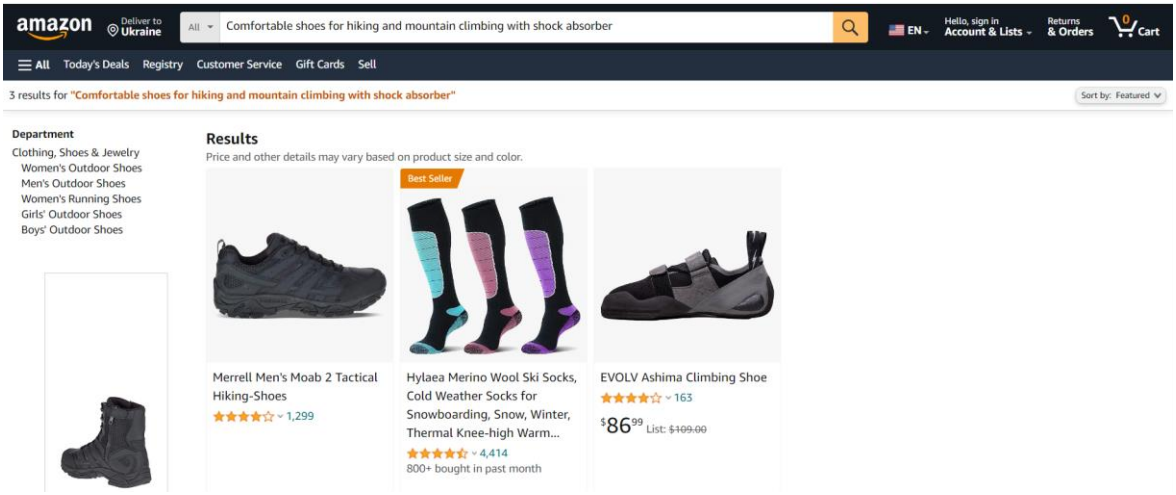


Рисунок 4.10 – Результати пошуку взуття на Amazon

При цьому модуль векторного пошуку спирався не на ключові слова, а на схожість семантики, завдяки чому отримані результати просто сортують взуття за ступенем зручності (рисунок 4.11).

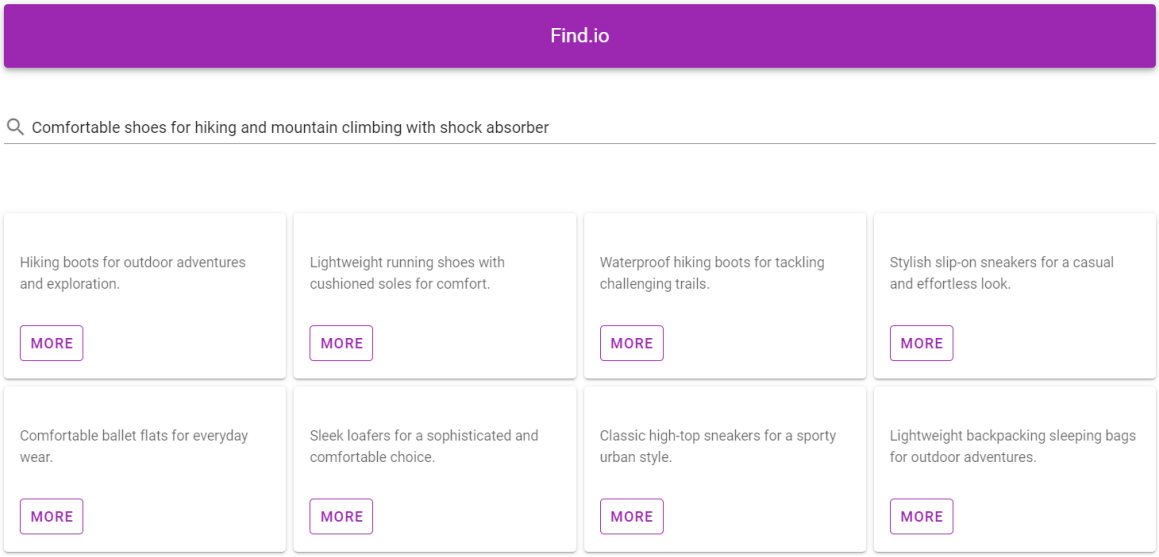


Рисунок 4.11 – Результати пошуку взуття за допомогою векторного пошуку

З отриманих результатів можна побачити, що релевантність наданих рекомендацій значно вище з використанням векторного пошуку. Користувач може максимально детально описувати свої вподобання і система поверне результати, відсортовані за семантичною схожістю термінів, що були використані. В той самий час видно, що торгівельний майданчик Amazon з величезною кількістю товарів, що обирає традиційні методи побудови рекомендацій, може не надати користувачу взагалі жодних варіантів для вибору, якщо запит буде занадто детальним і при цьому не міститиме в точності ті ключові слова, що є в описі.

Для порівняння впливу попередньої обробки на вартість запитів до OpenAI було створено більше ста описів товарів та розраховано кількість токенів за допомогою бібліотеки tiktoken для мови програмування Python. Скрипт для обрахування кількості токенів наведений на рисунку 4.12.

```python
def count_tokens(encoding, content):
    encoding = tiktoken.get_encoding(encoding)
    tokens = encoding.encode(content)
    return len(tokens)
```

Рисунок 4.12 – Код скрипту для розрахунку кількості токенів у фрагменті тексту

В результаті було сформовано csv файл з колонками:

– «original» - оригінальний фрагмент тексту;

– «original_tokens» - кількість токенів в оригінальному тексті;

– «processed» - текст, що було отримано після обробки;

– «processed_tokens» - кількість токенів тексту, що було отримано після обробки;

– «decrease in the number of tokens (pcs)» - кількість одиниць токенів, на які оброблений текст менший за оригінальний;

– «decrease in the number of tokens (%)» - відсоток, на який зменшилась кількість токенів після обробки.

Фрагмент результату експерименту наведено на рисунку 4.13.

original	original_tokens	processed	processed_tokens	decrease in the number of tokens (pcs)	decrease in the number of tokens (%)
Noise-canceling dishwashers for a quiet kitchen.	11	noise-cancel dishwash quiet kitchen	6	5	45.45
Decorative accent pillows with stylish patterns for your living room.	12	decor accent pillow stylish pattern live room	7	5	41.67
Cozy oversized sweaters for a casual and comfortable look.	12	cozi overs sweater casual comfort look	7	5	41.67
Lightweight running shoes with cushioned soles for comfort.	12	lightweight run shoe cushion sole comfort	7	5	41.67
GPS fitness trackers for monitoring and optimizing your workouts.	10	gp fit tracker monitor optim workout	6	4	40.00
Quilted puffer jackets to stay warm in the winter.	13	quilt puffer jacket stay warm winter	8	5	38.46
Comfortable ballet flats for everyday wear.	8	comfort ballet flat everyday wear	5	3	37.50
High-speed blenders for smoothies and food processing.	11	high-spe blender smoothi food process	7	4	36.36
Stylish denim jackets to complement your casual outfits.	11	stylish denim jacket complement casual outfit	7	4	36.36
Elegant pointed-toe flats for a polished finish.	11	eleg pointed-to flat polish finish	7	4	36.36
Expandable dining tables for hosting gatherings of any size.	11	expand dine tabl host gather size	7	4	36.36
Luxurious scented candles in glass jars for a touch of warmth.	14	luxuri scent candl glass jar touch warmth	9	5	35.71
High-quality bed sheets with a thread count for a restful sleep.	14	high-qual bed sheet thread count rest sleep	9	5	35.71
Elegant crystal vases â€“ perfect for displaying fresh flowers.	12	eleg crystal vase perfect display fresh flower	8	4	33.33
Robotic vacuum cleaners for effortless floor cleaning.	9	robot vacuum cleaner effortless floor clean	6	3	33.33
Ergonomic office chairs for comfortable and productive workdays.	12	ergonom offic chair comfort product workday	8	4	33.33
Outdoor lounge chairs for relaxing in the backyard.	9	outdoor loung chair relax backyard	6	3	33.33
Premium golf clubs for the avid golfer.	9	premium golf club avid golfer	6	3	33.33
Quirky doormats to welcome guests with a smile.	13	quirki doormat welcom guest smile	9	4	30.77
Stylish slip-on sneakers for a casual and effortless look.	13	stylish slip-on sneaker casual effortless look	9	4	30.77
Smart thermostats for optimal home temperature control.	10	smart thermostat optim home temperatur control	7	3	30.00
Classic tailored blazers for a polished professional look.	10	classic tailor blazer polish profession look	7	3	30.00
Folding dining sets for compact outdoor dining areas.	10	fold dine set compact outdoor dine area	7	3	30.00

Рисунок 4.13 – Результат попередньої обробки тексту

Як видно з результатів, за допомогою попередньої обробки тексту витрати можна скоротити в деяких випадках майже вдвічі, в деяких на 30%, але загалом це доволі значний результат за умов постійного використання OpenAI Embedding.

ВИСНОВКИ

В роботі було розглянуто сучасні підходи до рекомендацій та пошуку товарів, включаючи традиційні методи та методи, засновані на векторних представленнях, а також проаналізовано переваги та недоліки використання ВП в РС. Результат показав, що системи рекомендацій, засновані на ВП мають великі перспективи та є значно ефективнішими за традиційні методи пошуку та створення рекомендацій за рахунок можливості роботи з неструктурованими текстовими описами.

Ключовими перевагами текстових вбудовувань є семантичний пошук та можливість використовувати неструктуровані текстові описи, що зберігатиме час, який можна спеціалісти в даній предметній галузі зазвичай витрачають на налаштування БД, парсинг та структурування даних.

Вибір моделі вбудовування тексту є важливим етапом, і в даному випадку висвітлено вибір OpenAI GPT-3 Embedding API, зокрема text-embedding-ada-002. Ця модель демонструє високу якість та конкурентоздатність в порівнянні з іншими ведучими моделями, як показано на MTEB Leader Board та рейтингових показниках OpenAI.

Було розглянуто важливі аспекти використання векторних баз даних у сучасних технологіях, зокрема в контексті оптимізації пошуку векторів. Обрання Elasticsearch в якості векторної бази даних обґрунтовано його швидкістю, надійністю, безкоштовністю та наявністю додаткового стеку технологій Elastic Stack, що сприяє полегшенню використання Elasticsearch у різних сценаріях розробки.

Удосконалений метод створення персоналізованого переліку товарів, що базується на векторному пошуку та використанні векторних представлень

продуктів, було доповнено попередньою обробкою текстових даних, включаючи очищення тексту від спеціальних символів, токенізацію, приведення тексту до нижнього регістру та стемінг. Застосування такого підходу сприяє усуненню шуму опису, забезпечуючи краще відображення сутності тексту, а також сприяє зниженню вартості запитів, що надсилаються у запиті до моделі вбудовувань, за рахунок зменшення кількості токенів.

Структура вдосконаленого методу включає два ключові модулі: перший для створення векторних вбудовувань на основі даних про товар та текстового запиту, а другий для збереження цих вбудовувань в Elasticsearch та подальшого використання косинусної подібності у функції пошуку. Такий підхід дозволяє ефективно обробляти пошукові запити користувачів та надавати їм персоналізований перелік рекомендованих товарів.

Експериментальна перевірка якості пошуку із використанням створеного модуля показала його переваги у порівнянні із традиційними методами. Пошук за семантичною схожістю дозволяє рекомендувати товари, які при використанні традиційних методів просто не можуть потрапити до результатів пошуку через відсутність в описі ключових слів або занадто детальний опис. Розроблений метод дозволяє рекомендувати найбільш релевантні товари, що зможуть задовольнити споживачів якнайкраще. У роботі також експериментально доведено, що, з використанням попередньої обробки тексту, витрати на використання моделі OpenAI Embedding може бути суттєво скорочено за рахунок зменшення кількості токенів у запиті.

За результатами досліджень магістерської кваліфікаційної роботи підготовлено доповідь – «Shopping recommendation system design based on openai embeddings and elasticsearch» в рамках міжнародної науково-практичної конференції молодих вчених, аспірантів і студентів «Інформаційні технології в сучасному світі: дослідження молодих вчених».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Amer M. Text Embeddings Visually Explained. *Context by Cohere*. URL: https://txt.cohere.com/text-embeddings/ (date of access: 27.12.2023).
2. Embeddings – Open AI API. URL: https://platform.openai.com/docs/guides/embeddings/what-are-embeddings (date of access: 27.12.2023).
3. MTEB Leaderboard - a Hugging Face Space by mteb. *Hugging Face – The AI community building the future*. URL: https://huggingface.co/spaces/mteb/leaderboard (date of access: 27.12.2023).
4. Fanghua (Joshua) Yu | Text Embedding — What, Why and How? | Medium. URL: https://medium.com/@yu-joshua/text-embedding-what-why-and-how-13227e983ba7 (date of access: 27.12.2023).
5. The Beginner's Guide to Text Embeddings | deepset. *Enterprise ML/NLP Products and Solutions | deepset*. URL: https://www.deepset.ai/blog/the-beginners-guide-to-text-embeddings (date of access: 27.12.2023).
6. What is a Vector Database & How Does it Work? Use Cases + Examples | Pinecone. *Vector Database for Vector Search | Pinecone*. URL: https://www.pinecone.io/learn/vector-database/ (date of access: 27.12.2023).
7. Elastic Stack: Elasticsearch, Kibana, Beats & Logstash. *Elastic*. URL: https://www.elastic.co/elastic-stack (date of access: 27.12.2023).
8. Dense vector field type | Elasticsearch Guide [8.8] | Elastic. *Elasticsearch Platform – Find real-time answers at scale | Elastic*. URL: https://www.elastic.co/guide/en/elasticsearch/reference/8.8/dense-vector.html (date of access: 07.01.2024).

9. Types of Recommendation Systems & Their Use Cases [Електронний ресурс] // Types of Recommendation Systems & Their Use Cases - Режим доступу: https://medium.com/mlearning-ai/what-are-the-types-of-recommendation-systems-3487cbafa7c9, вільний.

10. Matrix Factorization For Recommendation Systems [Електронний ресурс] // Matrix Factorization For Recommendation Systems - Режим доступу: https://medium.com/@melih.kacaman/matrix-factorization-for-recommendation-systems-284f95c79ef6, вільний.

11. Comparing Different Vector Embeddings [Електронний ресурс] // Comparing Different Vector Embeddings - Режим доступу: https://thenewstack.io/comparing-different-vector-embeddings/, вільний.

12. Документація Weaviate [Електронний ресурс] // Документація Weaviate - Режим доступу: https://weaviate.io/developers/weaviate/quickstar, вільний.

13. Документація Vertex AI Vector Search [Електронний ресурс] // Документація Vertex AI Vector Search - Режим доступу: https://cloud.google.com/vertex-ai/docs/vector-search/overview, вільний.

14. Документація Generative AI [Електронний ресурс] // Документація Generative AI - Режим доступу: https://cloud.google.com/vertex-ai/docs/generative-ai/learn/overview, вільний.

15. Документація Azure Cognitive Search [Електронний ресурс] // Документація Azure Cognitive Search - Режим доступу: https://learn.microsoft.com/en-us/azure/search/search-what-is-azure-search, вільний.

16. Документація Azure OpenAI Embeddings models [Електронний ресурс] // Документація Azure OpenAI Embeddings models - Режим доступу:

https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models#embeddings-models, вільний.

17. Документація Milvus [Електронний ресурс] // Документація Milvus - Режим доступу: https://milvus.io/docs/overview.md, вільний.

18. Документація Towhee [Електронний ресурс] // Документація Towhee - Режим доступу: https://towhee.io/, вільний.

19. Документація ElasticSearch [Електронний ресурс] // Документація Towhee - Режим доступу: https://www.elastic.co/elasticsearch/vector-database, вільний.

20. Документація Amazon OpenSearch Service [Електронний ресурс] // Документація Amazon OpenSearch Service - Режим доступу: https://aws.amazon.com/blogs/big-data/amazon-opensearch-services-vector-database-capabilities-explained/, вільний.

21. Чалий, С. Ф., Лещинський, В. О., & Лещинська, І. О. (2018). Інтеграція локальних контекстів споживачів в рекомендаційних системах на основі відношень еквівалентності, схожості та сумісності. In Process mining Materials of the VII International Scientific Conference «Information-Control System and Technologies (pp. 142-144).

22. Chalyi, S., Leshchynskyi, V., & Leshchynska, I. (2020). Модель інтерфейсу пояснень з темпоральними параметрами в рекомендаційній системі. Системи управління, навігації та зв’язку. Збірник наукових праць, 2(60), 105-109. https://doi.org/https://doi.org/10.26906/SUNZ.2020.2.105

23. Чалий С. Ф. Темпоральні патерни вподобань користувачів в задачах формування пояснень в рекомендаційній системі / С. Ф. Чалий, В. О. Лещинський // Бионика интеллекта : научно-технический журнал. – 2020. – № 2 (95). – С. 21–27.

24. Чалий С. Ф. Моделювання пояснень щодо рекомендованого переліку об’єктів з урахуванням темпорального аспекту вибору користувача / Чалий С. Ф., Лещинський В. О., Лещинська І. О. // Системи управління, навігації та зв’язку. 2019. Т. 6. № 58. С. 97-101.

25. Чалий С. Ф. Ситуаційна модель користувацького вибору в рекомендаційній системі / С. Ф. Чалий, І. Б. Прібильнова // Системи управління, навігації та зв'язку. - 2019. - Вип. 2. - С. 159-163. - Режим доступу: http://nbuv.gov.ua/UJRN/suntz_2019_2_34.

26. Чалий С.Ф., Лещинський В.О., Лещинська І.О. (2018). Моделювання контексту в рекомендаційних системах. Проблеми інформаційних технологій, 1(023), 21-26.

27. Chalyi, S., Leshchynskyi, V., & Leshchynska, I. (2019). Концепція формування пояснень в рекомендаційних системах за принципом білого ящику. Системи управління, навігації та зв’язку. Збірник наукових праць, 3(55), 156-160.

28. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи (для студентів усіх форм навчання другого (магістерського) рівня програми "Інформаційні управляючі системи та технології") / Упоряд.:Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2021. – 30с.

29. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 22.06.2015. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

30. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складаня. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. – 20 с.