НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

«На правах рукопису»
УДК 004.89:[004.738.5:339](043.3)

До захисту допущено:
В.о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«___»_____2023 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

зі спеціальності 122 «Комп'ютерні науки»

на тему: «Побудова рекомендаційної системи з використанням підходів штучного інтелекту для e-commerce проекту»

Виконав:
студент ІІ курсу, групи КІ-21 мп
Фролкін Володимир Юрійович _____

Науковий керівник:
доцент кафедри ШІ, доктор філософії
Гуськова Віра Геннадіївна _____

Рецензент:
д.т.н., професор
Бідюк Петро Іванович _____

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

Київ – 2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«___»_____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Фролкіну Володимиру Юрійовичу

1. Тема дисертації «Побудова рекомендаційної системи з використанням підходів штучного інтелекту для e-commerce проекту», науковий керівник дисертації Гуськова Віра Геннадіївна, доцент, затверджені наказом по університету від «08» листопада 2023 р. №5200-с

2. Термін подання студентом дисертації 08.01.2024 р.

3. Об'єкт дослідження: процес створення рекомендаційної системи для вибору товарів на маркетплейсі bigl.ua.

4. Предмет дослідження: методи і алгоритми, які використовуються для створення систем персоналізації товарів.

5. Перелік завдань, які потрібно розробити:

a) здійснити огляд технічної літератури за темою роботи;
b) дослідити актуальність обраної теми;
c) ознайомитись із існуючими методами та підходами створення рекомендаційних систем
d) здійснити порівняльний аналіз наявних методів, виявити їх перевагита недоліки
e) розробити власну систему для надання рекомендацій

f) провести експеримент, що засвідчує працеспроможністьзапропонованої моделі, виконати аналіз результатів;
g) провести аналіз ринкових можливотсей запуску стартап проєкту;
h) розробити концептуальні висновки;
i) підготувати ілюстративний матеріал;
j) оформити пояснювальну записку.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: 24 таблиці, 17 ілюстрацій, презентація.

7. Орієнтовний перелік публікацій:

Заплановано апробування результатів роботи на ІІІ МНПК «Системи і технології зв'язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку», Київ, 2023.

8. Дата видачі завдання 30.08.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення літератури за темою роботи	01.09.2023–14.09.2023	Виконано
2.	Підготовка першого розділу.	14.09.2023 – 16.10.2023	Виконано
3.	Підготовка другого розділу.	16.10.2023 – 01.11.2023	Виконано
4.	Розробка програмного продукту.	01.11.2023– 15.12.2023	Виконано
5.	Підготовка третього розділу	15.12.2023 –21.12.2023	Виконано
6.	Підготовка частини стартап-проєкту	21.12.2023– 26.12.2023	Виконано
7.	Концептуальні висновки. Перспективи розвитку отриманих рішень	26.12.2023– 01.01.2024	Виконано
8.	Оформлення пояснювальної записки	01.01.2024– 07.01.2024	Виконано

Студент Фролкін В. Ю.

Науковий керівник Гуськова В. Г.

РЕФЕРАТ

Магістерська дисертація: 120 с., 17 рис., 24 табл., додаток, 16 посилань.

Тема магістерської дисертації: «Побудова рекомендаційної системи з використанням підходів штучного інтелекту для e-commerce проекту».

Мета роботи - дослідження методів та алгоритмів побудови рекомендацій, і підвищення їх ефективності та подальша інтеграція обраного алгоритму в існуючий маркетплейс.

Об’єкт дослідження - процес створення рекомендаційної системи для вибору товару на маркетплейсі Bigl.ua.

Предмет дослідження - методи і алгоритми, які використовуються для створення таких систем персоналізації товарів.

Для досягнення мети були поставленні такі задачі:

– огляд предметної області та аналіз існуючих рішень, архітектур нейромереж;
– розробка нових підходів на основі використання елементів глубокого навчання;
– розробка програмного комплексу, який забезпечуватиме просте використання розроблених методів для вирішення задачі побудови системи рекомендацій.

Програмний продукт реалізовано з використанням мови програмування Python який надає широкий спектр бібліотек для навчання нейронних мереж та аналізу даних.

Результати апробовано на МНТК.

НЕЙРОННІ МЕРЕЖІ, ЕМБЕДІНГ, РЕКОМЕНДАЦІЙНА СИСТЕМА, ТЕНЗОР, МАШИННЕ НАВЧАННЯ

ABSTRACT

Master's thesis: 120 pages, 17 figures, 24 tables, appendix, 16 references.

Master's thesis topic: "Building a recommendation system using artificial intelligence approaches for an e-commerce project."

The purpose of the work is to research methods and algorithms for building recommendations, and increase their effectiveness and further integration of the chosen algorithm into the existing marketplace.

The object of the study is the process of creating a recommendation system for product selection on the Bigl.ua marketplace.

The subject of research is the methods and algorithms used to create such systems of product personalization.

To achieve the goal, the following tasks were set:

– Overview of the subject area and analysis of existing solutions, neural network architectures;
– Development of new approaches based on the use of elements of deep learning;
– Development of a software complex that will ensure easy use of the developed methods to solve the problem of building a system of recommendations.

The software product is implemented using the Python programming language, which provides a wide range of libraries for training neural networks and data analysis.

The results were tested at the ISTC

NEURON NETWORKS, EMBEDDING, RECOMMENDATION SYSTEM, TENSOR, MACHINE LEARNING.

ЗМІСТ

ВСТУП

У сучасному цифровому світі, де обсяги інформації зростають з кожним днем, здатність ефективно обробляти та аналізувати дані стає критично важливою для успіху бізнесу, особливо в сфері електронної комерції (e-commerce). Рекомендації та поради були завжди важливі прийняття рішення людиною, і якщо раніше це був досвід, який накопичили інші люди, то сьогодні це ще й автоматизовані системи, які можуть виконувати ті самі функції, інколи навіть краще за саму людину. Завдяки розвитку технологій штучного інтелекту і машинного навчання, постала можливість створення інтелектуальних рекомендаційних систем, здатних аналізувати поведінку користувачів, їх переваги та потреби, пропонуючи їм релевантні товари та послуги. Рекомендаційні системи допомагають орієнтуватися у складному інформаційному ландшафті, виділяючи релевантні дані та пропонуючи користувачам те, що може відповідати їхнім інтересам та потребам.

Це, в свою чергу, не тільки підвищує ефективність пошуку і вибору продукції або контенту, але й сприяє формуванню більш особистісного і змістовного досвіду взаємодії з технологіями. Рекомендації можуть стимулювати відкриття нових інтересів та захоплень, допомагаючи користувачам вийти за межі звичного і знайти щось нове і надихаюче. Вони відіграють ключову роль у формуванні інформаційного середовища, яке адаптується до індивідуальних особливостей кожного користувача, забезпечуючи більш персоналізоване та емоційно значуще спілкування з цифровим світом.

Такі системи не тільки впливають на економічні показники та бізнес-процеси, але й мають глибокий вплив на культуру споживання та взаємодії людей з технологіями. Вони стають агентами змін у способі, яким ми вибираємо, сприймаємо та взаємодіємо з інформаційним світом, формуючи нові патерни поведінки та споживання. Враховуючи це, розуміння та оптимізація

рекомендаційних систем набувають особливого значення, адже вони впливають не тільки на економічну ефективність, але й на якість життя та добробут індивідів.

Ця дисертація присвячена розробці та впровадженню рекомендаційної системи для e-commerce проекту з використанням сучасних підходів ШІ. Попри значний прогрес в області рекомендаційних систем, їхнє створення та оптимізація залишаються складними завданнями через необхідність врахування великої кількості факторів, таких як точність рекомендацій, їхня релевантність та персоналізація. Додатковим викликом є потреба забезпечення масштабованості та високої продуктивності системи, щоб вона могла ефективно працювати з великими обсягами даних і великою кількістю користувачів.

Ця робота має на меті дослідити та розробити рекомендаційну систему, засновану на передових методах машинного навчання та аналізу даних, для забезпечення персоналізованих рекомендацій користувачам e-commerce платформи. Ми розглянемо різні підходи до побудови рекомендаційних систем, включаючи колаборативну фільтрацію, контент-орієнтовані методи та гібридні моделі, а також вивчимо, як впровадження алгоритмів ШІ може покращити якість та ефективність рекомендацій.

Враховуючи все вищесказане, дана дисертація ставить собі за мету не тільки розробити та впровадити ефективну рекомендаційну систему, але й дослідити вплив різних факторів на її продуктивність, а також розробити рекомендації щодо оптимізації системи для конкретних потреб e-commerce проекту. Таким чином, результати цієї роботи можуть стати важливим вкладом у розвиток технологій рекомендаційних систем, а також забезпечити практичні напрямки для їх впровадження в реальних бізнес-процесах.

РОЗДІЛ 1 ОГЛЯД СУЧАСНИХ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

У цьому розділі буде будуть розглянуті проблеми та потреби обраної галузі, які постають перед розробниками рекомендаційних систем. Також буде розглянуто та проаналізовано різні існуючі методи для побудови подібних моделей.

1.1 Актуальність дослідження

За останні десятиліття рекомендаційні системи стали невід'ємною частиною багатьох онлайн-платформ, від великих роздрібних магазинів, як Amazon, до стрімінгових сервісів, таких як Netflix. Їхня здатність аналізувати великі обсяги даних та визначати індивідуальні переваги користувачів дозволяє підвищувати рівень користувацького досвіду, збільшуючи задоволеність клієнтів і, відповідно, їх лояльність. У цьому контексті актуальність дослідження рекомендаційних систем стає важливим завданням, спрямованим на розуміння, оптимізацію та вдосконалення існуючих алгоритмів та підходів.

Персоналізація стала однією з основних тенденцій у сфері електронної комерції, і рекомендаційні системи відіграють вирішальну роль у її реалізації. Здатність точно прогнозувати вподобання та смаки користувачів і пропонувати товари, послуги чи сервіси, які відповідають їхнім інтересам, безпосередньо впливає на показники конверсії та загальну ефективність бізнес-моделі того чи іншого проекту. Дослідження в області рекомендаційних систем є дуже актуальним та затребуваним в наші часи, що доводить більша кількість статей, конференцій та змагань на дану тематику. Дослідження в області рекомендаційних систем відкриває нові можливості для підвищення точності та

релевантності рекомендацій, сприяючи подальшому розвитку персоналізованого підходу у електронній комерції.

Незважаючи на значні успіхи в області розробки рекомендаційних систем, все ще існує ряд викликів і проблем, які потребують подальшого дослідження. До них відносяться питання точності, прозорості, врахування контексту, доменої області, потреб та задач, а також етичні аспекти, пов'язані з приватністю та безпекою даних. Розробка більш точних, зрозумілих та безпечних рекомендаційних систем вимагає комплексного підходу та вивчення різноманітних аспектів, від алгоритмічних рішень до соціальних і етичних питань.

Розвиток технологій штучного інтелекту та машинного навчання відкриває нові горизонти для вдосконалення рекомендаційних систем. Впровадження глибокого навчання, обробка природної мови та розвиток інших передових технологій дозволяють створювати більш точні, ефективні та адаптивні системи рекомендацій. Дослідження в цій області має важливе значення для підтримки інноваційного розвитку та використання потенціалу ШІ в електронній комерції.

Ще одним доказом актуальності даної теми може слугувати успішні кейси розробки та використання рекомендаційних систем у вже готові продукти. Amazon є одним з найбільш яскравих прикладів успішного впровадження рекомендаційних систем у бізнес-процеси. Компанія використовує складні алгоритми машинного навчання для аналізу історії покупок, переглядів товарів та відгуків користувачів з метою надання персоналізованих рекомендацій. Це не тільки підвищує задоволеність клієнтів, а й значно збільшує продажі. За оцінками експертів, рекомендаційна система Amazon сприяє приблизно 35% всіх продажів компанії.

Рекомендаційні системи є ключовим елементом сучасної електронної комерції, впливаючи на взаємодію користувачів з платформами, збільшуючи продажі та підвищуючи задоволеність клієнтів. Актуальність дослідження в цій області обумовлена зростаючим впливом рекомендаційних систем на бізнес-процес

1.2 Аналіз та огляд предметної області

Історія рекомендаційних систем розпочалася в середині 1990-х років, коли інтернет став широко доступним для загального користування. Перші рекомендаційні системи базувалися на методах колаборативної фільтрації, які аналізували поведінку користувачів та рекомендували продукти на основі схожості між ними. Знаменитим прикладом цього періоду є система "Tapestry", розроблена в Xerox PARC, яка дозволяла користувачам фільтрувати електронні листи та новини на основі оцінок і коментарів інших користувачів, тобто він базувався на підході колаборативної фільтрації. Колаборативна фільтрація — це метод, який базується на ідеї, що користувачі, які згодилися в минулому, згодяться і в майбутньому щодо своїх переваг до певних об'єктів. Іншими словами, якщо користувач А має ті ж самі вподобання, що й користувач Б в одній певній ситуації, А ймовірно матиме ті ж самі вподобання, що й Б в іншій ситуації.

З початку 2000-х років розробники зрозуміли, що результати, які дають алгоритми колаборативна фільтрація, вже не покращуються, тому у рекомендаційних системах почали використовувати складніші алгоритми машинного навчання та штучного інтелекту, включаючи методи засновані на вмісті, які дозволяли знаходити схожості між конкретними елементами, а також почались дослідження з метою зрозуміти причину користувацьких побажань, а саме: чому користувач вибирає конкретний продукт із тієї чи іншої групи та чом вони схожі. Слід також згадати появу гібридних моделей, які комбінують різні типи даних та алгоритми для покращення якості рекомендацій, тому у розробників з’явилась можливість рекомендувати товари та послуги для геть нових користувачів, які навіть не мають історії користування певним сервісом, так і для тих, хто вже є постійним клієнтом. Також великий вклад вніс розвиток глибокого навчання. Такі компанії, як Amazon та Netflix, інвестували значні

ресурси в розвиток цих технологій, що призвело до значного покращення якості рекомендацій та користувацького досвіду їх продуктів.

Зі ростом обсягів даних та поширенням інтернету речей, рекомендаційні системи почали використовувати великі дані для більш точного прогнозування користувацьких уподобань. Також прийшло розуміння того, що в нинішніх реаліях не існує універсального алгоритма чи їх сукупності, і для кожної сфери та конкретного сервісу потрібне своє дослідження, але жоден алгоритм не зможе задовільнити всі потреби корстувача. Актуальним стало й питання приватності та безпеки даних, оскільки для роботи рекомендаційних систем необхідно збирати та аналізувати великі обсяги особистої інформації.

На сьогоднішній день рекомендаційні системи є невід'ємною частиною багатьох онлайн-сервісів, від електронної комерції та стрімінгових сервісів до соціальних мереж. Вони допомагають компаніям підвищувати ефективність, покращувати користувацький досвід і збільшувати доходи. Однак разом з тим, вони ставлять перед дослідниками та розробниками нові виклики, пов'язані з етикою використання даних, захистом приватності та розвитком більш ефективних і точних алгоритмів.

Сама ж предметна область рекомендаційних систем охоплює широкий спектр тем і питань, від технічних аспектів розробки алгоритмів до соціальних та етичних питань їх використання. Важливі напрямки дослідження включають в себе покращення точності та віддачі рекомендацій, розробку методів захисту приватності, вивчення впливу рекомендацій на суспільство та розвиток нових підходів до персоналізації. В цьому контексті, історія та сучасний стан рекомендаційних систем надають багатий матеріал для аналізу та дослідження, спрямований на подальший розвиток цієї важливої області.

1.3 Рекомендаційні системи в сфері електронної комерції

Підходи до персоналізації сьогодні є фундаментальним компонентом ефективних інтернет-рекламних кампаній. Клієнти давно звикли, що бренди, які вони вважають кращими та улюбленими, надають їм контент, який відповідає їхнім інтересам. Таким чином, персоналізовані рекомендації товарів не тільки сприяють зростанню продажів, але й стимулюють розвиток відносин між клієнтами та брендом.

Одним із основних інструментів, що використовується в інтернет-маркетингу для створення пропозицій, які відповідають уподобанням користувача, є веб-трекінг. Цей інструмент дозволяє збирати дані про поведінку кожного відвідувача веб-сайту за допомогою спеціального скрипту.

Проте для того, щоб точно націлювати веб-персоналізацію, необхідно збирати інформацію про взаємодію клієнта з брендом на всіх рівнях, як в інтернеті, так і за його межами. Це особливо важливо, оскільки, наприклад, якщо клієнт купив телевізор в звичайному магазині, рекомендаційна система повинна враховувати цю покупку, щоб не пропонувати товар, який вже був придбаний.

Доказом збільшення конверсії та прибутку загалом можуть слугувати наступні дані різних українських інтернет-магазинів та маркетплейсів:

- магазин "Антошка" дослідив, що їх персоналізовані email розсилки мають на 70% більшу конверсію, ніж неперсоналізовані
- колишній сервіс доставки “Rocket” збільшив к-ть своїх замовлень на 65% після впровадження персоналізованих рекомендацій
- маркетплейс “Shafa” тільки за 6 місяців використання рекомендаційної системи зміг збільшити кількість своїх замовлень у 2 рази
- “Foxtrot” після тесту персональних рекомендацій у категорії аксесуарів, дійшли висновку, що кількість успішних покупок зросла на 16%

Також за дослідженнями інших сервісів:

- за результатами опитування Barilliance 2018 року, товарні рекомендації становлять до 30% прибутків у сфері електронної комерції, а 12% товарів купють саме через них;
- дослідження Salesforce показує, що замовлення товарів, на які користувач потрапив через рекомендацію, становлять близько 25% та генерують 26% загального прибутку;
- коефіцієнт конверсії серед відвідувачів, які переходять на товарні рекомендації, у 5.5 раз вищий, ніж у тих, хто не переходив;
- у звіті Accenture стверджується, що персоналізація збільшує ймовірність купівлі на 75 відсотків.

Рекомендації товарів в сфері електронної комерції представляють собою картки товарів, які показуються користувачам на сайті або у мобільному додатку, або у електронних листах чи інших повідомленнях. Ці рекомендації генеруються за допомогою спеціального програмного забезпечення, яке аналізує онлайн-поведінку користувача та збирає інформацію про його покупки поза мережею. За допомогою цих даних алгоритм визначає продукти, які можуть виявитися привабливими для клієнта. Якість рекомендацій безпосередньо залежить від розширеності та ефективності програмного забезпечення, що, у свою чергу, позитивно впливає на обсяги продажів.

Товарні рекомендації можуть застосовуватись по-різному:

- на різних сторінках сайту чи мобільного додатку: головна сторінка, сторінка товару, 404-сторінка та інші;
- у різновидах розсилок–рекламні, промо, після покупки товару.

Далі розглянемо основні типи товарних рекомендацій:

Блок “Вас може зацікавити”. У цьому блоці виводять продукцію, яка найкраще підходить під поведінковий профіль споживача, враховуючи аспекти, як-от категорію продукції, марку, ціну та інші параметри (рис. 1.1). На веб-сайті рекомендується розміщувати такі рекомендації безпосередньо на сторінці товару. Крім того, цю функціональність можна інтегрувати в автоматизовані

розсилки, що повідомляють користувачів про товари, які вони переглядали, але не придбали, що допомагає залучити їхню увагу до більшого асортименту продукції в їхніх обранці категоріях.

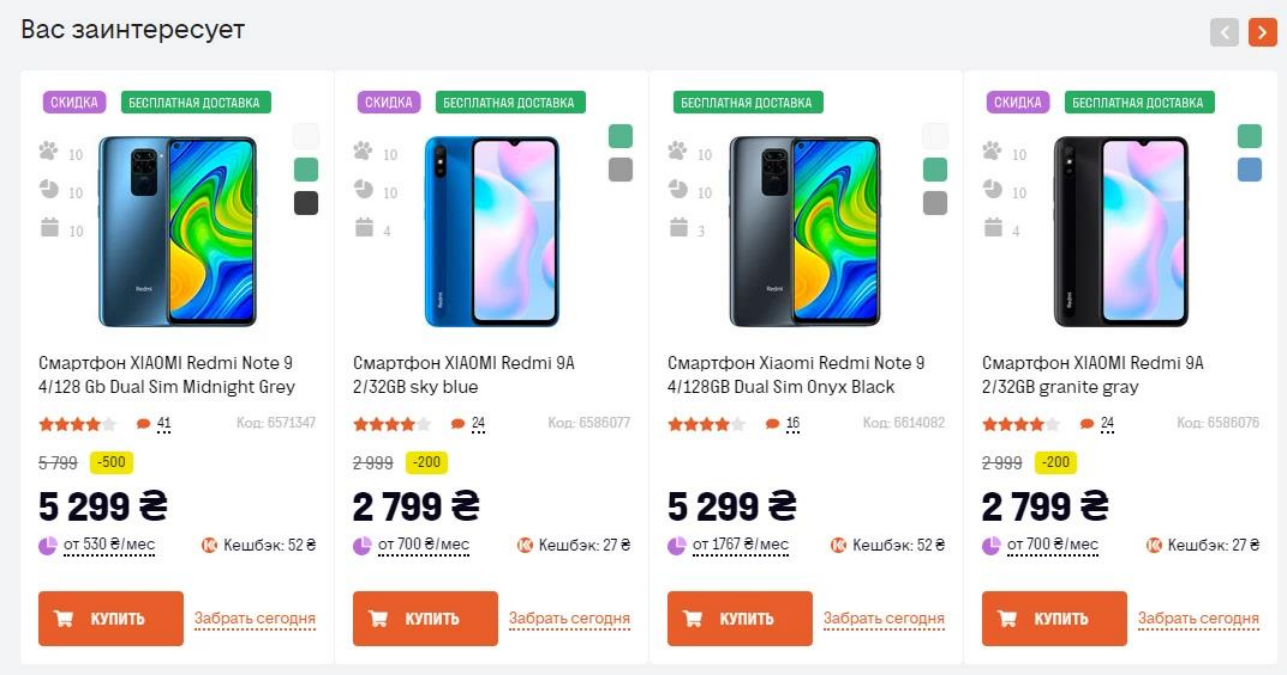

Рисунок 1.1 – Блок “Вас може зацікавити” на веб-ресурсі https://www.foxtrot.com.ua/

Блок “Разом з цим товаром купують”. Це зазвичай якийсь додаткових корисний товар до того, який клієнт має намір придбати. Такі рекомендації виявляються ефективними на фінальній стадії процесу покупки, і їх найкраще демонструвати на сторінці з кошиком чи оформленням замовлення на веб-сайті та у листах, які надсилаються клієнту після його покупки (рис. 1.2).

Блок “Вигідна пропозиція”. У таких блоках підбираються товари, які зараз продаються за зниженою ціною або мають великий попит серед покупців, з категорій, які зацікавили користувача. Розміщення такої колекції на сторінці помилки 404 може значно зменшити рівень невдоволення відвідувача від потрапляння на сторінку, якої не існує, і імовірно сприятиме збільшенню продажів товарів із цих акцій. Також такі рекомендації можуть бути використані

в розсилках, пов’язаних із товарими, які користувач переглядав, але так і не купив (рис. 1.3).

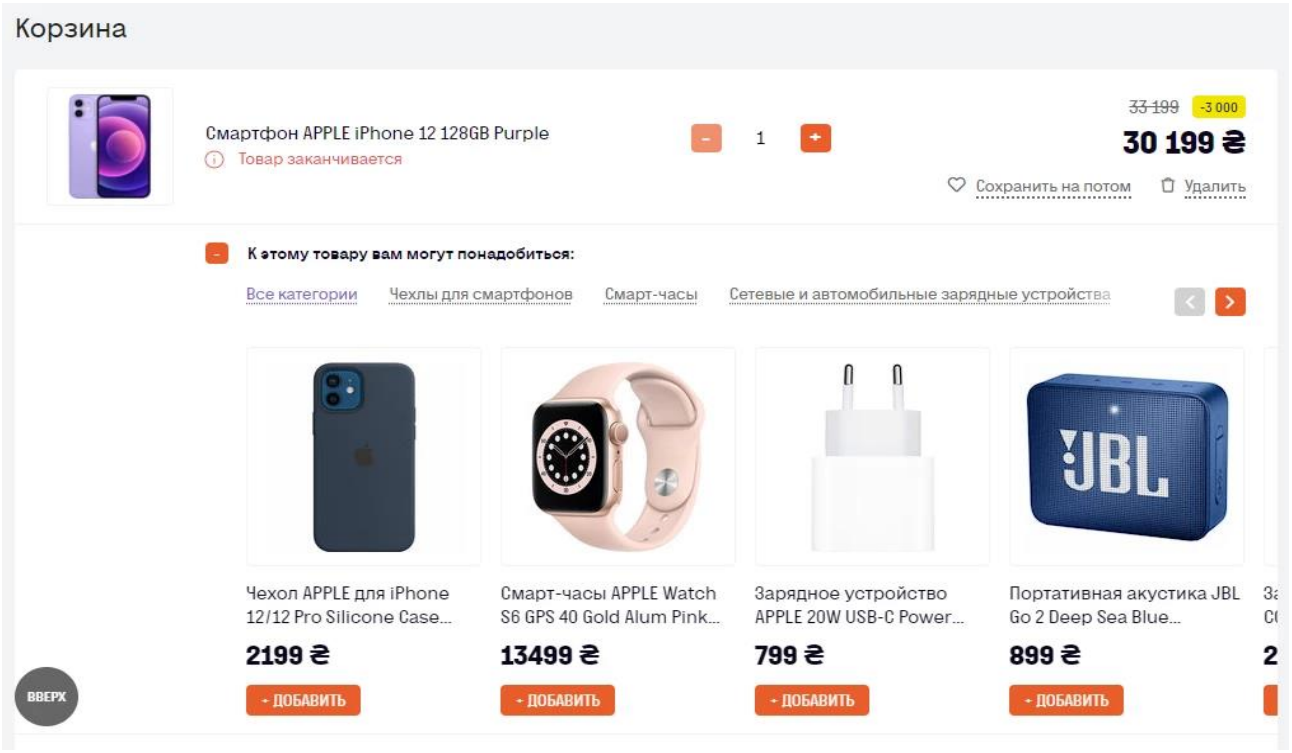

Рисунок 1.2 - Блок “Разом з цим товаром купують” на веб-ресурсі https://www.foxtrot.com.ua/

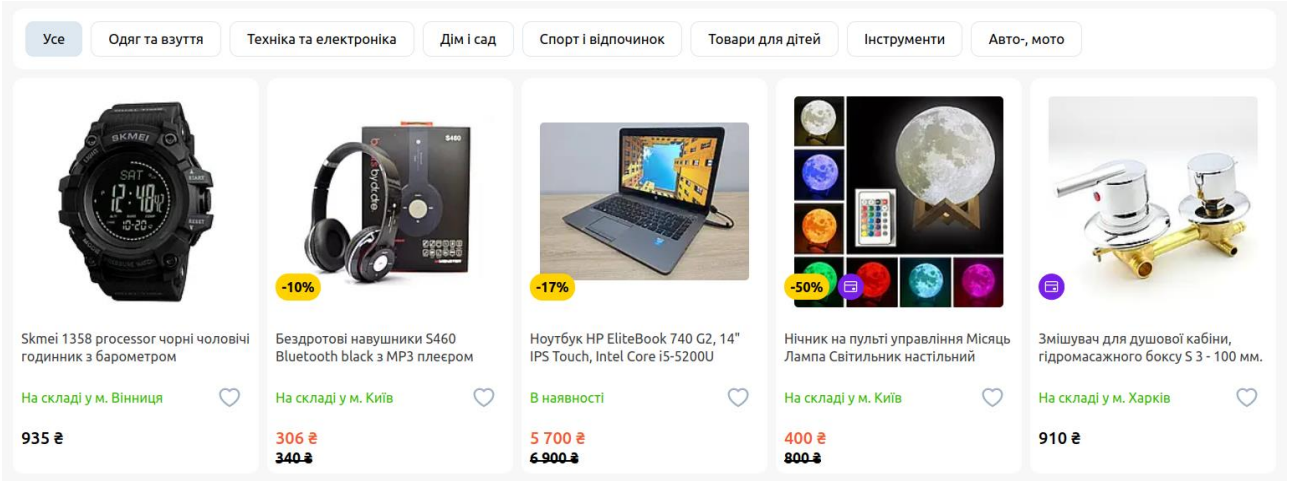

Рисунок 1.3 – Блок “Вигідна пропозиція” на веб-ресурсі https://bigl.ua/

Блок “Новинки з категорії”. Клієнту, який так і не придбав товар, надсилається лист із вибраною колекцією останніх новинок у категорії продукції, яка його зацікавила.

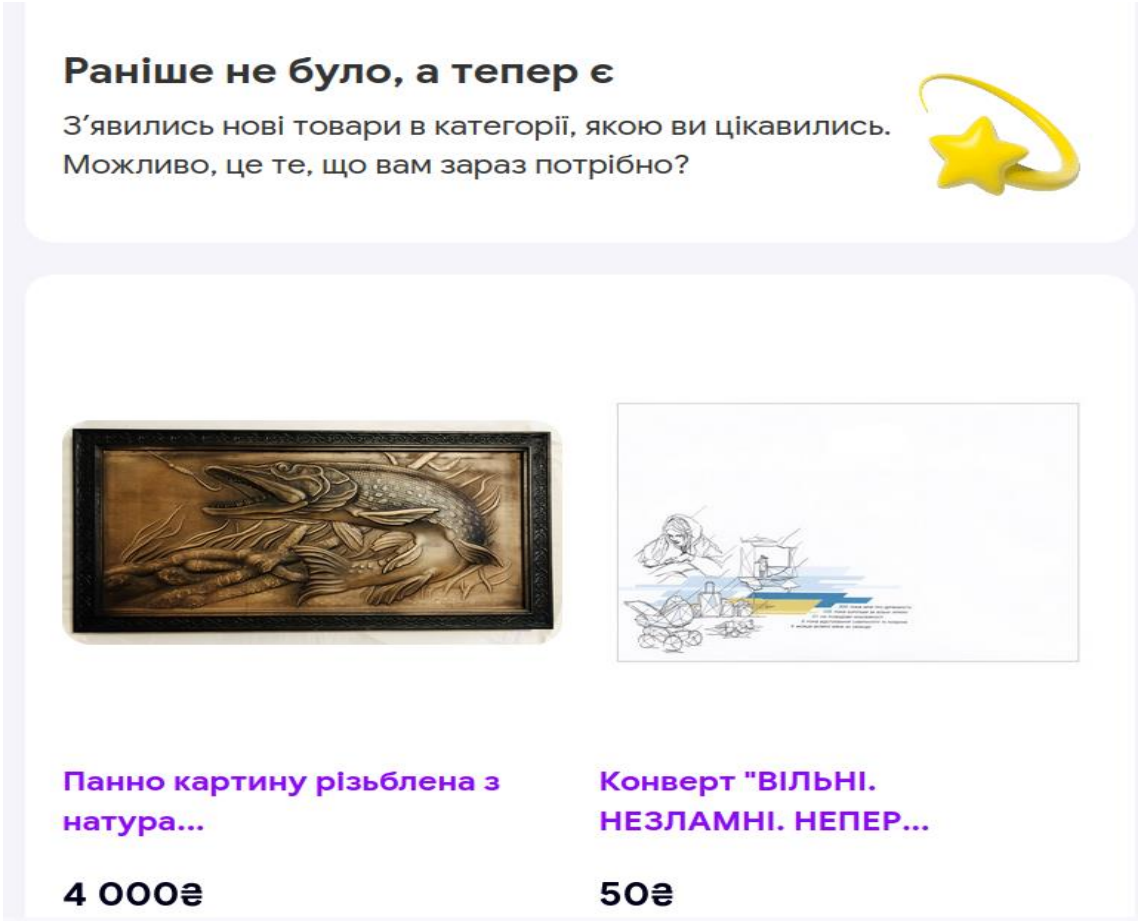

Рисунок 1.4 – Блок “Новинки з категорії” на порталі https://prom.ua/

Блок “Зниження ціни на подібні товари”. Цю категорію пропозицій можна ефективно втілити у розсилки. Наприклад, якщо користувач переглядав певні товари чи залишив їх у своєму кошику і протягом тижня не здійснив покупки, йому може бути відправлено електронний лист зі спеціальними пропозиціями і знижками на аналогічні продукти (рис. 1.5).

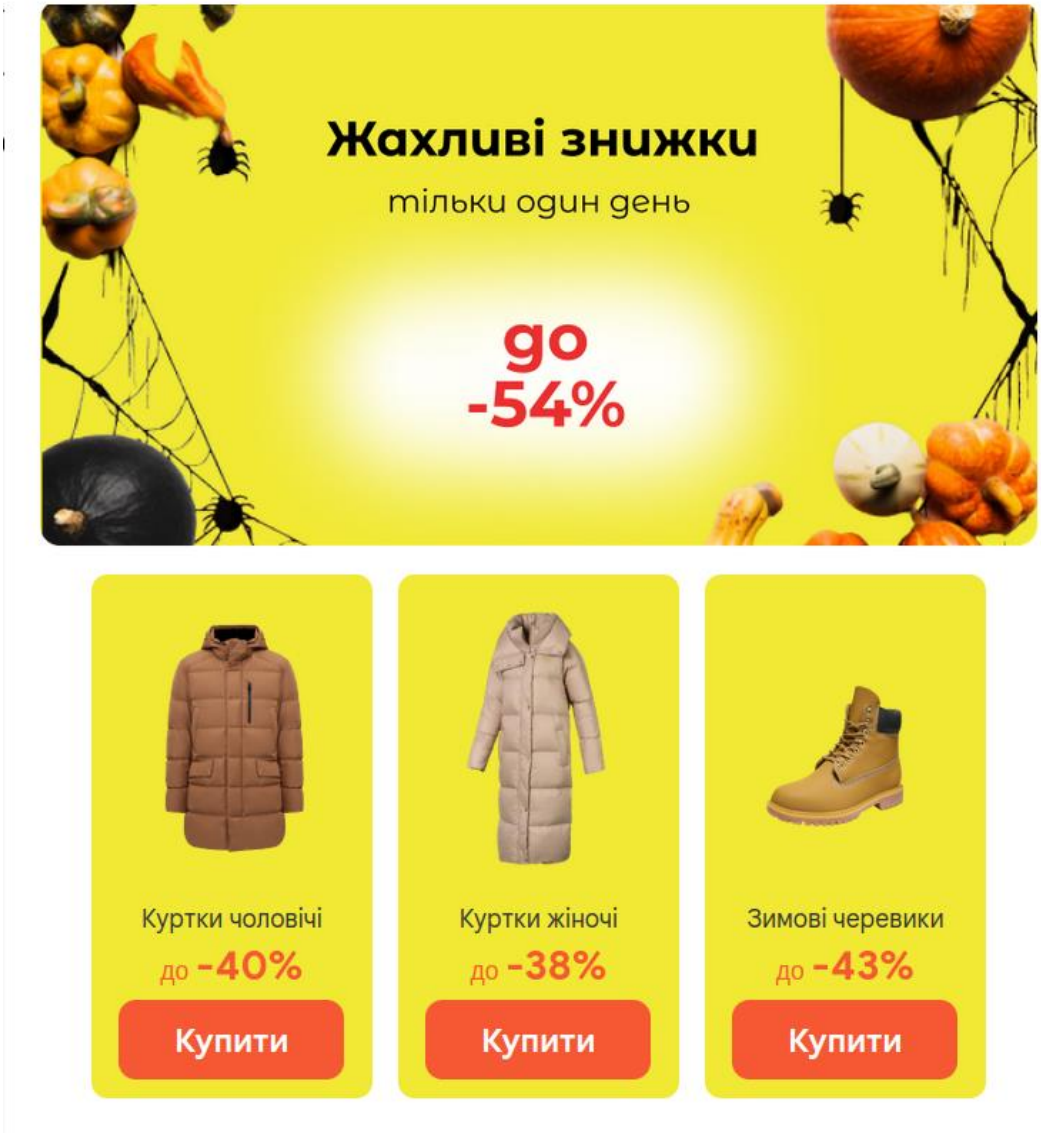

Рисунок 1.5 – Лист “Знижка на подібні товари” від веб-ресурсу https://bigl.ua/

1.4 Аналіз існуючих методів персоналізації

Загалом, рекомендації товарів можна класифікувати на дві основні категорії: персоналізовані та неперсоналізовані.

Неперсоналізовані рекомендації є досить базовими та простими, і часто вони не досягають поставленої мети впровадження. В рамках цього підходу користувачам пропонуються або найбільш популярні продукти, або ті, які

найкраще відповідають бізнес-задачам. Як можна припустити, такі рекомендації в основному використовуються як частина більш комплексних систем рекомендацій, які включають в себе кілька алгоритмів.

Персоналізовані рекомендації, в свою чергу, поділяються на три види: Контентна фільтрація (Content-based), Спільна фільтрація (Collaborative Filtering) та Гібридні методи рекомендаційних систем, які включають у себе поєднання двох або трьох зазначених підходів.

Collaborative Filtering (CF) є одним із фундаментальних підходів в системах рекомендацій, який базується на аналізі поведінки та вподобань користувачів з метою прогнозування їхніх інтересів та рекомендації відповідних продуктів чи послуг. Підход CF виходить з припущення, що якщо користувачі виявляли схожість у своїх оцінках та вподобаннях в минулому, то вони продовжать робити це і в майбутньому. Цей метод може бути поділений на дві основні категорії: Memory-Based CF та Model-Based CF.

Memory-Based CF використовує всі доступні дані для визначення схожості між користувачами або предметами, в той час як Model-Based CF застосовує статистичні та машинні методи навчання для прогнозування вподобань користувача. Ці рекомендаційні системи вимагають обширного набору вхідних даних для адекватного функціонування; при їх відсутності поради, які вони надають, ризикують бути помилковими чи некоректними. Основними викликами для CF є проблеми холодного старту, де важко зробити рекомендації для нових користувачів чи предметів, та масштабування, оскільки обчислювальна складність зростає з кількістю користувачів та предметів. Розв'язання цих проблем вимагає інноваційних підходів та методів оптимізації, які продовжують бути предметом інтенсивних досліджень у галузі рекомендаційних систем.

Content-based підходи в рекомендаційних системах розглядаються як ключові засоби персоналізації, оскільки вони базуються на аналізі вмісту об'єктів, які користувачі оцінили високо, і використовують цю інформацію для прогнозування їх вподобань до інших об'єктів. Цей метод вимагає ретельного

аналізу атрибутів товарів чи послуг, які користувачі обирають, а також створення докладного профілю користувача, що відображає його інтереси та переваги. Перевагою цього підходу є його здатність рекомендувати об'єкти, незалежно від їх популярності серед інших користувачів, та забезпечення високої релевантності рекомендацій. Однак, він також має свої недоліки, такі як труднощі з розпізнаванням нових інтересів користувача, оскільки система орієнтується лише на його попередню поведінку, а також схильність до "ефекту фільтраційної бульбашки", коли користувачу пропонуються лише ті товари чи послуги, які строго відповідають його існуючим вподобанням. Також існує виклик, пов'язаний із холодним стартом: методи, засновані на контенті, досягають ефективності тільки після того, як користувач висловив свою думку щодо достатньої кількості об'єктів.

Дослідження і розвиток методів подолання цих обмежень та покращення точності та діапазону рекомендацій, заснованих на вмісті, є актуальними напрямками в галузі рекомендаційних систем.

Гібридні методи в рекомендаційних системах являють собою інноваційний підхід, що поєднує в собі переваги різних типів рекомендаційних алгоритмів з метою покращення точності та релевантності рекомендацій. Ці методи інтегрують в себе підходи, засновані на вмісті, колаборативні фільтри та інші техніки, щоб скомпенсувати недоліки кожного окремого методу та забезпечити більш повне і точне уявлення про переваги користувача. Важливість такого підходу полягає в його здатності обробляти велику кількість даних з різних джерел, включаючи поведінкові дані користувача, описи товарів чи послуг, а також соціальні взаємодії, що дозволяє системі розробляти більш персоналізовані та точні рекомендації.

Злиття різних методологій в рамках гібридної системи рекомендацій дозволяє ефективно протидіяти недолікам кожного окремого підходу. Є кілька основних способів комбінування:

– незалежна реалізація колаборативного та контентного фільтрування і подальше їхнє об'єднання;

– інтеграція певних контентних характеристик в рамки колаборативної техніки;
– додавання колаборативних елементів до методів, заснованих на вмістіl
– створення єдиної моделі, яка враховує правила обох підходів, такий підхід дозволяє створити більш збалансовану та адаптивну систему рекомендацій.

Гібридні системи також ефективні для вирішення проблеми холодного старту, оскільки вони можуть використовувати інформацію з різних джерел для формування перших рекомендацій новим користувачам. Враховуючи широкий спектр можливостей, що відкривають гібридні системи, вони вважаються одним із найперспективніших напрямків розвитку рекомендаційних систем, що вимагає подальших досліджень і розробок.

1.5 Аналіз існуючих продуктів на ринку

Розглянемо існуючі онлайн-сервісів, які інтегрували рекомендаційні системи в свою структуру для покращення якості обслуговування та підвищення задоволеності користувачів. Рекомендаційні системи грають важливу роль в сучасному цифровому середовищі, де інформація переважає, і користувачі часто потребують допомоги для виявлення товарів, послуг чи контенту, який би відповідав їхнім потребам і вподобанням. Нижче представлені деякі з них:

Spotify ефективно використовує рекомендаційні системи для покращення досвіду користувачів та підтримки високого рівня особистісної взаємодії. Сервіс поєднує різні підходи, включаючи колаборативне фільтрування, аналіз контенту та обробку природної мови, щоб забезпечити персоналізовані плейлисти, рекомендації треків та відкриття нової музики. Алгоритми Spotify враховують історію прослуховування, популярність треків, жанри та артистів, а також індивідуальні вподобання користувачів. Це дозволяє створювати унікальний

музичний досвід для кожного користувача, підвищуючи залученість і стимулюючи більше часу проведеного в додатку. Однією з найвідоміших функцій є “Discover Weekly”, що щотижня надає користувачам новий плейлист, який складається з треків, вибраних на основі їх попередніх прослуховувань та прослуховувань інших користувачів з подібними музичними вподобаннями. Завдяки постійному аналізу даних та машинному навчанню, Spotify забезпечує високий рівень персоналізації, покращуючи задоволеність користувачів і підтримуючи свою конкурентоспроможність на ринку стрімінгових сервісів.

Netflix відомий своєю інноваційною системою рекомендацій, яка відіграє ключову роль у забезпеченні персоналізованого досвіду для його користувачів. Використовуючи складні алгоритми машинного навчання та методи обробки великих даних, Netflix аналізує широкий спектр інформації, включаючи історію переглядів, пошукові запити, час перегляду, а також рейтинги і відгуки користувачів. Ці дані допомагають сервісу розуміти індивідуальні вподобання та інтереси кожного користувача, дозволяючи формувати персоналізовані списки рекомендацій для фільмів, серіалів та інших відеоматеріалів. Одним з ключових елементів успіху рекомендаційної системи Netflix є її здатність до самонавчання та адаптації. Система постійно вивчає поведінку користувачів, враховує зміни у їхніх перевагах та відповідно до цього коригує рекомендації. Це дозволяє Netflix забезпечувати актуальний і релевантний контент, підвищуючи задоволеність користувачів та збільшуючи час, який вони проводять на платформі. Крім того, Netflix використовує рекомендації для оптимізації своєї контентної стратегії, аналізуючи тенденції та попит на певні жанри або серії. Ця інформація допомагає компанії приймати обґрунтовані рішення щодо закупівлі нового контенту та розробки власних оригінальних продукцій. Таким чином, рекомендаційні системи не тільки підвищують лояльність користувачів, але й сприяють ефективному розвитку бізнесу.

Amazon є одним з піонерів у впровадженні рекомендаційних систем в електронну комерцію, і їх підхід до персоналізації відіграє центральну роль у їхньому бізнес-моделі. Компанія використовує складні алгоритми машинного

навчання для аналізу інформації про поведінку покупців, включаючи історію пошуків, переглядів товарів, покупок, а також рейтинги і відгуки. Ці дані дозволяють Amazon створювати високо персоналізовані рекомендації, які виводяться на різних етапах користувацького досвіду, від головної сторінки сайту до сторінки продукту та етапу оформлення покупки. Підхід Amazon до рекомендацій також включає техніку "клієнти, які купили цей товар, також купили", що дозволяє виявити та показувати продукти, які часто купуються разом. Це не тільки підвищує продажі шляхом крос-продажу, але й покращує досвід користувача, пропонуючи їм товари, які можуть бути їм цікавими. Використання рекомендаційних систем дозволяє Amazon оптимізувати весь покупецький процес, збільшуючи конверсію та середній чек. Крім того, компанія активно використовує рекомендації у своїх маркетингових кампаніях, включаючи персоналізовані рекламні банери та електронні розсилки. Цей підхід дозволяє Amazon не просто продавати більше, але й побудувати більш міцні та тривалі відносини зі своїми клієнтами, надаючи їм високо персоналізовані та релевантні пропозиції.

Rozetka, один з найбільших онлайн-маркетплейсів в Україні, активно використовує рекомендаційні системи для підвищення ефективності продажів та поліпшення користувацького досвіду. Коли користувач переглядає певний товар, система автоматично генерує та відображає розділ із схожими товарами, базуючись на алгоритмах, які аналізують характеристики продукту, історію переглядів інших користувачів та їхні покупки. Це допомагає покупцям відкрити для себе продукти, які вони могли пропустити, та стимулює додаткові продажі. Крім того, Rozetka використовує персоналізовані розсилки для взаємодії зі своїми клієнтами. На основі інформації про попередні покупки та перегляди товарів користувачі отримують листи з пропозиціями, які можуть їх зацікавити. Такий підхід не тільки підтримує інтерес клієнтів до платформи, але й значно підвищує шанси на повторні покупки. На етапі оформлення замовлення система також рекомендує користувачам додаткові товари, які можуть доповнити їхню покупку. Це може бути все, від аксесуарів до товару, який перебуває у кошику,

до товарів, які часто купуються разом. Така стратегія допомагає максимізувати вартість кожного замовлення, а також покращує досвід користувача, оскільки пропонує їм все необхідне в одному місці. Rozetka постійно вдосконалює свої алгоритми рекомендацій, щоб забезпечити максимальну релевантність та ефективність своїх пропозицій, що в кінцевому результаті призводить до підвищення задоволеності клієнтів та зростання прибутків.

Megogo - популярний український онлайн-сервіс для перегляду фільмів, серіалів та телепередач, активно інтегрує рекомендаційні системи для покращення користувацького досвіду. Використовуючи алгоритми машинного навчання та аналізуючи історію переглядів та вподобань користувачів, Megogo пропонує персоналізовані рекомендації, які допомагають користувачам відкрити для себе новий контент відповідно до їхніх інтересів. Система рекомендацій Megogo не тільки аналізує вподобання окремих користувачів, але й враховує загальні тренди та популярність конкретних фільмів і серіалів серед широкої аудиторії. Це дозволяє сервісу пропонувати актуальний та релевантний контент, збільшуючи ймовірність задоволення потреб користувача. Крім того, Megogo використовує рекомендації для стимулювання перегляду нових релізів та популяризації контенту власного виробництва. Персоналізовані пропозиції допомагають підтримувати високий рівень залученості користувачів, підвищуючи їхню вірність сервісу.

1.6 Постановка задачі дослідження

Об’єктом дослідження в рамках магістерської роботи є процес створення рекомендаційної системи для вибору товару на маркетплейсі.

Предметом дослідження є методи і алгоритми, які використовуються для створення таких систем персоналізації товарів.

Метою дослідження є дослідження методів та алгоритмів побудови рекомендацій, і підвищення їх ефективності та подальша інтеграція обраного алгоритму в існуючий маркетплейс Bigl.ua.

Для того, щоб досягти поставлених цілей, необхідно виконати такий ряд задач:

- провести дослідження предметної області та наявних даних;
- порівняти існуючі методи побудови систем рекомендацій та обрати той, який найбільше підходить та задовольняє потреби та вирішує поставлені задачі;
- модифікувати існуючий обраний метод під потреби майданчику;
- розробити програмну реалізацію рекомендаційної системи, базуючись на попередньо створеній моделі, з використанням актуальної бібліотеки нейронних мереж для мови програмування Python;
- оцінити якість отриманої моделі за допомогою актуальних метрик для рекомендаційних систем.

Дослідження повинно завершитися розробкою методу для створення рекомендаційних систем, яка перевершує попередні підходи за ефективністю, а також програмне забезпечення, яке ралізовує дану концепцію системи рекомендацій для вибору продукції на маркетплейсі.

Висновки до розділу 1

Підсумовуючи розділ, який був присвячений дослідженню предметної області використання рекомендаційних систем, можна зробити важливі висновки. Перш за все, було обгрунтовано, що рекомендаційні системи відіграють вирішальну роль у взаємодії між користувачами та платформами, допомагаючи останнім висвітлювати найбільш відповідний контент і товари. Ми розглянули основні типи рекомендаційних систем, такі як колаборативний

фільтринг, контент-орієнтовані системи та гібридні моделі, кожна з яких має власні унікальні характеристики, сценарії застосування, а також свої переваги та недоліки.

Великі гравці ринку, такі як Netflix, Amazon, Spotify та інші, успішно інтегрували рекомендаційні системи у свої сервіси, підвищуючи задоволеність користувачів і збільшуючи прибутковість. В той же час, українські сервіси, такі як Rozetka та Megogo, також активно використовують рекомендації для покращення взаємодії зі своїми користувачами. Також було розглянути, як саме персоналізація використовується в сфері електронної комерції. Алгоритми рекомендацій можуть враховувати різні фактори, такі як популярність товару, його релевантність до потреб користувача, сезонність та актуальні тренди. Це дозволяє веб-магазинам бути на крок попереду, пропонуючи користувачам саме те, що їм може сподобатися, і тим самим підвищуючи конверсію та загальну ефективність бізнесу. Було також розглянути, як алгоритми машинного навчання використовуються у вигляді блоків та розсилок для користувачів.

Проте, реалізація ефективної рекомендаційної системи вимагає уважного аналізу інформації, зібраної з користувачів, а також вибору правильного алгоритму і методу. Створення точних та персоналізованих рекомендацій є складним завданням, яке вимагає глибокого розуміння не тільки технічних аспектів, але і психології користувача.

У подальших розділах ми розглянемо методології та техніки розробки рекомендаційних систем більш детально, з акцентом на ті виклики та можливості, які вони представляють для бізнесу та користувачів.

РОЗДІЛ 2 АНАЛІЗ ОСОБЛИВОСТЕЙ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

У цьому розділі розглянемо наявні базові методи побудови та оцінки рекомендаційних систем, а також обгрунтуємо вибір одного з них для вирішення поставленої задачі.

2.1 Опис існуючих методів, що використовуються у персоналізації

Постановка задачі:

Нехай дано U — множина користувачів, |U| = m, I — множина предметів, |I| = n, D — множина дій надання персоналізованих рекомендацій. Потрібно зробити прогноз персоналізованих рекомендацій U×I→D.

Потрібно знайти прогноз персоналізованих рекомендацій

$$I_{pred} = Predict(U_i, I_i),\ i \in \{1, ..., n\}, \quad (2.1)$$

де I_{pred} – предмет, запропонований користувачу, як рекомендація.

Методи неперсоналізованих рекомендацій. Не персоналізовані системи рекомендацій пропонують продукти клієнтам на основі середньої оцінки, яку інші клієнти дали цим продуктам. Рекомендації не залежать від конкретного клієнта, тому кожен клієнт отримує одні й ті ж рекомендації. Такі системи є автоматизованими, оскільки для генерації рекомендацій від користувача вимагається мінімальне зусилля, а також тимчасовими, оскільки система не розпізнає клієнта з сесії в сесію, оскільки рекомендації не базуються на інформації про клієнта. Не персоналізовані системи рекомендацій часто

використовуються у фізичних магазинах, оскільки їх можна налаштувати на дисплеї, який бачать всі клієнти без змін. Наприклад, виведені середні оцінки користувачів на вебсайтах Amazon.com та Moviefinder.com є прикладами не персоналізованих рекомендацій. Ці рекомендації абсолютно не залежать від конкретного користувача, на якого орієнтована система рекомендацій. eBay використовує трохи інший тип не персоналізованої рекомендації у своєму профілі відгуків. Клієнти залишають відгуки один про одного, а не про продукти! Потім середні та індивідуальні відгуки стають доступні для розгляду покупцями, щоб вирішити, чи варто довіряти конкретному продавцю, і для продавців, щоб вирішити, чи варто довіряти конкретному покупцю. Всі три з цих систем перебувають майже повністю на автоматизованому та тимчасовому полюсі. Інший тип не персоналізованих рекомендацій – текстові коментарі, які підтримуються в розділі коментарів клієнтів Amazon та профілі відгуків eBay. Обидва вони все ще є тимчасовими, але зрушують ближче до ручного кінця іншої вісі. По суті, система просто надає сиру інформацію користувачеві, якому потім самому потрібно систематизувати дані і самостійно витягти з них сенс.

Огляд моделей колаборативної фільтрації. Колаборативна фільтрація є ключовою концепцією в системах рекомендацій, яка заснована на ідеї використання оцінок та переваг одних користувачів для передбачення інтересів інших. Цей розділ дисертації має на меті розглянути основні моделі колаборативної фільтрації, їх переваги та недоліки, а також варіанти застосування у різних сферах.

Методи для вирішення завдання колаборативної фільтрації можна умовно розділити на дві великі групи:

– методи, засновані на евристиках або memory-based Визначають оцінки вподобань, використовуючи критерій подібності між користувачами або елементами;
– методи, засновані на побудові моделі вподобання або model-based.

Передбачають розробка моделі машинного навчання, що інтегрує латентні (приховані) характеристики як користувачів, так і елементів.

Формалізація задачі колаборативної фільтрації. Нехай U— множина користувачів, I — множина об’єктів.

Інформація про відомі вподобання представлена у вигляді набору трійок:

$$D = \{(u, I, r_{ui})\}\ (u, i) \in R, \tag{2.2}$$

де $r_{ui} \in R$ — кількісна величина вподобання об’єкта $i \in I$ користувачем $u \in U$, $U \times I$ — множина пар(користувач, об’єкт), про які відома ступінь вподобання. Для подальшої зручності, введемо також позначення:

$R_u = \{i : u, i \in R\}$— множина об’єктів, суміжних з користувачем u,

$R_i = \{u : i, u \in R\}$— множина користувачів, суміжних з об’єктом i.

За відомою інформацією D потрібно вміти будувати передбачення вподобання $r_{ui}` \approx r_{ui}$ для нових пар (u,i), які не входять у R.

Матрицею оцінок будемо називати матрицю $R \in (\mathbb{R} \cup \emptyset)^{U \times |I|}$, рядки якої відповідають користувачам, стовпці — об’єктам, а елементи приймають значення r_{ui}, якщо $(u,i) \in R$, інакше пропуск.

Memory-based підхід. Memory-based алгоритм колаборативної фільтрації являє собою зважування вподобань по користувачах (user-based) і по об’єктах (item-based).

$$\hat{r}_{ui} = \bar{r}_u + \frac{1}{\sum_{u' \in R(i)} |\mathrm{sim}(u,u')|} \sum_{u' \in R(i)} \mathrm{sim}(u,u')(r_{u',i} - \bar{r}_{u'}),$$
$$\hat{r}_{ui} = \bar{r}_i + \frac{1}{\sum_{i' \in R(u)} |\mathrm{sim}(i,i')|} \sum_{i' \in R(u)} \mathrm{sim}(i,i')(r_{u,i'} - \bar{r}_{i'}). \tag{2.3}$$

Міра схожості *sim(u, u')* обчислюється за матрицею оцінок *R*. Найбільш загальновживані прості метрики схожості — кореляція Пірсона і косинусна відстань відповідних рядків (стовпців) матриці оцінок:

$$\mathbf{sim}(u,u') = \frac{\sum_{i\in R(u)\cap R(u')}\left(r_{u,i}-\bar{r}_u\right)\left(r_{u',i}-\bar{r}_{u'}\right)}{\sqrt{\sum_{i\in R(u)\cap R(u')}\left(r_{u,i}-\bar{r}_u\right)^2\sum_{i\in R(u)\cap R(u')}\left(r_{u',i}-\bar{r}_{u'}\right)^2}}\ , \qquad (2.4)$$

$$\mathbf{sim}(u,u') = \frac{\sum_{i\in R(u)\cap R(u')} r_{u,i}\, r_{u',i}}{\sqrt{\sum_{i\in R(u)} r_{u,i}^2 \sum_{i\in R(u')} r_{u',i}^2}}\ . \qquad (2.5)$$

Model-Based підхід. Model-based алгоритми шукають оцінку ř, як функцію. Модель машинного навчання реалізується шляхом мінімізації регуляризованого емпіричного ризику:

$$\mathcal{L}(\theta) = \sum_{(u,i)\in R} l(\hat{r}(u,i;\,\theta), r_{ui}) + \lambda\Omega(\theta) \to \min_{\theta\in\theta}\ , \qquad (2.6)$$

де I(ř, r), — функція втрат регресії, λ — сила регуляризації,

Ω (θ) — регуляризатор на множині параметрів θ

Використовуючи результати методу SVD-розкладу, маємо таке представлення функції:

$$\hat{r}(u,i;\theta) = p_u^T q_i,\ \ \theta = (\{p_u\}_{u\in U},\{q_i\}_{i\in I}), \qquad (2.7)$$

де $p_u \in \mathbb{R}^d$ — вектор прихованих (латентних) ознак користувача u,

$q_i \in \mathbb{R}^d$— аналогічний вектор для об’єкта i,

d — розмірність латентних векторів (величина низькорангового наближення).

Моделі, які будуть розглядатися далі, навчаються шляхом оптимізації квадратичних втрат, I(ř, r) = (ř - r)^2 з квадратичною регуляризацією.

Моделі колаборативної фільтрації постійно розвиваються, адаптуючись до нових викликів та потреб користувачів. Покращення в області машинного

навчання та збільшення обсягів даних відкривають нові перспективи для створення більш точних та персоналізованих систем рекомендацій.

Огляд моделей контентної фільтрації. Для реалізації цих методів, потрібно встановити відповідність між користувачами та продуктами або контентом, який їх зацікавив. Для кожного слова ми ведемо реєстрацію його ймовірності з'явитися в контенті, з яким взаємодіяв користувач. Створюючи такі «профілі», ми розраховуємо схожість між користувачами та продуктами. Продукти слід рекомендувати користувачу, якщо 1) вони демонструють найвищу схожість з користувачем, або 2) виявляють значну схожість з іншими елементами, які переглядав користувач. Для цього можна застосовувати ті ж методи, які були обговорені раніше, включаючи косинусну схожість, кореляцію Пірсона та інші.

Метод TF-IDF (Term Frequency-Inverse Document Frequency) є популярною технікою в області обробки природної мови та інформаційного пошуку, яка часто використовується для рекомендаційних систем, зокрема в контент-базових підходах. Основна ідея методу полягає в оцінці важливості слова в контексті документа на основі його частоти в документі та оберненої частоти документів, що містять це слово.

1 **Частота терміна (TF)**: Ця компонента визначає, як часто певне слово з’являється в документі. Вона вираховується як кількість разів, коли слово з'являється в документі, поділена на загальну кількість слів у документі. Це дає нам відносну частоту слова, яка допомагає уникнути спотворення внаслідок різних довжин документів.
2 **Обернена частота документів (IDF)**: Ця компонента вимірює, наскільки унікальним або рідкісним є слово в усій колекції документів. Вона вираховується як логарифм відношення загальної кількості документів до кількості документів, які містять слово. Це допомагає зменшити вагу загальновживаних слів, які з'являються у багатьох документах, і підвищити вагу слів, які з'являються рідше.

Множення TF та IDF дає оцінку TF-IDF для кожного слова у документі. Ця оцінка відображає важливість слова в контексті конкретного документа відносно всієї колекції документів. Чим вища оцінка TF-IDF для слова, тим більше воно є важливим для документа.

У контексті рекомендаційних систем метод TF-IDF може використовуватися для визначення схожості між документами або між користувачами та документами. Наприклад, якщо у нас є база даних з описами фільмів, метод TF-IDF може допомогти визначити, які фільми є найбільш схожими на певний фільм на основі опису, тим самим надаючи рекомендації користувачеві. Це також може бути застосовано до текстових відгуків користувачів для визначення їхніх переваг та рекомендації продуктів або контенту, який їм може сподобатися.

Цей метод можна описати наступною формулою:

$$TF\text{-}IDF(t, d) = TF(t, d) * IDF(t), \quad (2.8)$$

де TF(t, d) - частота термина (TF) для слова "t" в документі "d",

IDF(t) - зворотня частота документа (IDF) для слова "t".

Гібридні моделі рекомендаційних систем існують у різних виглядах та можуть реалізовувати різні підходи, тому класифікуємо їх далі.

Із ваговими коефіцієнтами (рис. 2.1).

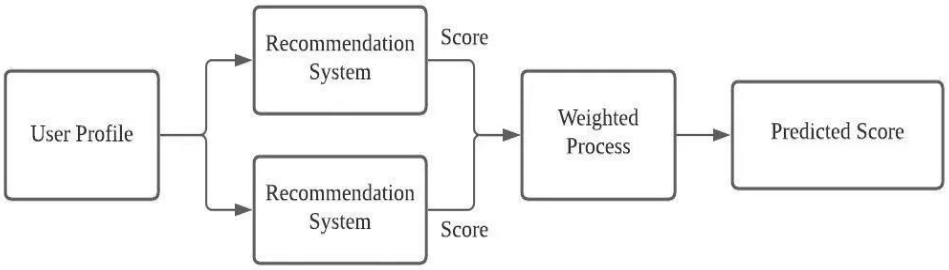

Рисунок 2.1 – Схема алгоритму із ваговими коефіцієнтами [1]

[1] Джерело зображення: https://miro.medium.com/v2/resize:fit:720/format:webp/1*N9TO6n2tlo1sN8GhsB-tqw.jpeg

У системі рекомендацій із ваговими коефіцієнтами можемо визначити декілька моделей, які здатні ефективно аналізувати набір даних. Така система буде об'єднувати результати від кожної з моделей, використовуючи статичні вагові коефіцієнти, що не змінюються протягом тренування та тестування.

Наприклад, ми можемо поєднати модель, засновану на вмісті, та модель колаборативного фільтрування з елементом до елементу, де кожна з них матиме вагу 50% у фінальному прогнозуванні.

Перевагою використання вагового гібридного підходу є те, що ми інтегруємо кілька моделей для підтримки набору даних у процесі рекомендацій лінійним чином.

З перемиканням (рис. 2.2).

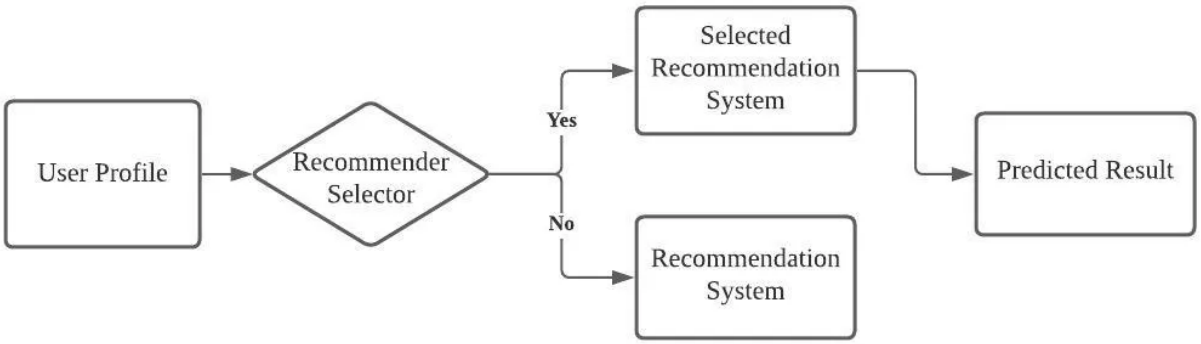

Рисунок 2.2 – Схема алгоритму з перемиканням [2]

Гібридний підхід з перемиканням обирає одну систему рекомендацій залежно від ситуації. Модель створюється для набору даних, чутливих на рівні окремих об'єктів, тому ми повинні встановити критерії вибору рекомендера залежно від профілю користувача або інших характеристик.

Гібридний підхід із перемиканням додає додатковий шар поверх моделі рекомендацій, який вибирає відповідну модель для використання. Система

[2] Джерело зображення: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*oyZqArrmwxb5n-sshMzq-w.jpeg

рекомендацій чутлива до сильних і слабких сторін складових моделей рекомендацій.

Змішаний (рис. 2.3).

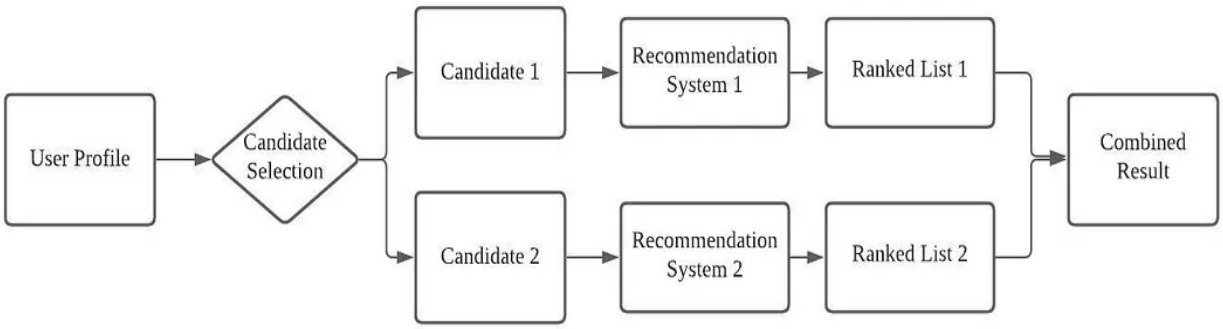

Рисунок 2.3 – Схема змішаного алгоритму[3]

Гібридний підхід із змішуванням спочатку використовує профіль користувача та характеристики для створення різних наборів кандидатів на датасети. Система рекомендацій подає різні набори кандидатів відповідно до моделі рекомендацій і комбінує прогнози для отримання результату рекомендацій.

Система рекомендацій із змішуванням здатна одночасно робити велику кількість рекомендацій і адаптувати частковий датасет до відповідної моделі з метою досягнення кращих результатів.

[3] Джерело зображення: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*jcpCf-mfB4AI7JL7x6WmqA.jpeg

З комбінацією ознак (рис. 2.4)

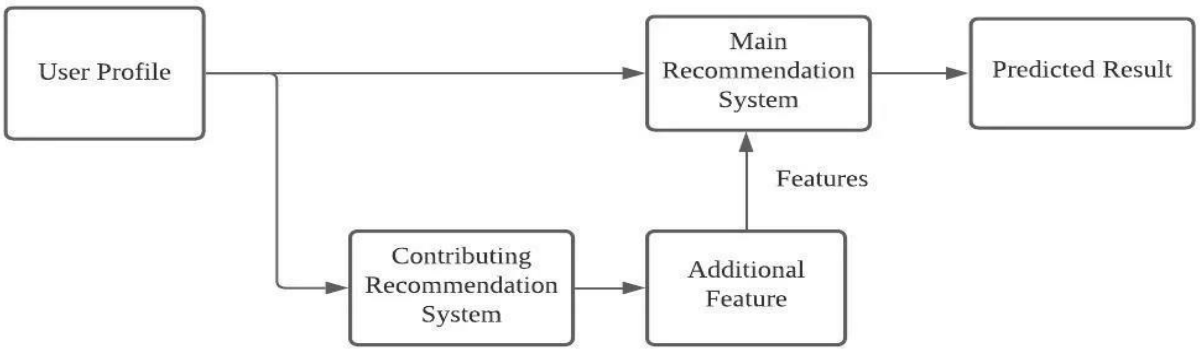

Рисунок 2.4 - Схема алгоритму з комбінацією ознак[4]

У гібридному підході з комбінацією ознак ми додаємо віртуальну модель рекомендацій, яка виступає у ролі інженерії характеристик щодо первинного датасету профілю користувача.

Наприклад, ми можемо інтегрувати характеристики моделі колаборативних рекомендацій у модель, засновану на вмісті. Ця гібридна модель здатна враховувати колаборативні дані з підсистеми, не покладаючись виключно на одну модель.

З аугментацією ознак (рис. 2.5)

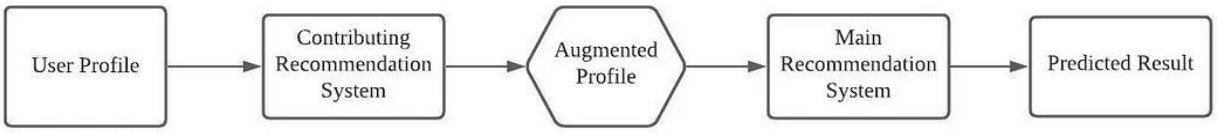

Рисунок 2.5 - Схема алгоритму з аугментацією ознак[5]

[4] Джерело зображення: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*K4ZRADAka8DUM3xh8H76DQ.jpeg
[5] Джерело зображення: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*1O_Ve4JyETRCybwT6-1ErA.jpeg

Ця модель рекомендацій використовується для генерації оцінки або класифікації профілю користувача/товару, яку потім використовують у основній системі рекомендацій для отримання кінцевого прогнозованого результату. Цей підхід здатний покращити продуктивність основної системи, не змінюючи при цьому основну модель рекомендацій. Наприклад, використовуючи асоціативні правила, ми можемо покращити датасет профілю користувача. З таким розширеним набором даних продуктивність моделі рекомендацій, заснованої на вмісті, буде підвищена.

2.2 Рекомендаційні системи з використанням нейронних мереж

У сучасному цифровому світі, де обсяги даних зростають з неймовірною швидкістю, рекомендаційні системи відіграють важливу роль у персоналізації користувацького досвіду. Нейронні мережі, завдяки своїй здатності виявляти складні залежності та взаємозв'язки в великих наборах даних, стали ключовим інструментом у розробці ефективних рекомендаційних систем.

Перш за все, варто зазначити, що нейронні мережі дозволяють втілити глибоке навчання (deep learning), яке значно підвищує точність рекомендацій шляхом вивчення великої кількості характеристик користувачів та продуктів. Наприклад, системи, засновані на глибоких нейронних мережах, можуть аналізувати історію переглядів, покупок, поведінкові та демографічні дані користувачів, щоб пропонувати більш точні та персоналізовані рекомендації. Одним з основних напрямків у цій області є використання конволюційних нейронних мереж (CNN) та рекурентних нейронних мереж (RNN) для аналізу візуального та текстового контенту. CNN ефективно використовуються для аналізу зображень товарів, виявлення особливостей, які можуть привернути увагу користувачів. RNN, у свою чергу, використовуються для аналізу текстових

даних, таких як описи продуктів або відгуки користувачів, щоб зрозуміти сутність відгуків та переваги користувачів.

Окрім цього, алгоритми машинного навчання, такі як мережі з автоенкодерами, використовуються для зменшення розмірності та виявлення прихованих факторів, які впливають на переваги користувачів. Це дозволяє системам рекомендацій виявляти неявні зв'язки та шаблони, які не завжди очевидні при традиційному аналізі.

2.2.1 Навчання нейронної мережі

Стохастичний градієнтний спуск (SGD) стоїть у центрі процесу тренування нейронних мереж, використовуючи метод зворотного поширення помилок для обчислення градієнтів параметрів. У цьому процесі, зворотне поширення втілює рекурсивне використання ланцюгового правила диференціального числення для визначення градієнтів уздовж структури нейронної мережі. Ця методика є фундаментальною для навчання мереж, дозволяючи точно визначити градієнти для функції втрат по кожному параметру. Попри те, що традиційний SGD може мати повільну збіжність, використання методів, таких як імпульс чи прискорений імпульс Нестерова, спрямоване на покращення швидкості збіжності. На початку 2010-х, було зроблено значні дослідження для поліпшення правил оновлення градієнтів, що призвело до створення нових оптимізаційних методів, таких як Adagrad, RMSProp та Adam. Ці методи пропонують адаптивне оновлення ваг, враховуючи особливості кожного параметра. Особливо, Adam (Adaptive Moment Estimation) поєднує переваги RMSProp та імпульсу, забезпечуючи швидке оновлення ваг. У рамках цієї роботи, алгоритм Adam було обрано за його високу ефективність у збіжності та здатність адаптувати швидкість навчання для кожного параметра, що забезпечує більш швидкі та якісні результати порівняно зі звичайним SGD.

Правильний вибір функції активації відіграє вирішальну роль у проектуванні та тренуванні нейронних мереж. Спочатку сигмоїдна функція користувалася популярністю, оскільки вона стискала вихід у діапазон від 0 до 1. Проте, згодом виявилося, що вона має кілька суттєвих недоліків, які можуть перешкоджати ефективному навчанню мереж. Основна проблема сигмоїдної функції полягає у зникаючому градієнті: коли активації дуже малі чи великі, градієнти цієї функції стають майже нульовими. Ще одна проблема - нецентрованість вихідних значень навколо нуля, що означає, що вихід завжди позитивний, що може ускладнити оновлення ваг в мережі. Крім того, сигмоїдна функція вимагає значних обчислювальних зусиль через свої експоненційні розрахунки, що стає особливо помітним у великих мережах та при великих обсягах даних, де важлива обчислювальна ефективність.

На ранніх етапах розвитку складних нейронних мереж, процес навчання часто зазнавав недоліків через недосконалі методи ініціалізації ваг. Застосування випадкових невеликих значень часто призводило до проблеми зникаючих градієнтів, що ускладнювало тренування глибоких мереж. Прогрес у цій сфері відбувся завдяки методу ініціалізації Xavier, розробленому Глоротом та Бенджо, який забезпечує рівномірний розподіл ваг, враховуючи кількість входів нейрону. Це допомагає стандартизувати розподіл вихідних значень між шарами та покращує градієнти у мережі. Ініціалізація Xavier сьогодні широко використовується і часто доповнюється нормалізацією партій, що сприяє регулюванню проміжних значень у мережі.

Важливу роль у розвитку нейронних мереж останніми роками відіграла зростаюча обчислювальна потужність сучасних комп'ютерів. Порівняно з комп'ютерами кінця 1950-х, коли нейронні мережі тільки зароджувалися, сучасні процесори демонструють астрономічне зростання в обчислювальній потужності. Використання мультиядерних CPU і, зокрема, графічних процесорів (GPU), значно зменшило час, необхідний для тренування складних мереж. Окрім того, збільшення обсягу доступних даних для навчання сприяло створенню великих датасетів, що містять мільйони прикладів і є доступними для дослідників.

2.2.2 Процес ініціалізації ваг нейронної мережі

Існує кілька підходів до ініціалізації ваг у нейронних мережах, кожен з яких має свої особливості та застосування. Традиційно, випадкова ініціалізація, така як ініціалізація з малими випадковими числами, була поширеною практикою. Інший відомий метод - це ініціалізація Xavier (або Glorot), яка враховує розміри попереднього та наступного шарів для визначення масштабу розподілу ваг, ініціалізація ваг w_i симетричним розподілом з дисперсією.

$$Var(w_i) = \frac{2}{n\_in + n\_out} \quad , \tag{2.9}$$

де *n_in* – це кількість входів, а *n_out* – це кількість виходів.

Існує також ініціалізація He, яка є модифікацією Xavier і особливо ефективна для ReLU активацій, це ініціалізація ваг w_i нейронів l-го шару за нормальним законом з нульовим середнім і дисперсією.

$$Var(w_i) = \frac{2}{n\_in^l}, \tag{2.10}$$

де n_in^l - це кількість нейронів l-го шару.

Ці методи допомагають запобігти проблемам зникаючих або вибухаючих градієнтів під час навчання. У цій роботі ми обрали ініціалізацію Xavier, оскільки вона добре збалансована для різних архітектур нейронних мереж і дозволяє зберегти розподіл активацій і градієнтів на стабільному рівні протягом тренування. Це робить її ідеальним вибором для нашої мережі, забезпечуючи ефективний початок процесу навчання і зменшуючи ризик неправильного пропагування сигналів через мережу.

2.2.3 Методи навчання нейронних мереж

Навчання нейронних мереж є фундаментальним процесом, який визначає їхню ефективність та здатність до вирішення різноманітних завдань. Серед численних методів оптимізації, які використовуються в машинному навчанні, особливо важливі стохастичний градієнтний спуск (SGD) та Adam.

Стохастичний Градієнтний Спуск (SGD). SGD є одним з найпопулярніших методів навчання нейронних мереж. Він є варіацією класичного градієнтного спуску, де ваги мережі оновлюються на кожному кроці, використовуючи тільки підмножину тренувальних даних. Це дозволяє SGD бути набагато швидшим і ефективнішим при роботі з великими наборами даних. Однак, через свою стохастичну природу, метод може бути менш стабільним і потребувати більше ітерацій для досягнення конвергенції.

Нехай y^*: X -> Y – це цільова залежність, відома на об'єктах навчальної вибірки. Нижче наведено алгоритм для звичайного лінійного класифікатора:

$$\alpha(x, W) = \varphi(\sum_{i=1}^{n} W_i - W_0), \tag{2.11}$$

де φ – це функція активації.

На вхід алгоритму подаються наступні параметри:

- X^l - навчальна вибірка;
- η - темп навчання;
- λ - параметр згладжування функціоналу Q.

Виходом є вектор ваг W.

Спочатку ініціалізовується вектор ваг w_j, а також функція оцінки

$$Q = \sum_{i=1}^{n} L(\alpha(x, W), y_i). \tag{2.12}$$

Ітеративно повторювати наступні кроки:

- вибрати певним чином об'єкт x_i із X_l , можливо, випадково;
- обрахувати результат алгоритму $\alpha(x, W)$ та помилку;
- виконати крок градієнтного спуску;

$$w := w - \eta \dot{L_a}(a(x_i, w), y_i)\dot{\varphi}(< w, x_i >)x_i \ . \tag{2.13}$$

- оцінити функцію якості Q.

$$Q := (1 - \lambda)Q + \lambda_{\xi_i} \ . \tag{2.14}$$

Далі повторювати кроки до стабілізації Q, або зміна ваг не буде змінюватись менше, ніж початково задані межі.

Adam (Adaptive Moment Estimation). Adam, який стоїть за "Adaptive Moment Estimation", є більш сучасним методом оптимізації, який об'єднує переваги двох інших методів оптимізації: RMSprop та SGD з імпульсом. Adam коригує швидкість навчання для кожного параметра індивідуально на основі оцінок першого (середнє) та другого (невизначене відхилення) моментів градієнтів. Це робить Adam особливо ефективним в контексті великих наборів даних та параметрів, а також у ситуаціях, де градієнти мають високу варіабельність. Розглянемо сам алгоритм.

Спочатку задається величина кроку ε (за замовчуванням 0.001), коефіцієнти експоненціального затухання для оцінок моментів ρ1 і ρ2, що належать діапазону [0, 1) (за замовчуванням 0.9 і 0.999 відповідно), константа δ для забезпечення чисельної стійкості, а також початкові значення параметрів θ.

На початку алгоритму ініціалізувується змінні для першого і другого моментів s = 0, r = 0, а також ініціалізувати крок за часом t = 0.

Далі повторювати наступне до виконання критерію зупинки:

– вибрати з навчального набору міні-пакет m прикладів $\{x_1, ..., x_m\}$ і мітки y_i;

– обчислити градієнт: g ← (1 / m) ∇ θΣi L (f (x (i); θ), y (i)). (t ← t + 1);

– оновити зміщену оцінку першого моменту: s ← ρ1s + (1– ρ1) g;

– оновити зміщену оцінку другого моменту: r ← ρ2r + (1 – ρ2) g Θ g;

– скорегувати зміщення першого моменту: $\hat{S} \leftarrow \frac{S}{1-p_1^t}$;

– скорегувати зміщення другого моменту: $\hat{r} \leftarrow \frac{r}{1-p_1^t}$;

– обчислити приріст ваги: $\Delta\theta = -\xi \frac{S}{\hat{r}+\delta}$;

– оновити ваги: θ ← θ + Δθ.

Налаштування ваг відбувається наступним чином:

$$S_t := a * S_{t-1} + (1-a) * \nabla E_t^2; S_0 := 0$$

$$D_t := \beta * D_{t-1} + (1-\beta) * \nabla E_t; D_0 := 0$$

$$g_t := \frac{D_t}{1-\beta} * \sqrt{\frac{1-a}{S_t}}$$

$$\Delta W_t := \eta * (g_t + p * W_{t-1}) + \mu * \Delta W_{t-1}$$

де η – коефіцієнт швидкості навчання,

∇ E – градієнт функції втрати,

μ – коефіцієнт моменту,

ΔW t-1 – зміна ваг на попередній ітерації,

ρ – коефіцієнт регуляризації,

W t-1 – значення ваг на попередній ітерації,

α = 0.999, β = 0.9.

Вибір між SGD та Adam залежить від конкретної задачі та обсягу даних. SGD може бути більш ефективним при правильному підборі розміру пакету та швидкості навчання, особливо в ситуаціях, де простір параметрів є відносно невеликим. З іншого боку, Adam, завдяки своїй адаптивності, є більш підходящим для складних архітектур нейронних мереж із великою кількістю параметрів. Його здатність адаптувати швидкість навчання для кожного параметра робить його ідеальним для прискорення процесу навчання у великих мережа.

2.3 Метрики оцінки якості моделі

Метрики оцінки якості є критично важливими для вимірювання ефективності нейронних мереж. Вони дозволяють не лише оцінити продуктивність моделі на даних, але й спрямовують процес налаштування та вдосконалення моделей. Оцінка якості рекомендацій, наданих за допомогою певного алгоритму, може бути виконана через застосування різноманітних методів вимірювання. Серед цих методів можна виміряти такі показники, як точність та охоплення. Точність визначається як пропорція коректних рекомендацій серед всіх можливих рекомендацій. Охоплення відображає пропорцію об'єктів у просторі пошуку, для яких система здатна надати рекомендації. Існують різні метрики для вимірювання точності, які можуть бути класифіковані як статистичні або специфічні метрики точності. Ефективність кожного з цих показників залежить від характеристик конкретного набору даних та сфери застосування. Для візних задач машинного навчання існують різні набори метрик. Для задач регресії існують такі метрики, як Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R^2 (R-Squared).

MSE - Середньоквадратична помилка є, одним із найпопулярніших показників, що використовується для задач регресії. Вона в основному визначає середнє значення квадратів різниці між цільовим значенням та значенням, передбаченим моделлю регресії.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_{true} - y_{pred})^2, \quad (2.15)$$

де y_{true} – значення, яке намагаємось передбачити,

y_{pred} – значення, яке є передбаченням

Можна виділити кілька важливих аспектів пов'язаних з MSE:

– вона диференційована, тому може бути краще оптимізована;

– малі помилки вона штрафує шляхом їхнього піднесення до квадрату, що, по суті, призводить до переоцінки того, наскільки поганою є модель;

– інтерпретація помилки має бути зроблена з урахуванням квадратичного масштабування;

– через фактор піднесення до квадрату вона фундаментально більш схильна до впливу викидів, ніж інші метрики.

MAE - Середня абсолютна помилка - це середнє значення різниці між реальними значеннями та передбаченими.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}|y_{true} - y_{pred}|. \quad (2.16)$$

Ось декілька ключових моментів щодо MAE:

– вона більш стійка до викидів порівняно з MSE, оскільки не перебільшує помилки;

– ця метрика надає міру того, наскільки далеко передбачення віддалені від фактичного результату. Однак, оскільки MAE використовує абсолютне значення решти, вона не дає уявлення про напрямок помилки, тобто не показує, чи ми недооцінюємо або переоцінюємо дані;

– інтерпретація помилки не вимагає додаткових роздумів, оскільки вона точно відповідає первісному розміру змінної;

– на відміну від MSE, яка є диференційованою, MAE не може бути диференційована.

RMSE (Root Mean Squared Error) - середньоквадратична помилка відповідає квадратному кореню з середнього значення квадратів різниці між цільовим значенням та значенням, передбаченим моделлю регресії. По суті, це sqrt(MSE). Математично це можна представити так: вона вирішує деякі недоліки MSE.

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}|y_{true} - y_{pred}|}. \quad (2.17)$$

Кілька ключових моментів, пов'язаних з RMSE:

– вона зберігає диференційовану властивість MSE;

– вона вирішує проблему підвищеної покарання за невеликі помилки в MSE шляхом використання квадратного кореня;

– інтерпретація помилок може бути здійснена легко, оскільки шкала тепер така ж, як і у випадкової змінної;

– оскільки фактори масштабування по суті нормалізовані, вона менш схильна до проблем у випадку викидів.

В свою чергу, моделі класифікації мають дискретний вихід, тому нам потрібен показник, який порівнює дискретні класи в певній формі. Метрики класифікації оцінюють ефективність моделі та показують, наскільки добре чи погано виконується класифікація, проте кожна з них оцінює це по-різному.

Отже, для оцінки моделей класифікації ми обговоримо детально такі метрики:

1. Точність (Accuracy).
2. Матриця помилок (Confusion Matrix) (не є метрикою, але є фундаментальною для інших).

3. Точність (Precision) та Відновлення (Recall).
4. F1-оцінка.
5. AU-ROC (Площа під кривою робочих характеристик приймача).

Усі вищеперераховані метрики не підходять для коректної оцінки алгоритмів в задачах надання рекомендацій, тому будемо використовувати більш специфічні метрики.

Precision at k - це співвідношення між кількістю відібраних релевантних елементів та загальною кількістю вибраних елементів, де ця кількість визначається параметром k. Релевантними вважаються елементи, з якими користувач взаємодіяв протягом тестового періоду. Ця цільова взаємодія може включати різні дії, такі як перегляд, оцінка або покупка, і залежить від конкретних завдань системи.

$$\text{P@K}=\sum_{i=1}^{k} rel(i), \tag{2.18}$$

де $rel(i)$ - це релевантність і-го елементу.

Для оцінювання як успішності потрапляння релевантних елементів у перші k позицій, так і важливості їх розташування ближче до початку списку, використовується метрика AP@k (AP@N). В цій метриці враховуються різні значення точності Precision@k для тих випадків, коли релевантний елемент знаходиться на позиції k.

$$\text{AP@N} = \frac{1}{m}\sum_{k=1}^{N}\left(P(k) \text{ if } k^{th} \text{ item was relevant}\right) = \frac{1}{m}\sum_{k=1}^{N} P(k)\cdot rel(k), \tag{2.19}$$

де m - дорівнює загальній кількості релевантних елементів, N – кількість елементів, які потрібно рекомендувати.

В свою чергу, метрика MAP@k (або MAP@N) є усередненням значень AP@k по тестовому набору:

$$\text{MAP@N} = \frac{1}{|U|}\sum_{u=1}^{|}U|(\text{AP@N})_u = \frac{1}{|U|}\sum_{u=1}^{|}U|\frac{1}{m}\sum_{k=1}^{N}P_u(k)\cdot rel_u(k) \quad . \qquad (2.20)$$

Іншою метрикою, яка бере до уваги розташування релевантних елементів, є Normalized Discounted Cumulative Gain (NDCG). Формула для NDCG виглядає наступним чином:

$$\text{NDCG@K} = \frac{DCG@K}{IDCG@K}, \qquad (2.21)$$

де DCG@K - Discounted Cumulative Gain at K, розрахована за допомогою формули DCG для топ-K результатів,

IDCG@K - Ideal Discounted Cumulative Gain at K, яка представляє ідеальний сценарій, де всі результати відсортовані за максимальною релевантністю.

NDCG метрика вимірює наскільки актуальне ранжування наближається до деякого ідеального варіанту, як наприклад, фактичний порядок взаємодій користувача з елементами.

Вважається, що тільки метрики точності не в змозі повноцінно виміряти якість рекомендаційної системи. Серед більш загальних метрик можна виділити:

- різноманітність: кількість унікальних і релевантних рекомендацій, що допомагають користувачеві відкрити для себе щось нове, але водночас відповідаючи його інтересам;
- охоплення: здатність системи пропонувати різноманітні предмети зі свого датасету користувачам. Це передбачає, що система не повинна ігнорувати нові товари, які ще не отримали відгуків чи оцінок від інших користувачів;
- покриття простору користувачів: це можна характеризувати як долю користувачів або їхніх взаємодій, для яких система здатна надавати рекомендації. Деякі системи можуть бути неефективними у випадках, коли про користувача відомо недостатньо інформації;
- конверсія: оцінка, яка показує як рекомендації впливають на продажі конкретних товарів. Вона показує, як перегляди конвертуються у покупки.

Також можуть бути оцінені к-ть товарів у замовлені або середня сума замовленя.

2.4 Моделі, що базуються на векторизації

Щодня ми працюємо з десятками мільйонів товарів, і наше завдання – виявити та порівняти схожі пропозиції (знайти відповідності) на нашому сайті, з метою об’єднання пропозицій від різних продавців в одній картці товару.

Для кожного продукту є наступна інформація: зображення, назва, опис та додаткові атрибути. Ми хочемо отримати та обробити всю цю інформацію для вирішення різних завдань, причому це особливо важливо для команди знаходження відповідностей товарів.

Для вилучення характеристик з продукту ми можемо створювати векторні представлення (ембеддинги) з використанням різних текстових моделей (fastText, transformers) для описів та назв, а також великої кількості конволюційних нейронних мереж (ResNet, Effnet, NFNet) для зображень. Ці вектори далі використовуються для генерації характеристик і пошуку відповідностей товарів.

Prod2Vec не є універсальним алгоритмом, а лише ідеєю, які слід використовувати, тому ми розглянемо цей алгоритм на основі статті “E-commerce in Your Inbox: Product Recommendations at Scale”, написаною командою Yahoo.

Модель prod2vec включає в себе вивчення векторних представлень продуктів із послідовності переглядів чи покупок, використовуючи поняття послідовності покупок як “речення” та продуктів у цій послідовності як “слів”, запозичуючи термінологію з області обробки природної мови. Більш конкретно, prod2vec вивчає представлення продуктів, використовуючи модель

skip-gram, максимізуючи цільову функцію по всьому S – множині журналів електронних чеків, який визначено наступним чином.

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{p_i \in s} \sum_{-c \leq j \leq c, j \neq 0} \log \mathbb{P}(p_{i+j} | p_i), \tag{2.24}$$

де продукти з однієї електронної квитанції замовляються довільно.

Ймовірність $P(p_{i+j} \mid p_i)$, що спостерігається сусідній продукт p_{i+j} при наявності поточного продукту p i , визначається за допомогою функції soft-max,

$$\mathbb{P}(p_{i+j} | p_i) = \frac{\exp(\mathbf{v}_{p_i}^\top \mathbf{v}'_{p_{i+j}})}{\sum_{p=1}^{P} \exp(\mathbf{v}_{p_i}^\top \mathbf{v}'_p)}, \tag{2.25}$$

де v_p та v_p' — це вхідні та вихідні векторні представлення продукту p, c — довжина контексту для послідовностей продуктів, а P — кількість унікальних продуктів у словнику. З рівнянь (2.24) та (2.25) бачимо, що prod2vec моделює контекст послідовності продуктів, де продукти з подібними контекстами (тобто з подібними сусідніми покупками) матимуть подібні векторні представлення. Однак prod2vec не враховує явно того факту, що електронний чек може містити кілька продуктів, придбаних одночасно, що було виправленно у подальших модифікаціях алгоритму, таких як bagged-prod2vec.

Навчання. Моделі оптимізувалися за допомогою стохастичного градієнтного підйому, що підходить для задач великого масштабу. Однак складність обчислення градієнтів ∇L в ріанняннях вище пропорційно розміру словника P, що є обчислювально витратним на практиці, оскільки P може легко досягати мільйонів продуктів. Як альтернативу було використано підхід negative sampling, який істотно знижує обчислювальну складність.

2.5 Запропонований алгоритм

Дано: Множина продуктів p ∈ P, множина переглянутих користувачем товарів s ∈ S, яка виступає тестовою множиною. У свою чергу множина продуктів P представлена:

- множиною картинок i ∈ I;
- множиною текстів-назв t ∈ T;
- множиною текстів-описів d ∈ D.

Необхідно: знайти схожі товари на ті, які уже були переглянуті користувачем.

Для того, щоб сформувати векторне представлення товару ми можемо скористатись наступними наявними даними:

1. Контентом, тобто текстовою інформацією про товар (назва, опис, атрибути), різною мета-інформацією (бренда, продавець), а також картинкою товару.
2. Користувацькими сессіями, тобто історією переглядів або попупок користувачів.

У рамках даної роботи буде розгядатися перший підхід.

Кожен продукт на сайті належить до якоїсь категорії, які в свою чергу формують дерево категорій, утворюючи ієрархічну структуру. Приклад наведено нижче на картинці

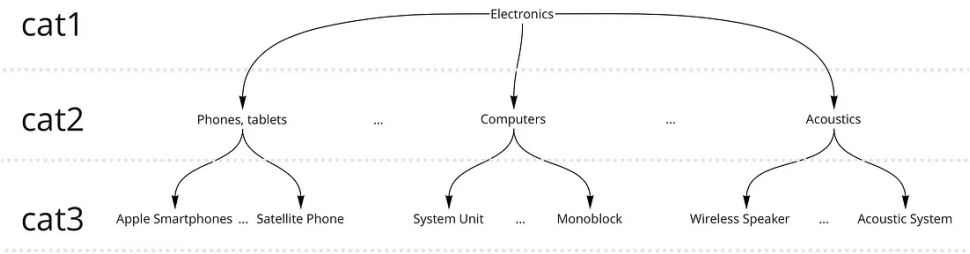

Рисунок 2.6 - Дерево категорій товарів

Кожен товар має картинку, назву та набір атрибутів (розмір, колір та інші), які і будуть слугувати вхідними даними для обраного алгоритму.

Розв’язувати поставлену задачу було вирішено нейронною мережею, яка базується на підході transfer learning та імплементує ідею prod2vec. Для розпізнавання зображень буде обрано одну з існуючих навчаних згорткових нейронних мереж. Для розпізнавання текстів назв та описів товарів буде обрано одну з можелей для обробки природної мови (NLP). Ці моделі схормують перший шар результуючої нейронної мережі, після якого буде додано ше один або декілька шарів з найбільш підходящими функціями активації. Виходом моделі має бути вектор із натуральних чисел заданої розмірності, що буде представляти собою векторне представлення вхідного товару. До кожного товару із тестовою вибірки буде примінено побудований алгоритм, в результаті якого отримаємо веторний простір товарів. У подальших версіях системи можна зберігати ці дані в сховищі, такому як реляційна, нереляційна база даних або хмарні сервіси. Далі буде обрано один з існуючих та найбільш оптимальних алгоритмів пошуку по цьому простору. В результаті, для вхідного продукту, який, до прикладу, нещодавно переглянув користувач, ми матимемо змогу знайти N найближчих сусідів у побудованому просторі, які й будуть необхідними рекомендаціями.

Висновки до розділу 2

У цьому розділі були вивчені математичні принципи існуючих технік рекомендації продуктів, використовуючи різні методології. Також обговорювалися алгоритми, що стосуються найсучасніших способів надання рекомендацій. Були проаналізовані та вибрані методи для оцінки якості рекомендаційних систем. Представлено та обґрунтовано власну модифікацію методу Prod2vec з метою покращення точності рекомендацій.

РОЗДІЛ 3 ОГЛЯД ПРОГРАМНОГО ПРОДУКТУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У цьому розділі розглянемо побудовану моделі та оцінемо якість її роботи на тестових даних за допомогою метрик.

3.1 Обґрунтування вибору платформи та мови програмування

Python відзначається своєю простотою у вивченні та високою читабельністю коду, що робить його ідеальним для широкого спектру розробників – від новачків до досвідчених професіоналів. Його синтаксис вирізняється лаконічністю та ясністю, дозволяючи швидко освоїти основи та зосередитися на вирішенні задач, а не на складнощах мови. Це робить Python особливо привабливим для різних категорій розробників, які цінують простоту та ефективність коду.

Однією з важливих переваг Python є його обширна та активна спільнота, що забезпечує велику кількість ресурсів, обговорень, навчальних матеріалів та бібліотек. Це сприяє легкому вирішенню проблем, які можуть виникнути у розробників, і робить Python відмінним вибором для великих проектів та командної роботи. Python володіє обширною колекцією бібліотек, таких як NumPy, Pandas, Matplotlib, Seaborn та SciPy, які надають потужні та гнучкі інструменти для роботи з даними.

У сфері машинного та глибокого навчання Python пропонує декілька ключових фреймворків, таких як Scikit-learn, TensorFlow та PyTorch. TensorFlow, розроблений Google, є одним з найпопулярніших фреймворків для глибокого навчання, відомий своєю масштабованістю та гнучкістю. Водночас, PyTorch, розроблений Facebook, виокремлюється своїм інтуїтивно зрозумілим

інтерфейсом і динамічним побудовою графів, що робить його особливо зручним для наукових досліджень і експериментів. Вибір PyTorch для проекту був обумовлений його здатністю забезпечувати більш гнучке та інтуїтивне програмування моделей, що є критично важливим для задач OCR та обробки зображень, де необхідно часто експериментувати та ітеративно вдосконалювати моделі.

Що стосується обробки зображень, була обрана бібліотека OpenCV-Python, яка є Python-обгорткою для оригінальної бібліотеки OpenCV, написаної на C/C++. Цей вибір був зроблений завдяки високій продуктивності та широкому спектру функціональності

3.2 Опис датасету

Ефективність роботи моделі та якість результату напряму залежить від якості та розміру датасету або даних, на яких вона навчається.

Розмір датасету має вирішальне значення для якості тренування моделей. Більші датасети забезпечують ширшу вибірку даних та можуть допомогти уникнути перенавчання. Вони можуть дати більшу різноманітність продуктів Тим не менш, тренування на великих датасетах вимагає більше обчислювальної потужності та часу, тому в умовах обмежених технічних ресурсів в рамках написання цієї роботи потрібно шукати баланс між об’ємом датасету та використаними ресурсами.

Поширеним є підхід використання синтетичні датасети для економії часу на збір даних. Для даної роботи цей підхід не підходить, тому що ми маємо контретну задачу, поставлену для конкретних наявних даних (продуктів), а також є доступний спосіб збору всієї потрібної інформації.

Для збору навчальних та тестових даних було розроблено окремий скрипт, також на мові програмування Python для уніфікації використаних технологій.

Джерелом даних слугувало GraphQL api, а саме метод productScroll, який дозволяє проітерувати всю наявну базу товарів частинами. Щоб скоротити час виконання було використано вбудовану бібліотеку asyncio та aiohttp для конкурентного завантаження даних по мережі. Отримані по мережі дані зберігаються в в файли бінарного формату HDF5 для економії пам’яті на диску.

Другий етап — це обробка текстових даних. Було застосовано доволі поширені підходи для обробки тексту: ловеркейсінг, видалення пробілів, розділових знаків, стеммінг, видалення стоп-слів. Усі атрибути, які було представлені масивом, були об’єднані в одну строку через спеціальні символи, які розділяють групи та значення атрибутів товару між собою. Також на даному етапі ми додаємо до датасету категорії поточного та вищого рівня. В свою чергу дерево категорій було окремо отримано з бази даних.

Останнім етапом підготовки даних є завантаження картинок. Для цього також було використано бібліотеки asyncio та aiohttp з тих самих причин, що й в попередніх пунктах.

Таким чином ми отримали датасет, який складається з 200 000 товарів, де кожен товар представлений картинкою, назвою, набором атрибутів, а також поточною категорією та категорією вищого рівня.

3.3 Опис моделі

В цьому розділі буде розглянуто архітектуру та всі компоненти побудованої системи.

3.3.1 Опис архітектури нейронної мережі

Проаналізувавши наявні методи обробки зображень та тексту, було розроблено архітектура нейронної мережі, яка представлена на схемі нижче.

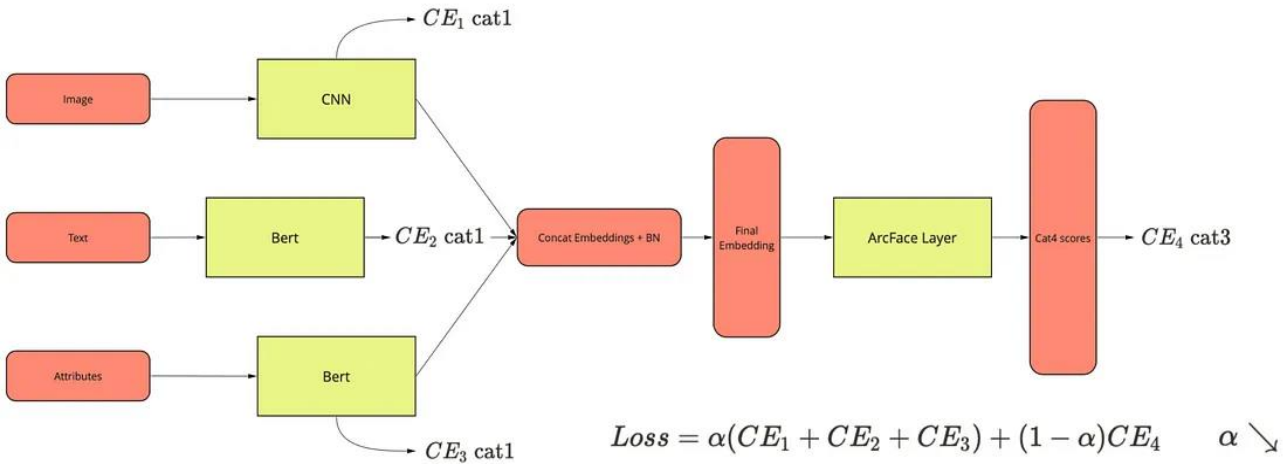

Рисунок 3.1 - Запропонована архітектура нейронної мережі

Розглянемо перший модуль у запропонованій архітектурі нейронної мережі — це нейронна мережа для обробки зображень. Вона виступає у якості енкодеру для зображення товару, і перетворює вхідний масив картинки на ембединг довжиною у 312 елементів. На сьогоднішній день існує декілька видів мереж, які підходять під дану задачу, такі як GoogleNet, MobileNet, EfficientNet VGGNet, ResNet та декілька інших. Для даниї роботи було обрано мережу ResNet34. На користь цього вибору вплинули наступні переваги:

Розв'язання проблеми зникання градієнтів: Найбільша перевага ResNet полягає у її здатності розв'язувати проблему зникання градієнтів у глибоких мережах. Це досягається завдяки використанню так званих "скіп-з'єднань" (skip connections), які дозволяють градієнтам прямо проходити через мережу.

Глибина мережі без втрати продуктивності: Завдяки цим скіп-з'єднанням, ResNet дозволяє побудову значно глибших мереж без втрати продуктивності. Глибші мережі здатні вчитися більш складних та абстрактних представлень даних.

Покращена навчальна здатність: ResNet може ефективно навчатися на великих масштабах даних та з комплексними задачами, що робить її ідеальною для задач, де потрібні глибокі знання та розуміння.

Широкий спектр застосувань: ResNet була успішно застосована у багатьох областях, включаючи розпізнавання образів, сегментацію зображень, обробку природної мови та інші.

Надійність: Оскільки ResNet була однією з перших глибоких мереж, яка показала чудові результати, вона добре вивчена, що робить її надійним вибором для багатьох додатків.

Нижче представлена архітектура мережі Resnet34 (рис.3.2).

Обрання ResNet34 серед інших версій ResNet було зумовлено її оптимальним співвідношенням розміру мережі та результатів, які вона показує. ResNet34, зі своїми 34 шарами, забезпечує відмінний баланс між складністю моделі та обчислювальною ефективністю, роблячи її ідеальною для ситуацій, де потрібна глибока архітектура, але ресурси обмежені. Ця версія ResNet дозволяє досягти високої точності у різноманітних задачах обробки зображень, не потребуючи при цьому такого обсягу обчислень, як її більш глибокі аналоги, такі як ResNet50 або ResNet101. Таким чином, ResNet34 стає нашим вибором, що забезпечує відмінне співвідношення продуктивності та ефективності, зберігаючи при цьому достатню потужність для більшості задач комп'ютерного зору, зменшуючи в той же час навантаження на обчислювальні ресурси. Нижче можна ознайомитись із іншими наявними версіями даної нейронної мережі.

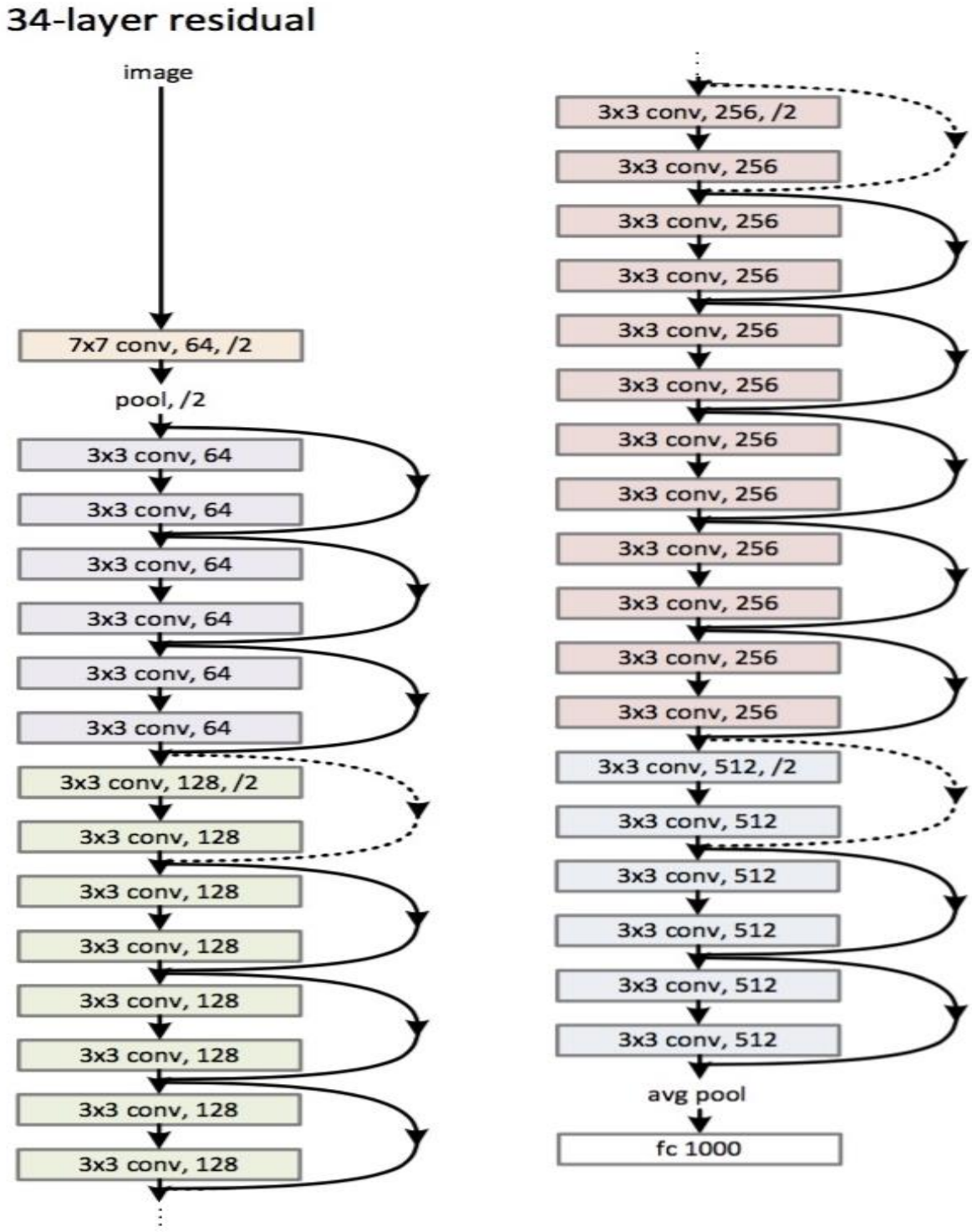

Рисунок 3.2 - Архітектура мережі Resnet34[6]

[6] Джерело зображення: https://neurohive.io/wp-content/uploads/2019/01/resnet-architecture-3.png

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рисунок 3.3 - Архітектура різних версій мережі Resnet[7]

Натупними модулями системи є дві нейронні мережі для обробки текстових даних. Серед різних наявних алгортимів NLP було обрано алгоритм BERT.

Метод BERT, що розшифровується як Bidirectional Encoder Representations from Transformers, є однією з найсучасніших моделей обробки природної мови (NLP), розробленою Google. BERT вирізняється тим, що він враховує контекст з обох сторін слова, а не тільки з лівої або правої, як багато інших моделей. Його застосовують наступним чином для персоналізації:

- **Розуміння Контенту**: BERT може аналізувати описи продуктів, відгуки користувачів та інший текстовий контент для створення високорівневого розуміння семантики і контексту. Це допомагає рекомендаційній системі краще зрозуміти, що саме подобається користувачам, і робити більш точні рекомендації.
- **Пошук і Фільтрація**: BERT може використовуватися для поліпшення пошукових запитів і фільтрації результатів на

[7] Джерело зображення: https://neurohive.io/wp-content/uploads/2019/01/resnet-architectures-34-101.png

маркетплейсах або в інших системах, де користувачі шукають певний контент.

- **Аналіз Відгуків**: Моделі, подібні до BERT, можуть допомагати аналізувати відгуки користувачів, визначати емоційний забарвлення і ключові аспекти продукту чи послуги, що в свою чергу може використовуватися для покращення рекомендацій.
- **Персоналізація**: Використовуючи глибоке розуміння тексту, BERT може допомогти створювати більш персоналізовані рекомендації, враховуючи індивідуальні переваги та інтереси користувача.
- **Використання в Чат-ботах**: BERT можна використовувати в чат-ботах для кращого розуміння запитів користувачів і надання релевантних відповідей, які, в свою чергу, можуть включати рекомендації продуктів або послуг.
- **Обробка Мов На Рівні Людської Мови**: BERT має здатність обробляти мову на рівні, наближеному до людського сприйняття, що робить його дуже потужним інструментом для рекомендаційних систем, які прагнуть до максимальної точності та релевантності.

Важливим фактором на користь обрання саме цієї нейронної мережі є наявність у відкритому доступі попередньо натренованих екземплярів під різні мови. В даному випадку ми потребуємо мережу навчено розпізнавати російську мову, адже усі назви та атрибути товарів спочатку створюються саме нею, а потім перекладаються на українську, тому щоб уникнути різних помилок, пов’язаних з перекладом було обрано саме мову оригіналу.

Враховуючи це все, BERT і подібні йому моделі стають все більш популярними в сфері рекомендаційних систем, оскільки вони дозволяють створювати більш точні, релевантні і персоналізовані рекомендації для користувачів. Нижче наведена схема робити алгоритму.

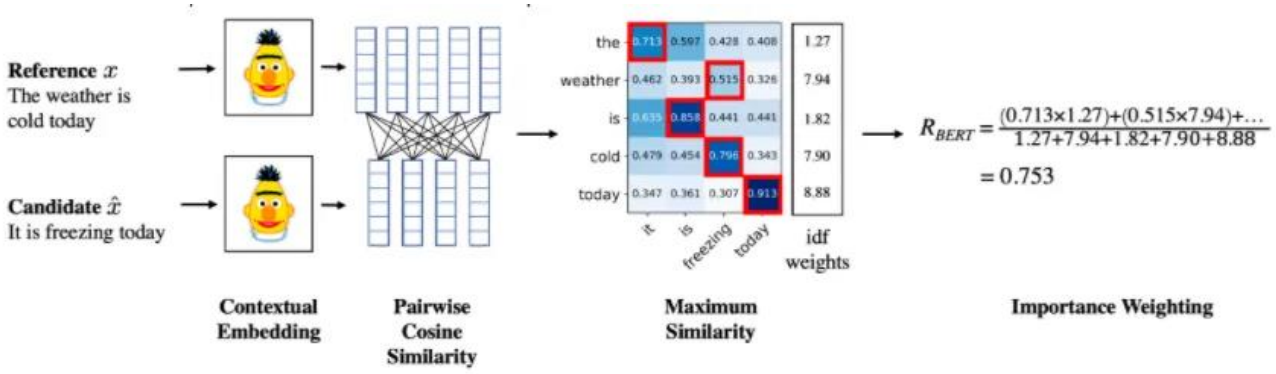

Рисунок 3.4 - Схема алгоритму BERT[8]

BERT має кілька ключових переваг у порівнянні з іншими моделями обробки природної мови, що робить його підходящим вибором для певних видів задач. Нижче наведені основні переваги BERT:

- **Двонаправлене Навчання на Контексті**: На відміну від попередніх моделей, які розглядали текст лінійно (зліва направо або навпаки), BERT аналізує контекст слів у обох напрямках. Це дозволяє моделі краще розуміти контекст та нюанси мови.
- **Загальна Предобробка**: BERT може бути предоброблений на великому корпусі тексту і потім налаштований для конкретних задач NLP, таких як класифікація тексту, відповіді на запитання, розпізнавання іменованих сутностей тощо. Це робить його гнучким та використовуваним у широкому спектрі застосувань.
- **Висока Точність**: BERT показав дуже високі результати на багатьох NLP задачах, перевершивши попередні моделі.
- **Обробка Складних Мовних Задач**: Завдяки своїй здатності до глибокого розуміння мовних контекстів, BERT ефективно справляється зі складними мовними завданнями, такими як виявлення сентименту, семантичний аналіз та інші.

[8] Джерело зображення: https://miro.medium.com/v2/resize:fit:720/format:webp/0*ViwaI3Vvbnd-CJSQ.png

Таким чином, з кожного енкодера ми отримуємо вихідні дані проміжного рівня — передбачення cat1 (категорії вищого рівня) і в процесі навчання додаємо ще один додатковий повністю зв’язаний шар, який потрібен для того, щоб обрахувати перехресну ентропію. Таким чином ми навчаємо наші енкодери передбачати категорію вищого рівня для вхідного продукта. Для даних трьох модулів був використаний метод оптимізації SGD (стохастичний градієнтний спуск) з кроком навчання рівному 0.001.

Наступною частиною моделі є повнозв’язний шар, який об’єднує вхідні вектори та видає один вектор, який і буде слугувати представленням продукту. Також на цьому етапі ми використовуємо підхід batch normalization.

Batch Normalization (BN) нормалізує вихід кожного шару за допомогою регулювання середнього значення та стандартного відхилення виходів. Він застосовується до активацій кожного шару перед нелінійною функцією активації. Нижче наведені основні кроки даного алгоритму.

1. **Визначення Середнього та Стандартного Відхилення**: Обчислюються середнє та стандартне відхилення активацій у пакеті (batch) даних.
2. **Нормалізація**: Вихід кожного нейрону нормалізується з урахуванням обчисленого середнього та стандартного відхилення.
3. **Масштабування та Зсув**: Після нормалізації застосовуються додаткові параметри масштабування та зсуву, які навчаються під час процесу навчання мережі.

Після цього слідує функція активації. У рамках експериментів було протестовано такі функції, як Softmax та ArcFace, яка використовується у задачах розпізнавання облич, тому що задачі векторизації продуктів та розпізнавання облич концептуально є дуже схожими, і використання даної функції може суттєво вплинути на результати моделі.

Після цього слідує ще один повнозв’язний шар, виходом якого є передбачення cat3 (категорії нижнього рівня). Цей шар тільки під час навчання для обрахунку перехресної ентропії.

Загальна функція втрати – це зважена сума всіх втрат, і спочатку ми проставляємо більшу вагу втратам для cat1, а потім поступово зміщуємо її до втрати cat3 за допомогою параметра α. Нижче наведена формула для обрахунку втрат.

$$Loss = \alpha\,(CE_1 + CE_2 + CE_3) + (1 - \alpha)\,CE_4\,, \qquad (3.1)$$

де CE_i $i \in \{1, 2, 3\}$ — перехресні ентропії перших трьох енкодерів

CE_4 - перехресна ентропія останнього шару.

α — спадає.

Після закінчення процесу навчання та переходу до тестування моделі нас більше не цікавить передбачення cat3, натомість цікавить векторне представлення продукту, тому ми зміщуємо вихід можелі до шару в ArcFace. В результаті ми отримуємо ембедінг, який і є веторним представленням вхідного товару.

Нижче схематично представлена повна архітектура готової нейронної мережі.

```
Layer (type:depth-idx)                   Param #
==========================================================================
├─ResNet: 1-1                            --
|    └─Conv2d: 2-1                       9,408
|    └─BatchNorm2d: 2-2                  128
|    └─ReLU: 2-3                         --
|    └─MaxPool2d: 2-4                    --
|    └─Sequential: 2-5                   --
|    |    └─BasicBlock: 3-1              73,984
|    |    └─BasicBlock: 3-2              73,984
|    |    └─BasicBlock: 3-3              73,984
|    └─Sequential: 2-6                   --
|    |    └─BasicBlock: 3-4              230,144
```

```
|    |    └─BasicBlock: 3-5           295,424
|    |    └─BasicBlock: 3-6           295,424
|    |    └─BasicBlock: 3-7           295,424
|    └─Sequential: 2-7                --
|    |    └─BasicBlock: 3-8           919,040
|    |    └─BasicBlock: 3-9           1,180,672
|    |    └─BasicBlock: 3-10          1,180,672
|    |    └─BasicBlock: 3-11          1,180,672
|    |    └─BasicBlock: 3-12          1,180,672
|    |    └─BasicBlock: 3-13          1,180,672
|    └─Sequential: 2-8                --
|    |    └─BasicBlock: 3-14          3,673,088
|    |    └─BasicBlock: 3-15          4,720,640
|    |    └─BasicBlock: 3-16          4,720,640
|    └─AdaptiveAvgPool2d: 2-9         --
|    └─Linear: 2-10                   160,056

├─BertModel: 1-2                     --
|    └─BertEmbeddings: 2-11          --
|    |    └─Embedding: 3-17          26,154,336
|    |    └─Embedding: 3-18          638,976
|    |    └─Embedding: 3-19          624
|    |    └─LayerNorm: 3-20          624
|    |    └─Dropout: 3-21            --
|    └─BertEncoder: 2-12             --
|    |    └─ModuleList: 3-22         2,301,552
|    └─BertPooler: 2-13              --
|    |    └─Linear: 3-23             97,656
```

```
|    |    └─Tanh: 3-24                --
├─BertModel: 1-3                      --
|    └─BertEmbeddings: 2-14           --
|    |    └─Embedding: 3-25           26,154,336
|    |    └─Embedding: 3-26           638,976
|    |    └─Embedding: 3-27           624
|    |    └─LayerNorm: 3-28           624
|    |    └─Dropout: 3-29             --
|    └─BertEncoder: 2-15              --
|    |    └─ModuleList: 3-30          2,301,552
|    └─BertPooler: 2-16               --
|    |    └─Linear: 3-31              97,656
|    |    └─Tanh: 3-32                --
├─Linear: 1-4                        61,974
├─Linear: 1-5                        61,974
├─Linear: 1-6                        61,974
├─BatchNorm1d: 1-7                   1,872
├─Linear: 1-8                        239,872
├─ArcFace: 1-9                       65,536
├─Linear: 1-10                       422,251
==========================================================================
Total params: 80,747,717
Trainable params: 80,747,717
Non-trainable params: 0
```

3.3.2 Опис алгоритму пошуку суміжних векторів

Надалі перед нами постає задача пошуку найближчих векторів у векторному просторі, тому з’являється потреба у методах, які можуть вирішити цю задачу. Ці методи можуть варіюватися від простих алгоритмів, як лінійний пошук у векторних просторах, до складніших, таких як дерева кластеризації або хешування. Серед них, бібліотека Faiss (Fast Approximate Nearest Neighbor Search Library) - це бібліотека для ефективного пошуку за схожістю та кластеризації щільних векторів. Вона містить алгоритми, які шукають у наборах векторів будь-якого розміру, включаючи ті, що можливо не поміщаються в оперативній пам'яті. Також у бібліотеці є підтримуючий код для оцінки та налаштування параметрів. Faiss написаний на C++ з повними обгортками для Python/numpy. Деякі з найбільш корисних алгоритмів реалізовані на GPU. Її основна розробка ведеться в групі Fundamental AI Research компанії Meta.

Faiss було обрано завдяки її здатності швидко обробляти великі набори даних, забезпечуючи при цьому точні та ефективні результати. Її оптимізація під GPU також робить її привабливою для використання у середовищах із високою обчислювальною потужністю, що є критично важливим для обробки великих обсягів даних. Також, гнучкість Faiss у підтримці різних типів індексів та методів навчання (включаючи некероване навчання) робить її підходящою для широкого спектра застосувань.

Faiss містить кілька методів для пошуку за схожістю. Припускається, що екземпляри представлені у вигляді векторів і ідентифіковані цілим числом, а вектори можуть бути порівняні за допомогою L2 (евклідової) відстані або скалярного добутку. Вектори, схожі на запитуваний вектор, це ті, які мають найменшу L2 відстань або найвищий скалярний добуток з запитуваним вектором. Також підтримується косинусна схожість, оскільки це скалярний добуток на нормалізованих векторах.

Деякі з методів, як наприклад ті, що базуються на бінарних векторах та компактних квантованих кодах, використовують лише стисле представлення векторів і не вимагають зберігання оригінальних векторів. Це, як правило, призводить до менш точного пошуку, але ці методи можуть масштабуватися до мільярдів векторів у головній пам'яті на одному сервері. Інші методи, як HNSW та NSG, додають структуру індексації поверх сирцевих векторів для більш ефективного пошуку.

GPU реалізація може приймати вхідні дані як з CPU, так і з GPU пам'яті. На сервері з GPU, індекси GPU можуть використовуватися як заміна індексам CPU (наприклад, замінити IndexFlatL2 на GpuIndexFlatL2), а копіювання до/з GPU пам'яті виконується автоматично. Результати будуть швидшими, якщо як вхідні, так і вихідні дані залишаються в пам'яті GPU. Підтримується як одно-, так і багатопроцесорне використання GPU.

Faiss побудований навколо типу індексу, який зберігає набір векторів та надає функцію для пошуку серед них за допомогою порівняння векторів L2 та/або скалярного добутку. Деякі типи індексів є простими базовими варіантами, такими як точний пошук. Більшість доступних структур індексації відповідають різним компромісам стосовно:

- часу пошуку;
- якості пошуку;
- пам'яті, використовуваної на вектор індексу;
- часу навчання;
- часу додавання;
- потреби в зовнішніх даних для некерованого навчання.

Опціональна реалізація GPU надає те, що, ймовірно, є найшвидшою реалізацією точного та наближеного (у стислому домені) пошуку найближчих сусідів для векторів високої розмірності, найшвидшим алгоритмом кластеризації Lloyd's k-means та найшвидшим алгоритмом small k-selection.

В рамках наших потреб будемо використовувати індекс IndexFlatL2. IndexFlatL2 вимірює L2 (або евклідову) відстань між усіма заданими точками між нашим вектором запиту та векторами, завантаженими в індекс. Це просто, дуже точно, але не надто швидко, проте в рамках заданих об’ємів даних буде оптимальним варіантом, адже з’економить час на навчанні індекса. Нижче наведена формула обрахунку евклідової відстані (рис. 3.5).

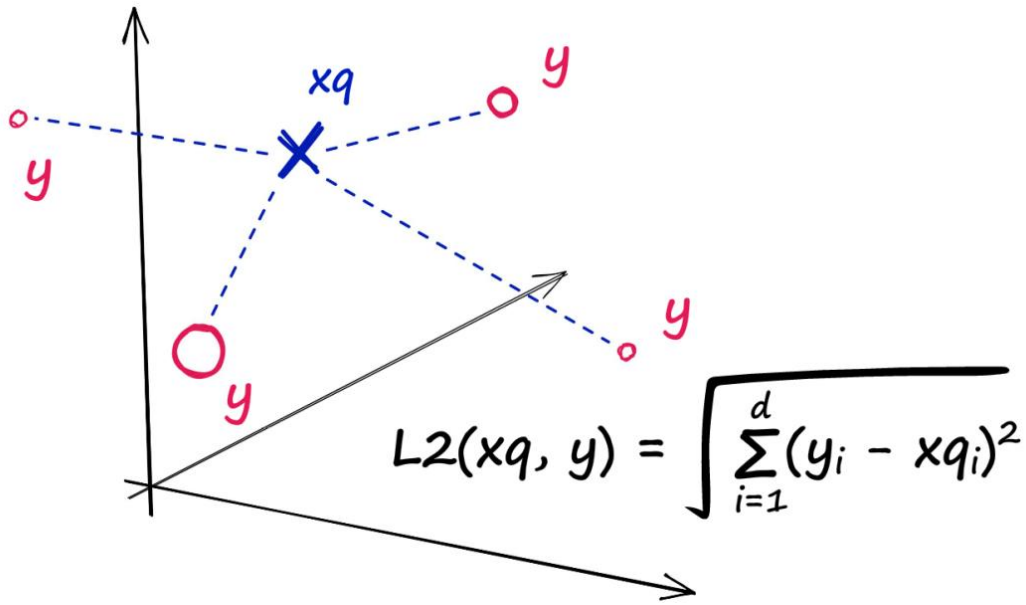

Рисунок 3.5 - Обрахунок евклідової відстані[9]

3.4 Процес навчання

Розмір вибірки складає 140000 продуктів. Розділимо її на тренувальну та тестову підвибірки у співвідношенні 80 до 20 відповідно. Тренувальні та тестові вибірки важливі в машинному навчанні для ефективного тренування та оцінки моделей. Тренувальна вибірка використовується для навчання моделі на відомих даних, тоді як тестова - для оцінки її точності на нових даних. Це допомагає

[9] Джерело зображення: https://cdn.sanity.io/images/vr8gru94/production/ea951a4be3acf9d379cc6f922be1468b37b7f9e5-1280x720.png

виявити перенавчання та гіперпараметри, забезпечуючи, що модель здатна ефективно узагальнювати на реальних завданнях. Навчання нейронної мережі буде відбуватися за допомогою батчінга. Розмір батча дорівнює 32. Для цього будемо використовувати вбудовані інструменти фреймворку Pytorch: Dataset та Dataloader. Вони дозволяють ефективно завантажувати великі об'єми даних за допомогою технології мультипроцесності, що дозволяє паралельно виконувати декілька задач одночасно.

Кільтістю навчальних було встановлено 10, за цей час модель має достатньо добре вивчити паттерни у даних та показувати задовільні результати. З кожною епохою будемо зменшувати коефі ціент α у функції втрат, таким чином зміщуємо акцент у навчанні з перших трьох енкодерів на останню частину моделі по мірі збільшення номеру епохи.

Розглнемо формулу для обрахунку коефіцієнту альфа.

$$\alpha = \alpha_0 * e^{\frac{-n_i}{n}}, \tag{3.2}$$

де α_0 — початкове значення α,

n_i — номер поточної епохи,

n — загаліна кількість епох.

Початкове значення параметра α встановлюємо як 0.5. Нижче на рис. 3.6 наведено графік залежності α від епохи.

Також розглянемо показники функції помилки в процесі навчання. Результати показані у табл. 3.1.

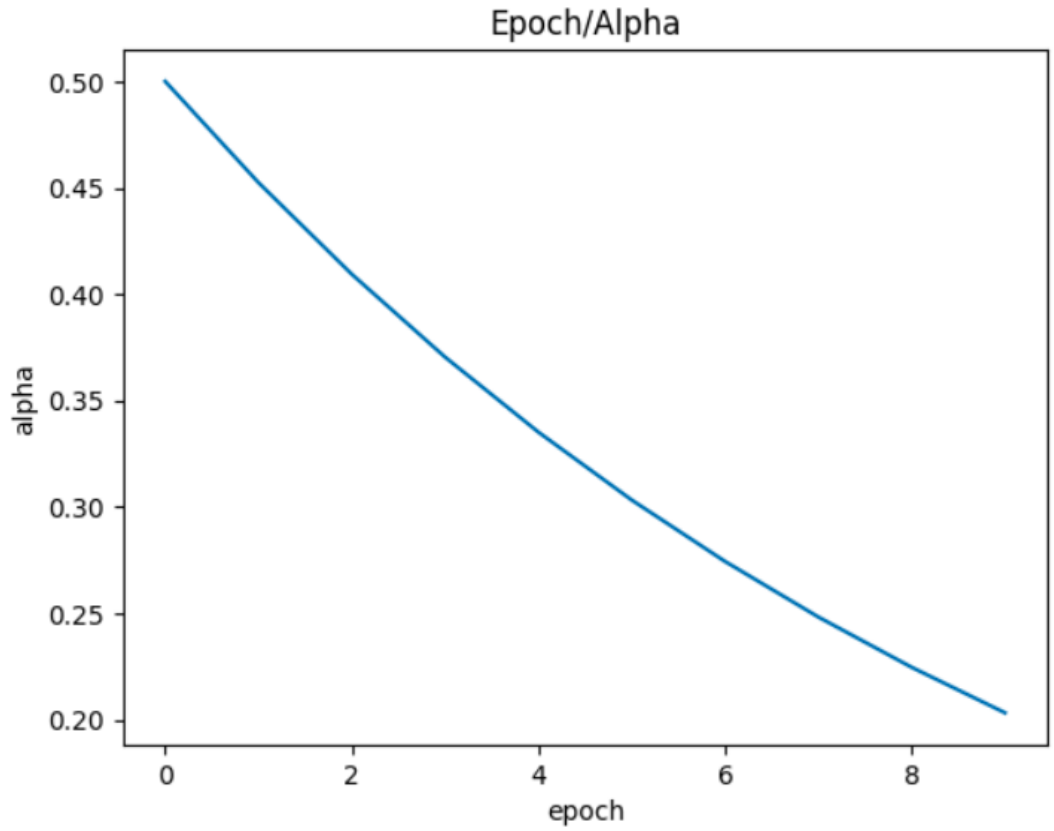

Рисунок 3.6 - Графік залежності альфа від епохи

Таблиця 3.1 - Показники функції втрат в процесі навчання

Епоха	*Loss*
1	13.053
2	11.823
3	10.14
4	9.678
5	8.742
6	8.139
7	7.564
8	6.21
9	6.136
10	5.426

3.5 Отримані результати

Розглянемо результати точності моделі нейронної мережі для надання рекомендацій.

Оцінювати модель будемо за наступними метриками: Precision@k, MAP@k, AP@K, NDCG.

Також в рамках експерименту протестуємо 2 види функції активації. Класичний підхід Softmax не має прямого впливу на відстань між вивченими векторами в межах одного класу та віддаленість між класами. У випадку ArcFace це було спеціально розроблено для цього: шляхом налаштування параметра штрафу за відступ m ми можемо контролювати близькість або віддаленість векторів в межах одного чи різних класів.

Формула функції активації softmax:

$$\sigma(z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}}, \tag{3.3}$$

де $Z = (Z_1, ..., Z_k) \in R^k$ – вхідний вектор,

K – розмірність вхідного вектору

i = 1, ..., K.

Формула функції автивації ArcFace:

$$-\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s*(\cos(\theta_{y_i}+m))}}{e^{s*(\cos(\theta_{y_i}+m))} + \sum_{j=1,j\neq y_i}^{n} e^{s*cos\theta_j}}, \tag{3.4}$$

де θ_j - кут між відповідним ваговим коефіцієнтом та ознакою

s – радіус гіперсфери

m - покарання за кутовий запас.

Для обрахування метрик візьмемо заздалегідь підготовлені дані про товари, які не переглядали, переглядали та купували різні користувачі. Далі побудуємо для них векторне представлення, по якому і будемо знаходити схожі товари.

Оцінювати релевантність будемо за допомогою метрик: P@K, MAP@K, AP@K.

Будемо вважати релевантність 0 або 1 в залежності релевантний чи елемент чи ні. Релевантним вважаємо елемент, який користувач переглядав або купував.

Для метрики NDCG будемо вважати релевантність елементу 0, 0.5, 1, що означає не переглянутий товар, переглянутий товар та товар, для якого було оформлене замовлення.

Оцінювати будемо послідосність з 10 рекомендованих товарів. Для метрик, які передбачають середнє по вибірці, будемо використовувати послідовності для 20 користувачів.

Нижче наведені отримані показники метрик для різних функцій активації (табл. 3.2).

Таблиця 3.2 – Метрики побудованої моделі.

Функція	*P@10*	*MAP@20*	*AP@10*	*NDCG*
Softmax	0.2	0.28	0.3	1.178
ArcFace	0.3	0.375	0.45	1.2265

Як видно з отриманих метрик можемо зробити висновок, що модель з функцією активації ArcFace показала себе краще, ніж функція активації Softmax. Це пояснюється тим, шо класичний Softmax безпосередньо не впливає на близькість вивченого вкладеності в межах одного класу і віддаленості в різних класах. ArcFace розроблено спеціально для цього: шляхом вибору маржинального штрафу m параметр, ми можемо налаштувати закриття/віддалення вбудовування того самого або різні класи. Також можна

відмітити, що модель загалом показує доволі гарні результати як для метрик, які просто враховують кількість корисних рекомендацій, так і для тих, які є більш просунутими і граховують не тільки кількість, а й позицію та ступіню того, наскільки могла би бути корисна конкретна рекомендація.

Висновки до розділу 3

У данному розділі роботи було проведено детальний аналіз побудови та результатів рекомендаційної системи, використовуючи підхід prod2vec. Було виявлено, що використання моделі prod2vec дозволяє ефективно моделювати зв'язки між схожими продуктами та забезпечувати точні та персоналізовані рекомендації.

Результати експериментів підтверджують доволі високу точність розробленої рекомендаційної системи. Модель демонструє здатність ефективно враховувати контекстні взаємодії між продуктами, а саме через категорії товарів, що призводить до покращення якості рекомендацій. В якості експерименту було проведено порівняльний аналіз двох функцій активації: Softmax та Arcface. В результаті було виявлено, що модель з використанням Arcface справляється краще з поставленою задачею та має позитивний вплив на результати нейронної мережі.

Компромісним компонентом побудованої системи є алгоритм пошуку сусідніх векторів, де було зроблено ставку на простоту використання, точність та економію ресурсів. В подальшому, цей момент можна покращити за рахунок використання інших методів та алгоритмів.

Рекомендаційна система, побудована на основі prod2vec, може знайти широке застосування у різних галузях, де необхідно надавати персоналізовані поради та рекомендації користувачам. Високі показники точності та ефективності роблять цей підхід привабливим для впровадження в реальних умовах, покращуючи користувацький досвід та збільшуючи задоволеність від використання системи.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

У цьому розділі буде проаналізовано те, якими шляхами можна реалізувати побудовану модель у вигляді реального стартап-проекту.

4.1 Опис ідеї проекту

"E-Commerce Insight" представляє собою новаторське рішення в сфері SaaS (Software as a Service), яке створене з метою трансформувати та удосконалити процес покупок в області електронної комерції. Основною особливістю цього сервісу є його здатність використовувати передові алгоритми машинного навчання та глибокий аналіз даних для створення точних та ефективних рекомендацій, що враховують унікальні потреби та переваги кожного користувача.

Унікальність "E-Commerce Insight" полягає в його здатності адаптуватися до широкого спектра товарів та послуг, що пропонуються в електронній комерції. Багато продуктів в цій сфері мають схожі набори характеристик та атрибутів, що створює сприятливі умови для розробки універсальних алгоритмів рекомендацій. Ці алгоритми можуть бути ефективно застосовані в різних інтернет-магазинах з урахуванням їх специфіки та потреб.

Ключовим аспектом є здатність системи аналізувати та обробляти великі об'єми даних, що стосуються поведінки споживачів, попередніх покупок, відгуків та інших важливих метрик. Це дозволяє "E-Commerce Insight" не тільки визначати поточні тренди та популярні продукти, але й прогнозувати майбутні зміни в споживацьких перевагах.

Додатково, система може інтегруватися з різними платформами електронної комерції, що дозволяє їй враховувати специфіку кожного онлайн-магазину. Використання машинного навчання допомагає сервісу "E-Commerce

Insight" постійно вчитися на поведінкових шаблонах користувачів, що робить рекомендації все більш точними та релевантними з часом.

Таблиця 4.1 - Опис ідеї стартап-проекту.

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Використання універсального алгоритму надання рекомендацій для торгових майданчиків, що базується на спільних властивостях товарів та послуг, але в той самий час може бути розширений під потреби кожного клієнта	Сервіс можна примінити у будь-якому напрямку електронної комерції, для клієнтів, які хочу покращити користувацький досвід, але не мають ресурсу на розробку власного рішення	Клієнти отримують кращий користувацький досвід, а також розвивають лояльність до свого бренду. В той самий час інтеграція із запропонованим сервісом не вимагає багато зусиль та ресурсів

Мета проекту "E-Commerce Insight" полягає у вирішенні критичної проблеми, яка стоїть перед сучасними e-commerce бізнесами: потреба в оптимізації та персоналізації процесу покупки для максимального задоволення потреб споживачів та збільшення ефективності продажів. У світі, де вибір продуктів є величезним і часто перевантажує споживачів, "E-Commerce Insight" намагається створити рішення, яке здатне інтелектуально аналізувати поведінку покупців та переваги, пропонуючи їм найбільш підходящі та цікаві товари.

Основною задачею є досягти плавного та інтуїтивно зрозумілого шопінг-досвіду для користувачів, водночас збільшуючи конверсію продажів та лояльність клієнтів для онлайн-магазинів.

"E-Commerce Insight" прагне стати не тільки інструментом для підвищення продажів, але й ключовим партнером для бізнесів у розумінні та задоволенні потреб їхніх клієнтів, враховуючи зростаючу важливість персоналізованого підходу в сфері електронної комерції.

Далі проаналізуємо та визначимо сильні, слабкі та нейтральних характеристик ідеї проекту (табл. 4.2).

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик проекту.

№ п/п	Техніко-Економічні характеристики ідеї	Потенційні товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Recombee	Algolia Recommend			
1	Універсальниі рекомендації	Так	Так	Так		+	
2	Швидкість роботи	Задовільна	Задовільна	Достатня			+
3	Доступність та простота у використанні	Висока	Висока	Посередня		+	
4	Ціна	$70/місяць	$100/місяць	$0.50/1K запитів/місяць			+

На основі табл. 4.2 можна зробити висновок, що розроблена система є конкурентоспроможною.

4.2 Технологічний аудит проекту

Основною метою проведення технічного аудиту є ідентифікація та оцінка комплексу технологічних рішень, які були використані для створення системи відповідно до концепції проекту. Аналіз технічної втіленості концепту проекту включає в себе дослідження наступних елементів (табл. 4.3):

- які технологічні інструменти були використані для розробки системи, що відповідає основній ідеї проекту;
- доступність цих технологій на ринку: чи є вони вільними для використання, чи вимагають вони додаткової розробки або придбання;
- чи мають розробники можливість доступу та використання зазначених технологій для реалізації проекту.

Таблиця 4.3 - Технологічна здійсненність ідеї проекту.

№п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Створення сервісу для надання товарних рекомендацій	Pytorch	Наявна	Доступна
2		Resnet34	Потребує донавчання	Доступна
3		RuBERT	Потребує донавчання	Доступна
4		Facebook Faiss	Наявна	Доступна
Обраною мовою програмування є Python.				

Після проведеного аналізу можливо встановити, що технічне втілення проекту є виконаним. Для технічної реалізації проекту було вибрано програмування на Python через наявність реалізацій потрібних бібліотек та гарну екосистему вцілому.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Перш за все проведемо аналіз попиту, а результати наведено у табл. 4.4 За результатами аналізу табл. 4.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики (табл. 4.5). Трасформовано орієнтовний перелік вимог до товару для кожної групи.

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап проекту.

№ п/п	Показники стану ринку(найменування)	Характеристика
1	Кількість систем-конкурентів на ринку	Недостатня
2	Загальний обсяг продаж, грн/ум.од	Невідомо
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту.

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Потреба пошуку релевантних товарів та послуг, а також наданні точних рекомендацій	Пересічний користувач торгівельного онлайн майданчика	Цільові групи клієнтів мають різні вподобання.	Система пропонує товари, які відповідають їм поточним потребам, а також вони змінюються з часом
Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару

Закінчення табл. 4.5

Потреба простого в інтеграції та ефективного сервісу рекомендацій	Розробники та власники інтернет магазинів	Вимагають простого і зрозумілого способу користування сервісом	Наявність API, яке б не вимагало великих витрат на додаткову розробку
Доступна ціна сервісу	Власники інтернет магазинів	Цільові групи клієнтів мають різний бюджет	Доступність сервісу для середньостатистичного власника бізнесу у сфері e commerce

Після аналізу потенційної цільової аудиторії було виконано обстеження ринкових умов: були створені таблиці, які відображають фактори, сприятливі для запуску розробленої системи у вигляді стартапу, а також фактори, які можуть утруднити цей процес (табл. 4.6, 4.7).

Таблиця 4.6 - Фактори загроз.

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкурентність на ринку	Поява на ринку конкурента, який може зайняти нішу через значну кількість ресурсів	Покращення точності моделі, та вдосконалення продукту вцілому
2	Економічний	Збільшення вартості на інфраструктуру, яка потрібна для функціонування сервісу	Знайти баланс між витратами на інфраструктуру та результатом, а також оптимізувати наявне рішення
3	Недостатня відомість	Відсутність попиту від бізнесу	Налагоджувати зв’язки та проводити рекламні компанії

Закінчення табл. 4.6

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
4	Недостатність реальних даних	Точність нейронних мереж залежить від даних на яких вона вчилась. Нажаль зібрати велику кількість реальних, не синтетичних даних доволі складно.	Знаходити різні джерела даних, для цього можна найняти окрему команду
5	Якість прогнозів на реальних даних	Точність рекомендацій на робочих даних може бути нижчою, ніж очікувалось	Постійне удосконалення моделі

Таблиця 4.7 - Фактори можливостей.

№п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Розвиток технологій	Швидкий розвиток штучного інтелекту та обробки природної мови створює нові можливості для покращення точності розпізнавання послідовностей	Впровадження нових ідей та розробок для покращення моделі
2	Ринковий попит	Зростання інтересу та потреби до товарних рекомендацій	Реклама сервісу для залучення більшої кількості клієнтів.
3	Застосування інших моделей	Застосування моделей інших типів моделей для підвищення точності отриманих прогнозів.	Витрачати ресурси на дослідження нових моделей та підходів для надання рекомендацій.

Наступним було досліджено пропозиції: визначили загальні характеристик конкуренції на ринку (табл. 4.8).

Таблиця 4.8 — Ступеневий аналіз конкуренції на ринку.

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - чиста	Різні незалежних компаній на рівних умовах конкурують одна з одною, але жодна з них не домінує на ринку	Потрібно створити конкурентоспроможний продукт, що задовольняє потреби користувача та донести йому переваги власного рішення
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти існують в різних країнах	Співпраця з іноземними спеціалістами, обмін досвідом, а також розширення стартапу та вихід на ринки інших країн
3.За галузевою ознакою - міжгалузева	Багато компаній пропонують послуги в різних галузях.	Постійне вдосконалення системи
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між унікальними характеристиками різних конкуруючих стартапів	Створення продукту, враховуючи позитивні сторони та недоліки конкурентів.
5. За характером конкурентних переваг - цінова	Різниця в ціні на конкурентні продукти на ринку	Забезпечення адекватної ціни за користування сервісом
6. За інтенсивністю - марочна	Серед аналогів є компанії з популярним брендом	Робота над створенням та просування власного бренду

Проведено більш детальний аналіз умов конкуренції в галузі за М. Портером у табл. 4.9.

Таблиця 4.9 - Аналіз конкуренції в галузі за М. Портером.

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження на ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
	Recombee, Amazon, Algolia Recommend	На ринку вже наявні схожі рішення	Доступ відбувається через мережі Інтернет	Якість запропонованого	Кращий функціонал, кращі результати
Висновок	Достатньо інтенсивна конкуренція на ринку із компаніями, що пропонують методи оптичного розпізнавання символів	Є можливість виходу на ринок, проте наявна конкуренція	Відсутні фактори сили постачальників	Клієнти диктують умови на ринку	Необхідно постійно вдосконалювати роботу моделі, робити швидшою та доступнішою

Після проведення аналізу, було встановлено, що робота на ринку є можливою враховуючи існуючу конкуренцію. Такий висновок був включений у процес визначення ключових факторів конкурентоспроможності, які будуть детально розглянуті в наступному розділі. Визначення та обґрунтування цих факторів базується на аналізі конкурентного середовища, представленому у таблиці, а також з урахуванням інформації з попередніх таблиць. Перелік факторів конкурентоспроможності визначається та обґрунтовується та наводиться у табл. 4.10

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності.

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Функціонал підходу	Підхід має задовольняти потреби користувачів і давати їм можливість отримати бажаний результат
2	Інноваційність	Рішення має постійно покращуватися, щоб мати перевагу у якості наданих послуг
3	Ціна	Ціна має бути доступною і привабливою у порівнянні з конкурентами

За визначеними факторами конкурентоспроможності (табл. 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту, що наведені у табл. 4.11.

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін проекту.

№ п/п	Фактор конкурентоспроможності	Бали 1-20						
			-3	-2	-1	0	1	2	3
1	Функціонал сервісу	17			+				
2	Інноваційність	10				+			
3	Ціна	14		+					

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання матриці SWOT-аналізу (сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities)). Матриця SWOT-аналізу наведена у табл. 4.12. Встановлені в рамках SWOT-аналізу альтернативи щодо ринкової стратегії для введення проекту на ринок детально розглядаються в контексті їхніх строків реалізації та ймовірності успішного здобуття необхідних ресурсів. Цей аналіз представлений у табл. 4.13, де кожна альтернатива визначається не лише тим, як вона відповідає можливостям і загрозам ринкового середовища, але й оцінюється з погляду реалізаційних термінів та ймовірності отримання необхідних ресурсів. Це дозволяє не лише

ідентифікувати потенційні ризики, але й створює стратегічну основу для прийняття обґрунтованих та ефективних рішень.

Таблиця 4.12 - SWOT- аналіз стартап-проекту.

Сильні сторони: Проект має зручну та не дуже коштовну в плані інтеграції реалізацію, а також доволі високу якість рекомендацій.	Слабкі сторони: високий рівень конкуренції, складність збирання даних, а також розробкою малочисельною командою
Можливості: розширення функціоналу шляхом додавання нових моделей та підходів, також розробка рішень для монетизації через інші джерела трафіку, не тільки рекомендації.	Загрози: швидке зростання конкуренції. Вірогідність виходу на ринок аналогу розробленого великою компанією, що призведе до появи біль привабливого конкурента, динамічний розвиток сфери, що вимагає постійного підлаштовуватись під нинішні реалії

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту.

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Вихід на ринок із програмним продуктом з додатковим функціоналом	90%	4 місяці
Створення продукту для співпраці з великими гравцями ринку електронної комерції	85%	9 місяців
Розширення функціоналу продукту	80%	6 місяців

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії передбачає визначення стратегії охоплення ринку: дослідження цільових груп потенційних клієнтів, які приведенні в табл. 4.14.

Таблиця 4.14 - Вибір цільових груп потенційних споживачів.

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
Покупці в мережі інтернет	Висока	Високий	Висока	Проста
Бізнеси в сфері ecommerce	Середня	Середій	Висока	Висока складність
Інші компанії або стартап проекти	Середня	Низький	Середня	Середня складність

У результаті будемо взаємодіяти з 1, 2 цільовою групою. Для роботи в обраному сегменті ринку необхідно сформувати базову стратегію розвитку. Особливості вибори стратегії відображено у табл. 4.15.

Таблиця 4.15 - Визначення базової стратегії розвитку.

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Інтеграція із відомими торгівельними майданчиками	Диференційованого маркетингу	Масштабування та максимізація	Стратегія спеціалізації

Обґрунтування вибору стратегії конкурентної поведінки наводиться у табл. 4.16. На основі вимог споживачів з обраних сегментів до стартап-проекту та до продукту та в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки, наведеної у табл. 4.17, описуються шляхи розроблення стратегії позиціонування.

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки.

Чи є проект «першопроходьцем» 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
Ні	Компанія буде шукати нових споживачів і конкурувати за вже існуючих	Ні	Стратегія зайняття конкурентної ніші

Таблиця 4.17 - Визначення стратегії позиціонування.

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати позицію власного проекту
Точність результату, простота інтеграції, доступна ціна	Стратегія диференціації	Універсальність моделі, доступність до претренованих моделей для окремих ніш бізнесі, наприклад, магазин одягу, а також доступна ціна	Точність результатів, простота інтеграції, доступність

4.5 Розроблення маркетингової програми стартап-проекту

Після визначення маркетингової концепції продукту, яка передбачає його унікальні характеристики та цільову аудиторію, у розділі наведеної таблиці 4.18 відображено результати аналізу конкурентоспроможності товару. Даний аналіз враховує широкий спектр факторів, що включають у себе якість продукту, ціноутворення, стратегії маркетингу конкурентів та їхній ринковий вплив.

Здійснюючи детальний огляд цих аспектів, ми можемо чітко визначити ключові переваги та слабкі сторони нашого продукту в порівнянні з аналогічними пропозиціями на ринку.

На основі цього аналізу формується стратегія маркетингу для стартап-проекту, яка враховує змогу використання сильних сторін продукту (табл. 4.18).

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару.

Потреба	Вигода, яку пропонує товар	Ключові переваги над конкурентами (існуючі або такі, що потрібно створити)
Якість рекомендацій	Якісні рекомендації	Постійна робота над покращенням моделі
Доступність	Невеликі об’єми витрат дозволяє зробити продукт доступним навіть для невеликих бізнесів	Ціна за послугу чи підписку
Простота інтеграції у проект	Просте API, за допомогою якого можна використати сервіс у власному рішенні	Достатньо технічної документації, щоб почати користуватись сервісом

У табл. 4.19 описано трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання , а також основні переваги, які споживач отримує при виборі даного товару чи послуги

Така трирівнева модель дозволяє комплексно вивчати та управляти всім життєвим циклом продукту чи послуги, забезпечуючи максимальну відповідність їхніх характеристик очікуванням ринку та споживачів.

Таблиця 4.19 - Опис трьох рівнів моделі товару.

Рівні товару	Сутність і складові
I. Товар за задумом	Рекомендаційна система у вигляді SaaS рішення для інтеграцій у інтернет магазини, що базується на контенті товарів та послуг, а також можливість використовувати знання зі схожих проектів
II. Товар у реальному виконанні	Програмне забезпечення яке буде доступним для користування і буде надавати точні результати
III. Товар з підкріпленням	Постійне оновлення функціоналу, покращення роботи і результатів
За рахунок чого потенційний товар буде захищено від копіювання: ліцензія, патент, захист інтелектуальної власності	

У табл. 4.20 проведено визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, а також аналіз рівня доходів цільової групи користувачів продукту.

Таблиця 4.20 - Визначення меж встановлення ціни.

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Немає даних	Немає даних	10000$/місяць	150-300$/місяць

У табл. 4.21 обґрунтовано визначення оптимальної системи збуту, в межах якого приймається рішення.

Таблиця 4.21 – Формування системи збуту.

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Підписка для доступу до розширеного функціоналу	Продаж	Однорівневий	Власними силами та через посередників
Кастомізація моделі під конкретного клієнта	Продаж	Однорівневий	Власними силами та через посередників

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування. Розроблення концепції маркетингових комунікацій проводиться у табл. 4.22.

Таблиця 4.22 – Концепція маркетингових комунікацій.

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Інтеграція з іншими системами	Веб-сайти, торгові майданчики	Доступна ціна, задовільна якість, простота використання	Переконання оптимальності продукту для бізнесу клієнта	Порівняння сервісу з іншими аналогами
Індивідуальні користувачі: покупка за допомогою веб-сервісів	Соціальні мережі, веб-сайти, торгові майданчики	Доступна ціна, задовільна якість, простота використання	Наголошення на перевагах продукту перед конкурентами	Порівняння сервісу з іншими аналогами

Висновки до розділу 4

У цьому розділі виконано вступний аналіз задля запуску розробленої системи у вигляді стартапу. Була вироблена основна концепція стартапу (система рекомендацій, заснована на контенті, як SaaS рішення), проведено порівняльний аналіз сильних та слабких сторін у порівнянні з можливими конкурентами, який показав актуальність цієї ідеї. Для реалізації проекту були вибрані відповідні технології, зокрема програмування на Python і використання бібліотеки машинного навчання Pytorch. Ринок та конкуруючі продукти були досліджені, визначені потреби споживачів і загрози, що впливають на ринок. Здійснено аналіз конкурентного середовища в секторі, виокремлено ключові фактори конкурентоспроможності та проведено SWOT-аналіз. Аналізувались потенційні ризики і можливості, а також були розраховані основні фінансові та економічні показники проекту, що підтверджують його доцільність.

Було встановлено, що ринок є відкритим для нових пропозицій, а існуючі конкуренти не володіють всіма необхідними для користувачів характеристиками та функціями. Окрім впровадження у власній системі, розглядається можливість інтеграції з проектами потенційних конкурентів. Ураховуючи проведене дослідження, можна стверджувати, що подальший розвиток проекту є обґрунтованим, оскільки він має потенціал знайти свою аудиторію та зайняти нішу на ринку.

ВИСНОВКИ

У цьому дослідженні було вивчено різноманітні методи формування рекомендацій, і було вдосконалено їх через впровадження модифікацій.

Проведено аналіз актуальності поставленої задачі, оцінено сучасний стан предметної області, проаналізовано існуючі підходи, вивчено сучасні напрямки досліджень та розглянуто методи та алгоритми, що використовуються для надання рекомендацій.

У результаті були виявлені переваги та недоліки кожного методу, що дозволило визначити оптимальний шлях вдосконалення та адаптації підходів для вирішення конкретних завдань. Особлива увага була приділена інтеграції інноваційних аспектів у модифіковані методи, щоб забезпечити їхню ефективність та актуальність у сучасному інформаційному середовищі.

В рамках даної роботи було зібрано та підготовлено необхідні для навчання та тестування дані з українського макретплейсу Bigl.ua. При цьому використовувались різні підходи та технології веб скрапінгу та парсингу інформації.

Було вивчено методи навчання та оцінки нейронних мереж у контексті надання рекомендацій та розроблено модель, яка реалізує ідею prod2vec. Також було проведено аналіз наявних алгоритмів та технологій для пошуку сусідніх векторів у просторі та обрано найбільш підходящий з них в даних умовах.

Розроблемний алгоритм показав доволі точні результати, які були оцінені на даних про взаємодію користувачів із різними товарами. Проте отримані результати можна покращити, якщо витратити більше часу та ресурсів на навчання, а також зібрати більшу навчальну вибірку продуктів.

Програмний продукт був розроблений за допомогою популярної мови програмування Python, з використанням фреймворку для глибокого навчання Pytorh, а також бібліотеки для векторного пошуку Faiss. Тренування моделі

відбувалося за допогою хмарного сервісу Google Colab та Jupyter Notebook, розгорнутого у ньому.

Останньою частиною дослідження було наведено аналіз можливості впровадження розробленого рішення у вигляді стартап-проекту. В результаті прийшли до висновку, що рішення, запропоноване у даній роботі, цілком можливо реалізувати у життя.

Результати роботи апробовано на МНТК:

Системи і технології зв’язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку: збірник матеріалів III Міжнародної науково-технічної конференції. - Київ: Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, 2023 - 379 с. DOI: https://doi.org/10.61929/viti.mntk.3.2023

ПЕРЕЛІК ПОСИЛАНЬ

1. Tanmayee S., Unnati N. Recommender Systems in E-commerce. 2022. DOI: 10.13140/RG.2.2.10194.43202.
2. “Personalized e-commerce product recommendations. A guide to implementing a recommendations system for increased brand engagement and conversions”, Grid Dynamics Whitepaper, 2022 URL: https://dou.ua/forums/topic/39070/ (дата звернення: 15.10.2023)
3. Tawfiq, Farah & Rahma, Abdul Monem & Abdulwahab, Hala. An E-Commerce Recommendation System Based on Dynamic Analysis of Customer Behavior. *Sustainability*. 2021. 13. 10786. 10.3390/su131910786.
4. Priyanka Vegesna, Dr. Chandra Sekhar Vasamsetty, “Web Recommendation System for E-Commerce Applications”, *International Journal of Engineering Research & Technology* (IJERT), 2016
5. Обробка даних реальних користувачів для створення рекомендаційної системи. Мій досвід виправлення помилок. URL: https://dou.ua/forums/topic/39070/ (дата звернення: 15.10.2023)
6. Building Recommendation engines in python cource. URL: https://app.datacamp.com/learn/courses/building-recommendation-engines-in-python (дата звернення: 29.09.2023)
7. A content-based recommender for e-commerce web store. URL: https://towardsdatascience.com/a-content-based-recommender-for-e-commerce-web-store-7554b5b73eac (дата звернення: 15.10.2023)
8. Feng Y Enhancing e-commerce recommendation systems through approach of buyer’s self-construal: necessity, theoretical ground, synthesis of a six-step model, and research agenda. Front. Artif. Intell. 2023. 6:1167735. DOI: 10.3389/frai.2023.1167735
9. Liu, C., & Wu, X. Large-scale recommender system with compact latent factor model. 2016. 64, 467–475. DOI:10.1016/j.eswa.2016.08.009

10. Hoang, L. Dealing with the new user cold-start problem in recommender systems : A comparative review. Information Systems, 58, 87–104. 2016. DOI:10.1016/j.is.2014.10.001
11. Karimova, F. A Survey of e-Commerce Recommender Systems. *European Scientific Journal*, *ESJ*, *12*(34), 75. 2016. https://doi.org/10.19044/esj.2016.v12n34p75
12. Jiang, L., Cheng, Y., Yang, L. *et al.* A trust-based collaborative filtering algorithm for E-commerce recommendation system. *J Ambient Intell Human Comput* **10**, 3023–3034. 2019. https://doi.org/10.1007/s12652-018-0928-7
13. Yang, Carl & Bai, Lanxiao & Zhang, Chao & Yuan, Quan & Han, Jiawei. Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation. 2017. 1245-1254. 10.1145/3097983.3098094.
14. Yuan, Weihua & Wang, Hong & Yu, Xiaomei & Liu, Nan & Li, Zhenghao. Attention-based context-aware sequential recommendation model. Information Sciences. 2019 510. 10.1016/j.ins.2019.09.007.

ДОДАТОК А Лістинг програми

Вміст файлу parse_products.py

```
import asyncio
import json
import typing as t
import logging
import re

import aiohttp
import h5py

GRAPHQL_URL = "https://bigl.ua/graphql"
PRODUCT_SCROLL_QUERY = """
query productScroll($limit: Int, $scroll_session_id: String, $exclude_adults: Boolean) {
biglProductsScrollResults(limit: $limit, scroll_session_id: $scroll_session_id, exclude_adults:
$exclude_adults) {
scrollSessionId
scrollFinished
products {
categoryId
name
mainImage {
url_origin
}
attributes {
name
values {
value
unitValue
}
}
}
}
}
```

```
"""
DEFAULT_VARS = {"limit": 100, "exclude_adults": False}

PRODUCTS_PER_CATEGORY = 125
ATTRS_LENGTH = 1000
STRINGS_LENGHT = 256
PRODUCTS_ANOUNT_FOR_LEARING = 200000
SAVED_PRODUCTS_TO_FILE = 0

FILENAME = "data/products-v1.hdf5"

class Attribute(t.TypedDict):
name: str
value: str

class Product(t.TypedDict):
category_id: int
name: str
image_url: str
attributes: str

class ScrollSessionResult(t.TypedDict):
scroll_session_id: str
scroll_finished: bool
products: list[Product]

logging.basicConfig()
log = logging.getLogger(__name__)
log.setLevel(logging.INFO)

async def graphql(query: str, variables: dict | None = None) -> t.Any:
data = json.dumps({"query": query, "variables": variables})

_headers = {
"content-type": "application/json",
"User-Agent": "Prom-Graph-Client",
}
```

```
async with aiohttp.ClientSession(
timeout=aiohttp.ClientTimeout(total=10),
) as session:
async with session.post(GRAPHQL_URL, data=data, headers=_headers) as resp:
response = await resp.json()

def postprocessor(resp: dict) -> t.Any:
return resp["data"] if "data" in resp else resp

if isinstance(response, list):
return map(postprocessor, response)

return postprocessor(response)

def prepare_attr_string(val: str | None) -> str | None:
return re.sub(r"\W", "", val) if val else None

def parse_attr_values(raw_attr_values: list[dict]) -> str:
values = []
for attr_value in raw_attr_values:
value = prepare_attr_string(attr_value["value"])
unit = prepare_attr_string(attr_value["unitValue"])
if value and unit:
values.append(value + unit)
elif value:
values.append(value)
return " ".join(values)

def parse_attributes(raw_attrs: list[dict]) -> str:
attributes: list[str] = []
for attr in raw_attrs:
if combined_attr_value := parse_attr_values(attr["values"]):
if attr_name := prepare_attr_string(attr["name"]):
formated_attribute = f"{attr_name}: {combined_attr_value}"
attributes.append(formated_attribute)
return "|".join(attributes)

def parse_product(product: dict) -> Product:
```

```
return dict(
category_id=product["categoryId"],
name=product["name"],
image_url=product["mainImage"]["url_origin"],
attributes=parse_attributes(product["attributes"]),
)

def parse_result(data: dict) -> ScrollSessionResult | None:
result = data.get("biglProductsScrollResults")
if not result:
return None
products = []
for p in result["products"]:
parsed_product = parse_product(p)
if parsed_product["attributes"]:
products.append(parsed_product)
return dict(
scroll_session_id=result["scrollSessionId"],
scroll_finished=result["scrollFinished"],
products=products,
)

async def graphql_request(vars: dict):
res = await graphql(PRODUCT_SCROLL_QUERY, vars)
return parse_result(res) if res else None

def save_products_to_file(products: list[Product]):
def flatten_products(
prods: list[Product],
) -> list[tuple[int, bytes, bytes, bytes]]:
return [
(
product["category_id"],
product["name"][:STRINGS_LENGHT].encode("utf-8"),
product["image_url"][:STRINGS_LENGHT].encode("utf-8"),
product["attributes"][:ATTRS_LENGTH].encode("utf-8"),
)
for product in prods
```

```
]

flattened_products = flatten_products(products)

# Append to HDF5
with h5py.File(FILENAME, "a") as f:
dtype = [
("category_id", "i"),
("name", f"S{STRINGS_LENGHT}"),
("image_url", f"S{STRINGS_LENGHT}"),
("attributes", f"S{ATTRS_LENGTH}"),
]

if "products" in f:
# Dataset exists, resize and append data
dataset = f["products"]
original_length = len(dataset)
dataset.resize((original_length + len(flattened_products),))
dataset[original_length:] = flattened_products
else:
# Dataset does not exist, create new
maxshape = (None,) # None indicates that the dataset is resizable
dataset = f.create_dataset(
"products", data=flattened_products, maxshape=maxshape, dtype=dtype
)

log.info(f"Products saved to file: {len(flattened_products)}")

def append_products(
products_map: dict[int, list[Product]],
products: list[Product],
):
global SAVED_PRODUCTS_TO_FILE
for product in products:
product_cat = product["category_id"]
products_array = products_map.get(product_cat, [])
if len(products_array) < PRODUCTS_PER_CATEGORY:
products_array.append(product)
```

```
products_map[product["category_id"]] = products_array
SAVED_PRODUCTS_TO_FILE += 1

def clear_file():
with h5py.File(FILENAME, "w"):
pass # Open and close immideatly

async def prepare_data():
res = await graphql_request(DEFAULT_VARS)
if not res or res["scroll_finished"]:
log.warning("Empty data in the first request")
return

graph_vars = DEFAULT_VARS.copy()
graph_vars["scroll_session_id"] = res["scroll_session_id"]

run = True

products_by_category: dict[int, list[Product]] = {}

while run and SAVED_PRODUCTS_TO_FILE < PRODUCTS_ANOUNT_FOR_LEARING:
try:
async with asyncio.TaskGroup() as tg:
tasks = [tg.create_task(graphql_request(graph_vars)) for _ in range(10)]
results = [task.result() for task in tasks]
except Exception as e:
log.exception(e)
continue

for result in results:
if result and (products := result["products"]):
append_products(products_by_category, products)
log.info(
f"Processed {SAVED_PRODUCTS_TO_FILE}/{PRODUCTS_ANOUNT_FOR_LEARING}
products",
)

if result["scroll_finished"]:
```

```
run = False

clear_file()
for prod_list in products_by_category.values():
save_products_to_file(prod_list)

log.info("Parsing is finished")

def main():
asyncio.run(prepare_data())

if __name__ == "__main__":
main()
```

вміст файлу process.py

```
import json

import h5py
import pandas as pd
import numpy as np
from pathlib import Path

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import RegexpTokenizer
from nltk.stem.snowball import SnowballStemmer

BASE_DIR = Path(__file__).parent.parent
NLTK_DIR = BASE_DIR / "nltk"
NLTK_DIR.mkdir(parents=True, exist_ok=True)
nltk.data.path.append(str(NLTK_DIR))

with open(BASE_DIR / "data" / "cats_for_learning.json") as f:
category_tree = json.load(f)

with h5py.File(BASE_DIR / "data" / "products.hdf5", "r") as f:
df = pd.DataFrame(np.array(f["products"]))
```

```
df["name"] = df["name"].str.decode("utf-8", errors="ignore")
df["image_url"] = df["image_url"].str.decode("utf-8")
df["attributes"] = df["attributes"].str.decode("utf-8", errors="ignore")

# Add categories tree to data
def get_cat_level_1(row) -> int | None:
try:
category_path = category_tree[str(row["category_id"])]
path_length = len(category_path)
if path_length == 3 or path_length == 4:
return int(category_path[-2])
elif path_length > 4:
return int(category_path[-3])
else:
return None
except KeyError:
return None

df["cat1"] = df.apply(get_cat_level_1, axis=1)
df = df.dropna()
df["cat1"] = df["cat1"].astype("int32")

# Lowercase characters
df["name"] = df["name"].str.lower()
df["attributes"] = df["attributes"].str.lower()

# Remove whitespaces
def remove_whitespaces(text: str) -> str:
return " ".join(text.split())

df["name"] = df["name"].apply(remove_whitespaces)

# Tokenize
nltk.download("punkt", download_dir=NLTK_DIR)

df["name"] = df["name"].apply(lambda X: nltk.word_tokenize(X, language="russian"))

# Load NLTK stopwords
```

```
nltk.download("stopwords", download_dir=NLTK_DIR)

stopwords = stopwords.words("russian")

# Remove the stopwords
def remove_stopwords(text_tokens: list[str]) -> list[str]:
return [token for token in text_tokens if token not in stopwords]

df["name"] = df["name"].apply(lambda X: remove_stopwords(X))

# Remove punctuation
def remove_punct(tokens: list[str]) -> list[str]:
tokenizer = RegexpTokenizer(r"\w+")
lst = tokenizer.tokenize(" ".join(tokens))
return lst

df["name"] = df["name"].apply(remove_punct)

# Lemmatize
stemmer = SnowballStemmer("russian")

def lemmatization(tokens: list[str]) -> list[str]:
return [stemmer.stem(word) for word in tokens]

# do not need lemmatization for now
# df["name"] = df["name"].apply(lemmatization)

# Convert tokens to string and filter 1 letter words
def tokens_to_text(tokens: list[str]) -> str:
return " ".join(
(token for token in tokens if not (token.isalpha() and len(token) == 1))
)

df["name"] = df["name"].apply(tokens_to_text)
df.rename(columns={"category_id": "cat3"}, inplace=True)

df = df[df["name"] != ""]
df = df[df["attributes"] != ""]
```

```
df["image_path"] = df["image_url"].apply(lambda url: url.split("=/")[1])
images_df = df[["image_url", "image_path"]]
images_df.to_csv(BASE_DIR / "data" / "preprod" / "images.csv", index=False)

df = df.drop("image_url", axis=1)

df = pd.get_dummies(df, columns=["cat1", "cat3"])

chunk_size = 5056

for i, index in enumerate(range(0, len(df), chunk_size)):
smaller_df = df.iloc[index : index + chunk_size]
print("smaller df len: ", len(smaller_df))
smaller_df.to_csv(
BASE_DIR / "data" / "learning" / "texts" / f"texts-{i}.csv", index=False
)
```

вміст файлу download_images.py

```
import asyncio
import pickle
import logging
from pathlib import Path
from typing import Iterable
from itertools import batched

import aiohttp
import pandas as pd

BASE_DIR = Path(__file__).parent.parent

logging.basicConfig()
log = logging.getLogger(__name__)
log.setLevel(logging.INFO)

failed_images = []

async def download_image(
```

```
session: aiohttp.ClientSession, url: str, filename: str
) -> None:
"""
Download an image from the provided URL and save it to the specified filename.
"""
try:
async with session.get(url) as response:
if response.status == 200:
print(f"Downloading image {url}")
# Read the content of the response
content = await response.read()
# Write the content to a file
file_path = str(BASE_DIR / "data" / "learning" / "images" / filename)
with open(file_path, "wb") as f:
f.write(content)
print(f"Image {filename} saved")
except Exception as e:
log.exception(e)
failed_images.append((url, filename))

async def download_images(items: Iterable[tuple[str, str]]):
"""
Download images from a list of URLs.
"""
async with aiohttp.ClientSession(
timeout=aiohttp.ClientTimeout(total=10),
) as session:
for batch in batched(items, 40):
async with asyncio.TaskGroup() as tg:
for url, filename in batch:
tg.create_task(download_image(session, url, filename))

def get_urls_iterable() -> Iterable[tuple[str, str]]:
df = pd.read_csv(str(BASE_DIR / "data" / "preprod" / "images.csv"))
for _, row in df.iterrows():
yield row["image_url"], row["image_path"]

def main():
```

```
urls_iter = get_urls_iterable()
log.exception("Start downloading")
asyncio.run(download_images(urls_iter))
print("List pickled successfully.")
```

вміст файлу model.ipynb

```
import torch
from torch import nn
from transformers import AutoTokenizer, AutoModel
import pandas as pd
from google.colab import drive
import numpy as np

drive.mount("/content/drive", force_remount=True)

device = (
"cuda"
if torch.cuda.is_available()
else "mps"
if torch.backends.mps.is_available()
else "cpu"
)
BERT_MODEL = "cointegrated/rubert-tiny2"

from typing import TypedDict, Callable
import cv2

from PIL import Image
from torch.utils.data import Dataset

class WordTokens(TypedDict):
input_ids: torch.Tensor
token_type_ids: torch.Tensor
attention_mask: torch.Tensor

class ProductData(TypedDict):
x: tuple[torch.Tensor, WordTokens, WordTokens]
```

```
y: tuple[torch.Tensor, torch.Tensor]

DEFAULT_IMAGE = "/content/drive/MyDrive/diploma/learning/default.jpg"

class ProductsDataset(Dataset):
default_image = "default.jpg"

def __init__(
self, base_dir: str, image_transform: Callable, onehoted: bool
) -> None:
self.base_dir = base_dir
self.transform = image_transform
data_df = pd.read_csv(self._get_texts_path(base_dir))
if not onehoted:
data_df = pd.get_dummies(data_df, columns=["cat1", "cat3"])
self.data_df = data_df

def load_image(self, filename: str) -> Image.Image:
image = cv2.imread(filename)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = cv2.resize(image, (221, 221))
return image

def _get_texts_path(self, base_dir) -> str:
return f"{base_dir}/texts.csv"

def _get_image_path(self, filename: str) -> str:
return f"{self.base_dir}/images/{filename}"

def _get_encoded_category(
self, data_row: pd.Series, field_pattern: str
) -> pd.Series:
one_hot_columns = [
col for col in data_row.keys() if col.startswith(field_pattern)
]
one_hot_array = data_row[one_hot_columns].to_numpy(dtype=np.float32)
return one_hot_array
```

```
def __len__(self) -> int:
return len(self.data_df)

def __getitem__(self, index: int) -> ProductData:
data_row = self.data_df.iloc[index]

name = data_row["name"]
attributes = data_row["attributes"]

image_path = data_row["image_path"]
try:
img = self.load_image(self._get_image_path(image_path))
except Exception as e:
print(f"No such image {image_path}. Using default")
print(e)
img = self.load_image(DEFAULT_IMAGE)

transformed_image = self.transform(img)

label = self._get_encoded_category(data_row, "cat3_")
higher_level_cat = self._get_encoded_category(data_row, "cat1_")

x = (transformed_image, name, attributes)
y = (torch.from_numpy(label), torch.from_numpy(higher_level_cat))

return {"x": x, "y": y}

import math
from torch.nn import functional as F

class ArcFace(nn.Module):
def __init__(self, in_features, out_features, s=10, m=0.5):
super().__init__()
self.in_features = in_features
self.out_features = out_features
self.s = s
self.m = m
self.weight = nn.Parameter(torch.FloatTensor(out_features, in_features))
```

```
nn.init.xavier_normal_(self.weight)

self.cos_m = math.cos(m)
self.sin_m = math.sin(m)
self.th = torch.tensor(math.cos(math.pi - m))
self.mm = torch.tensor(math.sin(math.pi - m) * m)

def forward(self, inputs, labels):
cos_th = F.linear(inputs, F.normalize(self.weight))
cos_th = cos_th.clamp(-1, 1)
sin_th = torch.sqrt(1.0 - torch.pow(cos_th, 2))
cos_th_m = cos_th * self.cos_m - sin_th * self.sin_m
cos_th_m = torch.where(cos_th > self.th, cos_th_m, cos_th - self.mm)

cond_v = cos_th - self.th
cond = cond_v <= 0
cos_th_m[cond] = (cos_th - self.mm)[cond]

if labels.dim() == 1:
labels = labels.unsqueeze(-1)
onehot = torch.zeros(cos_th.size()).cuda()
labels = labels.type(torch.LongTensor).cuda()
onehot.scatter_(1, labels, 1.0)
outputs = onehot * cos_th_m + (1.0 - onehot) * cos_th
outputs = outputs * self.s
return outputs

tokenizer = AutoTokenizer.from_pretrained("cointegrated/rubert-tiny2")

import torchvision.models as models
from transformers import AutoConfig

config = AutoConfig.from_pretrained("cointegrated/rubert-tiny2")

bert_out_features = config.hidden_size

CAT1_FEATURES = 198
CAT3_FEATURES = 1643
```

```
def check_invalid_tokens(tokens: dict) -> bool:
return any(
(
torch.isnan(token_vec).any() or torch.isinf(token_vec).any()
for token_vec in tokens.values()
)
)

class ProductVectorizer(nn.Module):
def __init__(self, num_classes: int):
super().__init__()
self.cnn = models.resnet34(pretrained=True)
self.name_bert = AutoModel.from_pretrained(BERT_MODEL)
self.attrs_bert = AutoModel.from_pretrained(BERT_MODEL)

# twick modules params
self.cnn.fc = nn.Linear(self.cnn.fc.in_features, bert_out_features)

self.image_classifier = nn.Linear(
in_features=self.cnn.fc.out_features, out_features=CAT1_FEATURES
)
self.name_classifier = nn.Linear(
in_features=bert_out_features, out_features=CAT1_FEATURES
)
self.attrs_classifier = nn.Linear(
in_features=bert_out_features, out_features=CAT1_FEATURES
)

# Concatenate Embeddings and Batch Normalization
self.concat_bn = nn.BatchNorm1d(
self.cnn.fc.out_features + bert_out_features * 2
)
self.final_embedding = nn.Linear(
self.cnn.fc.out_features + bert_out_features * 2, num_classes
)

self.arcface_layer = ArcFace(num_classes, num_classes)
```

```
self.category_classifier = nn.Linear(
in_features=num_classes, out_features=CAT3_FEATURES
)

# additional params
self.image_ce_input = None
self.name_ce_input = None
self.attrs_ce_input = None

def _forward_bert(self, bert_module: nn.Module, tokens: torch.Tensor):
model_output = bert_module(**tokens)
embeddings = model_output.last_hidden_state[:, 0, :]
embeddings = torch.nn.functional.normalize(embeddings)
return embeddings

def forward(
self,
images: tuple[torch.Tensor, WordTokens, WordTokens],
names: WordTokens,
attrs: WordTokens,
category: torch.Tensor | None = None,
) -> torch.Tensor:
image_embeddings = self.cnn(images)
name_embeddings = self._forward_bert(self.name_bert, names)
attrs_embeddings = self._forward_bert(self.attrs_bert, attrs)

if self.training:
self.image_pred = self.image_classifier(image_embeddings)
self.name_pred = self.name_classifier(name_embeddings)
self.attrs_pred = self.attrs_classifier(attrs_embeddings)

concatenated_embeddings = torch.cat(
(image_embeddings, name_embeddings, attrs_embeddings), dim=1
)
concatenated_embeddings = self.concat_bn(concatenated_embeddings)
final_embeddings = self.final_embedding(concatenated_embeddings)

if not self.training:
```

```
return final_embeddings

if category is not None:
final_embeddings = self.arcface_layer(final_embeddings, category)

return self.category_classifier(final_embeddings)

NUM_FEATURES = 256
model = ProductVectorizer(NUM_FEATURES).to(device)

from torch.utils.data import DataLoader
from torch.utils.data import random_split
from torchvision import transforms

class CustomLoss(nn.Module):
def __init__(self):
super(CustomLoss, self).__init__()

def forward(
self,
image_cce: torch.Tensor,
name_cce: torch.Tensor,
attrs_cce: torch.Tensor,
cat_cce: torch.Tensor,
coef: float,
) -> torch.Tensor:
return coef * (image_cce + name_cce + attrs_cce) + (1 - coef) * cat_cce

class ProductsBatch(TypedDict):
inputs: tuple[torch.Tensor, WordTokens, WordTokens]
labels: tuple[torch.Tensor, torch.Tensor]

def tokenize_text(text: str) -> WordTokens:
text_tokens = tokenizer(text, padding=True, truncation=True, return_tensors="pt")
return text_tokens

def custom_collate(data_arr: list[ProductData]) -> ProductsBatch:
images = []
```

```
names = []
attrs = []
labels = []
cats = []
for data_item in data_arr:
x, y = data_item.values()

images.append(x[0])
names.append(x[1])
attrs.append(x[2])
labels.append(y[0])
cats.append(y[1])

prepared_images = torch.stack(images, dim=0)
prepared_names = tokenize_text(names)
prepared_attrs = tokenize_text(attrs)

prepared_labels = torch.stack(labels, dim=0)
prepared_cats = torch.stack(cats, dim=0)

data = {
"inputs": (prepared_images, prepared_names, prepared_attrs),
"labels": (prepared_labels, prepared_cats),
}
return data

import torch.optim as optim

DEFAULT_LR = 0.0005

cnn_optim = optim.SGD(model.cnn.parameters(), lr=DEFAULT_LR)

name_optim = optim.SGD(model.name_bert.parameters(), lr=DEFAULT_LR)
attrs_optim = optim.SGD(model.attrs_bert.parameters(), lr=DEFAULT_LR)

model_params = [
p
for n, p in model.named_parameters()
```

```
if not any(n.startswith(prefix) for prefix in {"cnn", "name_bert", "attrs_bert"})
]

final_optim = optim.SGD(model_params, lr=DEFAULT_LR)

optimizers = (cnn_optim, name_optim, attrs_optim, final_optim)

cce = nn.CrossEntropyLoss()
loss_function = CustomLoss()

import numpy as np

def alpha_exponential_decay(
epoch: int, initial_alpha: float, final_epoch: int
) -> float:
return initial_alpha * np.exp(-epoch / final_epoch)

def calculate_loss(
model: nn.Module,
cat3_preds: torch.Tensor,
labels: torch.Tensor,
cat1_labels: torch.Tensor,
alpha: int,
) -> torch.Tensor:
cnn_cat1_pred = model.image_pred
bert_name_cat1_pred = model.name_pred
attrs_bert_cat1_pred = model.attrs_pred

loss_1 = cce(cnn_cat1_pred, cat1_labels)
loss_2 = cce(bert_name_cat1_pred, cat1_labels)
loss_3 = cce(attrs_bert_cat1_pred, cat1_labels)
loss_4 = cce(cat3_preds, labels)

total_loss = loss_function(loss_1, loss_2, loss_3, loss_4, alpha)

return total_loss

import os
```

```
from torch.nn.utils import clip_grad_norm_

MODEL_PATH = "/content/drive/MyDrive/diploma/model/state_dict.pth"

def optimizers_step():
for opt in optimizers:
opt.step()

def optimizers_zero_grad():
for opt in optimizers:
opt.zero_grad()

def clip_norm():
clip_grad_norm_(model_params, max_norm=1.0)
clip_grad_norm_(model.cnn.parameters(), max_norm=1.0)
clip_grad_norm_(model.name_bert.parameters(), max_norm=1.0)
clip_grad_norm_(model.attrs_bert.parameters(), max_norm=1.0)

def maybe_save_model(model: nn.Module, index: int):
if index > 0 and index % 25 == 0:
print(f"Saving model to file {MODEL_PATH}")
torch.save(model.state_dict(), MODEL_PATH)

def maybe_load_model(model: nn.Module, path: str = MODEL_PATH):
if os.path.exists(path):
print(f"Loading state dict from: {path}")
model.load_state_dict(torch.load(path))

def prepare_word_tokens(tokens: WordTokens):
return {k: v.to(device) for k, v in tokens.items()}

def train_step(
model: nn.Module,
dataloader: torch.utils.data.DataLoader,
epoch: int,
total_epochs: int,
):
model.train()
```

```
train_loss = 0

for batch_index, batch in enumerate(dataloader):
optimizers_zero_grad()

images, names, attrs = batch["inputs"]
names = prepare_word_tokens(names)
attrs = prepare_word_tokens(attrs)
images = images.to(device)

labels, cat1_labels = batch["labels"]
labels = labels.to(device)
cat1_labels = cat1_labels.to(device)

cat3_preds = model(images, names, attrs, labels)

alpha = alpha_exponential_decay(epoch, 0.5, total_epochs)
total_loss = calculate_loss(model, cat3_preds, labels, cat1_labels, alpha)

print(f"Train batch: {batch_index} epoch: {epoch} loss={total_loss}")

if torch.isnan(total_loss).any():
raise RuntimeError(f"{total_loss} is Nan")

total_loss.backward()
clip_norm()
optimizers_step()

train_loss += total_loss.item()

maybe_save_model(model, batch_index)

return train_loss / len(dataloader)

from tqdm.auto import tqdm

def train_loop(
```

```
model: torch.nn.Module,
train_dataloader: torch.utils.data.DataLoader,
epochs: int = 60,
):
results = {
"train_loss": [],
}

for epoch in tqdm(range(epochs)):
train_epoch_loss = train_step(
model=model, dataloader=train_dataloader, epoch=epoch, total_epochs=epochs
)

print(f"Epoch: {epoch+1} | " f"train_loss: {train_epoch_loss:.4f}")

results["train_loss"].append(train_epoch_loss)

return results

def get_dataset(index: int) -> tuple[ProductsDataset, ProductsDataset]:
transformers = transforms.Compose(
[
transforms.ToTensor(),
]
)
dataset = ProductsDataset(
f"/content/drive/MyDrive/diploma/learning/{index}", transformers, True
)

generator = torch.Generator().manual_seed(42)
train_data, test_data = random_split(dataset, [0.9, 0.1], generator=generator)

return train_data, test_data

from os import cpu_count as os_cpu_count

NUM_WORKERS = os_cpu_count()
```

```
def get_dataloaders(index: int) -> tuple[DataLoader, DataLoader]:
train_data, test_data = get_dataset(index)

train_dataloader = DataLoader(
train_data,
batch_size=32,
shuffle=True,
collate_fn=custom_collate,
num_workers=NUM_WORKERS,
drop_last=True,
)

test_dataloader = DataLoader(
test_data,
batch_size=32,
shuffle=True,
collate_fn=custom_collate,
num_workers=NUM_WORKERS,
drop_last=True,
)

return train_dataloader, test_dataloader

# load model params
path_to_load = "/content/drive/MyDrive/diploma/model/state_dict.pth"
maybe_load_model(model, path_to_load)

dataset_results = {}

directory_path = "/content/drive/MyDrive/diploma/learning/texts"

DATASETS = 28

data_set_gen = (index for index in range(5, DATASETS))

for index in data_set_gen:
print(f"Processing dataset of index: {index}")
train_dataloader, _ = get_dataloaders(index)
```

```
training_results = train_loop(model, train_dataloader, 10)
dataset_results.update({index: dataset_results})

import faiss

_, dl = get_dataloaders(0)

preds = []

for batch_index, batch in enumerate(dl):
model.eval()

images, names, attrs = batch["inputs"]
names = prepare_word_tokens(names)
attrs = prepare_word_tokens(attrs)
images = images.to(device)

embed = model(images, names, attrs)
print("predictions: ", embed)
preds.append(embed)

stacked_tensor = torch.cat(preds, dim=0)

dim = 256

index = faiss.IndexFlatL2(dim)

prepared_preds = stacked_tensor.detach().cpu().numpy()

index.add(stacked_tensor.detach().cpu().numpy())
topn = 10
distances, indices = index.search(prepared_preds[:5], topn)

print("distances:", distances)
print("indices:", indices)
```