НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.9

До захисту допущено
В.о. завідувача кафедри ШІ
Олена Іллівна Чумаченко
«___» _____ 2022 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 122 «Комп'ютерні науки»
на тему: «Нейромережева рекомендаційна система для вибору товару на основі оглядів користувачів і продуктів»

Виконала:
студентка 2 курсу, групи КІ-11мп
Кравченко Олександра Вадимівна _____

Керівник: доцент кафедри ММСА,
д.т.н., доц. Недашківська Н. І. _____

Рецензент: доцент кафедри системного
проектування КПІ ім. Ігоря Сікорського,
к.т.н. Безносик О.Ю. _____

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

Студент (*підпис*): _____

Київ
2022

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 »

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ШІ
_____ О.І. Чумаченко
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Кравченко Олександрі Вадимівні

1. Тема дисертації: «Нейромережева рекомендаційна система для вибору товару на основі оглядів користувачів і продуктів», науковий керівник роботи Недашківська Надія Іванівна, доцент кафедри ММСА, д.т.н., доц. затверджено наказом по університету від «03» листопада 2022 р. № 4046-с
2. Термін подання студентом дисертації 15.12.2022
3. Об'єкт дослідження: Процес формування рекомендацій для вибору товару.
4. Предмет дослідження: Методи і алгоритми, що використовуються для надання релевантних рекомендацій.
5. Перелік завдань, які потрібно зробити:
    1) здійснити огляд технічної літератури за темою роботи;
    2) дослідити актуальність обраної теми;
    3) ознайомитись із існуючими методами та моделями видачі рекомендацій на основі оглядів;

4) здійснити порівняльний аналіз наявних методів надання рекомендацій на основі оглядів

5) розробити модифікацію методу надання рекомендацій на основі оглядів

6) провести експеримент, що засвідчує працеспроможність запропонованої моделі, виконати аналіз результатів;

7) провести аналіз ринкових можливотсей запуску стартап проекту;

8) розробити концептуальні висновки;

9) підготувати ілюстративний матеріал;

10) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу.

7. Дата видачі завдання: 1 вересня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	01.09.2022 – 14.09.2022	Виконано
2.	Підготовка першого розділу.	14.09.2022 – 30.09.2022	Виконано
3.	Підготовка другого розділу.	01.10.2022 – 23.10.2022	Виконано
4.	Розробка програмного продукту.	24.10.2022 – 06.11.2022	Виконано
5.	Підготовка третього розділу	07.11.2022 – 16.11.2022	Виконано
6.	Підготовка частини стартап-проєкту	17.11.2022 – 23.11.2022	Виконано
9.	Концептуальні висновки. Перспективи розвитку отриманих рішень	24.11.2022 – 27.11.2022	Виконано
10.	Оформлення пояснювальної записки	28.11.2022 – 04.12.2022	Виконано

Студент Олександра КРАВЧЕНКО

Керівник Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Магістерська дисертація: 133 с., 36 табл., 68 рис., 27 джерел, 1 додаток.

РЕКОМЕНДАЦІЙНА СИСТЕМА, ОГЛЯДИ ДЛЯ РЕКОМЕНДАЦІЙ, DEEPCONN, ECOMMERCE, НЕЙРОМЕРЕЖА, PYTHON.

Об’єктом дослідження в рамках магістерської роботи є процес формування рекомендацій для вибору товару.

Предметом дослідження є методи і алгоритми, що використовуються для надання релевантних рекомендацій.

Мета дослідження полягає в дослідженні методів надання рекомендацій, і підвищенні їх ефективності.

Було проведене дослідження декількох класичних алгоритмів рекомендацій на основі оглядів і порівняно їх ефективність. На базі цього запропоновано модифікацію методу DeepCoNN і проведено його дослідження і порівняння з початковими.

Запропонований алгоритм виявився ефективнішим близько на 5-7% порівняно з минулими алгоритмами.

ABSTRACT

Master's thesis: 133 p., 36 tab., 68 fig., 27 references, 1 appendix.

RECOMMENDER SYSTEM, REVIEWS FOR RECOMMENDATIONS, DEEPCONN, ECOMMERCE, NEURAL NETWORK, PYTHON.

The object of research within the framework of the master's thesis is the process of forming recommendations for product selection.

The subject of research are methods and algorithms used to provide relevant recommendations.

The purpose of the study is to investigate the methods of providing recommendations and increase their effectiveness.

A study of several classical review-based recommendation algorithms was conducted and their effectiveness was compared. On the basis of this, a modification of the DeepCoNN method was proposed, and its research and comparison with the original method was carried out.

The proposed algorithm turned out to be more efficient by about 5-7% compared to previous algorithms.

ЗМІСТ

ВСТУП

В сучасному світі однією з найбільших проблем є перенавантаження контентом. Сотні фільмів, ігор, книг, товарів тощо виходять в світ щороку. Пересічна людина стикається з проблемою: контенту так багато, що вибір підходящого перетворюється в справжній кошмар. Рекомендаційні системи, здатні самостійно визначити інтереси і потреби користувача, дуже допомагають з цією проблемою.

Рекомендаційні системи – одна з найуспішніших і широко поширених застосувань технологій машинного навчання. Рекомендаційні системи допомагають збільшити дохід бізнесу і допомагають покупцям купувати найбільш підходящий для них продукт. Раніше для формування рекомендацій використовували список найбільш популярних продуктів, але с часом пропозиції (завдяки системам рекомендацій) стали більш персоналізованими.

Зараз існує багато методів для побудови рекомендаційних систем, що допомагають в різних сферах – кіно, музика, супермаркети. Майже ні один сервіс не існує без рекомендаційних систем, навіть найбільш примітивних.

Але завжди є шляхи покращення: наприклад, останніми роками дослідники припустили, що текст відгуків може сказати більше, ніж характеристики товару і вподобання користувача. Рекомендації на основі відгуків – відносно новий напрямок дослідження.

Тому в своїй магістерській дисертації я обрала тему нейромережева рекомендаційна система для вибору товару на основі оглядів користувачів і продуктів, в рамках якої проведу дослідження існуючих методів і запропоную власне покращення.

В роботі пропонується модифікація методу DeepCoNN формування рекомендацій на основі текстів оглядів. Також в роботі виконано порівняльний аналіз точності пропонованого методу з іншими відомими методами.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1. Актуальність задачі надання рекомендацій

Рекомендаційні системи займають важливе місце в процесі життєдіяльності багатьох великих компаній. Рекомендаційні системи підвищують лояльність користувачів до таких сервісів як YouTube, Spotify, Facebook, Netflix, IMDb та інших. Рекомендаційні системи допомагають користувачам знайти фільми, музику, інше за їх вподобаннями, рекомендують нові проекти.

Компанія Netflix в 2006 році оголосила конкурс Netflix Price - значуща подія, яка дала сильний поштовх розвитку рекомендаційних систем [1]. Метою змагання було перевершити існуючу систему рекомендацій Cinematch за метрикою RMSE на 10%. Для цього було надано великий на той час датасет, що містить 100 мільйонів оцінок. Завдання може здатися не таким складним, але для досягнення необхідної якості потрібно було відкривати змагання двічі — рішення було отримано лише на 3 рік змагання. Якби було необхідно отримати покращення в 15%, можливо, цього не вдалося б досягти на наданих даних. Компанія виплатила обіцяну нагороду переможцю, але, на жаль, так і не використала цей алгоритм. Новий алгоритм виявився невиправдано громіздким і дорогим в порівнянні з Cinematch.

Окрім того, рекомендаційні системи займають важливе місце в процесі роботи торгових майданчиків, таких як Amazon. В такому контексті рекомендаційна система виступає в ролі продавця-консультанта, пропонуючи користувачам альтернативні і супутні товари в автоматичному режимі. Таким чином, прибуток компаній різко зростає, а навантаження на консультантів-людей на інтернет-майданчиках зменшується.

З першого погляду можна сказати, що це цілковито прагматичний підхід: пропонуючи додаткові товари, компанія отримує додатковий

прибуток. Проте, варто зазначити, що зараз клієнт-покупець знаходиться у важкому становищі: товарів дуже багато, кожного дня з’являються нові товари, що покращують або полегшують життя, з’являються новинки, про які поки не знає широке коло користувачів. Таким чином, рекомендаційна система допомагає користувачу обрати товар, який найбільше відповідає його вподобанням і потребам.

Щороку проводяться дві події, на яких науковці діляться своїми здобутками в галузі рекомендаційних систем: це конференція the Association of Computing Machinery Conference Series on Recommender Systems і змагання RecSys Challenge.

Актуальність обраної теми щороку зростає, так як рекомендаційні системи можуть бути застосовані майже до будь-яких проектів і бізнесів. Рекомендаційні системи, у свою чергу, мають низку недоліків, над усуненням яких науковці зараз тільки працюють. Основними проблемами є проблема холодного старту – проблема підбору рекомендацій для користувачів і продуктів, про яких ще немає інформації і проблема масштабованості – необхідність можливості необмеженого нарощування числа користувачів і варіантів рекомендацій.

На сьогоднішній день розробка алгоритмів, що повинні справлятися з цими проблемами, залишається відкритим завданням.

1.2. Коротка історія і сучасний вигляд предметної області

Рекомендаційні системи – відносно нова галузь у інформаційних науках. Вони були вперше згадані в технічному звіті як «цифрові книжкові полиці» у 1990 році Юссі Карлгреном з Колумбійського університету і впроваджені в масштабі й опрацьовані в технічних звітах та публікаціях з 1994 року ним же.

Перший цифровий відеомагнітофон, який намагався врахувати переваги користувачів, щоб автоматично записувати відповідні їм передачі, з'явився наприкінці 1990-х. Пристрій називався TiVo і продавався кінцевим користувачам такими брендами, як Sony і Philips. У той час добре структурованих описів для всіх мовних телепередач не було і потужності пристроїв для складних обчислень теж не вистачало, тому в основу системи рекомендацій був покладений підхід колаборативної фільтрації.

Колаборативна фільтрація ґрунтується виключно на схожості поведінки користувачів та не використовує жодних описів телепередач та фільмів чи іншої семантики. Якщо на двох пристроях TiVo у двох різних домогосподарствах глядачі зазвичай вибирали одні й ті самі програми, а потім перший користувач починав регулярно дивитися якусь нову телепередачу, то цю телепередачу рекомендували і другому користувачеві. Всі пристрої TiVo вночі відправляли на центральний сервер дані про послідовності кліків, на підставі цих даних рекомендаційний алгоритм сервера обчислював персональні рекомендації, і вони потім завантажувалися на пристрої. Така організація процесу на центральному сервері дозволяла, по-перше, порівнювати і знаходити глядачів зі схожими смаками, а по-друге, не «вантажити» відеомагнітофони TiVo з їх обмеженими обчислювальними можливостями.

Незважаючи на те, що рішення не працювало в реальному часі і іноді неправильно визначало інтереси користувача, простота підходу швидко зробила його популярним у галузі. Чемпіоном застосування стала компанія Netflix, яка для свого бізнесу надання DVD-дисків в оренду залучила команду розробників, що налаштовують параметри рекомендаційного двигуна колаборативної фільтрації, і добивалася рік у рік найкращих результатів.

Однак у середині 2000-х якість рекомендацій, яка може забезпечити алгоритм колаборативної фільтрації, досягла певної межі.

Саме на цьому етапі Netflix оголосив конкурс, про який було сказано в 1.1. Як вже можна було помітити, колаборативна фільтрація зайшла в глухий кут.

У процесі розвитку нових підходів до вирішення завдання на гребні хвилі виявилися нові компанії, зокрема Jinni та Aprico, які зрозуміли, що якісна інформація про рекомендований контент могла б бути серйозною основою для тонкого підстроювання під інтереси глядачів. Ця нова хвиля досліджень в основному використовувала байєсовську класифікацію, яка дозволяє зрозуміти і оцінити, чому користувач віддає перевагу тому чи іншому контенту. Наприклад, якщо користувач дивиться фільми типу «Джеймса Бонда», але пропускає «Одинадцять друзів Оушена», система вважає, що коли йдеться про фільми, цей користувач любить бойовики і пригоди, але не особливо любить трилери або детективи.

У Netflix одразу оцінили перспективи нового напряму семантичного аналізу описів та знову вирішили зайнятися власною розробкою. Як і Jinni, вони стали створювати описи та класифікатори для фільмів свого каталогу. Цей підхід мав ряд проблем, пов'язаних з масштабуванням, тому що розумний і виправданий, тільки якщо каталог відео відносно малий і повільно збільшується. Інші компанії, що займаються семантичними рекомендаційними рішеннями, сфокусувалися на класифікації контенту на основі метаданих, які існували на ринку. Netflix знову відзначився і постарався дійти до меж можливого. Його каталог жанрів швидко виріс з вихідних 560 варіантів до 93 116, за даними на липень 2014 року, серед яких були, наприклад, такі як «емоційні драми 80-х, що отримали високу оцінку кінокритиків», куди в каталозі Netflix потрапляло лише 2 фільми. І це знову було занадто — категорії, яким відповідають один-два об'єкти, не придатні для аналізу та стають незначними для байєсівського класифікатора.

Тим часом, розвиток хмарних обчислень надав компаніям, які розробляють рекомендації, колосальні можливості для проведення натурних

тестів. Виявилося, на подив багатьох розробників, що вимоги користувачів жоден конкретний рекомендаційний алгоритм задовольнити не може. Користувачам потрібна комбінація редакторських рекомендацій, фільтрації за контентом на основі персональних уподобань, рекомендацій спільноти людей з близькими смаками, і щоб у цих рекомендаціях були вдалі знахідки, а підбір контенту відповідав миттєвому настрою та типу використовуваного пристрою. При цьому користувач повинен розуміти, що це не якийсь там "Великий брат" слідкує за ним, а він може вплинути на результат. Складні налаштування пропонувати не потрібно, скажімо, співвідношення між редакторськими рекомендаціями та фільтрацією за контентом визначає оператор, але, наприклад, якщо поруч із рекомендованим фільмом зазначено, що цей фільм вам радять подивитися тому, що вам подобається Дженніфер Лопес, а вона вам насправді не подобається, потрібно мати можливість відразу це відзначити та негайно отримати інші рекомендації. Це ідея, описана для рекомендації фільмів, проте подібні роздуми можуть бути адаптовані до будь-якої сфери застосування.

Таким чином, після періоду вивчення кліків користувача та після періоду семантичних рекомендацій ми прийшли до нового етапу, в якому основну роль відіграє контекст: сьогохвилинні наміри, інтереси та оточення користувача. Ясно, що це знову інший підхід, при якому рекомендаційний двигун повинен давати результат відразу, як будуть отримані дані про контекст, враховувати всі варіанти алгоритмів, пояснювати рекомендації користувачеві та давати постійний контроль над параметрами. Для цього необхідні зовсім нові підходи, або ж навіть комбінація декількох – тому зараз настає час для так званих гібридних рекомендаційних систем.

Рекомендаційні системи все ще знаходяться на етапі стрімкого розвитку, в наукових журналах постійно з’являються статті, які пропонують нові алгоритми надання рекомендацій і способи покращення вже існуючих. Докладніше найцікавіші ідеї надання рекомендацій будуть розглянуті в 2.3.

1.3. Аналіз методів надання персоналізованих рекомендацій

Рекомендації продукту загалом можна розділити на два великі типи: персоналізовані та неперсоналізовані.

Неперсоналізовані рекомендації дуже прості і примітивні, і зазвичай не відповідають меті впровадження. У випадку такого типу рекомендацій користувачу рекомендується найбільш популярні продукти, або найбільш відповідні бізнес-завданням. Як можна здогадатися, такі рекомендації мають місце лише як частина комплексної системи рекомендацій, що використовує більш ніж один алгоритм.

Персоналізовані рекомендації, у свою чергу, поділяються на чотири типи: Content-based, Case-based, Collaborative filtering та гібридні підходи, які поєднують у собі два або три попередні підходи. Перші три підходи будуть розглянуті у даному пункті, а гібридні підходи докладно розглянемо у 2.3, де будуть розглянуті сучасні алгоритми рекомендаційних систем.

Колаборативна фільтрація – це такий тип фільтрації, при якому користувачі групуються по схожості, і користувачу пропонуються продукти, які сподобалися іншим користувачам з тієї самої групи. Він заснований на трьох припущеннях: люди мають схожі вподобання і інтереси, їх інтереси стабільні, і ми можемо робити висновки про їх вибір виходячи з минулих вподобань. [2]

Схожість в методі колаборативної фільтрації задається наступною формулою:

$$sim(u,i) = \frac{\sum_{u \in U_{ij}} (r_{u,i} - \overline{r_u})(r_{u,j} - \overline{r_u})}{\sqrt{\sum_{u \in U}((r_{u,i} - \overline{r_u}))^2} \sqrt{\sum_{u \in U}((r_{u,j} - \overline{r_u}))^2}} \quad (1)$$

де $u \in U$ – множина користувачів, $i \in I$ – множина об’єктів,

$(r\_ui,\ u,\ i) \in D$ – множина подій або діяльність, яку виконують над об’єктами.

Для коректної роботи цього типу рекомендаційних систем потрібна велика кількість вхідної інформації, інакше рекомендації можуть бути абсурдними або неточними. Тому можна сказати, що цей тип систем найбільше потерпає від проблеми холодного старту.

Підходи колаборативної фільтрації можуть бути розділені на два основні: user-based і item-based. Підходи відрізняються залежно від того, хто знаходиться в фокусі: в першому випадку оцінюється схожість користувачів для рекомендації однакової продукції, в другому – схожість продукції, щоб рекомендувати користувачам, що купили щось одне, інший, схожий продукт. Методи і алгоритми, за якими оцінюється схожість, будуть докладно представлені в 2.1. Обидва підходи мають місце бути і використовуються для різних ситуацій, проте, зараз найчастіше використовуються гібридні моделі, що враховують і схожість продукту, і схожість користувачів.

Фільтрація, заснована на контенті, є основою багатьох рекомендаційних систем. Товари та послуги рекомендуються з урахуванням знань про них: функції, склад, тощо. За таким принципом найчастіше працюють системи інтернет-магазинів. Такий підхід є особливо ефективним, коли відома інформація про продукт, але невідома про користувача. Таким чином, метою контентно-орієнтованої рекомендаційної системи буде дослідити вподобання користувача відповідно до його дій в системі, і порекомендувати йому такі продукти, які будуть мати подібні характеристики. Іншими словами, така система базується на припущенні, що майбутні вподобання користувача будуть схожими до тих, що були в минулому.

Недоліком такого підходу є вірогідність того, що користувач застрягне в одноманітному контенті. Виходить замкнуте коло: користувач вподобав якийсь тип контенту, і система тепер пропонує тільки його. Користувач може вподобати або не вподобати щось, але системі забракне інформації, щоб

видати щось нове, а отже, користувач буде отримувати все більш і більш спеціалізований одноманітний контент. Проблемою також може бути проблема холодного старту: рекомендації на основі контенту стають ефективними тільки після того, як користувач оцінить певну кількість елементів.

Рекомендаційні системи, засновані на знаннях, ще називають експертними рекомендаційними системами. Вони працюють на основі знань про певну предметну область, про користувачів, про товари та інші аспекти, які допомагають прийняти рішення про вибір того чи іншого продукту. Ці системи діляться на декілька типів: case-based, demographicbased, utility-based, critique-based, whatever-you-want-based і т.д. Рекомендаційна експертна система може враховувати прямі обмеження від користувача – наприклад, максимальну ціну продукту.

Недоліком такого типу системи можна назвати велику складність її впровадження. Проте, система зазвичай це виправдовує. Такі системи доцільно використовувати в складних предметних областях, де історія покупок для користувача, наприклад, або повністю відсутня, або дуже мала – покупка квартири або автомобіля.

Ключовими ідеями, що застосовуються для цього типу рекомендацій, є:

- діалог, або цикл зворотного зв’язку для охоплення всіх можливих аспектів вибору. Таким чином, користувач може поступово сформулювати всі свої вимоги, або навіть визначитися з вподобаннями під час сеансу.
- обмеження, коли зворотній зв’язок реалізовано у вигляді запитання-відповіді, які обмежують набір відповідних елементів. Такі системи також можуть бути технічно реалізовані як пошуковий фільтр.
- критика – коли зворотній зв’язок надається в термінах критичних зауважень, які вказуються як запити на заміну деяких характеристик предмету, який в даний момент рекомендується користувачу. Критика використовується для рекомендації наступного кандидату.

Найбільшим недоліком такого типу систем є їх велика специфічність: побудовану експертну систему не можна буде застосувати до іншої предметної області. Окрім того, для побудови потрібен експерт в предметній області, який побудує систему асоціативних правил. Такий підхід, як уже було сказано, є найбільш затратним з боку фінансів, людських ресурсів і часу.

Комбінування кількох підходів в рамках гібридної системи дозволяє мінімізувати недоліки кожного. Існує кілька найбільш розповсюджених типів комбінування:

- реалізація окремо колаборативних та контентних алгоритмів та поєднання їх припущень;
- включення деяких контентних правил до колаборативної методики
- включення деяких колаборативних правил до контентної методики
- побудова загальної моделі, що включає правила обох методик.

Докладніше гібридні методи будуть описані в наступному розділі.

1.4. Етапи процесу рекомендацій

Перший етап - етап збору інформації. На цьому етапі збирається загальна інформація для створення профіля, зазвичай атрибути користувача, поведінка або зміст ресурсів, до яких звертається користувач. Рекомендаційна система не може працювати добре, поки профіль користувача не буде створено. Система має дізнатися якомога більше про користувача перш ніж надати релевантні рекомендації.

Рекомендаційна система також може використовувати різні типи вхідних даних, у тому числі явний зворотній зв’язок, що якісно надається користувачем, і неявний, коли користувацькі переваги визначаються шляхом

поведінки за користувачем. Також зворотній зв’язок може бути гібридним, якщо поєднує обидва підходи.

Якщо використовується явний зворотній зв’язок, система зазвичай має інтерфейс, в якому є певний рейтинг елементів. Точність рекомендації залежить від кількості наданих користувачем оцінок. Таке рішення потребує певних зусиль від користувача, але все ж вважається більш надійним варіантом

Неявний зворотній зв’язок відслідковує окремі дії користувача, наприклад історію покупок, історія навігації, час, проведений на веб-сторінках, відвідані посилання і таке інше. Цей метод не потребує ніяких дій зі сторони користувача, однак він менш точний.

Гібридний зворотній зв’язок в цілому допомагає звести до мінімуму слабкі сторони і отримати максимально ефективну систему. Наприклад, можна використовувати неявні дані щоб перевірити об’єктивність явного рейтингу, або пропонувати користувачу давати явний відгук лише коли він цього захоче.

Далі починається етап навчання. На цьому етапі система, використовуючи алгоритм навчання і інформацію про користувача, відгуки, поведінку, тренується надавати коректні рекомендації. Цей етап (особливо «алгоритм навчання») дуже сильно відрізняється в залежності від того, який підхід до рекомендації використовується.

Далі, нарешті – етап рекомендації. Система рекомендує (або «передбачає»), які елементи сподобаються користувачу. Після цього за допомогою зворотного зв’язку система отримує фідбек, на основі якого може коригувати наступні рекомендації.

1.5. Використання рекомендаційних систем для е-комерції

Як вже згадувалося раніше, на сучасному ринку товарів і послуг настільки багато пропозицій, що користувачу буває дуже складно обрати найвигіднішу і найбільш підходящу для себе пропозицію. Щоб допомогти користувачам сучасні сервіси використовують рекомендаційні системи, що значною мірою впливає на прибуток компанії.

Перевагами використання рекомендаційних систем є:

1. Збільшення кількості проданих товарів – найбільша перевага для комерційних рекомендаційних систем, яка, для прикладу, дозволяє зробити «допродажі», у форматі «разом з цим товаром купують». Як правило, такі товари також відповідають потребам користувачів.

2. Покращення залученості користувачів – підходить також для некомерційних проектів, якщо, наприклад, прибуток не залежить від грошей, що витрачає користувач. Наприклад, система YouTube заробляє на кількості показаної реклами, а отже, у його інтересах давати користувачам найбільш релевантний контент, що збільшити його залученість до платформи.

3. Продаж широкого спектру товарів – допомагає користувачам дізнатися про товари або послуги, які важко знайти без точної інформації: наприклад, якісь дуже специфічні послуги. Важливим для рекомендаційної системи є не тільки рекомендувати найбільш популярні товари і послуги, а й менш популярні, але більш релевантні для користувача.

4. Підвищення лояльності для користувачів – користувачі більш лояльно відносяться до сайтів, які впізнають його і підкреслюють його вподобання. Рекомендаційні системи, у свою чергу, можуть допомогти з персоналізацією веб-сайту.

5. Краще розуміння цільової аудиторії – наприклад, в розрізі цієї переваги рекомендаційні системи допомагають прогнозувати майбутні

продажі на основі історичних даних користувачів, планувати майбутні закупки та інше.

Окрім того, варто зазначити основні особливості роботи з рекомендаційними системами в е-комерції.

Зворотній зв’язок, як вже було вказано в 1.4, може бути явним або неявним. В е-комерції явним зворотнім зв’язком є оцінки, поставлені товару (рейтинги) від 1 до 5, або бінарні значення сподобалось-не сподобалось. До неявного зворотнього зв’язку відноситься історія покупок, переглядів, текстові відгуки.

Явний зворотній зв’язок є більш цінним, ніж припущення, зроблені на основі неявного, але частіше за все користувачі не знаходять часу, щоб залишити відгук, тому рекомендаційна система має вміти добувати якомога більше інформації з неявного зворотного зв’язку.

1.6. Сучасні напрямки досліджень рекомендаційних систем

На сьогоднішній день існує декілька популярних напрямків досліджень рекомендаційних систем для покращення точності наданих рекомендацій. Серед них можна виділити наступні:

1. Використання оглядів для рекомендацій

Загалом у цьому напрямку стверджується, що оскільки огляди «пояснюють» думки користувачів, вони повинні бути корисними для визначення базових параметрів, які передбачають оцінки чи покупки. Схеми для включення оглядів варіюються від простих регуляризаторів до підходів нейронної мережі. Деякі дослідники [3] навпаки стверджують, що такі моделі показують себе мало не гіршими за еталонні моделі, якщо помістити їх в повністю однакові умови.

2. Передбачення рейтингу переходів

За допомогою рейтингу переходів (click-through rate, CTR) ми можемо в певній формі виміряти, скільки кліків отримали рекомендації. Основне припущення полягає в тому, що більша кількість кліків на рекомендованих елементах означає, що рекомендації були більш релевантними для користувачів.

Прогноз рейтингу переходів відіграє вирішальну роль у рекламних онлайн-кампаніях і системах рекомендацій. Більшість найсучасніших моделей базуються на машинах факторизації, і деякі з цих моделей намагаються передати відображені функції поля компоненту глибокого навчання для вивчення інтересів користувачів шляхом моделювання взаємодії функцій. Розгортання моделі для CTR є онлайн-завданням, і воно повинно добре працювати з обмеженою кількістю даних і часу. Незважаючи на те, що ці моделі дуже добре підходять для висновків про прогнозування та вивчення взаємодій функцій, їхній глибокий компонент потребує величезної кількості даних і часу та не працює добре в обмежених ситуаціях.

3. Рекомендації, засновані на контексті

Контекстно-залежна система рекомендацій (CARS) застосовує визначення й аналіз контексту користувача для надання персоналізованих послуг. Контекстну інформацію можна отримати від датчиків, щоб підвищити точність рекомендацій. Проте генерування точних рекомендацій недостатньо для створення корисної системи з точки зору користувачів, оскільки певна контекстна інформація може спричиняти різні проблеми, наприклад розряджати акумулятор користувача, проблеми з конфіденційністю тощо. Додавання багатовимірної контекстної інформації може збільшити як розмірність, так і розрідженість моделі. Попередні дослідження пропонують зменшити кількість контекстної інформації шляхом вибору найбільш підходящої контекстної інформації за допомогою знань домену. Іншим рішенням є стиснення його в більш щільний прихований простір, таким

чином порушуючи можливість пояснити елемент рекомендації користувачеві та підриваючи довіру користувачів.

4. Використання автокодерів для рекомендацій

Автокодер — це тип нейронної мережі, який підходить для завдань навчання без нагляду, зокрема генеративного моделювання, зменшення розмірності та ефективного кодування. Він показав свою перевагу у вивченні представлення базових функцій у багатьох областях, включаючи комп'ютерне бачення, розпізнавання мови та моделювання мови. Враховуючи ці знання, нові архітектури рекомендацій включили автокодер і, таким чином, надали більше можливостей у переосмисленні користувацького досвіду для задоволення клієнтів. У той час як традиційні моделі працюють лише з одним джерелом даних (оцінка чи текст), моделі на основі автоматичного кодувальника можуть опрацьовувати неоднорідны джерела даних (рейтинг, аудіо, відео, відео).

Автоматичний кодер краще розуміє вимоги користувача та особливості товару, що забезпечує вищу точність рекомендацій, ніж традиційні моделі.

Крім того, автоматичний кодувальник допомагає моделі рекомендацій бути більш адаптованою в мультимедійних сценаріях і більш ефективною в обробці вхідних шумів, ніж традиційні моделі.

5. Послідовні рекомендації

Системи послідовних рекомендацій намагаються зрозуміти введення користувача з часом і моделюють у послідовному порядку. Взаємодія користувача введення по суті залежить від послідовності . Це означає, що якщо людина бронює рейс, вона також бронює таксі до пункту призначення та бронює номер. Ця інформація зберігається в послідовності. Якщо інша особа замовляє рейс і таксі, система дасть рекомендації щодо бронювання готелю чи номера. Уподобання користувачів і популярність товару є динамічними або змінюються з часом.

Щоб охопити такі закономірності, поширено два підходи: ланцюги Маркова (MC) і рекурентні нейронні мережі (RNN). Ланцюги Маркова припускають, що наступну дію користувача можна передбачити на основі його останніх (або останніх кількох) дій, тоді як RNN в принципі дозволяють розкрити довгострокову семантику. Взагалі кажучи, методи на основі MC найкраще працюють у надзвичайно розріджених наборах даних, де економія моделі є критичною, тоді як RNN працюють краще в щільніших наборах даних, де доступна більш висока складність моделі.

1.7. Аналіз існуючих сервісів з надання рекомендацій

Як вже згадувалося у 1.1, рекомендаційні системи на сьогоднішній день використовуються майже в усіх видах бізнесу. Можна розглянути найбільш відомі приклади використання різних типів рекомендаційних систем:

IMDB – онлайн база даних про фільми та серіали, яка зберігає інформацію про акторів, людей, хто створив фільм, короткий опис сюжету, рецензії критиків та глядачів, оцінки, які поставили фільму глядачі. Рекомендаційний аспект полягає в тому, що зареєстровані користувачі отримують персональні рекомендації фільмів на основі зроблених раніше оцінок. Для кожної рекомендації вказуються фільми та серіали, на основі яких рекомендація була зроблена. Рекомендуються фільми та серіали, які сподобалися людям зі схожими смаками – тобто, використовується user-based колаборативна фільтрація.

Netflix – потоковий сервіс, що дозволяє переглядати фільми та серіали як власне зняті сервісом, так і викуплені. Для кожного фільму та серіалу можна переглянути подібні, а також відсоток подібності (item-based колаборативна фільтрація). Можна попросити сервіс порекомендувати щось принципово нове, не пов’язане з профілем користувача (неперсоналізовані рекомендації). Можна переглянути список найкращих фільмів, фільми за

певним жанром, окремо Netflix надає персоналізовані content-based рекомендації на основі вподобань користувача. Система вподобань тут незвична: це градація оцінок від -2 (два дизлайка) до +2 (два лайка). Однак, на фільм чи серіал не можна залишити публічну оцінку або відгук. Градація оцінок необхідна лише сервісу, для уточнення вподобань.

LinkedIn – бізнес-орієнтована соціальна мережа. Як і в інших соціальних мережах, вбудована рекомендаційна система пропонує користувачу людей, яких він може знати, вакансії, які йго можуть зацікавити, групи, в які він міг би вступити, компанії, інше. Система використовує модифікований підхід колаборативної фільтрації, зробленої на технології Apache Hadoop.

Spotify – сервіс з прослуховування музики, який використовує для рекомендацій близько 400 атрибутів, основаних на метаданих пісень і виконавців, щоб згенерувати плейліст з подібними властивостями. Окремо аналізується зворотній зв'язок від користувача: коли йому не сподобалася певна після або виконавець важливість відповідних атрибутів зменшується, а коли сподобалися – навпаки. Сервіс використовує підхід, заснований на контенті.

Youtube – відеохостинг, що надає послуги зберігання та показу відео. Користувачі можуть завантажувати, переглядати, оцінювати, коментувати, додавати до плейлистів відео та інше. Хостинг дуже простий, інтуїтивний у використанні тому користується популярністю. Сервіс використовує два типи рекомендацій. Перший – це світові тренди, неперсоналізовані рекомендації відео, які є найпопулярнішими в даний момент, і другий – особисті рекомендації, засновані на діях користувача у минулому.

Steam – це онлайн платформа для розміщення, купівлі, обговорення ігр для ПК. Рекомендаційна система створює персоналізовані списки ігор на основі того, в що користувач грав раніше. Вона використовує машинне навчання на основі колаборативної фільтрації: На рекомендації також

впливають: мітки ігор, відгуки користувачів, популярність ігор, наявність ігор у друзів, наявність в списку лідерів продажів.

Amazon – один з найбільших майданчиків інтернет торгівлі – використовує контентно-орієнтовані рекомендації. Коли відвідувач вибирає для покупки будь-якої товар, Amazon на основі цього вихідного товару рекомендує відвідувачеві інші товари, придбані іншими користувачами за допомогою матриці покупки наступного товару на основі його схожості з попередньою покупкою. Цей підхід був запатентований як item-to-item колаборативна фільтрація, яка вже розглядалася в 1.3.

Gifts When You Need – сервіс, який компанія 1-800-Flowers.com називає «консьєржем з штучним інтелектом». Клієнт надає сервісу інформацію про одержувача подарунку, і програмне забезпечення адаптує рекомендації для вибору, основані на подарунках, придбаних для подібних користувачів.

North Face використовує рекомендаційну систему, щоб допомогти клієнтам визначитися, яка куртка їм найбільше підходить. Вибір ґрунтується на таких атрибутах, як місце розташування, пора року і гендерні особливості.

1.8. Постановка задачі дослідження

Об’єктом дослідження в рамках магістерської роботи є процес формування рекомендацій для вибору товару.

Предметом дослідження є методи і алгоритми, що використовуються для надання релевантних рекомендацій.

Метою дослідження є дослідження методів надання рекомендацій, і підвищення їх ефективності.

Для досягнення поставленої мети потрібно виконати наступні завдання:

- Дослідити предметну область побудови рекомендаційних систем
- Провести порівняльний аналіз методів надання рекомендацій на основі глибинного навчання

- Розробити модифікацію нейромережевої моделі формування рекомендацій
- Проаналізувати існуючі метрики якості моделей
- Оцінити якість і вибрати найкращу модель для досліджених даних
- Програмно реалізувати рекомендаційну систему на основі розробленої моделі з використанням сучасної нейромережевої бібліотеки мови Python – Pytorch.

Результатом таких досліджень має бути метод для створення рекомендаційних систем, який би показував кращі результати за попередні методи, а також прототип системи рекомендацій для вибору товару.

1.9. Висновки до розділу 1

В розділі 1 було проаналізовано актуальність задачі рекомендації продуктів клієнтам, розглянуто історію створення рекомендаційних систем від 1994 року до наших днів. Розглянуті і проаналізовані сучасні приклади використання рекомендаційних моделей для бізнесу.

Було розглянуто підходи до проектування рекомендаційних систем, а саме:

a. Неперсоналізовані
b. Персоналізовані
    1. На основі колаборативної фільтрації
    2. На основі контенту
    3. На основі знань
c. Гібридні підходи і основні їх напрямки розробки.

Відзначимо, що кожен підхід має слабкі і сильні сторони, тому практичні реалізації часто містять у собі декілька підходів – це і є гібридний підхід.

Розвиток рекомендаційних систем ще триває, і можна зазначити тенденцію до використання машинного навчання до побудови рекомендаційних систем, завдяки якому підвищується швидкість обробки запитів і якості персоналізації. Крім того, були розглянуті напрямки досліджень сучасних рекомендаційних систем, а саме рекомендації на основі оглядів, прогноз рейтингу переходів, рекомендації, засновані на контексті, використання автокодерів для рекомендацій, послідовні рекомендації.

Було сформовано завдання, цілі і задачі розробки.

РОЗДІЛ 2 МЕТОДИ І МОДЕЛІ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ФОРМУВАННЯ РЕКОМЕНДАЦІЙ

2.1 Опис класичних методів надання неперсоналізованих і персоналізованих рекомендацій

Розглянемо постановку задачі надання рекомендацій

Нехай $u \in U$ – множина користувачів, $i \in I$ – множина об'єктів,

$(r\_ui,\ u,\ i) \in D$ – множина подій або діяльність, яку виконують над об'єктами. Кожна з подій задана відповідним результатом r_ui, а також іншими характеристиками.

Знайти: прогноз і рекомендації для заданого користувача – це підхід user based recommendations або для заданого товару – item based recommendations.

Метод надання неперсоналізованих рекомендацій

Найпростішим випадком є рекомендація за популярністю. Метод полягає в тому, що розраховується один з наступних показників:

1. $Score = (Positive\ ratings) - (Negative\ ratings)$. Проблема цього підходу полягає в тому, що товари, у яких 200 лайків і 50 дизлайків і 1200 лайків і 1050 дизлайків будуть оцінені однаково.

2. $Score = (Positive\ ratings)\ /\ (Total\ ratings)$. Цей підхід не враховує кількість оцінок: товар з однією оцінкою 5 буде вище, ніж популярний товар з оцінкою 4.8.

3. Розраховується довірчий інтервал і дається відповідь на питання: «З огляду на наявні оцінки, з ймовірністю 95% якою є справжня частка позитивних оцінок» [1]. Використовується критерій Вілсона:

$$Wilson\ Score = \frac{\hat{p} + \frac{z_{\frac{\alpha}{2}}^2}{2n} \pm z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p}) + \frac{z_{\frac{\alpha}{2}}^2}{4n}}{n}}}{1 + \frac{z_{\frac{\alpha}{2}}^2}{n}} \quad (2)$$

Метод колаборативної фільтрації надання персоналізованих рекомендацій

Кожному користувачу ставиться у відповідність вектор його вподобань, і для вирішення задачі рекомендації потрібно оцінити схожість двох векторів. Для цього використовують різні коефіцієнти, серед них [4]:

1. Кореляція Пірсона:

$$p = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \quad (3)$$

Основним мінусом є те, що у випадку, якщо перетин за оцінками низький, кореляція може бути високою випадково.

2. Кореляція Спірмана:

$$p = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (4)$$

Основна відмінність – коефіцієнт ранговий, тобто працює не з абсолютними значеннями рейтингів, а з їхніми порядковими номерами. У цілому дає результат дуже близький до кореляції Пірсона.

3. Косинусна відстань

Розраховується косинус кута між векторами вподобань користувачів:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (5)$$

Метод ймовірнісної матричної факторизації (PMF) представлено імовірнісною лінійною моделлю з гаусовим спостережним шумом. Припустимо, у нас є N користувачів та M фільмів. Нехай Rij – значення рейтингу для i-го користувача і j-го фільму, Ui та Vj представляють D-

розмірні специфічні для користувача і для фільму вектори латентної функції, відповідно. Умовний розподіл за спостережними оцінками визначається за допомогою:

$$p(R|u,V,\alpha) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[N(R_{ij}|U_i^T V_j, \alpha^{-1})\right]^{I_{ij}} \quad (6)$$

$$p(U|\alpha_U) = \prod_{i=1}^{N} N(U_i|0, \alpha_U^{-1} I) \quad (7)$$

$$p(V|\alpha_v) = \prod_{j=1}^{M} N(V_j|0, \alpha_v^{-1} I) \quad (8)$$

Контент-орієнтований підхід

Для впровадження цього підходу треба зіставити користувачів з товарами або контентом, які йому сподобалися. Для кожного слова ми зберігаємо можливість появи цього слова в контенті, який вжив користувач.

Побудувавши ці «профілі», обчислюємо схожість між користувачами та предметами. Предмети повинні бути рекомендовані користувачеві, якщо: 1)вони мають найбільшу схожість із користувачем або 2) мають велику схожість з іншими елементами, прочитаними користувачем. Для цього можна використовувати ті ж способи, що розглядалися вище: косинусна схожість, кореляція Пірсона, Спірмана та інші.

2.2 Моделі глибоких нейронних мереж для формування рекомендацій

На рис.1 показано архітектуру гібридної моделі колаборативної фільтрації та глибинного навчання Neu-CF

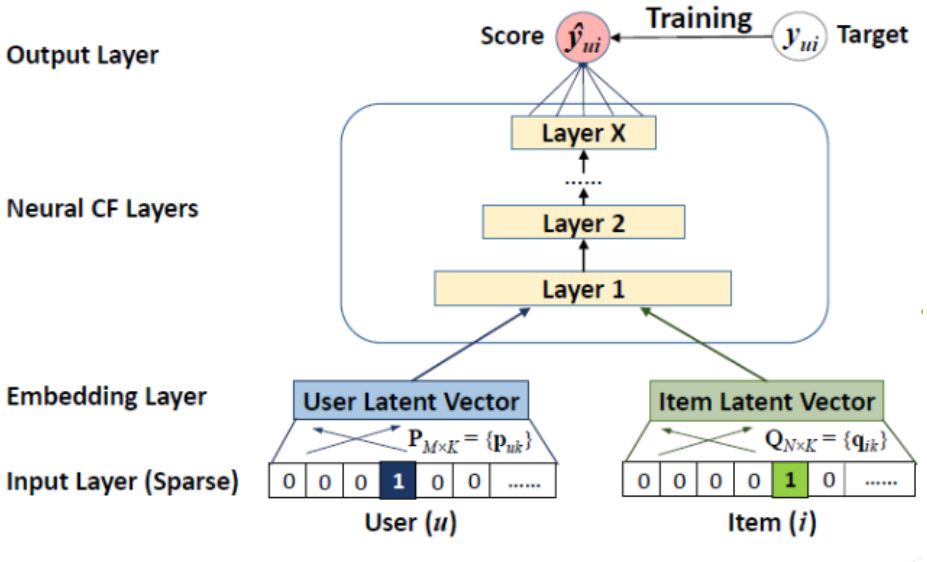

Рисунок 1 – Архітектура моделі Neu-CF

Результат моделі Neu-CF:

$$\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f) \tag{9}$$

де P, Q – матриці для прихованих факторів відповідно користувача і об'єкта. Функція f представляє багатошарову нейронну мережу, тому її можна описати наступним чином:

$$f\left(P^T v_u^U, Q^T v_i^I\right) = \phi_{out}(\phi_X(\dots \phi_2\left(\phi_1\left(P^T v_u^U, Q^T v_i^I\right)\right)\dots)) \tag{10}$$

де ϕ_{out}, ϕ_x визначають взаємозв'язок вихідного шару з x-м шаром NCF.

На рис.2 показано гібридну модель Neu-MF, яка поєднує матричну факторизацію та глибинне навчання. Модель Neu-MF є нелінійною та має достатньо велику гнучкість, оскільки шар, який відповідає за матричну факторизацію (GMF Layer), а також шари багатошарового персептрона (MLP Layer 1, 2, …, X) окремо вивчають ембедінги та відповідно приховані

взаємодії. Останній прихований шар виконує поєднання (Concatenation) результатів двох підмоделей, які представлені виходами шарів GMF Layer та MLP Layer 1, 2, …, X.

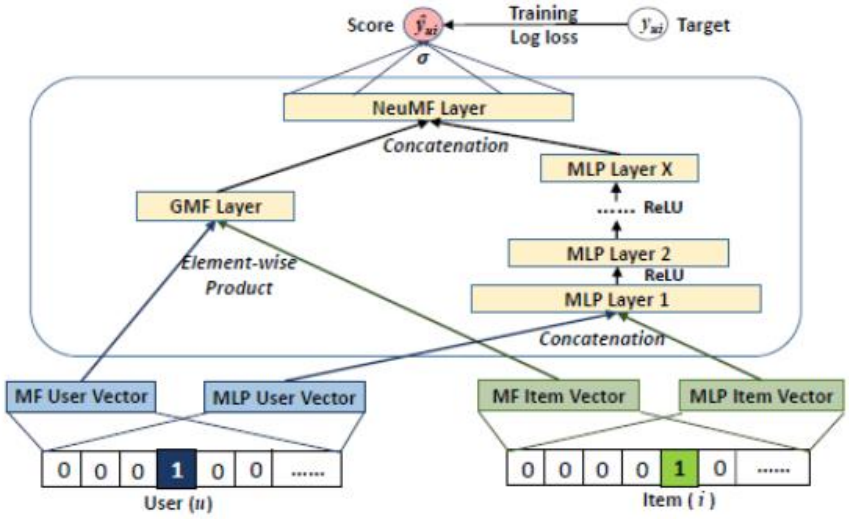

Рисунок 2 - Архітектура моделі Neu-MF

LightGCN – відома графова нейронна мережа, яка є спрощенням Neural Graph Collaborative Filtering (NGCF), але більш точна [5]. LightGCN повністю виключає матриці ваг, що навчаються, і нелінійні функції активації, тому єдиними навчальними параметрами є початкові ембедінги шару для кожного вузла. Це призводить до значного прискорення. Далі LightGCN виконує агрегацію сусідів, що допомагає поширити ембедінги за графом. Таким чином, правило оновлення ваг мережі LightGCN наступне:

$$x_i^{(k+1)} = \sum_{j \in N(i)} \frac{1}{\sqrt{|N(i)|}\sqrt{|N(j)|}} x_j^{(k)} \qquad (11)$$

LightGCN використовує агреговане представлення сусідів як нове представлення центрального вузла, тобто повністю відкидає початкову інформацію про вузол користувача/об'єкта, що може привести до втрати власних вподобань користувача або власних властивостей об'єкту. Для

уникнення цієї проблеми в деяких дослідженнях [5] об'єднують ці два представлення лінійно за допомогою операції підсумовування або середнього підсумовування.

У 2017 році група дослідників опублікувала свою роботу про Нейронну Колаборативну Фільтрацію (NCF) [6]. Вона містить узагальнений фреймворк для вивчення нейронних залежностей, що моделюються факторизацією матриць при колаборативній фільтрації за допомогою нейронної мережі. Автори також пояснили, як отримати залежності вищого порядку (MF має порядок лише 2), і як об'єднати обидва ці підходи.

Загальна ідея у тому, що нейронна мережа теоретично може засвоїти будь-яку функціональну залежність. Це означає, що залежність, яку модель колаборативної фільтрації виражає матричною факторизацією, може бути засвоєна нейронною мережею. NCF пропонує простий шар представлення відразу для користувачів та об'єктів (схожий на класичну факторизацію матриць), за яким слідує проста нейронна мережа на кшталт багатошарового перцептрону, яка повинна засвоїти залежність між уявленнями користувача та об'єкта, аналогічну до факторизованих матриць.

DeepFM – це змішаний підхід, що включає факторизацію матриць і глибоку нейронну мережу, причому обидва використовують один і той самий вхідний шар представлення (embedding)[6]. Необроблені ознаки трансформуються, щоб закодувати категоріальні ознаки унітарним кодом (one-hot encoding). Компонент факторизації матриць – це звичайна Машина Факторизації, оформлена у стилі архітектури нейронної мережі.

У моделі DeepFM кожна категоріальна ознака представлена латентним вектором, а постійні ознаки обробляються багатошаровим перцептроном таким чином, щоб на виході виходили вектори такого ж розміру як латентні вектори. На другому етапі розраховуються взаємні добутки всіх латентних векторів та вихідних векторів перцептрону. Після цього результати добутків

поєднуються разом і передаються в інший багатошаровий перцептрон, а зрештою – у функцію сигмоїди, що видає імовірність.

Модель DLRM можна розглядати як спрощену та модифіковану версію моделі DeepFM, оскільки латентні представлення категоріальних ознак не обов'язково мають проходити через багатошаровий персептрон [6].

Неповністю задані вподобання користувачів обробляються методом Neural Tangent Kernel [7]. Для отримання дрібних але найбільш суттєвих ознак з великої розрідженої матриці взаємодії в [7] запропоновано модель DISTILL-CF. Також використано підхід, орієнтований на дані, який ставить за мету покращити якість відгуків, що в подальшому використовується для моделювання вподобань користувачів. Для цього було використано концепцію диференційованої вибірки Гамбеля, що в подальшому дозволяє обробляти неоднорідність, розрідженість і напівструктурованість даних, а також виконувати масштабування у випадку наборів даних з мілліонами взаємодій користувача з елементом.

Модифікований метод колаборативної фільрації для електронної комерції запропоновано в [2]. Він складається з трьох етапів:

1. Вибір надійних даних. Коефіцієнтом «надійності» відгука є відношення оцінки «так» щодо корисності відгука до загальної оцінки так/ні.

2. Вирахування подібності користувачів на основі коефіцієнта кореляції Пірсона

3. Додавання схожості до вагового коефіцієнта вдосконаленого алгоритму нахилу один.

В [8] задача рекомендацій формулюється наступним чином: «Яка стаття найбільш вірогідно буде обрана користувачем в цьому сеансі?» і розглядається побудова системи рекомендацій новин. Запропоновано модель CHAMELEON глибокого навчання [8] для розв`язання вказаної задачі. Модель складається з двох частин: перша відповідає за вивчення тексту статей і отримання суттєвих ознак з цих текстів, а друга розраховує

рекомендації на основі цих ознак з використанням рекурентних нейронних мереж. Перша частина моделі являє собою згорткову нейронну мережу, друга включає моделювання послідовностей кліків користувача, використовуючи LSTM.

Слід зазначити, що модель CHAMELEON не має проблеми холодного старту і є більш ефективною за деякі інші моделі [8].

В [9] пропонуються методи напіватоматичного навчання SSL, щоб зменшити недостачу даних в задачах побудови рекомендаційних систем та обробляти різноманітний контент через впорядковування вподобань користувачів на основі графів контексту. Пропонований метод PACE об'єднує колаборативну фільтрацію і SSL, поєднує методи матричної факторизації і глибоку нейронну мережу, яка спільно вивчає вподобання користувачів і так звані точки інтересу (POI). Цей метод дозволяє прогнозувати як вподобання користувача, так і контекст, пов'язаний з користувачами і POI.

В [10] розглядається така проблема моделювання рекомендаційних систем як однонаправленість, яка не дозволяє оцінити приховані вподобання в поведінці користувачів. Для її усунення запропоновано модель послідовних рекомендацій BERT4Rec [10], яка використовує двонаправлену самоувагу для моделювання послідовності поведінки користувача. Для усунення іншої проблеми, повязаної з порушенням конфіденційності, у тій же роботі запропоновано використання ланцюга Клоуза до послідовних рекомендацій, з метою передбачити випадкові замасковані елементи в послідовності (контексті). Таким чином, було отримано модель двонаправленого представлення для надання рекомендацій, даючи можливість кожному елементу об'єднувати інформацію з лівої і правої сторони. Ця модель, за результатами представлених в [10] експериментів, дає значне покращення ефективності порівняно з класичними моделями.

Проблеми існуючих алгоритмів рекомендацій, такі як представлення рекомендацій як статичної процедури, ігноруючи динамічність процесу і

довгострокові неявні залежності у послідовностях рекомендацій, розглянуто в [11]. Щоб подолати ці обмеження в [11] запропоновано систему рекомендацій, яка заснована на глибокому навчанні з підкріпленням DRR. Ця система представляє рекомендацію як послідовну процедуру прийняття рішень і використовує схему навчання з підкріпленням «актор-критик» для моделювання взаємодії користувача з системою. Вона може включати в себе динамічну адаптацію і довгострокові зворотні зв'язки. Проведені в [11] експерименти доводять високу ефективність цієї системи у порівнянні з деякими іншими сучасними системами надання рекомендацій.

В [8] розглянуто такий напрямок систем рекомендацій, як оцінка рейтингу кліків, які виконує користувач, вважаючи його в поєднанні з глибоким навчанням дуже перспективним напрямком досліджень. В цій статті автори пропонують модель нейронної мережі Field Attentive Deep Field Factorization Machine (FAT-DeepFFM), яка заснована на моделі CENet оцінки рейтингу кліків з механізмом уваги. Окрім того, автори [8] порівнюють два механізма уваги: увагу до і після явної взаємодії з ознаками, і експериментальним чином демонструють, що перший механізм працює значно краще другого. За результатами експериментів автори [8] доводять, що модель FAT-DeepFFM перевершує за точністю деякі класичні моделі.

Напрямок рекомендаційних систем, як прогноз рейтинга кліків, розглянуто в [7]. В цій же роботі автори звертають увагу на таку проблему вказаного напрямку, як важкість виявлення вагомих ознак зацікавлення, і для цього пропонують нову модель FGCNN автоматичної генерації ознак на основі згорткової нейронної мережі (CNN), яка складається з двох компонентів: створення ознак і глибокий класифікатор. Генерація ознак використовує можливості CNN для створення локальних шаблонів і рекомбінації їх для створення нових ознак. Класифікатор – друга компонента моделі FGCNN – приймає структуру IPNN для вивчення взаємодій з розширеного простору функцій.

Контекстно-орієнтований напрямок розвитку рекомендаційних систем розглянуто в [12] і запропоновано модель Attentive Interaction Network, яка вносить покращення у відомі алгоритми за рахунок адаптивного захоплення взаємодій між контекстами і користувачами.

Метод надання рекомендації складається з наступних етапів:

1. На основі моделі Attentive Interaction Network описуються ефекти взаємодії контекстів з користувачами, а також враховуючи дві складові, орієнтовані окремо на користувача і на елемент, моделюється вплив взаємодії окремо на користувача і на представлення предметів.

2. Представлення користувачів і елементів під ефектами взаємодії поєднуються і в результаті розраховуються прогнозні значення оцінок рекомендацій.

3. Після цього в моделі використовується механізм уваги для агрегації багатьох ефектів взаємодії.

Експериментальним шляхом в [12] доводиться, що наведений вище метод працює краще аналогічних сучасних систем в цьому напрямку, а також здатний надати кращі пояснення стосовно контексту, ніж існуючі підходи.

В статті [14] автори розглядають промислові рекомендаційні системи, які складаються з етапу зіставлення і етапу ранжування, щоб обробляти міліарди користувачів і елементів. На етапі зіставлення вилучаються елементи-кандидати, які відповідають інтересам користувача, а не етапі ранжування елементи-кандидати сортуються за інтересами користувача.

Автори наголошують, що більша частина існуючих моделей представляють користувача як єдиний вектор, і цього недостатньо для того, щоб представляти різноманіття інтересів користувача.

Автори пропонують нову мультиінтересну мережу з динамічною маршрутизацією (MIND) для вирішення цієї проблеми. Ця система представляє користувача декількома векторами в залежності від аспектів його інтересу. Автори розробляють шар екстрактора з декількома інтересами на

основі механізму маршрутизації капсул, який можна використовувати для кластеризації історичної поведінки і вилучення різноманітних інтересів.

Окрім того, в системі присутній метод «Увага з урахуванням міток», який допомагає вивчити представлення користувача з декількома методами.

За результатами проведених експериментів можна побачити, що MIND досягає більшої продуктивності аналогічних методів.

В статті [15] розглядається напрямок прогнозу рейтинга кліків. Автори намагаються вирішити складну проблему цього напрямку: вхідні ознаки зазвичай розріджені і ефективне передбачення спирається на комбінаторні ознаки високого порядку, для яких треба дуже багато часу для ручного створення експертами в предметній області.

В статті пропонується метод AutoInt для автоматичного вивчення ознак високого порядку вхідних ознак. Запропонований алгоритм дуже універсальний, тому його можна використовувати не тільки з числовими, так і з категоріальними вхідними ознаками, які відображаються в маловимірний простір. Після цього пропонується багатоголова нейронна мережа з самостійною увагою і залишковими зв’язками для явного моделювання взаємодій функцій в низьковимірному просторі. За допомогою разних шарів нейронних мереж з самостійною увагою і декількома головами можна моделювати різні порядки комбінацій ознак вхідних ознак. Вся модель може бути ефективно адаптована до великих необроблених даних.

Експерименти, проведені на чотирьох наборах даних, показують, що запропонований підхід кращий за аналогічні і надає хороші пояснення.

Розглянемо моделі, що використовують текстові огляди для рекомендацій (рис. 3, 4).

```
Model(
 (word2vec): Embedding(53775, 64)
 (user_conv): TextCNN(
 (convs): ModuleList(
 (0): Conv2d(1, 100, kernel_size=(3, 64), stride=(1, 1), padding=(2, 0))
 )
 (fc): Linear(in_features=100, out_features=10, bias=True)
 (dropout): Dropout(p=0.6, inplace=False)
 )
 (item_conv): TextCNN(
 (convs): ModuleList(
 (0): Conv2d(1, 100, kernel_size=(3, 64), stride=(1, 1), padding=(2, 0))
 )
 (fc): Linear(in_features=100, out_features=10, bias=True)
 (dropout): Dropout(p=0.6, inplace=False))
(final): Sequential(
 (0): Linear(in_features=20, out_features=10, bias=True)
 (1): ReLU()
 (2): Dropout(p=0.6, inplace=False)
 (3): Linear(in_features=10, out_features=1, bias=True)
 )
 (fm): TorchFM(
 (lin): Linear(in_features=20, out_features=1, bias=True)
 )
 (dropout): Dropout(p=0.6, inplace=False)
 (relu): ReLU()
)
```

Рисунок 3 – Модель DeepCoNN формування рекомендацій, що враховує текстові огляди [16]

```
Model(
 (word2vec): Embedding(53835, 64)
 (user_embedding): Embedding(87273, 10)
 (item_embedding): Embedding(13211, 10)
 (user_conv): TextCNN(
 (convs): ModuleList(
 (0): Conv2d(1, 100, kernel_size=(3, 64), stride=(1, 1), padding=(2, 0)))
 (fc): Linear(in_features=100, out_features=10, bias=True)
 (dropout): Dropout(p=0.6, inplace=False))
 (item_conv): TextCNN(
 (convs): ModuleList(
 (0): Conv2d(1, 100, kernel_size=(3, 64), stride=(1, 1), padding=(2, 0)))
 (fc): Linear(in_features=100, out_features=10, bias=True)
 (dropout): Dropout(p=0.6, inplace=False))

(attention_scorer_user): Sequential(
 (0): Linear(in_features=20, out_features=10, bias=True)
 (1): ReLU()
 (2): Dropout(p=0.6, inplace=False)
 (3): Linear(in_features=10, out_features=1, bias=True))
 (attention_scorer_item): Sequential(
 (0): Linear(in_features=20, out_features=10, bias=True)
 (1): ReLU()
 (2): Dropout(p=0.6, inplace=False)
 (3): Linear(in_features=10, out_features=1, bias=True))

(final): Sequential(
 (0): Dropout(p=0.6, inplace=False)
 (1): Linear(in_features=10,
out_features=10,  bias=True)
 (2): ReLU()
 (3): Linear(in_features=10,
out_features=1,  bias=True) )
 (dropout): Dropout(p=0.6, inplace=False)
 (sigmoid): Sigmoid()
 (relu): ReLU())
```

Рисунок 4 – Модель NARRE формування рекомендацій, що враховує текстові огляди [17]

DeepCoNN [16] – глибока модель спільного вивчення властивостей предмета та поведінки користувача з тексту огляду. Модель складається з двох паралельних нейронних мереж, з’єднаних на останніх рівнях. Одна з

мереж зосереджена на вивченні поведінки користувачів, використовуючи відгуки, написані користувачем, а інша вивчає властивості предмета з відгуків, написаних для товару. Спільний рівень об’єднує ці дві мережі. Спільний рівень дозволяє латентним факторам, отриманим користувачами та елементами, взаємодіяти один з одним у спосіб, подібний до методів машинної факторизації.

NARRE [17] – аналізує важливість кожного огляду перед використанням. Автори стверджують, що менш корисні відгуки шкодять продуктивності моделі, а також менш значущі для користувача. Для дослідження важливості оглядів модель використовує механізм уваги.

HFT [18] – особлива модель у цьому списку, використовує огляди як регуляризатор. По суті, така модель не просто використовує текст огляду для визначення прихованих вподобань, але й визначає, наскільки об’єктивною (відносно тексту огляду) є відповідна оцінка.

TransNets [19] – розширення моделі DeepCoNN, що вводить додатковий латентний рівень, що представляє пару користувач-елемент. Під час навчання цей рівень впорядковується, щоб бути схожим на інше приховане представлення огляду цільового користувача цільового елемента.

MPCN [20] – модель з багатоієрархічною парадигмою, що ґрунтується на припущенні, що не всі огляди однакові, тобто важливі лише кілька обраних. Важливість, однак, має динамічно виводитися залежно від поточної цілі. З цією метою пропонується схема навчання на основі вказівника рецензії за рецензією, яка виділяє важливі рецензії, а потім зіставляє їх слово за словом. Це дає змогу не лише використовувати найбільш інформативні огляди для прогнозування, але й глибше взаємодіяти на рівні слів.

NRPA [21] припускає, що важливість відгуків різна для різних користувачів і елементів, а також одне і те ж слово або схожі відгуки можуть мати різну інформативність для різних користувачів і товарів. Тому модель використовує персоналізовану увагу, щоб врахувати це.

DAML[22] – модель, що використовує локальну та взаємну увагу згорткової нейронної мережі, щоб спільно вивчати особливості оглядів для покращення інтерпретації. Потім рейтингові функції та функції перегляду інтегруються в уніфіковану модель нейронної мережі, а нелінійна взаємодія ознак вищого порядку реалізується машинами нейронної факторизації для завершення прогнозу остаточного рейтингу.

2.3 Метод дропауту (dropout) для зменшення перенавчання глибоких моделей нейронних мереж

Одна з проблем, яка виникає під час навчання глибоких нейронних мереж (Deep Neural Networks) – це перенавчання або оверфітінг (overfitting), а саме, модель добре пояснює приклади з навчальної вибірки, цілком адаптується до навчальних прикладів, але не вчиться розуміти закономірності в даних, не може точно класифікувати приклади, які не брали участі в навчанні [12]. В останні роки для рішення проблеми перенавчання використовується дропаут (dropout, рис. 5).

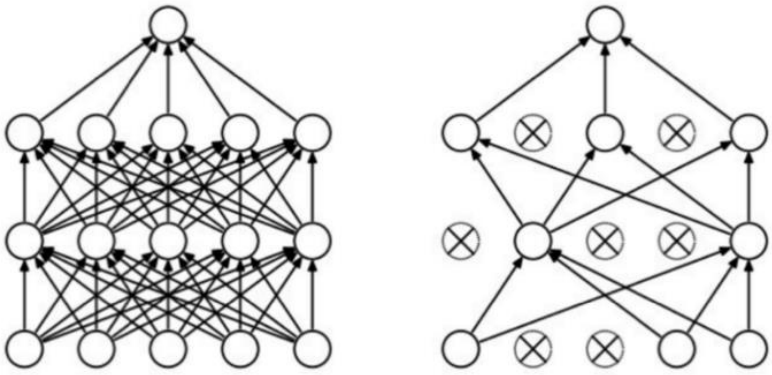

Рисунок 5 – Модель багатошарового персептрона до і після Dropout [13]

Зліва – початкова нейронна мережа, а праворуч – та ж мережа після Dropout. Головна ідея дропауту в тому, що замість навчання однієї глибокої нейронної мережі навчити ансамбль кількох глибоких нейронних мереж з наступним усередненням отриманих результатів.

Мережі – складові ансамбля отримуються шляхом виключення з мережі (dropping out) нейронів із заданою ймовірністю p (параметр) [12]. Тому, імовірність того, що нейрон залишиться в мережі, становить $q = 1 - p$. Виключення нейрона з мережі означає, що при будь-яких вхідних даних або параметрах цей нейрон повертає нуль.

Виключені нейрони не приймають участі в навчанні на жодному етапі алгоритму зворотного розповсюдження помилки, як наслідок, виключення хоча б одного з нейронів рівносильне навчанню нової нейронної мережі.

Нейрони вимикаються з ймовірністю p, тому $q = 1 - p$ – це імовірність залишення нейронів включеними. Імовірність виключення кожного нейрона є однаковою.

Припустимо, що $h(x) = xW + b$ – лінійна проекція вхідного d(i)-мірного вектора x на d(h) мірний простір вихідних значень; $a(h)$ – функція активації. Тоді застосування дропауту до даної проекції на етапі навчання можна представити як змінену функцію активації:

$$F(h) = D \odot a(h), \tag{12}$$

де $D = (x_1, \dots, x_{dh})$ – це вектор випадкових велечин X_i, розподіленний за законом Бернуллі, тобто вектор X_i, що має розмірність $d_h - 1$, має наступний розподіл ймовірностей:

$$f(k; p) = \begin{cases} p, \text{якщо } k = 1 \\ 1 - p, \text{якщо } k = 0 \end{cases} \tag{13}$$

де k – це усі можливі вихідні значення.

Дійсно випадкова величина за розподілом Бернулі ідеально відповідає дропауту, який застосовано до одного нейрона, оскільки нейрон вимикають з ймовірністю $p = P(k = 1)$, а в іншому випадку нейрон залишають включеним.

Застосування дропауту до і-го нейрона:

$$O_i = X_i a\left(\sum_{k=1}^{d_i} w_k x_k + b\right) = \begin{cases} a\left(\sum_{k=1}^{d_i} w_k x_k + b\right), \text{якщо } X_i = 1, \\ 0, \text{якщо } X_i = 0 \end{cases} \quad (14)$$

де $P(X_i = 0) = p$.

Оскільки під час навчання нейрон залишається в мережі з ймовірністю q, то на етапі тестування необхідно провести емуляцію поведінки ансамблю нейронних мереж, який було використано при навчанні. Для цього на етапі тестування треба помножити функцію активації на значення q.

Отже, на етапі навчання:

$$O_i = X_i a\left(\sum_{k=1}^{d_i} w_k x_k + b\right) \quad (15)$$

Під час тестування:

$$O_i = qa\left(\sum_{k=1}^{d_i} w_k x_k + b\right) \quad (16)$$

2.4Методи Глоро і Хе для ініціалізації ваг глибоких нейронних мереж

Відомо, що нейронна мережа чутлива до початкової ініціалізації параметрів. Методи ініціалізації ваг Глоро (Xavier/Glorot initialization) та Хе (Kiming He) покращують як швидкість, так і якість алгоритмів навчання глибоких нейронних мереж [14].

Розглянемо значення одного нейрону (до застосування функції активації):

$$y = w^T x + b = \sum_{i=1}^{n\_out} w_i x_i + b \tag{17}$$

де x – вектор вхідних значень, w – вектор параметрів. Позначимо і-й член суми як $y_i = w_i x_i$.

Припустимо, що x_i та w_i – незалежні, тоді:

$$Var(y_i) = Var(w_i x_i) = E[w_i x_i] - (M[w_i x_i])^2$$
$$= E[x_i]^2 Var(w_i) + E[w_i]^2 Var(x_i) + Var(x_i) Var(w_i) \tag{18}$$

де E[·] – оператор взяття математичного сподівання.

Таким чином, дисперсія $Var(y_i)$ не залежить від зміщення (члена b), і залежить лише від вектора x вхідних значень і w - вектора параметрів.

Нехай тепер функція активації є симетричною, наприклад логістичний сигмоїд або гіперболічний тангенс, а ваги ініціалізуються випадковим чином з нульовим середнім значенням. Тоді:

$$Var(y_i) = Var(x_i) Var(w_i) \tag{19}$$

Нехай xi і wi ініціалізуються незалежно один від одного і з одного розподілу, тоді дисперсія дорівнює:

$$D(y) = D\left(\sum_{i=1}^{n_{out}} y_i\right) = \sum_{i=1}^{n_{out}} D(w_i x_i) = n_{out} D(w_i) D(x_i) \tag{20}$$

де *n_out* – кількість нейронів результуючого (вихідного) шару.

Таким чином, отримано, що дисперсія виходів $Var(y_i)$ пропорційна дисперсії входів з коефіцієнтом $n\_out\ Var(w_i)$.

Ініціалізація Глоро – це ініціалізація ваг w_i симетричним розподілом з дисперсією

$$Var(w_i) = \frac{2}{n\_in + n\_out} \tag{21}$$

Наприклад,

• нормальним розподілом з нульовим середнім і дисперсією (21) або

• рівномірним розподілом з інтервалу [-a,a],

$$a = \sqrt{\frac{6}{n_{in} + n_{out}}} \tag{22}$$

Розглянемо шар з несиметричною функцією активації, наприклад, ReLU та її модифікації. Тоді вираз

$$Var(y_i) = Var(w_i x_i) = E[x_i]^2 Var(w_i) + E[w_i]^2 Var(x_i) + Var(x_i)Var(w_i) \tag{23}$$

перетвориться на наступний:

$$Var(y_i) = E[x_i]^2 Var(w_i) + Var(x_i)Var(w_i) = Var(w_i)E[x_i^2]. \tag{24}$$

Ініціалізація Хе – це ініціалізація ваг w_i нейронів *l*-го шару за

- нормальним законом з нульовим середнім і дисперсією

$$Var(w_i) = \frac{2}{n\_in^l} \tag{25}$$

де n_in^l - кількість нейронів *l*-го шару

- або рівномірним законом аналогічно до ініціалізації Глоро.

Перевага методів Глоро і Хе в простоті їх реалізації і відносно високій якості ініціалізації. Обмеження – не враховуються апріорні знання.

2.5Методи навчання глибоких нейронних мереж

Метод стохастичного градієнтного спуску (SGD)

Градієнтні методи представляють широкий клас методів оптимізації, які використовують не тільки в машинному навчанні. Градієнтний підхід розглядається як спосіб налаштування векторів синаптичних ваг w в шарах глибокої нейронної мережі. Нехай $y^* : X \to Y$ – цільова залежність, відома на об'єктах навчальної вибірки:

$$X^l = \left(x_j, y_i\right)^l_{i=1,} y_i = y^*(x_j) \tag{26}$$

Знайдемо алгоритм a(x,w) що наближає залежність y*. Для лінійного класифікатора цей алгоритм має вигляд:

$$a(x, w) = \varphi(\sum_{i=1}^{n} w_j x^j - w_0) \tag{27}$$

де $\varphi(z)$ представляє функцію активації.

Метод стохастичного градієнтного спуску (SGD):

Вхід:

X^l - навчальна вибірка

η - темп навчання

λ - параметр згладжування функціоналу Q

Вихід:

Вектор ваг w

Тіло:

Ініціалізувати ваги w_j $j = 0, .., n$.

Ініціалізувати поточну оцінку функціоналу:

$$Q \coloneqq \sum_{i=1}^{l} L(a(x_i, w), y_i) \quad (28)$$

Повторювати:

- Вибрати певним чином об'єкт x_i із X^l , наприклад, випадково.
- Обчислити результат алгоритму $a(x_i, w)$ та помилку:
-

$$\xi_i \coloneqq L(a(x_i, w), y_i) \quad (29)$$

Виконати крок градієнтного спуску:

$$w \coloneqq w - \eta \grave{L_a}(a(x_i, w), y_i)\grave{\varphi}(< w, x_i >)x_i \quad (30)$$

Оцінити функціонал якості:

$$Q \coloneqq (1 - \lambda)Q + \lambda_{\xi_i} \quad (31)$$

Поки значення Q не стабілізується та/або ваги w не припинять змінюватись.

Метод Адам

Adam «adaptive moments» (адаптивні моменти) – алгоритм оптимізації з адаптивною швидкістю навчання. Adam є комбінацією алгоритмів RMSProp та імпульсного з декількома відмінностями [17]. Adam вважається досить стійким до вибору гіперпараметрів, хоча швидкість навчання іноді потрібно брати відмінною від пропонованої за замовчуванням.

Алгоритм:

Дано:

- 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).
- невелика константа δ для забезпечення чисельної стійкості.
- початкові значення параметрів θ.

Крок 1. Ініціалізувати змінні для першого і другого моментів s = 0, r = 0

Ініціалізувати крок за часом t = 0

Крок 2. while критерій зупинки не виконано do

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 ⊙ g

Скорегувати зміщення першого моменту:

$$\hat{s} \leftarrow \frac{s}{1-p_1^t} \tag{32}$$

Скорегувати зміщення другого моменту:

$$\hat{r} \leftarrow \frac{r}{1-p_1^t} \tag{33}$$

Обчислити приріст ваги:

$$\Delta\theta = -\xi \frac{s}{\sqrt{\hat{r}}+\delta} \tag{34}$$

Оновити ваги: θ ← θ + Δθ. end while.

У методі Adam ваги налаштовуються наступним чином:

$$S_t := a * S_{t-1} + (1 - a) * \nabla E_t^2; S_0 := 0 \quad (35)$$

$$D_t := \beta * D_{t-1} + (1 - \beta) * \nabla E_t; D_0 := 0 \quad (36)$$

$$g_t := \frac{D_t}{1-\beta} * \sqrt{\frac{1-a}{S_t}} \quad (37)$$

$$\Delta W_t := \eta * (g_t + p * W_{t-1}) + \mu * \Delta W_{t-1} \quad (38)$$

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

2.6 Критерії якості задачі надання рекомендацій

Якість надання рекомендацій за обраним алгоритмом можна оцінити, використовуючи різні типи вимірювань. Вимірювати можна точність або охоплення.

Точність – це частка правильних рекомендацій від загальної кількості можливих рекомендацій. Охоплення – це частка об’єктів у просторі пошуку для яких система може надати рекомендації. Метрики для вимірювання точності, у свою чергу, поділяються на статистичні та метрики точності. Придатність кожного показника залежить від особливості набору даних і предметної області [4].

Найпопулярнішим статистичним показником точності є MAE – середня абсолютна похибка. Чим менше MAE, тим точніше прогнози. Вона обчислюється наступним чином:

$$MAE = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} \tag{30}$$

$$MAE = mean\ absolute\ error$$

$$y_i = prediction$$

$$x_i = true\ value$$

$$n = total\ number\ of\ data\ points$$

RMSE – середньоквадратична помилка – більш сильно штрафує неточні прогнози, тобто оцінка більш вимоглива:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{40}$$

Існують більш специфічні метрики якості. Наприклад, стаття пропонує наступне:

Precision – це відношення отриманих релевантних елементів до загальної кількості елементів, де кількість задано параметром k [23]. Релевантними вважаємо ті, з якими користувач взаємодіяв за тестовий період. Цільова взаємодія може бути різною: перегляд, оцінка, покупка – залежно від цілей системи.

$$Precisions@k = \frac{1}{k}\sum_{i=1}^{k} Relevance@i \tag{41}$$

Щоб врахувати як вдалі попадання в top-k, так і те, наскільки ці релевантні елементи близькі початку списку, використовується метрика

AP@k. У цій метриці підсумовуються різні значення Precision@k для яких у позиції k виявився релевантний елемент:

$$AP@N = \frac{1}{m}\sum_{k=1}^{N}(P(k) if\ k^{th} item\ was\ relevant) = \frac{1}{m}\sum_{k=1}^{N} P(k) * rel(k) \quad (42)$$

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

Метрика MAP@k – це усереднення значень AP@k по набору тестових користувачів:

$$MAP@k = \frac{1}{|Users|}\sum_{u \in Users} AP@k(u) \quad (43)$$

Ще одна метрика, що враховує позиції релевантних елементів – це Normalized Discounted Cumulative Gain, яка оцінює близькість ранжування до деякого ідеального значення, наприклад, реальний порядок взаємодій користувача з об'єктами:

$$nDCG_p = \frac{DCG_p}{I\ DCG_p} \quad (44)$$

$$I\ DCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i}-1}{log_2(i+1)} \quad (45)$$

F1@k – середнє гармонійне значення precision@k і recall@k, яке допомагає спростити їх в єдину метрику [24]. Усі перераховані вище показники можна розрахувати на основі матриці неточності (confusion matrix).

$$Precision = \frac{TP}{TP+FP} \quad (46)$$

$$Recall = \frac{TP}{TP+FN} F1 = 2 * \frac{precision*recall}{precision+recall} \quad (47)$$

$$TP = True\ positive$$

$$TN = True\ negative$$

$$FP = False\ positive$$

$$FN = False\ negative$$

Використовують також коефіцієнт Метьюза – коефіцієнт кореляції між спостережуваною та передбачуваною бінарною класифікацією:

$$Matthews\ correlation\ coefficient\ (MCC) = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}} \quad (48)$$

Вважається, що одні лише метрики точності не можуть об'єктивно оцінити якість рекомендаційної системи [24]. До більш абстрактних метрик відносяться:

- Різноманітність – кількість наданих релевантних рекомендацій різних типів, які дозволять користувачу знайти щось відмінне від того, що він обирає зазвичай, але все таки відповідає його вподобанням.
- Охоплення – здатність системи рекомендувати користувачам усі предмети з набору даних. Це означає, що система не повинна допускати ситуації, що нові товари ніколи не рекомендуються користувачам, адже не мають відгуків, оцінок і взаємодій з іншими користувачами.
- Покриття простору користувачів – може характеризуватися часткою користувачів або взаємодій користувачів, для яких система може надати рекомендації. Наприклад, деякі системи не можуть надавати рекомендації якщо про користувача відомо занадто мало інформації.

- Новизна рекомендацій – показник, який визначає частку рекомендованих нових об’єктів без оцінок і відгуків від інших користувачів
- Неочікуваність – відповідає за показ елементів, які ніколи не були рекомендовані користувачу, не зважаючи на його історію взаємодій. Цей параметр може негативно вплинути на точність рекомендацій, але підвищити параметр покриття.
- Надійність – показник, який визначає те, чи довіряє користувач системі рекомендацій.
- Відтік – частота змін рекомендацій після того, як користувач оцінює нові рекомендації.

А також важливими є такі бізнес-метрики як кількість кліків, кількість продажів, вплив на розподіл продажів та інше.

2.7 Пропонована модифікація нейромережевої моделі формування рекомендацій та її обґрунтування

Нехай дано:

- множину користувачів $u \in U$, множину об’єктів $i \in I$,
- множину подій $(r\_ui,\ u,\ i) \in D$ – діяльність, що виконують над об’єктами,
- множину текстів, яка відповідає відгуку, написаному користувачем для відповідного об’єкту.

Кожна з подій задана відповідним результатом r_ui, а також іншими характеристиками.

Знайти: прогноз і рекомендації

для заданого користувача - user based recommendations або

для заданого товару - item based recommendations.

Для вирішення задачі дослідження було обрано DeepConn представлену у статті [16] – модель глибокого навчання для моделювання користувачів і

продуктів окремо, яка використовує наскрізне навчання. Дві паралельні згорткові структури використовуються для моделювання текстів огляду користувачів і продуктів відповідно. Далі метод Factorization Machine використовується для прогнозування рейтингу.

Пропонована модифікація (рис.6, 7) базується на наступних ідеях:

1. За основу береться відома нейромережева модель DeepCoNN

2. Використовується Textual Preprocessing для аналізу вхідних текстів

3. Додається шар Attention на етапі зв’язування текстів оглядів користувачів і продуктів

4. Використовуються зміщення (bias) глобальні, користувачів і продуктів замість машини факторизації

```
Model(
  (word2vec): Embedding(42455, 64)
  (user_conv): TextCNN(
    (convs): ModuleList(
      (0): Conv2d(1, 100, kernel_size=(3, 64), stride=(1, 1), padding=(2, 0))
    )
    (fc): Linear(in_features=100, out_features=10, bias=True)
    (dropout): Dropout(p=0.6, inplace=False)
  )
 (item_conv): TextCNN(
    (convs): ModuleList(
      (0): Conv2d(1, 100, kernel_size=(3, 64),
stride=(1, 1), padding=(2, 0))
    )
    (fc): Linear(in_features=100, out_features=10, bias=True)
    (dropout): Dropout(p=0.6, inplace=False)
  )
  (attention): Sequential(
    (0): Linear(in_features=128, out_features=10, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.6, inplace=False)
    (3): Linear(in_features=10, out_features=1, bias=True)
  )
  (final): Sequential(
    (0): Dropout(p=0.6, inplace=False)
    (1): Linear(in_features=20, out_features=10, bias=True)
    (2): ReLU()
    (3): Linear(in_features=10, out_features=1, bias=True)
  )
  (dropout): Dropout(p=0.6, inplace=False)
  (relu): ReLU()
```

Рисунок 6 – Пропонована модифікована нейромережева модель формування рекомендацій

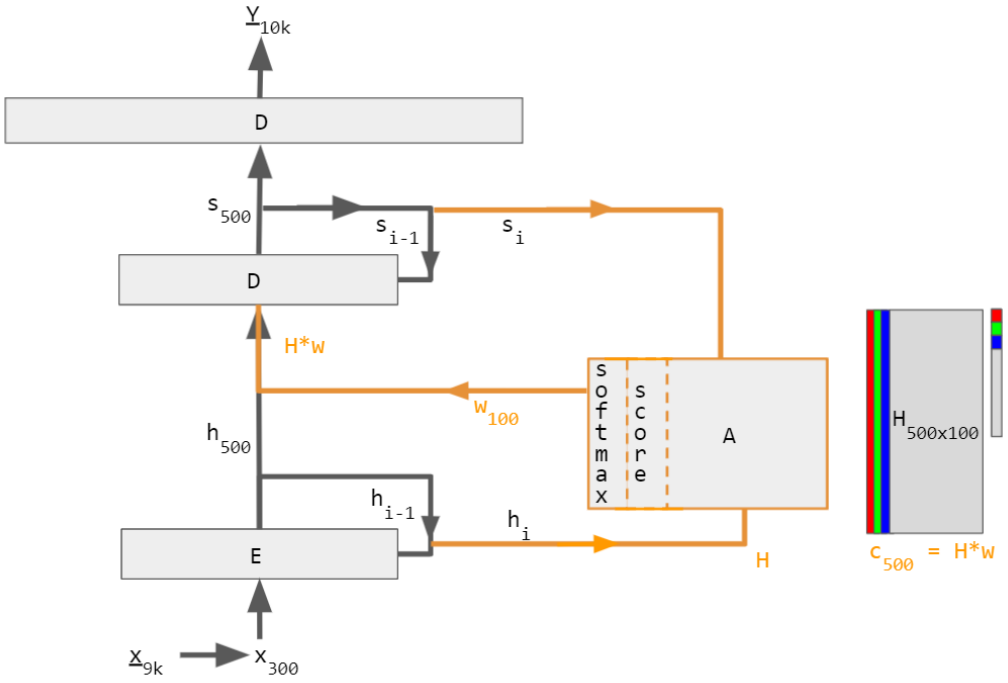

Рисунок 7 – шар Attention на етапі зв’язування текстів оглядів користувачів і продуктів [25]

Розглянемо технологію Textual Preprocessing для аналізу вхідних текстів (рис.8).

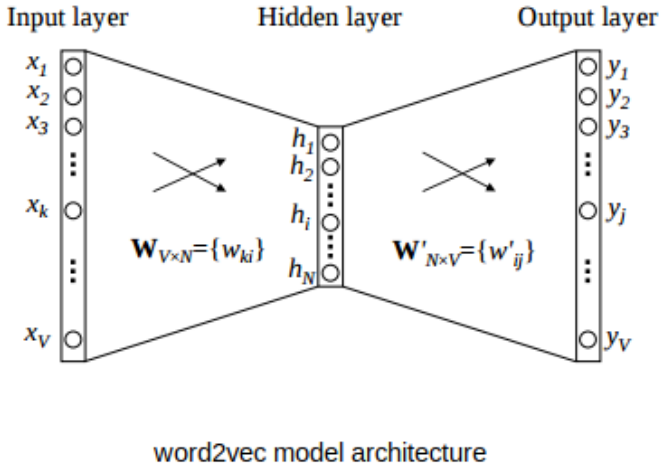

Рисунок 8 – Архітектура моделі Word2vec [26]

Word2vec – це техніка обробки природної мови (NLP), опублікована в 2013 році, яка використовує модель нейронної мережі для вивчення

асоціацій слів із великого масиву тексту. Кожне окреме слово представляється числовим вектором. Ці вектори навчаються так, щоб вони відображали семантичні та синтаксичні якості слів. Після навчання модель Word2vec може виявляти слова-синоніми або пропонувати додаткові слова для окремого речення.

2.8 Висновки до розділу 2

В даному розділі було розглянуто математичні основи існуючих методів рекомендацій товарів на основі різних підходів. Також були розглянуті алгоритми найбільш нових методів вирішення задачі надання рекомендацій.

Було проаналізовано і обрано методи оцінювання якості надання рекомендацій.

Запропоновано і обґрунтовано модифікацію методу DeepConn для підвищення точності рекомендацій.

РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МОДЕЛЕЙ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ФОРМУВАННЯ РЕКОМЕНДАЦІЙ

3.1. Опис і передобробка вхідних даних

Для дослідження алгоритмів, що використовують огляди для рекомендацій, в якості датасету буде використано Amazon product data [27]. Цей набір даних містить огляди продуктів і метадані від Amazon, включаючи 142,8 мільйона оглядів за період з травня 1996 року по липень 2014 року.

Набір даних також включає в себе малі підмножини даних, поділені на категорії, а також 5-core дані по категоріям, що означає, що у кожного продукту і користувача серії є як мінімум 5 відгуків, що спрощує навчання. Окремо можна завантажити метадані, які включають в себе зображення, інформацію про «купують разом» і окремі характеристики продукту, але для дослідження нам буде достатньо лише 5-core файлу.

Для урізноманітнення тестів було обрано 4 датасети різних розмірів, інформацію про які наведено в таблиці 1.

Таблиця 1 - Інформація про використані датасети

Dataset	Rewiews
Apps for Android	752,937
Video Games	231,780
Cell Phones and Accessories	194,439
Office Products	53,258

Для дослідження датасети були розділені по схемі 80-10-10 на сети для навчання, тестування і перевірки. Відповідно до налаштувань оригінальної моделі, у NARRE ми не видаляємо стоп-слова та підтримуємо словник із 50 тисяч найуживаніших слів. Для продуктивності ми використовуємо 64-вимірні вбудовування word2vec, навчені за допомогою Gensim3.

Аналогічним чином для DeepCoNN ми обмежуємо/доповнюємо довжину документа користувача/елементу до 1000 токенів, а для інших методів ми обмежуємо/доповнюємо довжину кожного огляду.

3.2. Дослідження точності деяких існуючих алгоритмів

Для дослідження точності алгоритмів, заснованих на оглядах, були використані два відомі алгоритми, що використовують огляди для рекомендацій: DeepCoNN і NARRE.

Також для об’єктивності дослідження була використана одна базова модель, що не працює з оглядами. NeuMF – це проста і розповсюджена модель нейронної колаборативної фільтрації, що використовує машину факторизації для заповнення пропущених зв’язків

Для об’єктивності дослідження всі моделі були поставлені у однакові умови:

'weight_decay': float(1e-6),
'lr': 0.002,
'epochs': 12,
'batch_size': 128,
'latent_size': 10,
'word_embed_size': 64,
'dropout': 0.6

І було проведено перше дослідження на вищевказаних датасетах. Для більшої об’єктивності результатів було використано три метрики точності:

MSE (середньоквадратична помилка) - вимірює усереднення квадратів похибок — тобто, середнє квадратичної різниці між оцінками значень та справжнім значенням. Середньоквадратична помилка є функцією ризику, яка відповідає математичному сподіванню квадрату похибкових втрат.

nDCG@5 - відношення DCG (вимірює корисність або приріст документа на основі його позиції в списку результатів) рекомендованого порядку до DCG ідеального порядку.

HR@1 – спосіб вимірювання точності, за яким ми беремо сет з 6 ітемів, причому п’ять з них користувач оцінив не на максимальний бал, а 1 – на максимальний і вимірюємо, скільки разів в середньому програма правильно поставила на перше місце предмет, оцінка якого була максимальна. Ця метрика була використана лише для тесту моделі після закінчення навчання для загальної оцінки ефективності моделі.

Результати дослідження надано в Таблицях 2, 3, 4. Жирним виділений кращий результат по датасету.

Таблиця 2 - Початкове дослідження MSE

	Моделі		
Датасети	DeepCoNN	NeuMF	NARRE
Android Apps	1.5078	**1.3843**	1.3873
Video Games	1.2737	**1.1203**	1.1549
Cell Phones	1.4073	**1.3384**	1.3488
Office products	0.8238	**0.7182**	0.7963

Таблиця 3 - Початкове дослідження NDCG

	Моделі		
Датасети	DeepCoNN	NeuMF	NARRE
Android Apps	**1.1408**	1.2186	1.2027
Video Games	1.4419	**1.1939**	1.2484
Cell Phones	1.4073	**1.2217**	1.2448
Office products	1.2761	**1.1254**	1.1342

Таблиця 4 - Початкове дослідження HR@1

	Моделі		
Датасети	DeepCoNN	NeuMF	NARRE
Android Apps	35.32	30.39	**39.05**
Video Games	30.91	**43.33**	31.97
Cell Phones	18.18	**30.00**	27.27
Office products	22.22	**45.45**	43.75

Як можна побачити, для таких вхідних параметрів використання відгуків робить результат нижчим, ніж у звичайної колаборативної фільтрації. Це можна пояснити високою розрідженістю представлених датасетів. Окремо варто виділити той факт, що результати моделювання на найменшому датасеті Office products виявилися в цілому найбільш чіткими незалежно від використаного алгоритма.

В рамках другого дослідження було проведене навчання протягом 10 епох, інші параметри залишилися без зміни. Результат дослідження наведений в Таблицях 5 і 6. Чорним позначена епоха з найкращим результатом для моделі і датасету.

Таблиця 5 - Дослідження моделей протягом 10 епох, метрика MSE

	Моделі MSE											
Epoch	DeepCoNN				NeuMF				NARRE			
Датасет	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products
1	1.4998	1.3319	1.4929	0.8669	1.4233	1.1293	1.3472	0.7241	1.4527	1.2477	1.378	0.8312
2	1.4769	1.2687	1.4371	0.8389	1.3952	1.1085	1.3248	0.7206	1.4022	1.1691	1.3195	0.7836
3	1.4724	1.2454	1.4234	0.8133	**1.3895**	1.1009	1.3236	**0.7198**	1.3795	1.1307	1.3001	0.7591
4	1.4597	1.2253	1.408	0.7935	1.3898	**1.1006**	**1.3301**	0.7235	**1.3737**	1.1134	**1.2995**	0.7457
5	1.4656	1.2167	1.4029	0.7824	1.3927	1.1035	1.3365	0.7236	1.3742	1.1044	1.307	0.739
6	1.4491	1.2162	1.4035	0.7748	1.3977	1.1055	1.3457	0.7272	1.3785	**1.1022**	1.3171	0.7343
7	1.4555	1.1955	1.397	0.7679	1.4042	1.1084	1.3524	0.7282	1.3838	1.1028	1.3284	**0.7317**
8	1.4475	1.1937	1.3972	0.7618	1.4104	1.1098	1.3615	0.7292	1.3895	1.1065	1.3392	0.7338
9	1.4519	1.1917	1.3944	0.7569	1.4165	1.1143	1.3685	0.7306	1.3957	1.1101	1.3513	0.7323
10	**1.4434**	**1.1807**	**1.3941**	**0.7533**	1.4266	1.1186	1.3745	0.735	1.4017	1.1138	1.3595	0.7337

Таблиця 6 - Дослідження моделей протягом 10 епох, метрика NDCG

	Моделі NDCG											
Epoch	DeepCoNN				NeuMF				NARRE			
Датасет	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products
1	1.2497	1.4044	1.4624	1.234	1.2605	1.2064	1.2639	1.1371	1.2819	1.2185	1.3118	1.0759
2	1.2703	1.3882	1.3754	1.2316	**1.2371**	1.1853	1.2504	1.1282	1.2477	1.1891	1.2736	1.0759
3	1.2341	1.3981	1.3637	1.2289	1.236	1.1734	1.2387	1.1333	1.2456	1.1839	1.2467	1.0881
4	1.2555	1.3846	1.2948	1.212	1.2403	1.168	1.2386	1.1168	1.2411	1.1559	1.2446	1.072
5	**1.1993**	1.3886	1.2941	1.2509	1.2433	1.1515	1.2578	1.1305	1.2405	1.1581	1.2236	1.0616
6	1.2618	1.4076	1.3317	1.2479	1.2429	1.1529	**1.2285**	**1.1182**	1.2345	1.1331	1.2096	1.0634
7	1.2185	1.3618	1.2773	1.2371	1.2528	1.1513	1.2586	1.1349	1.2328	1.1358	1.2121	1.0674
8	1.2303	1.3651	1.2853	1.2288	1.255	1.1531	1.2571	1.1348	1.2326	1.1291	1.2119	**1.0437**
9	1.2101	1.3792	**1.261**	**1.1774**	1.2427	1.1498	1.2649	1.1425	1.2312	1.1222	**1.2013**	1.0537
10	1.2512	**1.3575**	1.2659	1.1904	1.2452	**1.1375**	1.2798	1.1326	**1.227**	**1.1166**	1.2105	1.0535

І після цього найкраща модель була протестована за трьома параметрами для кожної пари модель/датасет. Результати представлені в Таблиці 7. Для кожної метрики чорним виділена модель, яка дає кращий результат для цього датасету.

Таблиця 7 - Фінальний тест кращих моделей

	Final Test											
Epoch	DeepCoNN				NeuMF				NARRE			
Датасет	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products
MSE	1.4475	1.1856	1.3621	0.7313	1.3789	1.1019	1.3364	**0.7129**	**1.3599**	**1.1013**	**1.3134**	0.7398
NDCG	1.2393	1.3913	1.2697	1.2025	**1.2218**	**1.1584**	1.2171	1.1294	1.2323	1.1847	**1.1987**	**1.1146**
HR@1	31.34	39.09	36.36	44.44	33.33	**44.17**	**70.0**	27.27	**37.62**	31.97	27.27	**50.0**

Як можна помітити, після довшого навчання кращі результати в середньому починає показувати вже NARRE. При чому як і в минулому випадку найкращі результати загалом дає найменший датасет, а найбільш низькі – найбільший датасет. Крім того, бачимо, що моделі починають перенавчатися в середньому після 5 епохи, тому у подальших дослідженнях будемо переривати навчання на цьому етапі. DeepCoNN має повільно спадаючу тенденцію по метриці MSE протягом всього навчання.

Графічно порівняльну залежність від епох для кожної моделі по датасетам можна побачити на Рис.9-16:

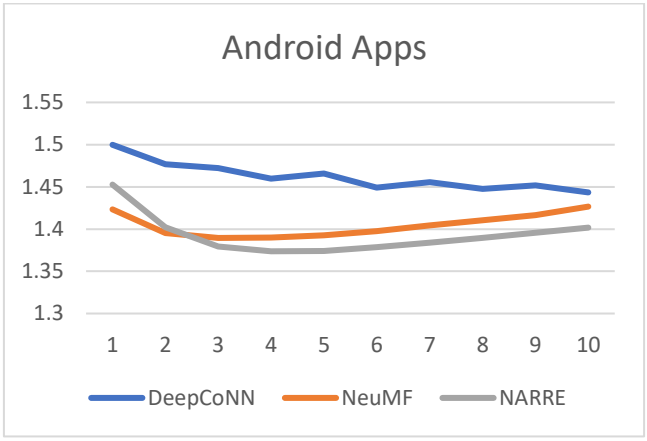

Рисунок 9 – Залежність MSE від кількості епох для датасету Android Apps

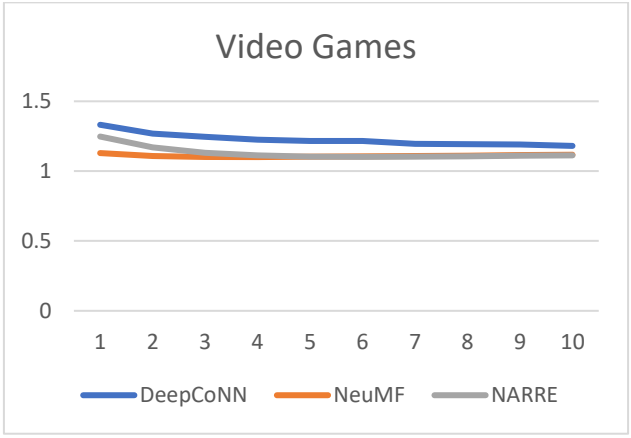

Рисунок 10 – Залежність MSE від кількості епох для датасету Video Games

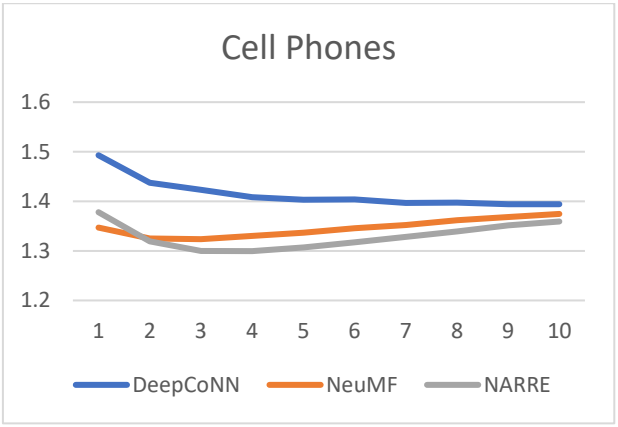

Рисунок 11 – Залежність MSE від кількості епох для датасету Cell Phones

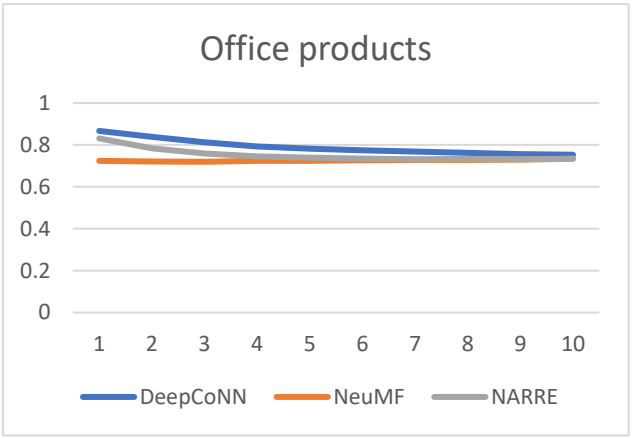

Рисунок 12– Залежність MSE від кількості епох для датасету Office Products

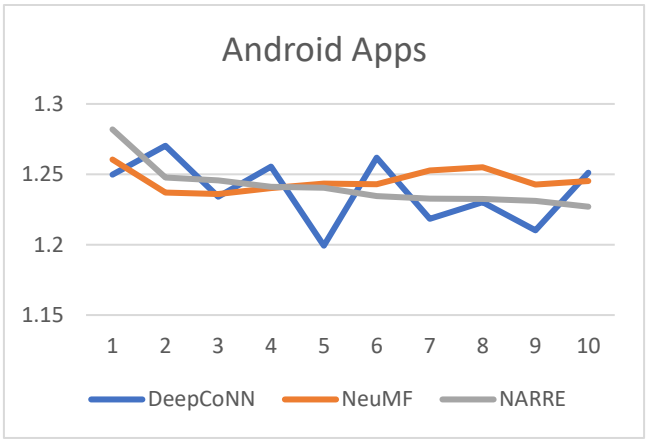

Рисунок 13 – Залежність NDCG від кількості епох для датасету Android Apps

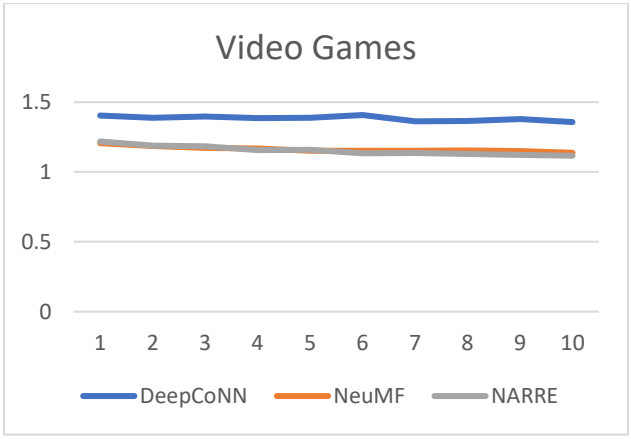

Рисунок 14 – Залежність NDCG від кількості епох для датасету Video Games

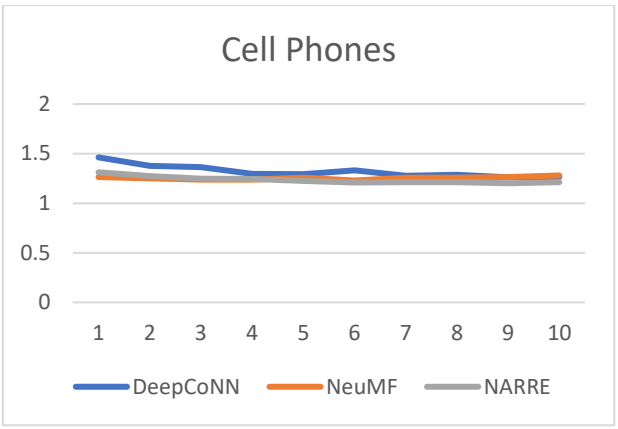

Рисунок 15 – Залежність NDCG від кількості епох для датасету Cell Phones

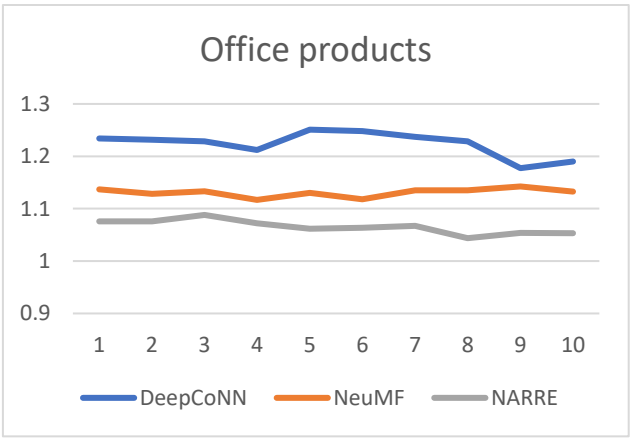

Рисунок 16 – Залежність NDCG від кількості епох для датасету Office Products

В наступному дослідженні перевіримо, як впливає Dropout на якість навчання. Для простоти, візьмемо все ті ж початкові параметри, але навчання обмежимо 5 епохами і занесемо в таблиці лише фінальний результат.

Результат дослідження можна побачити в Таблицях 8-10. Чорним відмічене краще значення Dropout для пари модель/датасет.

Таблиця 8 - Дослідження впливу Dropout, метрика MSE

	Моделі MSE											
Dropout	DeepCoNN				NeuMF				NARRE			
	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products
0.2	**1.4576**	**1.1729**	**1.3706**	**0.7444**	1.3983	1.1344	1.355	0.7361	1.3675	1.0971	1.3412	**0.7425**
0.4	1.4729	1.1798	1.3733	0.7607	1.3972	1.1229	1.3448	0.7202	**1.3597**	**1.0965**	1.3231	0.7475
0.6	1.472	1.2213	1.373	0.7679	1.3788	1.1035	**1.3368**	0.7149	1.361	1.1001	**1.3149**	0.7522
0.8	1.4803	1.2477	1.4007	0.7879	**1.3753**	**1.1026**	1.337	**0.7124**	1.3632	1.1086	1.3163	0.7531

Таблиця 9 - Дослідження впливу Dropout, метрика NDCG

	Моделі NDCG											
Dropout	DeepCoNN				NeuMF				NARRE			
	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products
0.2	1.1463	**1.2584**	**1.2138**	**1.2087**	1.2679	1.2245	1.2741	1.1805	1.2135	1.2108	1.2529	1.0982
0.4	**1.1444**	1.3109	1.2931	1.2614	1.226	1.1777	1.2593	1.1511	**1.2**	**1.1914**	1.2121	**1.0959**
0.6	1.1675	1.4013	1.2801	1.2971	**1.2162**	**1.1476**	**1.2242**	**1.1185**	1.256	1.2032	**1.2056**	1.099
0.8	1.2797	1.3667	1.3806	1.2332	1.2423	1.1757	1.2379	1.1538	1.256	1.2215	1.2217	1.1246

Таблиця 10 - Дослідження впливу Dropout, метрика HR@1

	Моделі HR@1											
Dropout	DeepCoNN				NeuMF				NARRE			
	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products	Android Apps	Video Games	Cell Phones	Office products
0.2	36.82	38.18	36.36	45.45	27.45	35.83	50.0	**45.45**	37.62	36.07	27.27	37.5
0.4	35.32	40.91	**54.55**	33.33	29.41	41.67	60.0	36.36	**39.52**	**36.89**	27.27	37.5
0.6	**38.31**	**41.82**	45.45	**54.55**	32.84	43.33	70.0	**45.45**	38.57	**36.89**	27.27	**50.0**
0.8	31.84	39.09	45.45	22.22	**33.82**	**45.83**	**80.0**	36.36	36.67	33.61	**36.36**	43.75

Як можна побачити, моделі NeuFM показують себе краще за умови використання більшого значення Dropout, DeepCoNN навпаки, за умови меншого, а NARRE не має чіткої залежності. Деякі сторонні дослідження

доводять, що чим більше розріджений датасет, тим менший розмір Dropout-у повинен бути, але на цих даних немає чіткої залежності.

Графічно порівняльну залежність від Dropout для кожної моделі по датасетам можна побачити нижче:

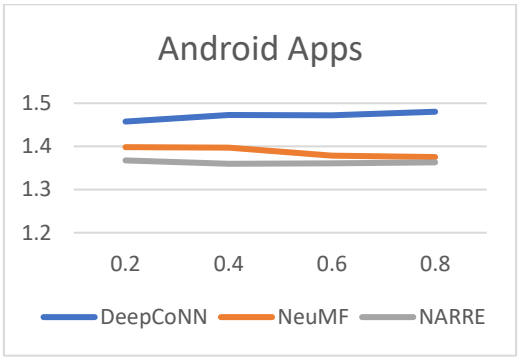

Рисунок 17 – Залежність MSE від Dropout для датасету Android Apps

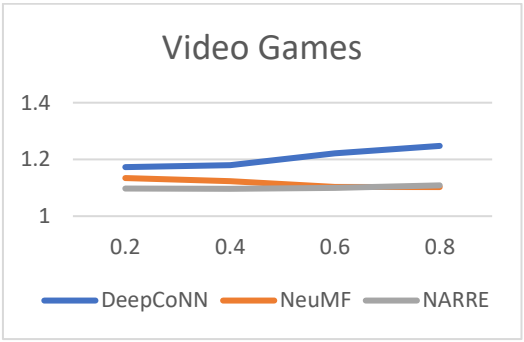

Рисунок 18 – Залежність MSE від Dropout для датасету Video Games

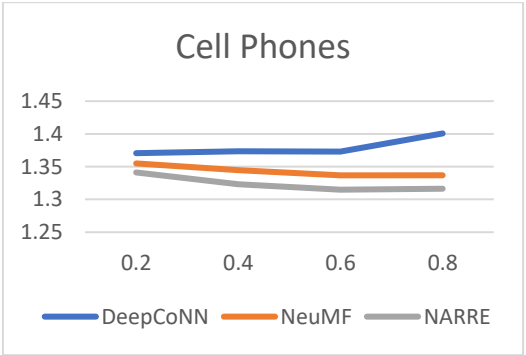

Рисунок 19 – Залежність MSE від Dropout для датасету Cell Phones

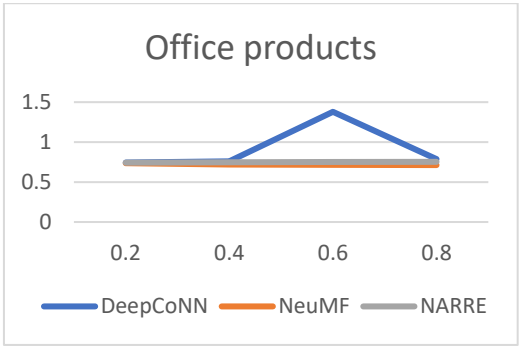

Рисунок 20 – Залежність MSE від Dropout для датасету Office Products

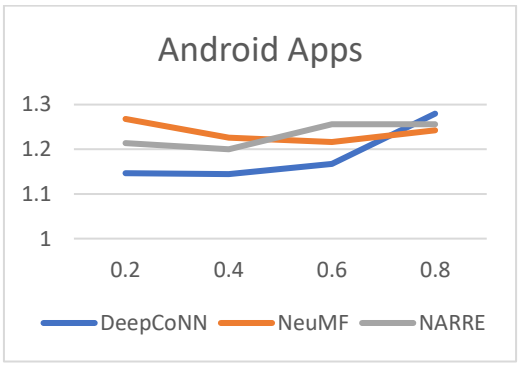

Рисунок 21 – Залежність NDCG від Dropout для датасету Android Apps

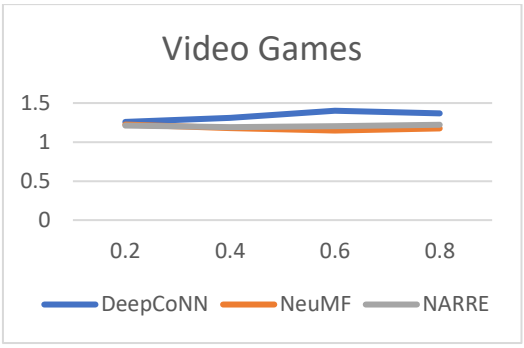

Рисунок 22 – Залежність NDCG від Dropout для датасету Video Games

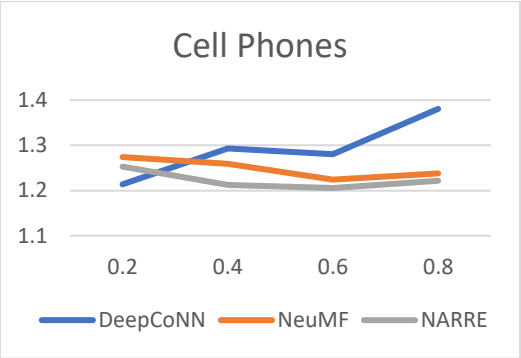

Рисунок 23 – Залежність NDCG від Dropout для датасету Cell Phones

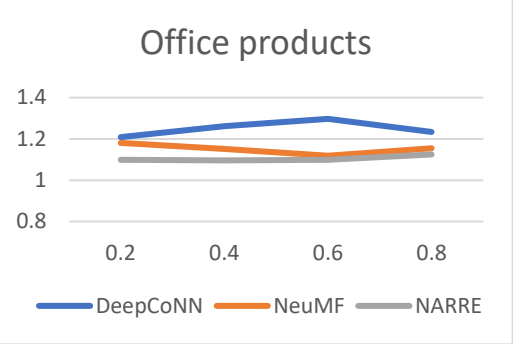

Рисунок 24 – Залежність NDCG від Dropout для датасету Office Products

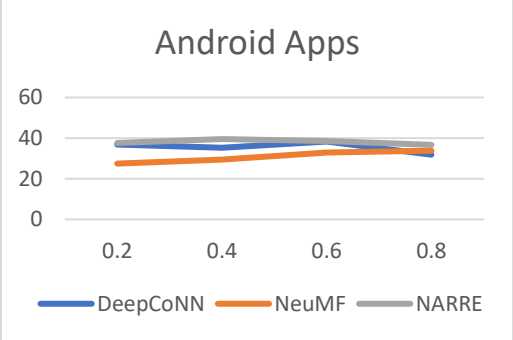

Рисунок 25 – Залежність HR@1 від Dropout для датасету Android Apps

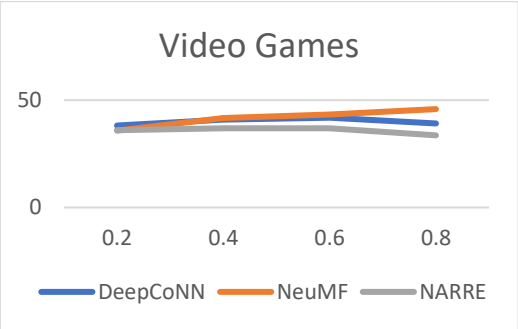

Рисунок 26 – Залежність HR@1 від Dropout для датасету Video Games

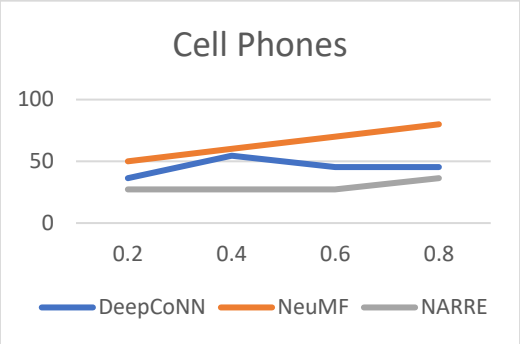

Рисунок 27 – Залежність HR@1 від Dropout для датасету Cell Phones

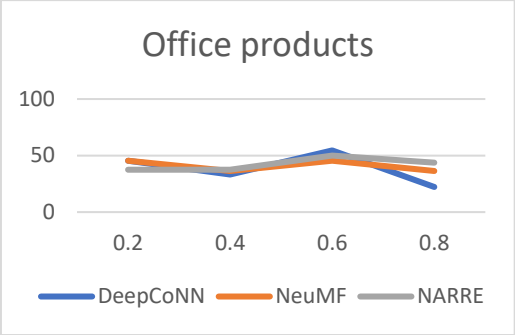

Рисунок 28 – Залежність HR@1 від Dropout для датасету Office Products

Таким чином, можна зробити проміжний висновок, що найбільш ефективним серед обраних алгоритмів є NARRE, хоча звичайний NeuFM у більшості випадків майже не відстає. DeepCoNN є найменш ефективним методом.

3.3. Дослідження ефективності запропонованого методу

В цьому пункті пропонуємо покращення класичної моделі DeepCoNN, яку назвали DeepAtnCoNN

Для дослідження точності проведено такі ж тести, як і для існуючих моделей. Результати наведено в Таблицях 11-14.

Таблиця 11 - Початкове дослідження

	Метрики		
Датасети	MSE	NDCG	HR@1
Android Apps	1.3966	1.2973	37.81
Video Games	1.1765	1.2267	43.64
Cell Phones	1.3489	1.2649	45.45
Office products	0.7692	1.2495	33.33

Бачимо, що вже на цьому етапі присутнє невелике покращення відносно попередніх моделей (крім NeuMF). Далі збільшуємо кількість епох до 10, як і в минулому дослідженні. Результати наведені в Таблиці 12.

Таблиця 12 - Дослідження впливу кількості епох на точність рекомендацій

Epoch	Android Apps		Video Games		Cell Phones		Office products	
	MSE	NDCG	MSE	NDCG	MSE	NDCG	MSE	NDCG
1	1.4778	1.331	1.2688	1.2491	1.4218	1.3092	0.8224	1.182
2	1.3997	1.3045	1.1816	1.2265	1.3745	1.2899	0.7845	1.2049
3	1.3717	1.2853	1.137	1.2089	1.3507	1.2728	0.7611	1.2085
4	**1.3635**	1.2702	1.1146	1.1954	1.3407	1.2593	0.7465	1.2002
5	1.3638	1.2585	1.1041	1.185	**1.3395**	1.2461	0.7379	1.198
6	1.3678	1.2492	**1.1008**	1.1691	1.3437	1.2352	0.7332	1.1949
7	1.3732	1.2419	1.1012	1.1609	1.3514	1.2216	0.731	1.1881
8	1.379	1.2363	1.1039	1.1541	1.3609	1.2179	**0.7307**	1.1855
9	1.3847	1.232	1.1077	1.1476	1.3715	1.2058	0.7315	1.1823
10	1.39	**1.2286**	1.1123	**1.1415**	1.3823	**1.2092**	0.7331	**1.1788**

Як бачимо, тенденція схожа: максимальна точність MSE досягається близько 5-6 епохи і надалі тільки знижується. В 3.4 результати будуть детальніше розглянуті в порівнянні з минулими методами.

Таблиця 13 - Фінальне тестування найкращих моделей

	Final Test			
	Android Apps	Video Games	Cell Phones	Office products
MSE	1.3708	1.1026	1.3166	0.7042
NDCG	1.2574	1.2053	1.2474	1.1752
HR@1	37.31	40.91	45.45	33.33

Графічно порівняльну залежність від епох для моделі по датасетам можна побачити нижче:

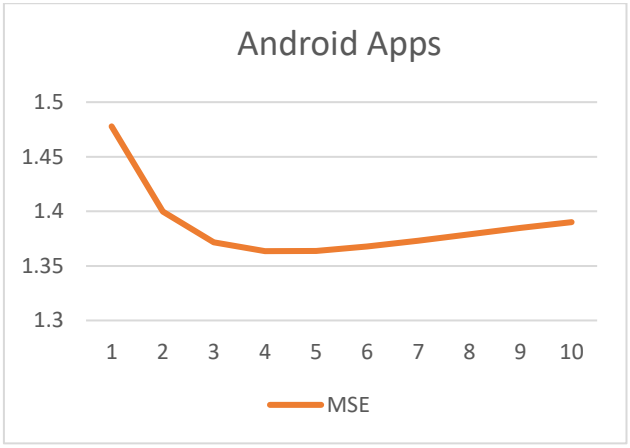

Рисунок 29 – Залежність MSE від кількості епох для датасету Android Apps

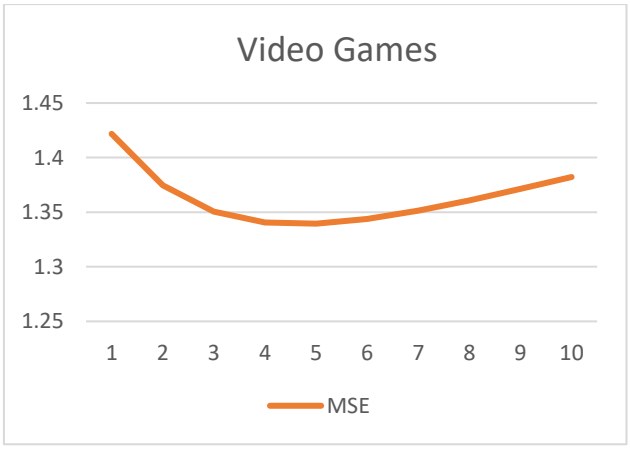

Рисунок 30 – Залежність MSE від кількості епох для датасету Video Games

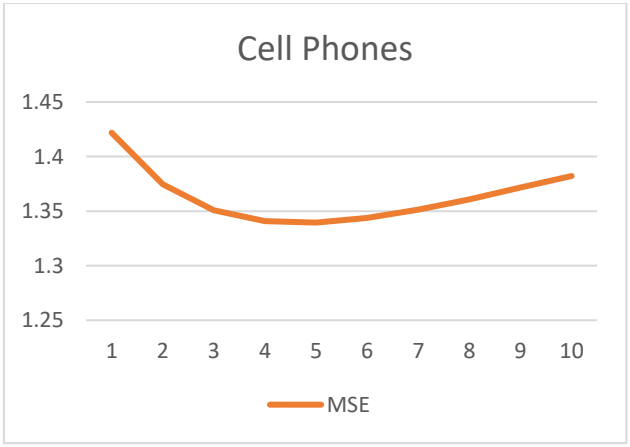

Рисунок 31 – Залежність MSE від кількості епох для датасету Cell Phones

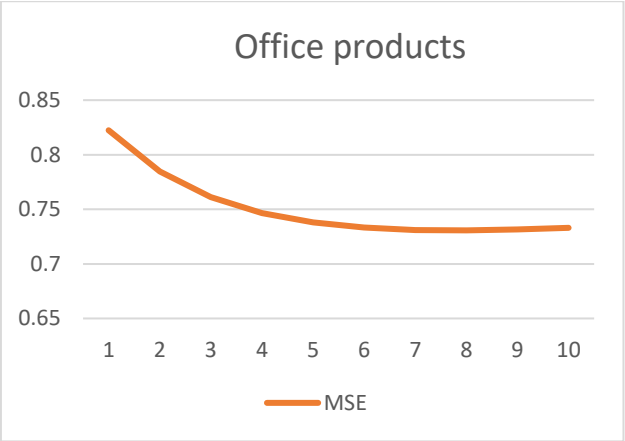

Рисунок 32 – Залежність MSE від кількості епох для датасету Office Products

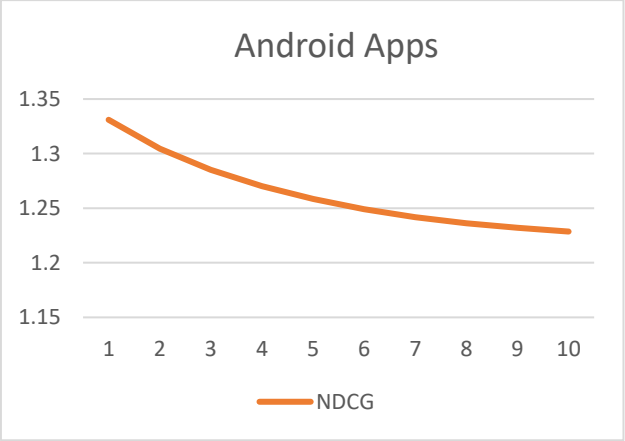

Рисунок 33 – Залежність NDCG від кількості епох для датасету Android Apps

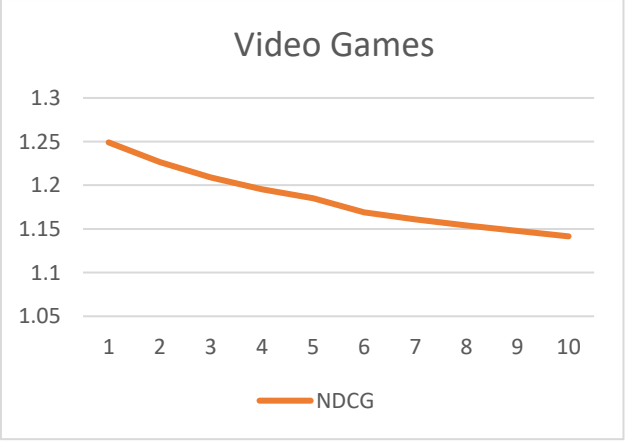

Рисунок 34 – Залежність NDCG від кількості епох для датасету Video Games

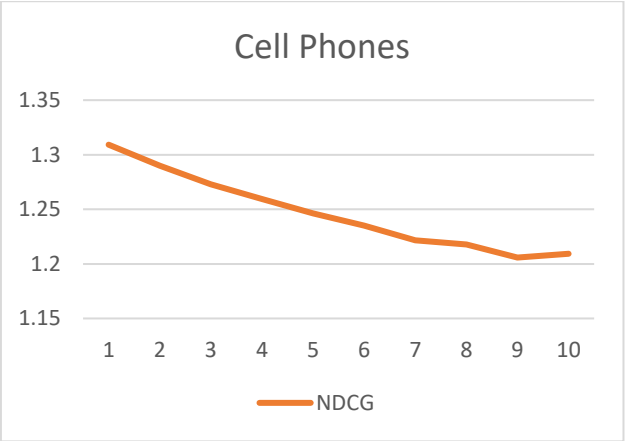

Рисунок 35 – Залежність NDCG від кількості епох для датасету Cell Phones

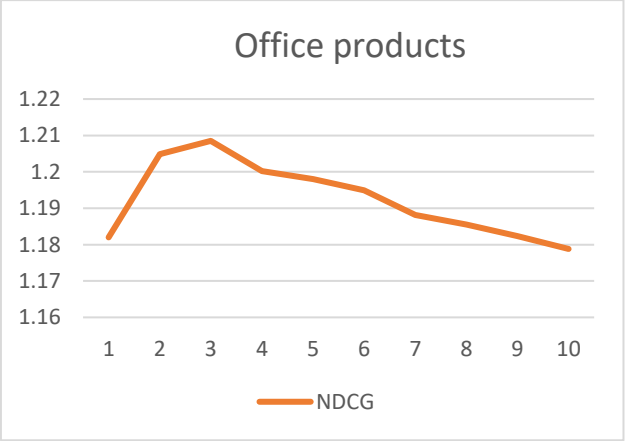

Рисунок 36 – Залежність NDCG від кількості епох для датасету Office Products

Аналогічно минулим моделям досліджуємо вплив Dropout при кількості епох=5. Результати можна побачити в таблиці 14.

Таблиця 14 - Дослідження впливу Dropout на точність рекомендацій.

Dropout	Android Apps			Video Games			Cell Phones			Office products		
Metrics	MSE	NDCG	HR@1	MSE	NDCG	HR@1	MSE	NDCG	HR@1	MSE	NDCG	HR@1
0.2	**1.3638**	1.2664	35.32	1.1058	1.2852	**42.73**	1.3169	1.2327	**45.45**	**0.7041**	1.1955	**33.33**
0.4	1.3671	1.2799	36.82	**1.0995**	1.2509	41.82	**1.3168**	1.2325	**45.45**	0.7153	1.2009	**33.33**
0.6	1.3677	**1.2484**	**37.31**	1.1048	**1.2109**	**42.73**	1.3169	**1.2311**	**45.45**	0.7147	**1.1914**	**33.33**
0.8	1.3708	1.257	**37.31**	1.1048	1.2143	41.82	1.3169	1.2342	**45.45**	0.7152	1.1964	**33.33**

Як можна побачити, ситуація найбільш схожа з аналогічною у DeepCoNN. Варто зазначити, що хоча видно, що модель на всіх датасетах працює точніше при середніх значеннях Dropout, у підсумку це не має вирішального значення, адже різниця між показниками точності є незначною.

Графічно порівняльну залежність від Dropout для моделі по датасетам можна побачити нижче.

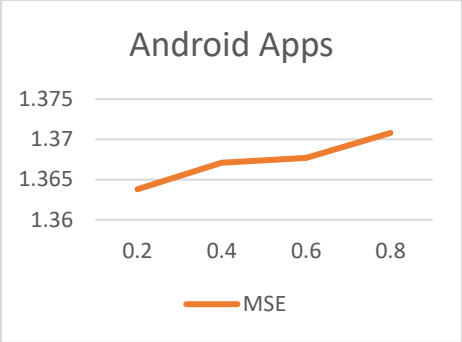

Рисунок 37 – Залежність MSE від Dropout для датасету Android Apps

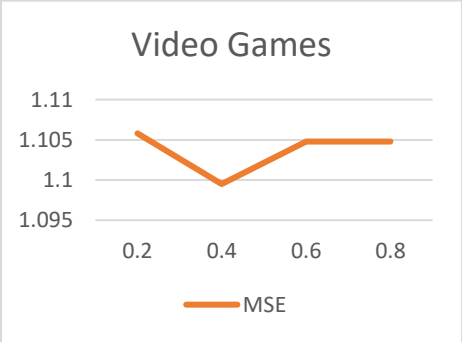

Рисунок 38– Залежність MSE від Dropout для датасету Video Games

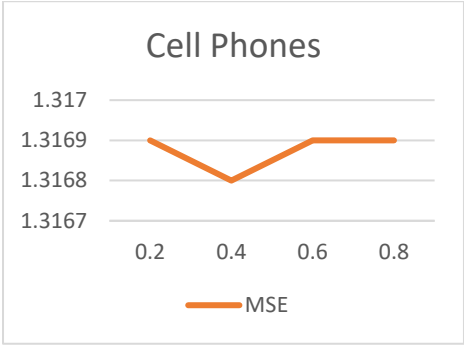

Рисунок 39 – Залежність MSE від Dropout для датасету Cell Phones

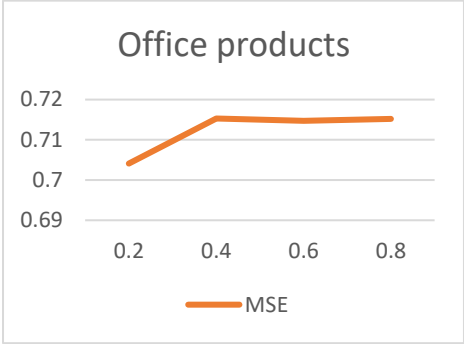

Рисунок 40 – Залежність MSE від Dropout для датасету Office Products

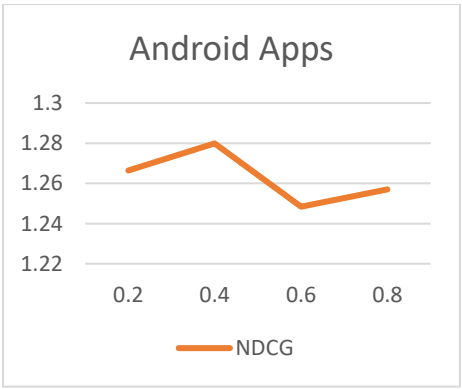

Рисунок 41 – Залежність NDCG від Dropout для датасету Android Apps

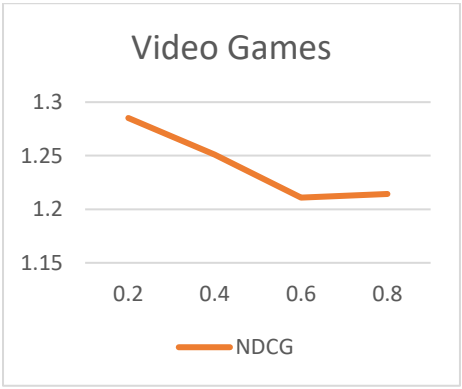

Рисунок 42– Залежність NDCG від Dropout для датасету Video Games

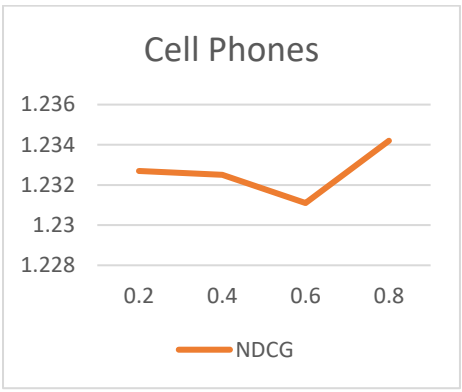

Рисунок 43 – Залежність NDCG від Dropout для датасету Cell Phones

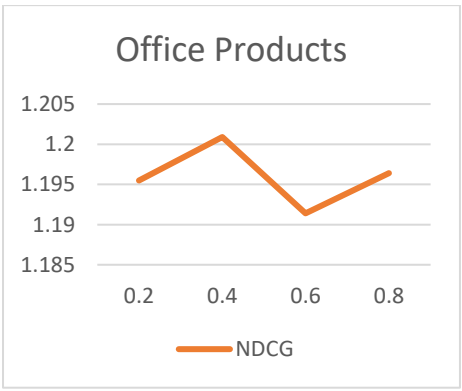

Рисунок 44 – Залежність NDCG від Dropout для датасету Office Products

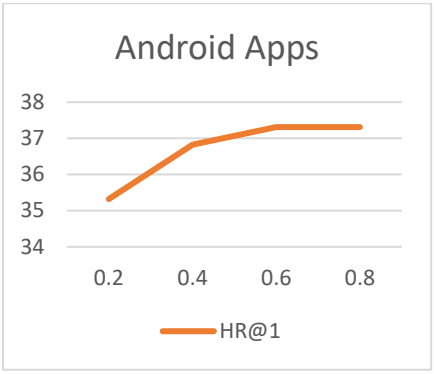

Рисунок 45 – Залежність HR@1 від Dropout для датасету Android Apps

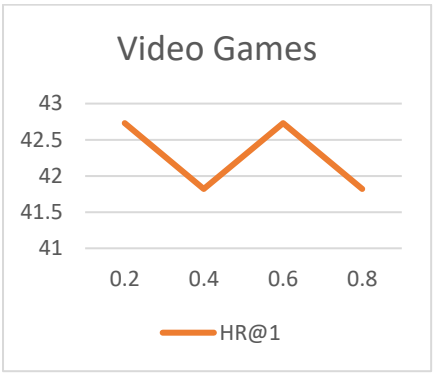

Рисунок 46– Залежність HR@1 від Dropout для датасету Video Games

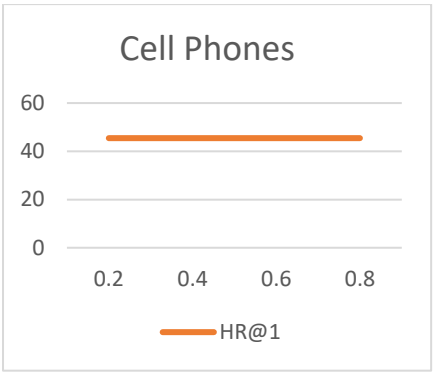

Рисунок 47 – Залежність HR@1 від Dropout для датасету Cell Phones

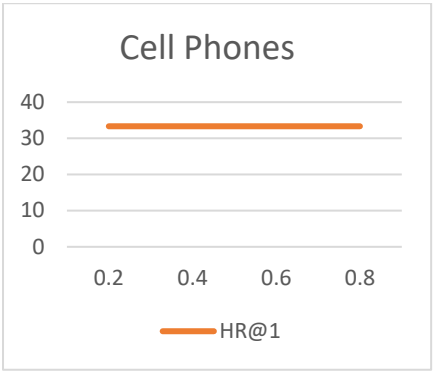

Рисунок 48 – Залежність HR@1 від Dropout для датасету Office Products

3.4. Аналіз отриманих результатів

Запропонована модель має кращу точність ніж три попередні на наборах даних Android Apps, Cells Phones і Video Games, а на останньому датасеті, Office Products, дає кращі результати ніж DeepCoNN і NARRE, однак ці результати майже не відрізняються від NeuMF, яка не використовує огляди для рекомендацій. Докладне порівняння точності протягом навчання для чотирьох моделей можна побачити на Рис. 49-56:

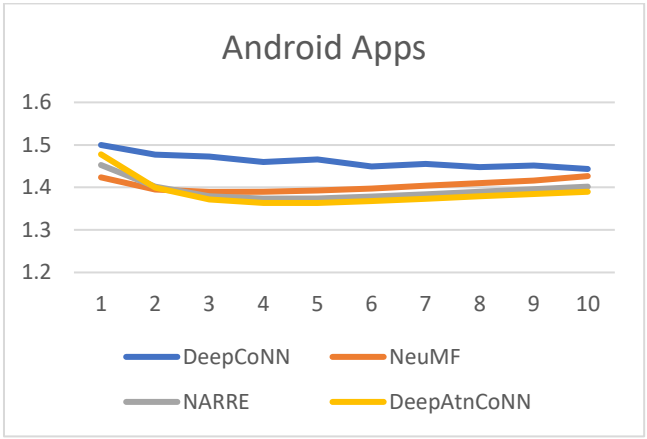

Рисунок 49 – Залежність MSE від кількості епох для датасету Android Apps

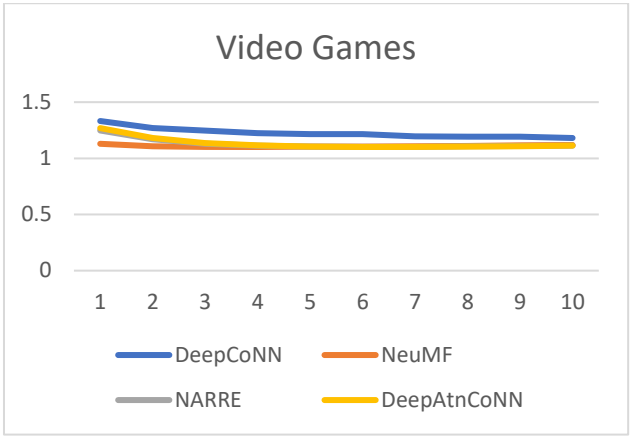

Рисунок 50 – Залежність MSE від кількості епох для датасету Video Games

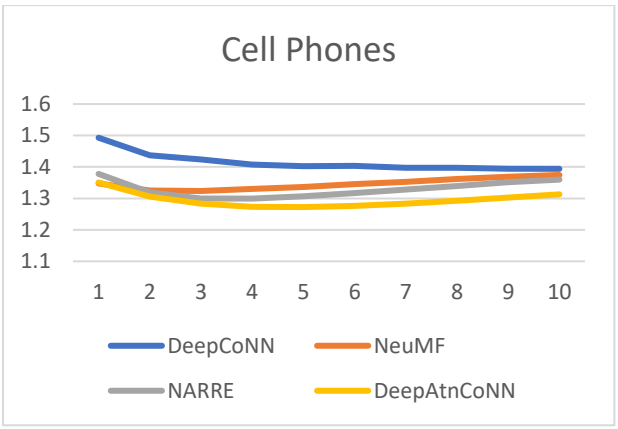

Рисунок 51 – Залежність MSE від кількості епох для датасету Cell Phones

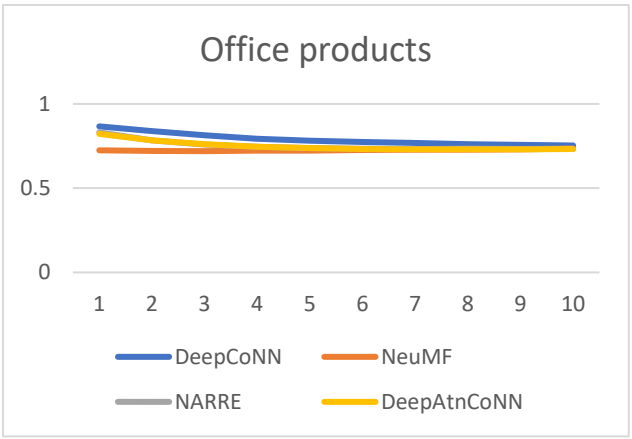

Рисунок 52 – Залежність MSE від кількості епох для датасету Office Products

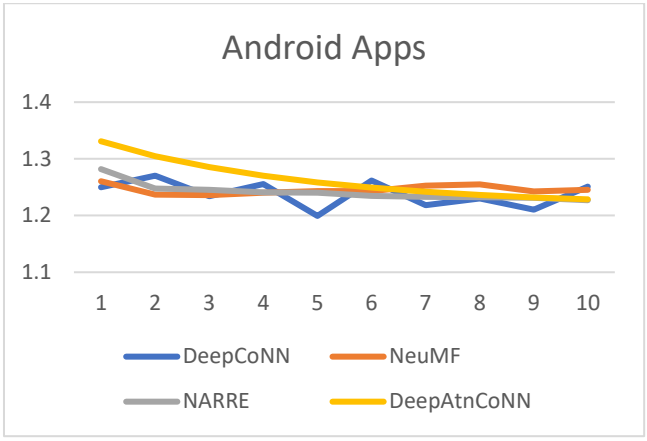

Рисунок 53 – Залежність NDCG від кількості епох для датасету Android Apps

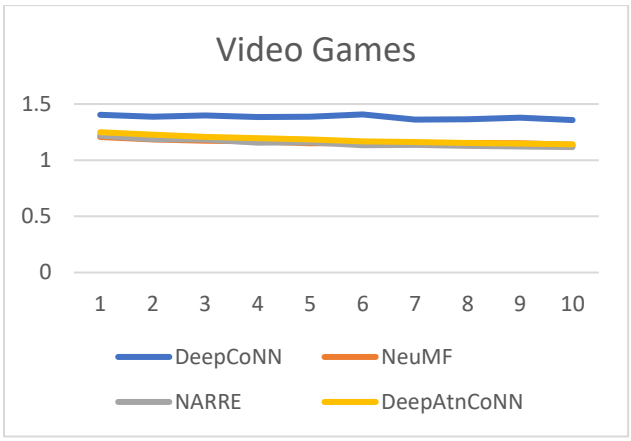

Рисунок 54 – Залежність NDCG від кількості епох для датасету Video Games

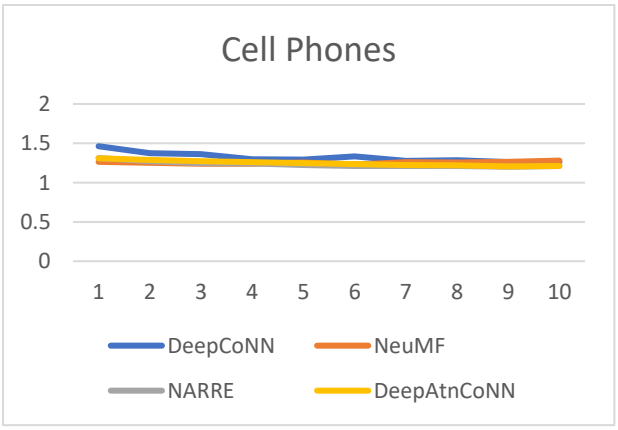

Рисунок 55 – Залежність NDCG від кількості епох для датасету Cell Phones

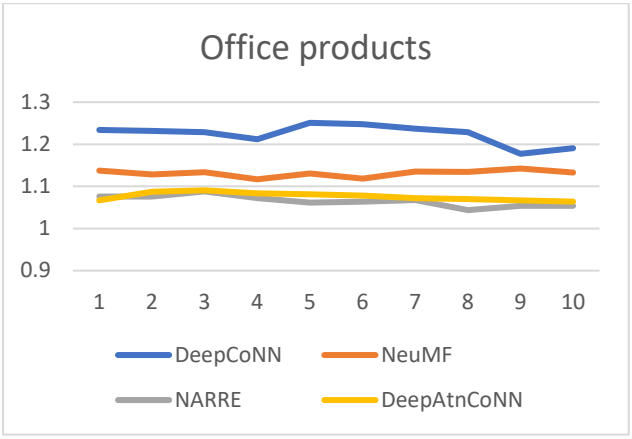

Рисунок 56 – Залежність NDCG від кількості епох для датасету Office Products

Таким чином, можна зробити висновок, що запропонована система буде працювати краще ніж інші за умови великого датасету. Для маленьких датасетів немає необхідності використовувати огляди, аналогічний показник точності дає звичайна NeuMF.

Порівнюючи показники Dropout, можна дійти висновку, що запропонований метод краще працює для Dropout 0.2, і в цьому випадку перевершує перші три моделі за точністю. Докладне порівняння точності для різних Dropout для чотирьох моделей можна побачити на Рис. 57-68:

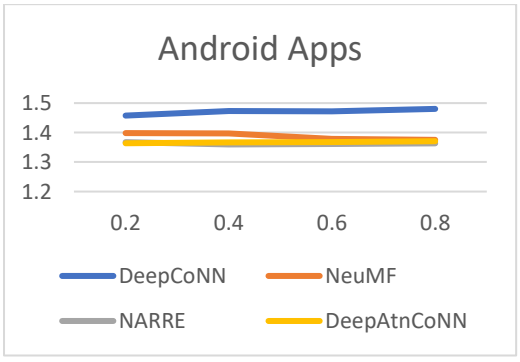

Рисунок 57 – Залежність MSE від Dropout для датасету Android Apps

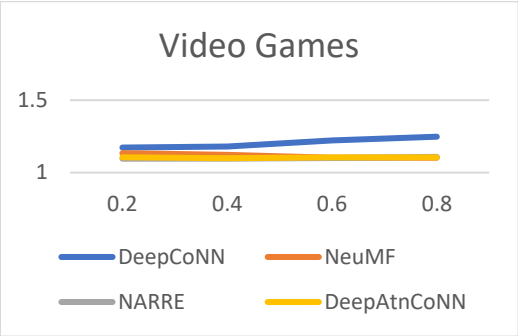

Рисунок 58 – Залежність MSE від Dropout для датасету Video Games

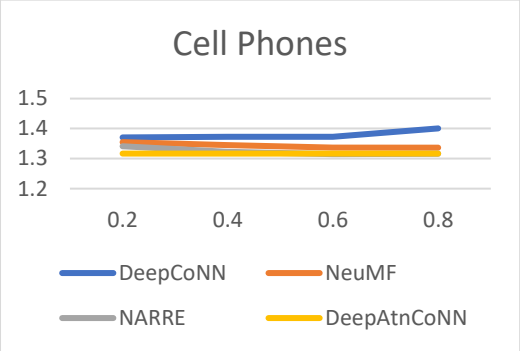

Рисунок 59 – Залежність MSE від Dropout для датасету Cell Phones

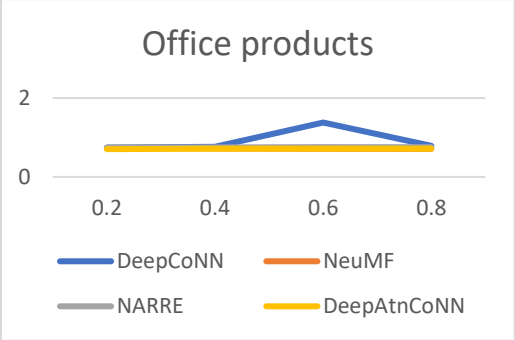

Рисунок 60 – Залежність MSE від Dropout для датасету Office Products

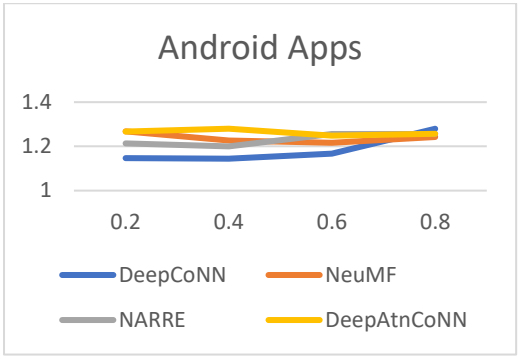

Рисунок 61 – Залежність NDCG від Dropout для датасету Android Apps

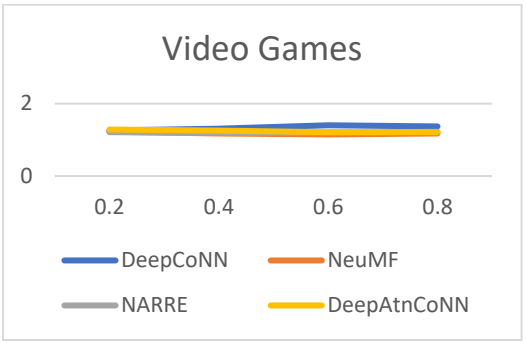

Рисунок 62 – Залежність NDCG від Dropout для датасету Video Games

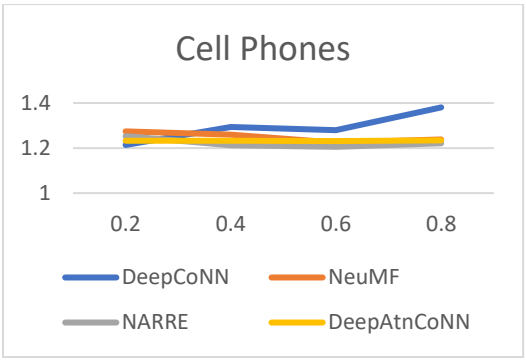

Рисунок 63 – Залежність NDCG від Dropout для датасету Cell Phones

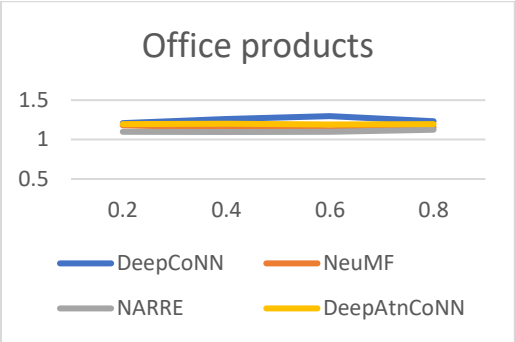

Рисунок 64 – Залежність NDCG від Dropout для датасету Office Products

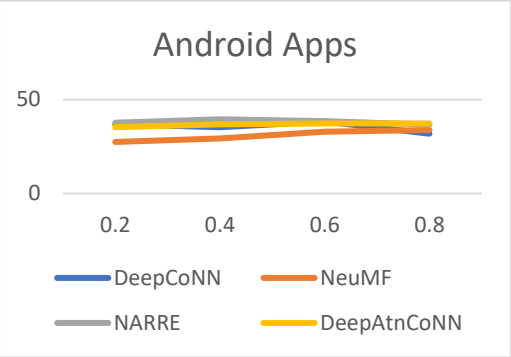

Рисунок 65 – Залежність HR@1 від Dropout для датасету Android Apps

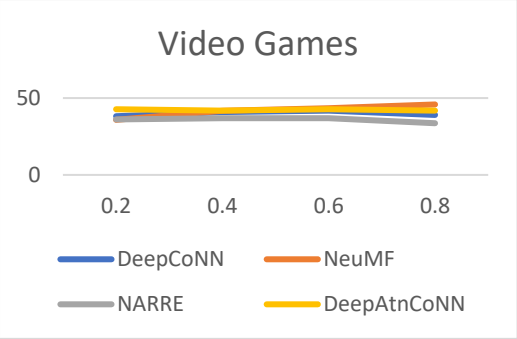

Рисунок 66 – Залежність HR@1 від Dropout для датасету Video Games

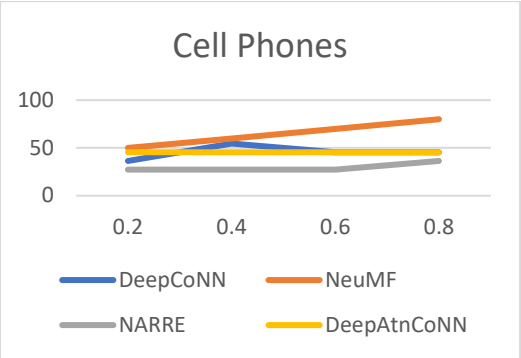

Рисунок 67 – Залежність HR@1 від Dropout для датасету Cell Phones

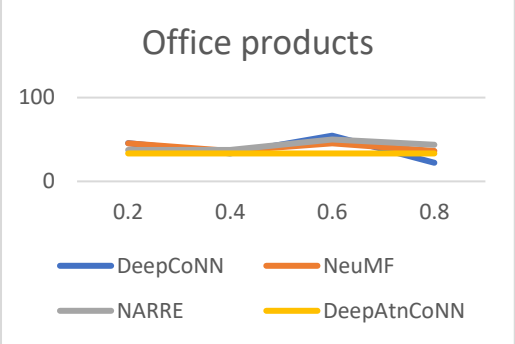

Рисунок 68 – Залежність HR@1 від Dropout для датасету Office Products

Для покращення точності використаних алгоритмів у майбутніх дослідженнях можна використати наступні ідеї:

1. Використання автокодерів замість стандартних TextCNN, запропонованих NARRE
2. Використання більш щільних датасетів для навчання
3. Додавання двонаправлених шарів GRU або LSTM з увагою
4. Додавання до стандартних алгоритмів, що не використовують тексти огляду, аспекти оглядів в якості регуляризації
5. Використання NLP для попередньої обробки тексту.

За умови використання однієї або декількох ідей зі списку може бути досягнута ще більша точність методів, що використовують тексти оглядів для рекомендацій.

3.5. Висновки до розділу 3

Було використано 64-вимірні вбудовування word2vec, навчені за допомогою Gensim3 для попередньої обробки чотирьох датасетів різного розміру Amazon product data з оглядами: Apps for Android, Video Games, Cell Phones and Accessories, Office Products.

Були проведені дослідження точності для класичних алгоритмів DeepCoNN, NARRE і NeuMF, а також запропонованого алгоритму DeepAtnCoNN, який додає додатковий шар уваги і змінює визначення ваг з Машини Факторизації на зміщення глобальні, користувачів і елементів.

Було проведено два дослідження: в рамках першого моделі навчалися протягом 10 епох на чотирьох датасетах. За його результатом було встановлено, що моделі досягають найкращої точності приблизно до 5 епохи.

Друге дослідження було про вплив Dropout на якість навчання. За його результатами було визначено, що в цілому значення dropout не має вагомого впливу на точність, але DeepAtnCoNN краще справляється на низьких показниках Dropout, ніж інші.

Було виявлено, що запропонована модель має кращу точність, ніж попередні три на наборах даних Android Apps, Cells Phones і Video Games, а на останньому датасеті, Office Products, дає кращі результати ніж DeepCoNN і NARRE, але такі ж, як і NeuMF, яка не використовує огляди для рекомендацій.

Було запропоновано декілька ідей для покращення алгоритмів для майбутніх досліджень.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї стартап-проекту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

- зміст ідеї;
- можливі напрямки застосування;
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані у вигляді таблиці (таблиця 15) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 15 - Опис ідеї стартап-проекту «Recommender Bot»

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система рекомендацій у вигляді чат-боту Telegram для надання персоналізованих рекомендацій щодо товарів інтернет-магазинів, базуючись на минулих діях, наданих відгуках і світових трендах.	Опрацювання інформації про користувача	Формування бази інформації про покупців і їх вподобання для подальшого використання для маркетингових цілей
	Надання рекомендацій	Збільшення прибутку компанії за рахунок покращення рекомендацій.
	Створення бази товарів	Збереження великої кількості контенту в одному місці
	Створення бази рейтингів і відгуків	Отримання оновленої актуальної бази взаємодій користувача для подальших досліджень.

Далі проведено аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів. Результати наведено в Таблиці 16.

Таблиця 16 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№п/п	Техніко-економічні характеристики ідеї	Продукція конкурентів				W	N	S
		Дипломний проект	Amazon	ASOS	Naked Wines			
1	Універсальний рекомендаційний сервіс	Так	Так	Ні	Ні			+
2	Зручність використання	Висока	Висока	Середня	Середня		+	
3	Складність реалізації	Висока	Висока	Висока	Висока		+	
4	Точність рекомендацій	Кращий результат за класичні	Власний алгоритм	Власний алгоритм	Власний алгоритм			+
5	Аналіз відгуків користувача	Присутні	Відсутні	Відсутні	Відсутні			+

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення

товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз складових, що наведений в Таблиці 17.

Таблиця 17 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№п/п	Ідея проекту	Технології і реалізація	Наявність технологій	Доступність технологій
1	Recommender Bot	Pytorch	Наявна	Доступна
2		Telegram API	Наявна	Доступна
3		TensorFlow	Наявна	Доступна
4		C#	Потребує доопрацювання	Доступна
Обрана технологія реалізації ідеї проекту: Pytorch за допомогою Jupiter Notebook, графічна візуалізація за допомогою Telegram API				

За результатам можна зробити висновок, що технологічна реалізація проекту можлива.

4.3 Аналіз ринкових можливостей запуску проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. Результат наведено в Таблиці 18.

Таблиця 18 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	>10
2	Загальний обсяг продаж, грн/ум.од	10000
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	20%

За результатами аналізу Таблиці 16 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та сформовано орієнтовний перелік вимог до товару для кожної групи. Результат наведено в Таблиці 19.

Таблиця 19 - Характеристика потенційних клієнтів стартап-проекту

№п/п	Потреба що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в пошуку релевантного контенту і наданні чітких рекомендацій	Пересічний користувач інтернет-сервісів	Кінцевий користувач, який буде використовувати стартап для отримання рекомендацій	Контент має задовольняти загальним або сьогохвилинним потребам Контент має бути новим і актуальним Контент має мати пояснення

Продовження табл. 19

№п/п	Потреба що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
2	Потреба поділитися власною думкою і дізнатися думки інших	Пересічний користувач інтернет-сервісів	Кінцевий користувач, який буде використовувати стартап для вибору товара на основі відгуків інших користувачів	Процес запису відгуку має бути простим, швидким, інтуїтивно зрозумілим і ненав'язливим. Відгуки інших людей мають бути корисними і швидко провантажуватися Відгуки мають проходити модерацію, але не занадто жорстку

№п/п	Потреба що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
3	Потреба залучення нових клієнтів для свого товару	Власники інтернет-магазинів різного розміру	Кінцевий користувач, який буде використовувати стартап як засіб реклами, з різними налаштуваннями: різні критерії клієнта, різний об’єм інформації.	Рекомендації мають надаватися правильним користувачам, а також підпадати під критерії компанії-замовника.
4	Зручний додаток з усіма рекомендаціями під рукою	Пересічний користувач інтернет-сервісів	Кінцевий користувач, який буде використовувати стартап як корисну і зручну рекомендаційну систему, що завжди під рукою.	Додаток має бути безкоштовним. Додаток має бути зручним в управлінні і інтуїтивно зрозумілим. Додаток має використовувати небагато пам’яті, інтернету і не навантажувати пристрій.

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають. Результат наведено в Таблицях 20, 21.

Таблиця 20 - Фактори загроз

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок гіганта з сумісних областей, здатного швидко зайняти нішу за рахунок більшої кількості ресурсів	Передбачити додаткові переваги стартапу. Вдосконалення власного продукту. Вибір нової цільової аудиторії.
2	Економічний	Подорожчання вартості та обслуговування обладнання, необхідного для роботи системи	Сформувати ціннісну пропозицію, щоб пропонувати максимально можливу пропорцію ціна/якість. Оптимізувати програмне забезпечення для економії ресурсів.
3	Зміна потреб користувача	Потреба користувачів у додатковому функціоналі	Постійне удосконалення у відповідь на потребу цільової аудиторії
4	Заборона збору персональних даних	Зміна в законодавстві, що стосується збору, зберігання та обробки персональних даних користувачів	Контроль за персональними даними, чітке реагування на зміни законодавства

№п/п	Фактор	Зміст загрози	Можлива реакція компанії
5	Недостатня популярність	Відсутність уваги з боку користувачів	Якісна рекламна кампанія з залученням спеціалістів з маркетингу для популяризації проекту
6	Якість рекомендацій на реальних даних	Рекомендації можуть бути неточними в залежності від якості наданих компаніями даних	Оновлення даних, перенавчання системи, додавання нових джерел рекомендацій
7	Вузький функціонал	Недостатня кількість функцій для повноцінного незалежного сервісу	Інтеграція з іншим сервісом, розширення функціоналу

Таблиця 21. Фактори можливостей

№п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Нові джерела інформації	Рекомендації на основі оглядів – відносно нове джерело інформації про вподобання користувача	Використання більш глибоких методів аналізу тексту
2	Простота використання	Інтуїтивний інтерфейс, що не потребує додатково навчання для використання	Додаткове спрощення інтерфейсу додатка, яскраві кнопки і додаткові рекомендації про використання
3	Грошова винагорода за рекламу	Додавання окремих комерційних рекомендацій	Після набору певного рівня популярності є можливість залучати клієнтів-власників магазинів до платного пріоритетного показу продукту для релевантних користувачів.
4	Інтеграція в системи великих компаній	Інтеграція методу в існуючі великі рекомендаційні системи монополістичних конкурентів.	Можливість випробувати ефективність і знайти недоліки методу рекомендацій.

Надалі було проведено аналіз пропозиції, а саме визначено загальні риси конкуренції на ринку. (Таблиця 22)

Таблиця 22. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив дії на діяльність підприємства
1. Тип конкуренції: чиста	Конкуренти існують, але ні один із них не є повністю аналогічним або значним.	Необхідність створення власної цінності та трансляції цінності і унікальності потенційним клієнтам.
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти в інших країнах	Побудова проекту таким чином, щоб він був робочим на національному ринку, але потенційно міг бути розшарений і на міжнародний.
3. За галузевою ознакою - міжгалузева	Продукт може бути використаний для різних галузей.	Постійне вдосконалення продукту, прагнення до його універсальності.
4. Конкуренція за видами товарів - товарно-видова	Конкуренція між сигнатурними особливостями різних аналогів стартапу.	При створенні орієнтуватись на недоліки і недоробки конкурентів.
5. За характером конкурентних переваг - цінова	Велика різниця між потенційними цінами на аналогічні продукти	Забезпечення конкурентної ціни на продукт
6. За інтенсивністю – марочна	Представлені компанії з впізнаваємим брендом	Створення і підтримка особистого бренду

Далі було проведено аналіз конкуренції у галузі за моделлю М. Портера (таблиця 23).

Таблиця 23. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Amazon, ASOS, Naked Wines.	Відсутні	Доступ надається за рахунок мережі Інтернет	Найбільша рушійна сила проекту	Системи Конкуруючих компаній
Висновки	Найбільшим конкурентом є перший, адже тримає значну частину ринку	Можливості входу існують, адже оптимальної стратегії надання рекомендації до цього часу ще не було створено.	Постачальники відсутні	Важливим для користувача є точність і зручність процесу надання рекомендацій	Можуть мати швидший і дешевший, але менш точний алгоритм для зменшення собівартості товару.

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок був врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу

конкуренції, проведеного в таблиці, а також із урахуванням характеристик з попередніх таблиць визначається та обґрунтовується перелік факторів конкурентоспроможності.

Аналіз оформлено у Таблиці 24.

Таблиця 24. Обґрунтування факторів конкурентоспроможності

№п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Більш низькі ціни	Так як проект лише розвивається, існує можливість надавати більш низькі ціни для перших впроваджень
2	Простота інтерфейсу	Інтуїтивно зрозумілий інтерфейс з простим доступом до найважливіших функцій.
3	Універсальність	Використання декількох, в тому числі новітніх, джерел для рекомендацій

За визначеними факторами конкурентоспроможності проведено аналіз сильних та слабких сторін стартап-проекту:

Таблиця 25. Порівняльний аналіз сильних та слабких сторін

№п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			-3	-2	-1	0	+1	+2	+3
1	Більш низькі ціни	15			x				
2	Простота інтерфейсу	15			x				
3	Універсальність	20	x						

Фінальним етапом було проведення SWOT- аналізу:

Таблиця 26. SWOT- аналіз стартап-проекту

Сильні сторони: Проект має зручну реалізацію в рамках сервісу, що під рукою у користувачів з різними пристроями. Інтерфейс інтуїтивно зрозумілий, а рекомендації використовують різні джерела, що робить рекомендації універсальними.	Слабкі сторони: Складність реалізації маленькою командою розробників, складність формування датасетів для навчання і використання, можлива висока конкуренція.
Можливості: Можливість інтеграції в інші додатки, додавання нових функцій, надання платних рекомендацій.	Загрози: Вихід гіганта з суміжної області може стати сильним і більш привабливим аналогом продукту. Подорожчання вартості з часом може зробити систему менш привабливою для користувачі. У користувачів можуть виникнути інші потреби, які продукт не зможе оперативно задовольнити. Зміна законів про збір персональних даних може унеможливити збір якісних даних. Проект може не набрати необхідну популярність в короткий термін.

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки для виведення стартап-проекту на ринок. Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів:

Таблиця 27. Альтернативи ринкового впровадження стартап-проекту

№п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розміщення проекту на відкритому ресурсі	Ресурси – вільні донати, залежить від популярності проекту	Тиждень
2	Розповсюдження через конференції, хатакони	Середня – можливість залучення потенційних інвесторів	Від півроку
3	Агресивна рекламна кампанія	Висока, але великі вкладення	Від півроку

Обрано синтез 1 і 2: проект буде розміщено у вільному доступі для напрацювання аудиторії і перших тестів, а розширений функціонал буде пропонуватися потенційним інвесторам.

4.4 Розроблення ринкової стратегії стартап-проекту

Першочергово визначено стратегію охоплення ринку: було проведено опис цільових груп потенційних споживачів:

Таблиця 28. Вибір цільових груп потенційних споживачів

№п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Інтернет-користувачі	Висока	Високий	Сильна	Просто
2	Бізнеси	Середня	Середній	Сильна	Складно
3	Інші стартап-проекти	Середня	Низький	Середня	Середнє
Які цільові групи обрано: 1,2,3					

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку:

Таблиця 29. Визначення базової стратегії розвитку

№п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспро-можні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розміщення проекту на відкритому ресурсі	Диференційова-ного маркетингу	Масштабування та максимізація	Стратегія спеціаліза-ції
2	Розповсюдже-ння через конференції, хатакони	Диференційова-ного маркетингу	Посилення якості і розширення можливостей	Стратегія спеціаліза-ції

Наступним кроком обрано стратегію конкурентної поведінки:

Таблиця 30. Визначення базової стратегії конкурентної поведінки

№п/п	Чи є проект «першопрохі-дцем» 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
1	Ні	Частково забирати існуючих, частково шукати нових	Ні	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розроблено стратегію позиціонування:

Таблиця 31. Визначення стратегії позиціонування

№п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота інтерфейсу, нові джерела для рекомендацій, надання точних результатів	Диференціація	Використання відгуків як нове джерело для рекомендацій і їх пояснення. Простий інтерфейс дозволяє користуватися проектом без додаткового навчання.	Точність результатів, простий інтерфейс, нові рекомендації.

4.5 Розроблення маркетингової програми стартап-проекту

Сформовано маркетингову концепцію товару, який отримає споживач.

Таблиця 32. Визначення ключових переваг концепції потенційного товару

№п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Якісні рекомендації	Якісні рекомендації з високою точністю	Постійне покращення існуючих моделей для кращого надання рекомендаційних систем
2	Комплексний аналіз	Використання комплексного підходу до рекомендацій	Єдина система, що пропонує велику кількість можливостей в одному місці
3	Простий інтерфейс	Інтуїтивно просте використання	Система з інтуїтивно зрозумілим інтерфейсом, що не потребує ніяких знань перед використанням

Розроблено трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання.

Таблиця 33. Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Система рекомендацій у вигляді чат-боту Telegram для надання персоналізованих рекомендацій щодо товарів інтернет-магазинів, базуючись на минулих діях, наданих відгуках і світових трендах.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Якісні рекомендації
2. Комплексний аналіз
3. Простий інтерфейс</td><td>Не матеріальна</td><td>Технологічна</td></tr>
<tr><td colspan="3">Якість: внутрішнє тестування додатку, система обробки помилок та логування непередбачених ситуацій</td></tr>
<tr><td colspan="3">Пакування: продукт доступний в офіційному додатку Telegram</td></tr>
<tr><td colspan="3">Марка: назва організації-розробника + назва товару</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: програмний код</td></tr>
<tr><td colspan="3">Після продажу: додаткова підтримка спеціалістів налаштування, підтримка розробника</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: ліцензія, патент, захист інтелектуальної власності</td></tr>
</table>

Визначено цінові межі відповідно до цін товарів-конкурентів і рівня доходів цільових груп:

Таблиця 34. Визначення меж встановлення ціни

№п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Дані не розповсюджуються	Дані не розповсюджуються	>10000$/ місяць	150-300$

Також визначено оптимальну систему збуту:

Таблиця 35. Формування системи збуту

№п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Підписка на додаткові послуги	Продаж	Однорівневий	Власні сили та через посередників
2	Замовлення реклами	Продаж	Однорівневий	Власні сили та через посередників

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів:

Таблиця 36. Концепція маркетингових комунікацій

№п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Інтеграція API у клієнтській системі	Інтернет	Низька ціна, простота використання, універсальність	Показати переваги і власну ціннісну пропозицію	Порівняння продукту з іншими

4.6 Висновки до розділу 4

В даному розділі проведено попередній аналіз для впровадження розробленої системи в якості стартап проекту. Було сформовано основну ідею стартап-проекту (рекомендаційна система на основі оглядів для боту Telegram), визначено слабкі і сильні сторони порівняно з потенційними конкурентами, за якими було визначено, що стартап є релевантною ідеєю.

Було підібрано необхідні технології для впровадження проекту, за яким було обрано мову програмування Python та бібліотеку машинного навчання Pytorch.

Досліджено ринок і товари-конкуренти, визначено потреби і фактори загроз, що формують ринок. Було проведено аналіз конкуренції в галузі. Виділено фактори конкурентоспроможності і проведено SWOT-аналіз.

Було проведено аналіз потенційних ризиків і можливостей, а також розраховано основні фінансово-економічні показники проекту. Отримані результати кажуть про те, що реалізація проекту є доцільною.

Визначено, що ринок є сприятливим, а потенційні конкуренти не мають повного набору важливих для користувача властивостей і функцій.

Проте, окрім окремого впровадження в власній системі, доцільною також може бути імплементація в проекти потенційних конкурентів.

З огляду на проведений аналіз, можна чітко сказати, що подальша імплементація проекту є доцільною, адже він може знайти свою цільову аудиторію та зайняти місце на ринку.

ВИСНОВКИ

В даній роботі було проведене дослідження різних методів до формування рекомендацій і запропоновано модифікований.

Було проаналізовано актуальність досліджуваної задачі, сучасний вигляд предметної області, аналіз існуючих підходів, сучасні напрямки досліджень і методи та алгоритми надання рекомендацій. Було визначено переваги і недоліки їх використання.

Було обрано підходящий датасет – Amazon Products Data і проведена його передобробка за допомогою 64-вимірних вбудовуваннь word2vec, навчених за допомогою Gensim3.

Проведено порівняльний аналіз методів надання рекомендацій на основі глибинного навчання. Проаналізовано існуючі метрики якості моделей надання рекомендацій.

Розроблено модифікацію нейромережевої моделі формування рекомендацій. Оцінено якість пропонованої модифікації і обрано найкращу модель для досліджених даних. Пропонована модифікація виявилася точнішою майже для всіх датасетів близько на 5-7% порівняно з минулими алгоритмами.

Програмно реалізувано рекомендаційну систему на основі розробленої моделі з використанням сучасної нейромережевої бібліотеки мови Python – Pytorch із використанням середовища Jupyter Notebook та Google Colab.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Веб-ресурс: https://habr.com/ru/company/jetinfosystems/blog/453792/

2. Liaoliang Jiang, Yuting Cheng, Li Yang, Jing Li, Hongyang Yan, Xiaoqin Wang A trust-based collaborative filtering algorithm for E-commerce recommendation system Journal of Ambient Intelligence and Humanized Computing https://doi.org/10.1007/s12652-018-0928-7

3. Noveen Sachdeva and Julian McAuley. 2020. How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements. In 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’20), July 25–30, 2020, Virtual Event, China. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401281 W. Yuan, H. Wang and X. Yu et al. / Information Sciences 510 (2020) 122–134

4. Веб-ресурс: https://habr.com/ru/company/lanit/blog/420499/

5. Количева А. Дослідження методів побудови рекомендаційних систем з використанням графових нейронних мереж

6. Веб-ресурс: https://proglib.io/p/sovremennye-rekomendatelnye-sistemy-2021-03-02

7. Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In Proceedings of the 2019 World Wide Web Conference (WWW ’19), May 13–17, 2019, San Francisco, CA, USA. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313497

8. Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. 2018. News Session-Based Recommendations using Deep Neural Networks. In 3rd Workshop on Deep Learning for Recommender Systems (DLRS 2018), October 6, 2018, Vancouver, BC, Canada. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3270323.3270328

9. Carl Yang, Lanxiao Bai, Chao Zhang, an Yuan and Jiawei Han University of Illinois, Urbana Champaign 201 N. Goodwin Ave, Urbana, Illinois 61801, USA Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation KDD'17, August 13–17, 2017, Halifax, NS, Canada

10. Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3357384.3357895

11. Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang Yunming, Haokun Chen, Huifeng Guo and Yuzhou Zhang (2019) Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling

12. Weihua Yuana, Hong Wanga, Xiaomei Yua, Nan Liub, Zhenghao Li Attention-based context-aware sequential recommendation model

13. Веб-ресурс: https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

14. Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Pipei Huang, Huan Zhao, Guoliang Kang, Qiwei Chen, Wei Li, Dik Lun Lee Multi-Interest Network with Dynamic Routing for Recommendation at Tmall

15. Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming, Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In The 28th ACM International, Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357925

16. Lei Zheng, Vahid Noroozi, Philip S. Yu Joint Deep Modeling of Users and Items Using Reviews for Recommendation WSDM '17: Proceedings of the

Tenth ACM International Conference on Web Search and Data MiningFebruary 2017 Pages 425–434 https://doi.org/10.1145/3018661.3018665

17. Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In The Web Conference 2018, April 23–27, 2018, Lyons, France. ACM, New York, NY, USA,10 pages. https://doi.org/10.1145/3178876.3186070

18. Julian McAuley, Jure Leskovec Hidden factors and hidden topics: understanding rating dimensions with review text RecSys '13: Proceedings of the 7th ACM conference on Recommender systems October 2013 Pages 165–172 https://doi.org/10.1145/2507157.2507163

19. Rose Catherine, William Cohen TransNets: Learning to Transform for Recommendation RecSys '17: Proceedings of the Eleventh ACM Conference on Recommender SystemsAugust 2017 Pages 288–296https://doi.org/10.1145/3109859.3109878

20. Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In KDD ’18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom. ACM, New York, NY, USA, 10 pages.https://doi.org/10.1145/3219819.3220086

21. Hongtao Liu, Fangzhao Wu, Wenjun Wang, Xianchen Wang, Pengfei Jiao, Chuhan Wu, and Xing Xie. 2019. NRPA: Neural Recommendation with Personalized Attention. In 42nd Int’l ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR’19), July 21–25, 2019, Paris, France. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331371

22. Donghua Liu, Jing Li, Bo Du, Jun Chang, Rong Gao DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation KDD '19: Proceedings of the 25th ACM SIGKDD International

Conference on Knowledge Discovery & Data MiningJuly 2019 Pages 344–352https://doi.org/10.1145/3292500.3330906

23. Веб-ресурс: https://habr.com/ru/company/prequel/blog/567648/

24. Веб-ресурс: https://neptune.ai/blog/recommender-systems-metrics

25. Веб-ресурс:https://medium.com/intel-student-ambassadors/implementing-attention-models-in-pytorch-f947034b3e66

26. Веб-ресурс: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71

27. Веб-ресурс: https://jmcauley.ucsd.edu/data/amazon/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
#Завантаження необхідних бібліотек
"""

!pip install scikit-surprise

import os
import time
import datetime as dt
from tqdm import tqdm

from enum import Enum, unique, auto

import torch
import surprise
import numpy as np
import torch.nn as nn
import torch.nn.functional as F
from torch.autograd import Variable
import json
import pickle
import time

"""# Встановлення гіперпараметрів"""

INF = 10000.0
hyper_params = {
   'dataset': 'DeepCoNN Office',
   'weight_decay': float(1e-6),
   'lr': 0.002,
   'epochs':10,
   'batch_size': 128,

   'latent_size': 10,
   'word_embed_size': 64,
   'input_length': 1000,
   'dropout': 0.6,
}

"""#Визначення необхідних класів"""

class Model(nn.Module): #Головний клас моделі
   def __init__(self, hyper_params):
      super(Model, self).__init__()
      self.hyper_params = hyper_params

      word_vectors = load_obj(hyper_params['data_dir'] + '/word2vec')
      self.word2vec = nn.Embedding.from_pretrained(FloatTensor(word_vectors))
      self.word2vec.requires_grad = False # Не потребує тренування

      self.user_conv = TextCNN(hyper_params)
      self.item_conv = TextCNN(hyper_params)

      self.attention=Attention(hyper_params['latent_size'], hyper_params['word_embed_size'])

      self.final = nn.Sequential(
         nn.Dropout(hyper_params['dropout']),
```

```
            nn.Linear(2*hyper_params['latent_size'], hyper_params['latent_size']),
            nn.ReLU(),
            nn.Linear(hyper_params['latent_size'], 1)
        )

        self.user_bias = nn.Parameter(FloatTensor([ 0.1 for _ in range(hyper_params['total_users'] + 2) ]),
requires_grad=True)
        self.item_bias = nn.Parameter(FloatTensor([ 0.1 for _ in range(hyper_params['total_items'] + 2) ]),
requires_grad=True)
        self.global_bias = nn.Parameter(FloatTensor([ 4.0 ]), requires_grad = True)

        self.dropout = nn.Dropout(hyper_params['dropout'])
        self.relu = nn.ReLU()

    def forward(self, data):
        _, _, _, user_reviews, item_reviews, user_id, item_id = data

        final_shape = (user_id.shape[0])
        first_dim = user_id.shape[0]
        if len(user_id.shape) > 1:
            final_shape = (user_id.shape[0], user_id.shape[1])
            first_dim = user_id.shape[0] * user_id.shape[1]

        # Для роботи з негативами
        user_reviews = user_reviews.view(first_dim, -1)
        item_reviews = item_reviews.view(first_dim, -1)
        user_id = user_id.view(-1)
        item_id = item_id.view(-1)

        # Вбудовування слів
        user = self.word2vec(user_reviews)
        item = self.word2vec(item_reviews)

        # Виділення ознак
        user = self.user_conv(user)
        item = self.item_conv(item)

        # Злиття в єдиний рейтинг
        cat=self.attention(torch.cat([ user, item ], dim = -1))

        rating = self.final(cat)[:, 0]
        user_bias = self.user_bias.gather(0, user_id.view(-1)).view(user_id.shape)
        item_bias = self.item_bias.gather(0, item_id.view(-1)).view(item_id.shape)
        return (rating + user_bias + item_bias + self.global_bias).view(final_shape)

class TextCNN(nn.Module): #Додатковий клас для моделювання корситувачів і ітемів
    def __init__(self, hyper_params, window_sizes = [3]):
        super(TextCNN, self).__init__()
        self.hyper_params = hyper_params

        self.num_filters = 100
        self.kernel_size = 5

        self.convs = nn.ModuleList([
            nn.Conv2d(1, self.num_filters, [window_size, self.hyper_params['word_embed_size']],
padding=(window_size - 1, 0))
            for window_size in window_sizes
        ])
```

```
        self.fc = nn.Linear(self.num_filters * len(window_sizes), hyper_params['latent_size'])
        self.dropout = nn.Dropout(hyper_params['dropout'])

    def forward(self, x):
        in_shape = x.shape

        # Застосування шара згортки + максимальний пул для кожного розміру вікна
        x = torch.unsqueeze(x, 1)
        xs = []
        for conv in self.convs:
            x2 = F.relu(conv(x))
            x2 = torch.squeeze(x2, -1)
            x2 = F.max_pool1d(x2, x2.size(2))
            xs.append(x2)
        x = torch.cat(xs, 2)

        x = x.view(x.size(0), -1)
        x = self.dropout(self.fc(x))

        return x

class Attention(nn.Module):
    def __init__(self, feature_dim, step_dim, bias=True,  **kwargs):
        super(Attention, self).__init__(**kwargs)

        self.supports_masking = True

        self.bias = bias
        self.feature_dim = feature_dim
        self.step_dim = step_dim
        self.features_dim = 0

        weight = torch.zeros(feature_dim, 1)
        nn.init.kaiming_uniform_(weight)
        self.weight = nn.Parameter(weight)

        if bias:
            self.b = nn.Parameter(torch.zeros(step_dim))

    def forward(self, x, mask=None):
        feature_dim = self.feature_dim
        step_dim = self.step_dim

        eij = torch.mm(
            x.contiguous().view(-1, feature_dim),
            self.weight
        ).view(-1, step_dim)

        if self.bias:
            eij = eij + self.b

        eij = torch.tanh(eij)
        a = torch.exp(eij)

        if mask is not None:
            a = a * mask
```

```
        a = a / (torch.sum(a, 1, keepdim=True) + 1e-10)

        weighted_input = x * torch.unsqueeze(a, -1)
        return torch.sum(weighted_input, 1)

class MSELoss(torch.nn.Module): #Клас для визначення точності
    def __init__(self, hyper_params):
        super(MSELoss, self).__init__()

    def forward(self, output, y, return_mean = True):
        mse = torch.pow(output - y, 2)

        if return_mean: return torch.mean(mse)
        return mse

class DataLoader(): #Клас для завантаження передоброблених даних
    def __init__(
            self, hyper_params, data, user_reviews, item_reviews, negs,
            this_index_user_item = None, test_reviews = None,
            train_loader = None
        ):
        self.data = np.array(data)
        self.hyper_params = hyper_params
        self.user_reviews = user_reviews
        self.item_reviews = item_reviews
        self.this_index_user_item = this_index_user_item
        self.test_reviews = test_reviews
        self.negs = negs

        if train_loader is None:
            self.count_train_counts()
            self.calculate_reviewed_map()
        else:
            self.user_count, self.item_count = train_loader.user_count, train_loader.item_count
            self.u_to_i_map = train_loader.u_to_i_map
            self.i_to_u_map = train_loader.i_to_u_map

        self.iter = self.iter_review

    def calculate_reviewed_map(self):
        self.u_to_i_map = {}
        self.i_to_u_map = {}

        for u in self.user_reviews:
            self.u_to_i_map[u] = []
            for index in range(len(self.user_reviews[u])): self.u_to_i_map[u].append(0)

            if u not in self.this_index_user_item: continue

            for item in self.this_index_user_item[u]:
                index = self.this_index_user_item[u][item][0]
                self.u_to_i_map[u][index] = item

        inv_map = {}
        for item in self.item_reviews: inv_map[item] = {}
        for user in self.this_index_user_item:
            for item in self.this_index_user_item[user]:
```

```
            inv_map[item][user] = self.this_index_user_item[user][item]

        for i in self.item_reviews:
            self.i_to_u_map[i] = []
            for index in range(len(self.item_reviews[i])): self.i_to_u_map[i].append(0)

            for user in inv_map[i]:
                index = self.this_index_user_item[user][i][1]
                self.i_to_u_map[i][index] = user

    def __len__(self):
        num_b  = len(self.data) // int(self.hyper_params['batch_size'])
        num_b += int(len(self.data) % int(self.hyper_params['batch_size']) > 0)
        return num_b

    def count_train_counts(self):
        self.user_count, self.item_count = {}, {}

        for rating_tuple in self.data:
            if rating_tuple[0] not in self.user_count: self.user_count[rating_tuple[0]] = 0
            if rating_tuple[1] not in self.item_count: self.item_count[rating_tuple[1]] = 0

            self.user_count[rating_tuple[0]] += 1
            self.item_count[rating_tuple[1]] += 1

    def pad_and_join(self, reviews, negs = False):
        ret = []

        if len(reviews) == 0: return reviews

        flag = False
        if negs == False:
            reviews = [ reviews ]
            flag = True

        for b1 in reviews:

            ret_temp = []

            for b in b1:
                concat = []

                for review in b:
                    concat += review

                while len(concat) < self.hyper_params['input_length']:
                    concat.append(0)

                concat = concat[:self.hyper_params['input_length']]

                ret_temp.append(concat)

            ret.append(ret_temp)

        if flag == True: ret = ret[0]

        return ret
```

```
    def remove_overlap(self, u_r, i_r, user_id, item_id):
        this_reviews = []

        users_who_gave, items_reviewed = [], []

        if self.this_index_user_item is not None:
            indices = self.this_index_user_item[user_id][item_id]

            u_r_new, i_r_new = [], []

            this_reviews.append(u_r[indices[0]])

            for i in range(len(u_r)):
                if i != indices[0]:
                    u_r_new.append(u_r[i])
                    items_reviewed.append(self.u_to_i_map[user_id][i])
                else:
                    assert self.u_to_i_map[user_id][i] == item_id

            for i in range(len(i_r)):
                if i != indices[1]:
                    i_r_new.append(i_r[i])
                    users_who_gave.append(self.i_to_u_map[item_id][i])
                else:
                    assert self.i_to_u_map[item_id][i] == user_id

            u_r, i_r = u_r_new, i_r_new

        else:
            items_reviewed = self.u_to_i_map[user_id]
            users_who_gave = self.i_to_u_map[item_id]

            if self.test_reviews is not None:
                this_reviews.append(self.test_reviews[user_id][item_id])
            else: this_reviews.append([0])

        return u_r, i_r, this_reviews, users_who_gave, items_reviewed

    def iter_review(self, eval = False, simple = False):
        user, item, user_reviews, item_reviews, y_ratings, this_reviews = [], [], [], [], [], []
        users_who_gave, items_reviewed = [], []
        batch_num = 0

        for rating_tuple in tqdm(self.data):

            user.append(int(rating_tuple[0]))
            item.append(int(rating_tuple[1]))
            y_ratings.append(rating_tuple[2])

            u_r = self.user_reviews[int(rating_tuple[0])]
            i_r = self.item_reviews[int(rating_tuple[1])]
            u_r, i_r, this_reviews_, users_who_gave_, items_reviewed_ = self.remove_overlap(u_r, i_r,
int(rating_tuple[0]), int(rating_tuple[1]))

            user_reviews.append(u_r)
            item_reviews.append(i_r)
            this_reviews.append(this_reviews_)
            users_who_gave.append(users_who_gave_)
```

```
        items_reviewed.append(items_reviewed_)

        if len(user) % int(self.hyper_params['batch_size']) == 0:

            for b in range(len(users_who_gave)):
                while len(users_who_gave[b]) < 10: users_who_gave[b].append(self.hyper_params['total_users'] + 1)
                while len(items_reviewed[b]) < 10: items_reviewed[b].append(self.hyper_params['total_items'] + 1)

                users_who_gave[b] = users_who_gave[b][:10]
                items_reviewed[b] = items_reviewed[b][:10]

            if simple == True:
                yield [
                    self.pad_and_join(this_reviews),
                    users_who_gave,
                    items_reviewed,
                    self.pad_and_join(user_reviews),
                    self.pad_and_join(item_reviews),
                    user,
                    item,
                ], y_ratings

            else:
                yield [
                    Variable(LongTensor(self.pad_and_join(this_reviews))),
                    Variable(LongTensor(users_who_gave)),
                    Variable(LongTensor(items_reviewed)),
                    Variable(LongTensor(self.pad_and_join(user_reviews))),
                    Variable(LongTensor(self.pad_and_join(item_reviews))),
                    Variable(LongTensor(user)),
                    Variable(LongTensor(item)),
                ], Variable(FloatTensor(y_ratings))

            batch_num += 1
            user, item, user_reviews, item_reviews, this_reviews, y_ratings = [], [], [], [], [], []
            users_who_gave, items_reviewed = [], []

    if len(user) > 0:

        for b in range(len(users_who_gave)):
            while len(users_who_gave[b]) < 10: users_who_gave[b].append(self.hyper_params['total_users'] + 1)
            while len(items_reviewed[b]) < 10: items_reviewed[b].append(self.hyper_params['total_items'] + 1)

            users_who_gave[b] = users_who_gave[b][:10]
            items_reviewed[b] = items_reviewed[b][:10]

        if simple == True:
            yield [
                self.pad_and_join(this_reviews),
                users_who_gave,
                items_reviewed,
                self.pad_and_join(user_reviews),
                self.pad_and_join(item_reviews),
                user,
                item,
            ], y_ratings

        else:
```

```
            yield [
                Variable(LongTensor(self.pad_and_join(this_reviews))),
                Variable(LongTensor(users_who_gave)),
                Variable(LongTensor(items_reviewed)),
                Variable(LongTensor(self.pad_and_join(user_reviews))),
                Variable(LongTensor(self.pad_and_join(item_reviews))),
                Variable(LongTensor(user)),
                Variable(LongTensor(item)),
            ], Variable(FloatTensor(y_ratings))

    def iter_negs(self, review):
        user, item, y_ratings, user_reviews, item_reviews, this_reviews = [], [], [], [], [], []
        users_who_gave, items_reviewed = [], []

        for u in tqdm(self.negs):

            i = self.negs[u][0][0]

            temp_u, temp_i = [], []
            temp_user_reviews, temp_item_reviews, temp_this_reviews = [], [], []
            temp_users_who_gave, temp_items_reviewed = [], []

            for i2 in [i] + self.negs[u][1]:

                temp_u.append(u)
                temp_i.append(i2)

                if review == True:

                    u_r = self.user_reviews[u]
                    i_r = self.item_reviews[i2]

                    u_r, i_r, this_reviews_, users_who_gave_, items_reviewed_ = self.remove_overlap(u_r, i_r, u, i)

                    temp_user_reviews.append(u_r)
                    temp_item_reviews.append(i_r)
                    temp_this_reviews.append(this_reviews_)
                    temp_users_who_gave.append(users_who_gave_)
                    temp_items_reviewed.append(items_reviewed_)

            user.append(temp_u)
            item.append(temp_i)
            y_ratings.append(0.0)

            if review == True:
                for b in range(len(temp_users_who_gave)):
                    while len(temp_users_who_gave[b]) < 10:
temp_users_who_gave[b].append(self.hyper_params['total_users'] + 1)
                    while len(temp_items_reviewed[b]) < 10:
temp_items_reviewed[b].append(self.hyper_params['total_items'] + 1)

                    temp_users_who_gave[b] = temp_users_who_gave[b][:10]
                    temp_items_reviewed[b] = temp_items_reviewed[b][:10]

                user_reviews.append(temp_user_reviews)
                item_reviews.append(temp_item_reviews)
                this_reviews.append(temp_this_reviews)
                users_who_gave.append(temp_users_who_gave)
```

```
            items_reviewed.append(temp_items_reviewed)

        if len(user) % int(self.hyper_params['batch_size']) == 0:

            yield [
              Variable(LongTensor(self.pad_and_join(this_reviews, negs = True))),
              Variable(LongTensor(users_who_gave)),
              Variable(LongTensor(items_reviewed)),
              Variable(LongTensor(self.pad_and_join(user_reviews, negs = True))),
              Variable(LongTensor(self.pad_and_join(item_reviews, negs = True))),
              Variable(LongTensor(user)),
              Variable(LongTensor(item)),
            ], Variable(FloatTensor(y_ratings))

            user, item, y_ratings, user_reviews, item_reviews, this_reviews = [], [], [], [], [], []
            users_who_gave, items_reviewed = [], []

      if len(user) > 0:
        yield [
          Variable(LongTensor(self.pad_and_join(this_reviews, negs = True))),
          Variable(LongTensor(users_who_gave)),
          Variable(LongTensor(items_reviewed)),
          Variable(LongTensor(self.pad_and_join(user_reviews, negs = True))),
          Variable(LongTensor(self.pad_and_join(item_reviews, negs = True))),
          Variable(LongTensor(user)),
          Variable(LongTensor(item)),
        ], Variable(FloatTensor(y_ratings))

"""#Визначення додаткових функцій"""

def get_common_path(hyper_params): # створення шляху до передоброблених даних
  method, fm = 'DeepAtnCoNN', False
  common_path  = str(method)
  common_path += '_' + str(hyper_params['dataset'])

  common_path += '_word_embed_size_' + str(hyper_params['word_embed_size'])
  common_path += '_latent_size_' + str(hyper_params['latent_size'])
  common_path += '_fm_' + str(fm)

  common_path += '_wd_' + str(hyper_params['weight_decay'])
  common_path += '_lr_' + str(hyper_params['lr'])
  common_path += '_dropout_' + str(hyper_params['dropout'])
  common_path += '_input_len_' + str(hyper_params['input_length'])

  return common_path

def load_obj(name):
  with open(name + '.pkl', 'rb') as f:
    return pickle.load(f)

def load_obj_json(name):
  with open(name + '.json', 'r') as f:
    return json.load(f)

def load_user_item_counts(hyper_params):
  user_count = load_obj(hyper_params['data_dir'] + 'user_count')
  item_count = load_obj(hyper_params['data_dir'] + 'item_count')
  return user_count, item_count
```

```
def file_write(log_file, s, dont_print=False):
    if dont_print == False: print(s)
    f = open(log_file, 'a')
    f.write(s+'\n')
    f.close()

def log_end_epoch(hyper_params, metrics, epoch, time_elpased, metrics_on = '(VAL)'):
    string2 = ""
    for m in metrics: string2 += " | " + m + ' = ' + str(metrics[m])
    string2 += ' ' + metrics_on

    ss  = '-' * 89
    ss += '\n| end of epoch {} | time: {:5.2f}s'.format(epoch, time_elpased)
    ss += string2
    ss += '\n'
    ss += '-' * 89
    file_write(hyper_params['log_file'], ss)

def xavier_init(model):
    for p in model.parameters():
        if p.dim() > 1:
            torch.nn.init.xavier_uniform_(p)

def load_data(hyper_params, load_negs = True):
    print("Loading data...")

    train = load_obj(hyper_params['data_dir'] + 'train')
    test = load_obj(hyper_params['data_dir'] + 'test')
    val = load_obj(hyper_params['data_dir'] + 'val')

    user_reviews, item_reviews = None, None
    user_reviews = load_obj(hyper_params['data_dir'] + 'user_reviews')
    item_reviews = load_obj(hyper_params['data_dir'] + 'item_reviews')

    negs = None
    if load_negs:
        negs = load_obj(hyper_params['data_dir'] + 'negs')

    this_index_user_item = load_obj(hyper_params['data_dir'] + 'this_index_user_item')
    test_reviews = load_obj(hyper_params['data_dir'] + 'test_reviews')
    num_users, num_items, num_words = load_obj(hyper_params['data_dir'] + 'num_users_items')

    hyper_params['total_users'] = num_users
    hyper_params['total_items'] = num_items
    hyper_params['total_words'] = num_words

    train_loader = DataLoader(
        hyper_params, train, user_reviews, item_reviews, negs,
        this_index_user_item = this_index_user_item
    )

    return train_loader, \
            DataLoader(hyper_params, test, user_reviews, item_reviews, negs,
                    test_reviews = test_reviews, train_loader = train_loader), \
            DataLoader(hyper_params, val, user_reviews, item_reviews, negs,
                    test_reviews = test_reviews, train_loader = train_loader), \
            hyper_params
```

```
def evaluate(model, criterion, reader, hyper_params, user_count, item_count, review):
    metrics = {}
    total_loss = FloatTensor([ 0.0 ])
    total_batches = 0.0
    mse_right, conv_loss = 0.0, 0.0
    total_temp, total_temp2 = 0.0, 0.0
    temp_ndcg=0.0
    n=0

    user_count_mse_map = {}
    item_count_mse_map = {}

    y_pred = []
    y_true = []

    temp_ndcg=[]

    model.eval()

    with torch.no_grad():
        at = 0

        for data, y in reader.iter(eval = True):
            _, _, _, _, _, user, item = data

            output = model(data)

            y_pred.append(y)
            y_true.append(output)

            mse = criterion(output, y, return_mean = False).data

            total_temp += torch.sum(mse)
            total_temp2 += float(int(output.shape[0]))

            for batch in range(int(y.shape[0])):
                user_id = int(user[batch])
                item_id = int(item[batch])

                if user_id not in user_count: user_count[user_id] = 0
                if item_id not in item_count: item_count[item_id] = 0

                if user_count[user_id] not in user_count_mse_map: user_count_mse_map[ user_count[user_id] ] = []
                if item_count[item_id] not in item_count_mse_map: item_count_mse_map[ item_count[item_id] ] = []

                user_count_mse_map[ user_count[user_id] ].append(float(mse[batch]))
                item_count_mse_map[ item_count[item_id] ].append(float(mse[batch]))

            total_batches += 1.0

        metrics['MSE'] = round(float(total_temp) / total_temp2, 4)
        i=0
        for vec in y_pred:
          vec_act=y_true[i]
          ndcg_at_k = torch_ndcg_at_k(vec.view(1, -1), vec_act.view(1, -1), k=5)
          temp_ndcg.append(ndcg_at_k.cpu().numpy())
          i+=1
```

```
        count=0
        NDCG_batch_temp=0.0

        for ng in temp_ndcg:
          count+=1
          NDCG_batch_temp+=ng

        NDCG_batch=NDCG_batch_temp/count
        print(NDCG_batch)
        metrics['NDCG']=NDCG_batch

    return metrics, user_count_mse_map, item_count_mse_map

class LABEL_TYPE(Enum):
    """ The types of labels of supported datasets """
    MultiLabel = auto()
    Permutation = auto()

def torch_dcg_at_k(batch_rankings, cutoff=None, label_type=LABEL_TYPE.MultiLabel, device='cuda:0'):
            if cutoff is None:
                        cutoff = batch_rankings.size(1)

            if LABEL_TYPE.MultiLabel == label_type:
                        batch_numerators = torch.pow(2.0, batch_rankings[:, 0:cutoff]) - 1.0
            elif LABEL_TYPE.Permutation == label_type:
                        batch_numerators = batch_rankings[:, 0:cutoff]
            else:
                        raise NotImplementedError

            batch_discounts = torch.log2(torch.arange(cutoff, dtype=torch.float,
device=device).expand_as(batch_numerators) + 2.0)
            batch_dcg_at_k = torch.sum(batch_numerators/batch_discounts, dim=1, keepdim=True)
            return batch_dcg_at_k

def torch_ndcg_at_k(batch_predict_rankings, batch_ideal_rankings, k=None, device='cuda:0',
label_type=LABEL_TYPE.MultiLabel):
            batch_sys_dcg_at_k = torch_dcg_at_k(batch_predict_rankings, cutoff=k, label_type=label_type,
device=device)
            batch_ideal_dcg_at_k = torch_dcg_at_k(batch_ideal_rankings, cutoff=k, label_type=label_type,
device=device)
            batch_ndcg_at_k = batch_sys_dcg_at_k / batch_ideal_dcg_at_k
            return batch_ndcg_at_k

def eval_ranking(model, reader, hyper_params, review = False):
    top_indices = torch.zeros(len(reader.data), 1).long()
    at = 0

    with torch.no_grad():
        for data, y in reader.iter_negs(review):
            output = model(data)

            for batch in range(int(y.shape[0])):
                vals, inds = torch.topk(output[batch], k = 1, sorted = True)
                top_indices[at, :] = inds
                at += 1

    metrics = {}
```

```
    ks = [1]
    for k in ks:
        hr, total = 0.0, 0.0

        for i in range(at):
            pred = top_indices[i].cpu().numpy()[:k]
            if 0 in pred: hr += 1.0
            total += 1.0

        if total==0.0:total=1.0

        metrics['HR@' + str(k)] = round(100.0 * hr / total, 2)

    return metrics

def train(model, criterion, optimizer, reader, hyper_params):

    model.train()

    metrics = {}
    metrics['MSE'] = 0.0

    total_x, total_batches = 0.0, 0.0

    for data, y in reader.iter():

        model.zero_grad()
        optimizer.zero_grad()

        all_output = model(data)

        loss = criterion(all_output, y, return_mean = False)
        metrics['MSE'] += float(torch.sum(loss.data))
        loss = torch.mean(loss)
        loss.backward()
        optimizer.step()

        try: total_x += float(int(all_output.shape[0]))
        except: total_x += float(int(all_output[0].shape[0]))
        total_batches += 1

    metrics['MSE'] = round(metrics['MSE'] / float(total_x), 4)
    return metrics

def train_complete(hyper_params, Model, train_reader, val_reader, user_count, item_count, model, review =
True):

    file_write(hyper_params['log_file'], "\n\nSimulation run on: " + str(dt.datetime.now()) + "\n\n")
    file_write(hyper_params['log_file'], "Data reading complete!")
    file_write(hyper_params['log_file'], "Number of train batches: {:4d}".format(len(train_reader)))
    file_write(hyper_params['log_file'], "Number of validation batches: {:4d}".format(len(val_reader)))

    criterion = MSELoss(hyper_params)
    optimizer = torch.optim.Adam(
        model.parameters(), lr=hyper_params['lr'], weight_decay=hyper_params['weight_decay']
    )

    file_write(hyper_params['log_file'], str(model))
```

```
    file_write(hyper_params['log_file'], "\nModel Built!\nStarting Training...\n")

    try:
        best_MSE = float(INF)

        for epoch in range(1, hyper_params['epochs'] + 1):
            epoch_start_time = time.time()

            metrics = train(
                model, criterion, optimizer, train_reader, hyper_params
            )
            metrics['dataset'] = hyper_params['dataset']

            metrics, _, _ = evaluate(
                model, criterion, val_reader, hyper_params,
                user_count, item_count, review = review
            )
            metrics['dataset'] = hyper_params['dataset']
            log_end_epoch(hyper_params, metrics, epoch, time.time() - epoch_start_time, metrics_on = '(VAL)')

            if metrics['MSE'] < best_MSE:
                print("Saving model...")
                torch.save(model.state_dict(), hyper_params['model_path'])
                best_MSE = metrics['MSE']

    except KeyboardInterrupt: print('Exiting from training early')

    model = Model(hyper_params)
    if is_cuda_available: model = model.cuda()
    model.load_state_dict(torch.load(hyper_params['model_path']))
    model.eval()

    return model

def main_pytorch(hyper_params, gpu_id = None):
    user_count, item_count = load_user_item_counts(hyper_params)
    review_based_model = True
    train_reader, test_reader, val_reader, hyper_params = load_data(hyper_params)

    model = Model(hyper_params)
    if is_cuda_available: model = model.cuda()
    xavier_init(model)

    start_time = time.time()
    model = train_complete(
        hyper_params, Model, train_reader,
        val_reader, user_count, item_count, model, review = review_based_model
    )

    criterion = MSELoss(hyper_params)
    metrics, user_count_mse_map, item_count_mse_map = evaluate(
        model, criterion, test_reader, hyper_params,
        user_count, item_count, review = review_based_model
    )

    _, test_reader2, _, _ = load_data(hyper_params)
    metrics.update(eval_ranking(model, test_reader2, hyper_params, review = review_based_model))
    log_end_epoch(hyper_params, metrics, 'final', time.time() - start_time, metrics_on = '(TEST)')
```

```
    return metrics, user_count_mse_map, item_count_mse_map

"""#Головний код"""

is_cuda_available = torch.cuda.is_available()

if is_cuda_available:
    print("Using CUDA...\n")
    LongTensor = torch.cuda.LongTensor
    FloatTensor = torch.cuda.FloatTensor
else:
    LongTensor = torch.LongTensor
    FloatTensor = torch.FloatTensor

common_path = get_common_path(hyper_params)
hyper_params['common_path'] = common_path
hyper_params['log_file'] = '/content/drive/MyDrive/saved_logs/' + common_path
hyper_params['model_path'] ='/content/drive/MyDrive/saved_models/' + common_path

os.makedirs('/content/drive/MyDrive/saved_logs/', exist_ok = True)
os.makedirs('/content/drive/MyDrive/saved_models/', exist_ok = True)

hyper_params['data_dir']  = "/content/drive/MyDrive/"
hyper_params['data_dir'] += hyper_params['dataset'] + "/"

gpu_id = None
method = main_pytorch

metrics, user_count_mse_map, item_count_mse_map = method(hyper_params, gpu_id = gpu_id)
```