ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочена назва	Повна назва
MLP	Multilayer Perceptron
RS	Recommendation Systems
VAE	Variational Auto Encoder
AE	Auto Encoder
NGCF	Neural graph collaborative filtering
MSE	Mean Square Error
ML	Machine Learning
DL	Deep Learning
DNN	Deep Neural Networks
PCA	Principal component analysis
GNN	Graph Neural Networks
DCG	Discounted Cumulative Gain
NCF	Neural Collaborative Filtering
EDA	Explorative Data Analysis

ЗМІСТ

ВСТУП

Предмет дослідження

Предметом дослідження є оцінка якості роботи алгоритмів побудови рекомендацій на основі глибокого навчання.

Завдання дослідження

Провести комплексний аналіз систем рекомендацій на основі моделей глибокого навчання. Дослідити ключові особливості алгоритмів рекомендацій на основі моделей глибокого навчання. Класифікувати і оцінити фактори впливу на якість роботи алгоритмів. Розробити стратегію для комплексної оцінки і порівняння алгоритмів рекомендацій на основі глибокого навчання.

Актуальність дослідження

Внаслідок розробки великої кількості нових рекомендаційних алгоритмів постало критичне питання у вітсутності єдиного підходу до оцінки їх ефективності, що призводитть до нерепродуктивних та несправедливих результів їх порівняння.

1 ОГЛЯД СУЧАСНОГО СТАНУ ОБЛАСТІ ПОБУДОВИ СИСТЕМ РЕКОМЕНДАЦІЙ

У наслідок зривного росту сфери інформаційних технологій і стрімким пересіченням буденного життя звичайних людей із системами обміну даними відкриваються нові можливості для впливу на процеси прийняття рішень, таких як - який фільм переглянути, прослухати музику або яку річ купити. Крім того за останнє десятиліття користувачі стикаються із дилемою вибору серед великої кількості варіантів. Як результат, автоматичні рекомендації є важливими для покращення досвіду користувачів та зменшення перевантаження інформації. Загалом, системи рекомендацій відіграли незамінну роль у різних системах фільтрації інформації для підвищення цінності бізнесу та полегшення процесів прийняття рішень. Netfilx - типовий приклад системи рекомендацій (Рис. 1.1).

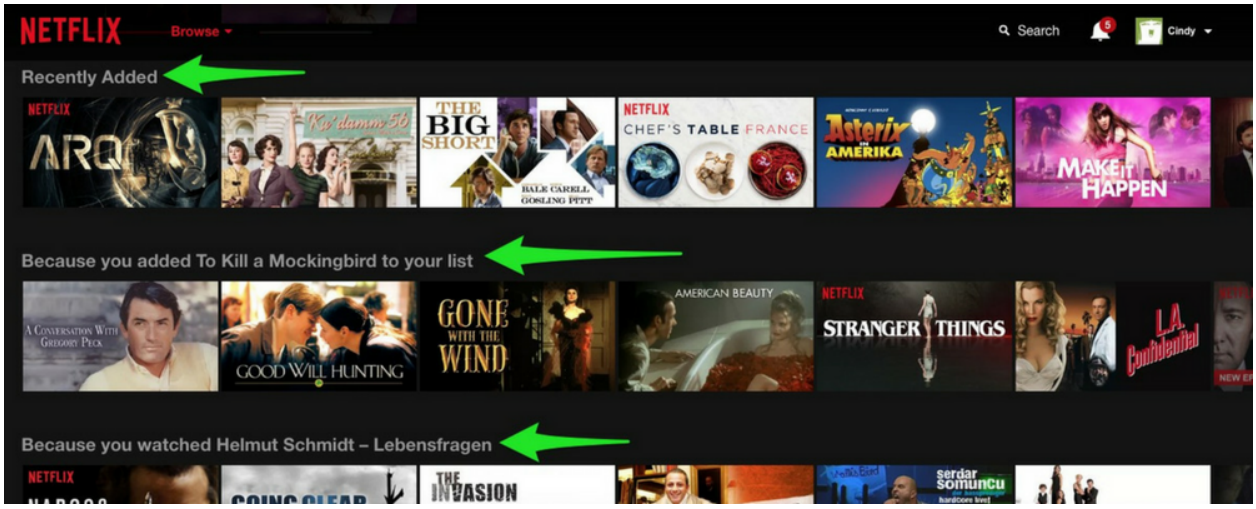


Рисунок 1.1: Інтерфейс головної сторінки стрімінгової платформи Netflix.

У своїй найпростішій формі персоналізовані рекомендації пропонуються як ранжовані списки елементів. Виконуючи це рейтингування, RS намагаються передбачити які продукти чи послуги є найбільш релевантними, виходячи з уподобань і обмежень користувача. Щоб виконати таке обчислювальне завдання, RS збирають інформацію від користувачів щодо їхніх уподобань, які або явно виражені, наприклад, як рейтинги для продуктів, або виводяться шляхом інтерпретації дій користувача. Наприклад, RS може розглядати навігацію до сторінки конкретного продукту як неявний знак переваги для елементів, показаних на цій сторінці.

В останні роки інтерес до рекомендаційних систем різко зріс, про це свідчать такі факти:

1. Системи рекомендацій відіграють важливу роль на високорейтингових інтернет-сайтах, таких як Amazon.com, YouTube, Netflix, Spotify, LinkedIn, Facebook, Tripadvisor, та IMDb. Крім того, багато медіакомпаній зараз розробляють і розгортають RSs як частину послуг які вони надають своїм передплатникам. Наприклад, Netflix, онлайн-провайдер потокового мультимедіа, присудив приз у мільйон доларів команді, яка першою зуміла суттєво покращити продуктивність своєї системи рекомендацій.
2. Активна робота введеться у наукові сфері. Популярні академічні журнали активно публікують результати досліджень у сфері систем рекомендацій. Існують різні методи проектування цих систем, починаючи від простого (наприклад, заснованого лише на номінальних елементах від одного користувача) до надзвичайно складного. Складні рекомендаційні системи використовують різноманітні джерела даних і часто використовують нелінійні методи навчання. Таким чином, завдання рекомендації забезпечує відмінний простір для застосування методологій машинного навчання. Оскільки користувачі продовжують споживати вміст і дають більше даних, ми можемо створити системи на основі машинного навчання для використання цих даних для надання кращих та кращих рекомендацій.

1.1 Призначення системи рекомендацій

Система рекомендацій знайшла своє призначення у багатьох сферах виконуючи наступні завдання:

- Збільшення кількості проданого товару. З точки зору бізнесу, збільшення конвертацій (співвідношення кількості переглядів до цільових дій - покупки, продажу, перегляду і т.д.) є головною метою.
- Збільшення різноманітності (diversity) проданого товару. Тобто реалізація непопулярного товару без ризику втратити прихильність користувачів
- Збільшення задоволеності користувачів. Добре розроблена система ре-

комендацій може покращити досвід користувачів у програмі. Користувач вважатиме рекомендації цікавими, актуальними та приємними. Це призводить до більш високого загального задоволення.
- Зрозуміти потреби користувача. Хороша система рекомендацій детально описує бажання користувачів, явно або неявно. Тоді бізнес може вирішити повторно використовувати ці знання для багатьох інших цілей.

1.2 Прикладні сфери використання

Дослідження рекомендаційних систем, як правило, проводяться з сильним акцентом на практичні та комерційні програми. Домен застосування суттєво впливає на тип алгоритмічного підходу, який слід застосовувати. Умовно можна надати таку таксономію систем рекомендацій що класифікує існуючі програми до конкретних доменів:

- Розваги - Рекомендації щодо фільмів (Netflix), Музика (Spotify, Pandora) та мобільних додатків (Apple Store, Android Store).
- Соціальні мережі.
- Вміст-персоналізовані газети (The New York Times, The Wall Street Journal), рекомендація щодо зображень (Pinterest), рекомендації вебсторінок, електронного навчання (Coursera) та електронних листів (Gmail).
- Електронна комерція - Рекомендації для споживачів продуктів для придбання, таких як книги (Amazon), косметика, одяг та багато іншого.
- Рекомендації туристичних послуг (Skyscanner), експертів з консультацій (Styleseat, ClassPass), будинки для оренди (AirBnB) або послуги із пошуку партнера (Tinder, Bumble).

1.3 Проблематика напрямку

Хоча завдання створення автоматичної рекомендації не є новим, залишається багато проблем які сприяють дослідженням сучасних систем, особливо у задачах колаборативної фільтрації, які мають на меті змоделювати поведінку користувачів для полегшення рекомендацій. Основі питання:

- Розрідженість (Data sparsity): Проблема розрідження є досить помітною у системах колаборативної фільтрації, оскільки лише невелика кількість користувачів надає рейтинги, і лише невелика кількість предметів має достатню кількість оцінок. Це явище відоме як "проблема холодного старту"(cold start problem), що запобігає системі рекомендацій генерувати змістовні прогнози для нових користувачів через обмеженість даних.
- Надмірна спеціалізованість (Over specialized): Система рекомендує об'єкти дуже схожі до тих які вже відомі користувачу. Незважаючи на те, що це свідчить про хороший загальний рівень прогнозувань моделі (її ефективність), система не привносить різноманіття, відкриття нового, не знайомого контенту.
- Упередження на основі популярності (Popularity bias | "Harry Potter Effect"): Частково проблема систем колаборативної фільтрації. Через нерівномірність матриці взаємодій алгоритм рекомендує об'єкти із великою кількістю оцінок (відгуків), а при недостатній кількості обє'кти ігноруються.
- Точність: Системи рекомендацій повинні забезпечити високий рівень точності прогнозування, щоб забезпечити якість, яку користувачі часто вимагають. У дослідженні точність зазвичай оцінюється/досліджується шляхом прогнозування рейтингу та рейтингу предметів.
- Маштабованість: Системи рекомендацій у реальному світі розгортаються в динамічному і інтерактивному середовищі - дані користувачів та елементів швидко надходять і виходять. Важливо вирішити обчислювальні вимоги та час навчання/виведення, необхідні для ефективної обробки цих даних.
- Шахрайство. У рекомендаційних системах, де кожен може ставити оцінки, люди можуть давати позитивні оцінки своїм предметам і погані своїм конкурентам. Також, рекомендаційні системи стали сильно впливати на продажі та прибуток, з тих пір як отримали широке застосування в комерційних сайтах. Це призводить до того, що недобросовісні постачальники намагаються шахрайським чином піднімати рейтинг своїх продуктів і знижувати рейтинг свої конкурентів.
- Різноманітність. Колаборативна фільтрація спочатку визнана збільши-

ти різноманітність, щоб дозволяти відкривати користувачам нові продукти з незліченної множини. Однак деякі алгоритми, зокрема основні на продажах і рейтингах, створюють дуже складні умови для просування нових і маловідомих продуктів, так як їх заміщають популярні продукти, які давно перебувають на ринку. Це в свою чергу тільки збільшує ефект «багаті стають ще багатшими» і приводить до меншої різноманітності.

- Білі ворони. До «білих ворон» відносяться користувачі, чия думка постійно не збігається з більшістю інших. Через унікальність смаку їм неможливо щось рекомендувати. Однак, такі люди мають проблеми з отриманням рекомендацій і в реальному житті, тому пошуки вирішення даної проблеми в даний час не ведуться.
- Проблема холодного старту. Нові предмети або користувачі представляють велику проблему для рекомендаційних систем. Частково проблему допомагає вирішити підхід, заснований на аналізі вмісту, так як він покладається не на оцінки, а на атрибути, що допомагає включати нові предмети в рекомендації для користувачів. Однак проблему з наданням рекомендації для нового користувача вирішити складніше.
- Синонімія. Синонімією називається тенденція схожих і однакових предметів мати різні імена. Більшість рекомендаційних систем не здатні виявити ці приховані зв'язки і тому відносяться до цих предметів як до різних. Наприклад, «фільми для дітей» та «дитячий фільм» відносяться до одного жанру, але система сприймає їх як різні.
- Розрідженість даних. Як правило, більшість комерційних рекомендаційних систем заснована на великій кількості даних (товарів), в той час як більшість користувачів не ставить оцінки товарам. В результаті цього матриця «предмет-користувач» виходить дуже великою і розрідженою, що представляє проблеми при обчисленні рекомендацій. Ця проблема особливо гостра для нових, щойно створених систем. Також розрідженість даних підсилює проблему холодного старту.

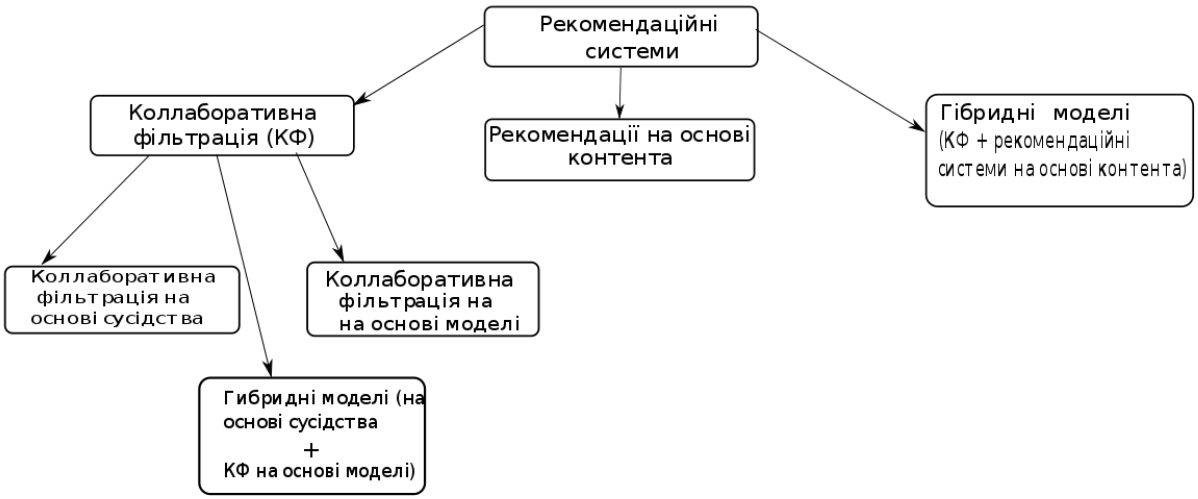


Рисунок 1.2: Поділ систем рекомендацій за категоріями

1.4 Класифікація систем рекомендацій

Існує два способи екстраполяції рейтингу від відомих об'єктів до невідомих. Перший полягає у створені евристики, яка визначається функцією подібності та емпірично підтверджує ефективність. Друга навчається використовуючи функцію втрат, наприклад MSE або cross-entropy loss. Отримавши оцінку рейтингу для об'єктів рекомендацій ми рекомендуємо top n елементів із найвищим рейтингом. Тому загальна класифікація систем рекомендацій наступна (Рис 1.2):

- Фільтрування на основі контенту: система рекомендує нові елементи із характеристиками подібними до існуючих елементів, яким користувач віддав перевагу в минулому. Отже, генерація рекомендацій повністю залежить від історії взаємодій із елементами.
- Колаборативна фільтрація: система рекомендує об'єкти новим користувачам зі смаками схожими до існуючих користувачів у базі даних.Отже, якість рекомендацій залежить від уподобань користувача.
- Гібридний підхід: гібридна система поєднує фільтрацію на основі вмісту та колаборативну фільтрацію для створення рекомендацій.

1.4.1 Фільтрування на основі вмісту

У системах рекомендацій на основі вмісту функція корисності $F(u, i)$ елемента i для користувача u оцінюється на основі функцій $f(u, i_k)$, призна-

ченого користувачем u для кожного елемента $i_k \in I$, схожий на предмет i. Наприклад, у системі рекомендацій музики, щоб рекомендувати нові пісні користувачеві U,підхід до рекомендацій на основі вмісту намагається зрозуміти подібність між піснями, які користувач u часто слухав у минулому. Тоді лише пісні, які мають високу ступінь подібності з будь-якими уподобаннями користувача, згодом рекомендуються. Основні обмеження систем рекомендацій на основі вмісту:

1. Обмежений аналіз: підходи на основі контенту обмежені функціями, які явно пов'язані з рекомендованими елементами. Отже, щоб мати можливість дати оцінку схожості, вміст повинен бути:
   1) У форматі, який можна обробити автоматично
   2) Або може бути призначений елементам вручну.

   У першому сценарії створитти функції оцінки схожості в неструктурованих даних непросто, такі як зображення та аудіо. У другому сценарії часто недоцільно призначати атрибути вручну через обмежені обчислювальні та людські ресурси.
2. Гомогенність: Коли система може рекомендувати лише елементи, які високо оцінюють профіль користувача, користувач обмежується лише предметами, подібними до тих, що вже оцінені. Іншими словами, підходи, засновані на контенті, не дають різноманітних рекомендацій. В ідеалі користувачеві слід представити широкий спектр варіантів, а не просто однорідний набір альтернатив.
3. Рекомендації для нового користувача: Користувач повинен оцінити достатню кількість предметів до того, як система рекомендацій на основі вмісту може зрозуміти його/її вподобання та представити його/їй довірливими предметами. Таким чином, новий користувач, який не має попередніх рейтингів, не отримає точних рекомендацій

1.4.2 Колаборативна фільтрація

У системах колаборативної фільтрації ми намагаємось передбачити корисність елементів для певного користувача на основі елементів, раніше оцінених іншими користувачами. Тобто, функція корисності $F(u, i)$ елемента I для користувача U оцінюється на основі функцій $F(u_j, i)$, призначеного елементу I тими користувачами $u_j \in U$, які схожі на користувача u. Наприклад,

у контексті програми рекомендацій щодо книги, щоб рекомендувати книги користувачеві U, система спільної фільтрації спочатку знаходить «однодумців» користувача U або інших користувачів із подібними смаками в книгах (які оцінюють однакові книги аналогічно). Тоді, рекомендується лише книги, які найбільше подобаються похожим користувачам.

Алгоритми колаборативної фільтрації можна розділити на два основні класи: на основі сусідства та на основі моделі.

1.4.3 Підхід оснований на сусідстві

Алгоритм, заснований на сусідстві, обчислює подібність двох користувачів або виробів, виробляє прогноз для користувача, приймаючи середнє зважене всіх оцінок. Обчислення схожості між виробами або користувачами є важливою частиною цього підходу. Багаторазові заходи, такі як кореляції Пірсона і схожість, заснована на скалярному добутку, використовується для цього.

Схожість двох користувачів X, Y через кореляцію Пірсона визначається як:

$$simm(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r}_x)(r_{y,i} - \overline{r}_y)}{\sqrt{\sum_{i \in I} (r_{x,i} - \overline{r}_x)^2 (r_{y,i} - \overline{r}_y)^2}} \tag{1.1}$$

- де $I_{x,y}$ набір елементів оцінений як користувачем x так і користувачем y.

Заснований на користувачеві алгоритм top-N рекомендації використовує засновану на подібності векторну модель для визначення K — більшості подібних користувачів до активного користувача. Після того, як знайдені найбільш схожі користувачі, їх відповідні матриці агрегуються для визначення рекомендованого набору елементів. Популярний метод, знаходження схожих користувачів — Locality-sensitive hashing, який реалізує механізм пошуку найближчих сусідів у лінійному часі.

Переваги цього підходу включають в себе: очікуваність результатів, що є важливим аспектом рекомендаційних систем; просте створення і використання; просте полегшення нових даних; добра масштабованість зі співавторами рейтингових пунктів.

Є також кілька недоліків при такому підході. Його продуктивність знижується, коли дані становляться розрідженими, що трапляється часто з виробами, пов'язаними з мережею. Це ускладнює масштабованість такого підходу

і створює проблеми з великими наборами даних. Хоча він може ефективно обробляти нових користувачів, тому що спирається на структури даних, додавання нових елементів стає більш складним, що, як правило, спирається уявленням про конкретну складову векторного простору. Додавання нових елементів вимагає включення нового пункту і повторного включення всіх елементів у структурі.

1.4.4 Підхід заснований на моделі

Даний підхід надає рекомендації, вимірюючи параметри статистичних моделей для оцінок користувачів, побудованих за допомогою таких методів як, метод баєсовских мереж, кластеризації, латентно-семантичної моделі , такі як сингулярний розклад, імовірнісний латентно-семантичний аналіз, прихований розподіл Дирихле і марковський процес вирішування на основі моделей. Моделі розробляються з використанням інтелектуального аналізу даних, алгоритмів машинного навчання, щоб знайти закономірності на основі навчальних даних. Число параметрів в моделі може бути зменшено в залежності від типу за допомогою методу головних компонент.

Цей підхід є більш комплексним і дає більш точні прогнози, оскільки допомагає розкрити латентні фактори, що пояснюють спостережувані оцінки.

Даний підхід має ряд переваг. Він обробляє розріджені матриці краще, ніж підхід заснований на сусідстві, що в свою чергу допомагає з масштабністю великих наборів даних.

Недоліки цього підходу полягають в «дорогому» створенні моделі. Необхідний компроміс між точністю і розміром моделі, тому що можна втратити корисну інформацію у зв'язку із скороченням моделей.

Алгоритми на основі пам'яті-це евристика, яка прогнозує рейтинги на основі всієї колекції раніше оцінених елементів користувачами системи. Алгоритми на основі моделі використовують колекцію рейтингів для навчання моделі (ML), які після валідації потім використовуються для прогнозування рейтингів.

1.4.5 Гібридні системи

Даний підхід об'єднує в собі підхід заснований на сусідстві і заснований на моделі. Гібридний підхід є найпоширенішим при розробці рекомендаційних

систем для комерційних сайтів, так як він допомагає подолати обмеження початкового оригінального підходу (заснованого на сусідстві) і поліпшити якість прогнозів. Цей підхід також дозволяє подолати проблему розрідженості даних і втрати інформації. Однак даний підхід складний і дорогий у реалізації та застосуванні.

1.5 Постановка задачі системи рекомендацій

У рекомендаційних системах релевантність об'єкта зазвичай представлена рейтингом, що вказує наскільки конкретному користувачеві подобається певний елемент. Корисність може бути довільною функцією і залежати від завдання. Наприклад, елемент буде кориснішим, якщо він збільшує задоволення користувачів або краще підкріплює потреби користувача. Залежно від постановки, оцінка F може бути вказана користувачем - explicit feedback(як це часто робиться в контексті заповнених користувачами рейтингів), або обчислюється програмно - implicit feedback (значення бінаризовані, приймають 0 або 1).

Зважаючи на релевантність, строго, проблема рекомендації може бути сформульована таким чином:

- Нехай U будете набором всіх користувачів і нехай I буде набором усіх елементів. Обидва ці простори можуть бути дуже великими - аж до потенційно мільйонів предметів.
- Нехай F - функція важливості (корисності), яка вимірює відповідність елемента i до користувача u наступним чином: $f : U \times I \rightarrow R$, де R - це упорядкований набір налаштувань користувачів для елементів.
- Для кожного користувача $u \in U$ ми хочемо вибрати елемент $i \in I$, який максимально збільшує задоволеність користувача.

Тобто, ми хочимо вирішити наступну задачу оптимізації:

$$\forall u \in U, i_s = argmax_{i \in I} F(u, i) \tag{1.2}$$

де кожен елемент із простору користувачів U можна визначити за допомогою профілю, який включає різні характеристики, такі як ідентифікатор користувача, вік, стать, дохід тощо. Аналогічно, кожен елемент простору елемента

I визначається за допомогою (зокрема) набору характеристик. Наприклад, у завданнях музичної рекомендації - Spotify, SoundCloud, Pandora, де I є колекцією пісень, кожна пісня може бути представлена не лише за її ідентифікатором, але і за назвою, жанром, виконавцем, роком випуску тощо.

Центральна проблема, з якою стикається будь-яка система рекомендацій, полягає в тому, що функція F зазвичай не визначається у всьому просторі $U \times I$ - вона, в кращому випадку, визначається лише на підмножині цього простору. Це означає, що F потрібно екстраполювати на весь простір $U \times I$. У системах рекомендацій, функція F зазвичай представлена рейтингом і спочатку визначається лише на елементах, які раніше оцінювали (існуючі) користувачі. Наприклад, у програмі рекомендацій книг, як PlayBooks, користувачі спочатку оцінюють деяку підмножину книг, які вони вже читали. Мета програми - передбачити рейтинг до книг які не відомі користувачу, та відобразити відповідні рекомендації на основі цих прогнозів.

Висновок

У розділі розглянуто постановку задачі системи рекомендацій. Наведено типові сфери використання, користь бізнесу і цілі. На основі розглянутої літератури було розглянуто класифікацію систем відносно підходу до побудови рекомендацій. Проаналізовано проблематику напрямку.

2 ПІДХІД ГЛИБОКОГО НАВЧАННЯ У ПОБУДОВІ РЕКОМЕНДАЦІЙ

Моделі і алгоритми глибокого навчання на основі штучних нейронних мереж в останні роки знаходять своє використання у багатьох прикладних сферах інформатики. Цей підхід також показав прекрасні результати у порівнянні із класичними моделями у задачах побудови рекомендацій,що є наслідком уміння нейромереж знаходитити не лінійні і не тривіальні зв'язки у навчальних даних. А також, можливість використання в якості джерела даних як візуальну, так і текстову, контекстну інформацію. Розглянемо класичну структуру нейронної мережі.

2.1 Нейронна мережа прямого поширення

Перцептрон - математична модель яка відтворює сприйняття інформації за подобою мозку людини.Елементарний перцептрон складається з трьох типів елементів: S-елементів, A-елементів та одного R-елемента (Рис 3.1). S - це шар датчиків або рецепторів. У фізичному варіанті вони відповідають, наприклад, до схожих чутливих клітин сітківки ока або фоторезисторів матриці фотокамери. Кожен рецептор може бути в одному з двох станів - спокою або збудження, і лише в останньому випадку він передає сигнал до наступного шару і іншим повязаним елементам.

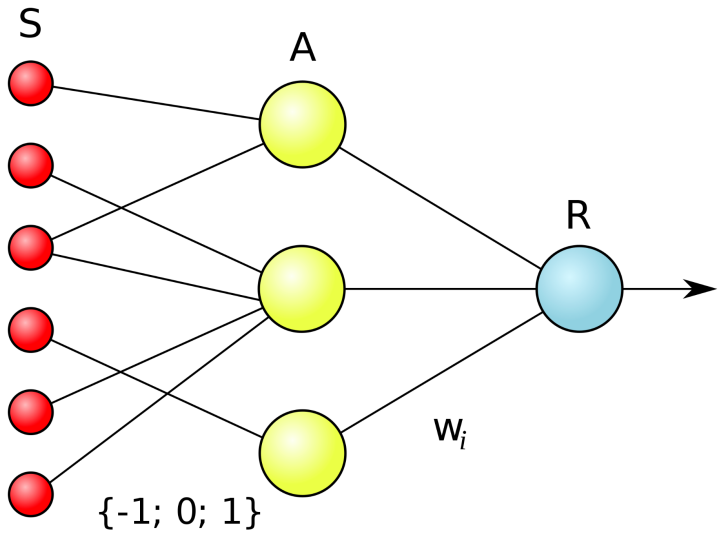


Рисунок 2.1: Логічна схема елементарного перспетрону.

A-елементи називаються асоціативними, оскільки кожен такий елемент, як правило, відповідає цілому набору (асоціацій) S-елементів. A-елемент активується, як тільки кількість сигналів від S-елементів на його вході перевищила певне значення θ.

Сигнали від збуджених A-елементів, у свою чергу, передаються на сумator R, кожен сигнал із i-того елемента передається із коефіцієнтом w_i. Цей коефіцієнт називається вагою зв'язків A-R.

R-елемент обчислює суму значень вхідних сигналів, помножених на вагу (лінійна форма). R-елемент, а разом з ним і елементарний перцептрон, видає 1, якщо сума перевищує поріг.

Навчання елементарного перцептрона полягає у зміні коефіцієнтів ваг w_i звя'зків A-R. Ваги з'єднань S-A (які можуть приймати значення $[-1; 0; +1]$) та значення порогів A-елементів вибираються випадковим чином на самому початку, а потім не змінюють.

Багатошарова нейронна мережа - це нейронна мережа, що складається з вхідного шару, виходу та розташованих прихованих шарів нейронів (Рис. 3.2). Такі мережі мають значно більші можливості, ніж персептрон, однак методи навчання нейронів прихованого шару були розроблені порівняно недавно.

Розглянемо це перетворення на прикладі лінійної регресії. Візьмемо X за вхідний вектор, зміщення b, вектор вагів W і оцінку виходу $\hat{y}$.

$$\hat{y} = b + W^T X \tag{2.1}$$

Мінімізуючи функцію втрати, наприклад, MSE ми шукаємо оптимальне рішення.По мірі навчання будее зменшуватись різниця між нашою оцінкою $\hat{y}$ та реальним значенням y. Іншими словами, неоптимальні параметри призводять до більших втрат, ніж оптимальні параметри. У нейронній мережі прямого поширення ми використовуємо деяку функцію активації a:

$$\hat{y} = a(b + W^T X) \tag{2.2}$$

Нелінійність надає моделі більшу гнучкість, шукаючи глобальний оптимум. Для ефективної роботи моделі нам потрібні ітераційні методи оптимізації.

Одним із найефективніших методів на сьогоднішній день є градієнтний спуск. Глибокі нейронні мережі (DNN) зазвичай виконують міні-пакетний

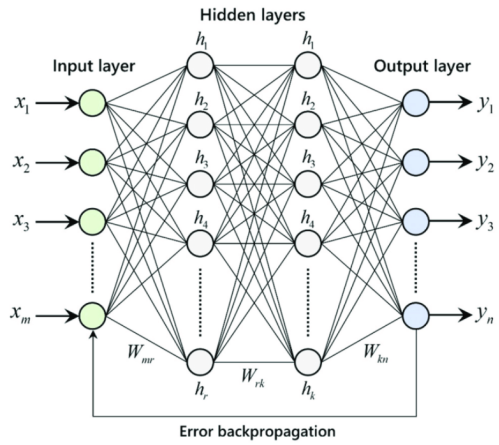


Рисунок 2.2: Багатошарова нейронна мережа

стохастичний спуск. Цей метод розбиває повну партію тренувань на частини (спліти), які збільшують частоту оновлення порівняно із навчанням на усіх даних (усіма навчальними зразками).

Алгоритим складається із прямого(forward propagation) і зворотнього проходу(backward propagation). На зворотньому проході ми оновлюємо наші параметри (ваги і зміщення) пропорційно обраному коефіцієнту α.

2.2 Автоенкодер

Генеративні моделі знайшли своє використання у задачах комп'ютерного зору та обробки природної мови.

Але останні дослідження показали що їх використання не обмежується тільки вищезгаданими сферами. У контексті побудови рекомендацій автоенкодери знайшли рішення групі важливих завдань.

До побудови рекомендацій можливо підійти як до завдання великих даних,внаслідок надзвичайно великого обє'му даних які генерються користувачами (так званий інформаційний слід). Але, із іншої сторони, ці дані в більшості випадків є ненасичені (користувачі взаємодіють лише із мізерною кількістю об'єктів). Що ускладнює завдання вивчення і передбачування побажань усіх користувачів. Для того щоб ефективно інтерпритувати розріджені сигнали було представлено ймовірносну модель засновану на Баєсовій статистиці яка здатна навчатись і поширюватись на скудно насичених даних.

Автоенкодер у своєму класичному виді - це група із двох моделей глибоких нейронних мереж які використовуються для задачі кодування і декодування

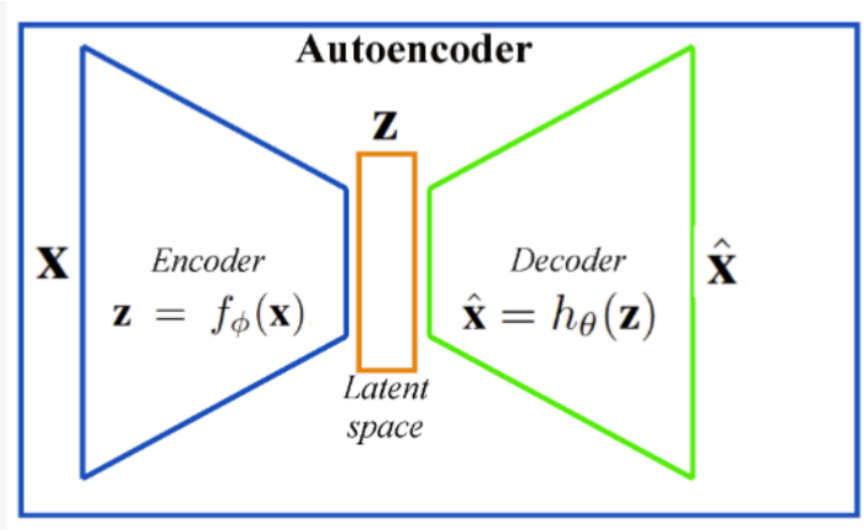


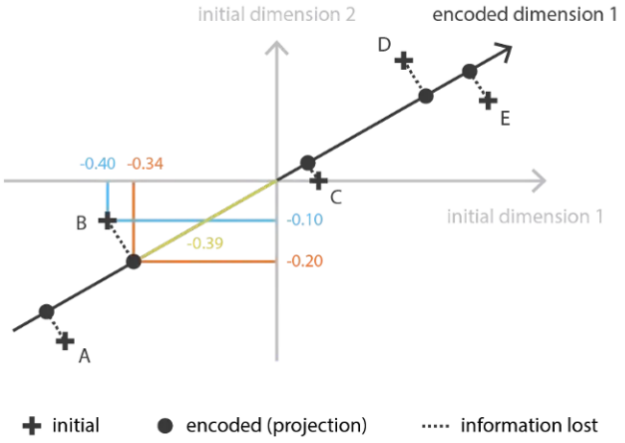
Рисунок 2.3: Архітектура нейронної мережі автоенкодера.

вхідних даних (Рис. 3.2). Це кодування можна сприймати як один із способів зниження розмірності і комплексності вхідного датасету. Що, спрощує структуру моделей і полегшує швидкість навчання череез меншу кількість гіперпараметрів. Для низькорівневого умовного прикладу автоенкодера можна використати алгоритм аналізу головних компонент (PCA). Ідея якого, це побудова нових признаків більш низької розмірності на основі лінійних комбінацій векторів базавого набору. Під час пошуку найкращої лінійної підмножини алгоритм мінімізує втрати інформації (Рис 3.3).

Але алгоритми такого класу суттєво обмежуються вимогами до не лінійності навчального набору, якщо дані лінійно не залежні алгоритм не знайде їх відображення - тому що його банально не існує. Внаслідок чого, були створені алгоритми де у якості енкодера і декодера використовуютсья нейронні мережі для пошуку найкращої функції ітераційним процесом оптимізації. Такий підхід також дозволяє не тільки ефективно кодувати простір, а і зберегти під час стискуваня важливу структурну інформацію.

Завдання навчання автоенкодера у класичному виді можна поставити наступним чином. Нехай $\mathbb{X} \in \mathbb{R}^m$ і $\mathbb{Z} \in \mathbb{R}^n$ є множини декодованих і закодованих об'єктів відповідно. А $E_\phi : \mathcal{X} \to \mathcal{Z}$ і $\mathcal{D}_\theta : \mathcal{Z} \to \mathcal{X}$ деякі сімейства функцій енкодера і декодера пареметризовані ϕ і θ відповідно. Кожен $x \in \mathcal{X}$ вважається закодованим об'єктом при $z = \mathbb{E}_\phi(x)$ і декодованим (відновленим) при $z \in \mathcal{Z}, x' = \mathbb{D}_\theta(z)$.

Також введемо функцію $d : \mathcal{X} \times \mathcal{X} \to [0, \infty]$ таку, що $d(x, x')$ вимірює наскільки x' відрізняється від x.



Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

Рисунок 2.4: PCA шукає найкраще відображення вхідних даних у лінійний підпростір меншого розміру

Функція втрат автоенкодера наступна:

$$L(\theta, \phi) := \mathbb{E}_{x \sim \mu ref}[d(x, D_\theta(E_\phi(x)))] \tag{2.3}$$

Тоді оптимізацію градієнтним спуском $\arg\min_{\theta,\phi} L(\theta, \phi)$ будемо називати як навчання автоенкодера. А у випадку коли функція d є евклідовою нормою (L2) функція втрат приймає наступний вигляд:

$$\min_{\theta,\phi} L(\theta, \phi), \text{where } L(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - D_\theta(E_\phi(x_i))\|_2^2 \tag{2.4}$$

У практичних реалізаціях використовують стандартний багатошаровий персептрон (MLP). Принцип його навчання є загально відомий і детального розгляду не потребує. У експериментальній частині буде наведено довідкову інформацію про практичну реалізацію. Розглянувши класичну структуру автоенкодера ми перейдемо до найбільш успішної його варіації - Варіаційного автоекодера, що показала прекрасні результати і є лідером у багатьох топ-чартах моделей побудови рекомендацій. Такий розвиток є наслідком Баєсового варіаційного підходу, що дає можливість навчатись не тільки на відомих реальних об'єктах а і на згенерованих самою моделею після вивчення і моделювання потрібного розподілу даних.

Також, однією із причині викоистання такого підходу є не регуляризованість простору прихованих змінних автоенкодера. На Рисунку 3.5 показана

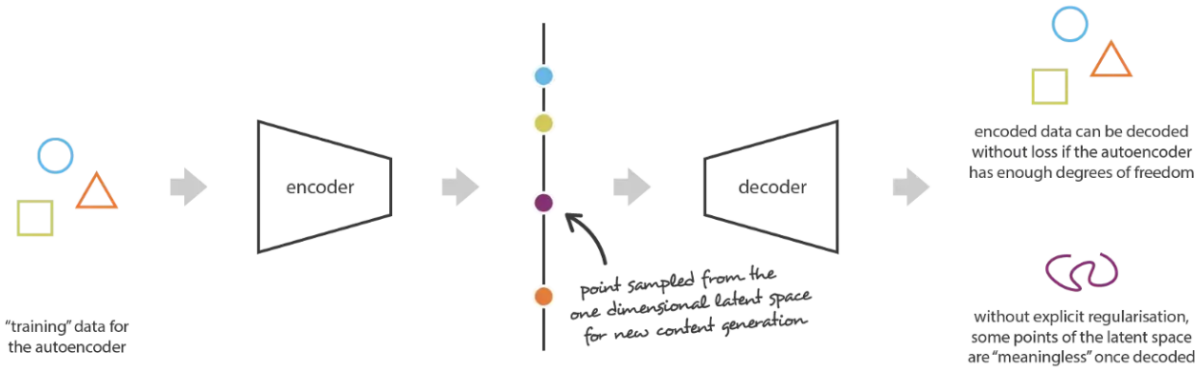


Рисунок 2.5: Не регуляризований простір кодування звичайного автоекодера має проблему при використанні в завданнях генерації нових об'єктів.

інтерпретація цієї проблеми.

Головна особливість варіаційного автоекодера є те, що він вчиться кодувати не вхідний сигнал мережі, а розподіл векторів прихованого простіру(коду).

Отже можна визначити наступні переваги автоенкодерів у завданнях рекомендацій:

- Навідмінно від класичних моделей які можуть використовувати тільки один із джерел даних (оцінки взаємодії або текст) автоенкодери використовують широкий спектр гетерогенних даних таких як: оцінки, аудіо, зображення або відео.
- Автоенкодер через свою нелінійність краще вивчає вподобання користувачів, що призводить до більш високих оцінок метрик якості.
- Автоенкодер адаптивний до багатьох сценаріїв і більш ефективний у випадку боротьби із вхідним шумом у даних.

2.3 Графові нейронні мережі

Графові нейронні мережі - класс нейронних мереж які оперують даними у вигляді графів (Рис. 3.6). У більш широкому розумінні такий клас мереж можна віднести до геометричного глибинного навчання, в якому існуючі архітектури інтерпретують як графові нейронні мережі. Як приклад - у контексті комп'ютерного зору, зображення можна сприймати як граф пікселів, а у випадку обробки мови, графом представляють слова у тексті. До недавного часу більшість моделей векторизації графів були повільні і використовували алгоритми на основі матричної факторизації або спектральної декомпозиції

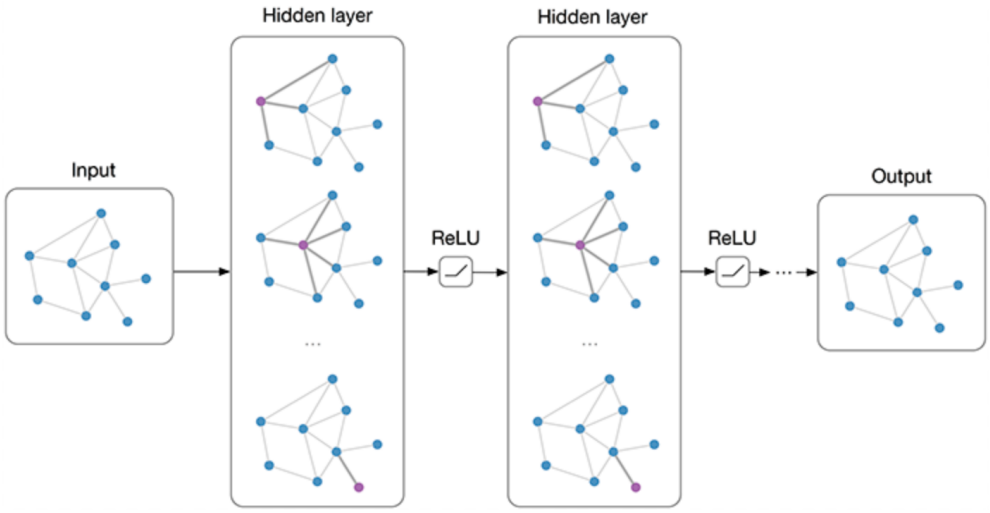


Рисунок 2.6: GNN - тип нейронних мереж які напряму працюють із структурою графу.

графів. Однак, додатково вони мали обмеження у використанні, при побудові графу, додаткової інформації, яка могла зберігатись у ребрах або вершинах. Поява графових нейронних мереж стала логічним подальшим розвитком досліджень у сфері побудови графових моделей і дозволило уніфікувати в один підхід минулі розробки.

В основі GNN лежить використання механізму передачі повідомлень (Message passing). Граф оброблюється набором модулів, які повязані між собою у відповідності до вузлів графу. Також, кожен із модулів пов'язаний із самими вузлами графа. Під час процесу навчання, модулі обновлюють свої стани і обмінюються інформаціюю. Цей процес відбувається до моменту, поки модулі не перейдуть у стан рівноваги. Вихід GNN обчислюється на базі значень модулів кожного вузла графу.

Нехай, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ вхідний граф із набором признаків вершин $X \in \mathbb{R}^{d \times |\mathcal{V}|}$. І стоїть завдання побудувати латентне відображення h_u для кожної $u \in \mathcal{V}$. На кожній ітерації GNN, відображення h_u оновлюється агрегацією сигналів із суміжних вершин $\mathcal{N}_u$ (Рис. 3.7):

$$h_u^{k+1} = f_1^k(h_u^k, f_2^k(h_v^k, \forall v \in \mathcal{N}(u))) \tag{2.5}$$

де f_1 і f_2 відповідають за операцію оновлення і агрегації, які підбирають в залежності від ситуації. Повідомленням у графових нейронних мережах принято називати:

$$m_{\mathcal{N}(u)} = f_2^k(h_v^k, \forall v \in \mathcal{N}(u)) \tag{2.6}$$

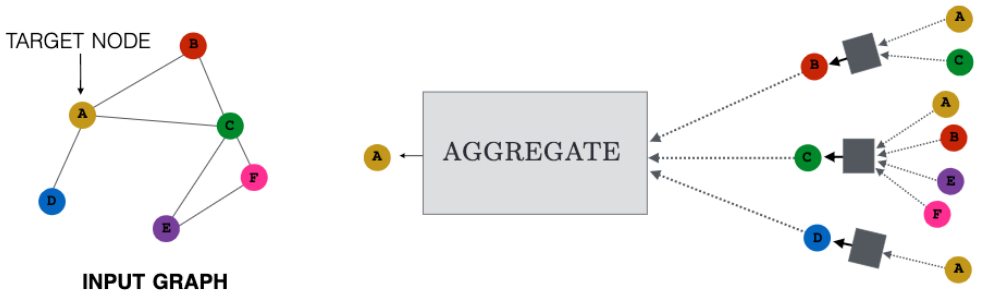


Рисунок 2.7: Ілюстрація агрегації сигналів сусідів для однієї вершини графу. Значення агрегації усереднюються.

На кожній ітерації k, GNN f_2 - функція агрегації приймає на вхід набір прихованих значень сусідів, а функція f_1 оновлює ембединг об'єкту використовуючи повідомлення $m_{\mathcal{N}(u)}$ і значення із попередньої ітерації h_k^u. Після K ітерацій message passing, ми отримуємо фінальні значення для кожної вершини на вихідному шарі:

$$z_u = h_u^K, \forall u \in \mathcal{V} \tag{2.7}$$

Під ітерацією k можна розуміти глибину передачі повідомлень, чим він більший, тим більша відстань від u до вершин які агрегуються. Такий підхід має ціль зібрати інформацію про структуру сусідніх елементів, а також про значення признаків їх вершин.

У разі використання моделі персептрону Рів. 2.6 приймає наступний вигляд:

$$h_u^k = \sigma\left(W_{self}^k h_u^{k-1} + W_{neigh}^k \sum_{v \in \mathcal{N}(u)} h_v^{k-1} + b^k\right) \tag{2.8}$$

де W_{self}^k і W_{neigh}^k матриці вагів мережі, а σ не лінійна функція активації.

Графові нейронні мережі у задачах побудови рекомендацій знайшли своє використання у таких завданнях:

- **Рекомендація послідовностей (Sequential Recommendation)**. Аналізуючи послідовність дій користувачів, такі моделі хочуть на основі деякого ланцюга подій $x_1, x_2, ..., x_n$ довжини n передбачити x_{n+1} об'єкт (Рис 3.8).
- **Рекомедація на основі сесії**. Одна із найскладніших завдань. У моделі стоїть ціль передбачити x_{n+1} об'єкт на основі активної сесії користувача, тобто в режимі реального часу.

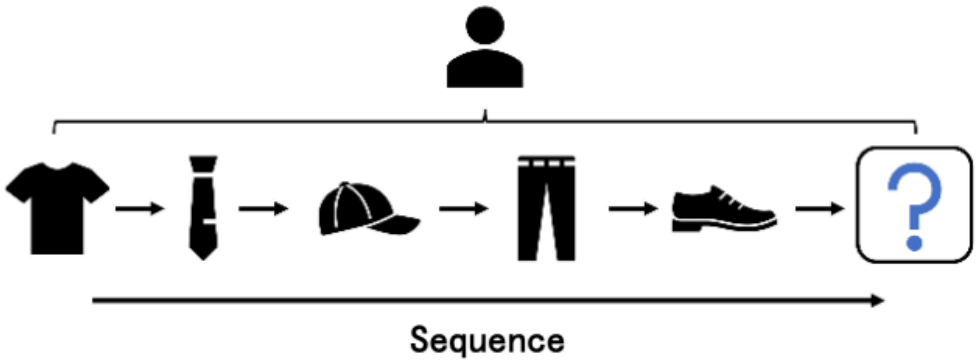


Рисунок 2.8: Ілюстрація завдання передбачення послідовності.

- **Соціальні рекомендації**. Великі соціальні мережі використовують моделі для передбачування поведінки і побажань своїх користувачів. В майбутньому ці дані можуть бути використані для продажу або реклами (наприклад Facebook Ads).

Висновок

У розділі розглянуто принцип роботи нейронної мережі прямого поширення на прикладі персептрону. Проаналізовано аріхтектури нейромереж на основі автоекодера і графової нейронної мережі. Розглянуто ключові деталі їх роботи: алгоритм навчання автоенкодера і механізм передачі повідомлень. Розглянуто принцип роботи і основні переваги використання вищезгаданих архітектур у завданнях побудови рекомендацій.

3 МЕТРИКИ ОЦІНКИ ЯКОСТІ

Система рекомендацій, як інструмент, повинна виконувати поставлені їй задачі. Як приклад візьмемо задачі із області е-коммерції, а саме ключові показники продуктивності бізнесу (Рис. 4.1):

- CR(Conversion Rate)
- CTR
- LT(Life Time)
- LTV(LifeTime Value)
- Retention
- User Egagement

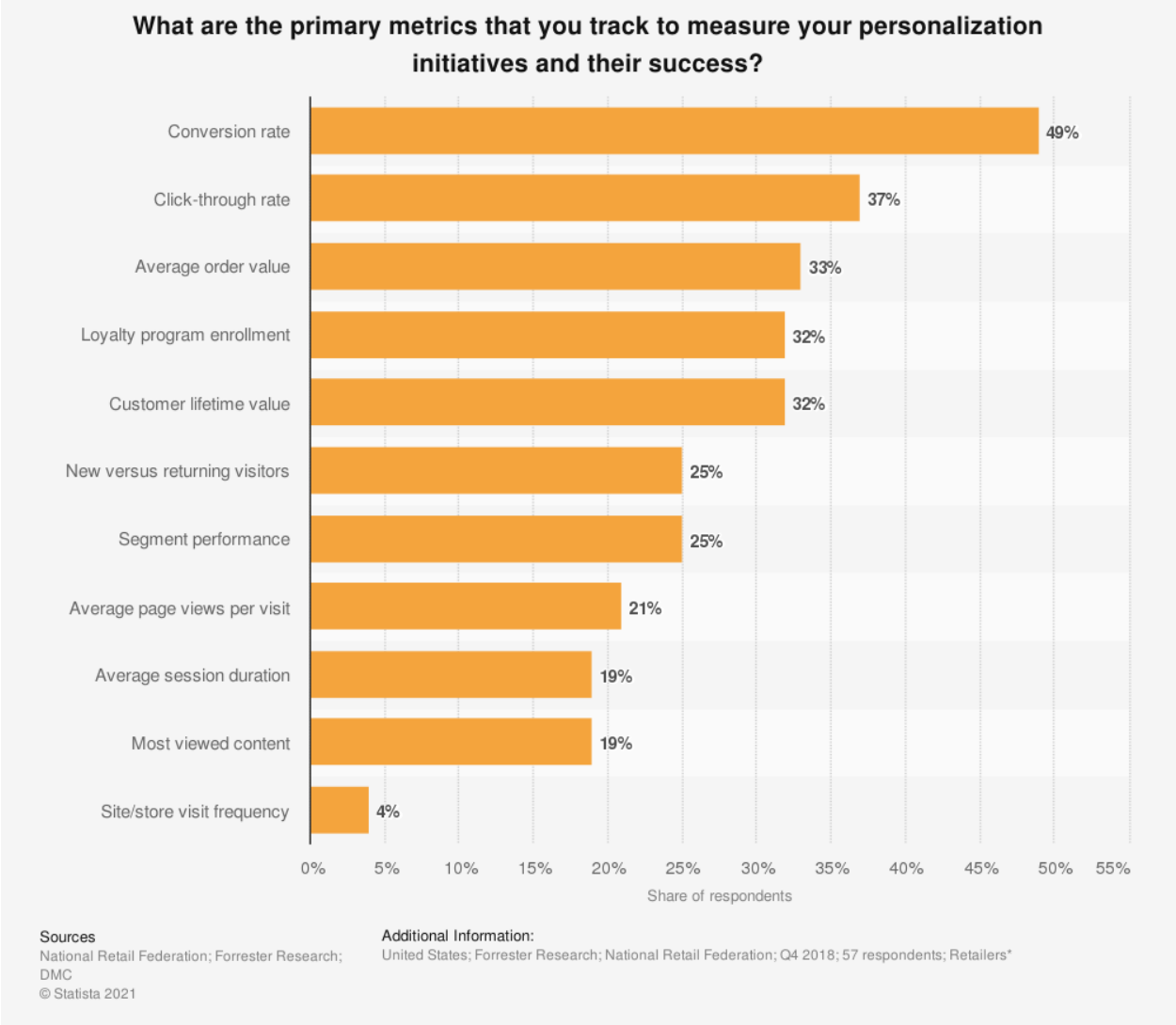


Рисунок 3.1: Рейтинг популярності комерційних метрик

Найкращий спосіб визначити, чи покращує RS ці показники - провести A/B тест за допомогою відгуків користувачів. Однак A/B тест, як правило,

є дорогим і довготривалим у імплементації. Тому, перш ніж провести тест A/B, бажано застосувати дешевший та швидший метод тестування в режимі офлайн, щоб зменшити можливість значного падіння якості наданих рекомендацій.

Це призводить до розробки різних підходів, таких як тестування на згенерованих даних та офлайн-оцінки, що є стандартним способом порівняння продуктивності різних моделей. Важливо підтримувати послідовність та рівномірність на всіх етапах, включаючи метричний розрахунок.

3.1 Класифікація метрик систем рекомендацій

В силу специфіки нашої задачі, метрики системи можна умовно поділити на групи взалежності від підходу - **Метрики Класифікації**, **Метрики Ранжування**, **Метрики Новизни і різноманіття**. Введемо означення для опису метрик:

- u ідентифікатор користувача
- i ідентифікатор об'єкта рекомендацій
- rec_k список рекомендацій для користувача u із з top-k обєктів рекомендацій
- $rel(u)$ список релевантних обєктів для користувача u із тестового набору
- $rank(u, i)$ позиція і обєкту у списку рекомендацій $rec_k(u)$
- $\mathbb{I}[\cdot]$ індикаторна функція

3.2 Метрики класифікації

Метрики класифікації оцінюють придатність до прийняття рішень систем рекомендацій. Вони є хорошим вибором для завдань визначення важливих або нерелевантних продуктів для користувача.

preference	recommended	Not recommended
Like	True-Positive TP	True-Negative TN
Not like	False-Positive FP	False-Negative FN

Рисунок 3.2: Матриця помилок

3.2.1 Precision and Recall

Метрики які широко використовуються у задачах класифікації знайшли своє використання і тут (Рис. 3.2):

$$Precision@k(u) = \frac{|rel(u) \cap rec(u)|}{k} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall@k(u) = \frac{|rel(u) \cap rec_k(u)|}{|rel(u)|} \tag{3.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

Precision можна інтерпретувати як частку об'єктів, що класифіковані як вірні та в той же час є вірними, і Recall показує, яка частка позитивного класу від всіх об'єктів вірного класу виявив алгоритм.

3.2.2 F1-Score

F1@K - це гармонічне середнє значення Precision@K і Recall@K, що допомагає спростити їх в одну метрику. Всі вищезазначені показники можна обчислити на основі матриці помилок(Рис. 3.2). Точна формула наведена нижче:

$$F1@k = 2 \times \frac{Precision@k \times Recall@k}{Precision@k + Recall@k} \tag{3.5}$$

Як ми бачимо, коефіцієнт F1 не враховує TN(True Negative). Це випадки, коли система рекомендацій не рекомендувала товар, який не має значення для користувача. Тобто метрика "сліпа"відносно певної групи обє'ктів рекомендацій. Цікавою та ідеально симетричною альтернативою є коефіцієнт кореляції Метьюса (MCC).

3.2.3 Matthews correlation coefficient (MCC)

Коефіцієнт кореляції Метьюса - це коефіцієнт кореляції між спостережуваною та прогнозованою бінарною класифікацією:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \quad (3.6)$$

Коли класифікатор ідеальний (FP = FN = 0) значення MCC становить 1, що вказує на ідеальну позитивну кореляцію.I навпаки, коли класифікатор завжди неправильно класифікує (TP = TN = 0), ми отримуємо значення -1, що представляє ідеальну негативну кореляцію.

3.2.4 Hit Rate

Приймає значення 1, коли хоч одна із рекомендацій є релевантною.

$$HitRate@k(u) = \mathbb{I}[|rel(u) \cap rec_k(u)| > 0] \quad (3.7)$$

3.2.5 Mean Average Precision

Середня відносна точність(MAP) усереднюється для користувачів, і всі відмінності трапляються в розрахунку середньої точності.

$$AP@k(u) = \frac{1}{x} \sum_{i \in rec_k(u)} \mathbb{I}[i \in rel(u)] Precision@rank(u, i)(u) \quad (3.8)$$

3.2.6 RocAuc

Крива ROC (характеристика операційної кривої приймача) - це графік, що показує продуктивність моделі класифікації на всіх порогах класифікації (Рис. 4.3). Метрика використовує наступні складові:

- TPR - True Positive Rate

$$TPR = \frac{TP}{TP + FN} \quad (3.9)$$

- FPR - False Positive Rate

$$FPR = \frac{FP}{TP + FN} \quad (3.10)$$

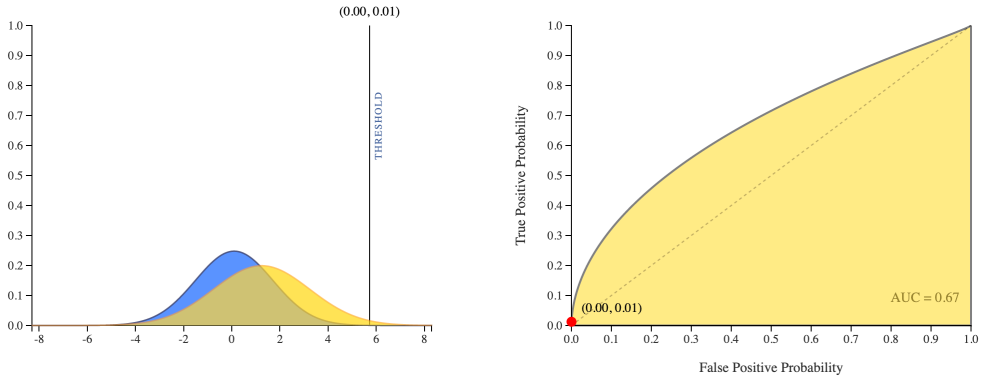


Рисунок 3.3: Умовний приклад метрики ROC AUC

AUC походить від "Area under the ROC Curve". Тобто AUC вимірює всю площу під кривою ROC від (0,0) до (1,1).AUC забезпечує сукупний показник продуктивності у всіх можливих порогах класифікації. Один із способів інтерпретації AUC - це як ймовірність того, що випадковий позитивний приклад буде оцінено більш високо, ніж випадковий негативний приклад. AUC ціниться з наступних причин:

- AUC інваріантний до масштабу. Він вимірює, наскільки добре оцінюються прогнози, а не їх абсолютні значення.
- AUC інваріантний до порогу чутливості. Він вимірює якість прогнозів моделі незалежно від того, який поріг класифікації обраний.

3.3 Метрики ранжування

Метрики ранжування використовуются у системах рекомендації через призму підходу як до задачі інформаційнного пошуку (Information Retrival).

Інформаційний пошук (ІП) в обчислювальній та інформаційній науці - це процес отримання ресурсів інформаційної системи, що відповідають інформаційній потребі збору цих ресурсів. Пошуки можуть базуватися на повнотекстовій або іншій індексації та на основі вмісту. Отримання інформації - це наука пошуку інформації в документі, пошук самих документів, а також пошук таких метаданих, які описують дані, а також для баз даних текстів, зображень чи звуків.

Автоматизовані системи пошуку інформації використовуються для зменшення того, що називається перевантаженням інформації. ІП-система - це програмна система, яка забезпечує доступ до книг, журналів та інших до-

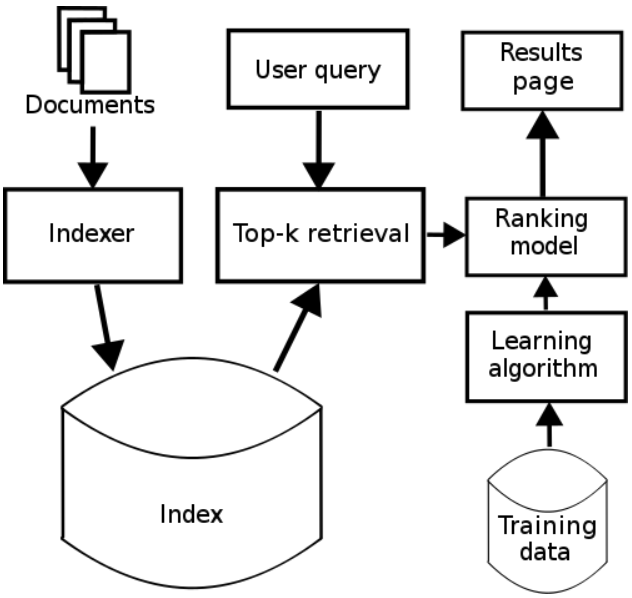


Рисунок 3.4: Типова архітектура системи інформаційного пошуку

кументів; зберігає та керує цими документами. Веб-пошукові системи - це найбільш помітні ІП-програми.

3.3.1 Сукупний інформаційний приріст

(Discounted Cumulative Gain) - це показник рейтингу, який може прийняти довільні значення релевантності. У ІП він часто використовується для вимірювання ефективності алгоритмів веб-пошукових систем або пов'язаних з ними додатків. Використовуючи метрику оцінки релевантності документів у наборі результатів пошуку, DCG вимірює корисність або її приріст, на основі позиції елементів у списку результатів. Чим вищий ранг обєкту у списку реезультатів тим більше підсилюється його релевантність, а для нижчих зменшується.

Для використання метрики нам потрібно пряйняти наступні припущення:

- Існування деякої міри релевантності для елементів набору.
- Елементи із більш високою релевантністю приносять більше користі якщо вони знаходятся вище у списку (мають більш високі ранги)

Для штрафування використовуємо log reducing factor. Загальноприйнята формула для DCG:

$$DCG@k(u) = \sum_{i \in rec_k(u)} \frac{2\hat{}rating(u,i) - 1}{log_2(rank(u,i) + 1)} \tag{3.11}$$

DCG обчислюється для одного набору документів (обєктів), для можливості порівняння таких наборів вводять і використовують $nDCG$, де n відповідає за нормалізацію:

$$NDCG@k(u) = \frac{DCG@k(u)}{IDCG@k(u)} \tag{3.12}$$

де $IDCG@k(u)$ (тобто $IdealDCG$) максимально можливе значення $DCG@k(u)$. Його ми можемо отримати у випадку ідеального сортування набору обєктів рекомендацій довжини k. Для цієї мети зазвичай використовуються рейтинги елементів із тестового набору.

Також для бінарних обєктів розглядають наступну варіацію:

$$DCG@k(u) = \sum_{i \in rec_k(u)} \frac{\mathbb{I}[i \in rel(u)]}{log_2(rank(u,i)+1)} \tag{3.13}$$

Можна передбачити наступні обмеження NDCG:

- NDCG не штрафує за низьку релевантність самих обєктів. Якщо оцінки одинакові (наприклад [1, 1, 1] і [1, 1, 1, 0]) значення метрики буде одинакове. Потрібно використовувати нормування із відємним значенням для негативних елементів.
- Також метрика не є ефективною для малих k (набір рекомендації фіксованого розміру може швидко заповнюватись першими n елементами).

3.3.2 Середній взаємний ранг

Mean Reciprocal Rank (MRR) або середній взаємній ранг - це середнє значення зворотньої позиції (рангу) першого релевантного обєкту у списку рекомендацій. Якщо жодних елементів не передбачано правильно, він визначається як 0.

$$MRR@k(u) = \frac{1}{k} \sum_{i=1}^{|k|} \frac{1}{\min\limits_{i \in rel(u) \cap rec_k(u)} rank(u,i)} \tag{3.14}$$

Grade	Satisfaction Probability
0	0
1	1/16
2	3/16
3	7/16
4	15/16

Рисунок 3.5: Приклад розрахунку ймовірності для ERR

3.3.3 Очікуваний взаємний ранг (Expected Reciprocal Rank)

Припустимо, що система повертає користувачу рейтинговий список документів (рекомендацій), де ймовірність $p(q, d_k)$, що документ задовільняє запит користувача:

$$ERR = \sum_{k=1}^{K} \frac{1}{k} p(q, d_k) \prod_{i=1}^{k-1} (1 - p(q, d_i)) \tag{3.15}$$

Вибір обє'кта із певним рангом свідчить що користувач задоволений документом, ймовірність якого $p(q, d_k)$. Також, це свідчить що всі рекомендації із вищим рейтингом $1, ..., k-1$ ймовірність яких $\prod_{i=1}^{k-1}(1 - p(q, d_k))$ не були обрані.

Для моделювання ймовірності що конкретна рекомендація буде обрана використовують так звану "редакційнку оцінку"(editorial grade) - цінність певного обєкту. В залежності від прикладної задачі можна ввести правила основані на схожості (із іншими типовими об'єктами), або передбачуванням моделі. В літературі розглядають 5 оцінок (0, 1, 2, 3, 4), де 0 це не релевантний об'єкт, а 4 дуже релевантний. Тоді ймовірність за "редакційною оцінкою":

$$R(g) := \frac{2^g - 1}{2^{g_{max}}}, g \in \{0, ..., g_{max}\} \tag{3.16}$$

Тобто, переглядаючи результат пошуку на запит із оцінкою 3, є шанс 7/16, що користувач буде задоволений цим документом і, отже, припинить пошук, або із шансом 9/16 пошук продовжится (Рис. 4.5).

- Однією приємною особливістю ERR є те, що вона завжди лежить між 0 до 1, причому 1 - найкраще. Це означає, що втрата внаслідок зіпсування

будь-якого окремого прикладу обмежена.

- Метрика дає можливість зменшити вплив низького рангу на об'єкт. Важливість списку рекомендацій сильно падає при збільшенні його розміру (елемент із рангом 20 має множник 1/20 до своєї релевантності). ERR частково це нівелює.

3.4 Метрики новизни і різноманітності

Новизна та різноманітність є одним із важливих аспектів оцінки систем рекомендацій, і дозволяє більш широко підходити до оцінки кандидатів. Додатково, вказані властивості є надзвичайно корисними як із точки зору бізнесу, так і для самих користувачів.

Прийняття кліків, переглядів або інших дій як докази прихильності користувача до контенту потребує великої долі впевненості, в тому що змодельовані оцінки справді віповідають потребам клієнта. Здебільшого, ми можем тільки надіятись що наші гіпотези будуть доказані. Навчаючи моделі ми оперуємо лише тою частиної інформації, яку змогли зібрати. Здебільшого, потреби користувачів є комплексні, динамічні, залежать від контексту (св'ято, пора року, час доби) і подеколи надзвичайно суперечливі. Наприклад, перегляд мультфільму Ранго можна трактувати як прихильність до мультфільмів, або прихильність до комедій. Тому, передбачення вподобань є складним завданням, у якому не можливо досягнути ідеальної точності.

Одним із варіантів часткового вирішення таких проблем, є підхід у рекомендації надзвичайно різних по своїй природі об'єктів. Різноманіття кандидатів підвищує шанс того, що хоч одна із рекомендацій буде релевантною. Враховуючи невизначеність щодо того, від яких характеристик залежить вподобання користувача, рекомендація фільму кожного жанру, як правило, окупається більше, ніж рекомендація, скажімо, трьох мультфільмів.

Різноманітність та новизна також знаходять мотивацію в бізнесі. Задоволеність клієнтів приносить користь бізнесу у вигляді посилення активності, доходів та лояльності клієнтів. Крім цього, диверсифікація продукту - це відома стратегія зменшення ризику та розширення бізнесу. Більше того, це long tail sell - стратегія отримання прибутку від ринкових ніш, продаючи і отримуючи більш високу норму прибутку на дешевші продукти які важко

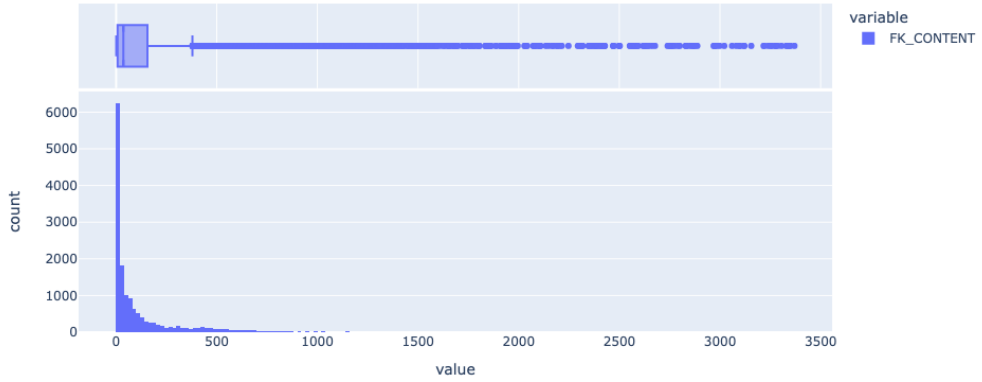


Рисунок 3.6: Типова гістрограма взаємодій користувачів, так званий "long tail"

знайти (Рис. 4.6).

Усі вищезазначені загальні міркування, звичайно, можуть бути замінені конкретними характеристиками конкретної області, ситуації та мети рекомендацій. Наприклад, отримання списку подібних продуктів (наприклад, фотокамер) до одного, який ми зараз вибираємо, може допомогти нам уточнити наш вибір серед великого набору дуже подібних варіантів. Рекомендації можуть слугувати навігаційною допомогою в такого типу ситуації. В інших областях є сенс споживати ті самі або дуже схожі предмети знову і знову, наприклад, покупки продуктів, одяг, тощо.

Новизна та різноманітність суттєво різні, але пов'язані поняття (Рис. 4.7). Новизна, як правило, посилається на різницю між сучасним та минулим досвідом, тоді як різноманітність стосується внутрішніх відмінностей у елементах рекомендацій. Різноманітність, як правило, стосується набору предметів, і має відношення до того, наскільки різні предмети стосовно один одного. В основному, різноманітність оцінюється в наборі елементів, рекомендованих кожному користувачеві окремо, і, як правило, усереднюється у всіх користувачах загалом.

Загалом, варто розглянути такі наступні метрики систем рекомендацій.

3.4.1 Зворотня середня частота (Mean Inverse Item Frequency)

Використовуючи підхід ІП, частоту появи певного об'єкта у наборі взаємодій, можна інтерпретувати як міру популярності цього об'єкту. Ця частота

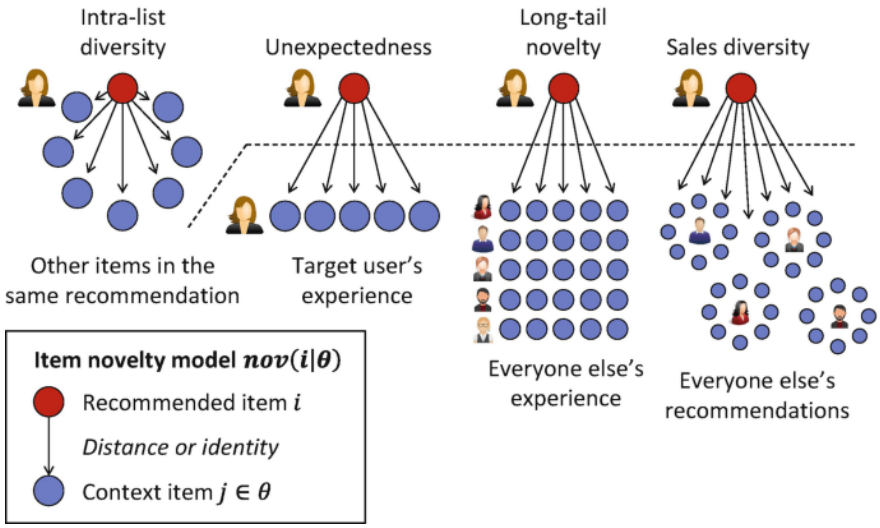


Рисунок 3.7: Новизна і різноманітність у системах рекомендацій

подається у вигляді, знайомому нам із поняття інформаційної ентропії:

$$MIUF@k = -\frac{1}{|k|}\sum_{i \in k} log_2 \frac{|U_i|}{|U|} \tag{3.17}$$

де U_i множина користувачів які знайомів із об'єктом i. У перспективі така новизна на даних із довгим хвостом показує наскільки релевантний об'єкт відносно його популярності.

3.4.2 Несподіваність (Unexpectedness)

Несподіваність розглядають у контексті неочікуваного, але приємного досвіду. Навідмінно від популярності, метрика цілком залежить від вподобань користувача і описує непохожість об'єктів рекомендацій із історичним досвідом користувача.

Існує декілька варіацій Unexpectedness. Простіший варіант оцінює складність об'єктів у наборі. Під складністю розуміють наскільки не похожі об'єкти рекомендацій відносно елементів які користувач із великою ймовірністю вважає релевантними. Для створення порівняльної вибірки використовують тривіальні методи рекомендацій - вибір любимого актора, популярного реежисера, тощо.

$$Unexpectedness@k = \frac{|K \backslash PM|}{|K|} \tag{3.18}$$

У випадку коли у вибірці для порівняння присутні лише не відомі користувачу

об'єкти:

$$Serendipity@k = \frac{|(K \backslash PM) \cap Rel|}{|K|} \tag{3.19}$$

де REL - це набір елементів які вважаються релевантними.

3.4.3 Новизна (Temporal Novelity)

Сприйняття користувача новизни також може розглядатися в межах взаємодії користувача з системою протягом деякого періоду. У цьому випадку ми визначаємо новизну як здатність систем рекомендацій не повторюватися, надаючи однакові або подібні рекомендації із часом. Ця перспектива оцінює здатність системи рекомендації генерувати нові знання про користувача та показує степінь адаптації рекомендацій до них.

$$TN(R_u^t) = \frac{|R_u^t \bigcup_{\tau<t} R_u^\tau|}{|R_u^t|} \tag{3.20}$$

Де R_u^t список рекомендацій у момент t.

3.4.4 Різноманіття (Intra-List Diversity)

Метрика оцінює наскільки відрізняються групи рекомендацій у користувачів. Ця оцінка є однією з найбільш вивчених у літературі, і стосується вирішення потреби користувачів у різноманітних рекомендаціях, уникаючи зайвих або монотематичних пропозицій. Метрика задається як середня парна дистанція між векторами об'єктів рекомендацій:

$$ILD(R) = \frac{1}{|R|(|R|-1)} \sum_{i,j \in R_u} ssim(i,j) \tag{3.21}$$

Де у якості ssim може виступати одна із багатьох відомих метрик відстані між векторами.

Висновок

На основі розглянутої літератури було відібрано і проаналізовано метрики систем рекомендацій. В результаті дослідження, метрики були поділені на 3

категорії в залежності від класу - метрики класифікації, метрики ранжування, метрики новизни і різноманіття. Для кожного класу наведено оцінки і їх опис.

4 АНАЛІЗ АЛГОРИТМІВ РЕКОМЕНДАЦІЙ

Моделі і алгоритми глибокого навчання на основі штучних нейронних мереж в останні роки знаходять своє використання у багатьох прикладних сферах інформатики. Цей підхід також показав прекрасні результати у порівнянні із класичними моделями у задачах побудови рекомендацій, що є наслідком уміння нейромереж знаходитити не лінійні і не тривіальні зв'язки у навчальних даних. А також, можливість використання в якості джерела даних як візуальну, так і текстову, контекстну інформацію.

Algo	MAP	nDCG@k	Precision@k	Recall@k	RMSE	MAE	R^2
ALS	0.004732	0.044239	0.048462	0.017796	0.965038	0.753001	0.255647
BiVAE	0.146126	0.475077	0.411771	0.219145	N/A	N/A	N/A
BPR	0.132478	0.441997	0.388229	0.212522	N/A	N/A	N/A
FastAI	0.025503	0.147866	0.130329	0.053824	0.943084	0.744337	0.285308
LightGCN	0.088526	0.419846	0.379626	0.144336	N/A	N/A	N/A
NCF	0.107720	0.396118	0.347296	0.180775	N/A	N/A	N/A
SAR	0.110591	0.382461	0.330753	0.176385	1.253805	1.048484	-0.569363
SVD	0.012873	0.095930	0.091198	0.032783	0.938681	0.742690	0.291967

Рисунок 4.1: Порівняльні результати роботи моделей на основі нейромереж

Для практичних експериментів було обрано актуальні алгоритми рекомендацій які були представлені у останні декілька років. Із інших критеріїв було взято до уваги популярність у науковій сфері і використання у практичних реалізаціях.

Також, кожна розглянута модель відповідає певній групі алгоритмів для нейромережевої побудови рекомендацій.

4.1 Алгоритм Neural Collaborative Filtering

Алгоритм нейронної факторизації NCF є потужним поєднанням алгоритму нейронної факторизації і багатошарового персептрону. Такий сплав лінійного і нелінійного підходу показує свою високу єфективність у моделюванні прихованих взаємозв'язків між користувчем і об'єктом.

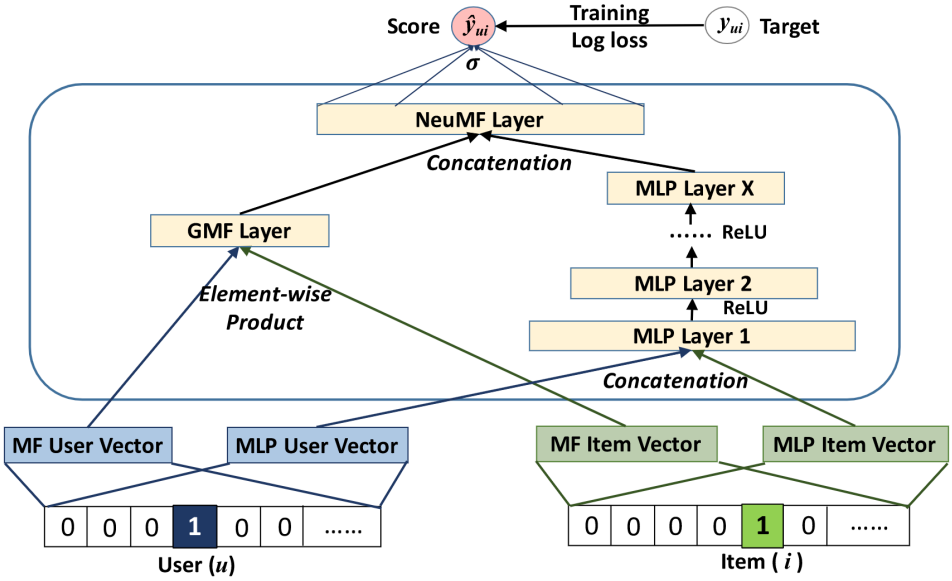


Рисунок 4.2: Умовна будова NCF

У звичайній матричній факторизації усі елементи знаходяться у одному прихованому просторі, що надає нам можливість визначити схожість між потрібними нам векторами використовуючи скалярний добуток/подібність косинуса або коефіцієнт Жаккара. Під матричною факторизацією маємо на увазі наступну модель оцінки взаємодії:

$$\hat{y}_{ui} = f(u, i|p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik} \tag{4.1}$$

Але у задачах рекомендацій із неявною взаємодією (impicit feedback) виникають проблеми із обрахуванням схожості внаслідок обмежень.

На вхід моделі подаються ембедінги користувача і об'єкта відповідно. Ці бінаризовані вектори можуть представляти широкий спектр корисної інформації - від вектора взаємодій до зжатих даних про вподобання користувача, історію або додаткову інформацію про об'єкт. Така особливість нівелює проблему холодного старту.

GMF(Generalized Matrix Factorization) - частина моделі яка відповідає за нейронну матричну факторизацію. Вхідні бінаризовані вектори можна інтерпритувати як приховані змінні користувача - p_u і об'єкта відповідно q_i. Вхідний вектор що поступає на вхід:

$$\phi_1(p_u, q_i) = p_u \odot q_i \tag{4.2}$$

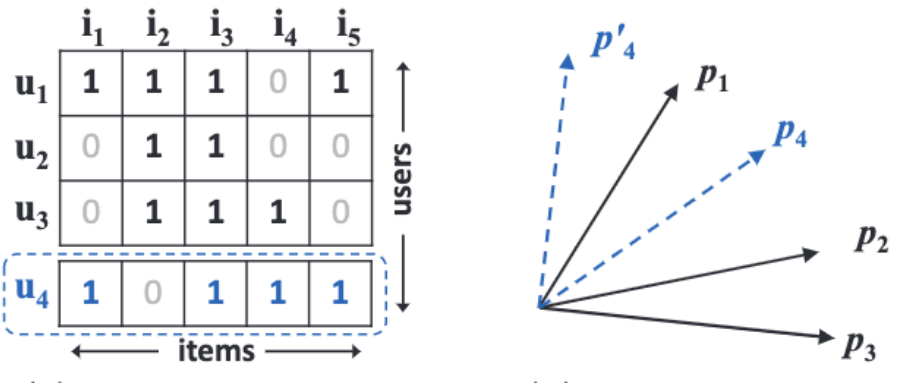


Рисунок 4.3: Зліва - матриця взаємодій користувача/об'єкта.Справа відображення у прихований простір. Положення вектора p4 відносно p3 ілюструє обмеженність. (Схожість p4 до p3 вища ніж до p2 але оптимально виразити ми не можемо).

де $\odot$ відповідає за по елементний добуток. Вихідний вектор GMF:

$$\hat{y}_{ui} = a_{out}(h^T(p_u \odot q_i)) \tag{4.3}$$

де a_{out} відповідають за функцію активації і ваги вихідного шару. При лінійній a_{out} і юніт векторі h відтворюється прямий вихід алгоритму матричної факторизації. Для базової імплеменатації була обрана сигмоїдна функція активації

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.4}$$

та логістична функція втрат.

MLP - використовуєтсья для моделювання не лінійної взаємодії. Це нейронна мережа прямого поширення із вежоподібною структурою (кожен наступний шар має вдвічі меншу розмірність). Формулювання мережі наступне:

$$\mathbf{z}_1 = \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \tag{4.5}$$

$$\phi_2(\mathbf{z}_1) = a_2\left(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2\right), \tag{4.6}$$

$$\cdots\cdots \tag{4.7}$$

$$\phi_L(\mathbf{z}_{L-1}) = a_L\left(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L\right), \tag{4.8}$$

$$\hat{y}_{ui} = \sigma\left(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})\right) \tag{4.9}$$

де W_x, b_x і a_x відповідають матриці вагів, вектору зміщення та функції втрат ReLU. Вибір функції активації обумовлений її властивістю до не насичення і більшою близькою до біологічної структури. Також, відомо що ReLu менш вразлива до перенавчання і є зручної при навчанні на розріджених даних.

Одними із важливих гіперпараметрів моделі є розмір вихідних векторів прихованих змінних. Чим вони більші, тим більш глибоко модель може генералізувати інформації про взаємодію. Тому для імплементації було відібрано декілька їх варіацій.

Фінальним поєднанням двох блоків моделі є шар NeuMF, на якому і обчислюється вихід мережі $\hat{y}_{ui}$. Комбінація результатів MLP і GMF відбувається за рахунок повнозвязного шару на вхід якого подається конкатенація двох векторів прихованих змінних. Використовуючи функцію активації ReLu ми отримуємо таку оцінку:

$$\hat{y}_{ui} = \sigma\left(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}\right) \tag{4.10}$$

тут $\mathbf{h}$ це вектор вагів який задає вплив кожного блоку. Також, він може використовуватись для ініціалізації із препідготовлених вагів. Спочатку навчити із випадковомими вагами, а після досягнення збіжності використати ваги для фінального навчання:

$$\mathbf{h} \leftarrow \begin{bmatrix} \alpha\mathbf{h}^{GMF} \\ (1-\alpha)\mathbf{h}^{MLP} \end{bmatrix}$$

До моделювання функції втрат мережі потрібно підходити як до задачі бінарної класифікації (через бінаризацію вхідних даних і implicit формулювання). Тому функції втрат регресії не підходять (MSE подібні). Значення прогнозувань моделі $y_{ui} = 1$ інтерпритуємо, що обє'кт і важливий для u, а при 0, що ні.

Тому ми обмежуємо значення y_{ui} інтервалі від [0, 1]. Додатково, це обмеження дозволяє використати ймовірносний підхід до інтерпритації. Тоді фун-

кція правдоподібності приймає наступний вигляд:

$$p(\mathcal{Y}, \mathcal{Y}^- \mid \mathbf{P}, \mathbf{Q}, \Theta_f) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{ui} \prod_{(u,j)\in\mathcal{Y}^-} (1 - \hat{y}_{uj}) \tag{4.11}$$

де $P \in \mathbb{R}^{M\times K}$ і $Q \in \mathbb{R}^{N\times K}$ відповідає за матриці прихованих змінних, θ_f за параметри моделі функції взаємозв'язків f. Наступну функцію потрібно мінімізувати використовуючи одим із бажаних варіантів градієнтного спуску:

$$L = -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log (1 - \hat{y}_{ui}) \,. \tag{4.12}$$

4.2 Алгоритм VAE

Для кожного користувача u семплюється вектор z_u розмірності K із багатовимірного нормального розподілу . K відповіє простору прихованих змінних і вміщує у собі інформацію про ключові структурні особливості об'єкта (у простому випадку для цифр це може бути їх колір, кут нахилу або ширина). Для кожного вектора прихованих змінних нелінійна функція $f_\theta(.) \in \mathbb{R}^I$ надає оцінку щільності розподілу для набору I об'єктів рекомендацій $\pi(z_u)$:

$$z_u \sim \mathcal{N}(0, I_K), \pi(z_u) = softmax\{f_\theta(z_u)\}, x_u \sim Mult(N_u, \pi(z_u)) \tag{4.13}$$

У якості нелінійної функції виступає багатошаровий персептрон із параметрами θ. Вихід цієї трансформації нормалізується функцією softmax і повертає ймовірносний вектор $\pi(z_u)$ який містить значення для кожного об'єкту рекомендацій. Припускаємо що вектор x_u належить поліноміальному розподілу із ймовірністю $\pi(z_u)$. Функція правдоподібності для користувача u відносно його прихованого відображення, нагороджує модель збільшуючи ймовірність ненульових елементів x_u:

$$\log p_\theta(x_u|z_u) = \sum_i x_{ui} \log \pi_i(z_u) \tag{4.14}$$

Але через обмеженість $\pi(z_u)$, який сумується до одиниці, модель змушена підвищувати ймовірність на об'єктах які більш правдоподібно будуть обрані

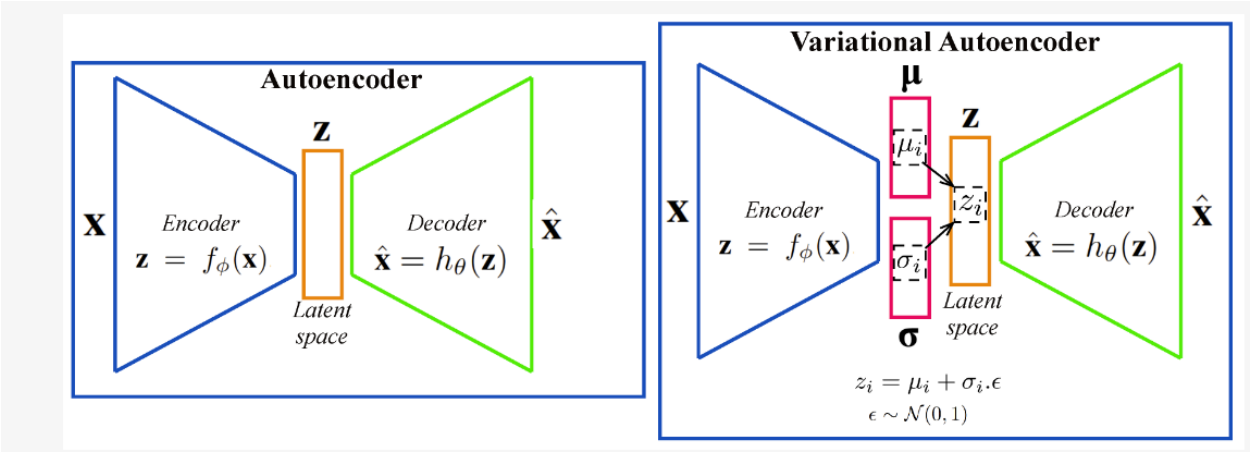


Рисунок 4.4: Ілюстративний приклад архітектури AE і VAE.

користувачем. У задачі колаборативної фільтрації розглядають дві функції правдоподібності. Допустим $f_\theta(z_u) \equiv [f_{u1}, \ldots, f_{uI}]^T$ є виходом генеративної функції $f_\theta(\cdot)$:

- Функція правдоподібності Гауса:

$$\log p_\theta(x_u|z_u) = -\sum_i \frac{c_{ui}}{2}(x_{ui} - f_{ui})^2 \tag{4.15}$$

- Логістична функція:

$$\log p_\theta(x_u|z_u) = \sum_i x_{ui} \log \sigma(f_{ui} + (1 - x_{ui} \log(1 - \sigma(f_{ui})))) \tag{4.16}$$

Апроксимація $\log p_\theta(x_u|z_u)$, через його складність, виконується простішим нормальним розподілом $q(z_u) = \mathcal{N}(\mu_u, \sigma_u^2)$ із допомогою методів вараційного аналізу.

Для поняття відстані або відмінності між двома розподілами використовують відстань (дивергенцію) Кульбака-Лейблера. Яка є несиметричної мірою віддаленості один від одного двох ймовірностних розподілів, що задані на спільній множині елементарних подій. Відстань від розподілу Q до P позначається як $D_{KL}(P||Q)$, де P сприймаємо за істинний розподіл, а Q за розподіл який потрібно перевірити. У Теорії Інофрмації використовують інтерпретацію що $D_{KL}(P||Q)$ це кількість втраченої інформації при заміні істини P на розподіл Q:

$$D_{KL}(P||Q) = \int_X p \log \frac{p}{q} \tag{4.17}$$

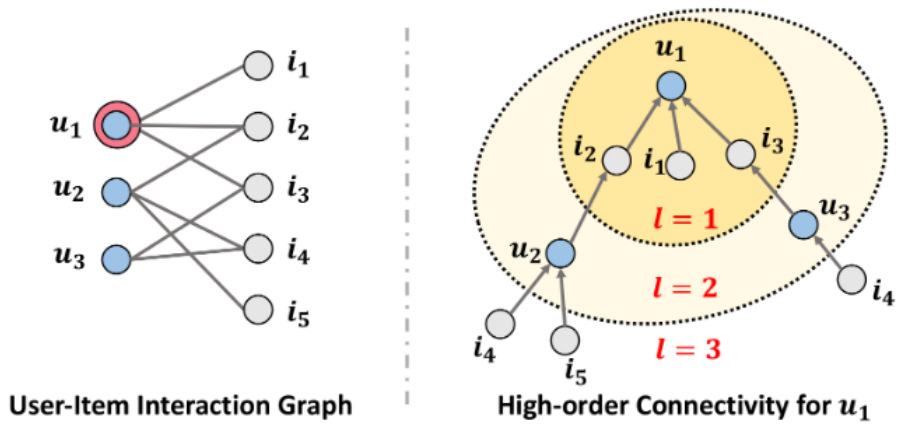


Рисунок 4.5: Приклад графу взаємодій користувач-об'єкт, а також ілюстрація взаємодій вищих порядків для NGCF

де $p = \frac{dP}{d\mu}$ і $q = \frac{dQ}{d\mu}$ неперервні функції із мірою μ на X.

Тому оптимізація варіаційних параметрів $\{\mu_u, \sigma_u^2\}$ відбувається в наслідок мінімізації $KL(q(z_u))||p(z_u|x_u)$, яка є частиною функції втрат:

$$\log p(x_u;\theta) \geq \mathbb{E}_{q_\phi(z_u|x_u)}[\log p_\theta(x_u|z_u)] - KL(q_\phi(z_u|x_u)||p(z_u)) \equiv \mathcal{L}(x_u;\theta,\phi) \tag{4.18}$$

Цей вираз також відомий як розмір нев'язки або нижньої варіаційної границі (ELBO). Для її знаходження градієнтним спуском використовують так званий reparametrization trick який гарантує неперервність яка необхідна для back propagation. Ми семпюємо із деякого нормального розподілу $\epsilon \sim \mathcal{N}(0, I_K)$ і модифікуємо $z_u = \mu_\phi(x_u) + \epsilon \odot \sigma_\phi(x_u)$. Що ізолює процесс генерації від пошуку градієнту напряму.

4.3 Алгоритм Neural Graph Collaborative Filtering

Алгоритм нейромережевої колаборативної фільрації на основі графів використовує підхід інтеграції взаємодій користувач-об'єкт у вигляді дводольного графу. Такий підхід надає можливість моделювання взаємодій високого порядку, кодуючи у приховане відображення такі складні сигнали як поведінка користувача, що є надзвичайно ефективним у задачах рекомендацій для соціальних мереж.

На Рис 4.5 зображено ієрархію взаємодій для умовного користувача u_1. На першому рівні l_1 міститься інформація про об'єкти з якими u_1 контактував

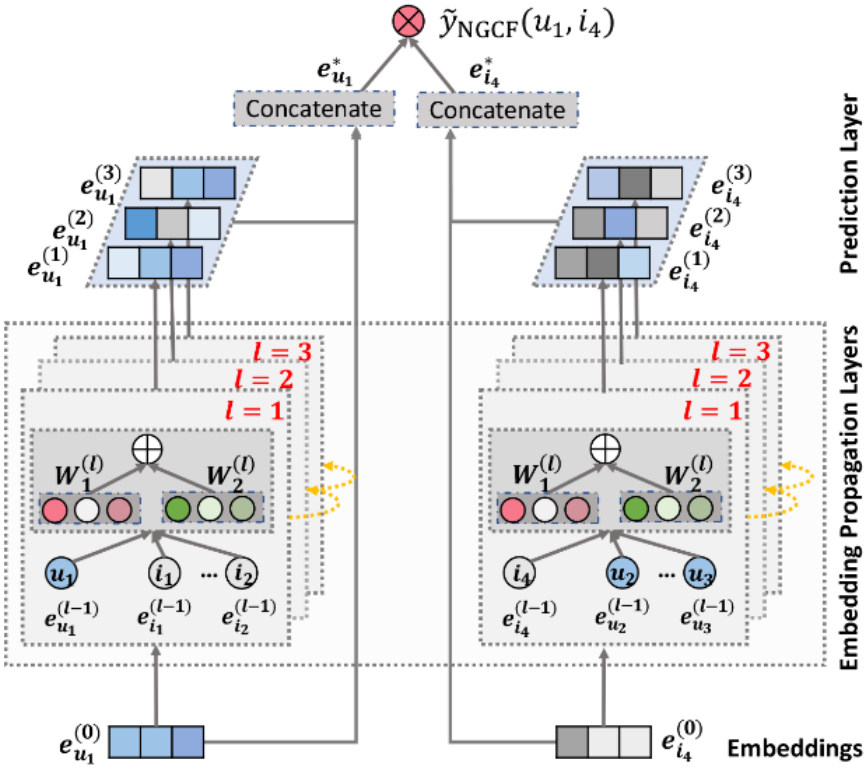


Рисунок 4.6: Архітектура моделі NGCF. Розглянуто приклад побудови оцінки рекомендації i_4 для користувача u_1

напряму (іншими словами, довжина шляху до яких дорівнює 1) і так далі для кожного наступного рівня (або шару). Використовуючи таку інтерпретацію ми описуємо подібність між користувачами на основі взаємодій. Наприклад ланцюг $u_1 \leftarrow i_2 \leftarrow u_2$ вказує на подібність поведінки u_1 і u_2 так як вони переглянули об'єкт i_2, а аналіз більш довшого шляху $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$ пропонує рекомендувати i_4 нашому цільовому користувачу u_1 (i_4 було обрано користувачем u_2, і користувачем u_3 додатково).

Головна ідея алгоритму лежить у семплюванні прикладів взаємодій із навчального набору і ітеративного кодування інформації нейронною мережею виконуючи так званий information propagation. Що насичує латентне відображення потрібною нам інформацією. Модель частково поворює структуру автоенкодера через присутність пари енкодер-декодер. На виході ми отримуємо оцінку релевантності об'єкта рекомендацій для користувача.

Модель складається із трьох компонентів - шару ініціалізації прихованих змінних, групи шарів насичення латентних змінних високорівневою інформацією, і шаром побудови оцінок (prediction). $e_u \in \mathcal{R}^d$ і $e_i \in \mathcal{R}^d$ вектор прихованих змінних користувача і об'єкта які описують їх поведінку, де d

відповідає розміру. Загальна матриця прихованих значень :

$$E = [e_{u_1}, \ldots, e_{u_N}] \times [e_{i_1}, \ldots, e_{i_N}] \tag{4.19}$$

.

У подальшому, ми будемо навчати матрицю E обновлюючи її значення. Цей процес називається агрегацією сигналів (message aggregation), і оперує так званим повідомленням:

$$m_{u \leftarrow i} = f(e_i, e_u, p_{ui}) \tag{4.20}$$

де $f(.)$ функція кодування повідомлення, що використовує в якості входу наші ембедінги e_i і e_u, p_{ui} відповідає за коефіцієнт віддаленості для взаємодій і штрафує пропорційно відстані між u до i. Розглядаємо наступну функцію кодування:

$$m_{u \leftarrow i} = \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_i + W_2(e_i \odot e_u)) \tag{4.21}$$

Де W_1, W_2 є матриці вагів нейромережі які відбирають корисну інформацію для передачі. А $\frac{1}{\sqrt{|N_u||N_i|}}$ відповідає за фактор віддаленості.

Зібравши інформацію із кожного сусіда u ми обновлюємо e_u використовуючи функцію активації LeakyReLU (дозволяє малий сигнал у випадку неактивації нейрону):

$$e_u = LeakyReLU(m_{u \leftarrow u} + \sum_{i \in \mathcal{N}_u} m_{u \leftarrow i}) \tag{4.22}$$

де LeakyReLU:

$$LeakyReLU(x) = \begin{cases} x, \text{if } x > 0 \\ 0.01x \end{cases} \tag{4.23}$$

$m_{u \leftarrow u} = W_1 e_u$ відповідає за збереження сигналу із попередніх ітерацій.

Висновки

Було проаналізовано моделі побудови рекомендацій на основі нейромереж. Обрано для експерементального дослідження моделі нейромережевої факторизації, варіаційного автоенкодера і графової колаборативної фільтрації. Для розглянутих алгоритмів було досліджено принцип їх функціонування, архітектуру, функції активацій, моделювання функцій втрат, наведено інструкції для реалізації.

5 АНАЛІЗ ФАКТОРІВ ВПЛИВУ

Внаслідок розробки великої кількості нових рекомендаційних алгоритмів постало критичне питання у вітсутності єдиного підходу до оцінки їх ефективності, що призводитть до нерепродуктивних та несправедливих результів їх порівняння.

Також, побудова кожної системи рекомендацій складається із багатьох етапів які можна класифікувати на дві умові групи:

- Фактори залежні від моделі
- Фактори не залежні від моделі

Кожен із них критично впливає на результат і залежить один від одного. Більше того їх оптимальні значення і характеристики не відомі. Тому важливо їх детально дослідити.

Фактори не залежні від моделі відповідають специфічним рішенням при пректуванні системи без прямого впливу на сам алгоритм і методів його оптимізації (тобто говоримо про вибір та підготовку навчальної вибірки, деталей проведення валідації).

Інші фактори, навпроти, включають у себе вибір таких речей як функція втрат, оптимізатор і метод регуляризації. Розглядаючи типовий процес побудови рекомендацій варто виділити наступні етапи:

1. Вибір спліт методу (Dataset Spliting Method).
2. Вибір метрики якості (Evaluation Metric Selection).
3. Формулювання функції втрат для оптимізації (Loss function design).
4. Методологія негативного семплінгу (Negative Samoling Strategy).
5. Стратегія ініціалізації вагів (Parameter Initializer Selection).
6. Вибір алгоритму оптимізації (Model Optimizer Selection).
7. Вибір методу регуляризації (Regularization Term / Dropout).
8. Критерії останова (Early Stop Mechanism).
9. Донавчання вагів (Hyper Params Tuning).

Для кожного із вищевказаних факторів впливу відберемо кандидатів для експерементального дослідження. Нашою метою є пошук оптимального підходу до побудови ефективної моделі рекомендацій.

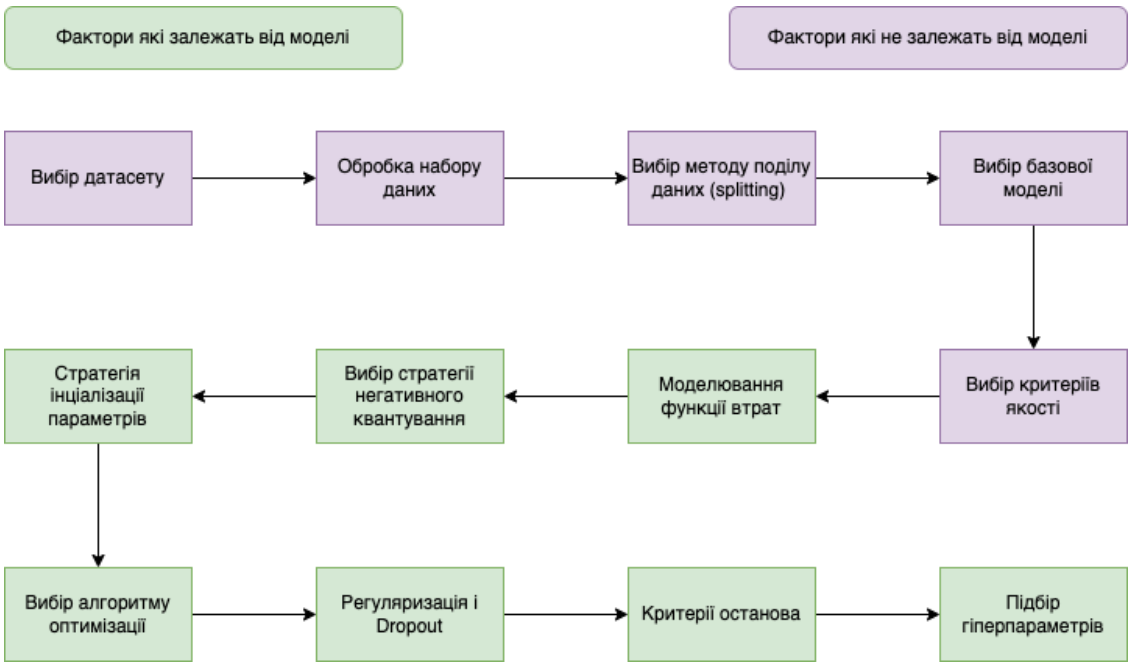


Рисунок 5.1: Розглянутий пайплайн із поділом факторів впливу на категорії

5.1 Вибір спліт методу

Розглянувши набори даних які використовують для побудови рекомендацій, потрібно звернути увагу на розділ датасету на навчальну вибірку і вибірки для тестування (валідації). Валідація виконується для упередження перенавчання моделі, тобто алгоритм перестає широко генералізовуватись і починає змінювати ваги виключно під навчальні екземпляри. Після падіння метрик якості протягом 3-10 епох потрібно зупинити навчання і перейти до тестування на фільній вибірці.

Вибір спліт методу залежить від структури даних, а саме від присутності у них відмітки про час виконання дії. Що, надає можливість обрати одну із двох стратегій поділу.

У класичному підході в задачах машинного навчання, ми вважаємо що кожен зразок навчальної вибірки є не залежним один від одного. І не розглядаємо ймовірність коваріацій між взаємодіями одного користувача у розрізі часу (час взаємодій не впливає на використання). У цьому випадку використовують так званий поділ за співвідношенням. Тобто, ми ділимо датасет на групи відносно деякої пропорції, зазвичай 80% на 20%, 80% відходить на навчальну вибірку (разом із валідацією), а 20% на тестування.

Другий можливий підхід - це поділ відносно часу. У навчальній вибір-

Табл. 5.1: Середня кількість екземплярів для кожного унікального користувача

Тип	ML-1M	LastFm	Netflix
Навчальний	86	39	161
Тест	64	10	84

ці беруть k спостережень певного користувача із максимальним timestamp (відміткою часу) t, а у тестовий набір попадають наступні n екземплярів із timestamp більше t. Тобто ми перевіряємо здатність моделі передбачити n наступних взаємодій.

5.2 Підготовка даних

Для тестового набору. Для завдання побудови рекомендацій потрібно зберігати повноту вибірки для тестування відносно користувачів. Що значить, що для кожного користувача який попадає у даний набір потрібно зберігати одинакову кількість його спостережень (наприклад не менше 5 оцінок на 1 унікальний user_id). У таблиці 7.1 наведено розраховані розміри вибірок для кожного користувача.

Для тренувального набору. Дані проходять додаткову фільтрацію від неактивних користувачів. Усі користувачі кількість взаємодій яких менше порогового значення видаляються із навчальної вибірки. Розглядаємо такі порогові значення [без фільтрації, 5, 10]

5.3 Вибір метрики оцінки якості

Із великого різноманіття розглянутих метрик, було обрано шість основних метрик: Precision@k, Recall@k, MAP, HitRatio, Mean Reciprocal Rank, NDCG. Для метрик які оцінюють фактичну присутність релевантних обєктів у рекомендаціях (*.@k) розглядаємо k у інтеравалі між [10,50].

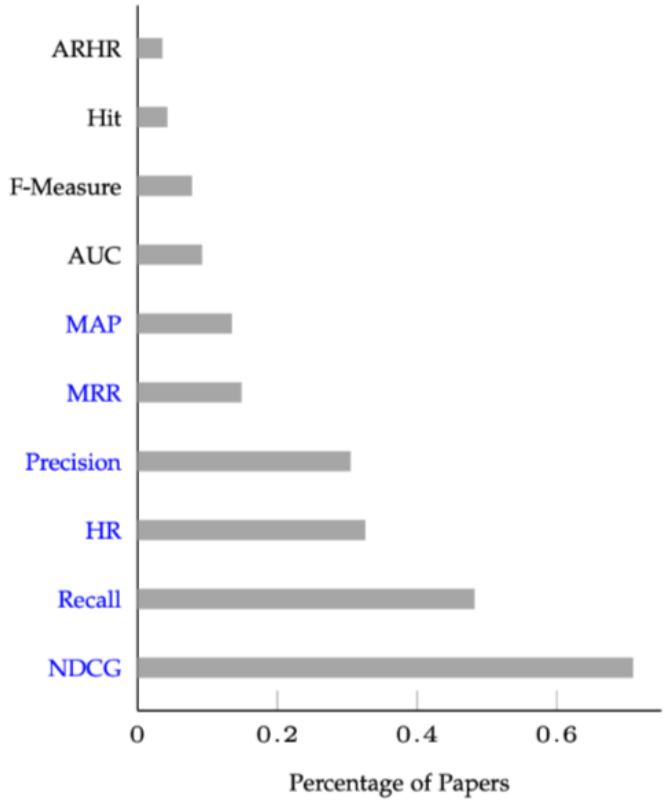


Рисунок 5.2: Розподіл використання метрик у наукових статтях

Метрики оцінюють передбачування моделей із точки зору класифікації і ранжування. Додатково потребує оцінка значень метрик різноманітності. Формулювання яких наведено у Розділі 4.

5.4 Стратегії негативного семплювання

Вибірка об'єктів рекомендацій часто досягає надзвичайно великих розмірів. Так, як моделі на виході повертають оцінку релевантності для цих об'єктів, на кожній ітерації навчання потрібно обновлювати ваги і розраховувати передбачення для кожного елементу. Що є не ефективно із точки зору вартості обчислень. Додатково, більшість користувачів знайомі лише із малою вибіркою елементів. Тому відсутність оцінки/взаємодії може вказувати не на низьку релевантність, а на те що об'єкт їм не відомий.

Для вирішення цієї проблеми використовують негативний семплінг.

Його ідея полягає у відборі певної групи елементів, які не оцінені користувачем в минулому.

В літературі розглядають декілька варіантів семплювання: із нормального розподілу, семплювання на основі низької популярності і семплювання із елементів високої популярності.

- **Семплювання із нормального розподілу**. Обираємо n об'єктів керуючись їх ймовірністю.
- **High popularity sampling**.На основі розрахованих оцінок популярності обираємо top n кандидатів. Відсутність оцінки у невідомого,але популярного об'єкта явно вказує на низьку релевантність для користувача, порівняно із випадковим кандидатом.
- **Low popularity sampling**. Співпадає із high popularity, але вибираємо елементи із хвоста.
- Комбінування обох підходів.

5.5 Стратегії ініціалізації параметрів

Зазвичай у моделях побудови рекомендацій присутній набір параметрів, оптимізація яких навчає алгоритм. Їх вид залежить від обраної архітектури моделі і може бути як матрицею взаємодій, так і вагами нейромереж. Привильний підхід до їх ініціалізації приводить до скорішого навчання і сходимості. Для моделей які будують приховане відображення взаємодій (латентні фактори) типовим є використання рівномірного $\mathcal{U}(0, a)$ і нормального $\mathcal{N}(0, \sigma^2)$ розподілів, із значеннями $a = 1$ і $\sigma = 0.01$ відповідно.

Нейромережі особливо чутливі до початкових значень вагів. При не оптимальному підході, функція втрат змінюватись не буде, навіть після декількох десятків епох. Занадто малі значення приводять до затухання, коли функції активацій у нейронах залишаються неактивними (навіть у випадку використання таких функцій як LeakyReLU). Великі значення, навпроти, приведуть до зривних градієнтів.

У завданнях побудови рекомендацій із допомогою нейронних мереж використовують метод ініціалізації Ксавєра:

$$W_{ij} \sim \mathcal{U}(-\frac{1}{\sqrt{n_{in} + n_{out}}}, \frac{1}{\sqrt{n_{in} + n_{out}}}) \tag{5.1}$$

Де W_{ij} матриці вагів нейромереж, n_{in}, n_{out} розмір вхідного і вихідного

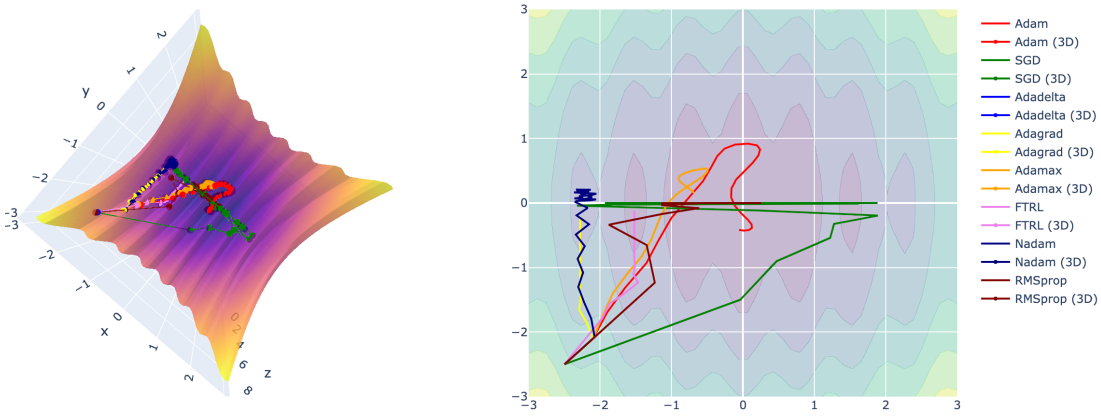


Рисунок 5.3: Результат роботи різних варіантів градієнтного спуску

шару.

5.6 Вибір методу оптимізації моделей

Оптимізатор використовується для оновлення параметрів моделі, мінімізуючи функцію втрат у пошуках глобального мінімуму. Різні оптимізатори впливають на роботу алгоритмів рекомендацій.

Градієнтний спуск. Один із найпопулярніших ітераційних методів оптимізації першого порядку, основа багатьох наступних його варіацій. Розглядається задача пошуку локального мінімуму деякої функції $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$:

$$\mathcal{L} \to \min_{\theta \in \mathbb{R}^n} \tag{5.2}$$

Ідея методу полягає в тому, щоб виконати оптимізацію в напрямку найшвидшого спуску, який задається антиградієнтом $-\nabla_\theta f$:

$$\theta_{t+1} = \theta_t - \lambda \nabla_\theta \mathcal{L}(\theta_t) \tag{5.3}$$

де λ обирається одним із декількох варіантів:

- Константа, у цьому випадку метод може не зійтись.
- Дробовим кроком, який змінюється.
- Найшвидшим спуском : $\lambda_t = \arg\min_\lambda \mathcal{L}(\theta_t) - \lambda \nabla_\theta \mathcal{L}(\theta_t)$.

Критерій останова для градієнтного спуску:

$$|\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta_t)| < \epsilon \tag{5.4}$$

де ϵ наперед задане невідємне число.

Стохастичний градієнтний спуск.Коли алгоритм проходить через навчальний набір, він виконує вищезазначене оновлення для кожного елемента датасету. Виконуючи перетасування, можна виконувати декілька проходів до збіжності, вибираючи оптимальний крок λ.

$$\theta_{t+1} = \theta_t - \lambda \nabla_\theta \mathcal{L}(\theta_t, (u, i)) \tag{5.5}$$

Пакетний градієнтний спуск. Результат подальшого розвитку градієнтного спуску. В результаті експериментів було виявлено ефективність обновлення вагів сумою групи (пакету) елементів. Тобто, на одній ітерації значення вагів акумулюються і оновлюється їх сумою. Що прискорює збігання

$$\theta_{t+1} = \theta_t - \lambda \nabla_\theta \mathcal{L}(\theta_t, B(u, i)) \tag{5.6}$$

Адаптивний градієнт AdaGrad. У завданнях обробки мови, фільтрації спаму чи побудови рекомендацій вхідний сигнал, тобто дані, часто бувають розріджені (У наших датасетах, наприклад Sparsity 99.99+%). Така ситуація приводить до того що інформативні признаки зустрічатимуться достатньо рідко, і із іншої сторони, глобальні шаблони будуть зустрічатись надзвичайно часто (більшість користувачів знайомі із фільмами Володар Перстнів). Тому, було б зручно, визначати наскільки деякий признак часто зустрічається у навчальній вибірці, або як часто він приводить до активації нейронів. Ідея вирішення достатньо проста - ми будемо зберігати для кожного параметру мережі частоту його обновлення. І у випадку коли параметр навчається повільно (частота обновлень низька) ми будемо збільшувати його вплив.

$$\theta_{t+1} = \theta_t - \frac{\lambda}{\sqrt{G_t + \eta}} \cdot \nabla_\theta \mathcal{L}(\theta_t) \tag{5.7}$$

де G_t відповідає за частоту оновлення, а η є деякий ненульовий фактор щоб вберегти ділення на 0. У параметра який часто обновлюється, великий G_t, і його вплив штрафується. Додатково, метод не сильно чутливий до вибору λ, достатньо обрати його один раз.

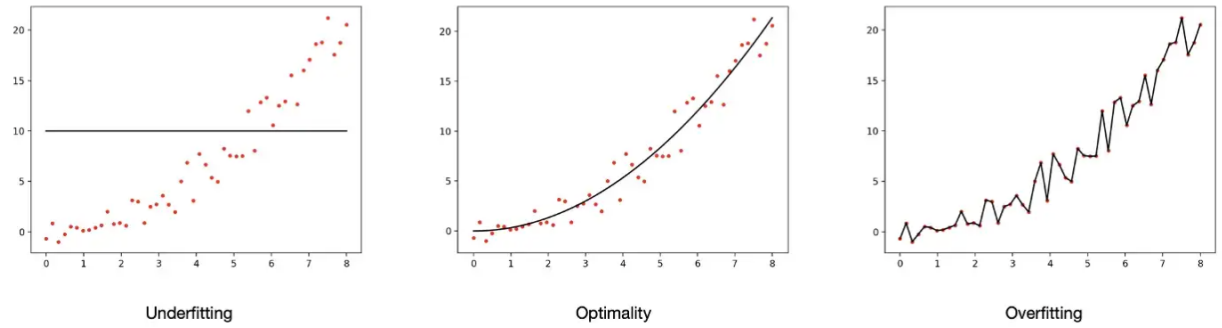


Рисунок 5.4: Приклад недонавчання, перенавчання і оптимального стану функцій втрат

RMSProp. Адаптивний градієнт має властивість затухати при великих значеннях G_t. Модифікацію, що вирішує проблему паралічу алгоритму називають RMSProp.

$$\theta_{t+1} = \theta_t - \frac{\lambda}{\sqrt{\mathbb{E}[g_t^2] + \eta}} \cdot \nabla_\theta \mathcal{L}(\theta_t) \tag{5.8}$$

Ми все ще збираємось оновлювати ваги, які занадто часто оновлюються, але замість повної кількості оновлень ми будемо використовувати квадрат середнього градієнту в історії. Ми використовуємо експоненціальне згасаюче середнє $\mathbb{E}[g^2]$:

$$\mathbb{E}(g^2)_t = \gamma \mathbb{E}[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{5.9}$$

5.7 Критерії останова та Регуляризація

У машинному навчанні використовуються різні стратегії для боротьби з проблемою перенавчання, яка приводить до гіршої генералізації, тим самим досягаючи низької ефективності на тестувальній вибірці (Рис. 6.4). Власне, найбільш широко використовувані методи регуляризації включають регуляризацію, механізм dropout та раньої зупинки.

Регуляризація. Як правило, фактор інтегрується в функцію втрат, щоб допомогти уникнути перенавчання під час навчання моделі рекомендації. В основному використовують два види, а саме регуляризація L1 та L2 (які також відомі як норми). Норма L1 відома як Манхетенська відстань, яка є найбільш природним способом вимірювання відстані між векторами. Це сума величин векторів у просторі, де всі компоненти вектора зважуються однаково. Норма L2 є найпопулярнішою нормою, також відомою як евклідова норма, відпо-

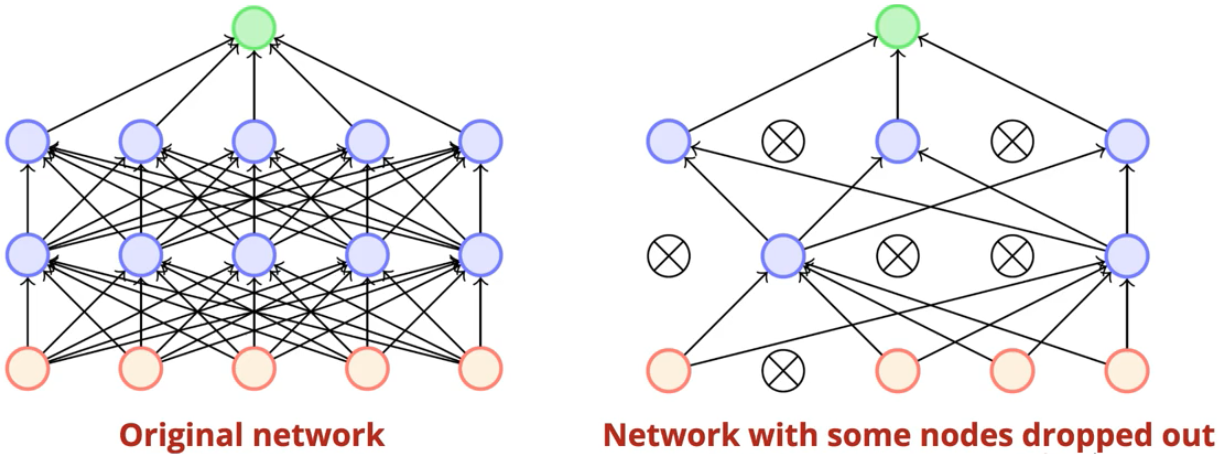


Рисунок 5.5: У результаті Dropout, кожен нейрон із деякою ймовірністю може бути ”виключеним”

відає найкоротшій відстані між двома точками. Головними відмінностями у використані цих двох норм: L1 регуляризація намагається оцінити медіану даних, коли L2 оцінює середнє. Додатково L1 норма виконує функцію feature selection - відбирає лише важливі признаки, що корисно, враховуючи велику кількість признаків у завданнях Глибокого навчання.

Dropout. Широко прийнятий у Глибокому навчанні метод, для упередження перенавчання. Ключова ідея полягає в тому, щоб випадковим чином відключити нейрони (разом із їх зв’язками) із мережі під час тренувань, що заважає мереді занадто сильно адаптуватись (Рис 6.5) . Отже, вводиться додатковий гіперпараметр, тобто ймовірність збереження нейрона P, для контролю інтенсивності виключення. Типові значення P для нейронів прихованих шарів знаходяться в діапазоні від 0,5 до 0,8.

Механізм ранньої зупинки. Рання зупинка - це також форма регулізації, яка використовується для уникнення надмірного перенавчання. Основна проблема моделей побудови рекомендацій(наприклад, LFMS та DLM) полягає у виборі кількості навчальних епох. Занадто багато епох можуть призвести до надмірної підстановки вагів до навчального набору даних, тоді як занадто мало може призвести до слабкої моделі. Рання зупинка - це метод, який дозволяє нам вказати довільну велику кількість навчальних епох і припинити навчання, як тільки продуктивність моделі перестає вдосконалюватись на валідаційному наборі протягом декількох епох.

5.8 Підбір гіперпараметрів

Кожен гіперпараметр заданий набором можливих значень (тобто простором пошуку) на основі емпіричного знання, а оптимальне налаштування отримується шляхом проходженням через увесь простор пошуку. Такий підхід називається Grid Search. Основним його недоліком є обчислювальна нефективність, а у випадку великої кількості гіперпараметрів, його використання не доцільне. Припустимо, модель має m гіперпараметрів, де кожен параметр має у середньому n можливих значень, модель потрібно навчати NM разів, щоб знайти оптимальні значення для всіх гіперпараметрів.

RandomSearch, використовує одинакой піхід, тільки простір можливих значень вибирається випадково.

Одним із більш оптимальних підходів, є так званий Bayesian HyperOpt. Підхід байєсівської оптимізації фокусується на моделі ймовірності $P(score|configur$ яка оновлюється через ітеративний процес завдання якого є максимізація оцінки, враховуючи конфігурацію "C". Hyperopt приймає байєсівську оптимізацію як свою передумову, зробивши деякі варіації в процесі вибірки, визначення та звуження простору пошуку та алгоритмів для максимізації моделі ймовірності.

Висновки

У розділі розглянуто широкий набір фаторів впливу на якість моделей побудови рекомендацій. Представлено їх поділ відносно характеру впливу. Проаналізовано природу факторів, запропоновані варіації факторів для експериментального дослідження.

6 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

Експермиментальне дослідження проводиться на базі сервісу хмарних обчислень Google Colab. Для навчання використано GPU Nvidia Tesla K80. Дані датасетів отримані із офіційних сторінок, розміщені у хмарному сховищі у вигляді текстових документів. Версія Python 3.8.15. Моделі імплементовані із використпанням бібліотеки torch. Також,у розділі наведено інформацію про деталі реалізації експериментів, ціллю яких є знаходження оптимальних параметрів розглянутих алгоритмів рекомендацій на основі результатів метрик якості. А також порівняння впливу не залежних від моделі факторів.

6.1 Датасети

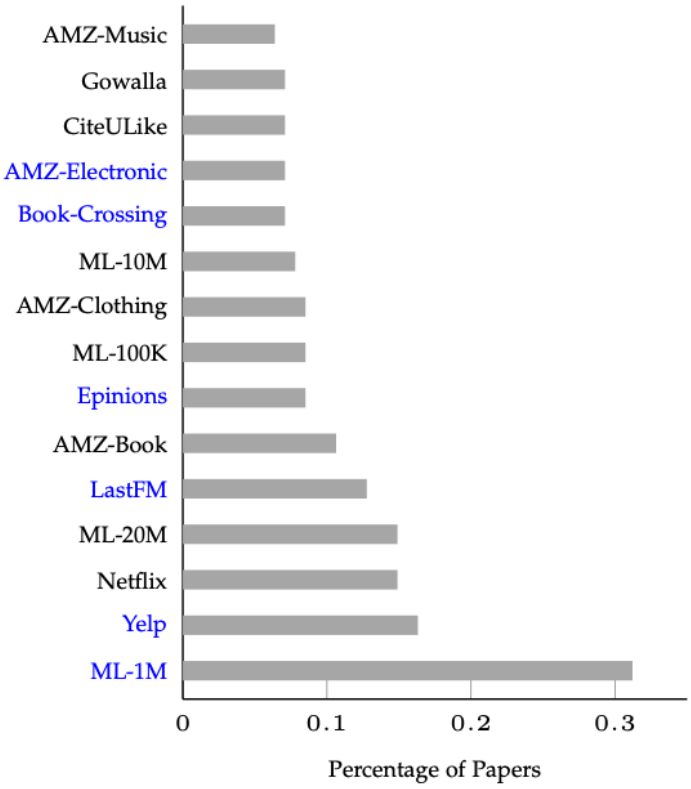


Рисунок 6.1: Розподіл використання наборів даних у наукових статтях.

Для практичних есперименті, а також навчання і валідації моделей будо обрано відкриті набори даних які є загально прийняті і використовуються для порівняння у академічній сфері (Рис. 6.1). Для кожного датасету проведений

розвідувальний аналіз даних (EDA) для оприділення їх основних структурних особливостей і відмінностей. Кожен набір є зрізом БД приватних комерційних компаній, тому можна вважати що поведінка моделей відповідає реальним результатам.

6.1.1 Movielens 1M

Набір даних Movielens (ML-1M) є найпопулярнішим серед великого різноманіття датасетів. Movielens - це мережева система рекомендацій фільмів для користувачів на базі їх відгуків і оцінок. Існує із 1996 року і містить у собі більше 11 мільйонів оцінок для 8600 об'єктів.

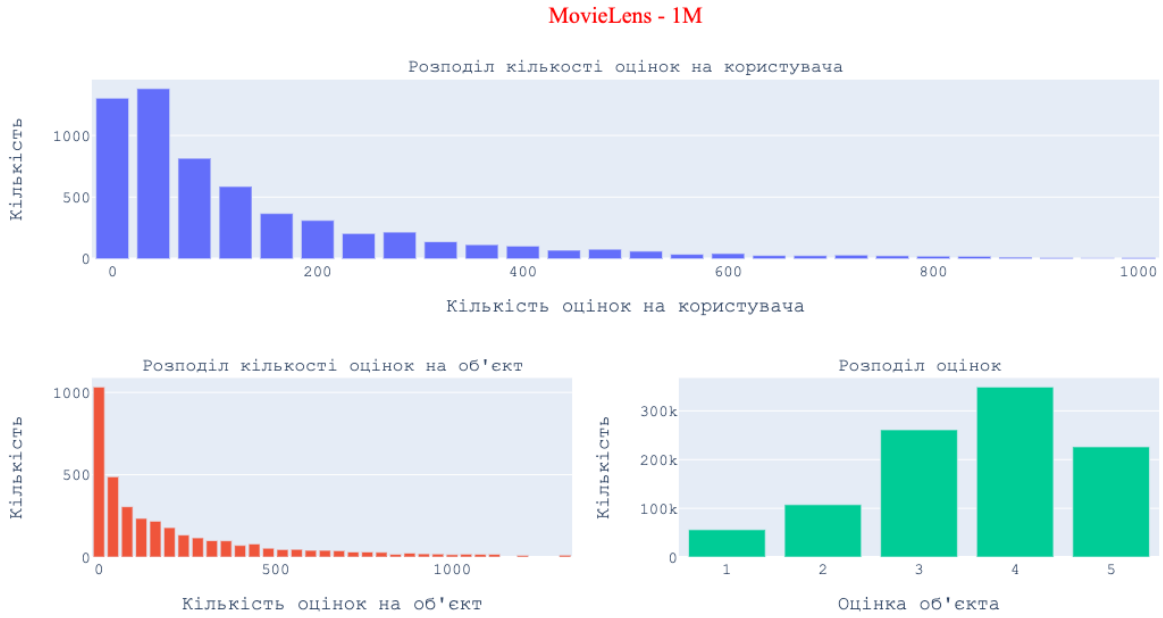


Рисунок 6.2: EDA для набору даних Movielens (ML-1M)

Структура набору наступна:

- $item\_id$ індентифікатор користувача.
- $user\_id$ індентифікатор об'єкта рекомендацій.
- $score$ оцінка.
- $timestamp$ дата і час дії.

Для аналізу відібрано 3706 об'єктів рекомендацій (фільмів) із 1 000 036 оцінок від корисутувачів. Що надає не менше 20 відгуків на один об'єкт.

Табл. 6.1: Характеристика набору даних Movielens 1M

Загальна статистика	Значення
Кількість взаємодій	1 000 029
Кількість користувачів	6 040
Кількість об'єктів	3 706
Розрідженність	99.9553%
Середня оцінка	3.58 / 5
Взаємодій на користувача	
Середнє	165.6
Медіана	96.0
Взаємодій на об'єкт	
Середнє	269.89
Медіана	123.5

Більшість оцінок приймають значення від 3 до 4-5. У розподілі взаємодій чітко замітний лівий перекіс, що типово для даних такого роду. Розрідженість висока.

6.1.2 LastFm

Набір даних LastFm містить у собі інформацію про музику яка прослуховується у медіа плеєрах користувачів. Структурно датасет подібний до ML-1M, але додатково присутня інформація про ключові теги класу для об'єктів. Теги можуть описувати жанр, автора, додаткову ключову інформацію, що є зручним і може бути використано для насичення моделей.

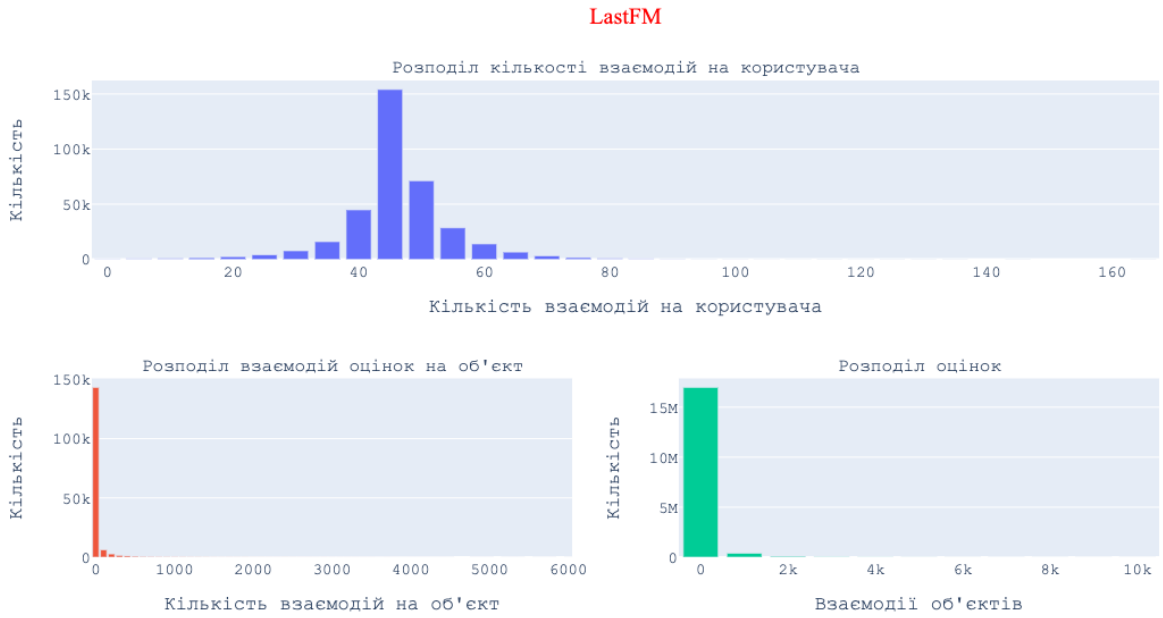


Рисунок 6.3: EDA для набору даних Last.fm

Табл. 6.2: Характеристика набору даних LastFM

Загальна статистика	Значення
Кількість взаємодій	17 535 654
Кількість користувачів	358 868
Кількість об’єктів	160 113
Розрідженність	99.9996%
Взаємодій на користувача	
Середнє	48.23
Медіана	48.0
Взаємодій на об’єкт	
Середнє	108.10
Медіана	6.0

6.1.3 Netflix

Набір даних Netflix є одним із найбільших відкритих датасетів із реальними даними. Набір був створений під егідою проведення змагання Netflix Prize. Ціллю якого було розробка системи рекомендацій, переможець який добився найвищих показників якості отримав призову винагороду у розмірі 1 мільйон доларів. Датасет має класичну структуру: включає у собі дані

про оцінки користувачів фільмів які були переглянути користувачами. Для кожного екземпляру є відмітка про час оцінки.

Дані мають типовий розподіл, і явно замітний перекіс. Із цікавого, через великий розмір і характер домену (рекомендація фільмів), датасет є одним із найнасичених. Розподіл оцінок - стандартний ([3, 4, 5]). Більше 90% користувачів оцінили менше ста об’єктів.

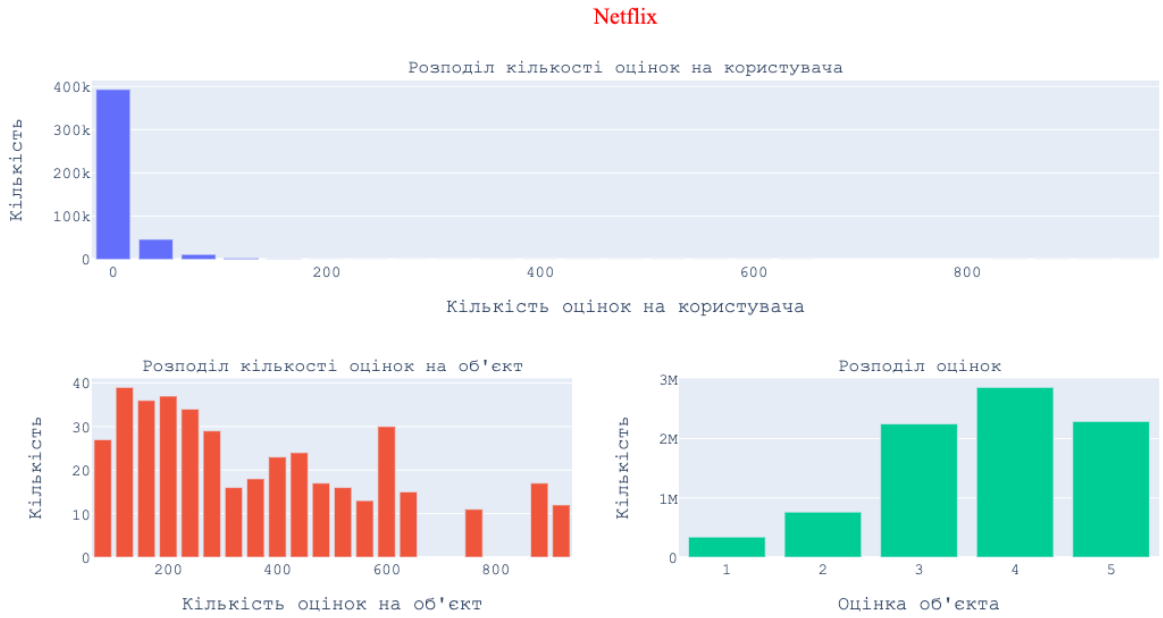


Рисунок 6.4: EDA для набору даних Netflix

У Таблиці 6.3 наведено докладна характеристика набору.

Табл. 6.3: Характеристика набору даних Netflix

Загальна статистика	Значення
Кількість взаємодій	100 480 507
Кількість користувачів	480 189
Кількість об’єктів	17 770
Розрідженність	98.8239%
Взаємодій на користувача	
Середнє	209.33
Медіана	87.0
Взаємодій на об’єкт	
Середнє	5654.50
Медіана	967

Середня кількісь оцінок на об'єкт є надзвичайно високою.

6.2 Середовище розгортання

Експермиментальне дослідження проводиться на базі сервісу хмарних обчислень Google Colab. Для навчання використано GPU Nvidia Tesla K80. Дані датасетів отримані із офіційних сторінок, розміщені у хмарному сховищі у вигляді текстових документів. Версія Python 3.8.15. Моделі імплементовані із використпанням бібліотеки torch.

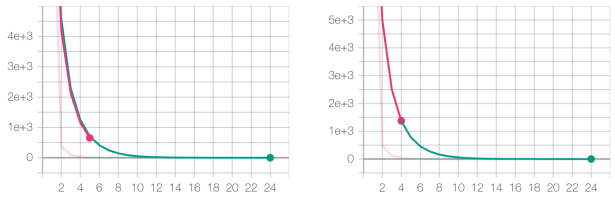
6.3 Результати дослідження

На основі аналізу факторів впливу, а також розглянутих перспективних нейромережевих моделей побудови рекомендацій було сформовано набір експериментів (Таблиця 8.1). Завдання експериментів, використовуючи набори даних, які відповідають реальним вибіркам із БД комерційних компаній дослідити вплив широкого спектру розглянутих факторів. Використовуючи мову програмування Python, а також платформу хмарних обчислень Google Coolab імплементувано розглянуті алгоритми Neural Matrix Factorization, Variational AutoEncoder , Neural Graph Colaborative Filtering.

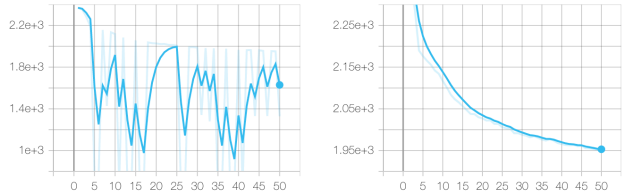
На основі імплементованих алгоритмів проведений порівняльний аналіз ефективності використовуючи метрики вказані у Таблиці 8.1 .

Табл. 6.4: Деталі експерименту

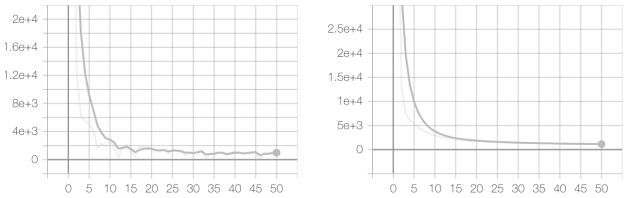
Фактор	Значення
Датасет	ML-1m, LastFM, Netflix
Модель	NeuMF, VAE, NGCF
Фільтрація датасету	Без фільтрації
Сплітінг датасету	За часом
Метрик оцінки якості	Precision@k, Recall@k, MAP, HitRatio, MRR, NDCG
Семплювання	Розподіл Гауса
Ініціалізація θ	Розподіл Гауса
Метод оптимізації	AdaGrad
Регуляризація	L2
Підбір гіперпараметрів	GridSearch



(а) Навчання NeuMF на 24 епохах



(б) Навчання MultiVae на 50 епохах



(в) Навчання NGCF на 50 епохах

Рисунок 6.5: Процес навчання алгоритмів. Зправа - Значення функції втрат, Зліва - дельта втрат відносно епохи.

KPI@K	1	5	10	20	30	50
Recall	0.01452	0.06699	0.1235	0.2215	0.3012	0.4135
MRR	0.6961	0.7929	0.79	0.7981	0.7983	0.7985
NDCG	0.6961	0.8175	0.8200	0.8112	0.8050	0.7978
Hit Ratio	0.6961	0.9246	0.9584	0.9636	0.9688	0.9766
Precision	0.6961	0.641	0.5906	0.5302	0.4785	0.3966

Табл. 6.5: Значення метрик якості для алгоритму MultiVAE. Набір даних MovieLens

Значення функцій втрат під час навчання для кожного із алгоритмів вказано на Рис. 8.1. Для NeuMF було застосовано ранню зупинку, loss не змінювався протягом 3 епох. Дельта втрат вказує на рівномірне зменшування функції втрат. Для MultiVAE доцільно використати сильнішу регуляризацію. У наступних таблицях вказано результи метри якості розраховані на наборі даних Movielens. K (розмір списку рекомендацій) був обраний із інтервалу [1, 50]. Для розрахунку була використане негативне семплювання, із розміро вибірки 1000 у кандидатів.

KPI@K	1	5	10	20	30	50
Recall	0.001757	0.008701	0.01956	0.02799	0.04172	0.05583
MRR	0.04318	0.07580	0.0839	0.0901	0.09206	0.0941
NDCG	0.04318	0.08929	0.1087	0.1315	0.1407	0.157
Hit Ratio	0.04318	0.1295	0.1926	0.2823	0.3322	0.4086
Precision	0.04318	0.04518	0.04186	0.03787	0.03665	0.03342

Табл. 6.6: Значення метрик якості для алгоритму NGCF. Набір даних MovieLens

KPI@K	1	5	10	20	30	50
Recall	0.003363	0.01157	0.02017	0.04535	0.0611	0.09025
MRR	0.1694	0.2382	0.2603	0.2688	0.2717	0.2732
NDCG	0.1694	0.2688	0.322	0.354	0.3701	0.381
Hit Ratio	0.1694	0.365	0.5348	0.657	0.73	0.7873
Precision	0.1694	0.1408	0.1355	0.1250	0.119	0.1136

Табл. 6.7: Значення метрик якості для алгоритму NeuMF. Набір даних MovieLens

Рисунок 6.6: Порівняльний графік метрик якості

Алгоритм MultiVae показав найвищі показники метрик якості на усіх вибірках K,. Варіаційний автоенкодер показує потужну властивість до вивчення поведінки користувачів.

Висновки

Було проведено експерементальне дослідження здатності до побудови рекомендацій алгоритмів NueMF, MultiVAE і NGCF. За результатами тестування, MultiVAE показав найвищі результати по всім розглянутим метрикам.

7 ВИСНОВКИ

7.1 Наукова новизна отриманих результатів

Враховуючи результати експерементів було знайдено ключові фактори впливу на якість та ефективність роботи моделей глибокого навчання у системах рекомендацій. Розроблений алгоритм дає можливість прискорити прототипування, перевірку і порівняння існуючих та майбутніх моделей систем рекомендацій.

7.2 Практичне значення отриманих результатів

Було сформульовано і змодельовано наступні фактори впливу

1. Вибір спліт методу. Методу Dataset Spliting Method
2. Вибір метрики якості Evaluation Metric Selection
3. Формулювання функції втрат для оптимізації Loss function Design
4. Мотодологія негативного семплінгу. Негативний Negative Samoling Strategy
5. Стратегія ініціалізації вагів. Parap Init Strategy
6. Вибір алгоритму оптимізації Model Optimizer Selection
7. Вибір методу регуляризації Regularization Term / Dropout
8. Критерії останова Early Stop Mechanism
9. Донавчання вагів Hyper Params Tuning

Проведене експерементальне дослідження їх впливу на якість роботи обраних алгоритмів рекомендацій. Розроблене ПЗ пришвидшує відбір оптимальних моделей і їх якісне порівняння. ПЗ може використовуватись як і для комерційних цілей так і (наукових) для порівняння нових алгоритмів.

У даній роботі було проаналізовано задачу побудови рекомендацій, розглянуто її проблематику. Розглянуто широкий спектр метрик якості із використанням підходу до класифікації, ранжування і різноманіття. На основі алгоритмів нейромережевої факторизації, варіаційного автоенкодера і графової нейронної колаборативної фільтрації було розглянуто підхід глибокого навчання до побудови рекомендацій. Сформовано перелік факторів впливу на метрики якості. Проведений аналіз їх впливу і можливі рішення для усунення. На основі відкритих наборів даних було імплементовано розглянуті

алгоритми рекомендації і проведене їх порівняння.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

[1] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. *Neural Collaborative Filtering*. In Proceedings of the 26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[2] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. *Variational Autoencoders for Collaborative Filtering*. In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[3] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. *Neural Graph Collaborative Filtering*. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19). Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/3331184.3331267

[4] William L. Hamilton. (2020). *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 14, No. 3 , Pages 1-159.

[5] Le, James, "MetaRec: Meta-Learning Meets Recommendation Systems"(2020). Thesis. Rochester Institute of Technology.

[6] Saúl Vargas. 2014. *Novelty and diversity enhancement and evaluation in recommender systems and information retrieval*. In Proceedings of the 37th international ACM SIGIR conference on Research development in information retrieval (SIGIR '14). Association for Computing Machinery, New York, NY, USA, 1281. https://doi.org/10.1145/2600428.2610382

[7] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. *Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison*. In Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20).

Association for Computing Machinery, New York, NY, USA, 23–32. https://doi.org/10.1145/3383313.3412489

[8] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. *Recommender Systems Handbook (1st. ed.)*. Springer-Verlag, Berlin, Heidelberg.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006. ISBN-10: 0-387-31073-8. Springer New York, NY

[10] Google Colab [Електронний ресурс]. – Режим доступу:https://colab.research.google.com/

[11] Netflix Prize Dataset [Електронний ресурс]. – Режим доступу:https://archive.org/download/nfprize_dataset.tar

[12] Movilens Dataset [Електронний ресурс]. – Режим доступу:https://grouplens.org/datasets/movielens/1m/

[13] LastFM Dataset [Електронний ресурс]. – Режим доступу:https://grouplens.org/datasets/hetrec-2011/

[14] Персептрон [Електронний ресурс]. – Режим доступу:https://ru.wikipedia.org/wiki/Перцептрон