Київський національний університетімені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра обчислювальної математики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 113 Прикладна математика

на тему:

Система рекомендацій фільмів на основі нейронних мереж

Виконав студент 4-го курсу
Холодулькін Гліб Олександрович _____

Науковий керівник:
асистент
Денисов Сергій Вікторович _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Роботу розглянуто й допущено до захисту на засіданні кафедри обчислювальної математики
«___»_____
____202_ р.,протокол № ___
Завідувач кафедри
С. І. Ляшко _____

Київ — 2021

Зміст

Вступ

Рекомендаційні системи – відносно новий напрямок розвитку інформаційних технологій, що виник з появою сучасних інтернет ресурсів, орієнтованих на користувача. Серед систем, для яких рекомендаційні алгоритми грають ключову роль, можна звернути увагу на соціальні мережі, онлайн кінотеатри або інтернет-магазини. Більшість сучасних таких ресурсів не спроможна продовжувати існувати на великому ринку без рекомендаційної системи. Слід зауважити, що подібні системи цікаві не тільки користувачам, а й власникам самих ресурсів, на яких розміщуються рекомендаційні системи. Так як за допомогою подібних інструментів підвищується привабливість самого сервісу та його контенту.

Розробка та впровадження алгоритмів надання рекомендацій може суттєво збільшити прибутки компаній, що було не раз доведено на практиці. Так, наприклад, 2/3 всіх фільмів, що було переглянуто користувачами Netflix, було знайдено саме завдяки алгоритмам рекомендацій, а продажі товарів, рекомендованих алгоритмами Amazon, складають приблизно 35% всього доходу. Тож рекомендаційні системи входять до сучасних засобів клієнтоорієнтованих систем, і складають важливу частину їх функціоналу.

Ця робота направлена на моделювання та створення якісної системи рекомендацій фільмів, яка у своїй основі використовує модель машинного навчання Bert (*Bidirectional Encoder Representations from Transformers*).

Проблематика рекомендацій фільмів

Проблема вибору відеоматеріалу залишається досить актуальною тому, що безліч людей витрачають значну кількість часу на пошук необхідного контенту. Із збільшенням кількості інформації в мережі інтернет цей процес буде займати все більше часу і його якість буде невпинно погіршуватися. Багато сучасних веб-ресурсів дозволяють фільтрувати набір фільмів за деякими параметрами, але швидко і якісно надати необхіну рекомендацію вони не спроможні. Їхнім недоліком є те, що вони не використовують методи штучного інтелекту при наданні користувачеві рекомендацій. Тому дана проблеми є досить актуальною і потребує дослідження та вирішення або покращення її стану.

Система рекомендації фільмів виявилася потужним інструментом для надання корисних пропозицій фільмів для користувачів. За допомогою таких систем надаються рекомендації, які допомагають користувачам справитись із надмірною кількістю інформації, а також допомагають швидко та зручно підібрати відповідні фільми. Формування системи рекомендацій фільмів – це цікаве і складне завдання, яке передбачає аналіз різних вподобань користувачів, різноманітності фільмів та інше. Тому для вирішення проблеми надання якісних рекомендацій у сучасному світі вже запропоновано багато методів.

Базові принципи вирішення проблеми рекомендацій

Серед підходів до вирішення проблеми рекомендацій фільмів розрізняють рекомендації основані на контенті (content-based), колаборативну фільтрацію (collaborative filtering) та гібридну техніку, яка поєднує два попередні методи.

Загальний принцип content-based методів - знайти спільні характеристики фільмів, які отримали схвальну оцінку від користувача, а потім рекомендувати для цього користувача нові фільми, які також мають ці характеристики. Рекомендаційні системи, які базуються лише на контенті

зазвичай мають проблеми аналізу обмеженого вмісту та надмірної спеціалізації. Аналіз обмеженого вмісту настає коли система має обмежену кількість інформації про користувачів та об’єкти. Наприклад, в додатку для рекомендацій фільмів, система може рекомендувати користувачу фільми такого ж жанру, або ті які мають однакових акторів з тими фільмами, які користувач вже подивився. Однак система не спроможна порекомендувати фільми, які відрізняються від попередніх і які також були б цікаві користувачу. Замість того щоб залежати від контентної інформації, методи колаборативної фільтрації використовують інформацію про оцінки фільмів в системі від інших користувачів. Ключовою ідеєю є те, що оцінка певного юзера для нового фільму буде схожою з тою, яку поставив інший користувач, якщо вони обидва оцінили інші фільми подібними оцінками. Колаборативна фільтрація змогла подолати певні обмеження, які має content-based. Наприклад, фільми, контент яких є відсутнім або важко доступним, можуть бути рекомендовані базуючись на відгуках інших користувачів. Також колаборативна фільтрація може рекомендувати різноманітні фільми з різним контентом, оскільки інші користувачі проявили інтерес до цих фільмів.

Проблема послідовної рекомендації

Основа якісної та ефективної системи рекомендацій полягає в точній характеристиці інтересів користувачів. Більшість реальних застосовуваних систем полягається на динамічніть інтересів, які змінюються під впливом поведінки користувача. Наприклад, на сайті інтернет-магазину можна побачити рекомендацію придбати мобільні аксесуари для конкретної моделі смартфону після його покупки. Хоча за інших умов поведінки користувача інтернет-магазин не буде рекомендувати такі аксесуари.

Для моделювання такої послідовної динаміки в поведінці користувачів були запропоновані різні методи для знаходження послідовних рекомендацій, заснованих на минулих взаємодіях користувачів [1, 2, 3]. Вони націлені на прогнозування наступних один за одним елементів, з якими

користувач, ймовірно, буде взаємодіяти з урахуванням його минулих дій. Останнім часом велика кількість робіт для послідовної рекомендації використовують послідовні нейронні мережі, наприклад, Recurrent Neural Network (RNN), які показують багатообіцяючі результати [4, 5, 6]. Основна парадигма попередніх робіт - закодувати історичні взаємодії користувача в вектор представлення його вподобань з використанням послідовної моделі «зліва направо» і дати рекомендації на основі цих вподобань.

Недоліки існуючих рішень послідовної рекомендації та ціль роботи

Односпрямовані моделі показали свою ефективність та активно застосовуються для послідовних рекомендацій. Однак, досить часто їх можливостей недостатньо для отримання оптимального представлення вподобань користувача.

Основне обмеження полягає в тому, що такі моделі обмежують можливості прихованого представлення для елементів в історичній послідовності, де кожен елемент може тільки кодувати інформацію з попередніх елементів. Ще одне обмеження полягає в тому, що попередні односпрямовані моделі вводилися та застосовувалися для послідовних даних з природним порядком, наприклад, текстових даних чи даних часових рядів. Вони часто полягаються на строго впорядковану послідовність даних, що не завжди вірно для поведінки користувачів в реальних системах. У такій ситуації вкрай важливо включити контекст з обох напрямків в моделювання послідовності поведінки користувача.

Щоб усунути згадані вище обмеження, використовуються двосторонні моделі для обробки представлень історичної послідовності поведінки користувачів. Так як модель BERT [7] показала вражаючі результати в розумінні текстів, активно вивчаються можливості її застосування в рекомендаційних системах.

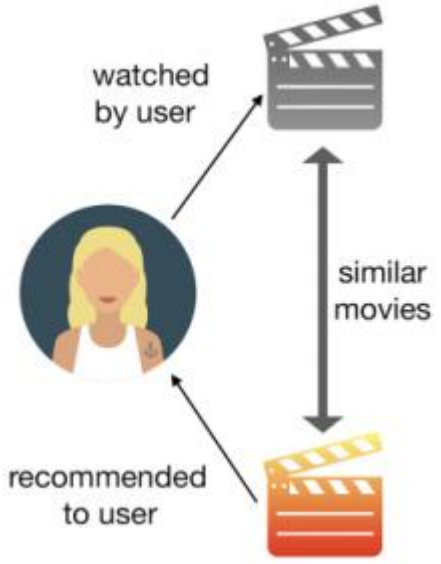
Чудові результати для глибоких двонапрямлених моделей в задачах моделювання текстових послідовностей показують, що корисно використовувати контекст з обох боків [7]. Така модель краще підходить, ніж односпрямовані моделі при моделюванні послідовності поведінки користувача, оскільки всі елементи двонаправленої моделі можуть використовувати контекст як з лівого, так і з правого боку.

Ціль цієї роботи полягає в моделюванні послідовності поведінки користувачів та демонстрації ефективності двонаправленної моделі за допомогою аналізу на двох еталонних наборах даних.

Підходи базових рекомендаційних систем

Content-Based

Такий підхід до рекомендацій фільмів не передбачає участі інших юзерів. Залежно від вподобань користувача алгоритм обере схожі елементи (тобто елементи з схожим змістом) та порекомендує їх. При такому підході різноманітність рекомендацій буде мінімальною, оскільки враховується лише те, що конкретно подобається користувачу. Наприклад, користувачу, якому подобаються бойовики, будуть рекомендуватися інші бойовики до тих пір, поки він самостійно не спробує подивитися якийсь інший жанр фільмів. Звичайно, є багато категорій, за якими ми можемо обчислити схожість між двома фільмами. Наприклад, можна знайти схожість за жанром, ключовим словом, акторським складом, режисером і так далі.



мал 1

Щоб знайти схожий контент для елементу, найчастіше використовується косинусна міра подібності. Скалярний добуток між двома векторами рівний проекції одного з них на інший. Отже, скалярний добуток двох однакових векторів рівний квадрату їх модулів. З іншого боку, якщо два вектора не мають спільних напрямків, добуток дорівнюватиме нулю. Загальна формула для розрахунку скалярного добтку приведена нижче:

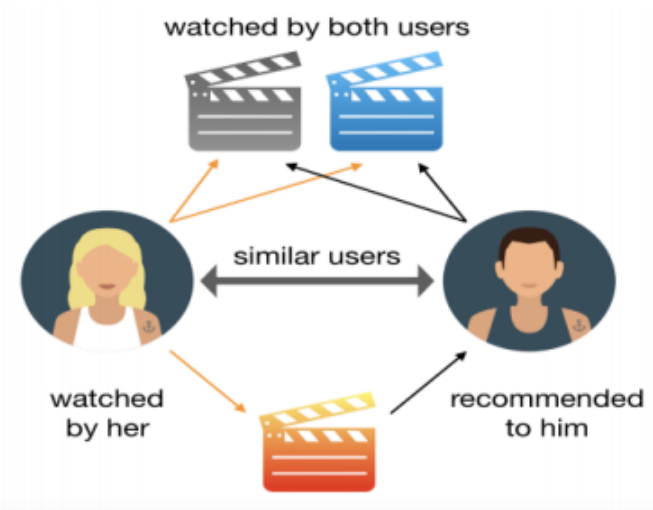
$$\boldsymbol{u} \cdot \boldsymbol{v} = [u_1\ u_2\ \ldots\ u_n] \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n = \sum_{i=1}^{n} u_i v_i$$

Визначення подібності між двома векторами u і v насправді є відношенням між їх скалярними добутками і добутком їх величин:

$$similarity = cos(\theta) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}}$$

Collaborative Filtering

Попередній підхід не використовує інших користувачів для надання рекомендацій і тому має деякі недоліки, які були описані вище. Для вирішення проблеми малої різноманітність рекомендацій використовується метод спільної фільтрації. Цей підхід будується на ідеї рекомендації фільмів, які користувач не дивився, але які подивилися і сподобалися іншим користувачам, схожим на нашого тестового користувача. Цей тип спільної фільтрації називається підходом спільної фільтрації між користувачами, оскільки ми знаходимо користувачів, схожих на нашого користувача. Щоб визначити, чи схожі два користувача, розглядається фільми, які дивилися обидва з них, і те, як вони їх оцінили. Таким чином, дивлячись на загальні елементи, ми прогнозуємо оцінки, які користувач поставить фільму, на основі аналогічних оцінок інших користувачів.



мал 2

Стандартний метод спільної фільтрації використовує алгоритм найближчого сусіда. Є матриця рейтингів розміру $n \times m$ з користувачем u_i, $i = 1, \dots n$ і елементом p, $j = 1, \dots m$. Тепер ми хочемо спрогнозувати рейтинг r_i, якщо цільовий користувач i не взаємодіяв з елементом j. Процес полягає в тому, щоб обчислити схожість між цільовим користувачем i і всіма іншими користувачами, вибрати X найбільш схожих користувачів і взяти середньозважене значення оцінок цих X користувачів зі схожістю в якості ваг:

$$r_{ij} = \frac{\sum_k Similaries(u_i, u_k) r_{kj}}{number\ of\ ratings}$$

Однак далеко не всі користувачі однаково оцінюють фільми. Деякі юзери зазвичай ставлять високі бали, в той час як інші досить суворі в своїх оцінках, навіть якщо вони задоволені фільмом. Щоб уникнути такої упередженості, корисно буде відняти середні оцінки кожного користувача по всіх елементах при обчисленні середньозваженого значення і додати його назад для цільового користувача, як показано нижче:

$$r_{ij} = \bar{r}_i + \frac{\sum_k Similaries(u_i, u_k)(r_{kj} - \bar{r}_k)}{number\ of\ ratings}$$

Matrix Factorization. Розрідженість матриці - це велика проблема, яку необхідно вирішити при створенні систем рекомендацій зі спільною фільтрацією. Підхід матричної факторизації створює матриці, в яких рядки являють собою унікальних користувачів, стовпці представляють різні фільми, а значення всередині є оцінками, які різні користувачі дають фільмам. Однак цілком очевидно, що не всі фільми будуть оцінюватися кожним користувачем. Таким чином, наша матриця стикається з проблемою розрідженості, яку необхідно вирішити. Для цього використовують матричну факторизацию. У цьому методі розкладається вихідна розріджена матриця на матриці низької розмірності з прихованими функціями. Таким чином, матрична факторизація показує, наскільки користувач узгоджений з набором прихованих функцій і наскільки фільм відповідає цьому набору прихованих функцій.

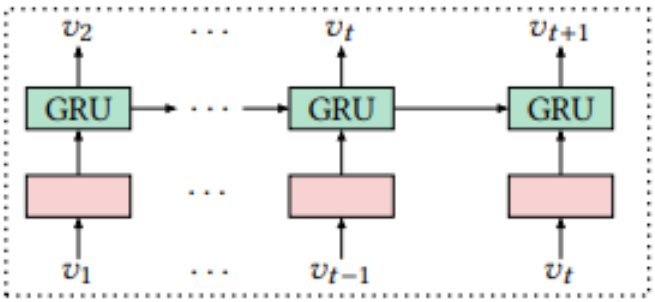
Перевага цього підходу в порівнянні з попереднім алгоритмом полягає в тому, що, хоча два користувача не оцінили одні й ті ж фільми, все ж можна виявити схожість між ними, якщо у них однакові приховані функції.

Підходи послідовних рекомендаційних систем

Основа якісної та ефективної системи рекомендацій полягає в точній характеристиці інтересів користувачів. Більшість реальних застосовуваних систем полягається на динамічність інтересів, які змінюються під впливом поведінки користувача.

Послідовна рекомендаційна система на основі RNN

Для моделювання послідовної динаміки поведінки користувачів були запропоновані різні методи для знаходження послідовних рекомендацій, заснованих на минулих взаємодіях користувачів. Вони націлені на прогнозування наступних елементів, з якими користувач, ймовірно, буде взаємодіяти з урахуванням його минулих взаємодій. Останнім часом велика кількість робіт використовують нейронні мережі, наприклад, Recurrent Neural Network (RNN). Такі рішення послідовної рекомендації отримали багатообіцяючі результати [4, 8]. Основна парадигма попередніх робіт - закодувати історичні взаємодії користувача в вектор (тобто представлення вподобань користувача) з використанням послідовної моделі «зліва направо» і дати рекомендації на основі цих вподобань. Принцип роботи такої моделі можна побачити на мал 1.

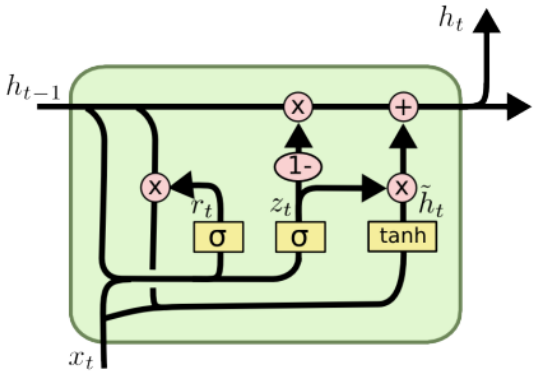


(d) RNN based sequential recommendation methods.

мал 3

У цій схемі GRU – це блок, який складається з трьох нейронів. Спочатку на вхід в сигмоідальний шар подається вектором $[h_{t-1}, x_t]$, на

виході маємо вектор оновлення z_t, розмірність якого відповідає вектору h_{t-1}. Потім за допомогою іншого сигмоідального шару аналогічним способом будується вектор сбросу r_t, який вказує які компоненти вектора h_{t-1} в якій степені потрібно використовувати для побудови оновлення вихідного вектора. І, наостанок, на третій нейрон з функцією активації $tanh$ подається вектор, складений з покоординатного добутку r_t , h_{t-1} і x_t. Після цієї операції ми отримуємо оновлення вихідного вектору h_t. Множимо покоординатно вектор h_{t-1} на $1 - z_t$, а оновлення вектора – на z_t, і складаючи отримані вектора покоординатно, отримуємо вихідний вектор h_t. Принцип роботи цієї моделі зображений на малюнку 2.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

мал 4

Послідовна рекомендаційна система на основі моделі BERT

Модель BERT - це мовна модель нової ери NLP [9]. В якості її базової структури полягає Трансформер, який використовує механізм уваги. Основна мета Трансформера — максимально правильно передбачити наступне слово або знак у реченні. В нашому випадку це настуний фільм, який ймовірно захоче переглянути користувач. Ми можемо використати цю здатність трансформера для вирішення проблеми послідовної рекомендації. Більш того, Google вже попередньо навчив модель BERT, нам потрібно тільки використовувати даний набір даних для його точного налаштування.

Трансформери

Трансформер - це модель глибокого машинного навчання, запроваджена в 2017 році, яка використовується насамперед у галузі обробки природних мов [10]. Як і рекурентні нейронні мережі (RNN), трансформери призначені для обробки впорядкованих послідовностей даних, таких як природна мова, для різних завдань, таких як машинний переклад та узагальнення тексту. Однак, на відміну від рекурентних нейронних мереж, трансформерам не потрібно, щоб послідовність оброблялася по порядку. Отже, якщо дані є природною мовою, трансформер не вимагає обробки спершу початку речення, а потім кінця. Завдяки цій особливості трансформер дозволяє набагато більше паралелізації, ніж рекурентні нейронні мережі під час тренування. З моменту їх введення трансформери стали основним складовим елементом більшості сучасних архітектур в обробці природної мови, замінивши в багатьох випадках рекурентні нейронні мережі, такі як довготривала-короткочасова пам'ять (LSTM). Оскільки архітектура Трансформера сприяє більшій паралелізації під час навчальних обчислень, вона дозволила навчатись на набагато більше даних, ніж це було можливо до її введення. Це призвело до розробки попередньо натренованих систем, таких

як BERT, яка пройшла навчання з величезною кількістю загальних мовних даних і тепер може бути налаштована за допомогою тонкої настройки та підготовлена до конкретного завдання, наприклад послідовної рекомендації фільмів [11].

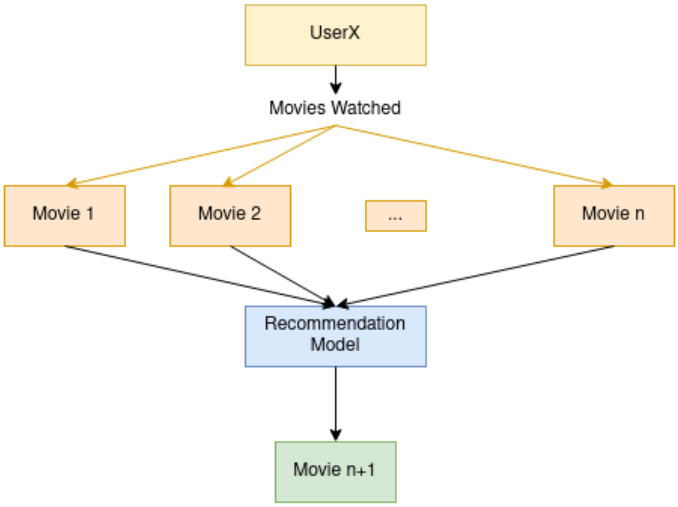
Архітектура моделі BERT4REC

Постановка задачі

В послідовній рекомендаційній системі нехай $U = \{u_1, u_2, \dots, u_{|U|}\}$ позначає множину користувачів, а $V = \{v_1, v_2, \dots, v_{|V|}\}$ позначає множину елементів. Тоді список $S_u = \left[v_1^{(u)}, \dots, v_t^{(u)}, \dots, v_{n_u}^{(u)}\right]$ буде позначати послідовність взаємодії в хронологічному порядку для користувача $u \in U$, де $v_t^{(u)} \in V$ це елемент, з яким взаємодіяв користувач u на кроці часу t, а n_u - довжина послідовності взаємодії для користувача u. Враховуючи історію взаємодії S_u, послідовна рекомендаційна система націлена на прогнозування фільму, з яким користувач u буде взаємодіяти на кроці часу $n_u + 1$. Це можна формалізувати як моделювання ймовірності всіх можливих елементів для користувача u на часовому кроці $n_u + 1$:

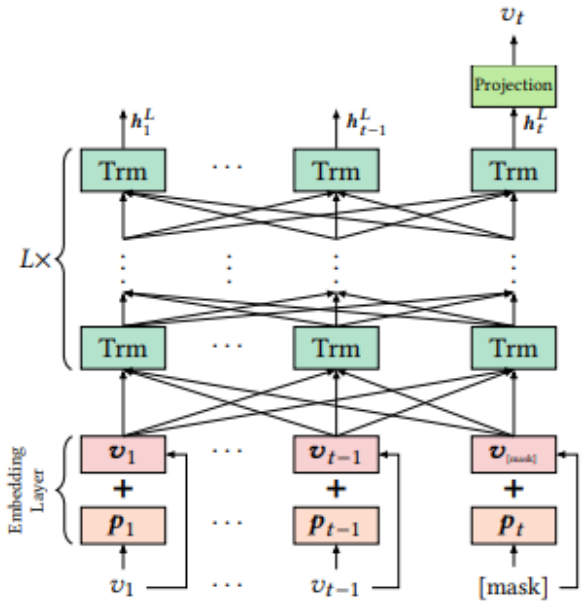
$$p\left(v_{n_u+1}^{(u)} = v \middle| S_u\right)$$

Принцип роботи цієї моделі для одного юзера можна побачити на мал. 2.



мал 5

BERT4Rec використовує двонаправлене кодування трансформерів для вирішення нового завдання – послідовних рекомендацій. Модель побудована на основі шару самоуваги. Як показано на малюнку 3, BERT4Rec складається з L шарів двонапрямлених трансформерів. На кожному шарі модель итеративно переглядає представлення кожного елементу, обмінюючись інформацією з позиціями всіх елементів на попередньому шарі паралельно з шаром Transformer. Замість того, щоб вчитися передавати відповідну інформацію послідовно крок за кроком, як це роблять методи на основі RNN на малюнку 1, механізм самоуваги наділяє BERT4Rec здатністю безпосередньо розпізнавати залежності між фільмами на будь-яких відстанях в історичній послідовності. Таким чином, пропонована модель може отримати більш потужне представлення про послідовність поведінки користувачів для поліпшення ефективності рекомендацій.

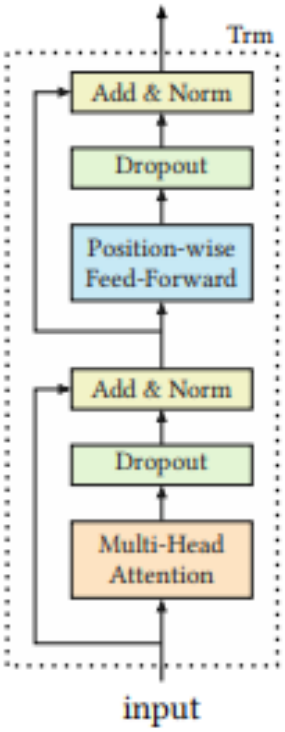


(b) BERT4Rec model architecture.

мал 6

Transformer Layer

Як показано на малюнку 3, враховуючи вхідну послідовність довжини t, ітеративно обчислюються приховані представлення (hidden representation) h_l^i на кожному шарі l для кожної позиції i, застосовуючи шар Transformer. Тут ми складаємо $h_l^i \in \mathbb{R}^d$ разом в матрицю $H^l \in \mathbb{R}^{t\times d}$, оскільки на практиці функція уваги обчислюється одночасно для всіх позицій. Як показано на малюнку 4, шар Трансформера Trm містить два підшара: підшар механізму самоуваги (*Multi-Head Self-Attention*) і підшар нейронної мережі прямого поширення (*Feed-Forward Network).*



мал 7

Multi-Head Self-Attention. Механізми уваги стали невід'ємною частиною моделювання послідовності в різних задачах, дозволяючи влавлювати залежності між парами представлень елементів без урахування відстані між ними в послідовності. В моделі використовується Multi-Head Self-Attention замість виконання однієї функції уваги. Зокрема, механізм уваги Multi-Head спочатку лінійно проектує H^l на h підпросторів, з різними лінійними проекціями, а потім застосовує h функцій уваги паралельно для створення вихідних представлень, які об'єднуються і знову проектуються:

$$\begin{aligned} \mathrm{MH}(\boldsymbol{H}^l) &= [\mathrm{head}_1; \mathrm{head}_2; \ldots; \mathrm{head}_h]\boldsymbol{W}^O \\ \mathrm{head}_i &= \mathtt{Attention}(\boldsymbol{H}^l\boldsymbol{W}_i^Q, \boldsymbol{H}^l\boldsymbol{W}_i^K, \boldsymbol{H}^l\boldsymbol{W}_i^V) \end{aligned} \quad (1)$$

Тут матриці проекцій для кожного ведучого елемента (head) є навчальними параметрами. Тоді функція уваги представляє собою масштабований скалярний добуток уваги:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d/h}}\right)\boldsymbol{V} \qquad (2)$$

де Q – матриця запитів, K – матриця ключів и V – матриця значень. Ці параметри проектуються з однією і тією ж матриці H^l за допомогою різних вивчених проекційних матриць, як показано в рівнянні 1. Параметр $\sqrt{d/h}$ був введений для отримання більш м'якого розподілу уваги, щоб уникнути надзвичайно малих градієнтів [10].

Позиційні нейронні мережі прямого поширення (Position-wise Feed-Forward Networ). Як описано вище, підрівень самоуваги в основному заснований на лінійних проекціях. Щоб надати моделі нелінійність і взаємодію між різними вимірами, застосовується позиційна нейрона мережа прямого поширення до виходів підшарів механізму самоуваги однаково в кожній позиції. Вона складається з двох афінних перетворень з активацією Гаусовської помилки (GELU) між ними:

$$\text{PFFN}(\boldsymbol{H}^l) = \left[\text{FFN}(\boldsymbol{h}_1^l)^{\top}; \ldots; \text{FFN}(\boldsymbol{h}_t^l)^{\top}\right]^{\top}$$
$$\text{FFN}(\boldsymbol{x}) = \text{GELU}\left(\boldsymbol{x}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}\right)\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} \qquad (3)$$

$$\text{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}(x/\sqrt{2})\right].$$

де $\Phi(x)$ - кумулятивна функція розподілу стандартного гауссова розподілу, $W^{(1)} \in \mathbb{R}^{d \times 4d}$, $W^{(2)} \in \mathbb{R}^{4d \times d}$, $b^{(1)} \in \mathbb{R}^{4d}$– навчальні параметри, однакові для всіх позицій.

Stacking Transformer Layer. Як було описано вище, ми можемо легко зафіксувати взаємодію елементів на всій послідовності поведінки

користувача за допомогою механізму самоуваги. Як правило, корисно вивчати більш складні паттерни переходів від елемента до елементу. Однак, мережу стає складніше навчати по мірі поглиблення. Тому використовуєтья залишковий зв'язок [12] навколо кожного з двох підшарів, як показано на малюнку 2. А потім проводиться нормалізація шара [13]. Більше того, також застосовується відсів (Dropout) [14] до виходу кожного підшару перед його нормалізацією. Тобто, вихід кожного підшара дорівнює наступному:

$$LN\left(x + Dropout\big(sublayer(x)\big)\right)$$

де $sublayer(\cdot)$ - функція, реалізована самим підшаром, LN - функція нормалізації шарів, яка використовується для нормалізації входів всіх прихованих блоків в одному шарі для стабілізації та прискорення навчання мережі.

Вцілому, BERT4Rec визначає приховані представлення кожного слова наступним чином:

$$\boldsymbol{H}^l = \mathtt{Trm}\big(\boldsymbol{H}^{l-1}\big), \quad \forall i \in [1, \ldots, L] \tag{4}$$

$$\mathtt{Trm}(\boldsymbol{H}^{l-1}) = \mathtt{LN}\Big(\boldsymbol{A}^{l-1} + \mathtt{Dropout}\big(\mathtt{PFFN}(\boldsymbol{A}^{l-1})\big)\Big) \tag{5}$$

$$\boldsymbol{A}^{l-1} = \mathtt{LN}\Big(\boldsymbol{H}^{l-1} + \mathtt{Dropout}\big(\mathtt{MH}(\boldsymbol{H}^{l-1})\big)\Big) \tag{6}$$

Embedding Layer

Як було описано вище, без рекуррентного модуля трансформаторний шар Trm не знає про порядок елеменів у вхідній послідовності. Для того, щоб використовувати послідовну інформацію на вході, вводяться позиційні вкраплення вхідних елементів у нижній частині стеків шару трансформера. Для даного елемента v_i, його вхідне представлення h_i^0 виконується за допомогою підсумовування відповідних елементів та позиційних вкраплень:

$$h_i^0 = v_i + p_i$$

де $v_i \in E -$ d-вимірне вбудовування (embedding) для елемента v_i. $p_i \in P - d$-вимірне позиційне вбудовування для індексу позиції i. У цій роботі використовується позиційні вбудовування замість фіксованих синусоїдних вбудовувань [10] для кращих результатів. Матриця позиційного вбудовування P дозволяє нашій моделі визначити, з якою частиною вхідних даних вона має справу. Однак це також накладає обмеження на максимальну довжину вхідної послідовності N, яку може обробити наша модель. Таким чином, потрібно урізати вхідну послідовність $[v_1, \dots, v_t]$ до останніх N елементів $[v^u_{t-N+1}, \dots, v_t\ ]$, якщо $t > N$.

Output Layer

Після L шарів, які ієрархічно обмінюються інформацією по всім позиціям попереднього шару, ми отримуємо кінцевий вихід H^l для всіх елементів вхідної послідовності. Припускаючи, що ми маскуємо елемент v_t на часовому кроці t, ми прогнозуємо замасковані елементи v_t на основі h^L_t, як показано на малюнку 3. Точніше, застосовуєтьс двошарова мережа прямого направлення і активація GELU між ними для отримання вихідного розподілу за цільовими елементами:

$$P(v) = \text{softmax}\left(\text{GELU}(\boldsymbol{h}_t^L \boldsymbol{W}^P + \boldsymbol{b}^P)\boldsymbol{E}^\top + \boldsymbol{b}^O\right) \qquad (7)$$

де W^P - навчальна проекційна матриця, b^P і b^O- члени зміщення, $E \in \mathbb{R}^{|\mathcal{V}| \times d} -$ матриця вбудовування для набору елементів $\mathcal{V}$. Також використовується загальна матриця вбудовування елементів у вхідному і вихідному шарі для зменшення надлишкової підгонки і скорочення розміру моделі.

Model Learning

Звичайні однонаправленні моделі послідовних рекомендацій навчають модель, передбачаючи наступний елемент для кожної позиції у вхідній послідовності. Зокрема, цілью вхідної послідовності $[v_1, \dots, v_t]$ є здвинута версія $[v_2, \dots, v_{t+1}]$. Однак, як показано на малюнку 3, спільне зумовлювання як лівого, так і правого контексту в двонаправленій моделі призведе до того, що кінцеве вихідне представлення кожного елемента буде містити інформацію цільового елемента. Це робить передбачення майбутнього тривіальним, і мережа не навчиться нічому корисному. Простим вирішенням цієї проблеми є створення $t-1$ вибірок (підпослідовностей з наступними елементами типу $([v_1], v_2)$ і $([v_1, v_2], v_3)$) з вихідної послідовності довжини t, а потім кодування кожної історичної підпослідовності за допомогою двонаправленної моделі для пророкування цільового елемента. Однак такий підхід вимагає великих витрат часу і ресурсів, оскільки нам необхідно створювати нову вибірку для кожної позиції в послідовності і передбачати їх окремо.

Щоб вирішити цю проблему, використовується метод, який у літературі часто називають задачею Cloze [7, 15]. Зокрема, ця процедура випадковим чином у вхідних послідовностях приховує деякі елементи, шляхом заміни їх спеціальним токеном [mask]. А потім прогнозує ідентифікатори цих прихованих елементів на основі їх навколишнього контексту. Таким чином, ми навчаємо модель, дозволяючи кожному елементу у вхідній послідовності використовувати лівий і правий контекст.

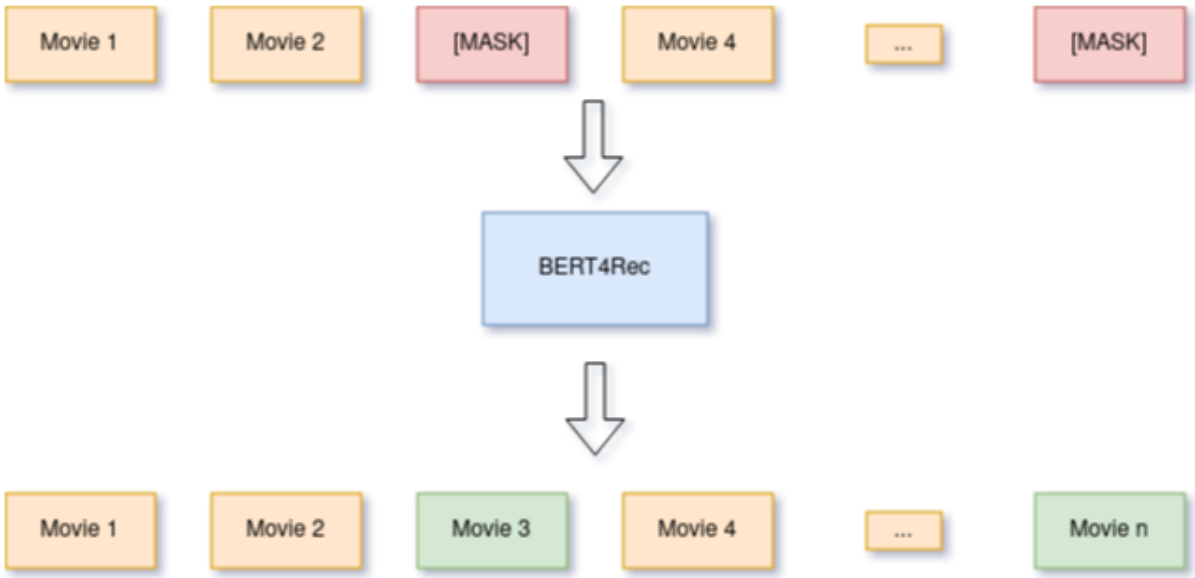
Кінцеві приховані вектори, які відповідають елементу [mask], подаються у вихідний softmax на множині елементів, як у звичайній послідовній рекомендації. В кінцевому підсумку, я визначаю втрати для кожного замаскованого входу S_u за допомогою негативної функції логарифмічної правдоподібності для замаскованих цілей:

$$\mathcal{L} = \frac{1}{|S_u^m|} \sum_{v_m \in S_u^m} -\log P(v_m = v_m^* | S_u') \qquad (8)$$

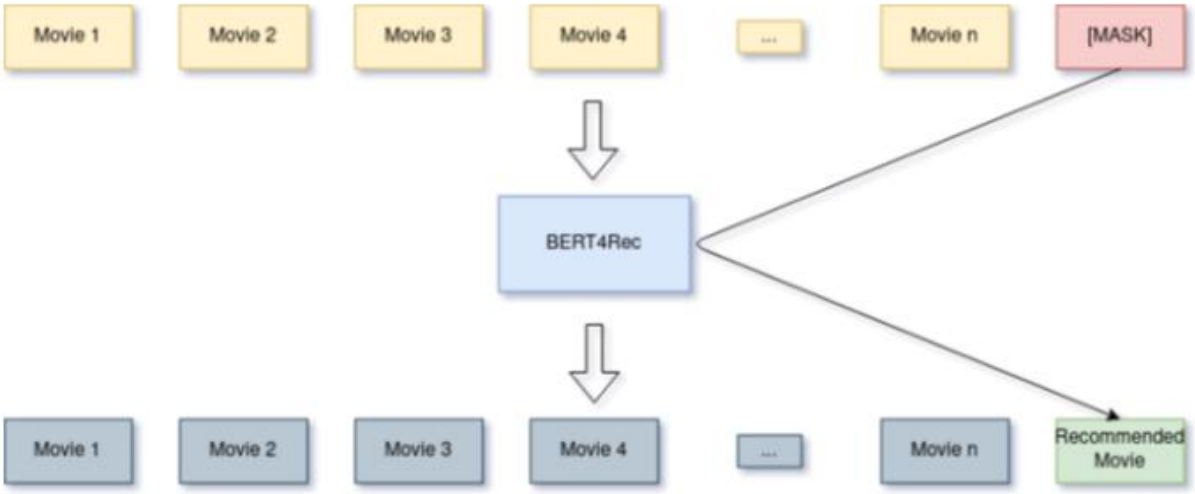
де S'_u – маскована версія для історії поведінки користувача S_u. S_u^m – випадкові замасковані елементи в ній, v_m^* – істинний елемент для замаскованого елемента v_m, а ймовірність $P(\cdot)$ визначена в рівнянні 7.

Крім навчання двонаправленної моделі, ще однією перевагою методу Cloze є можливість створювати більше зразків для навчання більш потужної моделі в кілька епох. Передбачаючи послідовність довжиною n, звичайні послідовні пророкування на рис. 1d дають n унікальних зразків для навчання, в той час як BERT4Rec може отримати n_k зразків (якщо ми випадковим чином приховуємо k елементів) за кілька епох. Це дозволяє нам навчити більш потужну модель двонаправленого представлення.

Однак зворотна сторона задачі Cloze полягає в тому, що вона не відповідає кінцевії цілі звичайної послідовної рекомендації. Тобто, вона не прогнозує наступний елемент. Щоб виправити це, додається спеціальний маркер [mask] в кінець вхідної послідовності, щоб визначити елемент, який потрібно передбачити. Принцип роботи метода Cloze зображений на малюнках 8 і 9.



мал 8


Movie 1
Movie 2
Movie 3
Movie 4
...
Movie n
[MASK]
BERT4Rec
Movie 1
Movie 2
Movie 3
Movie 4
...
Movie n
Recommended Movie

мал 9

Експеримент та результати

MovieLens - це популярний набір даних для оцінки алгоритмів рекомендацій. Для проведення експерименту використовуються два реальних та репрезентативных набора даних, які добре себе зарекомендували: MovieLens_1m і MovieLens_20m. Перший має 6040 юзерів, 3416 фільмів і щільність 4.79%. Другий набір містить 138493 юзерів, 26744 фільмів і має щільність 0.54%.

Сочатку проводиться попередня обробка наборів даних, слідуючи загальній практиці [2, 16]. Для двох наборів перетворюються всі числові оцінки або наявність відгуку в одиницю (тобто користувач взаємодіяв з фільмом). Після цього групуються записи взаємодії по користувачам і будується послідовність взаємодій для кожного користувача, сортуючи ці записи по часовим мітках. Для кожного користувача використовується останній елемент послідовності поведінки в якості тестових даних. Передостанній елемент як валідаційний набір, а решта елементів використовується для навчання. Далі кожний істинний елемент з тестового набору зіставляється зі ста випадково обраними «негативними» елементами, з якими користувач не взаємодіяв. Щоб зробити вибірку репрезентативною, ці сто негативних елементів відбираються відповідно до своєї популярності. Отже, завдання полягає в тому, щоб ранжувати ці негативні елементи з достовірним елементом для кожного користувача.

Для аналізу ефективності моделі використовуються три метрики оцінок, такі як Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG) та Mean Reciprocal Rank (MRR). З огляду на те, що використовується тільки один достовірний елемент для кожного користувача, HR еквівалентний Recall і пропорційний Precision. А MRR еквівалентний метриці Mean Average Precision (MAP). У цій роботі використовуються HR і NDCG з $k = 1, 5, 10$. Для цих двох метрик, чим вище значення, тим краще ефективність.

Модель BERT4Rec реалізується за допомогою модуля TensorFlow з Python. Встановлюється кількість шарів $L = 2$ і кількість заголовків(head)

$h = 2$. Також максимальна довжина послідовності $N = 200$ для кожного набору даних. Для настройки head, емпірично визначається розмірність кожного, яка дорівнює 32. Також налаштовується пропорція замаскованих елементів $p = 0{,}2$ у вхідній послідовності.

У таблиці №1 можна побачити, що модель BERT4Rec на двох наборах даних показує досить впевнені результати з точки зору обраних метрик оцінки.

Таблиця 1

	HR k=1	HR k=5	HR k=10	NDCG k=5	NDCG k=10	MRR
ML-1m	0.2863	0.5876	0.6970	0.4454	0.4818	0.4254
ML-20m	0.3440	0.6323	0.7473	0.4967	0.5340	0.4785

Висновки

У цій роботі було представлено глибоку двонаправлену послідовну модель під назвою BERT4Rec для послідовних рекомендацій фільмів. Для навчання моделі використовувався метод Cloze, який прогнозує замасковані елементи, використовуючи як лівий, так і правий контекст.

Експерименти з двома наборами даних різного об’єму свідчать, що ця модель стабільно показує високі результати, які перевершують багато сучасних базових підходів до вирішення проблеми послідовної рекомендації.

Також така двонаправленна модель дає можливості для подальших досліджень в області послідовної рекомендаційною системи фільмів. А саме модель можна вдосконалити включенням, наприклад, жанру для кожного фільму, або характеристик користувача.

Посилання

[1] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks.

[2] Wang-Cheng Kang and Julian McAuley. [n. d.]. Self-Attentive Sequential Recommendation.

[3] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation.

[4] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-based Recurrent Neural Network Recommendations.

[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks

[6] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[8] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations

[9] Doug Cairns and Xiangxiang Meng, SAS Institute Inc. 2020. NLP with BERT: Sentiment Analysis Using SAS® Deep Learning and DLPy.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need

[11] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition

[13] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting

[15] Wilson L. Taylor. 1953. âĂIJCloze ProcedureâĂİ: A New Tool for Measuring Readability

[16] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding.