НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

«На правах рукопису»
УДК 004.942.3

«До захисту допущено»
Завідувач кафедри ММСА
Тимощук О.Л.
«___»_____20__ р.

Магістерська дисертація
на здобуття ступеня магістра
зі спеціальності 124 Системний аналіз
на тему: « Рекомендаційні системи з використанням автоенкодерів та навчання з підкріпленням »

Виконав:
студент ІІ курсу, групи КА-02мп
Якубенко Олексій Петрович _____

Керівник:
директор ІПСА д.ф.-м.н.,
проф. Касьянов Павло Олегович _____

Рецензент:
професор кафедри інтегральних та диференціальних рівнянь
КНУ ім. Тараса Шевченка, д.ф.-м.н.,
проф. Капустян Олексій Володимирович _____

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

Київ
2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти – другий (магістерський)
Спеціальність – 124 «Системний аналіз»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.Л. Тимощук
(підпис)
«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Якубенку Олексію Петровичу

1. **Тема дисертації:** « Рекомендаційні системи з використанням автоенкодерів та навчання з підкріпленням », науковий керівник дисертації Касьянов Павло Олегович, доктор фізико-математичних наук, професор, затверджені наказом по університету від «02» листопада 2021 р. № 3651-с.
2. **Термін подання студентом дисертації** 12.12.2021
3. **Об'єкт дослідження**: рекомендаційні системи товарів.
4. **Предмет дослідження**: методи та алгоритми формування рекомендацій.
5. **Перелік завдань, які потрібно зробити:**
    1. Дослідити актуальність обраної теми та існуючих систем на ринку;
    2. Здійснити огляд технічної літератури за темою дослідження;
    3. Зробити дослідження та огляд найсучасніших алгоритмів покращення роздільної здатності зображень;
    4. Знайти та підготувати набори даних для навчання моделей та проведення аналізу;
    5. Розробити та протестувати систему, яка буде покращувати вхідні роздільну здатність зображення;
    6. Провести аналіз результатів;
    7. Провести аналіз ринкових можливостей запуску стартап проекту;
    8. Розробити концептуальні висновки;
    9. Підготувати ілюстративний матеріал;

10.Оформити пояснювальну записку.

6. Орієнтовний перелік ілюстративного матеріалу:

1. Методи та підходи рекомендаційних систем
2. Структура та робота нейронних мереж
3. Приклад результатів використання системи
4. Розрахункові таблиці з результатами

7. Дата видачі завдання: 1 вересня 2021 року.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Отримання завдання на магістерську дисертацію	01.09.2021 – 05.09.2021	Виконано
2.	Огляд технічної літератури за темою	06.09.2021 – 12.09.2021	Виконано
3.	Формулювання об’єкту, предмета, цілі, завдань, новизни, значущості результатів. Дослідження актуальності обраної теми	13.09.2021 – 19.09.2021	Виконано
4.	Перший розділ. Машинне навчання та комп’ютерний зір	20.09.2021 – 26.09.2021	Виконано
5.	Другий розділ. Методи розв’язку покращення якості зображення	27.09.2021 – 03.10.2021	Виконано
6.	Третій розділ. Система покращення роздільної здатності зображень	04.10.2021 – 10.10.2021	Виконано
7.	Розширення функціоналу системи	11.10.2021 – 17.10.2021	Виконано
8.	Аналіз результатів	18.10.2021 – 24.10.2021	Виконано
9.	Проведення аналізу ринкових можливостей стартап – проекту	25.10.2021 – 31.10.2021	Виконано
10.	Підготовка ілюстративного матеріалу	01.11.2021 – 14.11.2021	Виконано
11.	Оформлення пояснювальної записки	15.11.2021 – 26.11.2021	Виконано

Студент	_____	О. П. Якубенко
Науковий керівник дисертації	_____	П. О. Касьянов

РЕФЕРАТ

Магістерська дисертація виконана на 87 сторінках, містить 7 ілюстрацій, 21 таблиць 16 джерел та 1 додаток.

Об’єктом дослідження є рекомендаційні системи товарів.

Предметом дослідження є методи та алгоритми формування рекомендацій.

Актуальність роботи полягає в тому, що існує принципова проблема побудови рекомендаційних систем, а саме – проблема масштабованості. Наявність проблеми масштабованості вимагає від алгоритмів рекомендаційних систем можливість необмеженого нарощування числа користувачів, так і числа варіантів можливих рекомендацій. В той же час нарощування не повинно вимагати заміни у коді програми реалізації алгоритмів та повторної обробки всіх вже оброблених даних. Розробка алгоритмів, що вирішують цю проблему, дозволяє будувати рекомендаційні системи, які є адаптивними по відношенню до зростання кількості користувачів та товарів.

Мета дослідження є розробка рекомендаційної системи для вибору фільмів яку не потрібно повністю перенавчати з плином часу та додаванням нових фільмів.

Новизною роботи являється реалізація нейронної мережі яка може необмежено нарощувати варіанти рекомендації, та її навчання за допомогою методів навчання з підкріпленням.

ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, АВТОКОДУВАЛЬНИК, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ

ABSTRACT

The master's dissertation is made on 87 pages, contains 7 illustrations, 21 tables of 16 sources and 1 appendice.

The object of research is product recommendation systems.

The subject of research is methods and algorithms for forming recommendations.

The urgency of the work is that there is a fundamental problem of building recommendation systems, namely - the problem of scalability. The presence of the problem of scalability requires from the algorithms of recommendation systems the possibility of unlimited increase in the number of users and the number of options for possible recommendations. At the same time, the build should not require replacement in the code of the program implementation of algorithms and reprocessing of all already processed data. The development of algorithms that solve this problem allows you to build recommendation systems that are adaptable to the growing number of users and products.

The aim of the study is to develop a recommendation system for the selection of films that does not need to be completely retrained over time and the addition of new films.

The novelty of the work is the implementation of a neural network that can indefinitely increase the options for recommendation, and its training using reinforced learning methods.

DEEP NEAR NETWORKS, MACHINE LEARNING, AUTOENCODER, REINFORCED LEARNING, RECOMMENDATION SYSTEMS

ЗМІСТ

ВСТУП

Останні десятиліття характеризуються бурхливим розвитком мережі Інтернет: щодня у глобальному павутинні генеруються та накопичуються дуже великі обсяги інформації. Користувач працює з цими даними: обробляє, систематизує, знаходити релевантні для його інформаційних потреб дані. Людині складно відібрати цікаву для нього інформацію шляхом звичайного перегляду, оскільки часто релевантна інформація губиться серед надмірних обсягів даних. У зв'язку з цим, створюються інструменти, які можуть допомогти людині у пошуку, пропонуючи той контент, який є кращим для користувача. Такі програмні засоби отримали назву рекомендаційні системи.

Рекомендаційні системи – програмний продукт, який аналізує інтереси користувача та намагаються передбачити, що саме буде найбільш релевантне для конкретного користувача. Такі системи показують перевагу одного об’єкта рекомендації над іншим для конкретного користувача на основі даних, отриманих при взаємодії користувача з системою. Оскільки вподобання можуть значно відрізнятися у різних людей, рекомендаційна система повинна адаптуватися під конкретного користувача. Також деякі системи повинні мати змогу порекомендувати щось за межами типових інтересів конкретного користувача, для того щоб не замкнути його в інформаційній бульбашці. Все це робить ресурси, в яких використовуються рекомендаційні механізмами, привабливими для користувача. З іншого боку, подібні системи цікавлять і власників самих ресурсів, на яких розміщуються об’єкти рекомендації, так як за допомогою рекомендаційних систем підвищується привабливість самого ресурсу та його послуг.

Рекомендаційні системи знайшли своє застосування у багатьох сферах життєдіяльності людини: пошуку фільмів та наукових статей, роздрібної торгівлі, соціальних мережах, електронної комерції, онлайн-банкінгу і т.д.

РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ В РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ

1.1 Актуальність задачі надання рекомендацій

Одним з бурхливих напрямків удосконалення індустрії електронної комерції є розгортання рекомендаційних систем - інструментів автоматичної генерації пропозицій щодо послуг на основі вивчення персональних потреб клієнтів. Одними з перших рекомендаційних систем були система компанії Google і система Інтернет-магазину Amazon. У 1992 р. як основний алгоритм для рекомендаційних систем було запропоновано метод колаборативної фільтрації. Він заснований на використанні у рекомендаційній системі інформації про доступні дії всіх користувачів. Цей метод дозволяв вирішити завдання рекомендації достатньо ефективно.

Аналіз публікацій показує, що незважаючи на це, залишається низка принципових проблем побудови рекомендаційних систем, які вимагають розробки нових алгоритмів, специфічних для цього класу систем обробки даних. Перша з цих проблем зветься проблеми масштабованості, а друга - «холодного старту». Наявність проблеми масштабованості вимагає від алгоритмів рекомендаційних систем можливість необмеженого нарощування числа користувачів, так і числа варіантів можливих рекомендацій. При цьому нарощування не повинно вимагати заміни у коді програми реалізації алгоритмів та повторної обробки всіх вже оброблених даних та допускати високу паралельність обчислень. Проблема «холодного старту» полягає у необхідності забезпечення працездатності алгоритму для генерації рекомендацій користувачам, які вперше увійшли до системи, та інформація про них відсутня чи надзвичайно бідна. Розробка алгоритмів, що вирішують ці проблеми, дозволяє будувати рекомендаційні

системи, які є адаптивними по відношенню до зростання кількості користувачів та товарів (в електронній комерції) та обсягу даних, що є про користувачів.

Таким чином, потрібно розробити алгоритми генерування рекомендацій шляхом вибору релевантних із заданої множини, що мають прийнятну обчислювальну складність. На сьогоднішній день дослідження та розробка таких алгоритмів залишається відкритим завданням.

1.2 Аналіз існуючих підходів щодо надання рекомендацій

Рекомендаційні системи – одна з найпопулярніших додатків інтелектуального аналізу даних та машинного навчання в сфери інтернет-бізнесу. Рекомендаційна система намагається аналізувати поведінку користувачів інтернет-ресурсу, після чого оцінює вподобання користувача для того чи іншого об'єкта рекомендацій. Об'єктами рекомендацій може бути будь-що: фільми, книги, товари в інтернет-магазині, інші користувачі веб-ресурсу.

Рекомендаційні системи допомагають користувачам орієнтуватися у великій кількості контенту, розміщеного на сайті. У деяких випадках це потрібна функціональність.

Можна виділити три основні підходи до побудови рекомендаційних систем :

1) на підставі ознакових описів (content-based);
2) колаборативна фільтрація (collaborative filtering);
3) гібридний підхід.

Content-based. Підхід на підставі ознакових описів припускає, що про користувачів та про рекомендовані об'єкти відомо досить багато інформації. Наприклад, всі користувачі заповнюють анкету, в якій вказують інформацію про себе та свої вподобання. Про товари з інтернет-магазину може бути наявна така інформація як: опис, призначення, цінова категорія, бренд та інші характеристики. По історії взаємодії користувачів та об'єктів на сервісі можна побудувати навчальну вибірку та звести передбачення переваги до добре вивченої задачі навчання з прецедентів.

Насправді, використання такого походу дуже обмежена, т.к. збір описової інформації про користувачів та об'єкти важко автоматизувати, тому це дуже дорога процедура, зазвичай впровадження такої системи збору інформації шкодить якості використання сервісу, що робить рекомендаційну систему невиправданою.

Колаборативна фільтрація. Колаборативною фільтрацією називається передбачення ступеня переваги в умовах, коли рекомендаційна система не має будь-якої описової інформацією про користувачів та об'єкти (або не використовує), будує прогноз виключно на підставі взаємодії користувачів з об'єктами.

Величезним поштовхом у дослідженні математичних моделей колаборативної фільтрації послужив конкурс Netflix Prize[3]. Компанія Netflix займається інтернет-прокатом фільмів. Метою конкурсу було поліпшення якості прогнозованої оцінки користувача деякому фільму. Набір даних конкурсу містив четвірки (дата-час, користувач, фільм, оцінка). Оцінка вимірювалася від 1 до 5.

Гібридний підхід використовує композиції колаборативної фільтрації та алгоритмів заснованих на ознакових описах. Незважаючи на те, що алгоритми колаборативної фільтрації демонструють високі результати, аналіз додаткової

інформації може зробити показники ще кращими. Основним недоліком колаборативної фільтрації в порівнянні з методами, що ґрунтуються на ознаковому описі, є проблема холодного старту.

Зворотним зв'язком (feedback) користувача на об'єкт рекомендаційних системах прийнято називати дію користувача, за якою можна судити про вподобання користувача що до об'єкта. Наприклад:

- натискання на кнопку "подобається" або "не подобається";
- проставлення оцінки об'єкту;
- відвідування сторінки з об'єктом, клік на посиланням з об'єктом;
- повторне відвідування сторінки з об'єктом;
- додавання до обраного або кошика;
- купівля об'єкта.

Саме за допомогою зворотного зв'язку користувача для різних об'єкти, система формує матрицю оцінок переваг R, до якою потім застосовуються алгоритми колаборативної фільтрації. Перетворення зворотного зв'язку на числове значення переваги непросте і дуже важливе завдання у створенні рекомендаційних систем. При виборі схеми за якою визначається оцінка переваги оптимізується критерій, безпосередньо пов'язаний з бажаннями бізнесу.

За наявністю зворотного зв'язку, рекомендаційні системи прийнято розділяти на два види:

1) з явним зворотним зв'язком (explicit feedback);
2) з неявним зворотним зв'язком (implicit feedback).

Так, наприклад, рекомендації за оцінками із десятибальної шкали – приклад задачі з явним зворотним зв'язком. Рекомендаційні системи керовані актами

відвідуванням сторінок, кліків, покупок, - приклади завдань з неявним зворотним зв'язком.

У разі неявного зворотного зв'язку є похибка у тому, позитивно чи негативно впливає конкретний зворотній зв'язок на ступінь переваги. Покупка товару в інтернет-магазині в більшості випадків означає досягнення користувачем своєї мети (позитивну перевагу), але покупець міг після отримання товару в ньому розчаруватися, написати негативний коментар і правильно було б зарахувати негативний ступінь вподобання. Очевидно, що відвідування сторінок користувачами вебсервісу можуть відбуватися при абсолютно різному ступені зацікавленості користувача у контенті сторінок. Потрібно відзначити достатньо типову ситуацію, коли рекомендаційна система подається на вхід виключно позитивні приклади взаємодії користувачів та об'єктів. Веб-сервіс Twitter не має функціональності, що дозволяє висловити свою негативну оцінку контенту, а є тільки спосіб "заохотити" той чи інший контент, поширивши його своїм передплатникам у вигляді функції “репост”. Подібний зворотний зв'язок користувача дуже надійний (порівняно з рештою) вказує на позитивний ступінь переваги. Надійність "репостів" у сервісі Twitter підкріплена відповідальністю користувачів перед своїми передплатниками.

1.3 Особливості підтримки актуальних рекомендацій

Існує декілька факторі через які рекомендаційні системи втрачають свою актуальність. Розглянемо деякі з них.

З плином часу змінюються тенденції, тренди, сезонні товари. Тому рекомендаційні системи з часом втрацають свою актуальність, що призводить до

того, що їм стає необхідним повне перенавчання щоб вивчити нові тренди та тенденції.

Також з часом додаються нові товари які можна рекомендувати, але якщо рекомендаційна система навчалась на вибірці в якій їх не було то це також призводить до того, що систему потрібно повністю перенавчати.

Коли ми розмовляємо про такі платформи як онлайн кінотеатри, то в їх випадку навчання рекомендаційної системи може бути дуже важким з точки зору обчислень процесом. Проте такі платформи хочуть щоб їх системи були дуже актуальними. Це призводить до того що вони витрачають багато грошей на підтримку постійного перенавчання своїх рекомендаційних систем.

1.4 Висновки за розділом

В розділі було проаналізовано актуальність задачі формування рекомендацій, виявлено проблеми та можливі напрямки досліджень в цій галузі. Було розглянуто основні методи та особливості рекомендаційних систем, їхні переваги для впровадження в бізнесі.

Також було розглянуто основні принципи існуючих методів рекомендаційних систем, та зроблено їх аналіз в контексті підтримки актуальності рекомендацій.

РОЗДІЛ 2 ОГЛЯД ОСНОВИХ ПОНЯТЬ І МЕТОДІВ В РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ ТА НАВЧЧАННІ З ПІДКРІПЛЕННЯМ

2.1 Методи рекомендаційних систем

2.1.1 Критерії оцінки

Оцінка якості рекомендаційних систем зводиться до порівняння двох списків: списку рекомендацій та списку релевантних для користувача об'єктів. Список релевантних для користувача об'єктів має нефіксовану довжину, яка змінюється від користувача до користувача. Тому, порівнюючи його зі списком рекомендацій, який має постійну довжину k, важливо врахувати не тільки збіг елементів у списках, але й порядок об'єктів у списку рекомендацій: релевантні повинні бути вище нерелевантні.

Введемо основні критерії якості [1]. Метрика recall@k, що обчислюється за формулою (1), вважає частку вірно вгаданих рекомендацій серед об'єктів релевантних користувачеві. Метрика map@k – формула (2), крім оцінки релевантності рекомендацій, враховує порядок об'єктів у списку рекомендацій, штрафує, якщо нерелевантний об'єкт перебуває вище, ніж релевантний.

$$recall@k = \frac{\sum_{u \in U} \sum_{i=1}^{k} r_u(i)}{\sum_{u \in U} min(l_u, k)} \tag{1}$$

$$precision_u@k = \frac{1}{k} \sum_{i=k}^{k} r_u(i)$$

$$map@k = \frac{1}{|U|} \sum_{u \in U} \frac{1}{min(l_u, k)} \sum_{i=1}^{k} r_u(i) precision_u@i \tag{2}$$

де lu - кількість релевантних для користувача об'єктів, ru(i) - релевантний для користувача об'єкт чи ні (1 або 0).

Крім того, для оцінки якості рекомендацій товарів було введено два критерії «жорсткий» та «м'який».

"Жорсткий" критерій:

r(i) = 0, якщо товар і не релевантний користувачеві;

r(i) = 1, якщо товар і релевантний користувачеві.

"М'який" критерій

r(i) = 0, якщо категорія товару і не релевантна користувачеві;

r(i) = 1, якщо категорія товару і релевантна користувачеві.

Критерії введені через те, що в інтернет-магазині є широкий вибір товарів — кілька мільйонів різних пропозицій, але при цьому більшість товарів об'єднуються в групи, відрізняючись лише деякими властивостями. Тому вгадати точну перевагу користувача досить складно, але при цьому модель може добре пророкувати спільні інтереси користувача.

2.1.2 Класичні методи рекомендаційних систем

2.1.2.1 Колаборативна фільтрація

Цей клас систем почав активно розвиватися у 90-ті роки. У рамках цього методу рекомендації генеруються виходячи з вподобань інших схожих користувачів. Ця рекомендація є результатом «колаборації» багатьох користувачів. [2]

Класична реалізація [3] алгоритму полягає в принципі k найближчих сусідів. Для кожного користувача шукаємо найбільш схожих на нього (в термінах переваг) і доповнюємо інформацію про користувача відомими даними по його сусідах. Так, наприклад, якщо відомо, що ваші сусіди за інтересами у захваті від фільму «Кров та бетон», а ви його з якоїсь причини ще не дивилися, це чудова нагода запропонувати вам цей фільм для суботнього перегляду.

На малюнку (див. рис 2.1) проілюстровано принцип роботи методу. У матриці переваг жовтим кольором виділено користувач, для якого ми хочемо визначити оцінки нових товарів (знаки питання). Синім кольором виділено три його найближчі сусіди.

"Схожість" - в даному випадку синонім "кореляції" інтересів і може вважатися безліччю способів (крім кореляції Пірсона, є ще косинусне відстань, є відстань Жаккара, відстань Хеммінга та ін.).

У класичної реалізації алгоритму є один явний мінус - він погано застосовується на практиці через квадратичну складність. Як і будь-який метод який базується на найближчих сусідах, він потребує розрахунку всіх попарних відстаней між кожною парою користувачів. Легко порахувати, що складність

розрахунку матриці відстаней буде O(n^2m), де m – число товарів, а n – число користувачів.

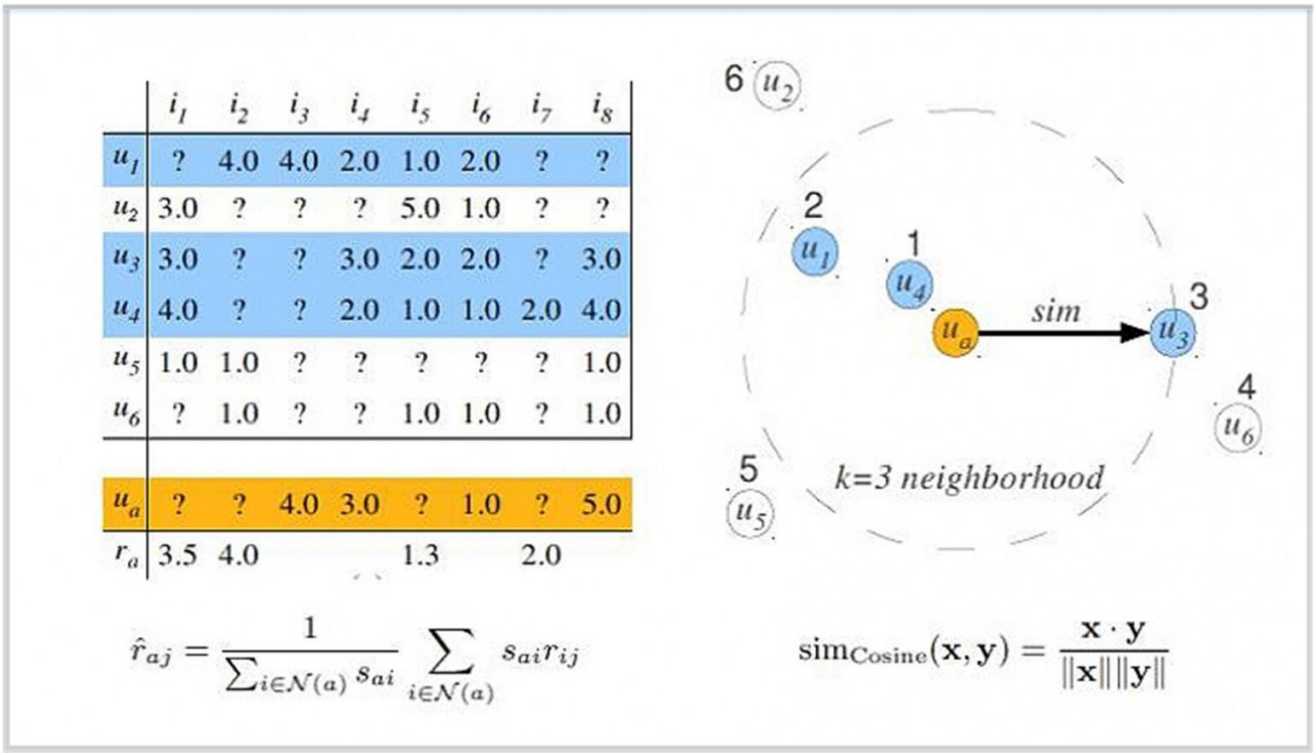

Рисунок 2.1 – Колаборативна фільтрація

Ця проблема може бути вирішена покупкою великих обчислювальних ресурсів. Але можливо ввести коригування в алгоритм:

- оновлювати відстані батчами(наприклад, щодня), а не при кожній покупці;
- оновлювати матрицю відстаней інкрементально;
- використовувати ітеративні та наближені алгоритми (наприклад ALS).

Для того щоб колаборативна фільтрація була ефективною, важливо, щоб виконувалося наступні умови:

- вподобання людей не змінюються з часом;

– якщо уподобання людей частково збігаються , то вони мають збігаються у всьому.

Наприклад, якщо два клієнти віддають перевагу одним фільмам, то книги їм теж подобаються однакові. Так часто буває, коли товари однорідні (наприклад, тільки фільми). Якщо ж це не так, то у пари клієнтів цілком можуть збігатися переваги в їжі, а політичні погляди бути прямо протилежними — тут алгоритм буде менш ефективним.

Околиця користувача у просторі переваг (його сусіди), яку ми аналізуватимемо для генерації нових рекомендацій, можна обирати по-різному. Ми можемо працювати взагалі з усіма користувачами системи, можемо задати якийсь поріг близькості, можемо вибрати кілька сусідів випадковим чином або брати найбільш схожих сусідів (це найбільш популярний підхід).

2.1.3 Рекомендаційні системи з використанням глибокого навчання

2.1.3.1 Автокодувальники

Автокодувальник— це тип штучної нейронної мережі, яка використовується для вивчення ефективного кодування нерозмічених даних (навчання без вчителя). Кодування перевіряється та удосконалюється шляхом спроби відновити вхідні дані з кодування. Автокодувальник вивчає представлення (кодування) для набору даних, як правило, для зменшення розмірності, навчаючи мережу ігнорувати незначні дані («шум»).

Існують варіанти, які мають на меті змусити вивчені уявлення набути корисних властивостей. Прикладами є регуляризовані автокодувальники (розріджуючі, знижуючи шум), які ефективні при навчанні представлень для майбутніх завдань класифікації, та варіаційні автокодери які є ефективними як генеративні моделі. Автокодувальники застосовуються до багатьох проблем, від розпізнавання обличчя, виявлення особливостей, виявлення аномалій до аналізу значення слів. Автокодувальники також є генеративними моделями: вони можуть випадковим чином генерувати нові дані, подібні до навчальних даних.[4]

Автокодувальник має дві основні частини: кодер, який відображає вхідний сигнал у код, і декодер, який відновлює вхідний сигнал на основі коду.

Найпростішим способом ідеального виконання завдання копіювання було б дублювати сигнал. Натомість автокодувальники зазвичай змушені приблизно реконструювати вхідні дані, зберігаючи лише найбільш релевантні аспекти даних у копії.

Ідея автокодувальників популярна протягом десятиліть. Перші застосування датуються 1980-ми роками. Їх найбільш традиційним застосуванням було

зменшення розмірності або вивчення ознак, але ця концепція стала широко використовуватися для генеративних моделей. Деякі з найпотужніших штучних інтелектів у 2010-х роках включали автокодувальники, розташовані всередині глибоких нейронних мереж.

Найпростіша форма автокодувальника (див. рис. 2.2) – це нерекурентна нейронна мережа з прямим зв’язком, подібна до одношарових персептронів, які використовуються у багатошарових персептронах (MLP) – з використанням вхідного і вихідного шарів, з’єднаних одним або кількома прихованими шарами. Вихідний шар має таку ж кількість вузлів (нейронів), що і вхідний шар. Його мета полягає в тому, щоб відновити його вхідні дані (мінімізуючи різницю між входом і виходом) замість того, щоб передбачати цільове значення. Тому автокодери навчаються без вчителя.

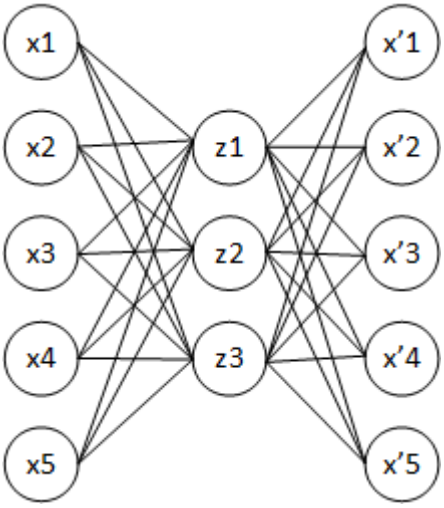

Рисунок 2.2 – Простий автокодувальник

Найпростіша модель (після мультишарового персептрону) глибокого навчання для рекомендаційних систем. На вхід отримує вектор об’єктів рекомендації з якими взаємодіяв юзер, кодує його і намагається відновити тим саме генеруючи нові об’єкти з якими може про взаємодіяти юзер. Можна

“приховувати” від вхідного шару деякі перегляди і на основі цього отримати більші узагальнюючі властивості автокодувальника, а також вимірювати ефективність. [5] [6]

2.1.3.2 Нейронна Колаборативна фільтрація

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

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

Перевага цього підходу полягає у нелінійності багатошарового перцептрону. Просте твір матриць, що використовується при матричній факторизації, завжди обмежуватиме модель взаємодіями 2-го ступеня, тоді як нейронна мережа з X шарами в теорії може засвоїти взаємодії набагато більших ступенів. Уявіть собі,

наприклад, три якісні змінні, всі взаємодії, що мають, один з одним – наприклад, чоловіча стать, підлітковий вік і рольові комп'ютерні ігри.

У реальних завданнях ми не просто використовуємо двійкові вектори, що відповідають користувачеві та об'єкту як необроблені дані для їх подання, але, очевидно, включаємо й іншу додаткову або мета-інформацію, яка може представляти цінність (наприклад, вік, країну, аудіо- та текстові) записи, мітки часу,...), так що в реальності у нас виходить набір даних дуже високої розмірності, сильно розріджений і суміш якісних змінних з кількісними. На цей момент представлену нижче (див. рис 2.3) нейронну мережу можна вважати рекомендацією на основі вмісту у вигляді простого нейронного бінарного класифікатора прямого проходу. І ця інтерпретація – ключова для розуміння того, чому цей підхід зрештою виявляється гібридним між колаборативною фільтрацією та рекомендаціями на основі вмісту. Ця нейронна мережа може засвоїти будь-які функціональні залежності - наприклад, взаємодії 3-го ступеня і вище, на зразок x1 * x2 * x3, або будь-які нелінійні трансформації в класичному розумінні, що використовується в нейронних класифікаторах - у вигляді sigma (.. sigma (w1x1 + w2x2) +w3x3+b)).

Озброївшись силою вивчення залежностей вищого порядку, ми можемо використовувати простий спосіб вивчення залежностей 1-го та 2-го порядків, скомбінувавши нейронну мережу з широко відомою моделлю для їхнього вивчення, Машиною Факторизації. Саме це автори DeepFM запропонували у своїй статті. Ідея цієї комбінації – паралельно вивчати залежності між ознаками низьких та високих порядків – ключова частина багатьох сучасних рекомендаційних систем, і в тому чи іншому вигляді її можна знайти майже в кожній архітектурі нейронної мережі, що пропонується в цій галузі. [7]

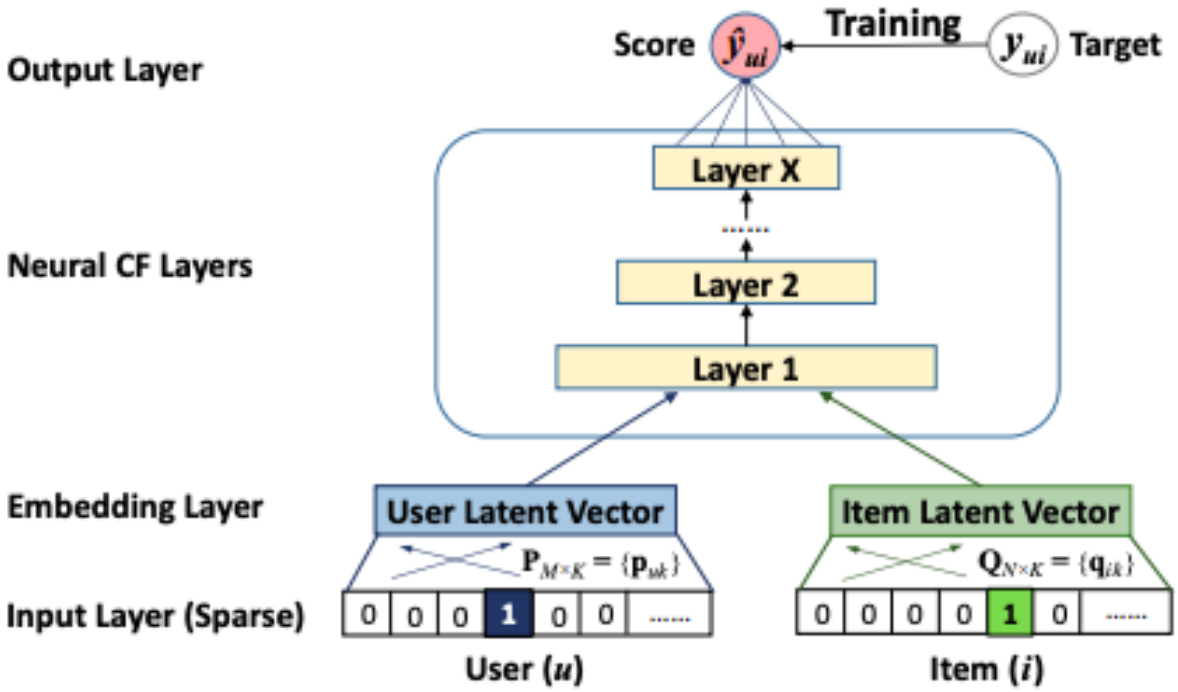

Рисунок 2.3 – Схема NCF

2.2 Задача навчання з підкріпленням

2.2.1 Ключові поняття та термінологія

Головними героями РЛ є агент і оточення. Навколишнє середовище — це світ, у якому агент живе і з яким взаємодіє. На кожному кроці взаємодії агент бачить (можливо, часткове) спостереження за станом світу, а потім приймає рішення про дію, яку слід зробити (див. рис 2.4). Середовище змінюється, коли агент діє на нього, але також може змінюватися самостійно. [8]

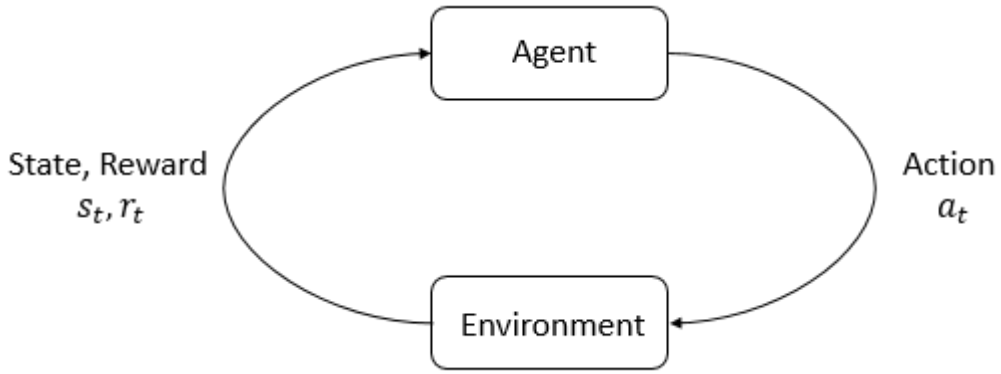

Рисунок 2.4 – Схема Агент-Середовище

Агент також сприймає сигнал винагороди з навколишнього середовища, число, яке говорить йому, наскільки хороший чи поганий поточний стан світу. Мета агента — максимізувати свою сукупну винагороду, що називається поверненням. Методи навчання з підкріпленням – це способи, за допомогою яких агент може навчитися поведінці для досягнення своєї мети.

Щоб говорити більш конкретно, що робить RL, нам потрібно ввести додаткову термінологію. Треба поговорити про:

- стани та спостереження;
- простори дії;
- політика;
- траєкторії;
- різні формулювання повернення;
- проблема оптимізації RL;
- і функції цінності.

2.2.2 Стан і спостереження

Стан s — це повний опис стану світу. Немає жодної інформації про світ, яка була б прихована від держави. Спостереження o є частковим описом стану, в якому може бути пропущена інформація.

У глибокій RL ми майже завжди представляємо стани та спостереження за допомогою вектора з дійсним значенням, матриці або тензора вищого порядку. Наприклад, візуальне спостереження може бути представлене матрицею RGB зі значеннями пікселя; стан робота може бути представлено його спільними кутами та швидкостями. [10]

Коли агент може спостерігати повний стан середовища, ми говоримо, що середовище повністю спостерігається. Коли агент може бачити лише часткове спостереження, ми говоримо, що навколишнє середовище спостерігається частково.

Нотація для навчання з підкріпленням іноді розміщує символ стану s у місцях, де технічно було б доцільніше написати символ для спостереження, o. Зокрема, це відбувається, коли ми говоримо про те, як агент вирішує дію: ми часто сигналізуємо в нотації, що дія зумовлена станом, коли на практиці дія зумовлена спостереженням, оскільки агент не має доступу до стану. [9]

2.2.3 Простір дій

Різне середовище допускає різні види дій. Сукупність усіх дійсних дій у даному середовищі часто називають простором дій. Деякі середовища, такі як Atari і Go, мають дискретні простори дій, де агенту доступна лише кінцева кількість ходів. Інші середовища, наприклад, де агент керує роботом у фізичному світі, мають простори безперервних дій. У безперервних просторах дії є векторами з дійсними значеннями.

Ця відмінність має деякі досить глибокі наслідки для методів глибокого RL. Деякі сімейства алгоритмів можуть бути безпосередньо застосовані лише в одному випадку, а для іншого їх доведеться суттєво переробити. [8] [11]

2.2.4 Політики

Політика — це правило, яке використовується агентом, щоб вирішувати, які дії слід зробити. Він може бути детермінованим, і в цьому випадку його зазвичай позначають μ або він може бути стохастичним, і в цьому випадку його зазвичай позначають $\boldsymbol{\pi}$:

$$a_t = \mu(s_t),$$

$$a_t \sim \pi(\cdot|s_t).$$

Оскільки політика, по суті, є мозком агента, нерідко можна замінити слово «політика» на «агент», наприклад, кажучи: «Політика намагається максимізувати винагороду».

У глибокій RL ми маємо справу з параметризованими політиками: політиками, вихідними результатами яких є обчислювані функції, які залежать від набору параметрів (наприклад, ваг і зміщень нейронної мережі), які ми можемо налаштувати, щоб змінити поведінку за допомогою певного алгоритму оптимізації. [8]

Ми часто позначаємо параметри такої політики як theta або phi, а потім пишемо це як індекс на символі політики, щоб виділити з'єднання:

$$a_t = \mu_\theta(s_t)$$
$$a_t \sim \pi_\theta(\cdot|s_t).$$

2.2.5 Детерміновані політики

Приклад: детерміновані політики [8]. Ось фрагмент коду для створення простої детермінованої політики для безперервного простору дій у PyTorch, використовуючи пакет torch.nn:

```
pi_net = nn.Sequential(
            nn.Linear(obs_dim, 64),
            nn.Tanh(),
            nn.Linear(64, 64),
            nn.Tanh(),
            nn.Linear(64, act_dim)
          )
```

Це створює багатошарову мережу персептрона (MLP) з двома прихованими шарами розміром 64 і функціями активації tanh. Якщо obs є масивом Numpy, що містить пакет спостережень, pi_net можна використовувати для отримання пакету дій наступним чином:

```
obs_tensor = torch.as_tensor(obs, dtype=torch.float32)
actions = pi_net(obs_tensor)
```

2.2.6 Стохастичні політики

Двома найпоширенішими типами стохастичних політик у глибокій RL є категоричні політики та діагональні політики Гаусса.

Категоричні політики можна використовувати в просторах дискретних дій, тоді як діагональні політики Гаусса використовуються в просторах безперервних дій.

Два ключових обчислення мають центральне значення для використання та навчання стохастичних політик:

- вибіркові дії з політики;
- і обчислення журналу ймовірності певних дій, $\log \pi_\theta(a|s)$.

Далі ми опишемо, як це зробити для категорійної та діагональної політики Гаусса.

Категорична політика схожа на класифікатор дискретних дій. Ви будуєте нейронну мережу для категорійної політики так само, як і для класифікатора: вхід – це спостереження, за яким слідує деяка кількість шарів (можливо згорткових або щільно пов’язаних, залежно від типу вхідних даних), а потім у вас є один останній

лінійний шар, який надає логіти для кожної дії, за яким слід softmax для перетворення логітів у ймовірності. [9]

Відбір проб. Враховуючи ймовірності для кожної дії, такі фреймворки, як PyTorch і Tensorflow, мають вбудовані інструменти для вибірки. Наприклад, дивіться документацію щодо категорійних розподілів у PyTorch, torch.multinomial, tf.distributions.Categorical або tf.multinomial.

Лог-вірогідність. Позначимо останній рівень ймовірностей як P_theta(s). Це вектор з будь-якою кількістю записів, скільки дій, тому ми можемо розглядати дії як індекси для вектора. Журнал ймовірності дії a може бути отримано шляхом індексації у векторі:

$$\log \pi_\theta(a|s) = \log \left[P_\theta(s)\right]_a .$$

Багатовимірний гауссовий розподіл (або багатовимірний нормальний розподіл, якщо ви віддаєте перевагу) описується середнім вектором, mu, та матрицею коваріації Sigma. Діагональний гауссовий розподіл є окремим випадком, коли коваріаційна матриця має лише записи на діагоналі. В результаті ми можемо представити його вектором.

Діагональна політика Гаусса завжди має нейронну мережу, яка відображає спостережень до середніх дій, mu_{theta}(s). Існує два різні способи представлення коваріаційної матриці.

Перший спосіб: існує єдиний вектор журналу стандартних відхилень, log sigma, який не є функцією стану: log sigma є окремими параметрами. (Ви повинні знати: наші реалізації VPG, TRPO і PPO роблять це таким чином.)

Другий спосіб: існує нейронна мережа, яка відображає стани в журнал стандартних відхилень, log sigma_{theta}(s). Він за бажанням може спільно використовувати деякі шари із середньою мережею.

Зауважте, що в обох випадках ми виводимо журнал стандартних відхилень замість стандартних відхилень безпосередньо. Це пояснюється тим, що log stds можуть вільно приймати будь-які значення в (-\infty, \infty), тоді як stds повинні бути невід'ємними. Навчити параметри легше, якщо вам не потрібно застосовувати такі види обмежень. Стандартні відхилення можна отримати відразу з журналу стандартних відхилень шляхом їх збільшення, тому ми нічого не втрачаємо, представляючи їх таким чином.

Відбір проб. Враховуючи середню дію mu_{theta}(s) і стандартне відхилення sigma_{theta}(s), і вектор z шуму від сферичного гаусса (z ~ N(0, I)), можна обчислити зразок дії:

$$a = \mu_\theta(s) + \sigma_\theta(s) \odot z,$$

де Θ позначає поелементний добуток двох векторів. Стандартні фреймворки мають вбудовані способи створення векторів шуму, наприклад torch.normal або tf.random_normal. Крім того, ви можете створювати об’єкти розподілу, наприклад, через torch.distributions.Normal або tf.distributions.Normal, і використовувати їх для створення зразків. (Перевага останнього підходу полягає в тому, що ці об’єкти також можуть обчислювати для вас логарифмічні ймовірності.)

Лог-вірогідність. Лог-вірогідність k-вимірної дії a для діагонального гаусса із середнім значенням mu = mu_{theta}(s) і стандартним відхиленням sigma = sigma_{theta}(s) визначається як:

$$\log \pi_\theta(a|s) = -\frac{1}{2}\left(\sum_{i=1}^{k}\left(\frac{(a_i-\mu_i)^2}{\sigma_i^2} + 2\log\sigma_i\right) + k\log 2\pi\right).$$

2.2.7 Траєкторія

Траєкторія tau - це послідовність станів і дій у світі,

$$\tau = (s_0, a_0, s_1, a_1, ...).$$

Найперший стан світу, s_0, випадковим чином відбирається з розподілу початкового стану, який іноді позначається rho_0:

$$s_0 \sim \rho_0(\cdot).$$

Переходи станів (що відбувається зі світом між станом у момент t, s_t і станом у момент t+1, s_{t+1}) регулюються природними законами навколишнього середовища і залежать лише від останніх дія, a_t. Вони можуть бути детермінованими,

$$s_{t+1} = f(s_t, a_t)$$

або стохастичний,

$$s_{t+1} \sim P(\cdot|s_t, a_t).$$

Дії походять від агента відповідно до його політики.

Траєкторії також часто називають епізодами або розгортанням. [8]

2.2.8 Нагорода

Функція винагороди R є критично важливою в навчанні з підкріпленням. Це залежить від поточного стану світу, щойно вжитої дії та наступного стану світу:

$$r_t = R(s_t, a_t, s_{t+1})$$

хоча часто це спрощується лише до залежності від поточного стану, r_t = R(s_t) або пари стан-дія r_t = R(s_t,a_t).

Мета агента — максимізувати деяке уявлення про кумулятивну винагороду за траєкторію, але насправді це може означати кілька речей. Ми позначимо всі ці випадки за допомогою R(tau), і з контексту буде або зрозуміло, який випадок ми маємо на увазі, або це не матиме значення (оскільки до всіх випадків застосовуватимуться ті самі рівняння).

Одним із видів прибутку є дохід без дисконтування на кінцевому горизонті, який є лише сумою винагород, отриманих за фіксоване вікно кроків:

$$R(\tau) = \sum_{t=0}^{T} r_t.$$

Інший вид прибутку — це прибуток зі знижкою на нескінченному горизонті, який є сумою всіх винагород, коли-небудь отриманих агентом, але дисконтованих від того, наскільки далеко вони будуть отримані в майбутньому. Ця формулювання винагороди включає коефіцієнт дисконтування gamma в (0,1):

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

Але навіщо нам коли-небудь використовувати коефіцієнт дисконтування? Чи не хочемо ми просто отримати всі нагороди? Ми так, але фактор знижки інтуїтивно привабливий і математично зручний. На інтуїтивному рівні: готівка зараз краще, ніж готівка пізніше. Математично: сума винагород за нескінченний горизонт може не сходитися до кінцевого значення, і з цим важко мати справу в рівняннях. Але з коефіцієнтом дисконту та за розумних умов нескінченна сума збігається. [8]

Хоча межа між цими двома формулюваннями прибутковості досить чітка у формалізмі RL, глибока практика RL має тенденцію до розмиття межі — наприклад, ми часто встановлюємо алгоритми для оптимізації недисконтованого прибутку, але використовуємо коефіцієнти знижок при оцінці функцій вартості.

2.2.9 Проблема Навчання з Підкріпленням

Незалежно від вибору міри прибутку (незалежно від того, чи буде зі знижкою на нескінченний горизонт, або без дисконтування за кінцевий горизонт), і яким би не був вибір політики, мета в RL полягає в тому, щоб вибрати політику, яка максимізує очікувану віддачу, коли агент діє відповідно до неї. [9]

Щоб говорити про очікувану прибутковість, ми спочатку повинні поговорити про розподіл ймовірностей за траєкторіями.

Припустимо, що і зміни середовища, і політика стохастичні. У цьому випадку ймовірність траєкторії T-кроку становить:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t).$$

Очікувана віддача (для будь-якого показника), що позначається J(\pi), буде тоді:

$$J(\pi) = \int_{\tau} P(\tau|\pi)R(\tau) = \underset{\tau\sim\pi}{\mathrm{E}}\left[R(\tau)\right].$$

Тоді центральна задача оптимізації в RL може бути виражена як:

$$\pi^* = \arg\max_{\pi} J(\pi),$$

з pi* оптимальною політикою.

2.2.10 Функції цінності

Часто корисно знати значення стану або пари стан-дія. Під значенням ми маємо на увазі очікувану віддачу, якщо ви починаєте в цьому стані або парі стан-дія, а потім продовжуєте діяти відповідно до певної політики. Функції значень використовуються, так чи інакше, майже в кожному алгоритмі RL. [8]

Тут варто відзначити чотири основні функції.

1. Функція внутрішньополітичного значення V^pi(s), яка дає очікувану віддачу, якщо ви починаєте в стані s і завжди дієте відповідно до політики pi.

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[R(\tau)\,|s_0 = s\right]$$

2. Функція значення дії в політиці, Q^pi(s,a), яка дає очікувану віддачу, якщо ви починаєте в стані s, виконуєте довільну дію a (яке, можливо, не походить від політики), і потім назавжди діяти відповідно до політики pi.

$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[R(\tau)\,|s_0 = s, a_0 = a\right]$$

3. Функція оптимального значення, V^*(s), яка дає очікувану віддачу, якщо ви починаєте в стані s і завжди дієте відповідно до оптимальної політики в середовищі.

$$V^*(s) = \max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi} \left[R(\tau)\,|s_0 = s\right]$$

4) Оптимальна функція-значення дії, Q^*(s,a), яка дає очікувану віддачу, якщо ви починаєте в стані s, виконуєте довільну дію a, а потім постійно дієте відповідно до оптимальної політики в середовищі:

$$Q^*(s,a) = \max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi} \left[R(\tau)\,|s_0 = s, a_0 = a\right]$$

2.2.11 Оптимальна Q-функція та Оптимальна дія

Існує важливий зв’язок між оптимальною функцією дія-значення Q^*(s,a) і дією, обраною оптимальною політикою. За визначенням, Q^*(s,a) дає очікувану віддачу для початку в стані s, виконання (довільних) дій a, а потім безперервної дії відповідно до оптимальної політики. [9]

Оптимальна політика в s вибере ту дію, яка максимізує очікувану віддачу від початку в s. В результаті, якщо ми маємо Q^*, ми можемо безпосередньо отримати оптимальну дію, a^*(s), через:

$$a^*(s) = \arg\max_{a} Q^*(s,a).$$

Примітка: може бути кілька дій, які максимізують Q^*(s,a), і в цьому випадку всі вони є оптимальними, і оптимальна політика може випадково вибрати

будь-яку з них. Але завжди існує оптимальна політика, яка детерміновано вибирає дію.

2.2.12 Рівняння Беллмана

Усі чотири функції значення підкоряються спеціальним рівнянням самоузгодженості, які називаються рівняннями Беллмана. Основна ідея рівнянь Беллмана така:

Цінність вашої стартової точки — це винагорода, яку ви очікуєте отримати, перебуваючи там, плюс вартість того, де ви приземлиться наступним чином. [9]

Рівняння Беллмана для функцій цінності політики є:

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\substack{a\sim\pi \\ s'\sim P}} \left[r(s,a) + \gamma V^{\pi}(s') \right],$$
$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{s'\sim P} \left[r(s,a) + \gamma \mathop{\mathrm{E}}_{a'\sim\pi} \left[Q^{\pi}(s',a') \right] \right],$$

де s' ~ P є скороченням для s' ~ P(cdot |s,a), що вказує, що наступний стан s' відбирається з правил переходу середовища; a ~ pi є скороченням для ~ pi(cdot|s); і a' ~ pi є скороченням для a' ~ pi(cdot|s').

Рівняння Беллмана для функцій оптимального значення є:

$$V^*(s) = \max_a \mathop{\mathrm{E}}_{s' \sim P} \left[r(s,a) + \gamma V^*(s') \right],$$
$$Q^*(s,a) = \mathop{\mathrm{E}}_{s' \sim P} \left[r(s,a) + \gamma \max_{a'} Q^*(s',a') \right].$$

Вирішальна відмінність між рівняннями Беллмана для функцій цінності на політиці та функцій оптимального значення полягає у відсутності або наявності max over дій. Його включення відображає той факт, що щоразу, коли агент вибирає свою дію, щоб діяти оптимально, він повинен вибрати ту дію, яка веде до найвищого значення.

2.2.13 Функція Переваги

Іноді в RL нам не потрібно описувати, наскільки хороша дія в абсолютному сенсі, а лише наскільки вона в середньому краща за інші. Тобто ми хочемо знати відносну перевагу цієї дії. Ми уточнюємо цю концепцію за допомогою функції переваги. [8]

Функція переваги A^pi(s,a), що відповідає політиці \pi, описує, наскільки краще виконати конкретну дію a в стані s, ніж випадковий вибір дії відповідно до pi(cdot|s), припускаючи, що ви будете діяти відповідно до \pi назавжди. Математично функція переваги визначається як:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s).$$

2.3 Висновки за розділом

У рамках цього розділу ми розглянули поняття рекомендаційних систем, його основні методи. Як методи глибокого навчання використовуються в рекомендаційних системах. Було запропоновано критерії порівняння рекомендаційних систем. [8]

Також було розглянуто розглянули основні поняття глибокого навчання з підкріпленням.

РОЗДІЛ 3 РЕКОМЕНДАЦІЙНІ СИСТЕМИ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ З ПІДКРІПЛЕННЯМ

3.1 Вирішення проблеми нових об’єктів рекомендації

Для того щоб стало можливим відмовитися від повного перенавчання системи, потрібно вирішити проблему додавання нових об’єктів рекомендації з часом (наприклад нові фільми для онлайн кінотеатрів). Розглянемо на прикладі простого автокодувальника чим відрізняється повністю перенавчена модель від її попередньої версії при додаванні нових об’єктів рекомендації.

Очевидно, що з точки зору архітектури моделі був змінений лише вхідний і вихідний шари (див. рис 3.1). А саме з розмірність входу та виходу збільшилась на 10 (матриці вагів кожного шару збільшились на 10x256 та 256x10 відповідно)

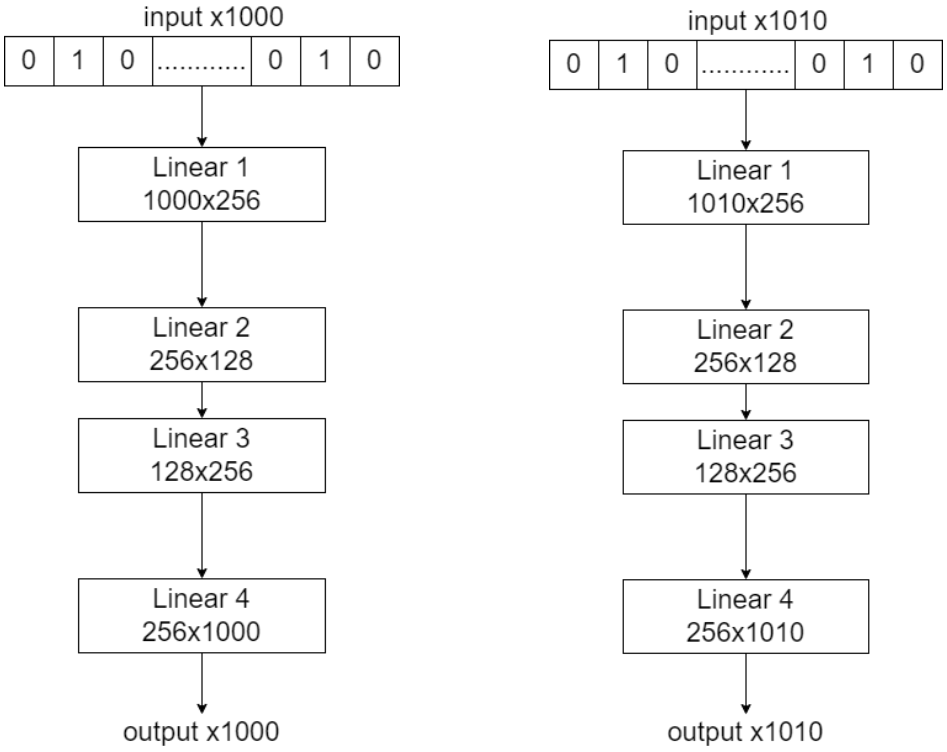

Рисунок 3.1 – Різниця архітектури автокодувальників

Тому було запропоновано створити вхідний та вихідний лінійниі шари які можуть динамічно збільшуватися, причому нові ваги повинні навчатись швидше за уже існуючі.

Де Linear 1, Linear 2, Linear 3, Linear 4 – це ваги взяті зі старої версії моделі, а Linear 5, Linear 6 – це нові ваги які відповідають за нові об’єкти рекомендації. Можна сказати що насправді архітектура запропонованої моделі (див. рис. 3.2) не відрізняється від архітектури при повному перенавчанні, а просто ініціалізує частину вагів старими значеннями, проте нова архітектура дозволяє більш зручно встановити різні параметри навчання для старих і нових вагів, або повністю заморозити старі (Linear 1, Linear 2, Linear 3, Linear 4) ваги для того щоб нові ваги “пристосувалися” до старих. Після навчання такої моделі можливо об’єднати паралельні лінійні шари в один.

Насправді таку прийом можливо застосовувати не тільки для лінійних слоїв, а також і до ембедінгів [16]. Що дає змогу використовувати цей прийом в більш складних моделях побудованих на ембедингах об’єктів рекомендації.

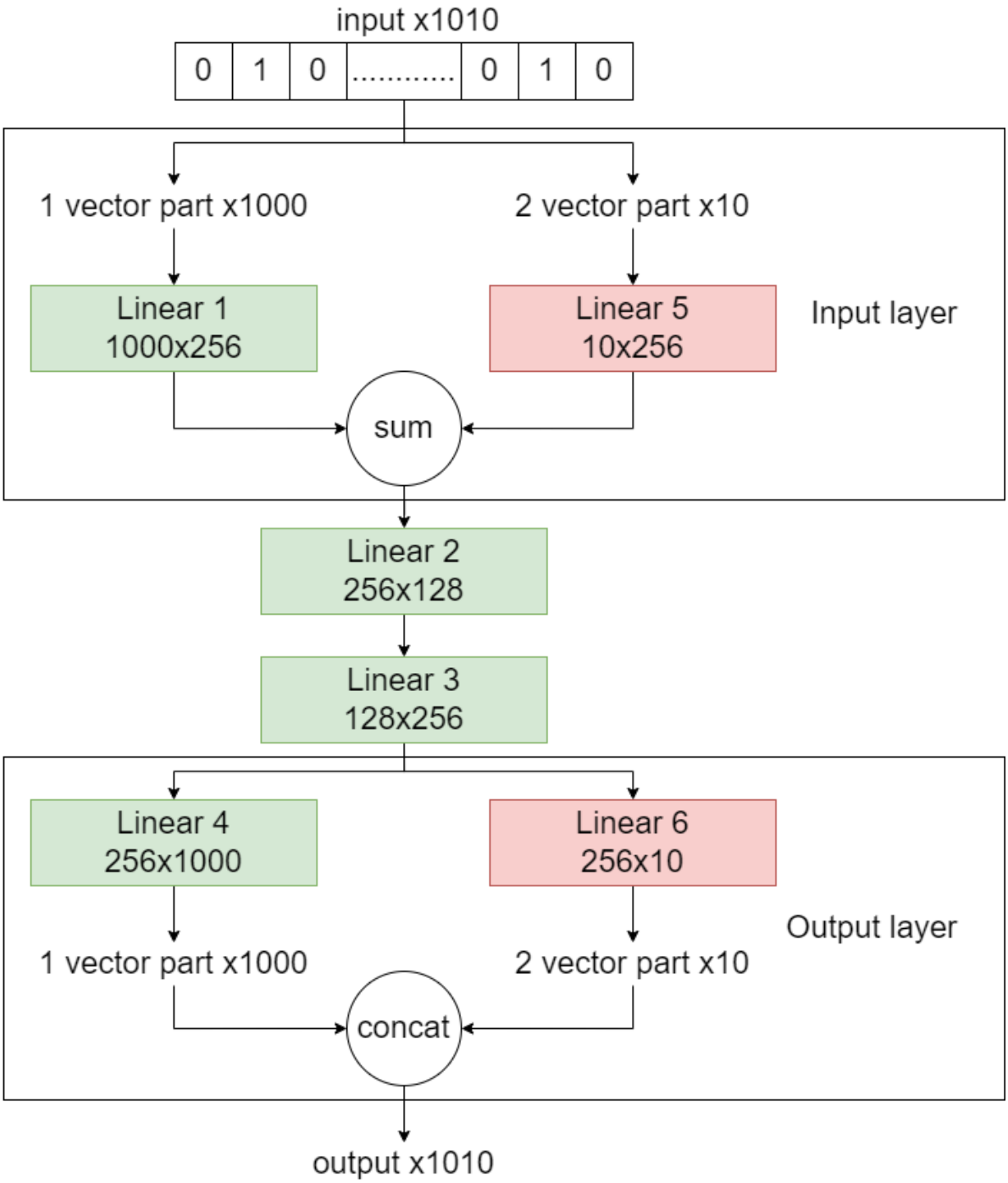

Рисунок 3.2 – Запропонована схема автокодувальника

3.2 Представлення навчання у вигляді агент – середовище

Представимо процес навчання у ітеративному вигляді. Де кожний крок це певний проміжок часу за який ми отримуємо новий фідбек від користувачів (перегляди, лайки …) та надаємо рекомендацію для користувачів на наступний проміжок часу.

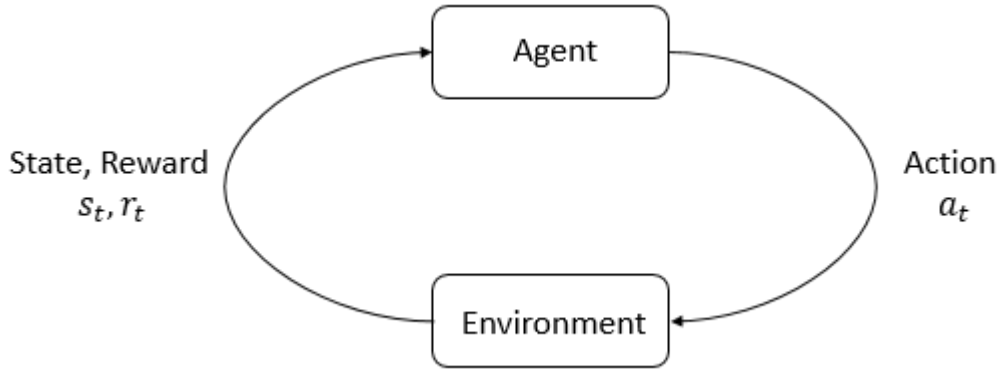

Рисунок 3.3 – Схема Агент-Середовище

Agent – Модель яка надає рекомендації

Environment – Платформа або сервіс який використовують користувачі

State (st) – Дії користувачів за проміжок часу t або дії за весь період часу до моменту часу t

Action (at) – Рекомендація (передбачення) на проміжок часу t+1 здійснене на основі st

Reward (rt) – якість рекомендації для проміжку часу t

При навчанні такої системи (див. рис 3.3) у реальному часі рекомендації (at) будуть впливати на наступний стан системи (st), а саме коричтувач побачить саме ті рекомендації які агент створив на попередньому кроці і відповідно на них відреагує (проігнорує, зробить клік, зробить покупку). Тобто агент отримує реакцію користувача не якісь випадкові об’єкти які опинилися в рекомендаціях користувача, а саме ті які він сам рекомендував на попередньому кроці, що дозволяє використовувати методи навчання з підкріпленням в цій задачі. Також ми отримуємо моніторинг якості рекомендацій агента в реальному часі базуючись на нагороді (rt).

Звичайно таку схему представлення навчання можна використовувати і для навчання на історичних даних, проте тоді ми втрачаємо переваги описані вище.

3.3 Зменшення обчислювальної потужності

При поєднані ітеративного тренування у вигляді агент-середовище та моделі яку не потрібно перенавчати у випадку появи нових контент об’єктів ми отримуємо значне зменшення обчислювальні потужності необхідної на надання таких рекомендацій. Наприклад якщо ми хочемо оновлювати нашу рекомендаційну модель кожного дня, і ми знаємо, що новий об’єкт рекомендації може з’явитися в будь-який момент, то нам буде необхідно повністю перенавчати нашу модель. Проте в випадку ітеративного навчання агент-середовище нам достатньо лише “донавчити” модель на нових даних які з’явилися за день.

3.4 Предметна область

Для перевірки якості обраного підходу будемо використовувати відкриті дані медіа платформи MEGOGO з Kaggle хакатону “Megogo Challenge” [12]

Які будуть представленні у вигляді таблиці з наступними стовпцями

1. Код користувача.
2. Код відео.
3. Дата перегляду.
4. Відсоток перегляду.

3.5 Аналіз якості рекомендацій.

Для розробки та проведення аналізу було реалізовано програмний продукт на мові Python [13] з використанням бібліотеки Pytorch [14] та використано програмне середовище JupyterLab [15].

Приведемо порівняльну таблицю якості роботи розробленого алгоритму по відношенню до вже існуючих методів за критеріями визначеними в параграфі 2. На даних описаних в пункті 3.4. Проміжок передбачення для метрики MAP@K становить 1 день. Переглядами користувачів за тестовий період вважалися лише ті відео об’єкти які користувач до цього не дивився. Результати оцінювання рекомендацій відображені у таблиці 3.1.

Таблиця 3.1 – Порівняльна таблиця

Назва методу	Метрика MAP@10	Метрика MAP@5	Необхідність повного перенавчання при виникненні нового об’єкту рекомендації
Простий Автокодувальник	0.02981	0.02842	Так
Автокодувальник з динамічним вхідним шаром. Навчений в системі середовище-агент	0.03437	0.03151	Ні

3.6 Висновки за розділом

Було розроблено і створено програмний продукт який дозволяє тренувати та використовувати рекомендаційні системи в режимі реального часу.

Програмний продукт був використаний для навчання алгоритму на даних з онлайн медіа платформи MEGOGO.

Проведений порівняльний аналіз запропонованої системи і з іншими методами. За результатами порівняльного аналізу можна зробити висновки що запропонована система демонструє кращі результати ніж інші методи.

РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

4.1 План розробки стартапу та масштабування його на ринок

Наведемо план розробки стартапу та виведення його на ринок.

Спочатку треба провести маркетинговий аналіз, який включає в себе:

- конкурентний аналіз, щоб зрозуміти, якими методами вирішення проблем вже користуються люди;
- формування ідеї самого проекту та виділення цільової аудиторії;
- розробити стратегію виведення товару на ринок, базуючись на аналізі ринкового середовища.

Наступним кроком являється організація самого стартапу. На цьому етапі мають бути:

- складений весь план та побудований таймлайн розробки та запуску продукту;
- запланований обсяг виробництва та оцінений потенційний обсяг ресурсу, який буде потрібен для виконання плану;
- розраховані витрати, необхідні для реалізації проекту, та витрати на запуск проекту.

Далі необхідно виконати фінансово-економічний аналіз та оцінити ризики стартап-проекту, в межах якого:

- визначити обсяг інвестиційних втрат;
- розрахувати основні фінансово-економічні показники проекту (собівартість, ціну продукту/послуги, податковий збір та чистий прибуток) та визначити показники інвестиційної привабливості проекту (рентабельність продажів, період окупності проекту);
- визначити основні ризики проекту та способи для їх запобігання.

Фінальним кроком являється розробка заходів з комерціалізації продукту.

Цей крок являється важливим для масштабування та збільшення розмірів продукту.

Для того, щоб залучити інвесторів та знайти різні способи фінансування проекту, необхідно:

- провести дослідження на предмет інтересів потенційних інвесторів та бізнесів;
- скласти інвестиційну пропозицію, яка включає в себе як опис самого продукту та його теперішні розміри, так і можливі шляхи розширення та розвитку;
- обрати канали комунікації із потенційно зацікавленими персонами.

Далі наведемо результати виконання кожного з описаних кроків.

4.2 Опис ідеї стартап-проекту

Стартап-проект полягає у вирішенні проблеми рекомендаційних систем. Це завдання, яке цікавить як комерційні організації, так і державні установи. Суть продукту полягає у тому, що система опрацьовує інормацію про користувача та надає рекомендацію для нього.

У таблиці 4.1 наведені основні характеристики продукту, а також описані ідея та ціль проекту.

Таблиця 4.1 – Інформаційна карта стартап-проекту

Назва проекту	Рекомендаційний алгоритм
Автори проекту	Якубенко Олексій Петрович
Коротка анотація	Проект спрямований рекомендаційні системи. А саме навчання і надання рекомендацій у реальному часу.
Термін реалізації проекту	6 місяців
Необхідні ресурси	Приміщення з комп'ютерами, доступом до Інтернету, електрики Програмне забезпечення, хмарне сховище, антивірусні програми Фінансові кошти на оплату заробітної плати виконавцям на термін 6 місяців, а також на операційні витрати, як оренда приміщення, комунальні послуги тощо
Опис проблеми, яку вирішує проект	Система дає змогу запровадити рекомендації для користувачів та зменшити обчислювальні ресурси для їх підтримки
Головні цілі та завдання проекту	Метою проекту є створення системи, яка буде ефективно та автоматично обробляти надавати рекомендації
Очікувані результати	Автономна система надання рекомендацій

4.3 Технологічний аудит ідеї проекту

Тепер можна розібрати ідею стартапу та провести конкурентний аналіз. У таблиці 4.2 наведений опис ідеї стартапу.

Таблиця 4.2 – Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основна ідея являється створення комплексної системи, яка буде приймати на вхід інформацію про дії користувача та надавати рекомендацію для нього	Опрацювання інформації про користувача	Користувач зможе отримувати автоматичні рекомендації
	Опрацювання інформації про користувача	Користувач зможе отримувати автоматичні рекомендації
	Опрацювання інформації про користувача	Користувач зможе отримувати автоматичні рекомендації

Далі проведемо порівняльний аналіз конкурентів проекту та наведемо результати у таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз конкурентів проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Власний проект	Respeecher	Transkriptor	Omilia			
1	Точність рекомендацій	Надає доволі хороший результат	Свій власний алгоритм	Свій власний алгоритм	Свій власний алгоритм			+
2	Доступність по ціні	Безкоштовний при запуску MVP	Більш доступний для комерції	Безкоштовний тріал	Більш доступний для комерції		+	
3	Сентиментальний аналіз	Присутній	Відсутній	Відсутній	Присутній, але за додаткову плату		+	
4	Точність рекомендацій	Достатньо хороша, конкурує із наведеним списком	Свій власний алгоритм	Свій власний алгоритм	Свій власний алгоритм		+	

Далі аналізуємо реальність технічно здійснити ідею проекту (таблиця 4.4).

Таблиця 4.4 – Технологічна здійсненність продукту

№ п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
1	Основна ідея являється створення комплексної системи, яка буде приймати на вхід інформацію про дії користувача та надавати рекомендацію для нього	Використання мови програмування Python	Наявні	Доступні
2		Використання мови програмування JavaScript	Не наявні, необхідні допрацювання	Доступні
3		Використання мови програмування C++	Наявні, необхідні допрацювання	Доступні
Обрана технологія реалізації ідеї проекту: Python				

4.4 Аналіз ринкових можливостей запуску стартап-проекту

Далі проведемо попередній аналіз ринку для запуску стартап-проекту (таблиця 4.5).

Таблиця 4.5 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	6000
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	10%

Тепер проведемо характеристику потенційних клієнтів, які можуть бути зацікавлені в проекті (таблиця 4.6).

Таблиця 4.6 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Рекомендаційна система	Малі та великі підприємства	Цікавлять окремі рекомендації для користувачів	Детальне звітування по кожному пункту
2	Аналіз дій користувача	Держава	Цікавить комплексний аналіз аудиторії	Простота у використанні
3	Рекомендаційна система	Банківська сфера та колектори	Цікавлять окремі рекомендації для користувачів	Можливість працювати з великою кількістю даних (своєї та чужих компаній)

Тепер для вибудови стратегії виходу на ринок, необхідно обрахувати важливі фактори загроз (таблиця 4.7) та можливостей (таблиця 4.8). Необхідно проаналізувати загрози, щоб зрозуміти можливі перешкоди при запуску продукт на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 4.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Ринок хоч ще рожевий океан, але вже є кілька великих гравців, які відомі на ринку	Знайти точки додаткової цінності для користувача
2	Ціна збуту	Конкуренти можуть коштувати менше через нижчу якість	Акцентувати увагу на якості та продумати комунікаційну стратегію
3	Якість аналізу	Через комплексність задачі, кожна зі складових може мати гіршу якість	Мати достатній штаб, щоб кожна команда займалася своїм напрямом

Таблиця 4.8 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Комплексний аналіз	Продукт пропонує повний пакет аналізу аудіо, а не окремі фічі	Максимально сильно промити та поширювати дану функцію для користувачів
2	Простота у використанні	Від користувача треба лише завантажити аудіо файл	Реалізувати зручний інтерфейс для завантаження аудіо файлу
3	Якість та гарантії	Надавати найбільш якісні послуги та сервіси	Пропонувати моделі з найкращими результатами, а також надавати усю необхідну технічну підтримку
4	Безкоштовне використання при MVP1	Максимально швидко набрати базу своїх клієнтів та заявити про себе на ринку	Розгорнути широкий маркетинг кампейн, а також стукатися до клієнтів конкурентів

Далі розглянемо питання конкуренції, а саме визначимо її тип та рівень (таблиця 4.9).

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: недосконала конкуренція	Представлено мало продуктів та експертів	Зробити максимально агресивний маркетинг, щоб швидко стати монополістами
2. За рівнем конкурентної боротьби: міжнародний	Представлені проекти, розроблені та доступні у різних країнах	Розширити цільову аудиторію, зробити мультимовність, додати компанії різних країн світу
3. За галузевою ознакою: внутрішньогалузева	Можуть працювати з різними галузями	Покращити масштабованість та персоналізації
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з аналізами інших систем та експертів	Підтримувати та покращувати якість існуючих функцій
5. За характером конкурентних переваг: нецінова	Різні компанії пропонують різну точність	Розробляти точніші та покращувати існуючі алгоритми
6. За інтенсивністю: марочна	Вже представлені компанії із сильним брендом	Предметно створити комунікаційну стратегію для вибудови свого бренду

Далі необхідно виконаємо аналіз конкуренції за моделлю 5 сил конкуренції Майкла Портера (таблиця 4.10).

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
	Інші існуючі системи та продукти	Якість, ціни, кількість користувачів, капіталовкладення	Фактори сили постачальників	Контроль якості, порівняння цін	Сила бренду, якість, ціна, масштаб
Висновки	Конкуренція з невеликою інтенсивністю, а також підігрітий ринок	Можливості входження на ринок, нові потенційні конкуренти	Постачальники відсутні	Клієнти не диктують умови роботи на ринку	Товарозамінники відсутні

Маючи результати аналізу конкуренції (таблиця 4.10), характеристики ідеї стартап-проекту (таблиця 4.5), характеристики потенційних клієнтів і їх вимоги до продукту (таблиця 4.6) та фактори ринкового середовища (таблиці 4.7 і 4.8) було сформульовано та обґрунтовано перелік факторів конкурентоспроможності (таблиця 4.11).

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Комплексний аналіз	Система комплексно аналізує аудіо файл, що об’єднує в собі визначення мови та репліки співрозмовника, сентиментальний аналіз
2	Простота у використанні	Для використання системою, клієнту необхідно лише завантажити аудіо файл
3	Якість та гарантії	Система має високу точність, яка з кожною ітерацією покращується, а технічна підтримка на зв’язку цілодобово
4	Безкоштовне використання при MVP1	Дає можливість спробувати користуватися системою як великим, так і середнім та маленьким бізнесам

Тепер можна провести аналіз сильних та слабких сторін продукту (таблиця 4.12).

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Комплексний аналіз	20	+						
2	Простота у використанні	16		+					
3	Якість та гарантії	10		+					
4	Безкоштовне використання при MVP1	17			+				

Далі проведемо SWOT-аналіз продукту (таблиця 4.13).

Таблиця 4.13 – SWOT-аналіз стартап-проекту

Сильні сторони Комплексний аналіз Простота у використанні Якість та гарантії Безкоштовне використання при MVP1	Слабкі сторони Відсутність сильного бренду Не сформована база клієнтів Не підключені альтернативні канали маркетингу
Можливості Покращення системи Збільшення фічей Інтеграція з існуючими CRM	Загрози Нові системи та експерти Збут

Дякуючи проведенню SWOT-аналізу, ми змогли визначити сильні та слабкі сторони, можливості та загрози, пов'язані з конкуренцією та плануванням стартап-проекту. Далі спроектуємо альтернативну ринкову поведінку для інтеграції стартап-проекту на ринок та приблизний час реалізації системного комплексу, з урахуванням потенційних проектів, що можуть бути виведені на ринок та наведемо результати у таблиці 4.14.

Таблиця 4.14 – Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок без якоїсь із фіч	70%	4 місяці
2	Пропонувати одразу платне використання	50%	6 місяців
3	Представлення користувачам системи без інтерфейсу	60%	4,5 місяці

У даному пункті був проведений детальний аналіз ринку та продукту. Також відповідно до результатів проведеного конкурентного аналізу, визначених факторів ринку та його сприятливість, описання ідеї та характеристик стартап-проекту, робимо висновок висновок, що існують дуже сприятливі умови для виходу продукту на ринок.

4.5 Розроблення ринкової стратегії стартап-проекту

Для розробки ринкової стратегії продукту, у першу чергу, необхідно проаналізувати цільову аудиторію проекту (таблиця 4.15).

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Банківська сфера	Висока	25%	Висока	Середня
2	Великі бізнеси	Середня	20%	Середня	Середня
3	Малі та середні бізнеси	Середня	20%	Середня	Середня
4	Держава	Низька	10%	Низька	Висока
Які цільові групи обрано: 1, 2					

Маючи аналіз цільових груп, далі визначимо базову стратегію розвитку продукту (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	1 та 2	Диференційованого маркетингу	Масштабування та максимізація	Оптимальних витрат

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 4.17, 4.18).

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Так	Так	Ні	Виклику лідера

Таблиця 4.18 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Комплексний аналіз Простота у використанні Якість результатів	Оптимальних витрат	Комплексний аналіз Простота у використанні Якість та гарантії Безкоштовне використання при MVP1	Система, яка краще всіх проводить аудіо аналіз Система з простим інтерфейсом та доступністю

4.6 Розроблення маркетингової програми стартап-проекту

Після проведеного комплексного аналізу, можемо повноцінно описати ключові переваги концепції потенційного товару (таблиця 4.19) та побудувати концепцію маркетингових комунікацій (таблиця 4.20).

Таблиця 4.19 – Ключові переваги концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Якісні рекомендації	Якісний аналіз системою із високою точністю	Постійне покращення та навчання моделей, які зможуть краще прибирати шум
2	Комплексний аналіз	Система комплексної рекомендації	Єдина система, яка пропонує всі фічі в одному продукті. Кількість фічей буде збільшуватися
3	Простий інтерфейс	Система дуже проста у використанні	Система із інтуїтивно зрозумілим інтерфейсом, який вимагає всього лише доступу до даних компанії

Таблиця 4.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пошук спеціалізованих систем	b2b продажі Зв'язок через теплі контакти Таргетована реклама у соціальних мережах Реферальна система	Точність Якість Комплексність	Поєднати повідомлення про те, що це якісна система, яка дає комплексний аналіз	Таргетована реклама на цільову аудиторію Реферальна система
2	Пошук доступного та дешевого продукту	Інтернет, форуми, реклами від інфлюенсерів	Простота Безкоштовне використання MVP	Вселити довіру у бренд та продукт	Реклама у лідерів думок Вивіски в публічних місцях Таргетована реклама на цільову аудиторію

4.7 Висновки до розділу 4

Даний розділ був присвячений дослідженню стартап-проекту. В якості такого була представлена система для рекомендації контент об’єктів.

У рамках розділу було досліджено розробку стратегій виходу на ринок та маркетинг-стратегії для цього. Зокрема, даний ринок являється блакитним океаном з невеликою кількістю представлених компаній. Оскільки вони дають лише частину функцій, а запропонована система є комплексною та доступною, то у стартап-проекту є всі шанси стати монополістами на ринку.

Також були опрацьовані сильні та слабкі сторони проекту, аналіз конкурентів та цільової аудиторії. На основі всіх досліджень був сформований концепт маркетингової стратегії для обраних цільових аудиторій.

Проведений порівняльний аналіз запропонованої системи і з іншими методами. За результатами порівняльного аналізу можна зробити висновки що запропонована система демонструє кращі результати ніж інші методи

ВИСНОВОК

Дана робота присвячена аналізу, побудові та використанню рекомендаційної системи навчену за допомогою методів навчання з підкріпленням в контексті підтримки актуальності рекомендацій. Після ознайомлення з теоретичним матеріалом щодо суті проблематики та принципів роботи основних методів рекомендаційних систем, було запропоновано архітектуру рекомендаційної системи, що потенційно дає змогу необмежено нарощувати варіанти рекомендації, що в свою чергу надає можливість не перенавчати рекомендаційну систему з плином часу та додаванням нових об’єктів.

Також після аналізу основних методів навчання з підкріпленням було запропоновано алгоритм навчання нейронної мережі описаної вище, який дає змогу навчати нейронну мережу, слідкувати за якістю рекомендацій та надавати рекомендації для користувачів в режимі реального часу, використовуючи при цьому менше обчислювальних ресурсів ніж аналогічні системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Mean Average Precision for Recommender Systems. 2016. URL: http://sdsawtelle.github.io/blog/output/mean-average-precision-MAP-for-recommender-systems.html (дата звернення 14.10.2021)
2. Greg Linden, Brent Smith, Jeremy York. Amazon Recommendations Item-to-Item Collaborative Filtering // university of Maryland. 2003. URL: http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf (дата звернення: 14.10.2021)
3. Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms // ra.ethz.ch. 2001. URL: https://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf (дата звернення: 14.10.2021)
4. Elyor Kodirov, Tao Xiang, Shaogang Gong. Semantic Autoencoder for Zero-Shot Learning // thecvf. 2017. URL: https://openaccess.thecvf.com/content_cvpr_2017/papers/Kodirov_Semantic_Autoencoder_for_CVPR_2017_paper.pdf (дата звернення: 14.10.2021)
5. Diana Ferreira, Sofia Silva, António Abelha, José Machado. Recommendation System Using Autoencoders // mdpi. 2020. URL: https://www.mdpi.com/2076-3417/10/16/5510/htm (дата звернення 14.10.2021)
6. Bahare Askari, Jaroslaw Szlichta, Amirali Salehi-Abari.Joint Variational Autoencoders for Recommendation with Implicit Feedback // arXiv. 2020. URL: https://arxiv.org/pdf/2008.07577.pdf (дата звернення: 14.10.2021)
7. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua. Neural Collaborative Filtering // arXiv. 2017. URL: https://arxiv.org/abs/1708.05031 (дата звернення: 14.10.2021)
8. OpenAI Key Concepts and Terminology in RL. 2018. URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html (дата звернення 14.10.20211)

9. Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, Joelle Pineau. An Introduction to Deep Reinforcement Learning // arXiv. 2018. URL: https://arxiv.org/pdf/1811.12560.pdf (дата звернення: 14.10.2021)
10. Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides. Behaviour Suite for Reinforcement Learning // arXiv. 2020. URL: https://arxiv.org/pdf/1908.03568.pdf (дата звернення: 14.10.2021)
11. Alexandre Heuilleta, Fabien Couthouisb, Natalia Díaz-Rodríguezc. Explainability in Deep Reinforcement Learning // arXiv. 2021. URL: https://arxiv.org/pdf/2008.06693.pdf (дата звернення: 14.10.2021)
12. Megogo Challenge Data. 2019. URL: https://www.kaggle.com/c/megogochallenge/data (дата звернення 14.09.2021)
13. Python programming language. URL: https://www.python.org/ (дата звернення 14.10.2021)
14. Pytorch. Open-source machine learning framework. URL: https://pytorch.org/ (дата звернення 14.10.2021)
15. JupyterLab. Open-source software. URL: https://jupyter.org/ (дата звернення 14.10.2021)
16. Omer Levy, Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization // source. 2014. URL: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf (дата звернення: 14.10.2021)

ДОДАТОК А КОД ПРОГРАМНОГО ПРОДУКТУ

```
import pandas as pd

import numpy as np

from datetime import date, datetime, timedelta

from tqdm.notebook import tqdm, trange

import gym

from ml_metrics import mapk

from scipy.sparse import coo_matrix, csr_matrix

import ml_metrics

from torch import nn

from torch.utils.data import Dataset, DataLoader

from catalyst import dl

import torch

from torch import optim, nn, FloatTensor, LongTensor



def load_data():

    train_data_full = pd.read_csv('data/train_data_full.csv.zip')

    train_data_full['session_start_datetime'] = pd.to_datetime(train_data_full['session_start_datetime'])

    train_data_full['video_id'] = pd.Categorical(train_data_full['primary_video_id']).codes

    train_data_full['user_id'] = pd.Categorical(train_data_full['user_id']).codes

    return train_data_full[['user_id', 'video_id', 'watching_percentage', 'session_start_datetime']].reset_index(drop=True)



class BasicEnv(gym.Env):
```

```
def __init__(self, data, train_size = 14, target_size = 1):
    min_date = data['session_start_datetime'].min().date()
    max_date = data['session_start_datetime'].max().date()

    self.data = data

    self.train_mask = []
    self.target_mask = []



    time = self.data['session_start_datetime'].dt.date
    for i in trange((max_date - min_date).days-train_size-target_size+2):
        train_left = min_date + timedelta(days=i)
        train_right = train_left + timedelta(days=train_size-1)

        target_left = train_right + timedelta(days=1)
        target_right = target_left + timedelta(days=target_size-1)

        train_mask = (train_left <= time) & (time <= train_right)
        target_mask = (target_left <= time) & (time <= target_right)

        self.train_mask.append(train_mask)
        self.target_mask.append(target_mask)

    self.i = 0
    self.max_i = len(self.train_mask) - 2
```

```
def get_state(self):
    train_X = self.data[self.train_mask[self.i]]
    train_y = self.data[self.target_mask[self.i]]
    valid_X = self.data[self.train_mask[self.i+1]]
    return train_X, train_y, valid_X

def get_reward(self, action):
    valid_y = self.data[self.target_mask[self.i+1]]
    valid_y = valid_y.groupby('user_id')['video_id'].apply(set).apply(list)
    idx = valid_y.index[valid_y.index.isin(action.index)]
    y = valid_y[idx]
    x = action[idx]
    return mapk(y, x, 10)

def step(self, action):
    reward = self.get_reward(action)

    self.i +=1
    if self.i >= self.max_i:
        done = True
    else:
        done = False

    state, reward, done, info = self.get_state(), reward, done, {}

    return state, reward, done, info
```

```
    def reset(self):
        self.i = 0
        return self.get_state()


def merge_layers(layers, dim):
    with torch.no_grad():
        weight = torch.cat([l.weight for l in layers], dim=dim)
        bias = torch.stack([l.bias for l in layers]).sum(dim=0)
        l = nn.Linear(weight.shape[1], weight.shape[0])
        l.weight = nn.Parameter(weight)
        l.bias = nn.Parameter(bias)
    return nn.ModuleList([l])


class ParallelInputLinear(nn.Module):
    def __init__(self, input_sizes, out_size):
        super().__init__()
        self.out_size = out_size
        self.layers = nn.ModuleList([nn.Linear(inp_size, out_size) for inp_size in input_sizes])

    def merge_layers(self):
        with torch.no_grad():
            weight = torch.cat([l.weight for l in self.layers], dim=1)
```

```
            bias = torch.stack([l.bias for l in self.layers]).sum(dim=0)
            l = nn.Linear(weight.shape[1], weight.shape[0])
            l.weight = nn.Parameter(weight)
            l.bias = nn.Parameter(bias)
        self.layers = nn.ModuleList([l])

    def add_layer(self, layer_input_size):
        self.layers.append(nn.Linear(layer_input_size, self.out_size ))



    def forward(self, inp):
        splits = np.cumsum([0]+[l.in_features for l in self.layers])
        splits = zip(splits[:-1], splits[1:])
        out = [l(inp[:, split[0]: split[1]]) for l, split in zip(self.layers, splits)]
        return torch.stack(out).sum(dim=0)



class ParallelOutputLinear(nn.Module):
    def __init__(self, inp_size, out_sizes):
        super().__init__()
        self.inp_size = inp_size
        self.layers = nn.ModuleList([nn.Linear(inp_size, out_size) for out_size in out_sizes])

    def add_layer(self, layer_output_size):
        self.layers.append(nn.Linear(self.inp_size, layer_output_size))

    def merge_layers(self):
```

```
        with torch.no_grad():
            weight = torch.cat([l.weight for l in self.layers], dim=0)
            bias = torch.cat([l.bias for l in self.layers])
            l = nn.Linear(weight.shape[1], weight.shape[0])
            l.weight = nn.Parameter(weight)
            l.bias = nn.Parameter(bias)
        self.layers = nn.ModuleList([l])

    def forward(self, inp):
        return torch.cat([l(inp) for l in self.layers], dim=1)




def create_mlp(layers_shape, activation=nn.ReLU):
    layers = [nn.Linear(layers_shape[0], layers_shape[1])]
    for i in range(1, len(layers_shape)-1):
        layers.append(activation())
        layers.append(nn.Linear(layers_shape[i], layers_shape[i+1]))
    return nn.Sequential(*layers)



class AutoEncoderModel(nn.Module):
    def __init__(self, input_size, layers):
        super().__init__()
        self.input_layer = ParallelInputLinear([input_size], layers[0])
        self.encoder = nn.Sequential(create_mlp(layers), nn.ReLU())
        self.decoder = create_mlp(list(reversed(layers)))
```

```
        self.output_layer = ParallelOutputLinear(layers[0], [input_size])



    def forward(self, features):
        x = self.input_layer(features)
        x = self.encoder(x)
        x = self.decoder(x)
        x = self.output_layer(x)

        x_intersection = x.clone()
        x[features>0] = 0
        return x_intersection, x




class AutoEncoderTrainDataset(Dataset):
    def __init__(self, views, split_dropout=0.5):
        self.views = views
        self.split_dropout = split_dropout

    def __len__(self):
        return self.views.shape[0]

    def __getitem__(self, idx):
        features = self.views[idx].toarray()[0]
        targets = features.copy()
```

```
        mask = torch.rand(features.shape[0]) < self.split_dropout
        features[mask] = 0
        targets[~mask] = 0

        return {
            'features': features,
            'targets': targets,
        }



class ModelTrainer:
    def __init__(self, config):
        self.optimizer_new_params = config['optimizer_new_params']
        self.optimizer_old_params = config['optimizer_old_params']
        self.train_dataloader_params = config['train_dataloader_params']
        self.test_dataloader_params = config['test_dataloader_params']
        self.num_epochs = config['num_epochs']
        self.logdir = config['logdir']

        self.video_id2code = {}
        self.model = AutoEncoderModel(0, **config['model_params'])

        self.criterion = {"mse": nn.MSELoss()}
        self.callbacks = [
            dl.CriterionCallback(input_key="logits_features",
                                 target_key="features",
                                 criterion_key="mse",
```

```
                    metric_key="recovery_loss"),
        dl.CriterionCallback(input_key="logits_targets",
                    target_key="targets",
                    criterion_key="mse",
                    metric_key="predict_loss"),
        dl.MetricAggregationCallback(metric_key="loss",
                        metrics={"recovery_loss": 1, "predict_loss": 2},
                        mode="weighted_sum")
    ]




def add_new_video_id(self, views_list):
    input_video_id = set(views_list['video_id'])
    new_video_id = input_video_id - self.video_id2code.keys()
    if not len(new_video_id):
        return

    video_id2code = {vid: len(self.video_id2code)+i for i, vid in enumerate(new_video_id)}
    self.video_id2code.update(video_id2code)

    self.model.input_layer.add_layer(len(new_video_id))
    self.model.output_layer.add_layer(len(new_video_id))
    if self.model.input_layer.layers[0].in_features == 0:
        self.model.input_layer.merge_layers()
        self.model.output_layer.merge_layers()
```

```
def pre_proc_data(self, views_list):
    user_id_code = pd.Series(views_list['user_id'].unique()).reset_index().rename(columns={'index':'code', 0:'id'})
    user_id2code = user_id_code.set_index('id')['code']
    views_list['user_id'] = views_list['user_id'].map(user_id2code)
    views_list['video_id'] = views_list['video_id'].map(self.video_id2code)

    views_matrix = csr_matrix(
        (
            views_list['watching_percentage'],
            (views_list['user_id'].to_numpy(), views_list['video_id'].to_numpy())
        ),
        dtype=np.float32,
        shape=(len(user_id2code), len(self.video_id2code))
    )
    return views_matrix

def get_optim(self):
    if len(self.model.input_layer.layers) == 1:
        self.model.input_layer.merge_layers()
        self.model.output_layer.merge_layers()
        return optim.Adam(self.model.parameters(), **self.optimizer_old_params)

    new_inp_params = list(self.model.input_layer.layers[-1].parameters())
    new_out_params = list(self.model.output_layer.layers[-1].parameters())
    new_params = new_inp_params + new_out_params

    old_inp_params = list(self.model.input_layer.layers[:-1].parameters())
```

```
    old_out_params = list(self.model.output_layer.layers[:-1].parameters())
    encoder_params = list(self.model.encoder.parameters())
    decoder_params = list(self.model.decoder.parameters())
    old_params = old_inp_params + old_out_params + encoder_params + decoder_params

    optimizer = optim.Adam(
        [
            {'params': new_params, **self.optimizer_new_params},
            {'params': old_params, **self.optimizer_old_params}
        ]
    )
    return optimizer


def train(self, views_list):
    self.add_new_video_id(views_list.copy())
    views_matrix = self.pre_proc_data(views_list)
    dataloader = DataLoader(AutoEncoderTrainDataset(views_matrix), **self.train_dataloader_params)
    optimizer = self.get_optim()

    runner = dl.SupervisedRunner(
        input_key=["features"],
        output_key=["logits_features", 'logits_targets'],
        target_key=["features", "targets"],
        loss_key="loss"
    )
    runner.train(
```

```
        model=self.model,
        criterion=self.criterion,
        optimizer=optimizer,
        loaders={'train': dataloader},
        num_epochs=self.num_epochs,
        callbacks=self.callbacks,
        logdir=self.logdir,
        verbose=True,
    )
    del runner
    del dataloader
    del optimizer

    self.model.input_layer.merge_layers()
    self.model.output_layer.merge_layers()

def predict(self, views_list, k=10):
    views_matrix = self.pre_proc_data(views_list.copy())
    dataloader = DataLoader(AutoEncoderTrainDataset(views_matrix), **self.test_dataloader_params)
    prediction = []
    with torch.no_grad():
        self.model.eval()
        for batch in tqdm(dataloader):
            logits = self.model(batch['features'].cuda())[1]
            batch_pred = torch.topk(logits, k, dim=1, sorted=True).indices.cpu().numpy().astype(np.int32)
            prediction.append(batch_pred)
        self.model.train()
```

```
        prediction = np.concatenate(prediction)


        video_code2id = {v: k for k, v in self.video_id2code.items()}
        prediction = np.vectorize(video_code2id.__getitem__)(prediction)


        user_id_code = pd.Series(views_list['user_id'].unique()).reset_index().rename(columns={'index':'code', 0:'id'})
        user_id2code = user_id_code.set_index('id')['code']
        prediction = pd.DataFrame(prediction, index=user_id2code.index).apply(list,axis=1)


        return prediction



    def step(self, observation):
        self.train(observation[-1].copy())
        prediction = self.predict(observation[-1].copy() , 10)
        return prediction



CONFIG = {
    'model_params': {
        'layers': [2512, 1520, 1024, 720]
    },
    'optimizer_new_params': {
      'lr': 0.001,
      'betas': [0.95, 0.999],
      'eps': 0.000000001,
      'amsgrad': True,
```

```
    },
    'optimizer_old_params':{
      'lr': 0.0001,
      'betas': [0.95, 0.999],
      'eps': 0.000000001,
      'amsgrad': True,
    },
    'train_dataloader_params':{
      'batch_size': 2048,
      'num_workers': 8,
      'pin_memory': True,
      'shuffle': True,
      'drop_last': False,
    },
    'test_dataloader_params':{
      'batch_size': 2048,
      'num_workers': 8,
      'pin_memory': True,
      'shuffle': False,
      'drop_last': False,
    },
    'num_epochs': 1,
    'logdir': 'logs'
}

data = load_data()
mt = ModelTrainer(CONFIG)
```

```
env = BasicEnv(data)



reward_history = []


observation = env.reset()
done = False
while not done:
    action = mt.step(observation)
    observation, reward, done, info = env.step(action)
    reward_history.append(reward)
    print(reward)
print(reward_history[-3:])
```