Дослідження масштабування та розробка рекомендаційної системи методами глибинного навчання та технологій великих даних

Владислав Гой

Кафедра інформаційних технологій
Прикарпатський національний університет імені Василя Стефаника
Івано-Франківськ, Україна

***Анотація*—У даній статті представлено розробку механізму рекомендацій із використанням технологій великих даних і методів оптимізації, створених для прискорення обробки даних.**

Ключові слова—python, apache spark, sql, pandas, великі дані, обробка даних, рекомендаційна система, оптимізаційні методи.

I. Вступ

Механізми або системи рекомендацій здебільшого використовуються для рекомендування відповідного та корисного вмісту потрібному користувачеві, щоб певним чином покращити загальний досвід користувача та надати йому нові функції. Такі механізми є потужними та широко використовуваними з точки зору використання величезної кількості даних і навчання розуміти переваги конкретних користувачів. Рекомендації допомагають користувачам легко переміщатися між мільйонами продуктів або тонами вмісту (статті/відео/фільми) і показують їм потрібну інформацію, яка може бути корисною для певної служби, якою вони користуються. Така функція допомагає знаходити інформацію від імені користувачів.

У більшості випадків користувачі вирішують, чи ефективний механізм щодо рекомендацій, і вони можуть вибрати вміст або ігнорувати його. Кожне рішення користувачів допомагає перенавчити модель машинного навчання за лаштунками на останніх даних, щоб мати можливість давати кращі рекомендації.

II. Аналіз останніх досліджень і публікацій

Багато дослідницьких робіт було проведено з використанням методів великих даних, включаючи інтелектуальний аналіз даних і методи машинного навчання. Застосувань цих технік дуже багато. Зокрема, механізм рекомендацій, який використовується в кількох сферах, таких як аналіз кошика (Amazon), соціальні мережі (LinkedIn, Twitter), уряд, освіта тощо.

Рекомендаційні моделі визначаються як системи, які створюють персоналізовані прогнози як результат. Ці системи застосовуються до електронної комерції, музики, фільмів, відео, книг, новин тощо. Залежно від способу створення рекомендацій їх поділяють на: системи рекомендацій на основі вмісту, системи рекомендацій спільної фільтрації та гібридні системи рекомендацій. Тим не менш, у всіх цих методах точність прогнозування цільового результату тісно пов’язана з кількістю інформації, отриманої від активних користувачів, для яких генеруються ці прогнози. У системі рекомендацій інтереси

користувачів, їхні переваги та антипатії використовуються для розробки профілю користувача, який використовується як фільтри. Створення точних профілів користувачів є важливим завданням, і точність продуктивності системи залежить від здатності навчених активних профілів споживачів представляти вподобання користувача. Початкова робота над рекомендаційними системами була з використанням спільної фільтрації, яка рекомендувала користувачам новинні статті та рекомендації музичних альбомів і виконавців із соціальної інформації. Після цього було багато робіт у сфері рекомендаційних систем, які допомагали споживачам ідентифікувати товари, послуги та контент, наприклад книги, дослідницькі статті, фільми, новини, роздрібний бізнес, цифрові продукти, споживчі товари тощо, застосовуючи різні алгоритми, які переглядає різних користувачів і предмети, щоб надати правильні пропозиції.

Лі та ін. [7] запропонував алгоритм для розпаралелювання частого виявлення набору елементів для пошуку прихованого шаблону для підтримки рекомендацій щодо запиту великого набору даних. Цей алгоритм дає змогу зменшити вартість обчислень шляхом розподілу обчислень між вузлами кластера таким чином, що кожен вузол виконує незалежний набір завдань видобутку. Вони досягли найкращої продуктивності завдяки розподілу обробки за допомогою інфраструктури MapReduce на кластер комп'ютерів.

Наджафабаді та інші., [2] представили глибоке навчання — це найшвидший сегмент машинного навчання, який використовує багато рівнів глибоких нейронних мереж для вивчення рівнів представлення та абстракції, що робить сенс даних, зазвичай за допомогою штучних нейронних мереж. Глибоке навчання сприяє значному прогресу в галузях, підприємствах та електронній комерції.

Багато систем рекомендацій для соціальних мереж було розроблено з використанням взаємодії та поведінки користувачів. У [3] запропоновано підхід до аналізу інтересу користувачів до платформи twitter. Вони поєднали аналіз настроїв і класифікацію твітів, аналізуючи теми, які обговорювали користувачі. Реалізація їх роботи дає обнадійливі результати.

III. Методи

У рамках цієї статті систему рекомендацій було реалізовано за допомогою Python Apache Spark ML Pipeline, реалізація альтернативних найменших квадратів. Модель приймає навчальний набір даних як кадр даних (pyspark.sql.DataFrame) і кілька параметрів, які керують процесом створення моделі [3].

Користувачі можуть легко сформулювати мету своїх запитів за допомогою Spark SQL, тоді як Spark виконує більш складний процес оптимізації запитів [4]. З моменту введення визначених користувачем функцій pandas у spark 2.3 [3] користувачі можуть визначати власні python-функції, які можна запускати пакетами, що дає їм гнучкість, їм потрібно створювати запити, які стосуються надзвичайно спеціалізованих випадків використання.

Кожен ідентифікатор моделі, пара ідентифікаторів і майбутній новий рядок повинні бути перевірені Spark SQL. Замість виконання грубого перехресного з'єднання Spark ми можемо просто перебрати всі рядки та пам'ять за допомогою UDF. Виявляється, для певного набору даних така програмна реалізація може майже подвоїти швидкість обробки.

Ідея інвертованих індексів також дуже корисна для оптимізації проблеми. Основний принцип цієї концепції полягає в тому, що кожен рядок сховища функцій містить лише невелику кількість відмінних функцій. Кожен рядок набору даних містить від 10 до 20 характеристик. Тому було б дуже корисно, якби ми змогли знайти техніку для повторення рядків і повернення відповідей, використовуючи 0 функцій на час рядка на відміну від 0 часу

фільтрів. Інвертовані індекси є методом для досягнення цього. Однак загальне уявлення полягає в тому, що нам потрібен алгоритм, який масштабується пропорційно до того, наскільки розрідженими є дані.

Існують альтернативні бібліотеки Python, такі як itertools, які призначені для того, щоб програми Python працювали набагато швидше [1]. Pandas зазвичай налаштовано на простоту використання, а не на продуктивність.

Як виявилося, потрібно використовувати itertools для нашого випадку використання. Reduce в NumPy може бути гідним варіантом для розгляду для інших UDF [3], але це не стосується нашого випадку використання.

Тому можна зробити це в 2,5 рази швидше, ніж наш попередній підхід, використовуючи цей модуль Python, який оптимізований для виконання дуже конкретних речей [2]. Програмна реалізація оптимізаційного методу з використанням itertools дає результат, який близько у 100 разів швидший, ніж нативне рішення Spark SQL.

```
def udf_wrapper(feature_data: pd.DataFrame) → pd.DataFrame:
    for ids in itertools.groupby(feature_data['feature_ids'], key=lambda x: x[0]):
        valid_ids = set()
        for _, id_ in ids:
            included = set(
                itertools.chain.from_iterable(
                    [incl_ids[f_id] for f_id in id_ if id_ in incl_ids]
                )
            )
            excluded = set(
                itertools.chain.from_iterable(
                    [excl_ids[f_id] for f_id in id_ if id_ in incl_ids]
                )
            )
            filtered = incl_ids.difference(excl_ids)
            valid_ids = valid_ids.union(filtered)

        for f_id in valid_ids: counter[f_id] += 1

    return pd.DataFrame([[id_, count] for id_, count in counter.items()])


groupby_udf = pandas_udf(
    f=udf_wrapper,
    returnType=StructType(
        [
            StructField('id', IntegerType()),
            StructField('batchCount', IntegerType())
        ]
    )
)
```

Рис. 1 Програмна реалізація оптимізаційного методу за допомогою itertools

IV. Результати

Результатом дослідження є реалізовані оптимізаційні методи для обробки даних для середовища Apache Spark, а саме: метод інвертованих індексів та за допомогою використання нативного Python (itertools).

Для всіх експериментів використовувалися мова програмування Python 3.10, PyCharm і MacBook Pro з операційною системою Ventura 13.0, процесором Apple M1 Pro 16 ГБ оперативної пам'яті

Оцінивши запропоновані методи, можна сформувати порівняльну характеристику швидкодії існуючих рішень та розроблених, зображену на рисунку 2.

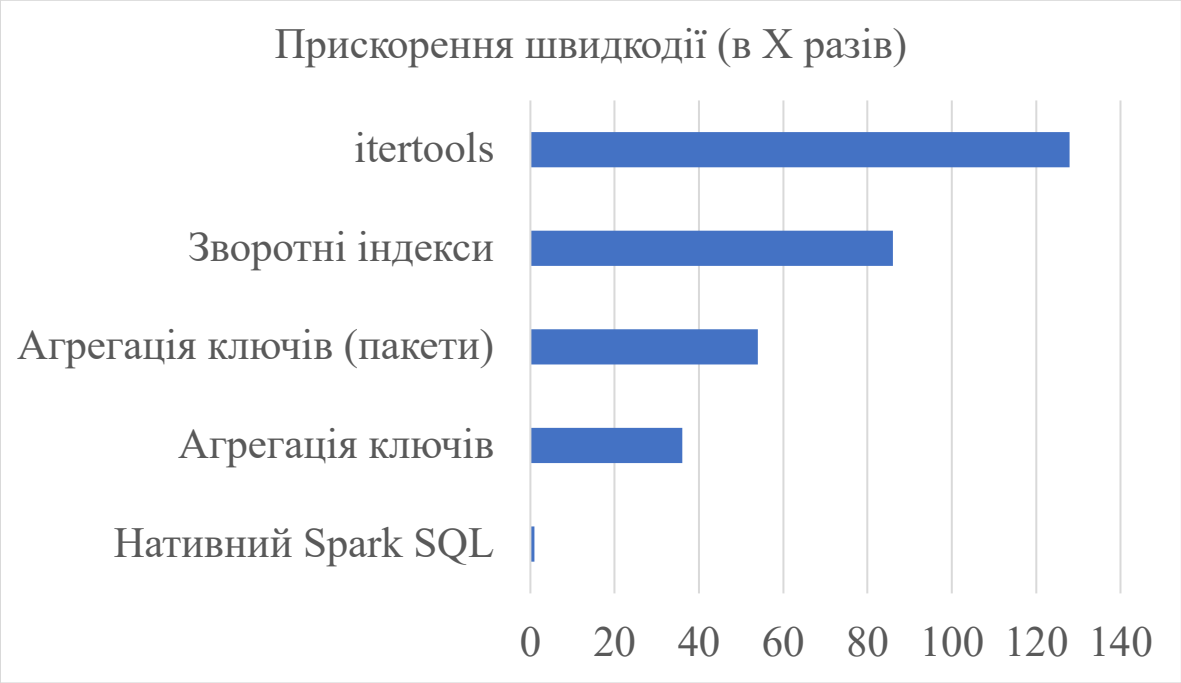

Рис. 2 Порівняння швидкодії методів

Як зображено на рисунку, дані методи дають змогу обробляти дані в рази швидше, порівняно із звичайним підходом в Spark SQL, а саме метод зворотних індексів прискорює обробку даних у 86 разів, а метод за допомогою використання itertools в 128 разів.

V. Обговорення

Запропоновані підходи для оптимізації обробки даних в межах рекомендаційної системи здатні значною мірою прискорити існуючі та нові методи обробки даних не тільки в рекомендаційних системах, а й в інших системах, які використовують обробку великих даних та моделі машинного навчання.

VI. Перспективи подальших досліджень

Запропоновані підходи є перспективними кандидатами на подальші вдосконалення, а саме їх дослідження в інших сферах машинного навчання та великих даних, оскільки можуть принести вагомий внесок та прискорити різного роду системи та алгоритми, які ними використовуються.

Отримані багатообіцяючі результати є лише відправною точкою. Можна розглянути можливість розгортання цієї реалізації на багатовузловому кластері, щоб оцінити його

масштабованість, продуктивність (зокрема час обчислень) і точність у розподіленому режимі.

Інші дослідницькі зусилля слід присвятити експерименту з іншими інструментами та фреймворком великих даних, таким як Apache Mahout, і порівняти його продуктивність з Apache Spark.

VII. Висновки

Системи рекомендацій були давно розроблені та інтегровані на багатьох веб-сайтах, особливо на веб-сайтах електронної комерції, давно. Вони довели свою потужність у наданні персоналізованого, налаштованого веб-вмісту різним користувачам, рекомендуючи вміст (тобто елементи у випадку веб-сайту електронної комерції), що цікавить кожного користувача, і таким чином пом'якшує проблему перевантаження інформацією для користувача.

Для систем рекомендацій існують різні техніки та підходи. Одним із широко використовуваних методів є спільне фільтрування, яке збирає записи про взаємодію між користувачами та товарами, історію покупок, щоб визначити смак користувача та таким чином рекомендувати товари, які відповідають його смаку.

Ця дослідницька робота робить внесок у сучасні рекомендаційні системи в тому сенсі, що забезпечує реалізацію великомасштабної рекомендаційної системи на основі Apache Spark. Це стало результатом інтенсивного вивчення літератури, а також проведення багатьох експериментів з використанням мови програмування Python на основі Apache Hadoop і Spark. Дослідження охоплювало кілька тем, а саме: феномен великих даних, різні техніки та підходи до рекомендаційних систем разом із їхніми плюсами та мінусами, виклики, пов'язані з великими даними щодо рекомендаційних.

Ця робота вирішила проблему масштабованості та оптимізації обробки даних шляхом використання реалізованих оптимізаційних методів на основі розробленої рекомендаційної системи.

VIII. Конфлікти інтересів

Автор зазначає, що дослідження проводилося за відсутності будь-яких комерційних чи фінансових відносин, які можна було б витлумачити як потенційний конфлікт інтересів.

Література

[1] Percival H.J.W., Gregory B., Architecture Patterns with Python Enabling Test- Driven Development, Domain-Driven Design, and Event-Driven Microservices, O'Reilly Media, 2020, p .304.

[2] Najafabadi, Maryam & Villanustre, Flavio & Khoshgoftaar, Taghi & Seliya, Naeem & Wald, Randall & Muharemagic, Edin, Deep learning applications and challenges in big data analytics, Journal of Big Data, 2015, pp. 34-45.

[3] Mengle S., Gurmendez M., Mastering Machine Learning on AWS Advanced Machine Learning in Python Using SageMaker, Apache Spark, and TensorFlow, Packt Publishing, 2019, p. 306.

[4] Ramalho L., Fluent Python Clear, Concise, and Effective Programming, O'Reilly Media, 2015, p. 790.

[5] Chambers B., Zaharia M., Spark The Definitive Guide: Big Data Processing Made Simple, O'Reilly Media, 2018, p. 603.

[6] Li H, Wang Y, Zhang D, Zhang M, Chang EY, Pfp parallel fp-growth for query recommendation, In Proceedings of the 2008 ACM conference on recommender systems, RecSys'08, Lausanne, ACM Press, 2008, p. 107.

[7] Sandy Ryza, Advanced Analytics with PySpark, Patterns for Learning from Data at Scale Using Python and Spark, 2022, pp. 120-168.

[8] Romeo Kienzler, Md. Rezaul Karim, Sridhar Alla, Siamak Amirghodsi, Apache Spark 2: Data Processing and Real-Time Analytics, 2018, pp 20-88.