DOI: https://doi.org/10.36910/6775-2524-0560-2025-61-20
УДК 004.89
Марченко Ірина Федорівна, к.т.н., доцент,
https://orcid.org/0000-0002-4566-3866
Бурлаков Олександр Олександрович, магістр
https://orcid.org/0009-0005-3412-3847
Державний вищий навчальний заклад «Приазовський державний технічний університет», м. Дніпро, Україна

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ РЕКОМЕНДАЦІЙ В СЕРЕДОВИЩІ C#

Марченко І. Ф., Бурлаков О. О. Порівняльний аналіз алгоритмів рекомендацій у середовищі C#. Стаття присвячена порівняльному аналізу ефективності алгоритмів рекомендаційних систем в екосистемі .NET. Розглянуто проблему інформаційного перевантаження та важливість персоналізації. Описано реалізацію двох ключових підходів: класичної матричної факторизації (MF) з використанням фреймворку ML.NET та сучасної нейромережевої архітектури Neural Collaborative Filtering (NCF/NeuMF) на базі SciSharp Stack (TorchSharp). Для порівняння точності прогнозування та якості ранжування проведено серію експериментальних досліджень на загальнодоступних наборах даних MovieLens та Amazon Reviews. Проаналізовано метрики RMSE, MAE та NDCG@K. Результати демонструють, що на досліджених даних модель матричної факторизації стабільно перевершує нейромережевий підхід у точності, робастності до обсягу даних та швидкодії. Дослідження підтверджує висновки світових аналізів про те, що класичні, добре налаштовані моделі часто є ефективнішими за складніші нейромережеві архітектури.
Ключові слова: рекомендаційні системи, C#, .NET, ML.NET, SciSharp Stack, матрична факторизація (MF), нейронна колаборативна фільтрація (NCF), RMSE.

Marchenko I., Burlakov O. Comparative Analysis of Recommendation Algorithms in the C# Environment. The article is devoted to a comparative analysis of the effectiveness of recommendation system algorithms in the .NET ecosystem. The problem of information overload and the importance of personalization are considered. The implementation of two key approaches is described: classical matrix factorization (MF) using the ML.NET framework and the modern neural network architecture Neural Collaborative Filtering (NCF/NeuMF) based on SciSharp Stack (TorchSharp). To compare the accuracy of predictions and the quality of rankings, a series of experimental studies were conducted on publicly available MovieLens and Amazon Reviews datasets. The RMSE, MAE, and NDCG@K metrics were analyzed. The results show that, on the data studied, the matrix factorization model consistently outperforms the neural network approach in terms of accuracy, robustness to data volume, and speed. The study confirms the conclusions of global analyses that classic, well-tuned models are often more effective than more complex neural network architectures.
Keywords: recommendation systems, C#, .NET, ML.NET, SciSharp Stack, matrix factorization (MF), neural collaborative filtering (NCF), RMSE.

Постановка проблеми у загальному вигляді та її зв'язок з важливими науковими чи практичними завданнями.

У цифрову епоху, коли щоденно створюється колосальний обсяг інформації (за оцінками, понад 400 мільйонів терабайт щодня у 2024 році), користувачі стикаються з явищем інформаційного перевантаження [1, 2]. Це ускладнює вибір релевантних об'єктів і прийняття рішень. Рекомендаційні системи (РС) покликані зменшити це когнітивне навантаження шляхом персоналізації інформаційного середовища.

Персоналізовані рекомендації нині є важливим чинником конкурентоспроможності цифрових платформ. У сфері електронної комерції персоналізація безпосередньо впливає на підвищення конверсії, середньої вартості замовлення та утримання клієнтів [3, 4]. Дослідження показують, що майже 40% споживачів залишали веб-сайти через надмірний вибір, що підкреслює необхідність ефективної фільтрації контенту [3, 4].

У контексті розробки програмного забезпечення, екосистема .NET (C#), що активно використовується у корпоративному секторі, довгий час мала обмежений вибір інструментів для машинного навчання. Ситуація змінилася з появою офіційного фреймворку ML.NET від Microsoft [5], що дозволяє реалізовувати класичні алгоритми машинного навчання, та стеку SciSharp Stack [6], який інтегрує у C# бібліотеки для глибокого навчання (наприклад, TorchSharp, аналог PyTorch). Це відкриває шлях до реалізації як класичних методів (матрична факторизація), так і сучасних нейромережевих архітектур (NCF) у межах єдиного технологічного середовища. Така диверсифікація інструментів породжує практичне завдання: провести порівняльний аналіз ефективності цих підходів саме в екосистемі C#/.NET.

Аналіз останніх досліджень і публікацій, у яких започатковано розв'язання даної проблеми.

Сучасні дослідження в галузі РС активно розвиваються у напрямку глибокого навчання (DL) для подолання обмежень класичних методів. У роботі Zhang та ін. [7] надається комплексний огляд систем рекомендацій на основі глибокого навчання. Автори класифікують сучасні підходи (включаючи MLP, CNN, GNN) та аналізують, як глибоке навчання допомагає моделювати складні нелінійні взаємозв'язки та враховувати додаткову інформацію (side information), що є обмеженням для класичних методів.

У статті He та ін. [8] запропоновано саму архітектуру Нейронної Колаборативної Фільтрації (NCF). Автори стверджують, що лінійний скалярний добуток, який використовується у матричній факторизації, є вузьким місцем для моделювання складної поведінки користувачів. Вони пропонують замінити його на багатошаровий перцептрон (MLP) для вивчення нелінійних взаємозв'язків. Також у цій роботі представлено гібридну модель NeuMF, що поєднує узагальнену матричну факторизацію (GMF) та MLP.

Проте домінування DL-підходів було поставлено під сумнів. У роботі Dacrema та ін. [9] представлено критичний аналіз прогресу в галузі РС. Автори емпірично доводять, що багато нових, складних моделей глибокого навчання, про які повідомлялося в літературі, насправді не можуть перевершити простіші, але ретельно налаштовані класичні алгоритми (наприклад, варіанти k-NN або MF). Це підкреслює проблему відтворюваності та методології тестування у сучасних дослідженнях.

Дослідження Rendle та ін. [10] є прямим переглядом оригінальної статті про NCF. Автори доводять, що перевага NCF над MF у статті He та ін. [8] могла бути зумовлена особливостями експериментального налаштування (наприклад, вибором функції втрат). При ретельному та справедливому порівнянні, добре налаштована класична модель матричної факторизації (MF) демонструє результати, співставні або навіть кращі за складніші нейромережеві архітектури NCF.

Продовження досліджень у цьому напрямі підтверджує попередні висновки. Зокрема, у розширеній версії роботи Dacrema та ін. [11] продемонстровано, що проблема відтворюваності результатів у сфері рекомендаційних систем залишається актуальною. Автори показують, що значна частина нових нейромережевих підходів, опублікованих у провідних виданнях, при суворих експериментальних умовах не здатна стабільно перевершувати добре налаштовані базові моделі, такі як матрична факторизація та методи найближчих сусідів (k-NN). Це свідчить про певну стагнацію в галузі, коли зростання складності моделей випереджає приріст їхньої практичної ефективності.

Цікавим продовженням розвитку підходів після NCF стала поява моделей на основі графових нейронних мереж. Однією з ключових робіт у цьому напрямі є LightGCN від He та ін. [12], у якій автори показали, що типові для складних GNN-архітектур операції трансформації ознак та нелінійні функції активації майже не покращують якість рекомендацій, а інколи навіть погіршують її. Спрощення моделі шляхом усунення цих компонентів дозволило досягти результатів рівня state-of-the-art, що додатково підтверджує ефективність менш складних підходів у колаборативній фільтрації.

Науковий інтерес до зазначеної проблематики зумовлює потребу в порівняльному аналізі алгоритмів у контексті промислових систем, де широко використовуються Java та C#. У цьому дослідженні увагу зосереджено на специфіці середовища C#, оскільки реалізації рекомендаційних моделей на основі оптимізованого ML.NET та більш гнучкого TorchSharp можуть демонструвати відмінності у продуктивності та масштабованості.

Мета дослідження.

Порівняння впливу алгоритмів (класичної матричної факторизації в ML.NET та нейронної колаборативної фільтрації в SciSharp Stack) на точність прогнозування уподобань користувачів з допомогою мови програмування C# та середовища .NET.

Виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів.

Для проведення експериментів було розроблено програмний комплекс мовою C# у середовищі Visual Studio 2022. Архітектура програмного забезпечення була розроблена для порівняння двох фундаментально різних підходів до рекомендацій в екосистемі .NET.

Для реалізації MF використано бібліотеку Microsoft.ML [5]. Центральним компонентом є MatrixFactorizationTrainer, який реалізує оптимізований алгоритм матричної факторизації (подібний до ALS) і дозволяє ефективно навчати моделі для прогнозування явних рейтингів.

Для реалізація NCF використано бібліотеку TorchSharp зі стеку SciSharp [6]. TorchSharp є .NET-обгорткою для бібліотеки libtorch (PyTorch), що надає всі необхідні інструменти для побудови та навчання моделей глибокого навчання. Було реалізовано гібридну архітектуру NeuMF, що поєднує гілки GMF та MLP, як описано в роботі He та ін., у якій з'являється уперше дана архітектура [8]. Ця модель складалася з шарів ембедингів (Embedding Layer) для користувачів та об'єктів, які паралельно подавалися у дві гілки GMF та MLP.

Суть гілки GMF (Generalized Matrix Factorization) була в тому, щоб виконуватипоелементний добуток векторів для моделювання лінійних взаємодій.

У той час гілка MLP (Multi-Layer Perceptron) відповідає за нелінійну взаємодію, що є відмінним від класичного методу матричної факторизації. Вона здійснювала конкатенацію векторів та кілька повнозв'язних шарів (з активацією ReLU) для моделювання складних нелінійних взаємозв'язків.

У кінцевому результаті виходи цих обох гілок конкатенувалися перед фінальним вихідним шаром, що генерував прогноз рейтингу.

Дослідження проводилось на трьох наборах даних для оцінки точності та відповідності вподобанням користувача.

MovieLens (100k) – перший набір даних із інформацією про виставлені рейтинги фільмам користувачами. Крім рейтингів додатково є інформація про вік, стать та зайнятість користувачів, а також про жанр та рік випуску фільмів. Даний тестовий набір використовувався для 2 різних експериментів. У першому було використано повний набір даних, що становить 100 000 рейтингів. У другому випадку перевірялась якість рекомендаційних систем на 20% від початкової кількості рейтингів, тобто 20 000.

Amazon Books – інший набір даних, який містить інформацію про книги, її автора, категорію, рік публікації. Даний набір є більшим за попердній і містить 995 800 рядків з рейтингами книг.

Усі моделі навчалися на 80% даних та тестувалися на 20% (для Amazon Books test fraction = 0.1). Експерименти включали варіацію гіперпараметрів: для MF (ML.NET), а саме Approximation Rank (кількість латентних факторів) та Lambda (регуляризація); для NCF (TorchSharp) – Embedding Dim, Hidden Layers (архітектура MLP) та Learning Rate. Кожна конфігурація оцінювалася за метриками RMSE (Root Mean Square Error) та MAE (Mean Absolute Error) для точності прогнозування, а також NDCG@K (Normalized Discounted Cumulative Gain) для якості ранжування.

Експеримент 1: Повний набір MovieLens (100k)

На цьому еталонному наборі даних модель MF, реалізована в ML.NET, продемонструвала значно вищу точність прогнозування. Перші 5 найкращих результатів для кожного алгоритму впорядкованих за меншим значенням метрики RMSE можна побачити у таблиці 1.

Таблиця 1. Топ 5 результатів за RMSE кожної моделі на наборі даних Movie Lens

MF	NCF	NCF_Add
0.1825	0.228	0.2290
0.1826	0.2286	0.2291
0.1834	0.2287	0.2295
0.1839	0.2294	0.23
0.1845	0.2297	0.2302

Як видно з Таблиці 1, найкращий показник RMSE для MF (ML.NET) склав 0.1825, тоді як найкращий RMSE для NCF (TorchSharp) – 0.228. Це демонструє статистично значущу перевагу класичної моделі у завданні регресії рейтингу. Спроба покращити NCF шляхом додавання метаданих, таких як жанр чи вік, не дала позитивного ефекту для точності (у таблиці результати під назвою NCF_Add - RMSE 0.2290), але, як видно з Рис. 1, значно збільшила час навчання.

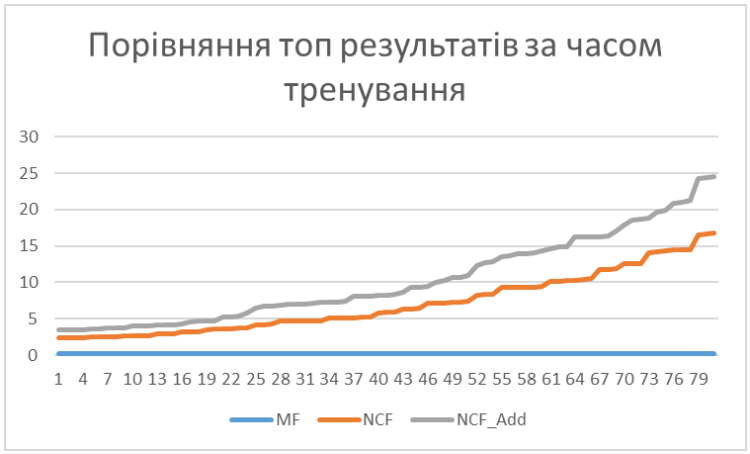

Рис. 1. Крива часу тренування на наборі даних MovieLens з різними конфігураціями для алгоритмів MF та NCF

Критичною є різниця у швидкості тренування між даними алгоритмами. MF (ML.NET) навчалася менш ніж за 0.2 секунди. Це дозволяє проводити часті, майже в реальному часі, оновлення моделі. Для користувачів у такому випадку майже не буде помітно, чи модель уже була раніше натренована чи вона тренується у реальному часі.

На противагу, NCF (TorchSharp), залежно від конфігурації, потребувала від 3 до 18 секунд. NCF Add (сіра лінія на Рис. 1) вимагала ще більше часу, до 25 секунд. Це робить NCF-підхід у C# значно дорожчим в експлуатації.

Експеримент 2: Набір Amazon Books (995k)

На більшому наборі даних, який майже в 10 разів перевищує MovieLens, тенденція збереглася. MF (ML.NET) стабільно показувала кращі результати за метриками RMSE та MAE, ніж архітектура NCF та NCF з додатковими колонками.

Таблиця 2. Топ 5 результатів за RMSE кожної моделі на наборі даних Amazon Books

MF	NCF	NCF_Add
0.3484	0.3778	0.3896
0.3486	0.3875	0.3933
0.3488	0.3878	0.4017
0.349	0.3898	0.403
0.3498	0.3921	0.4056

Результати на наборі Amazon Books (Таблиця 2) підтвердили висновки, отримані на MovieLens, але у більшому масштабі. MF (ML.NET) не лише зберегла перевагу в точності (найкращий RMSE 0.3484 проти 0.3778 у NCF), але й продемонструвала набагато кращу масштабованість часу навчання.

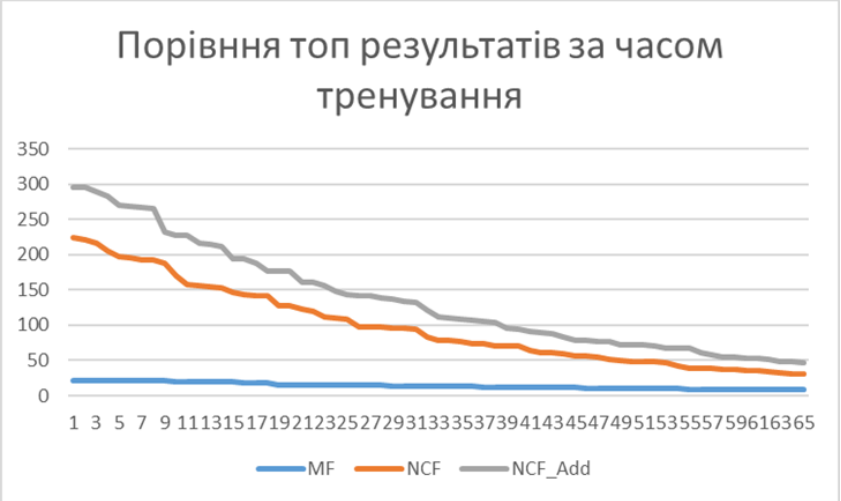

Рис. 2. Крива часу тренування на наборі даних Amazon Books з різними конфігураціями для алгоритмів MF та NCF

Як видно з Рис. 2, час навчання MF зріс до 20-30 секунд, залишаючись контрольованим. Натомість час NCF зріс значно сильніше, досягаючи понад 300 секунд для складних конфігурацій. Це вказує на те, що NCF у C# (TorchSharp) є обчислювально значно дорожчою та гірше масштабується для даного завдання. У випадку впровадження даного алгоритму у реальну систему, необхідно буде здійснювати розпаралелення і оптимізацію тренування, щоб вирішити проблему довгого часу тренування.

Експеримент 3: 20% набору Movie Lens (20k)

Цей експеримент був вирішальним для оцінки узагальнювальної здатності моделей в умовах дефіциту даних.

Таблиця 3. Топ 5 результатів за RMSE кожної моделі на наборі даних Movie Lens (20%)

MF	NCF	NCF_Add
0.1956	0.9795	0.9793
0.1958	0.9808	0.9795
0.1958	0.9822	0.9802
0.1959	0.984	0.9804
0.1964	0.9847	0.9827

Результати цього експерименту у таблиці 3 є найбільш показовими. Якість MF (ML.NET) залишилася стабільно високою (RMSE ~0.1956) , що є дуже близьким до результату на повному наборі (0.1825).

Натомість NCF (TorchSharp) продемонструвала повну нездатність до узагальнення. Показники RMSE деградували. RMSE зріс з ~0.228 до ~0.9795. Це чітко вказує на те, що NCF є жадібною до даних ("data-hungry") моделлю. В умовах дефіциту даних (20k записів), її складна архітектура призводить до перенавчання (Overfitting), оскільки модель, ймовірно, «запам'ятовувала» шум замість вивчення прихованих патернів. Це робить NCF абсолютно непридатною для сценаріїв «холодного старту» або нішевих ринків з малою кількістю взаємодій.

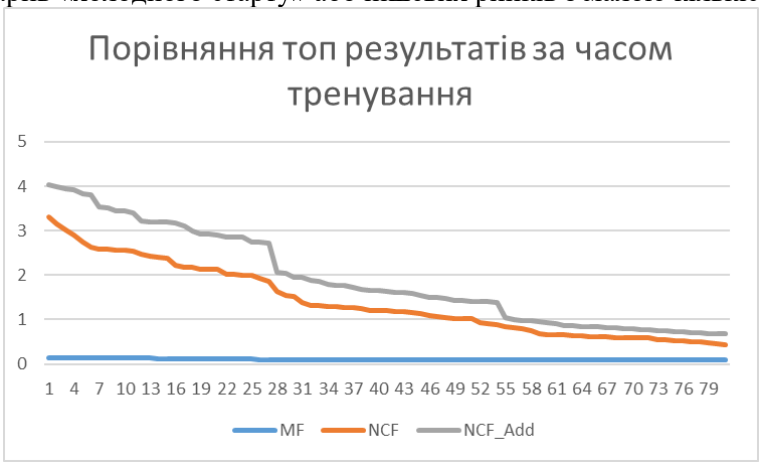

Рис. 3. Крива часу тренування на наборі даних MovieLens (20%) з різними конфігураціями для алгоритмів MF та NCF

Також з кривої часу тренування на рис. 3 чітко видно загальну для всіх датасетів з експериментів тенденцію, що для алгоритму MF потрібно менше часу тіж для NCF. Також NCF з додатковими колонками для тренування нейронної мережі займає більше часу ніж NCF без них.

Отже, на всіх трьох наборах даних класична MF (ML.NET) виявилася точнішою та стабільнішою. Також її перевага на малому наборі даних свідчить про кращу узагальнювальну здатність. Модель NCF, маючи значно більшу кількість параметрів (ваг у нейронній мережі), продемонструвала високу схильність до перенавчання (Overfitting). На малих даних (20k) вона, ймовірно, «запам'ятовувала» шум замість вивчення прихованих патернів, що є типовою проблемою для моделей глибокого навчання.

Додатковим фактором, що міг вплинути на низьку ефективність NCF, є сама постановка задачі. Обидві моделі оптимізувалися для задачі регресії (прогнозування явного рейтингу) з використанням метрик RMSE/MAE. Проте, як зазначається у [9, 10], багато сучасних DL-моделей (включаючи NCF) часто демонструють кращі результати на завданнях ранжування з неявним

зворотним зв'язком, використовуючи специфічні функції втрат (наприклад, Binary Cross-Entropy або BPR). Класична MF, оптимізована під RMSE, виявилася більш адекватною саме для задачі прогнозування явного рейтингу.

MF (ML.NET) є надзвичайно швидкою (навчання за <0.2с на 100k даних) та ефективно працює на CPU. NCF (TorchSharp) є значно повільнішою (10-18с) і для масштабування на промислові дані вимагатиме GPU-прискорення, що значно ускладнює інфраструктуру розгортання.

Висновки з даного дослідження і перспективи подальших розвідок у цьому напрямку. У роботі проведено комплексний аналіз алгоритмів рекомендаційних систем у середовищі C# на основі розробленої експериментальної платформи. Отримані результати показали, що за умов прогнозування явних рейтингів на наборах даних обсягом до 1 млн записів класична матрична факторизація (MF), реалізована засобами Microsoft.ML, стабільно забезпечує вищу точність (RMSE, MAE) та якість ранжування (NDCG@K), ніж нейромережева архітектура NCF/NeuMF, реалізована у SciSharp Stack (TorchSharp).

Модель NCF продемонструвала нижчу узагальнювальну здатність, особливо на обмежених обсягах даних, що вказує на схильність до перенавчання. Крім того, MF характеризується значно меншими обчислювальними витратами й коротшим часом навчання. Наукова новизна роботи полягає у першому прямому порівнянні цих підходів у межах єдиної .NET-екосистеми.

З практичної точки зору MF у Microsoft.ML наразі є доцільнішим вибором для розробників C#/.NET, оскільки забезпечує високу якість рекомендацій за мінімальних ресурсних витрат.

Перспективи подальших досліджень включають аналіз моделей на ще більших наборах даних (десятки мільйонів записів). Також доцільно дослідити в середовищі C# (TorchSharp) більш сучасні архітектури глибокого навчання, зокрема моделі на основі графів (наприклад, LightGCN), які вважаються state-of-the-art для колаборативної фільтрації, та моделі на основі Transformer (наприклад, SASRec) для задач послідовних рекомендацій. Майбутні дослідження NCF у .NET-середовищі мають також зосередитися на завданнях ранжування з неявним зворотним зв'язком та відповідними функціями втрат (BPR).

Список бібліографічного опису

1. Wedia. Information Overload in 2025: Risks, Impact & DAM Solutions. 2024. Режим доступу: https://www.wedia-group.com/blog/how-to-avoid-information-overload-for-your-clients-and-employees
2. RPI News. Information Overload Is a Personal and Societal Danger. 2024. Режим доступу: https://news.rpi.edu/2024/03/13/information-overload-personal-and-societal-danger
3. One World Direct. The Ultimate 2024 Guide on Ecommerce Personalization. - 2024. - Режим доступу: https://owd.com/ecommerce-personalization/
4. Storyly. Why Personalization Matters in eCommerce. 2024. Режим доступу: https://www.storyly.io/post/why-personalization-is-important-for-ecommerce
5. Microsoft. ML.NET. 2024. Режим доступу: https://dotnet.microsoft.com/en-us/apps/ai/ml-dotnet
6. SciSharp STACK. TorchSharp. 2024. Режим доступу: https://scisharp.github.io/TorchSharp/
7. Zhang S., Yao L., Sun A., Tay Y. Deep learning based recommender system: A survey and new perspectives // ACM Computing Surveys. 2019. Vol. 52, No. 1. P. 1-38.
8. He X., Liao L., Zhang H., Nie L., Hu X., Chua T.-S. Neural collaborative filtering // Proceedings of the 26th International Conference on World Wide Web (WWW). 2017. P. 173-182.
9. Dacrema M. J., Cremonesi P., Jannach D. Are We Really Making Much Progress? A Worrying Analysis of Recent Recommender Systems // Proceedings of the 13th ACM Conference on Recommender Systems (RecSys'19). New York: ACM. 2019. P. 101-109.
10. Rendle S., Krichene W., Zhang L., Anderson J. Neural collaborative filtering vs. matrix factorization revisited // Proceedings of the 14th ACM Conference on Recommender Systems (RecSys). New York: ACM, 2020. P. 240-248.
11. Dacrema M. F., Boglio S., Cremonesi P., Jannach D. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research // ACM Transactions on Information Systems (TOIS). – 2021. – Vol. 39, No. 2. – P. 1–49.
12. He X., Deng K., Wang X., Li Y., Zhang Y., Wang M. LightGCN: Simplifying and Powering Graph Convolutional Networks for Recommendation // Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. – 2020. – P. 639–648.

References

1. Wedia. Information Overload in 2025: Risks, Impact & DAM Solutions. 2024. Retrieved from: https://www.wedia-group.com/blog/how-to-avoid-information-overload-for-your-clients-and-employees
2. RPI News. Information Overload Is a Personal and Societal Danger. 2024. Retrieved from: https://news.rpi.edu/2024/03/13/information-overload-personal-and-societal-danger
3. One World Direct. The Ultimate 2024 Guide on Ecommerce Personalization. - 2024. - Retrieved from: https://owd.com/ecommerce-personalization/

4. Storyly. Why Personalization Matters in eCommerce. 2024. Retrieved from: https://www.storyly.io/post/why-personalization-is-important-for-ecommerce
5. Microsoft. ML.NET. 2024. Retrieved from: https://dotnet.microsoft.com/en-us/apps/ai/ml-dotnet
6. SciSharp STACK. TorchSharp. 2024. Retrieved from: https://scisharp.github.io/TorchSharp/
7. Zhang S., Yao L., Sun A., Tay Y. Deep learning based recommender system: A survey and new perspectives // ACM Computing Surveys. 2019. Vol. 52, No. 1. P. 1-38.
8. He X., Liao L., Zhang H., Nie L., Hu X., Chua T.-S. Neural collaborative filtering // Proceedings of the 26th International Conference on World Wide Web (WWW). 2017. P. 173-182.
9. Dacrema M. J., Cremonesi P., Jannach D. Are We Really Making Much Progress? A Worrying Analysis of Recent Recommender Systems // Proceedings of the 13th ACM Conference on Recommender Systems (RecSys'19). New York: ACM. 2019. P. 101-109.
10. Rendle S., Krichene W., Zhang L., Anderson J. Neural collaborative filtering vs. matrix factorization revisited // Proceedings of the 14th ACM Conference on Recommender Systems (RecSys). New York: ACM, 2020. P. 240-248.
11. Dacrema M. F., Boglio S., Cremonesi P., Jannach D. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research // ACM Transactions on Information Systems (TOIS). – 2021. – Vol. 39, No. 2. – P. 1–49.
12. He X., Deng K., Wang X., Li Y., Zhang Y., Wang M. LightGCN: Simplifying and Powering Graph Convolutional Networks for Recommendation // Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. – 2020. – P. 639–648.