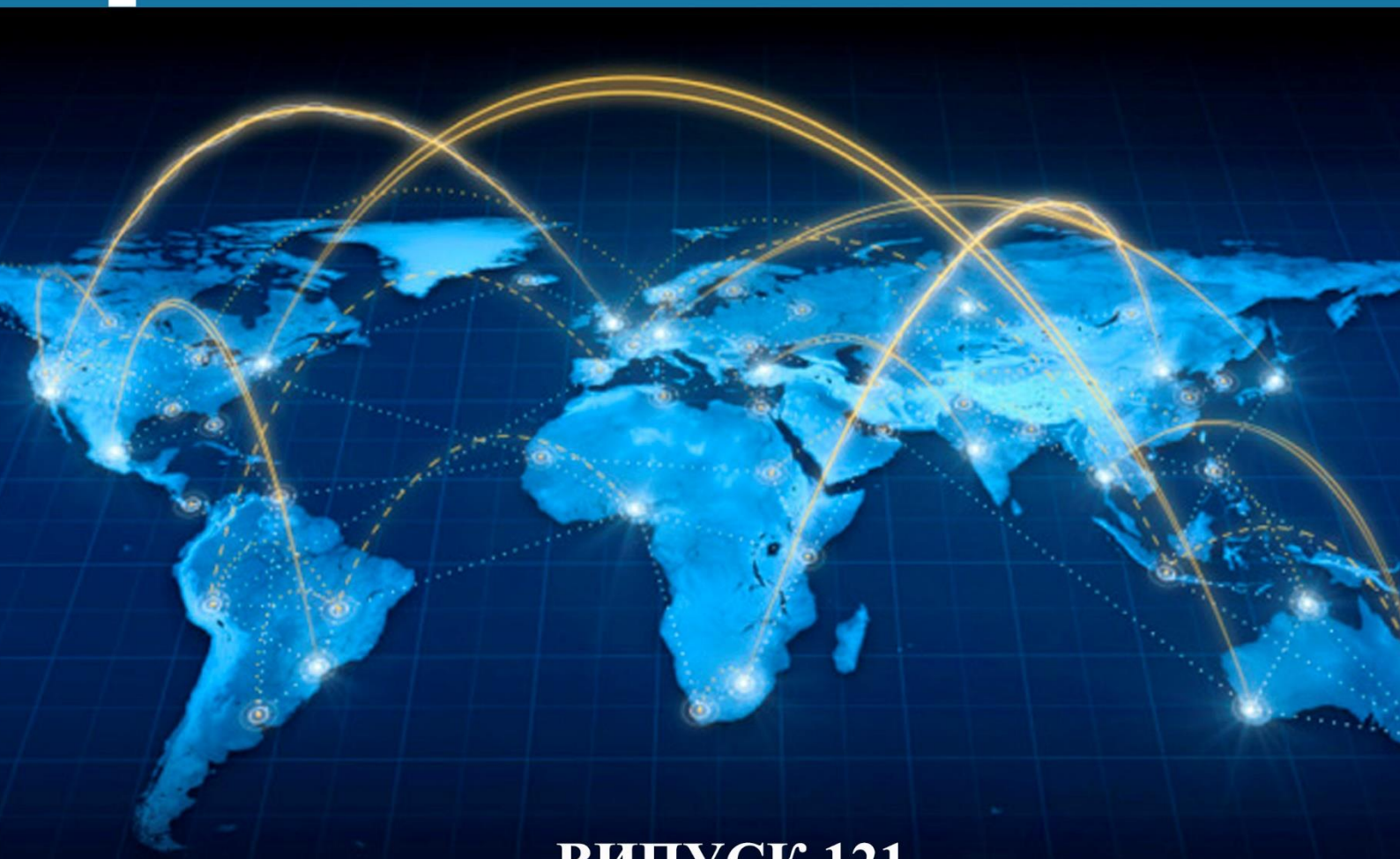
МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА
ІНТЕРНЕТ-КОНФЕРЕНЦІЯ

«ТЕНДЕНЦІЇ ТА ПЕРСПЕКТИВИ РОЗВИТКУ НАУКИ І ОСВІТИ В УМОВАХ ГЛОБАЛІЗАЦІЇ»



ВИПУСК 121

30 вересня 2025 р.

м. Переяслав

УНІВЕРСИТЕТ ГРИГОРІЯ СКОВОРОДИ
В ПЕРЕЯСЛАВІ

Рада молодих учених університету

Матеріали
Міжнародної науково-практичної інтернет-конференції
«ТЕНДЕНЦІЇ ТА ПЕРСПЕКТИВИ РОЗВИТКУ НАУКИ І ОСВІТИ В УМОВАХ ГЛОБАЛІЗАЦІЇ»
30 вересня 2025 року
Вип. 121

Збірник наукових праць

Переяслав – 2025

УДК 001+37(100)
ББК 72.4+74(0)
Т 33

Матеріали Міжнародної науково-практичної інтернет-конференції «Тенденції та перспективи розвитку науки і освіти в умовах глобалізації»: Зб. наук. праць. Переяслав, 2025. Вип. 121. 222 с.

Збірник матеріалів конференції вміщує результати наукових досліджень наукових співробітників, викладачів вищих навчальних закладів, докторантів, аспірантів, студентів з актуальних проблем гуманітарних, природничих і технічних наук

Відповідальність за грамотність, автентичність цитат, достовірність фактів і посилань несуть автори публікацій

2. Barba L. A. Defining the Role of Open Source Software in Research Reproducibility. *Computer.* 2022. Vol. 55, no. 8. P. 40–48. URL: https://doi.org/10.1109/mc.2022

3. Duan C. Advancing open education through open-source software: examining UTAUT 2 factors in adoption and implementation. *Asian Association of Open Universities Journal.* 2024. URL: https://doi.org/10.1108/aaouj-09-2024-0119

4. Nirmani I.A.P. Barriers to digital participation in developing countries: Identifying technological, social, and cultural obstacles to community involvement. *GSC Advanced Research and Reviews.* 2025. Vol. 23, no. 2. P. 061–071. URL: https://doi.org/10.30574/gscarr.2025.23.2.0130

5. Razi M., Batan A. Opportunities and Challenges of Cloud Computing in Developing Countries. *Artificial Intelligence in Society.* 2023. Vol. 3, No 1. P. 1–8. URL: https://researchberg.com/index.php/ai/article/view/93

6. Xu Q., Yu Q., Qian L. Analysis and Evaluation of Open Source Scientific Entity Datasets. Data Intelligence. 2024. URL: https://doi.org/10.3724/2096-7004.di.2024.0009

УДК 004.94

Yehor Momot, Maryna Vyshnevska
(Kyiv, Ukraine)

MODERN ARCHITECTURES FOR INFORMATION RETRIEVAL AND RECOMMENDER SYSTEMS

This paper explains, in simple words, how modern search and recommendation systems work. We start from keyword matching (BM25) and then describe vector-based methods (dual-encoders), token-level matching (ColBERT) and re-ranking. We also show why fast vector indexes like FAISS and HNSW are needed to search large collections. For recommendations, we outline the two-stage design (candidate generation → ranking) and neural models such as Neural Collaborative Filtering. We finish with evaluation notes and practical trade-offs.

Keywords: *information retrieval; recommender systems; BM25; dense retrieval; vector indexes; two-stage ranking; evaluation.*

У цій статті простими словами пояснюється, як працюють сучасні системи пошуку та рекомендацій. Ми починаємо з пошуку за ключовими словами (BM25), а потім описуємо векторні методи (подвійні кодери), пошук на рівні токенів (ColBERT) та переранжування. Ми також показуємо, чому для пошуку у великих колекціях потрібні швидкі векторні індекси, такі як FAISS та HNSW. Щодо рекомендацій, ми описуємо двоступеневий дизайн (генерування кандидатів → ранжування) та нейронні моделі, такі як Neural Collaborative Filtering. На завершення ми наводимо примітки щодо оцінки та практичні компроміси.

Ключові слова: *пошук інформації; системи рекомендацій; BM25; щільний пошук; векторні індекси; двоступеневе ранжування; оцінка*

Search (IR) and recommendation (RS) solve very similar problems. In search, the user enters a query and expects to receive high-quality documents. In recommendations, the system suggests items that may be of interest to the user. In both cases, the collection is huge, and the response must be provided quickly. Therefore, many real-world systems use a simple idea: a fast first pass to collect candidates and a slower second pass to rank them carefully [1, pp. 191–193]. This division makes it easier to scale and debug the entire pipeline. The document and query can be represented as lists of words with weights. BM25 is a practical formula that increases the weight of important words but normalises the length of the document. It works okay without any training data and remains a strong choice for the first pass in real-world systems [6, pp. 333–340; 5, pp. 1–4].

This separation also corresponds to classical IR intuition: simple matching identifies a set that is likely to contain the answer, while a separate ranking stage applies a more precise relevance assessment [5, pp. 1–4]. In the BM25 family, term frequencies are saturated so that very frequent words do not dominate, inverse document frequency reduces the weight of common words, and the length normalisation parameter controls how heavily longer documents are penalised. Smart default settings already provide a solid foundation and are easy to configure in production; it is common practice to first fix the BM25 base and then add trained components [6, pp. 333–389].

Instead of exact word matching, dense search maps texts to a shared vector space so that similar meanings are close together. However, using only one vector per text can lead to the loss of subtle signals at the word level, such as names or numbers. ColBERT stores a vector for each token and compares tokens at a later stage of processing. This preserves most of the accuracy of heavy cross-encoders but remains indexed and fast at runtime [4, pp. 39–41]. A practical pipeline is BM25 → dense search → ColBERT re-ranking. This combination covers exact words, semantic matches, and final accuracy.

A useful way to understand ColBERT is that it defers heavy computation to the efficient generation of candidates. Instead of evaluating a document with a single scalar product, query tokens find their best matching document tokens and then aggregate these matches. This ‘late interaction’ allows us to pre-compute and index document token embeddings, while still capturing many fine-grained patterns that single-vector models miss [4, pp. 39–43]. Combined with lightweight

The first stage of the search results in a conveyor that is accurate and can be deployed on a large scale.

Vector search requires special indexes for speed. FAISS groups vectors into clusters, compresses them using product quantisation, and uses very fast GPU operations. This makes billion-scale searches practical [3, pp. 1–3]. In practice, many production systems also use approximate nearest neighbour graph structures; the exact choice depends on the latency/replay budget. Engineers tune a small set of parameters to find a compromise between quality and speed.

In FAISS, in particular, it is common practice to build an inverted file (IVF) over centroids and check only a small number of lists during a query; product quantisation (PQ) stores compact codes for vectors within each list so that we can quickly scan many candidates on the CPU or GPU [3, pp. 1–3]. The number of lists and the number of lists checked per query determine the recall/latency boundary, while the PQ configuration determines the memory/accuracy boundary. This division of tasks is convenient from an operational point of view: we can first select latency targets and then gradually optimise memory and recall [3, pp. 1–3].

Large-scale recommendation systems typically follow the same two-stage design. First, candidate generation selects a small subset from millions of items using user and context characteristics. Then, ranking sorts the candidates using additional characteristics and goals (freshness, diversity). The YouTube paper clearly describes this model and shows why it scales okay [1, pp. 191–195]. Traditional matrix factorisation uses a simple scalar product between user and item vectors. Neural collaborative filtering (NCF) replaces this with a small neural network that can learn a better interaction function based on data and often outperforms classical models on standard datasets [2, pp. 173–175]. In practice, teams combine trained models with business rules to avoid trivial popularity cycles. If the task requires exact words (legal codes, product identifiers), the BM25 method remains the best. If the task concerns meaning and paraphrases, dense search helps.

A safe hybrid option is to use BM25 for recall, dense methods for semantic coverage, and ColBERT as a fine-grained final step [6, pp. 352–360; 4, pp. 39–43].

From a modelling perspective, popular candidate generation architectures train separate user and item encoders and optimise them for maximum internal product search. This enables fast approximate search on embedded items during service and scales to very large catalogues [1, pp. 191–195]. At the ranking stage, more rich signals (context, relevance, calibration goals) are

used and training is performed with goals aligned with business goals, such as number of clicks or viewing time [1, pp. 191–197]. Within the NCF family, the generalized matrix factorization path captures linear interactions, the MLP path captures nonlinear interactions, and the combined NeuMF architecture combines their strengths for top-K recommendations [2, pp. 173–182].

The choice of index determines the practical parameters. In FAISS, we choose how many clusters to explore and how much compression to apply; more exploration improves recall but takes time [3, pp. 1–3]. Systems often maintain a compressed 'cold' index for the entire catalogue and a small 'hot' index for popular items. Training dual encoders benefits from hard negatives – very similar but incorrect fragments – that force the model to pay attention to details.

Offline metrics such as Precision@k, Recall@k, and nDCG are useful, but we also care about coverage and useful novelty in recommendations to avoid a narrow 'filter bubble.' Online A/B tests check whether offline benefits translate to real users [1, pp. 195–197]. For search, it is useful to segment queries by type and language to ensure that improvements are broad and not limited to a small subset of simple cases [5, pp. 2–3].

It is also important to conduct measurements using a reliable methodology. Relevance ratings should be collected for a representative sample of queries; scoring supports metrics such as nDCG, where higher positions in the ranking are discounted less heavily [5, pp. 2–3]. Pooling strategies help to collect ratings efficiently, but queries that have been heavily optimised during development should be avoided. Short-term click metrics are useful for recommendations, but teams typically validate improvements through controlled experiments and track long-term engagement and diversity [1, pp. 195–197].

For clarity, we will define the terms used in the text. A query is what the user enters; a document is what the system returns. Relevance means how useful a document is for the query. Embedding is a list of numbers that reflects the value of the text. An index is a data structure that helps to quickly find elements. ANN (approximate nearest neighbour) is a family of methods that quickly finds vectors close to a given vector. Candidate generation is the first step, which finds a small set of potentially relevant elements. Ranking is the second step, which orders them [5, pp. 1–4; 3, pp. 1–3].

A simple search system can be created in a few steps. (1) Prepare documents and create a BM25 index to obtain a robust base without any training [6, pp. 333–340]. (2) Add a double encoder and compute vector embeddings for snippets; store them in FAISS with IVF-PQ and start with a moderate number of clusters, adjusting them based on latency [3, pp. 1–3]. (3) Train using the extracted hard negatives so that the model learns to distinguish subtle differences. (4) Add ColBERT re-ranker for a few hundred top candidates to gain accuracy [4, pp. 39–41].

A simple recommendation system works in similar steps. (1) Collect user interactions with products and basic product metadata (category, price). (2) Create embeddings for each product and build a fast ANN index (e.g., IVF-PQ in FAISS) to quickly find similar products [3, pp. 1–3]. (3) To generate candidates, combine the search for similar items with a simple popularity check to ensure diversity. (4) Train a ranking model (e.g., NCF) based on past clicks and add constraints for freshness and diversity [2, pp. 173–175; 1, pp. 191–195]. (5) Evaluate offline on unused data and validate with small A/B tests.

Common mistakes include: focusing only on accuracy (which can reduce catalogue coverage), omitting hard negatives during training (which leads to weak dense models), and ignoring cost (which leads to overly complex pipelines). Use compression, caching, and clear service-level objectives. Finally, don't forget to update your data regularly to avoid outdated recommendations; plan for re-indexing and model updates [3, pp. 1–3; 1, pp. 195–197].

Many real-world systems are multilingual and multimodal. In search, the simplest baseline is to index each language separately and then apply the same BM25 + dense + ColBERT pipeline to each language. If cross-language search is needed, multilingual encoders help but require careful evaluation. In recommendations, metadata and embedding from different modalities improve the cold start problem and can be mixed at the ranking stage [5, pp. 1–4; 1, pp. 191–195].

The practical checklist is short. Stick to a clear baseline (BM25 for search; simple popularity plus item similarity for recommendations). Add one improvement at a time and measure carefully. Use a small, stable set of metrics: nDCG@k and latency for search; CTR, reach, and diversity for recommendations. Control costs with compression (PQ), caching, and smart packet sizes; first choose FAISS settings that fit your latency budget, then gradually increase the level of reproduction [3, pp. 1–3]. Prefer a two-stage design so that a fast first pass protects tail latency and a second pass focuses on quality [1, pp. 191–195]. Document assumptions and data filters (language, time frame, security rules) to ensure that results are reproducible [5, pp. 1–4]. Finally, record data update and retraining schedules (e.g., daily reindexing, weekly retraining) to keep the system from becoming obsolete [6, pp. 352–360; 3, pp. 1–3; 1, pp. 195–197].

Conclusion. Modern IR and RS are based on a small set of ideas that work okay together: fast first pass, stronger second pass, and efficient vector index. BM25 is a reliable start; dense search and ColBERT add semantic power; two-stage recommendation systems scale in production. With careful evaluation and simple operating rules, these systems can be implemented by small teams and student projects [5, pp. 1–4; 6, pp. 333–340; 1, pp. 191–195]. In practice, the winning recipe is a hybrid: BM25 for retrieval, a dense search engine for semantic matches, and a precise re-ranker for the best results, as well as candidate generation and pair ranking on the recommendation side [4, pp. 39–41; 1, pp. 191–195].

REFERENCES

1. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations. In: RecSys ’16. ACM, 2016. PP. 191–198.

2. He X., Liao L., Zhang H., Nie L., Hu X., Chua T.-S. Neural Collaborative Filtering. In: WWW ’17. ACM, 2017. PP. 173–182.

3. Johnson J., Douze M., Jégou H. Billion-Scale Similarity Search with GPUs. arXiv:1702.08734, 2017. 25 p.

4. Khattab O., Zaharia M. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In: SIGIR 2020. PP. 39–48.

5. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 482 p. (Online edition: Stanford NLP).

6. Robertson S., Zaragoza H. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval. 2009. Vol. 3(4). PP. 333–389.

УДК 004.94

Artemii Muzychenko, Maryna Vyshnevska
(Kyiv, Ukraine)

THE RISE OF QUANTUM COMPUTING: OPPORTUNITIES AND RISKS

This article provides a comprehensive analysis of the phenomenon of quantum computing as a technology capable of fundamentally changing science, the economy, and the global security system. It examines the fundamental principles of quantum computers, such as superposition and entanglement, and analyzes their potential capabilities in the fields of medicine, materials science, and artificial intelligence.

Keywords: *quantum computing, qubit, superposition, quantum advantage, cryptography, Shor's algorithm, post-quantum cryptography, cybersecurity.*

У статті проводиться комплексний аналіз феномену квантових обчислень як технології, що здатна кардинально змінити науку, економіку та систему глобальної безпеки. Розглядаються фундаментальні принципи роботи квантових комп'ютерів, такі як суперпозиція та заплутаність, а також аналізуються їхні потенційні можливості у сферах медицини, матеріалознавства та штучного інтелекту.