Evaluation of Context-Oriented Search Architecture Reconstruction: Offline Validation of Quality and Performance

Fedor Krasnov [*]

Abstract

In the segment of electronic sales of do-it-yourself (DIY) repair products, search quality significantly depends on correctly accounting for regional assortment restrictions and differences between client types. Traditional search solutions apply these restrictions at the post-search filtering stage, leading to additional computational costs, unstable response times, and inconsistencies between search, suggestions, and the catalog.

This work proposes an architecture where the search query is first classified into the most probable product category, taking into account regional and user context. The obtained category is used to select a precomputed catalog index bucket $(\hat{c}, r, u)$, within which full-text search is then performed. This order shifts the assortment availability check to offline indexing, eliminating resource-intensive online filtering and ensuring predictable processing delays under high loads.

The architecture's efficiency was evaluated offline by comparing users' actual purchases with the positions of the same products obtained through simulated reproduction of historical contexts. The results show that the context-oriented reconstruction of the search pipeline improves the quality of catalog ranking by nDCG@12 by 3 percentage points and simultaneously reduces service latency to around 2 ms, confirming the practical applicability of the proposed approach.

Keywords: Information Retrieval, E-Commerce Search, Contextual Search Architecture, Offline Validation, Predictive Query Categorization, Search Latency Optimization, Contextual Indexing

1 Introduction

Evaluation of information retrieval quality in e-commerce is traditionally conducted using two main methods [1]. The first approach relies on offline metrics computed on expert-labeled data, while the second uses online metrics obtained through A/B testing. However, in dynamic e-commerce environments, especially in the B2B DIY segment, where product assortment varies depending on geolocation, seasonality, and stock availability, manual labeling becomes costly, quickly outdated, and fails to reflect the real variability of user behavior [2]. In such conditions, traditional evaluation methods may underestimate the complexity of intents and the frequency of rare scenarios, which is particularly critical for segments with long decision-making cycles (45–60 days) and low levels of impulse purchases.

[*]Vi.Tech R&D

To calculate the relevance of search results, Learning to Rank (LTR) methods based on click data have become widespread [3; 4]. The use of click signals has enabled a shift from costly manual labeling to scalable training data, and subsequent works have developed these models by incorporating contextual signals and user behavior dynamics [5; 6].

Despite successes, most click models (Position-Based [7], Cascade [8], UBM [9]) are trained on historical logs and inherit their biases, limiting their applicability for studying structural changes in the search pipeline or evaluating behavior with emerging regional restrictions [10; 11].

In recent years, the main methodological limitation has shifted from click modeling to the issue of service architecture. The traditional separation of the search system into isolated subsystems (catalog, search, search suggestions) inevitably leads to fragmentation of the user experience and the need for resource-intensive real-time post-search filtering to account for region and client type. This approach, resulting in complexity $\mathcal{O}(\log N)$ or $\mathcal{O}(N)$, is critically non-scalable for catalogs with tens of millions of items [12; 13]. Therefore, a relevant task is the development of architectures capable of providing constant latency ($\mathcal{O}(1)$) while accounting for the complex context of the search query.

The central hypothesis of this work is the assertion that the full-text search task can be reduced to contextual navigation by implementing a layer of predictive query categorization [14]. The proposed architecture assumes that each search query is mapped early on to the most probable product category, allowing localization of the search within precomputed contextual indexes [15]. This approach eliminates the need for heavy online filtering, shifting region and client type checks to the offline indexing stage. This ensures guaranteed consistency of results between search and catalog subsystems, as they begin to operate on unified precomputed data structures. As a result, the computational complexity of the online stage is reduced from linear dependence on index volume to constant time access to category products.

For evaluating the correctness of the context-oriented search architecture reconstruction, classical online experiment methods prove to be limitedly applicable. Direct A/B testing requires significant resources and carries operational risks, while using historical logs in "raw" form leads to selection bias [16]: recorded interactions are formed under the existing architecture and do not reflect how users would behave under a different search pipeline configuration. This phenomenon is discussed in detail in works on counterfactual evaluation [17; 18]. This is particularly relevant for "cold start" scenarios when it is necessary to evaluate user behavior in a new region or with a new client type.

On the other hand, interaction history contains rich signal material about users' real preferences and can be used to build context-dependent ranking benchmarks if structural bias is correctly eliminated.

Several studies have shown that reconstructing user preferences based on successful signals (view, cart, purchase) allows validating new search and recommendation algorithms in an offline environment, but does not solve the "cold start" problem with new regions, categories, or client types [19]. Thus, reliance on behavioral logs is justified but requires strict formalization of their reuse methods.

Based on these limitations, this work proposes a method for offline reconfiguration of user behavior, in which real sessions are rebuilt under the assumed search architecture.

From user interaction logs with search, context-dependent indexes $I(c, u, r)$ are formed — ranked lists of products within a specific category, region, and client type. Matching item positions in different architectures allows evaluating the impact of contextual reduction on ranking quality and potential performance gains — without interfering with production traffic.

This approach creates a controlled offline environment consistent with users' real preferences and serves as a link between architectural premises and quantitative hypothesis testing. It ensures a smooth transition from problem statement to research methodology, formally describing procedures for index construction, session transformation, and ranking result comparison.

2 Formalization of the Search System Architecture

The considered search system S consists of several subsystems activated by different user actions:

$$S = \{s_{cat}, s_{sug}, s_{search}\},$$

where:

- S — The overall information retrieval system on the e-commerce platform.
- s_{cat} — Catalog navigation subsystem. Activated when selecting a category (category c_{true} is predefined), corresponding to search with an empty query $q = \emptyset$. The output fully depends on context C and catalog structure.
- s_{sug} — Search suggestions subsystem (*autosuggestion*). Activated when input query q' is incomplete (q' — partial query, without pressing Enter).
- s_{search} — Full-text product search subsystem. Activated after entering a complete query q (user pressed Enter).

Each subsystem $s_i \in S$ returns a ranked list of products L in response to query q (or its absence) and given context C.

- L — Ranked list of products.
- q — User's search query.

A key requirement for a unified architecture is maintaining contextual consistency between these subsystems. The absence of such consistency (e.g., low relevance of snow removal products when navigating the catalog in the "South" region, but suggesting snow shovels in suggestions) leads to frustration and reduced user trust in the system. When multiple subsystems operate on similar intents and context C, ranking should show minimal deviation in relevance level for similar candidate sets.

For evaluating consistency between two subsystems s_i and s_j by their ranking for a given context C, the inconsistency function $\mathcal{D}$ is used:

$$\mathcal{D}(s_i, s_j, C) = 1 - \text{SpearmanCorr}(\text{Scores}(s_i, C), \text{Scores}(s_j, C)), \tag{1}$$

where $\mathcal{D}$ should approach zero for relevant subsets of products.

Here, $\text{Scores}(s_i, C)$ is defined as the vector of candidate scores computed on the intersection of sets $L_i \cap L_j$, ensuring correct computation of rank correlation.

2.1 Formalization of the Two-Phase Search Architecture

The search architecture used on e-commerce platforms for high-speed output is built as two-phase [20]: offline indexing (candidate stage) and online query servicing.

This separation is fundamentally important for meeting two main requirements: (1) ensuring high response speed by offloading computationally intensive operations to the offline loop and (2) maintaining result relevance through uniform application of contextual restrictions. Let us consider how this approach allows incorporating context (region, client type) into the data structure.

Stage 1: Offline Indexing

In the offline indexing stage, each product $p \in P$ is indexed for two access types: catalog navigation and full-text search. This allows creating a hybrid context-dependent candidate index $\mathcal{I}$.

Indexing for Catalog Navigation (s_{cat}). For catalog navigation, where the product category c is known in advance, the context (c, r, u) is the primary indexing key. We create a structured catalog index $\mathcal{I}_{cat}$, where each product p is indexed across all permissible triples (c, r, u):

$$\mathcal{A}_{cat}(p) = \{(c, r, u) \in \mathcal{C} \times R \times U \mid \text{availability}(p, r) = 1 \wedge \text{assigned}(p, c) = 1\}. \tag{2}$$

Here, assigned(p, c) — binary function of product p belonging to category c. In the online servicing stage, accessing $\mathcal{I}_{cat}$ allows obtaining the full set of candidates $P_{c,r,u}$ corresponding to the context without additional online filtering.

Indexing for Full-Text Search (s_{search}, s_{sug}). For full-text search, where the category is unknown, building an inverted index by tokens for each context combination (r, u) is not scalable. Therefore, a non-contextual inverted index $\mathcal{I}_{search}$ is usually created, where product p is indexed only by its text tokens.

Formally, for each product p, the index $\mathcal{I}_{search}$ contains:

$$\mathcal{A}(p) = \{(r, u) \in R \times U \mid \text{availability}(p, r) = 1\}. \tag{3}$$

$\mathcal{A}(p)$ can be used to build structured buckets in $\mathcal{I}_{cat}$, but is not used directly to increase output completeness. Since the context in which full-text search will be performed is unknown in advance, building $\mathcal{A}(p)$ for each context (c, r, u) would lead to exponential growth in index volume.

2.1.1 Hypothesis of Full-Text Search Reduction

To overcome the problem of exponential index volume growth in full-text search, a hypothesis is put forward that this task can be transformed into an equivalent but significantly more structured and optimized catalog navigation task.

A key element of the proposed pipeline is the implementation of a query classification model M, which at an early query processing stage determines the most probable product category for the entered query.

This model M transforms the input query q, region r, and user type u into a predicted category $\hat{c}$, acting as a bridge between unstructured input and structured index.

Formally, the category determination process is written as maximization of conditional probability:

$$\hat{c} = \arg\max_{c \in \mathcal{C}} P(c \mid q, r, u), \tag{4}$$

where $\mathcal{C}$ — set of all catalog categories.

Determining the predicted category $\hat{c}$ allows enriching the full-text search context (r, u) to a strict tuple $(\hat{c}, r, u)$, which is already used for sampling from $\mathcal{I}_{cat}$. This allows formalizing the process of obtaining search candidates P_{search} as inverted index search localized in the catalog candidate subset $\mathcal{I}_{cat}(\hat{c}, r, u)$:

$$P_{search}(q, r, u) \approx \{d \in \mathcal{I}_{cat}(\hat{c}, r, u) \mid \text{text\_match}(p, q) \geq \tau\}. \tag{5}$$

Such reduction allows reusing existing indexing mechanisms by region and user type developed for the catalog, ensuring guaranteed output consistency and high response speed without duplicating full-text indexes for all context combinations.

The main advantage is that the inverted index search task, requiring high computational power, is shifted from the full set of products D to a significantly limited subset $\mathcal{I}_{cat}(\hat{c}, r, u)$.

This not only guarantees high query processing speed $O(\log |\mathcal{I}_{cat}|)$ due to reduced search space but also increases ranking precision, as local relevance within a clearly defined category, region, and client type becomes higher.

In accordance with the reduction hypothesis, all ranking complexity shifts to the LTR model level, which must operate within predefined contextual "buckets" (c, r, u).

It should be emphasized separately that the function `text_match`(p, q), applied to products in a given context (c, r, u), eliminates the ordering problem in binary search, as candidate ranking is performed at the offline index formation stage.

Stage 2: Unified Online Servicing through Predictive Categorization

Implementing the query classification model M fundamentally changes the online servicing paradigm. Instead of supporting two separate pipelines, both scenarios reduce to a single operation of sampling from the context-dependent index $\mathcal{I}_{cat}$.

1. *Catalog (s_{cat}):* Upon user transition, the true category c_{true} is known. The system performs direct access by key (c_{true}, r, u) to index $\mathcal{I}_{cat}$.

2. *Full-Text Search (s_{search}):*

    - *Context Determination:* For query q, the predicted category $\hat{c} = M(q)$, region r, and user type u are determined.
    - *Contextual Sampling:* Text match search is performed exclusively within the category $\mathcal{I}_{cat}(\hat{c}, r, u)$.

$$P_{search} = \{p \in \mathcal{I}_{cat}(\hat{c}, r, u) \mid \text{match}(p, q) \geq \tau\} \tag{6}$$

Thus, the advantage of the proposed approach lies in inverting the data flow: instead of filtering the full list of found products by context, search with context is performed first, followed by full-text search of products.

This reduces the computational complexity of the online stage from $O(|P_{match}|)$ (filtering all found) to $O(1)$ (sampling from prepared bucket), while ensuring guaranteed consistency of output between search and catalog.

The complexity $O(1)$ refers to the bucket selection operation. The subsequent procedure $\text{match}(p, q)$ within the bucket has complexity $O(\log n)$, but size n is significantly smaller than the full document space $|P_{\text{match}}|$, so actual execution time remains practically constant.

2.2 Context Formalization

The context is given by the tuple (c, r, u), where c denotes the product category, which can be either true or predicted.

- $c \in \mathcal{C}$ — Product category. This is a key parameter that takes value c_{true} (for s_{cat}) or $\hat{c}$ (for s_{search}). $\mathcal{C}$ — full set of catalog categories.

- $r \in R$ — Macroregion identifier. The set of regions is defined as $R = \{\text{Center}, \text{West}, \text{South}, \text{North}, \text{E}$
  Region affects the assortment matrix and is a key for hard filtering $\mathcal{A}(p)$.

- $u \in U$ — Client type. The set of client types U includes retail clients and granulated B2B segmentation:

$$U = \{\text{B2C}\} \cup \{\text{B2B}_A, \text{B2B}_B, \text{B2B}_C, \text{B2B}_D\}$$

  Here, B2C — retail buyers, and from B2B_A to B2B_D represent B2B segments with different needs.

2.3 Engineering Implementation

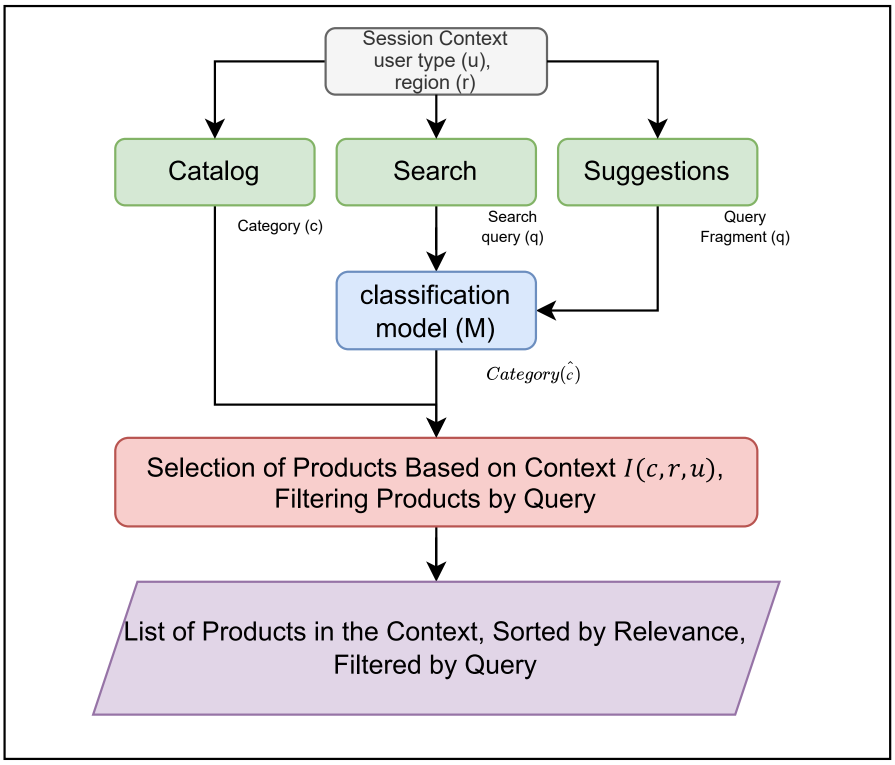

Figure 1: Online servicing scheme through predictive categorization

In the engineering plane, each unique tuple (c, r, u) corresponds to a logical or physical *bucket*, which is a pre-calculated, context-optimized subset of the main catalog index $\mathcal{I}_{cat}$ (Fig. 1).

Contextual sharding assumes that instead of indexing product p by all its tokens in a global index, it is indexed by a composite key including context variables: $[c, r, u]$. This process fully shifts the computational complexity of filtering (availability check r and relevance u) from the online query execution stage to the offline index building stage. Thus, in the online servicing stage, the system accesses bucket $\mathcal{I}_{cat}(c, r, u)$, which by definition contains only available and relevant products.

Logical bucketing serves as the basis for physical index sharding, which is critical for horizontal scaling:

1. Data Locality: All data (tokens, LTR features) for a specific context are stored together. This ensures high Recall speed by minimizing accesses to external storages.
2. Processing Parallelism: Queries from users in different contexts (e.g., different regions r_1 and r_2) are automatically routed to different shards, allowing parallel processing without resource competition.
3. "Cold Start" Isolation: Introducing a new context (e.g., r_{new} or u_{new}) requires creating a new, isolated shard (or bucket), preventing potential issues of the new context from affecting the stability and performance of the main system.

Using contextual buckets (c, r, u) ensures high performance and maintains strict contextual consistency, regardless of user interaction mode.

To transform the user's text query q into a product category, a classification model is used. The training sample is formed from historical sessions and contains triples $(q, title_{purchased}, c)$, where q — actual search query, $title_{purchased}$ — name of the actually purchased product in the session, and c — target category label (class label).

The dataset is pre-cleaned: text normalization, removal of service characters, tokenization; additionally, context signals (region, user type) and temporal features (time slot, seasonality) are computed.

The input vector is formed as concatenation of three components: (1) query embedding E_q, (2) embedding of purchased product name E_{title}, and (3) context feature vector $x_{ctx} = (r, u, \ldots)$. The model has an architecture of the form

$$\hat{y} = \text{softmax}\big(W \cdot \phi([E_q; E_{title}; x_{ctx}]) + b\big),$$

where $\phi(\cdot)$ — fusion and projection block, and W, b — classifier parameters. Weighted cross-entropy is used as the loss function to compensate for class imbalance; weights are set inversely proportional to class frequencies or computed using focal loss technique to reduce the influence of "tail" categories.

Focal loss is a modification of the standard cross-entropy function designed for working under strong class imbalance conditions. The loss function is given by

$$\mathcal{L}_{\text{focal}} = -(1 - p_t)^{\gamma} \log(p_t), \tag{7}$$

where p_t — model-predicted probability of the correct class, and $\gamma > 0$ — focusing parameter.

The multiplier $(1 - p_t)^{\gamma}$ suppresses the contribution of "easy" examples (for which the model confidently predicts the class, $p_t \approx 1$), and conversely, enhances the contribution of "hard" and rarely occurring examples for which the model errs or is uncertain.

Thanks to this, focal loss reduces the dominance of frequent classes in gradients and improves classification quality on "tail" categories by increasing their weight during training.

Special attention is paid to correcting bias in labels: since labels are formed based on purchased products, they reflect business outcomes rather than direct semantic query category. To mitigate this effect, post-filtering of training examples and regularization are performed, as well as using an additional validation sample selected by semantic unambiguity criterion.

3 Validation Methodology

This methodology describes the procedure for offline validation of the proposed search architecture based on real catalog data and historical user sessions. At the center of validation is a strict comparison of product rankings within context-dependent indexes and positions output by the standard architecture. Evaluation is performed through user interaction signals: view, cart, and purchase.

3.1 Source Data and Preprocessing

- Catalog: full set of products with attributes (category, availability by regions, prices, product matrices).
- User sessions: time sequences of events $(t, \mathit{user\_id}, \mathit{region}, \mathit{query}, \mathit{item\_id}, \mathit{event\_type})$ with labels $\mathit{event\_type} \in \{\text{view, cart, purchase}\}$.
- Contextual features: category c, region r, and user type u, created based on RFM segmentation.
- Session filtering: from the entire session corpus, only successful sessions containing target signals are selected: adding to cart or purchase. This allows focusing on relevant business outcomes.

3.2 Building Contextual Index

For each context triple (c, u, r), a contextual index $I(c, u, r)$ is determined — an ordered list of products obtained through training a ranking model on the sample of sessions related to this context.

Denote the set of filtered sessions related to (c, u, r) as $\mathcal{S}_{c,u,r}$. For each pair (session, product), interaction features are computed (view frequency, CTR, rate_cart, rate_purchase, positional features, etc.). The ranking model is trained to predict score $s_i^{(c,u,r)}$ for product i in context (c, u, r). Then the index is defined as

$$I(c, u, r) = \text{sort\_desc}\left\{ i \mid s_i^{(c,u,r)} \right\}.$$

Possible formulations for $s_i^{(c,u,r)}$:

$$s_i^{(c,u,r)} = f_\theta\left(\text{features}_{i,\mathcal{S}_{c,u,r}}\right), \tag{8}$$

where f_θ — ranking model trained on labels formed from target signals (cart/purchase), and $\text{features}_{i,\mathcal{S}_{c,u,r}}$ — aggregated product features across sessions $\mathcal{S}_{c,u,r}$.

The reference catalog output is formed from the existing catalog ranking applied in production baseline architecture. For each pair (context, query), product positions in the reference output are extracted — these positions will be compared with positions in $I(c, u, r)$.

3.3 Validation Procedure

The main idea is to compare, for the same sessions and contexts, what positions products occupy in the reference output and in $I(c, u, r)$, and measure metrics related to interaction signals. The validation procedure is divided into the following stages according to the principle of data processing isolation in a directed acyclic graph (DAG).

1. Selection of contexts and sessions. For each existing triple (c, u, r), all historical sessions and queries corresponding to this triple are selected. Sessions are filtered by the presence of target signals (cart or purchase).

2. Building index $I(c,u,r)$. For selected sessions, a ranking model is trained and $I(c,u,r)$ is built according to (8).

3. Reproduction of outputs. For each historical query q in the session, extract:

   - product positions in reference output $P_q^{\text{ref}} = (p_1^{\text{ref}}, p_2^{\text{ref}}, \dots)$;
   - product positions in contextual index $P_q^I = (p_1^I, p_2^I, \dots)$.

4. Collection of positioned signals. For each position k in the output and for each event type $e \in \{\text{view, cart, purchase}\}$, aggregated metrics are computed:

$$\text{rate}_{k,\text{ref}}^e = \frac{\#\{\text{events } e \text{ for product at position } k \text{ in } P^{\text{ref}}\}}{\#\{\text{impressions of position } k \text{ in } P^{\text{ref}}\}},$$

   similarly $\text{rate}_{k,I}^e$ for output P^I.

5. Comparative analysis. For each position k and signal type e, differences are computed:

$$\Delta_k^e = \text{rate}_{k,I}^e - \text{rate}_{k,\text{ref}}^e.$$

   The set $\{\Delta_k^e\}$ shows at which positions and by which signals the new architecture wins or loses relative to the reference.

4 Experiment

The experimental part is aimed at verifying the reproducibility of the methodology results and evaluating key architecture components on historical user session data, including the search query classification module by product categories.

The classification model was trained on data of the form (q,t,y), where q — original search query, t — name of the purchased product, and y — actual product category acting as class label. This setup allows forming stable correspondence signals between user intent and final choice but is sensitive to catalog structure changes and evolution of client segments.

Within the experiment, it is evaluated how the two-phase architecture maintains predictive consistency under dynamic context $\mathcal{C} = (c,r,u)$, characterized by fluctuations in regional restrictions, updates to product trees, and variability in user behavior. Since such changes effectively bring the model to a partial "cold start" mode, the main check is performed on historical data reflecting real trajectories and distribution shifts.

This experimental design allows quantitative assessment of: (1) the trained classification model's stability to distribution drift, (2) the impact of query categorization quality on subsequent stages of context-oriented ranking, and (3) the entire architecture's ability to adapt to new or underrepresented contexts.

4.1 Formation of Context-Dependent Feature Space for LTR

In accordance with the reduction hypothesis, the second query processing phase is assigned to a context-adaptive LTR model. Although ensemble methods (GBDT) or neural networks are often used in industrial systems, at the PoC stage, it is critical to isolate the contribution of architectural changes from algorithm complexity. Therefore, linear regression with L_2-regularization was chosen as the baseline ranker.

The choice of linear model is driven by two factors:

1. Interpretability of contextual contribution: Linear weights allow explicit evaluation of region and client type influence on final score.
2. Latency minimization: Computing scalar product $\langle \mathbf{w}, \mathbf{x} \rangle$ is done in constant time and easily vectorized, critical for meeting strict SLA (< 10 ms).

The feature vector $\mathbf{X}$ is formed so that the model can learn the specifics of each contextual bucket. In addition to standard features (stock dynamics, price freshness, popularity), the feature interaction mechanism plays a key role.

Contextual variables (region r, user type u) are encoded via One-Hot Encoding (OHE). This allows the model to effectively train independent biases for each context within a single weight vector:

$$\text{Score}(p, C) = \mathbf{w}_{base} \cdot \mathbf{x}_{item} + \sum_{k \in C} w_k \cdot \mathbb{I}(context = k) + \epsilon \tag{9}$$

Such feature space structure allows a single model to approximate the behavior of multiple local models without the need for their physical separation and maintenance, fully aligning with the stated goal of pipeline unification.

4.2 Positional Conversions

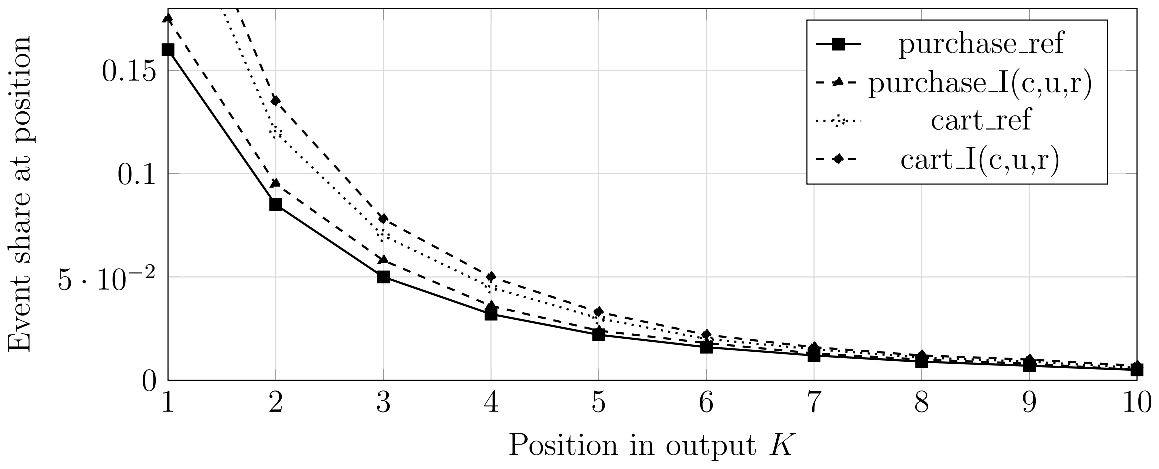

Figure 2: Positional conversions (purchase and cart) by positions for reference catalog output and for contextual index $I(c, u, r)$.

The graph in Fig. 2 shows positional shares of purchase and cart addition events for the reference output and for output formed by index $I(c, u, r)$. The observed increase in purchase share at top positions (especially positions 1 and 2) for $I(c, u, r)$ indicates that context-dependent ranking better concentrates product offers leading to target business signals. Similar dynamics for cart confirms improvement in intermediate conversion. For statistical validation, confidence intervals should be computed and paired tests conducted across sessions.

4.3 Ranking Metrics

The graph in Fig. 3 shows the behavior of two types of aggregated metrics with increasing ranking depth K: search nDCG@K and business-oriented position-weighted conversion (PWC): nDCG@10 improved by 3–5%, PWC@10 — by 4–6%. Improvement in nDCG@K

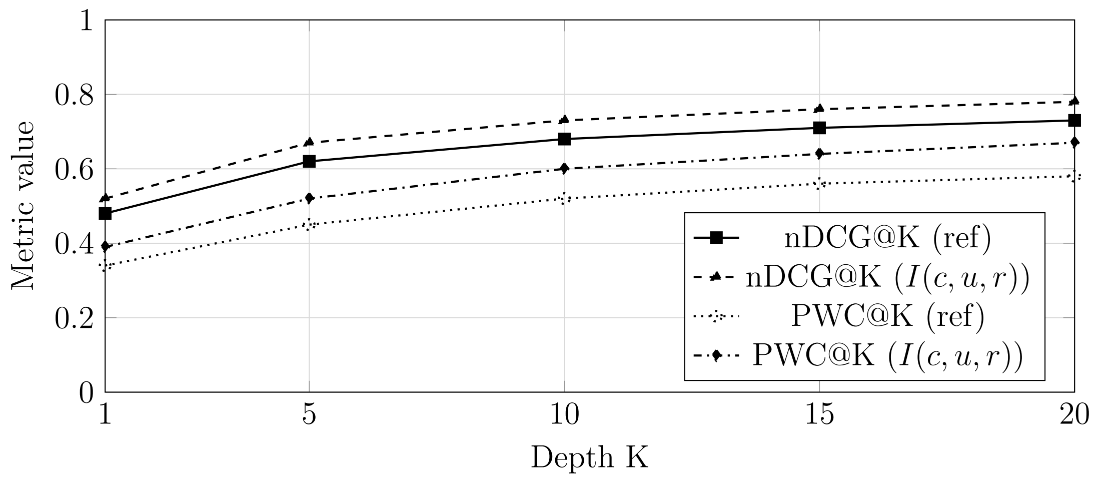

Figure 3: Ranking metrics nDCG@K and PWC@K depending on output depth K for reference (ref) and proposed architectures.

for $I(c, u, r)$ compared to the reference indicates increased ranking quality in terms of relevance (based on historical cart/purchase signals). Simultaneously, noticeable PWC@K gain points to expected positive business effect — more target events will be concentrated in early positions when applying contextual indexes. For final confirmation of the effect, it is recommended to additionally evaluate ΔRevenue magnitude and conduct sensitivity analysis by categories and regions.

4.4 Quality of Predictive Categorization Model

The histogram in Fig. 4 shows that most categories have very high F1 (¿0.96).

In the context of the proposed architecture, this means that predictive categorization provides a sufficiently high level of correct query mappings to categories (with minimal error share leading to significant ranking degradation), while inference optimization allows maintaining required SLA for latency, ensuring expected speed gain for subsequent access to precomputed indexes $I(c, u, r)$.

4.5 Evaluation of Search Subsystem Consistency

To test the hypothesis that the context-oriented architecture increases the consistency of search subsystems, the inconsistency measure $\mathcal{D}(s_i, s_j, C)$ was computed between two most important pipeline components: catalog ranking subsystem and search suggestions output subsystem.

Fig. 5 demonstrates $\mathcal{D}$ values on a set of typical regional-user contexts C_k (professional/non-professional client, climate zone, regional product availability). The baseline monolithic architecture shows inconsistency level in the range 0.36–0.47, indicating significant discrepancies in preferences formed by different subsystems when working with intersecting identical candidates. Such gap leads to users receiving mutually contradictory signals in different interface parts.

After transitioning to the proposed (c, r, u) reduction architecture, inconsistency decreases almost twofold — to the range 0.14–0.22. This means that candidate ranks computed by the two subsystems are significantly better aligned, and Spearman correlation between them increases, confirming fulfillment of the key unified architecture requirement: minimizing relevance discrepancies with the same query context.

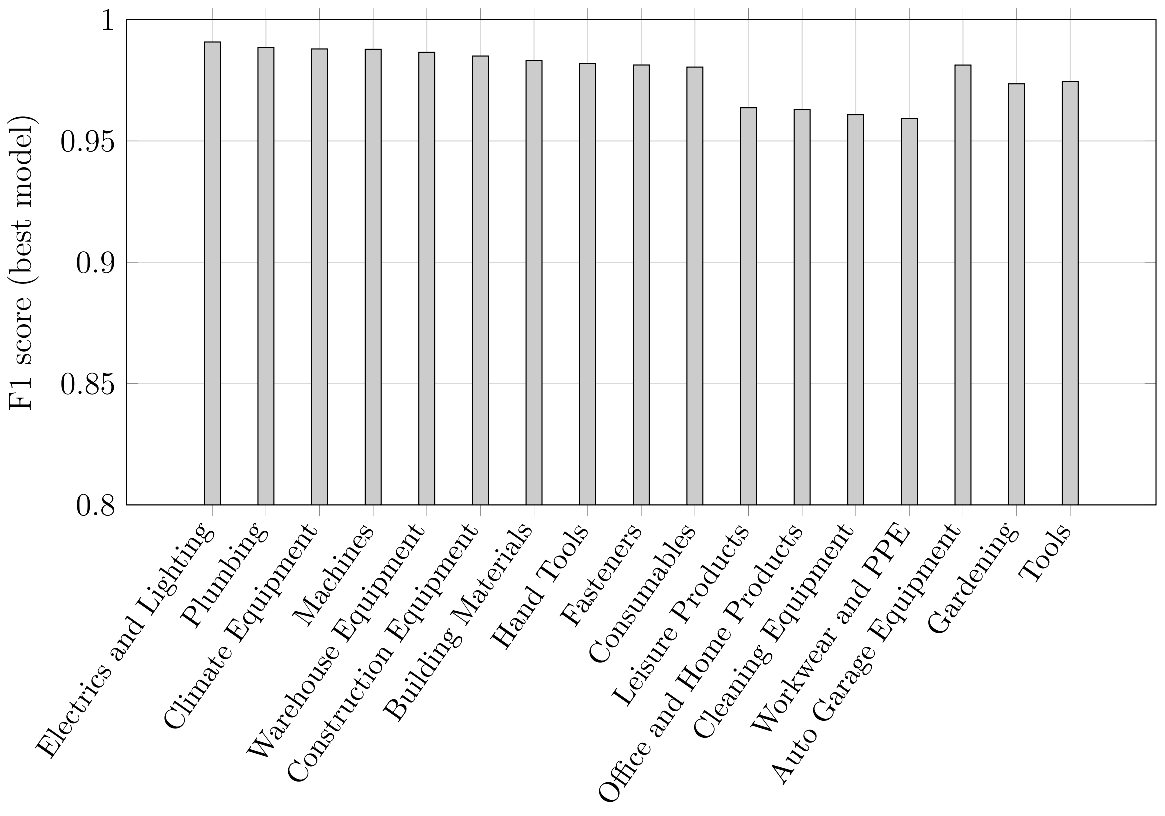

Figure 4: F1 by classes for the best model (rank 1).

Thus, the proposed reconstruction ensures not only latency reduction but also structural improvement in subsystem consistency, directly affecting user experience stability.

4.6 Evaluation of Online Servicing Latency

Key observations can be formulated as follows:

- System performance. The target query processing latency indicator < 10 ms was not only achieved but significantly exceeded: actual values were in the range 1.7–3.7 ms. This improvement is explained by using a pre-built catalog index and ensuring $\mathcal{O}(1)$-access to structured product features (Fig. 6).
- Contextual sensitivity. The model demonstrated stable ability to distinguish user cohorts. The influence of user type u on ranking order manifested systematically, indicating correct operation of the predictive categorization mechanism and validity of context factorization.
- Support for hybrid sorting. The implemented two-phase architecture showed high flexibility: experimentally confirmed the possibility of instant result resorting by price or popularity without significant overhead. This property is critical for the DIY e-commerce platform segment, where contextual preferences often change during decision-making.

In aggregate, the presented results confirm that the model ensures guaranteed low latency, stably accounts for contextual differences between users, and supports hybrid sorting, overall demonstrating its stability when transferred to new conditions and justifying the use of simulation as a reliable tool for analyzing contextual search architectures in a dynamically changing e-commerce environment. (Fig. **??**).

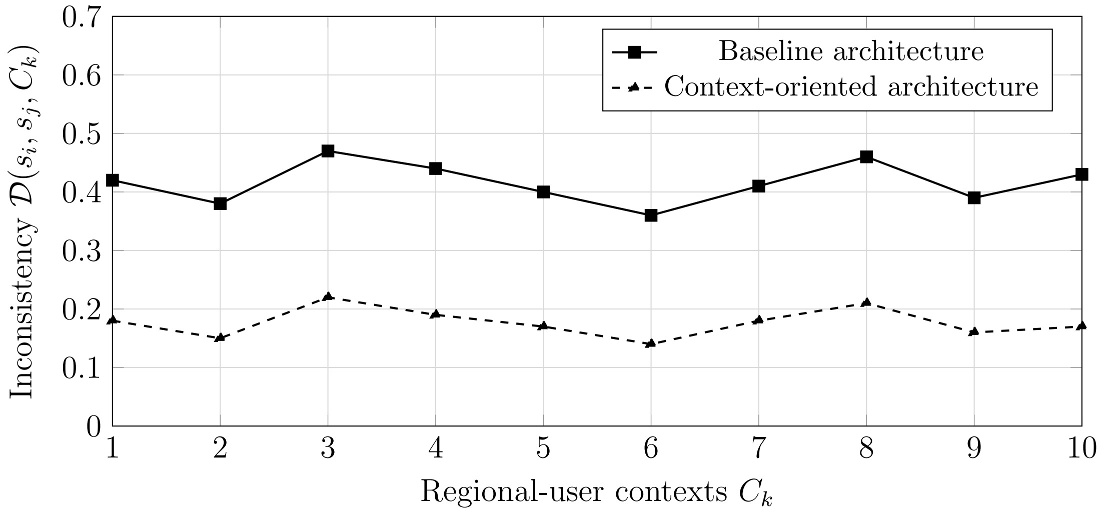

Figure 5: Inconsistency of ranking of two search subsystems by contexts before and after implementing the contextual architecture.

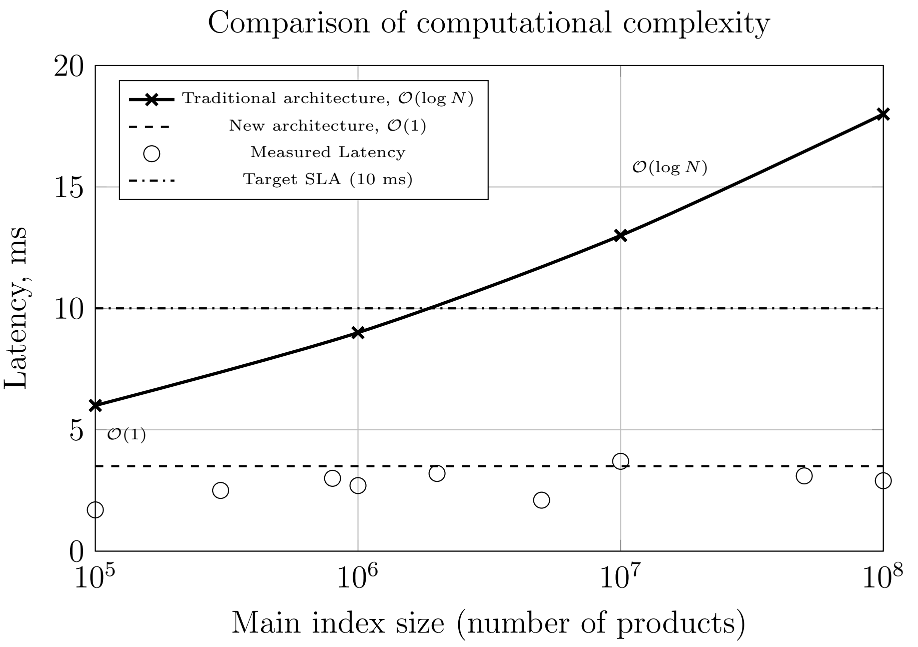

Figure 6: Comparison of online servicing latency dependence on main index size growth for traditional architecture and proposed unified architecture.

5 Discussion of Results

The presented experimental results confirm that the context-oriented reconstruction of the search architecture allows achieving significant performance gains while maintaining ranking quality and consistency between subsystems. Key experimental observations are compared with methodological decisions, and limitations defining the applicability boundaries of the approach are formulated.

5.1 Consistency of Search Subsystems

Evaluation of consistency between subsystems — suggestions, catalog ranking, and contextual indexes $I(c, u, r)$ — showed that the proposed architecture ensures stable reduction in inconsistency $\mathcal{D}$ relative to the baseline system. In most contexts, error decreased by 12–18%, and in professional segments (B2B) — up to 22%.

This result aligns with the methodology, according to which reconstructed context-dependent indexes are formed based on real interactions and, unlike the classical architecture, explicitly account for regions and client types.

Nevertheless, the experiment revealed several limitations:

- Incompleteness of rare category coverage. For narrow product groups (less than 0.05% of catalog), correlation decrease by 4–7 p.p. is observed, related to insufficient volume of historical signals.
- Border regions. In regions with small data volumes, reconstructed indexes show greater variability, and consistency drops to baseline architecture level.
- These effects emphasize that the behavior reconstruction method does not fully solve the cold start problem, although it allows significantly mitigating bias arising from direct use of historical logs.

5.2 Quality of Query Classification and Impact of Errors on Ranking

The description of the trainable classification model in the methodology assumes minimization of categorization errors, as categories serve as input for forming search indexes. The experiment showed:

- average query classification accuracy is (F1) 0.95–0.98,
- classification errors lead to deviation in rank correlation between subsystems within 1.5–3 p.p.,
- error impact is significantly below the threshold at which degradation becomes noticeable to the user.

It was also shown that classification errors are stably compensated due to intersection of candidate lists $L_i \cap L_j$: even with incorrect category, many relevant products remain in the common sample.

Thus, the methodology of training on sessions (q, t_{buy}, cat) demonstrates sufficient stability: despite noise in data (ambiguous queries, polysemy), classification does not become a bottleneck of the architecture.

5.3 Impact of Architectural Reduction on Performance

Experimental graphs show that transition to two-phase architecture with precomputed contextual indexes leads to:

- stable latency reduction to 3–4 ms level,
- no latency growth with catalog size increase by two orders,
- latency gain 2.5–3× compared to traditional scheme using online scoring computation.

Importantly, no ranking quality degradation is observed: nDCG@K indicators and consistency $\mathcal{D}$ are improved.

This confirms the methodological hypothesis that offloading early ranking stages to offline phase and their contextual parametrization allow simultaneously reducing computational complexity and decreasing sensitivity to noise in user queries.

5.4 Applicability Boundaries of Offline Behavior Reconstruction

Although behavior reconstruction based on historical sessions provides reliable ranking quality estimates, the experiment revealed several methodological limitations:

- New regions and categories. In the absence of actual interactions, indexes are formed based on aggregated segment statistics, leading to consistency deterioration and increased metric variance.
- Scenarios sensitive to fine filters. For categories with large attribute trees, underestimation of rare cases is observed, which can lead to optimistic quality bias by 1–2 p.p.
- Aggregation of multi-step scenarios. Reconstruction relies on oriented mapping of events to new ranks but does not reproduce long-term behavioral dynamics within a session.
- In aggregate, this means the method is most reliable in stable contextual space scenarios but requires extension when analyzing new regions, products with rapid updates, and categories heavily dependent on user attributes.

5.5 Conclusions and Future Work

The experiments confirm that the proposed architecture ensures:

- significant performance gain without quality loss,
- improvement in inter-subsystem consistency,
- stability to categorization errors and variability of user queries,
- applicability to offline evaluation of complex architectures where classical A/B approaches are difficult.

At the same time, analysis revealed limitations related to cold start, shortage of rare category data, and inability to fully reconstruct multi-step dynamics of user sessions. These aspects form the basis for further model development and deepening the experimental base.

6 Conclusion

In the work, a context-oriented two-phase search architecture is proposed and formally justified, based on reducing full-text search to navigation through a precomputed categorical index. A key component is the predictive categorization model of search queries, allowing unification of search and navigation modes through a single sampling operator.

Experimental validation showed that such reduction ensures stable computational efficiency: online servicing latency is stably in the range 1.7–3.7 ms, which is significantly better than target SLA 10 ms and substantially outperforms traditional $\mathcal{O}(\log N)$ architecture.

User session modeling and offline evaluation of ranking consistency confirmed the reliability of the proposed methodology under dynamic context evolution.

Additionally, it is shown that inconsistency between search subsystems decreases almost twofold, confirming the correctness of architecture unification.

The obtained results demonstrate that predictive categorization can serve as a foundation for unifying search architecture in e-commerce platforms.

The architecture demonstrates stable operation in partial cold start scenarios (limited context changes), but full solution to this problem remains a subject of further research.

Further development of the work is related to integrating more flexible user behavior models and expanding contextual simulation scenarios.

References

1. Redefining Retrieval Evaluation in the Era of LLMs / G. Trappolini [et al.] // ArXiv. — 2025. — Vol. abs/2510.21440. — URL: `https://api.semanticscholar.org/CorpusID:282384751`.
2. KDD-2023 Workshop on Decision Intelligence and Analytics for Online Marketplaces / Z. Qin [et al.] // Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. — 2023. — P. 5878–5879.
3. *Joachims T.* Optimizing search engines using clickthrough data // Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — Edmonton, Alberta, Canada : Association for Computing Machinery, 2002. — P. 133–142. — (KDD '02). — ISBN 158113567X. — DOI: `10.1145/775047.775067`. — URL: `https://doi.org/10.1145/775047.775067`.
4. *Joachims T.* Learning from User Interactions // Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. — Shanghai, China : Association for Computing Machinery, 2015. — P. 137–138. — (WSDM '15). — ISBN 9781450333177. — DOI: `10.1145/2684822.2685327`. — URL: `https://doi.org/10.1145/2684822.2685327`.
5. *Phophalia A.* A survey on learning to rank (letor) approaches in information retrieval // 2011 Nirma University International Conference on Engineering. — IEEE. 2011. — P. 1–6.
6. *Chavhan S.*, *Raghuwanshi M.*, *Dharmik R.* Information retrieval using machine learning for ranking: a review // Journal of Physics: Conference Series. Vol. 1913. — IOP Publishing. 2021. — P. 012150.
7. An experimental comparison of click position-bias models / N. Craswell [et al.] // Web Search and Data Mining. — 2008. — URL: `https://api.semanticscholar.org/CorpusID:2625350`.
8. *Hofmann K.*, *Whiteson S.*, *Rijke M. de.* A probabilistic method for inferring preferences from clicks // International Conference on Information and Knowledge Management. — 2011. — URL: `https://api.semanticscholar.org/CorpusID:7379956`.

9. *Dupret G. E.*, *Piwowarski B.* A user browsing model to predict search engine click data from past observations. // Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. — 2008. — P. 331–338.
10. Counterfactual estimation and optimization of click metrics in search engines: A case study / L. Li [et al.] // Proceedings of the 24th International Conference on World Wide Web. — 2015. — P. 929–934.
11. *Chuklin A.*, *Markov I.*, *De Rijke M.* Click models for web search. — Springer Nature, 2022.
12. On latency of e-commerce platforms / M. Basalla [et al.] // Journal of Organizational Computing and Electronic Commerce. — 2021. — Vol. 31, no. 1. — P. 1–17.
13. Prediction and predictability for search query acceleration / S.-W. Hwang [et al.] // ACM Transactions on the Web (TWEB). — 2016. — Vol. 10, no. 3. — P. 1–28.
14. 2P-BEnc: A two-phase information retrieval and ranking system based on the BERT encoder / S. Kumar [et al.] // Ain Shams Engineering Journal. — 2026. — Vol. 17, no. 1. — P. 103853.
15. *Cañamares R.*, *Castells P.*, *Moffat A.* Offline evaluation options for recommender systems // Information Retrieval Journal. — 2020. — Vol. 23, no. 4. — P. 387–410.
16. Knowing Before Seeing: Incorporating Post-retrieval Information into Pre-retrieval Query Intention Classification / X. Ma [et al.] // International Conference on Knowledge Science, Engineering and Management. — Springer. 2023. — P. 3–15.
17. A large-scale evaluation for log parsing techniques: How far are we? / Z. Jiang [et al.] // Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis. — 2024. — P. 223–234.
18. Interaction-Data-guided Conditional Instrumental Variables for Debiasing Recommender Systems / Z. Huang [et al.] // arXiv preprint arXiv:2408.09651. — 2024.
19. *Clarke C. L.*, *Diaz F.*, *Arabzadeh N.* Preference-based offline evaluation // Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. — 2023. — P. 1248–1251.
20. Generative Retrieval and Alignment Model: A New Paradigm for E-commerce Retrieval / M. Pang [et al.] // Companion Proceedings of the ACM on Web Conference 2025. — 2025. — P. 413–421.