International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Request-Level Training Paradigms for Efficient Large-Scale Recommendation Systems

Siddharth Narayanan

Abstract: Modern recommendation systems operate at a massive scale and are expected to provide increasingly personalized experiences under strict latency, storage, and computational constraints. Traditional training pipelines typically structure data at the level of individual impressions, which has historically been effective but introduces substantial redundancy in both data representation and computation. This redundancy becomes a significant bottleneck as platforms grow in complexity and volume. Request-level training paradigms offer an alternative by redefining the basic unit of learning from the individual impression to the full request or grouped interaction context. This article examines the conceptual foundations, architectural implications, and systems benefits of request-level training in large-scale recommendation systems. It argues that aligning training data structures more closely with real interaction patterns enables better computational efficiency, richer contextual modeling, and more scalable personalization. The discussion also explores the practical infrastructure changes required for adopting this paradigm and considers its broader implications for the future of machine learning systems used in retrieval and ranking.

Keywords: *Recommendation Systems, Request-Level Training, Ranking Systems, Machine Learning Infrastructure, Personalization*

Introduction

The scale of recommendation systems has become the backbone of contemporary digital experiences, making the content discovery and personalization experience visible to billions of users every day. These systems dictate which videos are shown in the streaming feeds, which products are shown in the search engines of e-commerce, which articles are shown in the news aggregators, and which ads are shown on the web. The technical requirements are high: recommendation systems have to process a vast amount of user behavior data, represent highly diverse preference models under a variety of conditions, and produce useful predictions in milliseconds to ensure high latency constraints [1]. With the increasing user base and sophistication of the personalization features of platforms, the machine learning architectures under their hood are under increasing pressure to provide higher-quality recommendations without a correspondingly higher computational cost or complexity of the infrastructure.

The architecture of training pipelines is important in the process of defining the quality and efficiency of the recommendation systems at scale. A successful training infrastructure should trade off several conflicting goals, such as accuracy of models, computer efficiency, economy of storage, and system reliability. Relevant to the fact that recommendation systems are no longer modest collaborative filtering systems but deep learning mechanisms with billions of parameters, the role of an efficient training design has been exacerbated. Systems that spend resources on unnecessary computation or an inefficient data representation are constrained by hard limits on the complexity of models that they can represent with models using real infrastructure budgets.

Another important yet frequently under-researched cause of underperformance in contemporary recommendation systems is the very structure and representation of the training data. The prevailing approach to the production systems considers each individual impression as a separate training item, in which an impression is a single interaction between a user and a candidate item. The conceptual simplicity and the pipeline have obvious benefits for this training paradigm on the level of impression. Although these benefits may be practical, impression-level training presents a problem of

Independent Researcher, USA

structural inefficiency that is more problematic with scale. In a normal course of recommendation, users are presented with a number of candidate items at the same event. Everything shown on the same page is being exposed to the same contextual conditions, such as user identity, browsing history, state in the current session, device information, and geographical location, as well as time-based features. In impression-level training, this shared request context is replicated wholly in all the individual training examples produced with respect to that request [2].

Request-level training paradigms are yet another paradigm that can be used to overcome these inefficiencies by radically redefining the unit of learning. Instead of using individual impressions as training examples, request-level training uses complete request events as the fundamental training unit. Contextual features, which are common to all of the items in a request, are represented once in a request, and item-specific features are represented hierarchically under that request structure. This design is closer to the way recommendation systems are used in practice, in production settings, where decisions are made regarding the sets of items and ranking of sets of items within a common interaction environment, as opposed to single predictions about user-item pairs.

The paper discusses the theoretical basis, architecture, and system advantages of request-level training in large-scale recommendation systems. The discussion has found the particular structural inefficiencies and modeling constraints of impression-level training and request-level training in the literature as a hierarchical alternative to eliminate redundancy; discusses the architectural and infrastructure adaptations necessary to realize it; analyzes the efficiency benefits in a variety of dimensions; and puts request-level training in context with more general evolutionary trends in the design of a recommendation system. The thesis is that the closer the training data structures are to the realistic patterns of the recommendation decision-making, the more significant the difference in various aspects of system quality and capability and the opportunities of architectural sophistication are generated, which, in the conditions of impression-level capabilities, would be unfeasible.

2. Impression-Level Training versus the Request-Level Alternative Structural Inefficiencies

In recommendation systems, the natural default methodology developed was impression-level training, which can implicitly give a simple and intuitive abstract characterization of supervised learning pipelines. Under this paradigm, one training example is a user-item interaction opportunity, also known as an impression. There are such features as characterizing the user, characterizing the item, contextual signals capturing the interaction environment, and a label that tells whether the user interacted with the item in terms of actions, including clicking, viewing, purchasing, and consuming the content. This formulation can be succinctly formulated as a binary classification or regression task, with the models modeling the likelihood of engagement of each user-item pair separately. This simplicity has enabled it to be the default standard representation in industry and research, and systems in large-scale platforms traditionally structure their training data in terms of individual impressions as the fundamental unit of learning [3].

These practical advantages notwithstanding, there is an underlying inefficiency to impression-level training, which is even more problematic with the size of the system. The root of the problem is the representation of the contextual information in the case of having several items when the user is presented with the items in a single request event. In a real-world recommendation situation, a user is usually presented with more than one candidate item at a time. Everything represented alongside in a single request will have the same contextual conditions since it is exhibited to a user at the same time under the same environmental conditions. This common context consists of features that describe user identity; session-specific signals such as device type and browser specifications; time features based on timestamps; geographic features based on the location of the user; and historical engagement features based on the past user behavior.

In impression-level training, the entirety of this shared context information is replicated verbatim in each training example in each and every training example produced using a single request. A recommendation system with twenty items shown to a user would make twenty distinct training examples, all of which involve the exact set of user features, session state, temporal cues, and geographic features as well as contextual metadata

of that particular user. It is precisely and fully duplicated. The redundancy factor is represented by the number of items delivered per request and generally tens to hundreds in manufacturing systems. The first dimension of inefficiency is storage waste. Preprocessing pipelines that convert raw features into inputs that can be fed to a model have to reprocess the same contextual signals on all impressions in a request. The issue of model training doubles the computational waste to the extent that neural networks learn to encode shared contextual features in every impression within each batch of impressions [4].

In addition to wasting resources, the process of training at the impression level causes the basic modeling restrictions that limit the expressiveness of the recommendation systems. The current trend in recommendation has seen the creation of structured choices regarding groups of items as opposed to forecasting individual relevance results. Real recommendation objectives involve ranking quality, diversity across results, coverage of user interests, and trade-offs between exploration and exploitation. Impression-level training makes these relationships difficult to model effectively because each example is processed independently. The training framework provides no mechanism for reasoning about relative ranking, comparing items shown together, or optimizing for group-level objectives.

Request-level training addresses both the efficiency and modeling problems by fundamentally redefining what constitutes a training example. Instead of creating one example per impression, the complete request event becomes the unit of learning. A single training example now corresponds to an entire user interaction, including all candidate items that were presented together in the request. The data structure is hierarchical: shared contextual features that apply to all items are represented once at the request level, and item-specific features are nested within that request structure, associated with their respective candidates. The hierarchical structure eliminates contextual duplication entirely. Storage requirements for shared context decrease by a factor equal to the average number of items per request. Preprocessing efficiency improves correspondingly as feature transformation pipelines compute shared contextual features once per request rather than once per impression.

Request-level training also provides a natural foundation for modeling grouped decisions and within-request relationships. Because multiple items are represented together in each training example, models can incorporate architectural components that reason about their relative properties. Learning objectives can evolve to reflect multi-item decision criteria more directly. Request-level training enables richer objectives, including pairwise ranking losses that compare items shown together, listwise losses that optimize overall rankings and request-level metrics that reward group qualities like diversity or long-term engagement. The transition from impression-level to request-level training addresses multiple limitations simultaneously by eliminating massive redundancy in data representation while providing an architectural foundation for modeling the within-request relationships and group-level objectives that characterize real recommendation decisions.

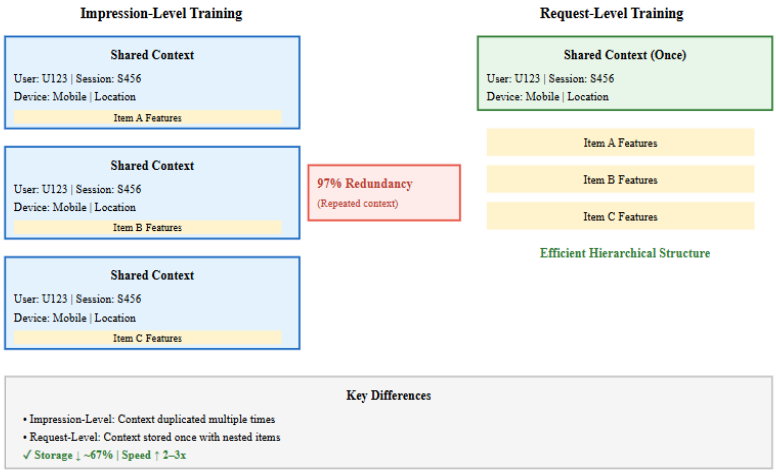

Figure 1: Comparison of Impression-Level and Request-Level Training Representations [3, 4]

3. Model Implications of Architecture and Infrastructure

Request-level training paradigms require a major change in the model architecture and system infrastructure far beyond mere data format adjustments. Models that are trained on impression-level inputs are architecturally clean to accept input pairs of users and items and give a scalar prediction based on one concatenated feature vector as input and a set of transformations. Request-level training, in turn, necessitates models that can concurrently process shared request context and sets of candidate items of variable length and reason about their interactions and produce predictions that represent aggregated group decision goals and not individual relevance scores.

The typical subject to architectural change is the hierarchical processing structure that decouples request-level feature encoding with item-level scoring functions. Common contextual elements are initially represented by specialized neural network elements that produce request-level representations that represent user state, session properties, temporal factors, and environmental indications. Request-level encodings are combined with item-specific features during candidate scoring either via concatenation, attention mechanisms, or more elaborate interaction layers. This separation allows computational reuse in terms of costly operations on shared context since such operations only have to be computed once per request and are not repeated across items of a candidate set. In the case of complex request-level features, the gains in terms of efficiency can be high. Contemporary recommendation systems tend to use sequential models of user interaction histories to express changing preferences and behavior patterns using recurrent networks, transformers, or other sequential models, which demand significant computation [5]. This sequence encoding is done separately with each item impression, although the user history is the same for all items in a request in impression-level training.

In addition to computational efficiency, request-level architectures support more sophisticated modeling of relationships between items that are displayed together. Architectures may include some element that explicitly reasons about item interactions, relative ranking, and group-level qualities that are based on candidate sets. Scoring by cross-attention mechanisms permits items to share information within scoring, which is important because the model is able to detect complementary or substitutable items and modify predictions. These architectural features are helpful for learning purposes that are more aligned with the actual recommendation system purposes. Request-level training supports more complex goals such as pairwise ranking losses, which optimize for the correct relative ranking of items directly; listwise losses, which work with full ranking quality measures; and group-level losses, which are aimed at properties such as diversity, coverage, or long-term user satisfaction [6].

Figure 2: Request-Level Model Architecture and Processing Flow. [5]

Nonetheless, at the request level, there is complexity in the implementation that has to be handled with care. The variable-length candidate sets will be a basic problem since varying numbers of items may be specified in different requests, so it becomes challenging to create consistent batches of items, which can be efficiently processed by the GPUs. It can be solved by means of dynamic batching, which puts requests with the same count of candidates together before masking shorter requests to a length limit, or more advanced methods such as nested batching, in which request-level and item-level features are solved individually. Request-level architectures also make debugging and model interpretation more difficult as predictions are determined using intricate interactions between shared context and multiple candidate items and possibly attention mechanisms or other interaction layers that enable items to affect the scores of other items.

The concept of infrastructure needs is not limited to the model architecture but to the whole data pipeline, which builds training datasets. Creation of request-level training data needs systems that can correctly cluster impressions that are of the same request event, identify common contextual features at the correct place, relate item-specific features with their candidates, and maintain any ordering or structural semantics used during the original interaction. When the logging is done in a distributed system across multiple services, the same user interaction could be recorded in different parts by different components. A complete request event reconstruction involves combining these distributed logs using request identifiers or timestamps and managing clock skew, partial failure of logging, and timing ambiguity [7].

Consistency problems occur on various levels of request-level data construction. Shared request context should be properly conformed with all the candidate items in such a way that all the items in the request are given the same contextual features. Misalignment errors in which objects in one request are context-sensitized by the other request may silently corrupt the quality of the model since the training data is a biased reflection of the conditions of the real interaction. Although these complications exist, request-level infrastructure can offer more than just the removal of redundancy. The hierarchical revision supports a more conceptual arrangement of features into shared context and item-specific signals, typically resulting in superior feature engineering customs and respondent feature pipelines generally. Request-level data construction can also enhance consistency between training and serving environments, eliminating train-serve skew that can result in worse performance in production than offline evaluation indicates.

Request-level training involves a large amount of infrastructure. Groups have to re-architect data pipelines to support grouped event processing, add strong join logic between distributed logging systems, create validation to identify grouping failures and misalignment of context, and create tooling to debug hierarchical training data. The adoption of request-level training should be done with cold realism towards the advantages and costs of the decision and be aligned to architectural decisions, to particular aspects of the system, and to organizational strengths and weaknesses as opposed to the adoption of paradigms on the premise of theoretical attractiveness.

4. Efficiency Benefits, Appraisal Model, and Implementation Factors

One of the strongest pragmatic reasons why request-level training paradigms should be adopted in large-scale recommendation systems is efficiency improvements. The nature of production systems is usually severely resource-constrained, which constrains the architectural sophistication (more than hypothetical possibilities). Request-level training cuts waste on several different dimensions of the training pipeline, effectively increasing the range of model architectures that are possible without corresponding hardware investment or operational cost increases.

Metric Category	Impression-Level Training	Request-Level Training	Improvement Factor
Storage Requirements	30B examples (1B requests × 30 items) with full context duplication	1B request examples with nested items	97% reduction in shared context storage
Preprocessing Throughput	Process 30B feature sets independently	Process 1B request contexts + 30B item features	30× reduction in shared context computation
Training Computation	Encode shared context 30× per request during forward/backward passes	Encode shared context once and reuse across items	2-3× overall training speedup

Table 1: Efficiency Comparison—Impression-Level vs. Request-Level Training. [8]

The efficiency benefits are seen in three major components of the system, which include storage infrastructure, preprocessing pipelines, and model training computation. The need to store information reduces significantly when the shared contextual features are modeled on a single occasion during each request, as opposed to being copied in all impressions. Preprocessing throughput scales linearly with the amount of redundant computation that is eliminated because feature transformation pipelines are now used to compute shared context only once per request instead of once per impression. This efficiency benefit results in the systems being able to handle bigger effective data volumes within the same time window and therefore be able to cover more history in training datasets [8].

The benefit of model training is the minimized redundant computation that is performed in terms of the forward and backward passes of neural networks. Request-level architectures have the ability to encode shared contextual properties in a single request and reuse those encodings when scoring multiple candidate items, avoiding repeated lookups of embeddings, dense layer transformations, and nonlinear activations of duplicated inputs.

These savings in efficiency are cumulative in the generation of strategic value, which is not based on immediate cost reductions. Request-level training eliminates waste and liberates computational capacity, which can be used to increase model sophistication. The paradigm shift is a force multiplier of engineering capacity, which allows architectural innovations that would otherwise be beyond the infrastructure budget. Request-level training modifies this calculus, making the consumption of base resources lower, and allowing complexity to grow without corresponding growth of infrastructure

To assess request-level training, elaborate models are needed to represent the various facets of system quality since the paradigm has implications of both efficiency and modeling ability in a system. At least three different layers are to be evaluated rigorously, namely, the metrics of computational efficiency that measure resource consumption, the metrics of quality that measure the predictive performance, and the metrics of downstream outcomes that measure the actual impact on user experience and business goals [9]. Computational efficiency metrics give quantitative information on the saving of resources throughout the training pipeline. Ranking quality measures provide an evaluation of whether request-level models are more effective at modeling the structure of recommendation decisions than impression-level baselines. Ranking-specific measures such as normalized discounted cumulative gain, mean average precision, and ranking correlation measures are used to measure consistency between predicted and observed item rankings in requests. Downstream outcome measures relate the modeling improvements with the measurable effects such as user engagement rates, session quality measures, content diversity measures, and business measures [10].

To have a credible evaluation of request-level training paradigms, comparison against strong baselines is a must. The value proposal is most persuasive when proved with respect to the mature and well-tuned systems at the impression level instead of the weak strawman systems. Where request-level strategies are found to outperform a strong impression-level baseline in both efficiency and quality factors at the same time, the case for

adoption is very strong since there is no one aspect of system performance that is compromised in favor of the other.

Evaluation Layer	Metrics	Purpose
Computational Efficiency	Storage compression ratio, preprocessing throughput, training wall-clock time, infrastructure cost reduction	Quantify resource savings across storage, preprocessing, and training pipelines
Ranking Quality	NDCG, mean average precision, ranking correlation, diversity metrics, calibration accuracy	Assess whether models better capture grouped decision structures and ranking objectives
Downstream Outcomes	User engagement rates, session quality, content diversity, user satisfaction, business metrics	Measure real-world impact on user experience and platform objectives

Table 2: Evaluation Framework for Request-Level Training. [9]

The implementation challenges should be considered against the benefits anticipated since not every system would have a positive value proposition. Situations with low fan-out per request have a proportionately smaller efficiency improvement since there is less contextual duplication to remove. The most notable practical obstacle to implementation is usually compatibility with current infrastructure. Numerous production systems have years of investment in impression-level systems in data engineering, feature computerization, model training systems, offline assessment tools, and production serving systems. To make the transition to request-level training, it may be necessary to make changes to all of these components at the same time. The effort of coordination between the various units can greatly delay adoption even in circumstances where the technical benefits are apparent.

Interpretability and debugging remain a challenge even post-implementation, as request-level models use more complicated prediction paths than those of impression-level models. To tell why a given item was given a given predicted score, one has to follow the hierarchical encoding process in which request-level features get processed independently of item-level features and see how the representations are integrated in the scoring of the candidates and possibly look at cross-item interactions. Request-level training teams should have better instrumentation and debugging to ensure that the quality of the models is good and that issues can be diagnosed successfully.

A variety of considerations, such as system scale, request characteristics, modeling needs, and organizational requirements, are trade-offs in deciding when request-level training is warranted. Large-scale platforms with billions of requests per day and a high average fan-out have the best case to be adopted since the efficiency improvements are the most stark as well as the investment in infrastructure amortizes over huge data volumes. On the other hand, small systems that do not have a significant volume of requests can lack efficiency gains to cover the migration costs. It is necessary to make the decision based on the critical analysis of the system-specific features, anticipated benefits, cost of implementation, and organizational priorities instead of adhering to the architecture fads blindly.

5. General Implications and Future Projections

Request-level training paradigms are not just an incremental optimization to recommendation system infrastructure. They are an ideal evolution of the way machine learning systems are designed to be applied to complex interactive scenarios with decisions based on structured contexts, grouped candidates, and multi-dimensional goals. With the maturity of the digital platforms and increasing demands of individuals on personalized experiences, the shortcomings of the simplistic data abstractions are becoming more visible. The systems should develop beyond the representations that push the complex decisions in the real world to excessively convenient yet structurally incorrect formats. The concept of request-level training is a more general trend towards architectural fidelity in which data structures, model architectures, and infrastructure design are more aligned with the

nature of recommendation problems than was the case in simpler and smaller historical systems.

Conceptual development of request-level training is an effort to capture the changing knowledge about the essence of recommendation systems. Early methods tended to think of recommendation as mostly being about the isolation of individual user-item affinity scores. Nonetheless, contemporary systems are becoming more conscious of the fact that recommendation is the act of coming up with structured decisions of groups of items in rich contextual settings in which user state, dynamic situations, time-based trends, and environmental conditions all have a bearing on what defines an optimal recommendation. Request-level training is used to meet this requirement; it arranges data in terms of complete interaction events and does not split them into independent impressions. The paradigm is consistent with a number of complementary trends, such as context-aware modeling, which considers more significant signals about user state; multi-objective optimization, which trades one goal against another; and session-based recommendation systems, which model continuous user walks as opposed to individual events [11].

Request-level training has implications on architecture that go beyond the existing functionality to influence the future development of recommendation systems. It is likely that next-generation architectures will use models that are more sophisticated and are able to jointly optimize many items, time steps, and objectives in parallel. Request-level paradigms give such a base by structuring the data in a hierarchy based on real decision events. The trend toward request-level training is also indicative of wider acceptance that data representation decisions are far-reaching in their consequences about what models might learn and how they can be trained. On a large scale, representational decisions tend to dictate what is possible over what is sophisticated in architecture.

Another dimension of the transition to request-level paradigms, which is more vital, is infrastructure co-evolution. As the complexity of training representations increases, all stages of the machine learning process, starting with the raw recording of events and ending with serving production, need to change together. This end-to-end integration enhances the integrity of the system and minimizes discrepancies between the offline training and online serving on many production recommendation systems [12]. Request-level training also has strategic significance in the future development of the system, as it is efficient. With the advance in recommendation models that become more complex, characterized by more extensive networks, larger embedding dimensions, and more elaborate attention mechanisms, the computational requirements also grow proportionate to these factors. Paradigms capable of minimizing waste and permitting more aggressive modeling with limited budgets become more important with an increasing scale of the systems.

There are some good directions that can be developed further due to the maturity of request-level training paradigms and their increased acceptance. Standard frameworks and reusable tooling would also help reduce adoption barriers, which may demand a lot of custom engineering at present. Studies of model architectures that are specially tailored to request-level inputs, as opposed to the adaptation of impression-level designs, may open further opportunities. Combining with other emerging paradigms offers synergistic opportunities such as multitask learning and reinforcement learning strategies for recommendation and online learning apps wherein training is a continuous process that is constantly updated by real-time feedback [13].

Recent experience with request-level training in recommendation systems can be argued to be relevant to other application tasks where grouped predictions, structured contexts, or batched decisions are applicable. Search ranking algorithms have similar issues wherein more than one document is accessed and ranked for each query. The conversational AI systems decide on response generation in a dialogue situation in common for more than one turn. The online advertisement systems pick and rank a variety of advertisements in parallel situations shared between the users as well as the pages. In the future, the request-level paradigm seems to be at the core of recommendation and ranking systems in the next generation. They give the representational base required to support advanced architectures that capture complex interactions in the context as well as run well on a big scale. The training at the request level has shown that a radical re-evaluation of representational traditions that have existed over decades could bring about significant improvements even in fully developed areas where the process of optimization is years old.

Conclusion

Traditional impression-level training has served as the foundation of recommendation system development for years, providing simplicity and compatibility with standard machine learning frameworks. However, its limitations have become increasingly apparent as systems scale to serve billions of daily interactions. The duplication of shared context across every impression in a request creates substantial waste in storage, preprocessing capacity, and computational resources. The independent processing of impressions constrains models from reasoning effectively about within-request relationships and optimizing for grouped decision objectives. These inefficiencies impose real costs on system capability and resource utilization that become unsustainable at the scales where modern platforms operate. Request-level training paradigms address these problems by restructuring the fundamental unit of learning around complete request events rather than isolated impressions. Shared contextual features are represented once per request with item-specific features nested within hierarchical structures. This design eliminates contextual duplication, reduces resource consumption across storage and compute dimensions, and enables richer modeling of ranking and selection decisions that reflect how recommendation systems actually function in production environments. The benefits extend across computational efficiency through elimination of redundant operations, modeling expressiveness through architectures that reason about item relationships and optimize request-level objectives, and infrastructure coherence through clearer separation of shared and item-specific features with better alignment between training and serving. Implementation requires significant engineering effort, including redesigned data pipelines to construct grouped request records, adapted model architectures to handle hierarchical inputs and variable-length candidate sets, and modified serving systems to realize efficiency benefits in production environments. Debugging and interpretation become more complex when predictions arise from grouped contexts, and coordination across teams and systems can be substantial. The value proposition depends heavily on system characteristics and organizational context. Large-scale platforms with high request fan-out, sophisticated ranking requirements, and mature engineering capacity often find compelling benefits in the combination of improved efficiency and enhanced modeling that request-level training provides despite implementation challenges. Looking forward, request-level training paradigms are positioned to become increasingly central to recommendation system design as they provide the representational foundation needed for next-generation architectures that must model complex contextual interactions while operating efficiently at massive scale. As platforms continue evolving toward more sophisticated personalization and context-aware recommendations, the alignment between training representations and actual decision structures becomes more critical. Request-level training offers that alignment along with practical efficiency benefits that matter in resource-constrained production environments. The paradigm shift represents a maturation of recommendation system design, reflecting growing recognition that training representations should match the true structure of recommendation decisions rather than conforming to convenient but mismatched abstractions. This evolution toward greater architectural fidelity characterizes the future of large-scale machine learning systems across recommendation, ranking, and related applications where efficiency and contextual modeling must advance together.

References

[1] Xiangnan He et al., "Neural Collaborative Filtering," ACM Digital Library, 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3038912.3052569

[2] Paul Covington et al., "Deep Neural Networks for YouTube Recommendations," ACM Digital Library, 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2959100.2959190

[3] Heng-Tze Cheng, et al., "Wide & Deep Learning for Recommender Systems," ACM Digital Library, 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2988450.2988454

[4] Guorui Zhou et al., "Deep Interest Network for Click-Through Rate Prediction," ACM Digital Library, 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/3219819.3219823

[5] Ashish Vaswani et al., "Attention is All You Need," NIPS, 2017. [Online]. Available: https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[6] Chris Burges et al., "Learning to rank using gradient descent," ACM Digital Library, 2005.

[Online]. Available: https://dl.acm.org/doi/10.1145/1102351.1102363
[7] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," ACM Digital Library, 2008. [Online]. Available: https://dl.acm.org/doi/10.1145/1327452.1327492
[8] Steffen Rendle, "Factorization Machines," ACM Digital Library, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5694074
[9] Thorsten Joachims, "Optimizing search engines using clickthrough data," ACM Digital Library, 2002. [Online]. Available: https://dl.acm.org/doi/10.1145/775047.775067
[10] Lihong Li et al., "A contextual-bandit approach to personalized news article recommendation," ACM Digital Library, 2010. [Online]. Available: https://dl.acm.org/doi/10.1145/1772690.1772758
[11] Balázs Hidasi et al., "Session-based Recommendations with Recurrent Neural Networks," arXiv:1511.06939 [cs.LG], 2016. [Online]. Available: https://arxiv.org/abs/1511.06939
[12] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2348283.2348408
[13] Michael Bendersky, W. Bruce Croft, "Modeling higher-order term dependencies in information retrieval using query hypergraphs," ACM Digital Library, 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2348283.2348408