Intelligenza Artificiale

Boosting RAG Efficiency with dyRAG: Dynamic Candidate Selection for Optimal Retrieval

--Manuscript Draft--

Manuscript Number:	
Full Title:	Boosting RAG Efficiency with dyRAG: Dynamic Candidate Selection for Optimal Retrieval
Short Title:	
Article Type:	Research Article
Keywords:	Retrieval-Augmented Generation (RAG); Dynamic Retrieval (dyRAG); large Llanguage model; Query Complexity; Relevant Information
Corresponding Author:	SALEM Mohammed University of Mascara: Universite de Mascara MASCARA, ALGERIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Mascara: Universite de Mascara
Corresponding Author's Secondary Institution:	
First Author:	Zoubida Asmaa Boudjenane
First Author Secondary Information:	
Order of Authors:	Zoubida Asmaa Boudjenane
	SALEM Mohammed
Order of Authors Secondary Information:	
Abstract:	Retrieval-Augmented Generation (RAG) enhances language model responses by incorporating external knowledge. However, its effectiveness heavily depends on the quality of the retrieved documents. Using a fixed number of retrieved documents K may fail to adapt to varying query complexity, often leading to suboptimal retrieval either including irrelevant documents or missing crucial ones. To address this issue, we propose dyRAG, a hybrid method that dynamically adjusts K based on query characteristics while maintaining computational efficiency. Our method improves retrieval performance by maximizing relevant information and minimizing noise. Experimental results show that it outperforms fixed-K retrieval, offering a more effective solution for optimizing RAG systems. Compared to traditional methods, it dynamically adjusts based on the query and dataset characteristics, offering greater adaptability across various contexts.
Opposed Reviewers:	
Additional Information:	
Question	**Response**
By submitting this article I agree with the IOS Press author copyright agreement	Yes
By submitting this article I agree with the IOS Press privacy policy	Yes

Boosting RAG Efficiency with dyRAG: Dynamic Candidate Selection for Optimal Retrieval

Journal Title
XX(X):1–9

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Zoubida Asmaa Boudjenane [1] and Mohammed SALEM [2]

Abstract

Retrieval-Augmented Generation (RAG) enhances language model responses by incorporating external knowledge. However, its effectiveness heavily depends on the quality of the retrieved documents. Using a fixed number of retrieved documents K may fail to adapt to varying query complexity, often leading to suboptimal retrieval either including irrelevant documents or missing crucial ones. To address this issue, we propose dyRAG, a hybrid method that dynamically adjusts K based on query characteristics while maintaining computational efficiency. Our method improves retrieval performance by maximizing relevant information and minimizing noise. Experimental results show that it outperforms fixed-K retrieval, offering a more effective solution for optimizing RAG systems. Compared to traditional methods, it dynamically adjusts based on the query and dataset characteristics, offering greater adaptability across various contexts.

Keywords

Retrieval-Augmented Generation (RAG), Dynamic Retrieval (dyRAG), large Llanguage model, Query Complexity, Relevant Information

Introduction

Large Language Models (LLMs) have emerged as state-of-the-art artificial intelligence systems that excel at understanding and generating human-like text Naveed et al. (2023), Built on deep learning architectures, particularly transformers Vaswani et al. (2017),these models showcase impressive adaptability across diverse tasks, such as machine translation Gu et al. (2018), text generation Liang et al. (2024) and question answering Zhang et al. (2023). Notable examples of LLMs include OpenAI GPT series Brown et al. (2020), Google BERT Devlin et al. (2019) and T5 Raffel et al. (2020). Despite their impressive capabilities, LLMs suffer from hallucination Naveed et al. (2023), often generating confident but incorrect information. This is a critical challenge for knowledge-intensive tasks requiring high accuracy Naveed et al. (2023). Additionally, their static nature being trained on fixed datasets Naveed et al. (2023) limits their ability to update knowledge dynamically, making them prone to outdated or inconsistent responses, especially for time-sensitive information Mousavi et al. (2025).

Researchers have explored various approaches to address this limitation, including transfer learning Zhuang et al. (2021), fine-tuning Howard and Ruder (2018) which adapt pre-trained models to specific tasks. Despite their strengths, they are computationally expensive and impractical for real-time knowledge updates. A promising alternative is Retrieval-Augmented Generation (RAG) Chuyuan et al. (2025), which integrates real-time retrieval from external knowledge sources with generative models, enables LLMs to access up-to-date information without retraining,(RAG) faces several limitations. For instance, The performance of RAG heavily depends on the quality of the retriever Karpukhin et al. (2020), as irrelevant or noisy documents can degrade the generator output. Additionally, a critical challenge is determining the optimal number of retrieved passages K to feed into the generator Lewis et al. (2020b). Often resulting in either insufficient or excessive information being retrieved, reduce coherence and increase computational overhead. This trade-off makes K a key factor in balancing performance and efficiency.

Current approaches, such as static K selection Lewis et al. (2020b), dynamic K adjustment attempt to address this trade-off but often introduce complexity or latency, Hybrid approaches Yuan et al. (2024),integrates static retrieval with dynamic retrieval. However, these approaches face significant limitations, including, high computational overhead, and a dependency on high-quality metadata for optimal performance. These limitations highlight the need for a more K selection robust approach, capable of balancing efficiency and accuracy across varying query types.

In this paper, we investigate the problem of selecting K in RAG and propose a novel approach to dynamically adjust K based on thresholds and a Mixture of Logits (MoL) Ding and Zhai (2024) scoring mechanism, ensuring efficient and context-sensitive candidate selection while reducing computational costs. The use of adaptive thresholds algorithm improves the relevance of retrieved documents, refining the set to include only the most contextually appropriate matches. Unlike traditional approaches, dyRAG

[1]Computer Science Department, University of Mascara, Algeria
[2]LISYS Laboratory, University of Mascara, Algeria

Corresponding author:
Mohammed SALEM, LISYS Lab, University of Mascara, BP 305 Route Mamounia, Mascara, 29000, Algeria.
Email: salem@univ-mascara.dz

adapts to the query and dataset characteristics, making it more flexible for diverse contexts. Furthermore, its design for handling large datasets ensures scalable and precise retrieval, which is crucial for RAG systems working with extensive corpora.

The remainder of this paper is as follows :We start by presenting an overview of the research problem, We summarize existing solutions, and highlight the gaps that our work addresses through a more dynamic approach to k selection.
Section 2: summarizes existing solutions, additionally, it provides a critical analysis of the strengths and limitations of these methods, as it identifies key gaps in their design and performance. As a result, there is a critical need for a more effective approach.
Section 3: aims to provide a detailed explanation of the RAG framework, including its components and how it operates.
Section 4: introduces the novel hybrid dynamic selection algorithm. Additionally, it provides detailed description of its methodology, explains its design, and discusses the manner in which it overcomes the limitations of existing approaches. Particular emphasis is placed on the benefits of this approach, particularly in improving retrieval accuracy and computational efficiency.
Section 5: presents experimental results that demonstrate the effectiveness of the hybrid dynamic selection algorithm. Highlight its superiority in terms of retrieval performance and adaptability to diverse query complexities.

Related works

The selection of an optimal number of retrieved items K in Retrieval-Augmented Generation (RAG) systems plays a crucial role in ensuring accurate and efficient responses. Several methods, including static, dynamic, and hybrid approaches have been proposed to address this challenge While these methods show promise in improving performance, they often face trade-offs between adaptability, efficiency when applied to complex or diverse query patterns. In this section, we will explore these methods, highlight their respective strengths and weaknesses, and introduce our hybrid solution as a more robust and adaptable alternative.

Static Top-K selection in Retrieval :In traditional Retrieval-Augmented Generation (RAG) systems, static K-selection is a widely used method where a fixed number of top-k documents are retrieved based on their relevance scores. Sparse retrieval Bai et al. (2020) methods where documents and queries are represented as high-dimensional, sparse vectors In these method, the ”keys” represent documents to be retrieved, and the ”values” are the importance or weight of those terms in the document such as query likelihood models Lafferty and Zhai (2017), TF-IDF Robertson and Walker (1997) and BM25 Zaragoza and Robertson (2009) often rely on this static approach to identify and rank documents. For instance, if k=5, the system retrieves the top 5 documents deemed most relevant to a query, regardless of the query's complexity or the variability in relevance distribution among documents. A major drawback of this approach is its rigidity. Since the number of retrieved documents K is fixed, it fails to adapt to the varying complexity of queries. This one-size-fits-all strategy can lead to suboptimal performance across diverse datasets or tasks.

Dynamic top-K Selection in Retrieval :addresses the limitations of static K by varying the number of retrieved documents based on the specific query or context. These approaches typically involve heuristics, machine learning models, or adaptive algorithms to determine the optimal K for each query such as Dynamic Query Cutoff (DQC) Culpepper et al. (2016) This approach uses machine learning models trained on query-document features to predict the optimal K for retrieval, a reinforcement learning-based approach Zhou and Agichtein (2020) where the agent learns to adjust K by maximizing downstream task performance (e.g accurate response generation in RAG) Another example is the Dynamic Selection of K Nearest Neighbors Hulett et al. (2012) where the selection process often depends on the distribution of neighbors, their distances to the query point, or their relevance scores. However, these methods come with limitations. They often introduce computational overhead due to the need for dynamic evaluation, which can increase latency in large-scale systems. Additionally, their performance heavily depends on the quality of the underlying models or distance metrics, making them sensitive to noisy data. As a result, achieving generalization across diverse datasets or domains remains a significant challenge.

Hybrid Approaches For Top-K selection in Retrieval: combine elements of static and dynamic K-selection to achieve a balance between simplicity and adaptability. These systems might use a static K as a baseline but dynamically adjust it such as, the ”Blended RAG” Sawarkar et al. (2024) approach enhances retrieval accuracy by combining semantic search techniques with hybrid query strategies, Furthermore STAYKATE (Static-Dynamic Hybrid Selection) Zhu et al. (2024) enhances LLM performance in scientific information extraction by integrating representativeness sampling from active learning with a retrieval-based strategy. Another example is DR-RAG (Dynamic Relevance for Retrieval-Augmented Generation) Hei et al. (2024) First, it retrieves an initial set of k_1 documents. Then, a classifier dynamically evaluates their relevance and retrieves additional relevant K_2 documents to enhance recall. However hybrid methods may face challenges in determining when to apply static or dynamic K-selection effectively, leading to potential trade-offs between retrieval accuracy and system efficiency. These limitations highlight the ongoing need for robust mechanisms that dynamically and efficiently adapt K to the unique characteristics of each query and dataset.

Our method leverages adaptive thresholds to dynamically refine the list of retrieved documents, to make sure that only the most contextually relevant matches are included. In contrast to traditional static or fixed K-selection techniques, which employ a consistent retrieval strategy for every query, our approach adapts to the specific characteristics of both the query and the dataset.

Overview of Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a method designed to enhance the performance of large language

models (LLMs) by integrating external, reliable knowledge sources into their response generation process **?**. RAG combines search capabilities with LLM prompting. The process involves feeding both the userss query and the relevant information retrieved by a search algorithm into the LLMs prompt Gao et al. (2023), enabling the model to generate answers based on the provided context. As

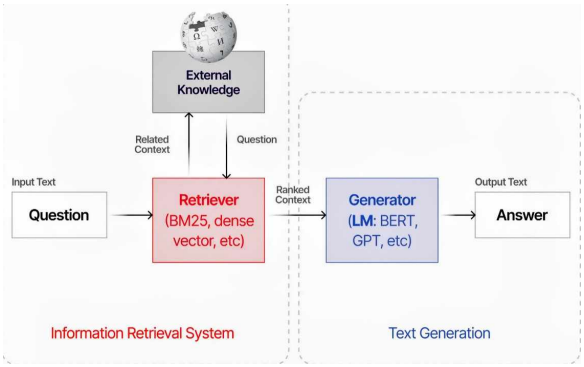


Figure 1. RAG system component Gupta et al. (2024).

illustrated in figure 1, the RAG system is built around two main components: the retriever and the generator. The retriever's role is to find relevant information from a pre-built data store, while the generator utilizes this retrieved data to produce coherent and contextually appropriate content. The overall RAG process operates as follows:

Retrieval in RAG Systems

Retrieval is the process of identifying and gathering information system resources that match a specific information need. To break it down, imagine these resources as a key-value store, represented as pairs:

$$\{(k_i, v_i)\}_{i=1}^{N} \tag{1}$$

where each key k_i is linked to a corresponding value v_i (often, the key and value are the same). When a query q is provided, the goal is to find the top-k keys that are most similar to the query using a similarity function s, and then retrieve their associated valuesGupta et al. (2024).

Depending on the similarity function used, retrieval methods can be grouped into categories such as sparse retrievalBai et al. (2020)which relies on exact term matching techniques like BM25 and TF-IDF, dense retrievalKarpukhin et al. (2020)which Uses deep learning models to encode queries and documents into dense vector embeddings, retrieving relevant documents based on semantic similarity rather than exact word matching, and others. For the widely used sparse and dense retrieval approaches, the process typically involves two main steps:

- Encoding each object into a specific representation (e.g., term-based for sparse retrieval, vector embeddings for dense retrieval).
- Building an index to organize and efficiently retrieve relevant data during search .

This structured approach ensures that the most relevant information is retrieved quickly and accurately Zhao et al. (2024).

Generator in RAG Systems

In Retrieval-Augmented Generation (RAG) systems, the generator Huang and Huang (2024) is a key component responsible for crafting the final output. It works by combining the retrieved information with the users input query to produce a well-structured and contextually relevant response.

Once the retriever fetches relevant data from external sources,the generator processes and integrates this information into a well-structured and meaningful response. At the core of this process is a Large Language Model (LLM), which ensures that the generated text is not only fluent and accurate but also remains relevant to the original query.

In various generative tasks, different models are selected based on the specific requirements: Models like BART Lewis et al. (2020a), GPT Brown et al. (2020)and and T5 Raffel et al. (2020) are commonly used for tasks such as text summarization and translation, Vision-Language Models (VLMs) Radford et al. (2021) are employed to generate textual descriptions from images, for text-to-Code Generation: Models like OpenAIs Codex Chen et al. (2021) are designed to translate natural language prompts into executable code.

dyRAG: The proposed Solution

In traditional candidate selection systems, the reliance on static or fixed top-K retrieval lead to suboptimal retrieval performance, especially in scenarios where the relevance distribution of candidates is highly variable. In this section, we introduce the Mixture of Logits (MoL) and describe its role in representing a learned similarity function. Table 1 summarizes the notations used in this paper and we propose Dynamic Candidate Selection, a novel algorithm that adaptively refines the candidate selection process by leveraging component-level embeddings and a Mixture of Logits (MoL) scoring mechanism.

Mixture of Logits

The Mixture of Logits (MoL) Zhai et al. (2023); Ding and Zhai (2024) method provides a flexible and adaptive mechanism for computing similarity scores between queries and items. Given a query q and an item x, MoL assumes that both are mapped to P groups of low-rank embeddings, denoted as $f_p(q)$ and $g_p(x)$, respectively. These embeddings are parameterized by neural networks that extract features from the query and item. The similarity between q and x is computed as a weighted sum of the inner products of these embeddings, where the weights $\pi_p(q, x)$ Zhai et al. (2023); Ding and Zhai (2024) are adaptive gating parameters constrained to sum to 1:

$$\text{Similarity}(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x) \rangle. \tag{2}$$

This formulation allows MoL to capture complex relationships between queries and items, making it particularly suitable for dynamic candidate selection tasks.

To efficiently scale MoL for large datasets and hardware-optimized implementations, the formulation is extended by decomposing the dot products into batched outer

Notation	Description
q (Q, $\|Q\|$)	A single query (set of all queries, total number of queries).
x (X, $\|X\|$)	A single item (set of all items, total number of items).
$\phi(q, x)$	The learned similarity function, Mixture-of-Logits (MoL), which computes the similarity between q and x.
P (P_q, P_x)	Total number of low-rank embeddings ($P = P_q \times P_x$), where P_q and P_x are the number of embeddings for q and x, respectively.
$\pi_p(q, x)$	Weight for the p-th (or p_q-th and p_x-th) embedding set for (q, x).
$f(q)$ ($f_p(q)$)	Learned embedding for the query (p-th component-level embedding for q).
$g(x)$ ($g_p(x)$)	Learned embedding for the item (p-th component-level embedding for x).
$\langle f(q), g(x)\rangle$	Dot product similarity function: $\langle f(q), g(x)\rangle = g(x)^T f(q)$.

Table 1. Notation Table

products of query-side and document-side embeddings. This decomposition improves computational efficiency, particularly on accelerators like GPUs, by normalizing the embeddings using the l_2-norm: Zhai et al. (2023)

$$\phi(q, x) = \sum_{pq=1}^{P_q} \sum_{px=1}^{P_x} \pi_{pq,px}(q, x) \frac{\langle f_{pq}(q), g_{px}(x)\rangle}{||f_{pq}(q)||_2 ||g_{px}(x)||_2} \quad (3)$$

Since embedding normalization can be precomputed, both formulations remain interchangeable in practical applications. Furthermore, it is possible to decompose any high-rank matrix into a mixture of logits based on low-rank matrices, demonstrating the flexibility and scalability of this approach in large-scale information retrieval tasks.

Retrieval Algorithm

We propose an **adaptive threshold mechanism** (referred to as (Dynamic Candidate Selection) that leverages the Mixture of Logits (MoL) framework to enhance the candidate retrieval process. The mechanism operates as follows:

1. **Component-Level Embeddings**: Component-level embeddings are generated for all items in the dataset X. These embeddings facilitate efficient similarity computations during retrieval. Formally,

$$X_p \leftarrow \{g_p(x) \mid x \in X\}. \quad (4)$$

2. **Initial Candidate Retrieval:** This step involves computing similarity scores between a query representation and item representations for each feature component $p \in P$.

$$S_p \leftarrow \{\langle f_p(q), g_p(x)\rangle : x \in X_p\} \quad (5)$$

Here, $f_p(q)$ and $g_p(x)$ represent the feature embeddings of the query q and item x for the p-th component, respectively. The dot product $\langle f_p(q), g_p(x)\rangle$ measures the relevance of each item $x \in X_p$ with respect to q, producing a set of scores S_p .

3. **Dynamic Threshold Adjustment**: To improve retrieval quality, **Mixture of Logits (MoL)** scores are computed for each candidate $x \in G$. The adaptive gating weights $\pi_p(q, x)$ allow the algorithm to dynamically adjust the retrieval threshold T_{adaptive} based on the MoL scores. The scoring function is defined as:

$$\phi(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x)\rangle. \quad (6)$$

The adaptive threshold T_{adaptive} is set as the minimum score among the candidates:

$$T_{\text{adaptive}} = \min\{s \mid s \in G\}. \quad (7)$$

4. **Refinement and Top-K Selection**: Using the adaptive threshold T_{adaptive}, additional relevant candidates are retrieved, expanding the candidate set G'. This is achieved by including candidates whose scores exceed the threshold:

$$G' \leftarrow G \cup \{x \mid s_p \geq T_{\text{adaptive}}\}. \quad (8)$$

The algorithm then sorts G' based on MoL scores to select the most relevant top-k candidates.

4. **Exact Top-K Selection**: Finally, the candidates in G' are sorted by their MoL scores, and the exact top-k items are extracted:

$$G_{\text{final}} = \text{Top-k}(G', \phi(q, x)). \quad (9)$$

As shown in Algorithm1, the retrieval process dynamically adjusts the threshold based on MoL scores.

Evaluation metrics in recommendation system

The retrieval algorithm is integrated into a recommendation system, and its effectiveness is assessed using standard ranking metrics. These metrics are widely used in sequential recommendation tasks to evaluate the model's ability to rank relevant items higher in a user's preference list:

- **Hit Rate at** K (HR@K) evaluates the proportion of users who have at least one relevant item in their top-K recommendations. It is computed as Jadon and Patil (2023):

$$HR@k = \frac{1}{|U|} \sum_{u \in U} I(|\text{rel}(u) \cap \text{rec}_k(u)| > 0) \quad (10)$$

where $\text{rel}(u)$ denotes the relevant items for user u, $\text{rec}_k(u)$ represents the top-K recommended items, and $I(\cdot)$ is an indicator function that returns 1 if at least one relevant item is present in the recommendations, otherwise 0.

Algorithm 1 Hybrid Exact Top-k with Threshold-Based k Selection

Input: Query q, Set of items X, Component-level embeddings: $f_p(q)$, $g_p(x)$ for $p \in P$, $x \in X$, Initial threshold T_{init}

Output: Exact top k items based on dynamic threshold selection, G_{final}

Set $G \leftarrow \emptyset$ ▷ Initial candidate set
for each component $p \in P$ **do**
  $X_p \leftarrow \{g_p(x) \mid x \in X\}$ ▷ Precompute embeddings
end for
1. Initial Candidate Retrieval:
for each component $p \in P$ **do**
  Compute dot product scores: with (eq(5))
  Retrieve items with scores $S_p \geq T_{\text{init}}$
  Add these items to G
end for
2. Adjust k Dynamically:
for each $x \in G$ **do**
  Compute MoL scores $s \leftarrow \phi(q, x)$ using : eq(6)
  Set $T_{\text{adaptive}} = \min\{s : s \in G\}$
end for
3. Refine Candidate Set with Adaptive k:
$G' \leftarrow \emptyset$
for each component $p \in P$ **do**
  Retrieve items from X_p with scores $S_p \geq T_{\text{adaptive}}$
  Add these items to G'
end for
4. Select Exact Top-k Items:
for each component $p \in P$ **do**
  Compute MoL scores for all items in G'
  Sort G' by MoL scores in descending order
  Select the top k items from G' where k is the number of items in G' exceeding T_{adaptive}
end for
Return: G_{final} ▷ Retrieve Top k items from G'

HR@K can be evaluated at different values of K (e.g., 3, 5, 10) to assess model performance across various ranking depths.

- **Mean Reciprocal Rank (MRR)**:is a ranking quality metric that measures how quickly a system retrieves the first relevant item. It is calculated as the average of reciprocal ranks across all users or queries,MRR ranges from 0 to 1, with higher values indicating better performance Jadon and Patil (2023).

$$\text{MRR} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{\text{rank}_u}, \tag{11}$$

where rank_u is the position of the first relevant item for user u within the top-K results.
U represents the total number of users (for recommendation systems) or queries (for information retrieval tasks) in the dataset.

- **Normalized Discounted Cumulative Gain (NDCG)**: Assesses the ranking quality while accounting for position importance. The NDCG@k is computed as:Jadon and Patil (2023)

$$\text{NDCG@}k = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{\text{DCG@}k}{\text{IDCG@}k}, \tag{12}$$

where DCG@k is the Discounted Cumulative Gain at position k:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}, \tag{13}$$

and IDCG@k is the Ideal DCG@k, computed by sorting the items by their true relevance scores.

Experimental Results

In order to measure the effectiveness of our proposed method, we perform a comprehensive evaluation of both the retrieval algorithm and the Mixture-of-Logits (MoL) approach. This assessment focuses on the next-item prediction task, a fundamental challenge in recommendation systems Zhu et al. (2018) Kang and McAuley (2018), where the goal is to predict the most relevant item a user will interact with next based on their past interactions.

Dataset description

We conducted experiments using two MovieLens datasets : ML-100K and ML-1M Harper and Konstan (2015)which are widely used benchmarks for evaluating sequential recommendation models as detailed in Table2.

Dataset	Description
MovieLens-100K	100,000 ratings (1-5 scale) from 943 users on 1,682 movies. Each user has rated at least 20 movies. Includes user demographic information (age, gender, occupation, zip code).
MovieLens-1M	1,000,209 ratings from 6,040 users on approximately 3,900 movies. Data collected from users who joined MovieLens in 2000. Represents a larger-scale recommendation scenario.

Table 2. Summary of MovieLens datasets

Experiments Setup

For both datasets, We utilize the SASRec architecture as our sequential user encoder, a model renowned for achieving state-of-the-art performance in next-item prediction tasks. This architecture processes the user's historical interaction sequence, generating embeddings that encapsulate the user's preferences at each time step. These embeddings serve as the foundation for predicting the next item in the sequence.

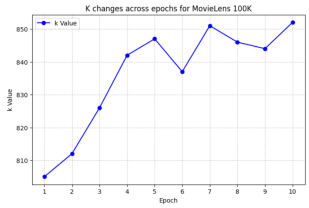
The query q represents the user's state at a specific time step, derived from their interaction history. In the MoL (Mixture of Logits) framework, q is transformed into Pq embeddings through a multi-layer perceptron (MLP).

Impact of Hyperparameters

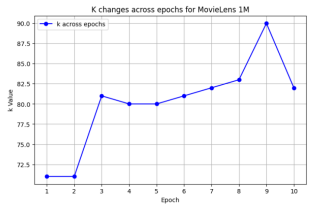
For fair comparison, we maintained consistent architectural choices and training conditions across all experiments, We conducted an extensive hyperparameter analysis comparing both approaches (Hybrid+SAS and MoL+SAS) across different architectural configurations. All experiments were Implemented in TensorFlow and trained on a Google Colab environment with a T4 GPU. We use the Adam optimizer with a learning rate of 0.001. For the hybrid algorithm, we initialize the threshold (Tinit) to 0.3 and adaptively adjust it during training. We discuss detailed hyperparameter settings in table 3, table 4

Both approaches demonstrate significant performance improvements as model capacity increases, with larger configurations consistently delivering better results. For instance, when increasing the model's capacity such as expanding the (Max Sequence Length: 512, Embedding Dimension: 512, Number of Heads: 4, Feed-Forward Dimension: 512) the hybrid algorithm combined with SASRec (Hybrid+SAS) achieves notable gains, particularly in the most resource-intensive setup. On the ML-100K dataset, Hybrid+SAS reaches a score of**0.8120** compared to the baseline's **0.8016**, while on ML-1M, it achieves **0.8350** versus **0.8276** The ML-1M dataset generally benefits more from increased capacity, with the performance gap between datasets narrowing as the model scales. While smaller configurations provide a balance of efficiency and performance, the largest configuration, despite its higher computational demands, yields the best results, making it suitable for scenarios where resources are not a constraint. Both approaches show similar benefits from scaling, but Hybrid+SAS maintains a consistent edge in performance.

As shown in Figures 2a and 2b, the value of k fluctuates across epochs for both MovieLens 100K and MovieLens 1M datasets. These variations indicate the adaptive nature of k in response to the dataset characteristics and training progress. In the following section, we analyze how these changes influence model performance.



(a) K changes across epochs for MovieLens 100K



(b) K changes across epochs for MovieLens 1M

Figure 2. Comparison of K changes across epochs for different datasets

the value of K changed dynamically during training, with different behaviors. For the MovieLens 100K dataset, K showed a steady increase across epochs, whereas for the MovieLens 1M dataset, K began at a lower value, experienced fluctuations, peaked, and then stabilized. This indicates that the larger dataset necessitated more adaptive adjustments in retrieval compared to the smaller dataset.
Figure 3 summarizes the performance of our hybrid algorithm compared to the MoL-based approach on the MovieLens 100K and 1M datasets, both tested using a Max Sequence Length of 50, an Embedding Dimension of 128, 2 Attention Heads, a Feedforward Dimension of 128, a Batch Size of 128, and trained for 10 epochs .

The variations in K help explain the performance

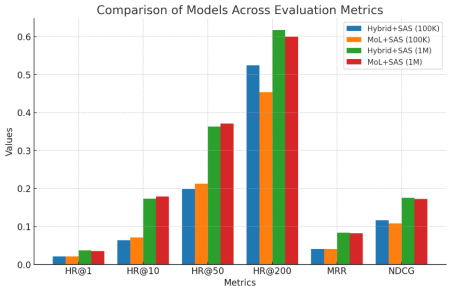


Figure 3. Comparison of Models Across Evaluation Metrics

differences seen in the fig 3. The Hybrid+SAS model applied to the MovieLens 1M dataset, which exhibited more dynamic changes in K achieved higher HR@50(0.3631) and HR@200(0.6170) scores, reflecting better long-range recommendation quality. The fluctuations in K in the 1M dataset likely allowed the model to balance exploration and precision, improving its overall ranking performance.

On the other hand, the more gradual K increase in the 100K dataset resulted in relatively lower scores, suggesting that a static or overly conservative K selection might limit retrieval effectiveness.

Additionally, the MoL+SAS model achieved better performance than Hybrid+SAS in HR@10 for both datasets. This aligns with the observation that MoL+SAS tended to retrieve fewer but more relevant items at shorter ranking positions. This behavior can be attributed to how K evolved during training?smaller K values in earlier epochs likely enabled MoL+SAS to maintain higher precision at shorter ranks. These findings highlight the importance of dynamically tuning K based on dataset characteristics to optimize recommendation effectiveness.

The results demonstrate that the Hybrid Exact Top-k algorithm effectively leverages both sequential user behavior and item metadata to improve recommendation quality. The adaptive MoL threshold allows the model to dynamically refine candidate items during training, leading to better performance. The improvements are consistent across both datasets, highlighting the robustness of our approach.

Conclusion

Choosing the right value for K in retrieval-based systems is a significant challenge, as it directly impacts both the efficiency and accuracy of information retrieval. While using a fixed K is straightforward, it often fails to adapt to the varying complexities of different queries. This can result in either missing important information or retrieving irrelevant data, which adds noise to the results. On the other hand, dynamically adjusting K offers greater flexibility but comes with its own set of challenges, such as increased computational costs and inconsistencies in determining the optimal retrieval size.

To address these limitations, we introduced a hybrid dynamic K selection strategy that strikes a balance between

Table 3. Performance comparison of Hybrid+SAS and MoL+SAS models on the 100kMovies dataset

Model	Max Seq. Len.	Embed. Dim.	Heads	FFN Dim.	Batch Size	Epochs	Val. Loss	Val. Acc.
Hybrid+SAS	50	128	2	128	128	12	5.3036	0.1584
MoL+SAS	50	128	2	128	128	12	5.3152	0.1582
Hybrid+SAS	128	256	4	256	128	10	3.6364	0.4301
MoL+SAS	128	256	4	256	128	10	3.6374	0.4299
Hybrid+SAS	512	512	4	512	128	10	**1.2808**	**0.8120**
MoL+SAS	512	512	4	512	128	10	1.3050	0.8016

Table 4. Performance comparison of Hybrid+SAS and MoL+SAS models on the 1M Movies dataset

Model	Max Seq. Len.	Embed. Dim.	Heads	FFN Dim.	Batch Size	Epochs	Val. Loss	Val. Acc.
Hybrid+SAS	50	128	2	128	128	10	**4.5721**	**0.1530**
MoL+SAS	50	128	2	128	128	10	4.5733	0.1526
Hybrid+SAS	128	256	4	256	128	10	**3.4152**	**0.3680**
MoL+SAS	128	256	4	256	128	10	3.4671	0.3627
Hybrid+SAS	512	512	4	512	128	10	**1.0275**	**0.8350**
MoL+SAS	512	512	4	512	128	10	1.0350	0.8276

adaptability and efficiency. Our approach adjusts K based on the unique characteristics of each query while keeping computational demands manageable. This method has shown promising results, improving retrieval performance by minimizing unnecessary noise and ensuring that critical information is consistently captured.

Future work can explore enhancing our model with learning-based retrieval strategies such as reinforcement learning or attention mechanisms could refine the dynamic selection of K Additionally, testing our approach in real-world scenarios and conducting large-scale evaluations across diverse domains will help validate its robustness and adaptability in various knowledge bases and retrieval settings.

Declaration of conflicting interests

The authors have no Conflict of Interests to share.

References

Bai Y, Li X, Wang G, Zhang C, Shang L, Xu J, Wang Z, Wang F and Liu Q (2020) Sparterm: Learning term-based sparse representation for fast text retrieval. *CoRR* abs/2010.00768. URL https://arxiv.org/abs/2010.00768.

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al. (2020) Language models are few-shot learners. *CoRR* abs/2005.14165. URL https://arxiv.org/abs/2005.14165.

Chen M, Tworek J, Jun H, Yuan Q, de Oliveira Pinto HP, Kaplan J, Edwards H, Burda Y, Joseph N, Brockman G, Ray A et al. (2021) Evaluating large language models trained on code. *CoRR* abs/2107.03374. URL https://arxiv.org/abs/2107.03374.

Chuyuan W, Ke D, Shengda Z, Hongchun W, Shuqiang H and Jie L (2025) Enhanced recommendation systems with retrieval-augmented large language model. *JAIR* 82. DOI:https://doi.org/10.1613/jair.1.17809.

Culpepper JS, Clarke CLA and Lin J (2016) Dynamic cutoff prediction in multi-stage retrieval systems. In: *Proceedings of the 21st Australasian Document Computing Symposium*. Melbourne, Australia. URL https://culpepper.io/publications/ccl16-adcs.pdf.

Devlin J, Chang MW, Lee K and Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C and Solorio T (eds.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, Minnesota. DOI:10.18653/v1/N19-1423.

Ding B and Zhai J (2024) Efficient retrieval with learned similarities. *CoRR* abs/2407.15462. DOI:10.48550/ARXIV.2407.15462.

Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Guo Q, Wang M and Wang H (2023) Retrieval-augmented generation for large language models: A survey. *CoRR* abs/2312.10997. URL https://doi.org/10.48550/arXiv.2312.10997.

Gu J, Wang Y, Cho K and Li VO (2018) Search engine guided neural machine translation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. pp. 5133–5140. DOI:10.1609/AAAI.V32I1.12013.

Gupta S, Ranjan R and Singh SN (2024) A comprehensive survey of retrieval-augmented generation (RAG): evolution, current landscape and future directions. *CoRR* abs/2410.12837. DOI:10.48550/ARXIV.2410.12837.

Harper FM and Konstan JA (2015) The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5(4). DOI:10.1145/2827872.

Hei Z, Liu W, Ou W, Qiao J, Jiao J, Song G, Tian T and Lin Y (2024) DR-RAG: applying dynamic document relevance to retrieval-augmented generation for question-answering. *CoRR* abs/2406.07348. DOI:10.48550/ARXIV.2406.07348.

Howard J and Ruder S (2018) Universal language model fine-tuning for text classification. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL , Melbourne, Australia,*. pp. 328–339. DOI:10.18653/V1/P18-1031.

Huang Y and Huang J (2024) A survey on retrieval-augmented text generation for large language models. *CoRR* abs/2404.10981. DOI:10.48550/ARXIV.2404.10981.

Hulett C, Hall A and Qu G (2012) Dynamic selection of k nearest neighbors in instance-based learning. In: *2012 IEEE 13th International Conference on Information Reuse & Integration (IRI)*. Las Vegas, NV, USA. DOI:10.1109/IRI.2012.6302995.

Jadon A and Patil A (2023) A comprehensive survey of evaluation techniques for recommendation systems. *CoRR* abs/2312.16015. DOI:10.48550/ARXIV.2312.16015.

Kang W and McAuley JJ (2018) Self-attentive sequential recommendation. *CoRR* abs/1808.09781. URL http://arxiv.org/abs/1808.09781.

Karpukhin V, Oguz B, Min S, Wu L, Edunov S, Chen D and Yih W (2020) Dense passage retrieval for open-domain question answering. *CoRR* abs/2004.04906. URL https://arxiv.org/abs/2004.04906.

Lafferty J and Zhai C (2017) Document language models, query models, and risk minimization for information retrieval. *SIGIR Forum* 51(2): 251–259. DOI:10.1145/3130348.3130375. URL https://doi.org/10.1145/3130348.3130375.

Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V and Zettlemoyer L (2020a) BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension : 7871–7880DOI:10.18653/V1/2020.ACL-MAIN.703.

Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih Wt et al. (2020b) Retrieval-augmented generation for knowledge-intensive nlp tasks. *CoRR* abs/2005.11401. URL https://arxiv.org/abs/2005.11401.

Liang X, Wang H, Wang Y, Song S, Yang J, Niu S, Hu J, Liu D, Yao S, Xiong F and Li Z (2024) Controllable text generation for large language models: A survey. *CoRR* abs/2408.12599. DOI:10.48550/ARXIV.2408.12599.

Mousavi SM, Alghisi S and Riccardi G (2025) Llms as repositories of factual knowledge: Limitations and solutions. *CoRR* abs/2501.12774. DOI:10.48550/ARXIV.2501.12774.

Naveed H, Khan AU, Qiu S, Saqib M, Anwar S, Usman M, Barnes N and Mian A (2023) A comprehensive overview of large language models. *CoRR* abs/2307.06435. DOI:10.48550/ARXIV.2307.06435. URL https://doi.org/10.48550/arXiv.2307.06435.

Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G and Sutskever I (2021) Learning transferable visual models from natural language supervision. *CoRR* abs/2103.00020. URL https://arxiv.org/abs/2103.00020.

Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W and Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21: 140:1–140:67. URL https://arxiv.org/abs/1910.10683.

Robertson SE and Walker S (1997) On relevance weights with little relevance information. In: Belkin NJ, Narasimhalu AD, Willett P, Hersh WR, Can F and Voorhees EM (eds.) *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Philadelphia, PA, USA, pp. 16–24. DOI:10.1145/258525.258529.

Sawarkar K, Mangal A and Solanki SR (2024) Blended RAG: improving RAG (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. *CoRR* abs/2404.07220. DOI:10.48550/ARXIV.2404.07220.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L and Polosukhin I (2017) Attention is all you need. In: *Advances in Neural Information Processing Systems*, volume 30. URL https://arxiv.org/abs/1706.03762.

Yuan Y, Liu C, Yuan J, Sun G, Li S and Zhang M (2024) A hybrid RAG system with comprehensive enhancement on complex reasoning. *CoRR* abs/2408.05141. DOI:10.48550/ARXIV.2408.05141.

Zaragoza H and Robertson SE (2009) The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3(4): 333–389. DOI:10.1561/1500000019.

Zhai J, Gong Z, Wang Y, Sun X, Yan Z, Li F and Liu X (2023) Revisiting neural retrieval on accelerators. *CoRR* abs/2306.04039. DOI:10.48550/ARXIV.2306.04039.

Zhang Q, Chen S, Xu D, Cao Q, Chen X, Cohn T and Fang M (2023) A survey for efficient open domain question answering. In: Rogers A, Graber JB and Okazaki N (eds.) *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada, pp. 14447–14465. DOI:10.18653/v1/2023.acl-long.808.

Zhao P, Zhang H, Yu Q, Wang Z, Geng Y, Fu F, Yang L, Zhang W and Cui B (2024) Retrieval-augmented generation for ai-generated content: A survey. *CoRR* abs/2402.19473. DOI:10.48550/ARXIV.2402.19473.

Zhou J and Agichtein E (2020) Rlirank: Learning to rank with reinforcement learning for dynamic search. In: Huang Y, King I, Liu T and van Steen M (eds.) *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24*. pp. 2842–2848. DOI:10.1145/3366423.3380047.

Zhu C, Shimada K, Taniguchi T and Ohkuma T (2024) STAYKATE: hybrid in-context example selection combining representativeness sampling and retrieval-based approach - A case study on science domains. *CoRR* abs/2412.20043. DOI: 10.48550/ARXIV.2412.20043.

Zhu H, Zhang P, Li G, He J, Li H and Gai K (2018) Learning tree-based deep model for recommender systems. *CoRR* abs/1801.02294. URL http://arxiv.org/abs/1801.02294.

Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H and He Q (2021) A comprehensive survey on transfer learning. *Proceedings of the IEEE* 109(1): 43–76.