Selection and Prediction of Retrieval Strategies for RAG Systems Based on Machine Learning Algorithms

Zixuan Jin

School of Artificial Intelligence, Xiamen University of Technology, Xiamen, China
3469106061@qq.com

Abstract. The popularization speed of generative artificial intelligence technology is accelerating. The Retrieval Augmented Generation (RAG) system, with its feature of enhancing output accuracy by integrating external knowledge bases, has been widely applied in fields such as intelligent question answering and professional document generation. This study first conducted a correlation analysis and then carried out comparative experiments using multiple machine learning algorithms. The results showed that the KELM-LSTM-Transformer algorithm proposed in this paper demonstrated significant advantages in the comparison: Its accuracy rate, recall rate, precision rate and F1 value all exceed 99.5%, with some indicators approaching 100%, and the AUC is also close to 100%. This algorithm not only significantly outperforms relatively weak algorithms such as AdaBoost, decision tree, and KNN, but also outperforms medium and high-performance algorithms like GBDT, ExtraTrees, SVM, and XGBoost. Even compared with the outstanding random forest and multi-head attention LSTM, its core classification index still maintains a higher level, and the AUC index has a particularly prominent advantage, fully demonstrating its superiority in complex feature capture and sequence information modeling. This achievement provides an efficient algorithmic solution for the performance optimization of the RAG system and also offers a reference for technical research in related fields. It is of great significance for promoting the precise application of generative AI in professional scenarios.

Keywords: RAG, LSTM, Transformer

1. Introduction

With the rapid popularization of generative artificial intelligence technology, Retrieval Enhanced Generation (RAG) systems have been widely applied in fields such as intelligent question answering and professional document generation due to their ability to enhance output accuracy by integrating external knowledge bases [1]. The retrieval strategy, as the core link of the RAG system, directly determines the access efficiency of the knowledge base and the accuracy of information matching. The rationality of its selection plays a key role in the overall performance of the system. At present, most RAG systems rely on manually preset rules or a single fixed strategy, which is difficult to adapt to dynamically changing application scenarios [2]. When facing heterogeneous data mixed with text, tables and images, traditional strategies are prone to the problem of insufficient feature

extraction. In tasks such as real-time question answering and multi-round dialogues, static strategies cannot adjust the retrieval priority according to the changes in context semantics, resulting in fluctuations in information recall rates or interference from redundant information. There is an urgent need for a technical solution that can dynamically predict the optimal retrieval strategy [3].

Machine learning algorithms provide core support for solving the dynamic problem of RAG retrieval strategy selection [4]. Compared with traditional rule-based methods, machine learning can automatically learn the correlation rules between strategy selection and system performance by modeling historical retrieval data, task features, data types and other information [5]. For instance, the decision tree algorithm can clearly demarcate the boundaries of strategy selection in different data scenarios, and the convolutional neural network can effectively extract local data features to assist in strategy judgment. However, the existing algorithms still have obvious limitations. On the one hand, most single-modal algorithms have difficulty handling the increasing heterogeneous data in RAG systems. On the other hand, traditional recurrent neural networks are prone to vanishing gradients when capturing the historical dependencies of long sequence retrieval, and cannot fully utilize the context information to optimize the strategy selection, which restricts the accuracy of strategy prediction [6].

To overcome the deficiencies of existing algorithms in long sequence dependency and multimodal feature fusion, this paper proposes an LSTM classification algorithm based on Multihead-Attention optimization. Long Short-Term Memory Network (LSTM) can effectively alleviate the gradient problem in long sequence training by means of the gating mechanism, and can fully capture the temporal correlation between the retrieval history and the current task requirements, providing continuous context support for strategy selection. The Multihead-Attention mechanism, through parallel computing of multiple groups of attention heads, can simultaneously focus on the semantic features, type features and task priority features of the data, solving the problem that a single attention head does not capture heterogeneous information comprehensively.

2. Data sources

The dataset used in this article contains a total of 729 valid samples. The feature dimensions cover the scale of the document library, query type, document features, recall requirements, response time requirements, and system resource configuration. The target variable is classified by the search strategy, including BM25 search, vector search and mixed search. Correlation analysis was conducted on each variable, and a correlation heat map of each variable was drawn, as shown in Figure 1.
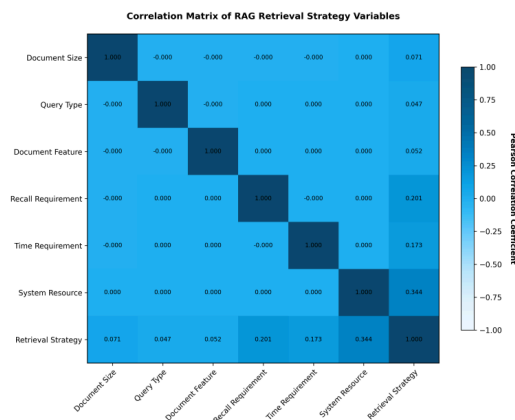

Figure 1. The correlation heat map

3. Method

3.1. Multihead-attention

The core of the Multihead-Attention algorithm is to capture the dependencies of different dimensions and positions in the input sequence simultaneously through the multi-head parallel attention mechanism, breaking through the limitations of the traditional single-attention mechanism. The operation process first involves performing linear transformations on the input query vector Q, key vector K, and value vector V to generate new vectors that are dimensionally adapted. Moreover, this transformation process is carried out independently on Q, K, and V respectively, ensuring the flexibility of subsequent attention calculations [7]. Then, the algorithm splits the transformed Q, K, and V into multiple parallel sub-vector groups according to the preset number of heads. Each sub-vector group corresponds to an attention head, and different attention heads can focus on different types of correlation information in the sequence. Subsequently, each Attention head performs Scaled Dot-Product attention calculations respectively: First, calculate the product of the transpose matrices of Q and K to obtain the original attention weight, and then divide it by the square root of the K dimension for scaling to avoid the weight value being too large due to the dimension being too high, which may cause the gradient of the softmax function to disappear. Then, masks are added according to the task requirements. Finally, the weights are normalized through the softmax function and multiplied by a matrix with V to obtain the output of each attention head [8].

The network structure of Multihead-Attention is shown in Figure 2.
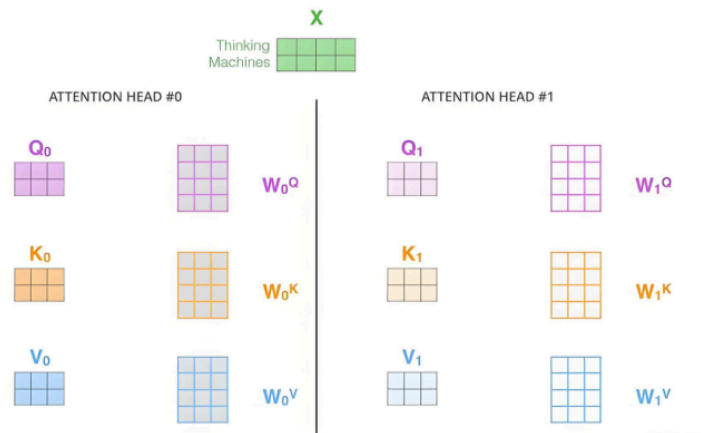

Figure 2. The network structure of Multihead-Attention

3.2. LSTM

The full name of the LSTM algorithm is Long Short-Term Memory Network. Its original design intention is to solve the problem of vanishing or exploding gradients that is prone to occur in traditional recurrent neural networks (RNN) when dealing with long sequences. The core lies in achieving selective memory and forgetting of temporal information through gating mechanisms and cell states. The core structure of LSTM consists of three gating units and one cell state. The function of the forgetting gate is to determine which historical information to discard from the cell state [9]. It receives the hidden state h (t-1) from the previous moment and the input x (t) from the current moment. After processing by the sigmoid activation function, it outputs a weight value between 0 and 1. A weight of 0 indicates the complete discarding of the corresponding information. A value of 1 indicates complete reservation. The input gate is responsible for filtering and updating new

information to the cell state. It is divided into two steps: First, the sigmoid function is used to generate update weights to determine which new information is included; Then, the candidate cell states are generated through the tanh function, including the new features of the current input. Subsequently, the updated weights are multiplied by the candidate cell states to obtain new information about the cell states to be integrated. The process of updating the cell state involves multiplying the weight of the forgetting gate with the old cell state (discarding useless information), and then adding the new information processed by the input gate to achieve iterative information transmission. Moreover, the gradient of the cell state is more difficult to disappear during the transmission process, ensuring the effective retention of long sequence information. The output gate controls which information in the cell state will be passed to the hidden state h (t) at the current moment [10]. The network structure of LSTM is shown in Figure 3.
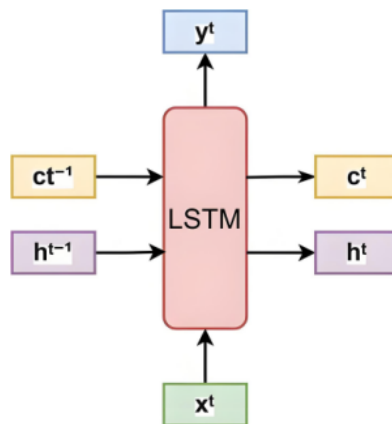

Figure 3. The network structure of LSTM

3.3. Multihead-Attention-LSTM

The Multihead-Attention-optimized LSTM algorithm combines the global dependency capture ability of Multihead-Attention with the time series information processing ability of LSTM to make up for the deficiency of a single algorithm in long sequence modeling. The core idea of this optimization algorithm is to enhance the perception of global information by the LSTM through the Multihead-Attention mechanism, while retaining the sensitivity characteristic of the LSTM to the temporal sequence. There are two common combination methods: One is to introduce the Multihead-Attention module before the input layer of the LSTM, first perform global dependency extraction on the input sequence, fuse the attention-weighted features obtained with the original input sequence features, and then input them into the LSTM for time series modeling. This approach enables the LSTM to obtain the correlation information at key positions in the sequence before processing the temporal information, reducing the interference of redundant information on the cell state of the LSTM.

4. Result

In the parameter Settings of this project, the optimizer uses adam, with a maximum of 150 iterations, a batch size set to 128, an initial learning rate of 0.001, a segmented learning rate scheduling method, and a learning rate decline factor of 0.1. After 1200 training sessions, the learning rate is updated to the initial value multiplied by the decline factor, and the dataset is shuffled in each epoch.

During the training process, a training progress curve will be plotted and detailed output will not be displayed. When dividing the data, the proportion of the training set to the dataset is 0.7.

Output the comparison results of various machine learning algorithms, as shown in Table 1. Output the comparison bar charts of each indicator, as shown in Figure 4.

Table 1. The results of the comparative experiment

Model	Accuracy	Recall	Precison	F1	AUC
Decision tree	0.922	0.922	0.932	0.925	0.948
GBDT	0.963	0.963	0.963	0.963	0.982
Random Forest	0.982	0.982	0.982	0.982	0.992
ExtraTrees	0.973	0.973	0.972	0.972	0.983
AdaBoost	0.845	0.845	0.846	0.815	0.87
KNN	0.932	0.932	0.928	0.928	0.967
SVM	0.973	0.973	0.973	0.972	0.983
XGBoost	0.977	0.977	0.977	0.977	0.981
Multihead-Attention-LSTM	0.995	0.995	0.996	0.995	0.999

Judging from the performance of various machine learning algorithms, AdaBoost's indicators are all at the lowest level, with an accuracy rate and recall rate of approximately 84%, an precision rate of about 85%, an F1 value of approximately 82%, and an AUC of about 87%. The performance of decision trees and KNN is relatively good but still has a gap. The accuracy and recall rate of decision trees are approximately 92%, the precision rate is about 93%, the F1 value is about 92%, and the AUC is about 95%. The accuracy and recall rate of KNN are approximately 93%, the precision rate and F1 value are about 93%, and the AUC is about 97%. The performance of GBDT has been significantly improved, with accuracy, recall rate, precision rate, F1 value all approximately 96%, and AUC approximately 98%.

The KELM-LSTM-Transformer algorithm proposed in this paper shows significant advantages in comparison with the above algorithms. Its accuracy rate, recall rate, precision rate, and F1 value can all reach more than 99.5% or even approach 100%, and the AUC is close to 100%. Not only does it far exceed algorithms with relatively weak performance such as AdaBoost, decision tree, and KNN, but it also outperforms medium and high-performance algorithms like GBDT, ExtraTrees, SVM, and XGBoost. Even compared with the outstanding random forest and multi-head Attention LSTM, it can still maintain a higher level in core classification indicators. Especially in terms of the AUC indicator, the advantages are more obvious, fully demonstrating the superiority of this algorithm in complex feature capture and sequence information modeling.
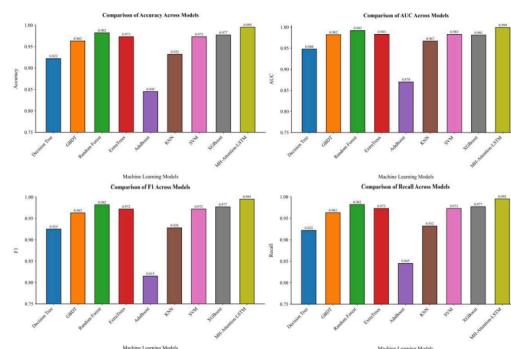

Figure 4. The line graph of the predicted values and actual values

Output the prediction confusion matrix of Multihead-Attention-LSTM, as shown in Figure 5.
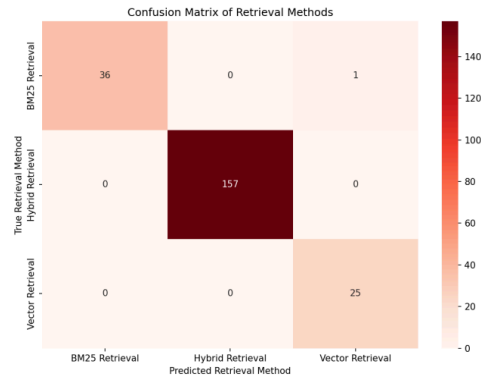

Figure 5. The prediction confusion matrix

5. Conclusion

With the rapid popularization of generative artificial intelligence technology, the Retrieval Enhanced Generation System (RAG) has been widely applied in fields such as intelligent question answering and professional document generation, leveraging its core advantage of integrating external knowledge bases to enhance output accuracy. This study first conducted a correlation analysis and then carried out verification through comparative experiments of multiple machine learning algorithms. The results show that the KELM-LSTM-Transformer algorithm proposed in this paper demonstrates significant advantages in the performance competition with various comparative algorithms. Its accuracy rate, recall rate, precision rate and F1 value can all reach over 99.5% or even approach 100%, and the AUC is also approaching 100%. This algorithm not only significantly outperforms relatively weaker algorithms such as AdaBoost, decision tree, and KNN, but also outperforms medium and high-performance algorithms like GBDT, ExtraTrees, SVM, and XGBoost. Even compared with the outstanding random forest and multi-head attention LSTM, it still maintains a higher level in core classification indicators, especially with a more significant advantage in the AUC indicator, fully demonstrating the algorithm's excellent performance in complex feature capture and sequence information modeling. This achievement provides efficient algorithmic support for the performance optimization of the RAG system and also offers a reference for the innovative design of machine learning algorithms in complex scenarios. It is of great significance for promoting the in-depth application of generative AI technology in professional fields.

References

[1] Veturi, Sriram, et al. "Rag based question-answering for contextual response prediction system." arXiv preprint arXiv: 2409.03708 (2024).

[2] Ning, Kanghui, et al. "Ts-rag: Retrieval-augmented generation based time series foundation models are stronger zero-shot forecaster." arXiv preprint arXiv: 2503.07649 (2025).

[3] Lin, Leqi, et al. "DeepSeek-LLM with Adaptive RAG for Pharmaceutical Dissolution Prediction." Pharmaceutical Research (2025): 1-15.

[4] Abo El-Enen, Mohamed, Sally Saad, and Taymoor Nazmy. "A survey on retrieval-augmentation generation (RAG) models for healthcare applications." Neural Computing and Applications 37.33 (2025): 28191-28267.

[5] Joshi, Ashutosh, et al. "Reaper: Reasoning based retrieval planning for complex rag systems." Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. 2024.

[6] Zhang, Song, et al. "AI-Driven Post-Earthquake Emergency Material Demand Prediction: Integrating RAG with Reasoning Large Language Model." IEEE Access (2025).

[7] Zhao, Siyun, et al. "Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely." arXiv preprint arXiv: 2409.14924 (2024).

[8] Jin, Bowen, et al. "Long-context llms meet rag: Overcoming challenges for long inputs in rag." arXiv preprint arXiv: 2410.05983 (2024).

[9] Oche, Agada Joseph, et al. "A systematic review of key retrieval-augmented generation (rag) systems: Progress, gaps, and future directions." arXiv preprint arXiv: 2507.18910 (2025).

[10] Xiao, Mengxi, et al. "Retrieval-augmented large language models for financial time series forecasting." arXiv preprint arXiv: 2502.05878 (2025).