

PARASHARD: Knowledge Sharding for Effective Parametric Retrieval-Augmented Generation

Changjiang Zhou^{1,2,3}, Ruqing Zhang^{1,2,3*}, Jiafeng Guo^{1,2,3*},
Yingchen Zhang^{1,2,3}, Fuyu Lv⁴, Xueqi Cheng^{1,2,3}

¹State Key Laboratory of AI Safety

²Institute of Computing Technology, Chinese Academy of Sciences

³University of Chinese Academy of Sciences

⁴Taobao & Tmall Group of Alibaba

{zhouchangjiang23s,zhangruqing,guojiafeng}@ict.ac.cn

Abstract

Retrieval-Augmented Generation (RAG) enhances large language models with external knowledge, but long retrieved contexts incur high computational cost and often degrade performance due to context-knowledge mismatch. Parametric RAG addresses this by integrating external knowledge into model parameters via lightweight adapters. However, existing PRAG methods are fundamentally *document-centric*, treating each document as an independent unit of knowledge, which leads to poor scalability and weak supervision. We argue that knowledge is inherently distributed across semantically related documents, and document-level parameterization is a suboptimal granularity. To this end, we propose PARASHARD, a *knowledge-centric* framework that organizes knowledge into semantically coherent knowledge shards and parameterizes each shard with a shared adapter. This design enables richer supervision across related documents while significantly reducing the number of required adapters. At inference time, PARASHARD dynamically selects and composes multiple shards using a diversity-aware strategy to ensure comprehensive knowledge coverage. Experiments on four knowledge-intensive QA benchmarks show that PARASHARD achieves superior average performance across diverse knowledge-intensive tasks compared with strong RAG and PRAG baselines, with particularly strong gains on multi-hop reasoning, and substantially reduces storage cost compared to PRAG, demonstrating the knowledge injection effectiveness and storage efficiency of knowledge-centric parameterization.

1 Introduction

Large language models (LLMs) have achieved remarkable success across a wide range of tasks (Touvron et al., 2023; OpenAI et al., 2024), yet their parametric knowledge remains static after training,

limiting their ability to incorporate new or domain-specific information. This limitation is particularly critical for knowledge-intensive tasks such as open-domain question answering.

Retrieval-augmented Generation (RAG) addresses this issue by injecting retrieved documents into the input context, enabling models to access external knowledge on demand (Lewis et al., 2021; Gao et al., 2024). However, this in-context paradigm suffers from two key limitations: (i) long contexts introduce substantial computational overhead, and (ii) models often fail to effectively utilize noisy or lengthy evidence, leading to degraded reasoning performance (Liu et al., 2024).

To mitigate these issues, Parametric Retrieval-Augmented Generation (PRAG) (Su et al., 2025a) has emerged as an alternative paradigm. Specifically, PRAG integrates external knowledge directly into model parameters via lightweight adapters (e.g., LoRA), eliminating the need for long contexts (Su et al., 2025a). Despite its promise, existing PRAG methods adopt a fundamentally *document-centric* design, treating each document as an independent unit of knowledge. This design leads to two critical issues: (i) *Poor scalability*: one adapter per document or a single adapter for the entire corpus results in prohibitively high storage and training costs for large knowledge bases (Su et al., 2025b), and limits the ability to selectively activate relevant knowledge. (ii) *Limited supervision*: training each adapter on a single document provides a weak signal, leading to overfitting and failing to leverage shared knowledge across related documents, which causes redundant and inefficient knowledge encoding.

More fundamentally, these limitations stem from a *mismatch between documents and knowledge*: knowledge is rarely contained within a single document, but instead distributed across multiple semantically related pieces of evidence. This motivates a key question: *what is the appropriate granularity*

*Corresponding authors

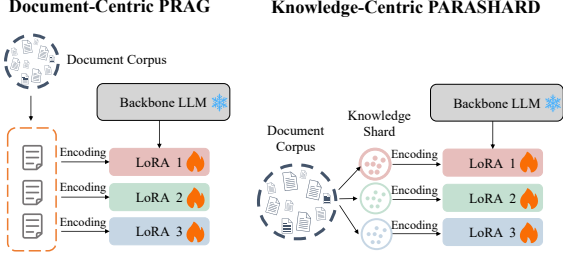


Figure 1: Comparison of document-centric PRAG and knowledge-centric PARASHARD.

for parameterizing knowledge?

In this paper, we propose PARASHARD, a *knowledge-centric* framework that shifts parameterization from documents to semantically coherent *knowledge shards*. As shown in Figure 1, this design enables richer supervision across documents, reduces redundant adapters, and scales efficiently to large knowledge bases. We first define the concept of “knowledge” within the scope of our work from both macro-structural and micro-representational perspectives. Then, we propose a knowledge parameterization pipeline via knowledge organization via clustering, knowledge representation via triples and knowledge injection via training. Since PRAG is conducted at the knowledge granularity, we then perform document-agnostic shard routing to retrieve relevant knowledge shards for the input query. During inference, PARASHARD dynamically composes multiple shards based on the query’s semantic complexity. Inspired by the maximal marginal relevance (MMR) theory in fairness learning to rank (Xia et al., 2015), we incorporate diversity-aware multi-shard selection to balance relevance and coverage, ensuring comprehensive knowledge integration.

Experiments on four knowledge-intensive QA benchmarks demonstrate that PARASHARD can achieve superior average performance against standard and parametric RAG baselines at distinct model scales while significantly reducing storage requirements compared to PRAG. Specifically, PARASHARD outperforms PRAG by 22.60%, 9.69%, and 3.01% average F1 scores on LLaMA 3.2-1B, Qwen2.5-1.5B, and LLaMA3-8B, respectively. The datasets and code are available at <https://github.com/Sherlock-coder/ParaShard>.

2 Preliminaries: PRAG

PRAG (Su et al., 2025a) treats individual documents as the fundamental unit for organizing and

parameterizing external knowledge.

Document augmentation. PRAG adopts a document augmentation strategy consisting of two key steps: (i) document rewriting and (ii) QA pair generation. Specifically, each document d_i is transformed into multiple rewritten documents $\{d_i^1, d_i^2, \dots, d_i^n\}$ by prompting an LLM. Additionally, the LLM generates question-answer (QA) pairs based on the original document d_i , producing a set of questions and their corresponding answers: $\{(q_i^1, a_i^1), (q_i^2, a_i^2), \dots, (q_i^m, a_i^m)\}$, where m is a tunable hyperparameter representing the number of QA pairs per document. Integrating multiple rewrites with corresponding QA pairs transforms each document d_i into a more comprehensive training dataset $\mathcal{T}_i$:

$$\mathcal{T}_i = \{(d_i^k, q_i^j, a_i^j) \mid 1 \leq k \leq n, 1 \leq j \leq m\}, \quad (1)$$

where each (d_i^k, q_i^j, a_i^j) triple corresponds to a rewritten document d_i^k from the original document d_i , coupled with a question q_i^j and its respective answer a_i^j .

Document parameterization. Let the pretrained LLM be parameterized by θ and the external corpus be $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$. In PRAG, each document $d_i \in \mathcal{D}$ is independently mapped into a parametric representation $p_i = f_\phi(d_i)$, where $f_\phi(\cdot)$ is a document parameterization function. In practice, p_i is implemented as a Low-Rank Adaptation (LoRA) module (Hu et al., 2021) that injects low-rank updates into the LLM parameters, typically applied to feed-forward network (FFN) weights. For an FFN weight matrix $W \in \mathbb{R}^{h \times k}$, the document-specific update is:

$$W'_i = W + \Delta W_i, \quad \text{where} \quad \Delta W_i = A_i B_i^\top, \quad (2)$$

with $A_i \in \mathbb{R}^{h \times r}$, $B_i \in \mathbb{R}^{k \times r}$, and $r \ll \min(h, k)$. The original parameters W remain frozen, and only the low-rank matrices $\Delta\theta_i = \{A_i, B_i\}$ are trainable. The LoRA parameters $\Delta\theta_i$ are then optimized by minimizing the language modeling loss over $\mathcal{T}_i$:

$$\mathcal{L}_{\text{lm}}(\Delta\theta_i) = \mathbb{E}_{x \sim \mathcal{T}_i} \left[\sum_{t=1}^{|x|} -\log P_{\theta + \Delta\theta_i}(x_t \mid x_{<t}) \right]. \quad (3)$$

3 Our Method

We propose PARASHARD, a shard-centric framework for storage-efficient and coherent knowledge integration in LLMs. As illustrated in Figure 2,

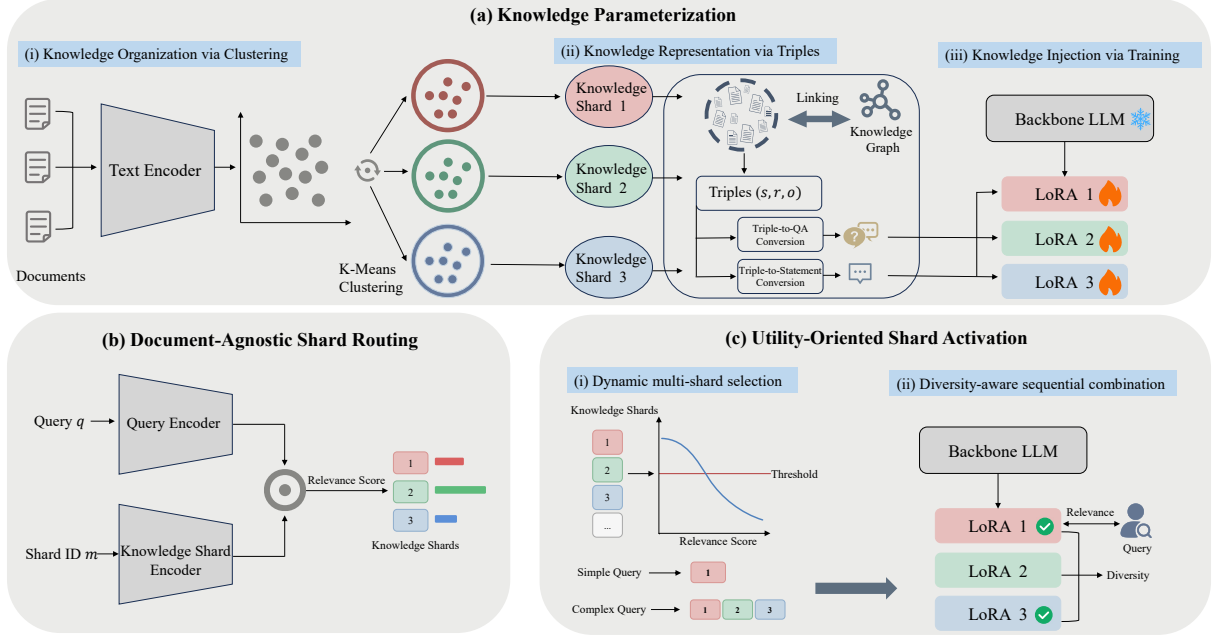


Figure 2: Framework of PARASHARD. (a) illustrates knowledge parameterization, where documents are clustered, represented as QA pairs and statements, and injected into parameter-efficient LoRA modules. (b) details the dual-tower document-agnostic shard router, (c) depicts the mechanism of utility-oriented shard activation.

PARASHARD first organizes large-scale corpora into semantically coherent knowledge shards and parameterizes each shard as a dedicated LoRA module (Section 3.1). It then trains a document-agnostic shard router to retrieve relevant shards for a given query (Section 3.2). Finally, at inference time, a utility-oriented activation mechanism adaptively selects and sequentially composes multiple LoRA modules to balance coverage and redundancy (Section 3.3). All operations are performed at the shard level, avoiding document-level reasoning and enabling scalable knowledge composition.

3.1 Knowledge parameterization

Knowledge definition. Within our proposed framework, we conceptualize "knowledge" not as a monolithic, unstructured corpus of text, but through a dual theoretical lens that directly governs its organization and representation. (i) From a macro-structural perspective, knowledge is characterized by *semantic modularity*. Facts and concepts naturally form cohesive, domain-specific clusters, where related entities and relations are densely interconnected (Newman, 2006; Ji et al., 2021). (ii) From a micro-representational standpoint, knowledge is defined by its *relational atomicity*. Consistent with foundational knowledge graph theory (Hogan et al., 2021; Chen et al., 2020), knowledge can be distilled into discrete subject-relation-object triples (s, r, o) , capturing fundamental factual interactions. These

two theoretical properties directly motivate our pipeline design: *semantic modularity* corresponds to knowledge organization via K-means clustering, which groups semantically related documents into coherent shards; *relational atomicity* corresponds to knowledge representation via triples, which distills knowledge into atomic (s, r, o) triples for focused parameterization.

Knowledge organization via clustering. The inherent semantic modularity dictates that knowledge should not be partitioned arbitrarily, but rather organized into semantically cohesive bounds. Hence, we organize documents into *knowledge shards* based on semantic similarity rather than arbitrary partitioning. Concretely, let $\mathcal{D} = \{d_1, \dots, d_N\}$ denote the full document collection. We first encode each document d_i into a dense vector representation using a pretrained text encoder E . We then apply K-means clustering to partition the document embeddings into M clusters, where M is a hyperparameter controlling the granularity of knowledge organization. All documents assigned to the same cluster form a knowledge shard $\mathcal{D}_m$:

$$\mathcal{D}_m = \{d_i \mid \text{cluster}(d_i) = m\}, \quad m \in \{1, \dots, M\}. \quad (4)$$

Each knowledge shard $\mathcal{D}_m$ is associated with a dedicated LoRA module L_m with trainable parameters ϕ_m . During training, the backbone model parameters θ are frozen while the LoRA parameters ϕ_m are optimized to encode the knowledge within

the corresponding shard. At inference time, one or more LoRA module are activated based on the input query in a plug-and-play manner, ensuring low computational overhead. Importantly, in the subsequent PARASHARD pipeline, the notion of individual documents is entirely abstracted away, i.e., all operations are performed at the knowledge shard granularity, enabling more coherent knowledge injection and efficient retrieval.

Knowledge representation via triples. Each knowledge shard $\mathcal{D}_m$ is represented as a set of atomic triples (s, r, o) , capturing fundamental factual interactions. For each shard, triples are produced via distant supervision by linking to an authoritative knowledge graph (Mintz et al., 2009), ensuring high-quality factual grounding.

To enrich the training data, we apply two complementary post-processing transformations to each extracted triple: (i) **Triple-to-QA Conversion:** We convert each triple into c question-answer pairs (q, a) by generating relation-specific question templates using an LLM. The question template is conditioned on the predicate r , ensuring that the generated questions are semantically aligned with the underlying relation. (ii) **Triple-to-Statement Conversion:** Similarly, we transform each triple into c declarative factual statements s with relation-specific statement templates produced by an LLM, providing an alternative representation that preserves the original semantic content in natural language form. Hence, the knowledge representation $\mathcal{T}_m$ corresponding to $\mathcal{D}_m$ is defined as follows:

$$\mathcal{T}_m = \{(q \oplus a)\} \cup \{s\}, \quad \forall (q, a), s \in \mathcal{D}_m. \quad (5)$$

Knowledge injection via LoRA training. To inject knowledge representation $\mathcal{T}_m$ into the parameterized module, we employ the same optimization objective as in Equation 3 to train a dedicated LoRA module L_m with $\mathcal{T}_m$ while keeping the backbone LLM parameters frozen. Given the one-to-one correspondence between the semantic clusters and the LoRA modules, we use the term “knowledge shard” hereafter to refer to the knowledge-parameterized LoRA modules as well.

3.2 Document-agnostic shard routing

To dynamically select the most relevant knowledge shards for a given query, we formulate the routing problem as dense retrieval over knowledge shard identifiers. We train a dual-tower dense retriever that maps a natural language query to knowledge

shards corresponding to the most relevant knowledge shards.

Specifically, we employ two Transformer-based encoders: a query encoder $f_q(\cdot)$ and a knowledge shard encoder $f_l(\cdot)$. Given a query q , the query encoder produces a dense representation:

$$\mathbf{q} = f_q(q) \in \mathbb{R}^d. \quad (6)$$

Each knowledge shard L_m is associated with a unique identifier m representing its corresponding knowledge shard $\mathcal{D}_m$. The knowledge shard encoder maps the shard to an embedding:

$$\mathbf{l}_m = f_l(m) \in \mathbb{R}^d. \quad (7)$$

The relevance between a query and a knowledge shard is computed via inner product similarity:

$$s(q, m) = \mathbf{q}^\top \mathbf{l}_m. \quad (8)$$

To construct training data for the router, we leverage a document-level retriever R (i.e., BM25). For a given query q , we retrieve a set of top- r relevant documents $\mathcal{R}(q)$. A knowledge shard L_{m^+} is treated as a positive sample if its associated knowledge shard contains at least one retrieved document:

$$m^+ \in \{m \mid \mathcal{D}_m \cap \mathcal{R}(q) \neq \emptyset\}. \quad (9)$$

Negative knowledge LoRAs m^- are sampled via in-batch negative sampling for simplicity. The shard router is trained exclusively on queries from the training splits of benchmark datasets, completely separate from evaluation questions.

The router is trained using a contrastive objective. Given a query q , a positive knowledge LoRA m^+ , and negative samples m^- , the routing loss is:

$$\mathcal{L}_r = -\mathbb{E}_{(q, m^+)} \log \frac{\exp(s(q, m^+)/\tau)}{\exp(s(q, m^+)/\tau) + \sum_{m^-} \exp(s(q, m^-)/\tau)}, \quad (10)$$

where τ is a temperature hyperparameter. Note that in the shard routing phase, documents are agnostic while only the knowledge shards are visible.

3.3 Utility-oriented shard activation

At inference time, PARASHARD dynamically retrieves and activates knowledge shards based on the input query. We introduce two key mechanisms: Dynamic multi-shard selection and diversity-aware sequential combination.

Dynamic multi-shard selection. Empirically, restricting retrieval to a fixed top- k bottlenecks the

utility of knowledge shards, as complex queries often span multiple semantic clusters while simple queries may only incorporate a single cluster. To optimize retrieval utility, we propose an adaptive thresholding strategy that dynamically determines the optimal number of activated shards. For a given query q , we filter the relevance scores $s(q, m)$ using a confidence-based threshold to form an optimal candidate set:

$$\mathcal{M}^* = \{m \mid s(q, m) > \alpha \cdot \max_{m'} s(q, m')\}, \quad (11)$$

where $\alpha \in (0, 1)$ controls selectivity. This allows the active set size $|\mathcal{M}^*|$ (typically 1–3) to adapt to the query’s semantic complexity.

Diversity-aware sequential combination. Activating all candidates in $\mathcal{M}^*$ simultaneously often introduces redundant information, diminishing overall utility. We therefore frame the multi-shard combination as a utility-maximization problem. Drawing on maximal marginal relevance (MMR) (Xia et al., 2015), we sequentially select shards that maximize information gain while penalizing redundancy. Given the initial shard L_{m_1} with the highest relevance, subsequent shards are selected by maximizing their marginal utility:

$$m_{i+1} = \arg \max_{m \in \mathcal{M}^* \setminus \mathcal{M}_i} \left[\lambda \cdot s(q, m) - (1 - \lambda) \cdot \max_{m_j \in \mathcal{M}_i} s(L_{m_i}, L_{m_j}) \right], \quad (12)$$

where $\mathcal{M}_i = \{m_1, \dots, m_i\}$ denotes the previously selected LoRAs, $s(\cdot, \cdot)$ measures the inner product similarity, and λ balances relevance against diversity to ensure broader knowledge coverage.

To fully exploit the ordered sequence of $K = |\mathcal{M}^*|$ selected LoRAs, we activate them sequentially rather than performing parallel merging. Let $C_0 = \emptyset$ be the initial context. At each step $t \in [1, K]$, we activate the corresponding module L_{m_t} to generate intermediate background knowledge c_t :

$$c_t = \text{LLM}_{\theta+\phi_{m_t}}(q \oplus C_{t-1}), \quad (13)$$

where $C_{t-1} = c_1 \oplus \dots \oplus c_{t-1}$ denotes the accumulated context from previous activations. The final response is produced at step K . This autoregressive accumulation allows each newly activated shard to build on prior context, improving the complementary use of knowledge across shards.

4 Experimental Settings

Datasets and evaluation metric. We adopt four knowledge-intensive QA datasets: (i) 2Wiki-

MultiHopQA (2WQA) (Ho et al., 2020) that includes four reasoning types, i.e., Compare, Bridge, Inference and Compose; (ii) HotpotQA (HQA) (Yang et al., 2018); (iii) PopQA (PQA) (Mallen et al., 2023); (iv) ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). Following previous works (Su et al., 2025a; Tan et al., 2025; Su et al., 2025b), the first 300 questions from each sub-dataset are used, and we employ F1 scores (%) as the evaluation metric.

Baseline methods and implementation details.

We compare PARASHARD with five baselines: Vanilla, Standard RAG, PRAG, DyPRAG, and Poly-PRAG (see Appendix A). Following Su et al. (2025a); Tan et al. (2025); Su et al. (2025b), we evaluate models using three backbone LLMs of different scales: LLaMA3.2-1B-Instruct¹ (Grattafiori et al., 2024), Qwen2.5-1.5B-Instruct² (Yang et al., 2025), and LLaMA3-8B-Instruct³ (Grattafiori et al., 2024). During the knowledge organization phase, the text encoder E used for document representation is Sentence-T5-base⁴ (Ni et al., 2022), and we set the hyperparameter M to 2101 to ensure that the average number of documents contained in each cluster is approximately 10,000. In the knowledge representation stage, we adopt the external knowledge graph⁵ proposed by Hu et al. (2017), which contains 198,969,771 relational triples. c is set to 10 for each relational triple. All the experiments are performed on 32 NVIDIA 80G GPUs. We repeat our experiment 3 times to get the average results. In the LoRA training stage, the training batch size per device is set to 32, the learning rate is set to $5e-5$, and the max length is set to 512. For each LoRA module, rank is set to 16 and alpha is set to 32. We inject the LoRA adapter to both the self-attention layers and the feed-forward networks following Su et al. (2025b). The training for all 2101 LoRA modules take approximately 24 hours. For shard router training, we use Sentence-T5-base (Ni et al., 2022) as the query encoder $f_q(\cdot)$ and the knowledge shard encoder $f_l(\cdot)$, the document-level retriever R is BM25 (Robertson and Zaragoza, 2009) implemented by elasticsearch⁶, r is set to 3, the

¹<https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct>

²<https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct>

³<https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

⁴<https://huggingface.co/sentence-transformers/sentence-t5-base>

⁵<https://github.com/pkumod/gAnswer>

⁶<https://github.com/elastic/elasticsearch>

temperature τ is set to 1, the training batch size per device is set to 32, the learning rate is set to $2e-5$, and the max length is set to 100. The training for shard router takes approximately 2 hours. During the model inference phase, α is set to 0.5, and λ is set to 0.8.

5 Experimental Results

We aim to address four key questions: (RQ1) Does shard-level knowledge parameterization improve QA performance compared with vanilla LLMs, standard RAG, and PRAG-based methods? (RQ2) How do key components (knowledge representation, shard routing, and shard combination) affect performance? (RQ3) How does shard capacity (LoRA rank) influence the balance between accuracy and storage efficiency? (RQ4) How does the number of knowledge shards M affect clustering quality and downstream QA performance?

5.1 Main results (RQ1)

As shown in Table 1, PARASHARD achieves the best average performance across datasets and model scales, demonstrating its effectiveness and robustness for knowledge-intensive reasoning.

Comparison to vanilla LLMs. PARASHARD consistently outperforms vanilla LLMs on all benchmarks, confirming that shard-level parameterization effectively injects external knowledge into frozen models and alleviates the inherent knowledge bottleneck.

Comparison to standard RAG. Compared to standard RAG, PARASHARD achieves absolute gains of 5.64, 7.58, and 11.61 F1 points on LLaMA3.2-1B, Qwen2.5-1.5B, and LLaMA3-8B, respectively. These gains may stem from internalizing knowledge into model parameters, which reduces reliance on lengthy retrieved contexts and mitigates long-context interference. An exception appears on the Bridge subset of HotpotQA with smaller models (LLaMA3.2-1B and Qwen2.5-1.5B), where standard RAG performs better (e.g., 21.38 vs. 15.65). We attribute this to the inherent difficulty of bridge-type multi-hop reasoning: small-capacity models struggle to encode cross-document dependencies into LoRA parameters, while standard RAG exposes all evidence sequentially, allowing even smaller models to perform shallow pattern matching across documents.

Comparison to PRAG variants. Compared with PRAG variants (PRAG, DyPRAG, Poly-PRAG),

PARASHARD achieves stronger overall performance, particularly on multi-hop reasoning tasks such as 2WQA and HQA-Compare. The gains stem from shard-level clustering, which builds semantically coherent knowledge units, and diversity-aware sequential activation, which improves coverage across multiple evidence sources. While Poly-PRAG performs better on specific bridge-type reasoning tasks, ParaShard demonstrates more balanced and robust performance across task types. However, PARASHARD does not outperform all baselines in every case. On HQA-Bridge, Poly-PRAG maintains a slight advantage across model scales due to the need for precise sequential evidence chaining, which can be partially disrupted when interdependent facts are distributed across multiple shards. Similarly, on PopQA with LLaMA3.2-1B and LLaMA3-8B, Poly-PRAG slightly outperforms PARASHARD because entity ambiguity causes passages with the same entity name to spread across multiple shards, leading to interference and reduced factual precision.

5.2 Ablation study (RQ2)

Effect of Knowledge Injection. We assess the impact of different knowledge representation strategies using four variants: (i) QA only, using only triple-to-QA conversion; (ii) Statement only, using only triple-to-statement conversion; (iii) Original data, a document-centric approach with QA pair generation; (iv) Extra document, augmenting the dual representation with raw document text. Table 2 shows that: (i) *Original data* significantly degrades performance, especially on PQA and CWQ, due to mismatch between document-level inputs and shard-level parameterization, where each shard aggregates multiple documents and raw text introduces unnecessary noise. (ii) *QA only* and *statement only* both underperform PARASHARD in terms of average F1, showing that neither representation is sufficient alone: QA captures interrogative structure for recall, while statements provide complementary factual grounding. (iii) *Extra documents* further hurt performance, indicating that unstructured text introduces noise rather than useful signal. This confirms that triple-based abstraction yields more compact and effective knowledge for parameterization.

Effect of knowledge shard routing. Removing the learned router and randomly selecting shards (w/o router) leads to a substantial performance drop across all datasets. The results demonstrate

Base LLM	Method	2WQA				HQA		PQA	CWQ	Avg.
		Compare	Bridge	Inference	Compose	Bridge	Compare			
LLaMA3.2-1B	Vanilla	42.89	24.17	16.91	7.87	13.25	40.26	2.26	34.94	22.82
	Standard RAG	41.23	26.78	22.51	10.21	21.38	42.46	17.65	37.39	27.45
	PRAG	50.20	24.34	19.11	8.24	13.65	40.90	23.58	35.86	26.99
	DyPRAG	51.25	48.15	17.35	7.54	14.05	43.90	11.33	36.86	28.80
	Poly-PRAG	53.24	49.23	23.11	12.45	16.78	44.30	24.70	37.60	32.68
	PARASHARD	53.31	52.66[†]	23.14	13.33[†]	15.65	44.99[†]	23.94	37.71	33.09[†]
Qwen2.5-1.5B	Vanilla	45.74	39.06	17.04	7.27	12.18	39.46	2.87	26.47	23.76
	Standard RAG	38.75	38.84	11.87	5.68	16.19	37.13	9.97	27.33	23.33
	PRAG	44.96	43.96	19.29	11.14	13.27	40.42	21.55	30.82	28.18
	DyPRAG	43.03	47.20	17.04	8.55	13.72	41.39	6.64	31.94	26.19
	Poly-PRAG	46.74	48.74	20.04	12.03	15.18	42.46	24.03	32.85	30.26
	PARASHARD	47.40[†]	49.38[†]	20.78[†]	12.64[†]	14.73	43.01[†]	24.97[†]	34.34[†]	30.91[†]
LLaMA3-8B	Vanilla	54.90	55.20	24.59	14.43	19.00	45.63	7.96	42.44	33.02
	Standard RAG	58.43	47.77	19.20	11.07	19.68	42.10	16.13	35.45	31.23
	PRAG	57.78	58.93	27.61	19.17	33.68	65.88	26.13	43.54	41.59
	DyPRAG	57.39	56.43	25.33	18.88	24.85	58.89	13.60	41.87	37.16
	Poly-PRAG	58.39	59.70	28.64	20.03	34.03	67.04	28.20	45.37	42.68
	PARASHARD	59.95[†]	59.70	29.31[†]	20.92[†]	32.09	68.13[†]	26.91	45.67[†]	42.84

Table 1: The experimental results (F1 scores %) of PARASHARD on four knowledge-intensive tasks. The best performance is bolded. The Avg is the average score for each subtask. The results for all baseline methods are taken from (Su et al., 2025b). [†] denotes significant improvement with t-test at $p < 0.05$ over Poly-PRAG.

Method	2WQA				HQA		PQA	CWQ	Avg.
	Compare	Bridge	Inference	Compose	Bridge	Compare			
PARASHARD	53.31	52.66	23.14	13.33	15.65	44.99	23.94	37.71	33.09
original data	46.33	48.03	23.81	8.11	13.78	43.00	9.15	30.79	27.88
QA only	46.09	45.66	21.95	11.13	15.25	41.69	6.03	38.21	28.25
statement only	59.12	49.45	24.42	11.80	10.95	44.57	8.62	35.06	30.50
extra document	50.91	49.58	27.15	9.27	11.19	39.82	8.88	35.34	29.02
w/o router	31.72	33.27	16.88	4.69	9.42	33.40	5.91	26.27	20.20
top-1 shard	47.39	50.06	22.72	8.01	12.37	40.85	10.34	35.03	28.34
sequential top-2	48.51	51.96	22.00	10.74	14.98	39.88	21.54	32.57	30.27
sequential top-3	48.51	51.96	21.64	10.74	13.52	39.09	21.54	32.96	30.00
merge top-2	50.85	43.41	23.74	9.07	13.66	43.15	2.92	37.69	28.06
merge top-3	50.85	43.41	23.74	9.07	13.76	42.78	2.92	37.69	28.03

Table 2: Empirical results of ablation study. All reported results are obtained using LLaMA 3.2-1B as the backbone model. The evaluation metric is the F1 score.

that random shard selection severely impairs the model’s ability to access relevant knowledge, leading to consistently inferior performance compared to PARASHARD. These findings underscore the critical role of the contrastive learning-based router in dynamically identifying and activating semantically relevant knowledge shards, thereby enabling effective knowledge retrieval and utilization.

Effect of shard Combination. We analyze different strategies for combining multiple shards: (i) Top-1, activating only the highest-ranked shard; (ii) Sequential Top-2/Top-3, sequentially activating the top 2 or 3 shards, feeding each subsequent shard the query plus accumulated context; (iii) Merge

Top-2/Top-3, linearly combining top shards’ parameters. Key observations can be concluded as follows: (i) Activating only the *Top-1* shard enables the model to inject relevant knowledge and achieve reasonable question-answering capability since the required knowledge is included in a single shard for some queries. (ii) *Sequential activation* outperforms parameter merging in terms of average F1, as the latter requires explicit training for parameter composition. The gap is particularly pronounced on PopQA, where parameter merging causes destructive interference between entity-specific knowledge representations, while sequential activation preserves the integrity of each shard’s learned knowledge. (iii) Our *diversity-inspired*

Rank	Parameters (LoRA)	Proportion (%)	Storage (MB)	F1 Score (%)
4	2.82M	0.23	10.78	31.06
8	5.64M	0.45	21.53	31.65
16	11.27M	0.90	43.03	33.09
32	22.54M	1.79	86.03	33.43

Table 3: Shard capacity analysis. The LoRA hyperparameter α is set to twice the value of the rank. All results are obtained using LLaMA 3.2-1B as the backbone model. Storage refers to the disk storage of each LoRA module.

shard combination mechanism achieves superior performance compared to all fixed top-k strategies. This advantage stems from its explicit consideration of marginal utility: while fixed top-k methods solely rely on query-shard relevance scores, PARASHARD’s adaptive approach balances relevance with diversity, preventing redundant knowledge activation and ensuring complementary information coverage across selected shards.

5.3 Impact of shard capacity (RQ3)

To study the effect of shard capacity, we vary the LoRA rank, which controls the representational capacity allocated to each knowledge shard adapter. Larger ranks increase expressiveness but also introduce higher storage. As shown in Table 3, performance exhibits a clear monotonic trend: low-rank adapters are under-expressive while high-rank adapters exhibit relatively stronger performance. We finally set the rank to 16 to achieve a trade-off between effectiveness and storage efficiency.

Importantly, the finally chosen rank 16 differs from PRAG-style settings (e.g., rank 2) because each PARASHARD LoRA encodes a knowledge shard containing thousands of documents, rather than a single document. This increases intra-shard diversity and requires higher capacity to jointly represent heterogeneous yet related facts. In contrast, PRAG adapters operate at document level with much more localized content, thus requiring smaller ranks.

Despite this increased capacity requirement, PARASHARD remains highly parameter-efficient: rank 16 achieves relatively high performance (33.09%) while using only 0.9% of backbone parameters and 43.03MB storage. This demonstrates that shard-level parameterization achieves a favorable trade-off between effectiveness and storage efficiency, avoiding both per-document adapter explosion in PRAG and long-context overhead in RAG.

M	WCSS ($\downarrow$)	BCSS ($\uparrow$)	TSS	BCSS/TSS ($\uparrow$)
210	4,215,734	1,553,003	5,768,737	0.2692
700	3,811,120	1,957,617	5,768,737	0.3393
2101	3,456,308	2,312,429	5,768,737	0.4009

Table 4: Clustering quality under different numbers of knowledge shards M

5.4 Scalability of shard granularity (RQ4)

We study the effect of shard granularity by varying the number of knowledge shards $M \in \{210, 700, 2101\}$, corresponding to average shard sizes of approximately 100K, 30K, and 10K documents.

Cluster quality. We evaluate clustering quality using standard metrics: WCSS, BCSS, TSS, and BCSS/TSS. See Appendix B for the metric details. As shown in Table 4, increasing the number of shards M consistently improves clustering quality: WCSS decreases while BCSS and BCSS/TSS increase monotonically. This indicates that finer-grained partitioning yields more compact and better-separated semantic clusters. In other words, increasing M does not merely split the corpus, but produces progressively more structured and semantically coherent knowledge shards, which provides a stronger foundation for downstream shard-level parameterization.

Effect on QA performance. We further evaluate the downstream QA performance across the three M configurations using LLaMA 3.2-1B as the backbone model. As shown in Table 5, finer-grained sharding (larger M) consistently improves average performance, with average F1 score increasing monotonically. When average shard size decreases from $\sim 100K$ to $\sim 10K$ documents, each shard becomes more semantically coherent, enabling its LoRA adapter to learn more focused knowledge representations. This monotonic improvement directly aligns with the clustering quality metrics in Table 4, validating that semantically coherent shards yield more focused LoRA representations.

6 Related Work

RAG enhances LLMs by retrieving relevant documents from external corpora and incorporating them into the input context (Lewis et al., 2021; Guu et al., 2020). Subsequent work has explored various aspects of the RAG pipeline, including retrieval timing (Jiang et al., 2023), document selection (Yan et al., 2024; Hei et al., 2024), and

M	Avg Size	2WQA				HQA		PQA	CWQ	Avg.
		Comp.	Bri.	Inf.	Cps.	Bri.	Comp.			
210	100,073	35.18	46.73	15.72	4.98	4.87	36.57	10.75	20.02	21.85
700	30,022	43.89	55.05	16.52	8.85	6.55	40.67	13.16	22.84	25.94
2101	10,003	53.31	52.66	23.14	13.33	15.65	44.99	23.94	37.71	33.09

Table 5: Effect of the number of knowledge shards M on QA performance (F1 %).

knowledge organization (Asai et al., 2024). Despite these advances, in-context injection inevitably increases context length, introducing computational overhead and degrading performance on complex reasoning tasks (Liu et al., 2024).

To address the limitations of in-context injection, PRAG integrates external knowledge directly into LLM parameters via LoRA adapters (Su et al., 2025a). However, the one-to-one document encoding scheme leads to prohibitive storage and training costs. Poly-PRAG (Su et al., 2025b) mitigates this by using a small set of shared LoRA adapters with a latent routing function, reducing storage while capturing common knowledge across documents. DyPRAG (Tan et al., 2025) further improves efficiency by training a lightweight parameter translator to dynamically convert documents into parametric knowledge at test-time, eliminating per-document training.

In contrast, our approach shifts the parameterization from individual documents to semantically coherent knowledge shards, aggregating related information into unified LoRA modules. This design provides richer training signals, more storage-efficient knowledge injection, and better supports complex reasoning that requires integrating evidence across multiple sources.

7 Conclusion

We introduce PARASHARD, a knowledge-centric parametric retrieval-augmented generation framework that shifts knowledge parameterization from the document level to the shard level. By organizing external knowledge into semantically coherent knowledge shards and encoding them with dedicated LoRA adapters, PARASHARD effectively integrates complementary evidence across multiple semantic sources, enhancing multi-hop reasoning and factual accuracy. This design addresses the key limitations of prior document-centric PRAG methods, namely, poor scalability and limited supervision. Experiments on multiple knowledge-intensive QA benchmarks show that PARASHARD achieves superior average performance across diverse knowledge-intensive tasks, with particularly

strong gains on multi-hop reasoning. While Poly-PRAG performs better on specific bridge-type reasoning tasks, ParaShard demonstrates more balanced and robust performance across task types. Moreover, PARASHARD exhibits superior storage efficiency compared to PRAG.

Limitations

PARASHARD has several limitations: (i) **Scale of ablation studies.** Due to the computational cost of training multiple LoRA adapters, all ablation experiments were conducted only on 1B-parameter models. The behavior of PARASHARD on larger-scale LLMs (e.g., more than 8B) remains unexplored. While our main results on LLaMA3-8B demonstrate that the approach scales to larger models, conducting full ablation studies at that scale is left for future work. (ii) **Knowledge updating.** Corpus updates currently require re-clustering and retraining the affected LoRA modules. A more efficient alternative is to assign new documents to the nearest existing cluster and fine-tune only its LoRA module. However, this unvalidated strategy may accumulate drift, necessitating periodic full re-clustering. (iii) **Inference latency.** The sequential multi-shard activation strategy introduces higher per-query latency compared to Standard RAG, due to multiple forward passes. While this is a trade-off for improved knowledge coverage, parallel shard decoding or cached intermediate representations may reduce this overhead in future work.

Ethical Considerations

In this paper, all the models and datasets used in our experiment are publicly released. The methods we propose aim to enhance the generation reliability of LLMs and do not encourage or induce the model to produce any harmful information or leakage of user privacy. Therefore, we believe that our research work meets the ethics of ACL.

Acknowledgements

This work was funded by the National Key Research and Development Program of China under Grants No. 2025ZD0123301, the Strategic Priority Research Program of the CAS under Grants No. XDB0680102, and the National Natural Science Foundation of China (NSFC) under Grants No. 62472408, U25B2076 and 62441229, and All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.
- Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey](#). *Preprint*, arXiv:2312.10997.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.
- Zijian Hei, Weiling Liu, Wenjie Ou, Juyi Qiao, Junming Jiao, Guowen Song, Ting Tian, and Yi Lin. 2024. Dr-rag: Applying dynamic document relevance to retrieval-augmented generation for question-answering. *arXiv preprint arXiv:2406.07348*.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.
- Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D’amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. [Knowledge graphs](#). *ACM Computing Surveys*, 54(4):1–37.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#). *Preprint*, arXiv:2106.09685.
- Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2017. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.
- Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.
- Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7969–7992.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). *Preprint*, arXiv:2005.11401.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 9802–9822.
- Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.
- Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582.
- Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the association for computational linguistics: ACL 2022*, pages 1864–1874.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Found. Trends Inf. Retr.*, 3(4):333–389.

- Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. 2025a. Parametric retrieval augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1240–1250.
- Zhan Su, Fengran Mo, Jinghan Zhang, Yuchen Hui, Jiaao Sun, and Jian-yun Nie. 2025b. Parametric retrieval-augmented generation using latent routing of lora adapters. *arXiv preprint arXiv:2511.17044*.
- Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.
- Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. 2025. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *arXiv preprint arXiv:2503.23895*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. *Llama: Open and efficient foundation language models*. *Preprint*, arXiv:2302.13971.
- Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2015. Learning maximal marginal relevance model via directly optimizing diversity evaluation measures. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 113–122.
- Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.
- Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. *Qwen2.5 technical report*. *Preprint*, arXiv:2412.15115.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Appendix

A Baseline methods

Following Su et al. (2025b), we select five baseline methods as follows: (i) **Vanilla** is the base

LLM without any external knowledge augmentation, serving as a lower bound to assess the necessity of knowledge injection. (ii) **Standard RAG** retrieves the top-k most relevant documents using BM25 and concatenates them with the query as input context for the LLM. (iii) **PRAG** (Su et al., 2025a) trains a dedicated LoRA adapter for each document in the corpus. (iv) **DyPRAG** (Tan et al., 2025) trains a lightweight parameter translator to dynamically convert documents into LoRA parameters at test time. (v) **Poly-PRAG** (Su et al., 2025b) maintains a small fixed set of shared LoRA adapters and learns a latent routing function to map documents to adapter combinations.

B Metrics for clustering quality

We evaluate clustering quality using standard metrics: (i) **WCSS** (Within-Cluster Sum of Squares): the sum of squared Euclidean distances from each data point to its assigned cluster centroid; (ii) **BCSS** (Between-Cluster Sum of Squares): the weighted sum of squared distances from each cluster centroid to the global mean; (iii) **TSS** (Total Sum of Squares): $TSS = WCSS + BCSS$, representing the total variance in the dataset; (iv) **BCSS/TSS**: the proportion of total variance explained by the clustering.