

ULUSLARARASI 3B YAZICI TEKNOLOJİLERİ
VE DİJİTAL ENDÜSTRİ DERGİSİ

INTERNATIONAL JOURNAL OF 3D PRINTING
TECHNOLOGIES AND DIGITAL INDUSTRY

ISSN:2602-3350 (Online)

URL: https://dergipark.org.tr/ij3dptdi

ADAPTIVE TURKISH FEDERATED RAG ARCHITECTURE FOR LOW-RESOURCE SYSTEMS

Yazarlar (Authors): Mehmet Taha Toy, Ahmet Ali Süzen*

Bu makaleye şu şekilde atıfta bulunabilirsiniz (To cite to this article): *Toy. M. T., Süzen A. A., "Adaptive Turkish Federated Rag Architecture for Low-Resource Systems" Int. J. of 3D Printing Tech. Dig. Ind., 10(2): 300-317, (2026).*

DOI: 10.46519/ij3dptdi.1895770

Araştırma Makale/ Research Article

Erişim Linki: (To link to this article): https://dergipark.org.tr/en/pub/ij3dptdi/archive

ADAPTIVE TURKISH FEDERATED RAG ARCHITECTURE FOR LOW-RESOURCE SYSTEMS

Mehmet Taha Toy [a], Ahmet Ali Süzen [b]*

[a]Isparta University of Applied Sciences, Graduate Education Institute, Department of Computer Engineering, TÜRKİYE
[b]Isparta University of Applied Sciences, Faculty of Technology, Department of Computer Engineering, TÜRKİYE

* *Corresponding Author: ahmetsuzen@isparta.edu.tr*

(**Received:** 23.02.26; **Revised:** 09.07.26; **Accepted:** 27.07.26)

ABSTRACT

Today, Large Language Models (LLMs) perform many tasks in the field of natural language processing with high success, from text generation to translation, semantic analysis to code writing. However, these models have some fundamental limitations that make their reliable use challenging. They can produce factual errors known in the literature as hallucinations and cannot directly access developments after their training period. They also sometimes reproduce biases present in the training data. The Retrieval-Augmented Generation (RAG) approach aims to produce more up-to-date and verifiable outputs by dynamically feeding the model with external information sources, thereby reducing hallucination and temporal limitations. This study examines distributed learning approaches aimed at protecting data privacy and QLoRA-based fine-tuning strategies within a mathematical framework. It also addresses 4-bit NormalFloat (NF4) quantization techniques used to improve system efficiency. Furthermore, it details morphology-aware hybrid access methods and adaptive routing mechanisms that can make decisions based on query complexity to achieve results more suitable for Turkish. The study demonstrates that combining RAG-based systems with federated learning and homomorphic encryption creates a secure, decentralized architecture, and that LLM can be efficiently run on low-resource systems using the NF4-QLoRA combination. However, limitations such as encryption latency, erroneous information from untrusted sources, and the inadequacy of standard evaluation methods for Turkish also exist.

Keywords: Federated Learning, RAG, Turkish NLP, QLoRA, NF4 Quantization, Data Privacy.

1. INTRODUCTION

One of the most significant developments in the field of artificial intelligence in recent years has been the emergence of the Transformer architecture and the widespread adoption of Generative Pre-trained Transformers (GPTs) built upon this foundation [1]. These large language models, with billions of parameters, achieve results that approach or exceed human performance in tasks such as text classification, summarization, translation, code generation, and complex question-answering. When trained using extensive text corpora from the internet, these models gain the ability to learn the statistical structure of language and world knowledge. However, the models' parametric memory, i.e., the information encoded in the neuron weights during training, is limited to the date when the training data set was created [2]. This prevents the models from having knowledge about events that occurred after training, real-time data, or internal organizational data.

More critically, hallucination the tendency to generate statistically plausible but factually incorrect outputs instead of acknowledging missing information is one of the biggest obstacles to the industrial adoption of reliable AI systems [3]. For example, a legal assistant referring to a non-existent law or a medical advisor recommending incorrect drug interactions can lead to unacceptable consequences. The RAG approach has been developed in the literature to overcome these structural problems. This approach provides context to the generative

model, requiring that the generated response be based on evidence from an external information source [4,28]. Thus, the model generates responses using dynamic, up-to-date, and verifiable data rather than outdated or incomplete information in its parametric memory. RAG eliminates the distinction between what the model knows and what it can access, ensuring the updatability of information without the need to retrain the model.

However, standard RAG architectures inherently have a centralized structure. The requirement to fragment all corporate documents into vectors and collect them in a single centralized vector database poses unacceptable risks in terms of data security. Particularly in scenarios involving sensitive data, such as financial services, healthcare organizations, and law firms, moving data to a centralized cloud is inconsistent with regulations such as the Personal Data Protection Law and the European Union General Data Protection Regulation and increases the cyberattack surface. Data breaches can lead not only to financial penalties but also to irreversible damage to an organization's reputation.

At this point, the concept of Federated RAG, which combines the Federated Learning paradigm with the RAG architecture, emerges as a strategic solution. Federated Learning protects privacy by moving computation to where the data resides, rather than moving the data to the model [5]. In this approach, raw data does not leave the local node; only model updates or encrypted intermediate representations are shared with the server. However, implementing Federated RAG architectures presents additional challenges, particularly in resource-constrained environments and with morphologically complex languages such as Turkish.

The agglutinative structure of Turkish causes word roots and suffixes to lose meaning and access performance to decline when tokenization and embedding models developed for English are used [6]. For example, although the words “evden” and “eve” are semantically related, they can be mapped to completely different vectors in standard models. Furthermore, installing large language models on every branch or device is not cost-effective.

This study will examine the RAG literature and its evolution starting from the fundamental components and will analyze different RAG paradigms comparatively. The study will then focus on the mathematical infrastructure of the system along with adaptive RAG mechanisms; in particular, the attention mechanism and vector similarity calculations, which are the cornerstones of the Transformer architecture, will be detailed.

Additionally, Turkish natural language processing optimizations, federated learning strategies, and security protocols will be discussed in depth. In this context, the mathematical foundations of advanced quantization techniques such as QLoRA and NF4 and their applications in federated systems will be emphasized. Finally, solution proposals for existing problems, standards, and legal regulations will be discussed, and predictions about future developments will be shared. Thus, the study aims to shed light on individual and institutional applications in both academic and industrial fields.

2. RAG ARCHITECTURE

The RAG architecture is based on the principle of combining a static language model with a dynamic and external knowledge source [4,28]. RAG technology, which has evolved from a simple text matching mechanism to a complex, multi-layered cognitive architecture, is currently studied under three main paradigms: Naive, Advanced, and Modular [7]. This evolution has brought with it mechanisms that enable the model not only to access correct information, but also to parse complex queries, synthesize from multiple sources, and validate its own answers.

2.1. Core RAG Components

The original RAG architecture, introduced to the literature by Lewis and colleagues in 2020, consists of two main modules: Retriever and Generator [4]. This structure is designed to enhance the model's performance in knowledge-intensive tasks.

When a user query is submitted x, the Retriever model ($\rho\eta$) selects the k documents (z) that are semantically most relevant to the query from a large document collection ($\mathcal{Z}$). This selection process is typically performed using a "Nearest Neighbor" (ANN) search in vector space. Vector databases (ChromaDB, Milvus, etc.) use

indexing algorithms such as HNSW (Hierarchical Navigable Small World) to scan millions of documents in milliseconds and retrieve the closest vectors. Then, the generative model ($\rho\theta$) takes both the original query (x) and the selected documents (z) as context to generate the most probable answer (y).

In the RAG-Sequence model proposed by Lewis and colleagues, the entire answer sequence is generated in one go based on the same set of accessed documents. Mathematically, the RAG-Sequence model is expressed as in Equation 1 [4]:

$$P_{RAG-Sequence}(y|x) \approx \sum_{z \in top-k(p(.|x))} p_\eta(z|x) p_\theta(y|x,z) \quad (1)$$

The terms here are:
x: Input Query,
y: Generated Answer,
z: Retrieved Document,

$p_\eta(z|x)$: Represents the probability that the retriever model selects document z for the query x. This value is typically the normalized form of the cosine similarity or inner-product score between the query vector and the document vector.

$p_\theta(y|x,z)$: Represents the probability that the generator model produces the answer y given the query x and document z. This corresponds to the conditional probability of a standard seq2seq model.

2.2. Comparative Analysis of RAG Paradigms

RAG technology has evolved in parallel with the challenges encountered and increasing user expectations. Initially a simple search-and-retrieval process, RAG has been enriched over time with layers such as query optimization, multi-stage access, and answer verification. Table 1 summarizes the key features and differences of the three main RAG paradigms:

Table 1. RAG Paradigms Comparison Table

Feature	Naive RAG	Advanced RAG	Modular RAG
Fundemental Philosophy	Index, Access, Produce	Pre/post processing to improve access quality	Flexible structure composed of independent modules
Access Method	Simple Vector Search (Cosine Similarity)	Hybrid Search (Dense + Sparse), Re-ranking	Adaptive Routing, Subdivision into Subqueries, Iterative Access
Query Processing	Used as is	Rewrite, Query Expansion	Separation, Abstraction
Context Management	The Top-K document is provided in raw format.	Contextual Compression, Noise Filtering	Memory module, Fusion-in-Decoder (FiD), Knowledge Graph Integration
Flexibility	Low (Fixed Pipeline)	Medium (Optimized Pipepiline)	High (Programmable Flow)
Integration	Single Data Source	Multiple Indexing	Agents, Tool Use

2.3. Modular RAG and Adaptive Mechanisms

The modular RAG approach transforms the RAG process from a fixed pipeline into a set of independent modules that can be activated as needed. With this flexibility, the system can dynamically adapt to different query types and difficulty levels.

Specifically, the routing module decides which data source to go to based on the complexity, domain, or type of information required by the incoming query [8]. For example, for the query "What is the dollar exchange rate?", the system directs to a finance API, while for the query "What is the company's internal leave policy?", it directs to the local vector database. This routing decision can be made by an LLM-based classifier or simpler rule-based algorithms.

Adaptive RAG ensures resource efficiency by using a classifier that analyzes the complexity of the query [9]. Running a full RAG process for every query is costly and slow. Adaptive RAG follows this strategy:

• **Basic Queries**: For information that is generally accepted or definitively stored in the model's parametric memory, such as "Hello" or "How are you?", the costly access step is skipped, and the LLM produces the response directly.

• **Medium Complexity**: The standard RAG flow is used for questions that can be answered based on a single document.

• **High Complexity**: Iterative or multi-step access strategies are used for questions that require analyzing and synthesizing multiple documents, such as "By what percentage did the company's revenue increase in 2023 compared to 2022?"

This approach reduces the system's latency while increasing accuracy for difficult questions [9]. New theoretical frameworks such as ExpertRAG combine this routing with the Mixture-of-Experts (MoE) architecture, directing the query to the most suitable expert model or data source [8]. In short, this mechanism can be likened to "smart traffic routing." The aim is to optimize both speed and accuracy by sending the query to the most suitable processing path.

2.4. Mathematical Framework: Attention and Cosine Similarity

At the core of RAG systems lie two fundamental mathematical operations that enable both the language model's understanding capability and the retriever module's similarity calculation. This section examines the mathematical answers to the questions "How are queries matched with documents?" and "Which words does the LLM pay attention to?"

2.4.1. Attention Mechanism

How does an LLM understand whether the pronoun "he" in the sentence "Ahmet went to the market yesterday, he was very tired" refers to 'Ahmet' or "the market"? The answer is the attention mechanism. In the Transformer architecture, attention is a mathematical mechanism that calculates the relationship between a word and other words in a sentence. Although there are many variants of attention (additive attention, multiplicative attention, etc.), the most common type used in the Transformer architecture is Scaled Dot-Product Attention [1]. This mechanism forms the cornerstone of modern language models because it is superior in terms of computational efficiency and parallelizability.

In Transformer models, scores are calculated for each word that answer the question "How related is this word to all other words?" This process is performed across three matrices [1]:

• $\boldsymbol{Q}$ **(Query):** It represents the question "Who am I? What am I looking for?" A query vector is created for each word.
• $\boldsymbol{K}$ **(Key):** It represents the question "Who am I related to? Which words should I match with?" Key vectors define the relationship of each word with other words.
• $\boldsymbol{V}$ **(Value):** It contains the actual information being transferred. After attention scores are calculated, these value vectors are weighted to obtain the final representation.
Mathematically, the Scaled Dot-Product Attention formula is expressed as in Equation 2 [1].

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (2)$$

1. $\boldsymbol{QK^T}$ **(Dot Product):** The product of the transpose of the query and key matrices gives the "coincidence scores" of each word with other words. A high score indicates a strong relationship between two words. For example, the query vector of the word 'o' may produce a high score with the key vector of the word "Ahmet".

2. $\sqrt{\boldsymbol{d_k}}$ **(Scaling Vector):** It prevents scores from growing too large (gradient stability). Inner product results can grow as the vector size increases, and the gradients of the softmax function can become very small. This scaling improves the training process by ensuring gradient stability.

3. Softmax: It converts raw scores into a probability distribution between 0 and 1. This determines how much "attention" each word will pay to other words. For example, the softmax output for the word "o" could be: 70% Ahmet, 5% market, 20% others.

4. Finally $\boldsymbol{V}$ **ile by multiplication:** The weights obtained with Softmax are multiplied by the Value matrix. This process extracts the information of the words with the highest scores and creates a weighted combination. Thus, the word

"o" primarily takes on the semantic representation of "Ahmet's."

In the RAG architecture, the Attention mechanism comes into play at two critical points:

• **Query-Document Matching :** The user query (Q) is matched with the key representations (K) of all documents in the document database, and the most relevant document contents (V) are retrieved. For example, in the query "credit interest rate," the attention mechanism assigns a high score to documents containing the words "interest," "credit," and "rate."

• **Context Integration:** The selected documents are provided as context to the attention layers of the Generator model. By applying attention to both the original query and the document content, the model dynamically determines which pieces of information to use in generating the response. This allows the model to reduce the risk of hallucination by stating, "I obtained this information from the document." With this mechanism, the model produces more accurate and up-to-date answers by paying attention not only to its parametric memory but also to external sources.

2.4.2. Cosine Similarity

There are millions of documents in the vector database. The question "Which document is closest to the query?" must be answered quickly and accurately. To measure the semantic proximity of two documents (or a query and a document) in vector space, the cosine of the angle between the vectors is calculated. This method provides a robust similarity measure that is independent of text length, as it only considers the orientation of the vectors, regardless of their magnitude. This similarity measure is mathematically expressed as in Equation 3 [10].

$$Similarity(A,B) = cos(\theta) = \frac{A.B}{\| A \|.\| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (3)$$

Here:

• A ve B: The two vectors being compared (e.g., query vector and document vector)

• $A \cdot B$: The dot product of vectors

• $\| A \|$ve $\| B \|$: Euclidean norms of vectors (magnitudes)

• θ: The angle between two vectors

Value Ranges and Meanings:

• **Close to 1:** The vectors point in the same direction. High similarity (same topic). For example, the cosine similarity between the query "credit interest rate" and the document "banking interest rates" could be 0.95.

• **Close to 0:** Vectors are perpendicular For example, the similarity between the query "credit interest rate" and the document "soccer match results" receives a very low value such as 0.02.

• **Close to -1:** Vectors in opposite directions.Opposite meaning. For example, a negative score of -0.8 may appear between the expressions "good service" and "bad service."

Advantages of the Cosine Similarity:

1.Size Independence: Whether a document is 100 words or 10,000 words makes no difference; only the orientation of the content matters.

2.Computational Efficiency: Can be computed in parallel with matrix operations, enabling fast searches across millions of documents.

3.Normalized Measure: Produces results within a standard range between -1 and +1, which facilitates the determination of threshold values.

Use in RAG Systems: In the RAG architecture, after the embedding vector of the user query is created, the cosine similarity is calculated with all document vectors in the vector database. The k documents with the highest similarity scores (e.g., top-5) are selected and provided as context to the generator model. For example:

Query: "What are the 2024 credit interest rates?"

- **Document 1 (score: 0.92):** "2024 mortgage interest rates fell to 2.5%..."
- **Document 2 (score: 0.87):** "Consumer loan interest rates increased in 2024..."
- **Document 3 (score: 0.15):** "Weather forecasts..." → This document falls below the threshold value and is eliminated.

In summary, cosine similarity is the fundamental metric that answers the question "Which document should I retrieve?" for RAG systems.

While the attention mechanism manages the model's internal attention, cosine similarity ensures that the most relevant content is selected from external information sources. Together, these two mathematical operations form the backbone of RAG's information access and integration processes.

3. CHALLENGES IN TURKISH NLP AND RAG

Turkish differs significantly from English in terms of its agglutinative structure, vowel harmony rules, and free word order [6]. These structural differences can lead to serious performance losses in the retrieval phase, which is the most critical component of RAG systems trained on English datasets.

3.1. Tokenization and Morphological Segmentation Problem

Standard large language models typically use tokenization algorithms trained on English-dominant data [11]. These algorithms create tokens by combining frequently occurring character sequences in the dataset. However, in languages with a high capacity for word derivation, such as Turkish, these algorithms tend to split words into meaningless fragments that do not follow morphological rules.

Example: The word "Gözlükçü" (optician):

- **Ideal Morphological Analysis**: Göz (Root/Noun) + -lük (Derivational Suffix/Noun from Noun) + -çü (Derivational Suffix/Occupation). Meaning: A person who sells or makes eyeglasses.
- **Standard BPE Tokenization:** Fragments that do not correspond to morphological reality, such as Göz + l + ük + çü or Gözl + ükçü, may be observed.

This type of incorrect segmentation weakens the semantic connection between the word "Optician" and the concepts of 'Eye' or "Seeing" in the vector space, leading to a decrease in the quality of the embedding vector. In RAG systems, this situation leads to documents containing "gözlükçü" having a low semantic similarity score and becoming inaccessible in a scenario where the user searches for "eye health specialist." Furthermore, Turkish's rich inflectional suffixes (possessive, tense, person) can cause words derived from the same root (e.g., "geliyorum," "geleceğim," "gelmişlerdi") to drift apart in the vector space [12].

Table 2. Turkish Tokenization Issues and Comparative Analysis

Word	Correct Morphological Analysis	GPT/Llama BPE Tokenization	Result
gözlükçü	göz + lük + çü	göz + l + ük + çü	"göz" (eye) connection is weak
geliyorum	gel + iyor + um	geli + yor + um	root "gel" (come) is missing
kitaplarımdan	kitap + lar + ım + dan	ki + tap + lar + ım +dan	root "kitap" (book) is broken
çalıştırmak	çalış + tır + mak	ça + lış + tır + mak	root is corrupted
okuyamadım	oku + ya + ma + dım	oku + yama + dım	partially correct
computer	[single token]	comput + er	optimal for English

- **Vector quality decreases:** The search for "optician" gives a low score to "eye doctor" documents ("göz" => "gözlükçü")
- **Token count increases:** Turkish texts use more tokens than English texts = Cost increases
- **Meaning is lost:** he contrast that should exist between "I'm coming" and "to go" is not captured in the vector space (geliyorum / gitmek)

3.2. The Effect on Stemming and Embedding Models

The use of morphological analysis tools in the preprocessing stage of Turkish RAG systems is critical for morphological normalization. Comprehensive morphological analyzers and natural language processing tools stand out in this field [13].

Studies in the literature emphasize the importance of root finding in Turkish information retrieval. In traditional word matching-based algorithms, root finding can improve performance. However, there are also findings that

root finding does not always improve performance in dense access models. In some cases, the semantic nuances added by inflectional suffixes play a critical role in determining the meaning of the vector. For example, only context and suffixes determine the difference between a negative command and a noun-verb.

Therefore, in the proposed architecture, the root finding process is used not before vectorization, but in the sparse access leg of hybrid search, i.e., keyword matching. Thus, both the general search power of the root word and the specific meaning provided by the suffixes are utilized.

Table 3. Hybrid Search vs. Dense-Only Retrieval Quantitative Performance Comparison

Metric	Dense -Only	Hybrid BM25+Dense	Improvement
MRR@3	0.351	0.433	+23.4%
Recall@5	0.587	0.695	+18.4 %
NDCG@10	0.466	0.551	+18.2%

Table 3 shows that there is a significant difference in search efficiency between various hybrid search strategies and dense vector-based search strategies. Combining the BM25 sparse search method with dense vector search via the Reciprocal Row Fusion (RRF) mechanism yields more consistent results compared to when either method is used alone. The hybrid BM25+Dense system outperforms the dense vector-based BM25 search system alone, achieving an improvement of 23.4% in MRR@3 (0.433) and an improvement of 18.4% in Recall@5 (0.695). This improvement was achieved by running T2-RAGBench on a baseline of 23,088 queries across 7,318 financial documents. Additionally, the NDCG@10 score increased from 0.466 to 0.551. This represents an 18.2% improvement compared to the dense search method alone [14]. In this context, achieving a significant performance improvement is particularly important given the morphological complexity of Turkish. Turkish is an agglutinative language, and the semantic shift that occurs during the tokenization process is a common issue for standard vector models. However, BM25's accurate keyword matching restores the connections between documents. This morphological sensitivity thus serves as a crucial balancing mechanism in semantic vector search and provides a level of search quality that dense models alone cannot achieve.

3.3 Turkish Hallucination Detection: TurkLettuceDetect

Hallucination detection is critical for the reliability of Turkish RAG systems. Detection models developed for English are insufficient because they cannot adapt to the grammatical structure of Turkish. Recently developed TurkLettuceDetect is the first hallucination detection model suite specifically designed for Turkish RAG applications [15]. This model uses encoders customized for Turkish to analyze, at the token level, how well the generated response matches the given context. Trained on machine-translated Turkish datasets, this model scores the reliability of the RAG system's generated response, alerting the user in suspicious cases or triggering re-generation.

4. FEDERATED LEARNING AND PRIVACY-FOCUSED RAG ARCHITECTURE

In scenarios where data privacy and data locality must be preserved, Federated Learning, which offers the ability to process data without transferring it to a central server, is a critical complementary technology for RAG systems [5].

4.1. Federated Learning Architectures and RAG Integration

Federated learning is essentially a learning paradigm that enables distributed data owners to train models collaboratively without requiring data to be pooled in a central repository. This paradigm is based on the principle of training models on local devices and sending only weight updates to the server, rather than collecting training data on a central server [5]. This preserves data privacy and ensures compliance with regulatory requirements. Federated learning architectures are addressed under three main headings based on the data distribution among data owners and the nature of the participants [5]. The relevant architectures are shown in Figure 1.

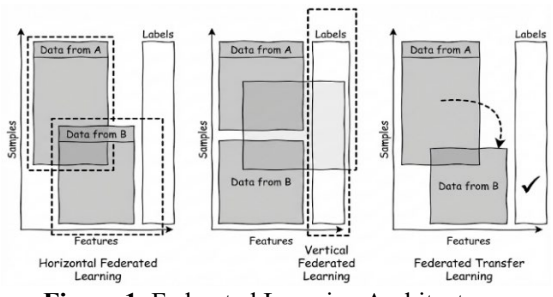


Figure 1. Federated Learning Architectures

Horizontal Federated Learning: Horizontal Federated Learning is suitable for scenarios where participants share the same space but have different sample sets [5]. For example, although banks operating in different geographic regions each record similar customer features (transaction amount, transaction frequency, credit history, etc.), customer prototypes largely do not overlap. The fundamental goal of this architecture is to improve the model's generalization performance by increasing the sample diversity obtained from distributed data sources [5]. In the context of RAG, each participating institution can run a local retrieval component on its own document collection or knowledge base. The common model trained through federated learning learns the signals from these different information sources holistically, while the documents themselves do not leave the institution. This situation can offer critical advantages, especially for multi-branch banking, retail chains, and regional public institutions.

Vertical Federated Learning: Vertical federated learning is used when participants share largely the same sample space but have different feature sets [5]. A typical scenario involves collaboration between institutions that hold data on the same individuals but of different types (e.g., a hospital and an insurance company). The fundamental challenge in this architecture is incorporating different features into a common learning process while preserving their privacy. Cryptographic techniques such as secure multi-party computation and homomorphic encryption are used for this purpose [5]. From the perspective of RAG systems, vertical FL enables the use of complementary pieces of information from different institutions (e.g., clinical notes and financial risk indicators) to enhance the contextual richness of a single generative model. Thus, the model can produce more holistic and context-aware outputs.

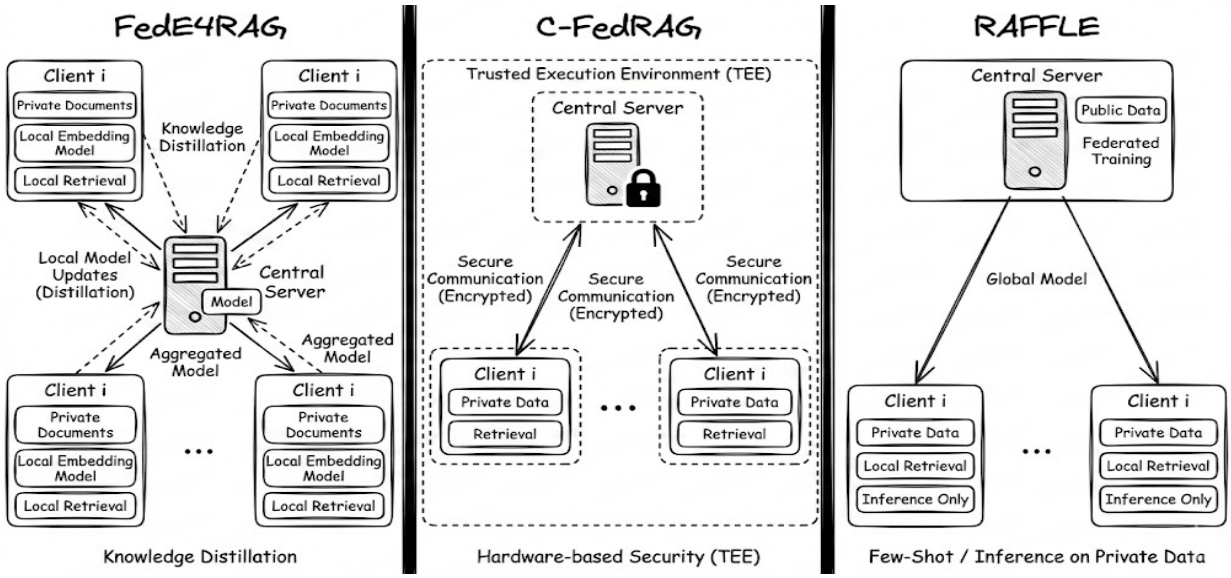


Figure 2. Federated RAG Architecture Comparisons

Federated Transfer Learning: Federated transfer learning targets scenarios where both the sample space and feature space are significantly different [5]. In such cases, classical horizontal or vertical federated learning approaches are insufficient. FTL enables representations learned from partially overlapping datasets to be transferred to other domains [5]. In terms of RAG integration, FTL is particularly prominent in applications requiring cross-domain knowledge transfer. For example, among institutions operating in different countries and holding documents in different languages, information retrieval and production processes can be improved by learning a common semantic representation space. This enables the indirect transfer of institutional knowledge without direct data sharing.

4.2. Federated RAG Architectures: FedE4RAG, C-FedRAG, and RAFFLE

Federated RAG approaches do not start from the assumption that information must be collected in a central pool, unlike classical RAG. Instead, they adopt a structure where information sources are stored in distributed and

local environments, and training and information retrieval processes are carried out according to federated learning principles. In this respect, federated RAG aims to meet privacy and security requirements, especially in areas where sensitive data is processed and subject to regulation (finance, healthcare, law) [16,17]. As shown in Figure 2, the prominent architectures in the literature differ in terms of where the data/access components are located and how the security layer is established.

FedE4RAG (Federated Embedding Learning for RAG): FedE4RAG is based on running local RAG components on the private and sensitive data owned by clients and improving these components in a federated manner. Each client hosts a local access point on its own document collection; raw data or document content is not shared with the central server. On the server side, only model updates are combined within the federated learning flow [16]. In the FedE4RAG diagram in Figure 2, it is clearly visible that the "local embedding model" and "local retrieval" components are located close to the local data on the client side.

One of the distinguishing features of FedE4RAG is that it establishes an indirect information transfer between the client and the server through information distillation mechanisms. Thus, while clients contribute the representation information they learn from local data to the central model, the risk of privacy violation is minimized. In the RAG context, this design aims to improve the quality of local accessors, enabling the retrieval of more relevant documents for the query and enhancing the contextual accuracy of the generative model [16].

C-FedRAG (Confidential Federated Retrieval-Augmented Generation): C-FedRAG takes confidentiality a step further in a federated RAG scenario by leveraging hardware-based security mechanisms. A critical advantage of this architecture is that it aims to provide a stronger layer of protection not only against federation participants but also against the central server infrastructure and operating system. This provides a more robust defense against threat models such as the "honest but nosy server." Figure 2, which shows C-FedRAG with a central server located within the TEE, visually supports the fundamental claim of this architecture (isolation of the processing environment) [17]. For this reason, C-FedRAG is considered a particularly suitable option for finance, defense, and public applications with high security requirements.

RAFFLE (Retrieval-Augmented Few-Shot Federated Learning): The RAFFLE architecture combines federated learning and retrieval-augmented learning with the few-shot paradigm. In this approach, the federated learning process is performed on publicly available datasets that do not pose privacy risks. The resulting shared model is distributed to clients; clients use this model only during the inference stage on their own private data [18]. Figure 2 shows the separation of the RAFFLE flow into training on public data and inference on private data, directly highlighting the privacy advantage of the architecture.

This design offers a strong privacy advantage because local data is not directly included in training; however, the limited contribution of client data to the federated training process may constrain the model's contextual richness specific to the domain/institution. In the RAG context, RAFFLE offers a practical solution, especially for clients with limited data; however, it may show more limited performance compared to approaches such as FedE4RAG or C-FedRAG in highly customized enterprise scenarios [18].

4.3. Security Mechanisms: Homomorphic Encryption and Differential Privacy

Although federated learning and federated RAG architectures provide a significant privacy advantage by preventing raw data from being transferred to a central server, this alone does not guarantee absolute security. Studies in the literature show that, under certain conditions, the original data can be partially or completely re-estimated through model updates, gradients, or vector representations shared by clients. Therefore, additional security mechanisms are needed to strengthen privacy in federated systems.

In this context, three fundamental security approaches commonly used in federated RAG architectures stand out: Homomorphic Encryption, Secure Aggregation, and Differential Privacy.

Homomorphic Encryption: Homomorphic encryption is a cryptographic approach that allows operations to be performed on encrypted data. While classical encryption methods require decryption before operations can be performed on the data, homomorphic encryption eliminates this requirement. Thus, computations can be performed without ever revealing the content of the data.

In the Federated RAG scenario, this approach is particularly important in terms of query privacy. For example, a query vector created by a user and document vectors stored on the server can be kept encrypted. The server can calculate the similarity between these vectors without decrypting the data. This prevents the server from learning what the user is searching for or the content of the documents.

The total homomorphic encryption feature can be expressed mathematically in a simplified form as in Equation 4 [19].

$$E(x).E(y) = E(x + y) \quad (4)$$

This feature ensures that operations performed on encrypted values correspond to operations on plaintext data. However, the most significant disadvantage of homomorphic encryption is its high computational cost. Especially in RAG systems involving large vectors and heavy query traffic, delays can reach the order of seconds. For this reason, HE is generally used with limited or hybrid solutions in practical systems [19].

Table 4. Homomorphic Encryption Computational Overhead vs. Plaintext Inference

Task / Scenario	Plaintext Time	Encrypted (HE) Time	Overhead Factor
BERT-base (128 tokens, CKKS)	~0.02 s	~1,103 s	~55,000x
CNN ResNet-20 / CIFAR-10 (CKKS, GPU)	~0.001 s	~1.4 s	~1,400x
2-layer NN inference (CKKS, CPU)	~0.1 s	~2.4 s	~24x
CNN 1.2M params (Paillier/CKKS, CPU)	~13 s	24–81 min	~110–370x
Selective HE	Baseline	~10x overhead	90% faster than full FHE

The processing speed results in Table 4 demonstrate the additional overhead that homomorphic encryption imposes on operations. As seen, performing everything with full HE is much slower compared to doing so with normal, plaintext data [20]. Although the CKKS (Cheon-Kim-Kim-Song) method should be faster at processing the floating-point numbers required by AI models due to its approximate computational advantage, it is actually complex and significantly slows down operations.

For example, in a complex transformer system like BERT-base, finding the answer to a 128-segment problem takes up to 1.103 seconds when the data is encrypted, which is 55,000 times longer than when using unencrypted data [21]. Even in simpler CNN tasks and on the fastest graphics cards, the time remains around 1.4 seconds. On a smaller computer without a graphics card, the delay is 24 to 370 times longer [22].

When all these factors are considered together, it becomes clear that using full FHE directly or obtaining real-time responses in RAG on devices with limited processing power poses a significant speed issue. However, Table 4 also shows that by encrypting only the most critical data segments, the total computational load for FHE can be reduced by 90%. To strike a balance between ensuring data security and continuing to use a RAG system, a combination of these methods emerges as the most reasonable and feasible option [20].

Secure Aggregation: Secure aggregation is a protocol that prevents the server from seeing individual client updates in federated learning systems. In this approach, the server only obtains the sum or average of the updates from the clients; information belonging to individual clients remains confidential.

This method, proposed by Bonawitz and colleagues, is based on cryptographically masking client updates. Each client hides its own update with random noise or masks. These masks cancel each other out when the clients are evaluated collectively, and the server can only obtain the total result. Secure federated learning is particularly critical in scenarios such as multi-branch banking, corporate information systems, or public institutions. This ensures that no participating institution or branch's contribution to the model is individually revealed; only the collective learning outcome is shared [23].

Differential Privacy: Differential privacy is a mathematical privacy framework that guarantees that a system's output is not meaningfully affected by the presence or absence of a single data record. This approach is based on the principle of adding a controlled amount of random noise to gradients or vector representations sent from the client to the server. In federated RAG systems, differential privacy is used specifically to prevent potential data leaks through model updates. The added noise makes it difficult to distinguish individual data points while largely preserving the model's ability to learn general patterns.

The main advantage of this approach is its relatively low computational cost and easy integration into scalable systems. However, as the noise level increases, model performance may decline, so the privacy-accuracy trade-off must be carefully balanced [24].

5. QLORA AND NF4 QUANTIZATION

One of the key factors limiting the practical use of large language models in real-world systems is their high memory and computational requirements. Especially in federated learning and privacy-focused RAG scenarios, it is often not feasible for every client to have high-capacity hardware. In this context, quantization methods that aim to reduce the memory footprint of models play a critical role.

Quantization is the process of converting numerical values from high-precision representations (e.g., FP32 or BF16) to representations with lower bit width. This conversion reduces memory consumption and memory bandwidth requirements while resulting in limited performance loss when appropriate methods are used. The QLoRA approach, one of the most effective methods in this field, enables large-scale models to be trained even on a single processor [25]. QLoRA's fundamental contribution relies on storing a pre-trained large base model in memory in a low-bit format while only updating small, trainable adapter layers with high precision. This approach provides a significant advantage for decentralized learning processes, especially in federated systems considering hardware heterogeneity and resource constraints.

5.1. Quantization Theory and NF4

Traditional quantization methods typically map weights to integer levels at equal intervals. However, when examining the weights of large pre-trained language models, it is observed that these values mostly exhibit an approximate normal distribution. Equal-interval quantization can cause more errors in information-critical regions because it assigns the same sensitivity to regions where the distribution is dense and to regions where it is sparse.

To address this issue, NF4 is proposed as a quantization data type optimized from an information theory perspective for data with a normal distribution [25]. NF4 is based on the quantile-based quantization principle and adapts the quantization intervals according to the distribution of the data. Thus, representation error is minimized by creating narrower intervals in regions where weights are densely concentrated and wider intervals in sparse regions. This feature makes NF4 more effective than classical 4-bit quantization methods, especially for long-tailed weight distributions seen in large language models.

Table 5. NF4 (QLoRA) vs. BF16: Memory Requirements and Task Performance Comparison

Model Size	BF16 VRAM	NF4 VRAM	Memory Reduction
7B parameters	~14 GB	~3.5 GB	4x
13B parameters	~26 GB	~6.5 GB	4x
33B parameters	~60 GB	~19 GB	~3.2x
65B parameters	>780 GB	<48 GB	>16x
Non-English languages (4-bit avg.)	Baseline	—	—

Table 5 clearly shows that NF4 quantization uses significantly less memory than traditional 16-bit models and illustrates how this affects speed. Specifically, it reduces GPU memory usage by a factor of four to sixteen in models with 7 billion to 65 billion parameters [25]. This makes it possible for people to actually fine-tune massive models on ordinary consumer-grade computers. When looking at English tests (such as MMLU and Elo), studies show that NF4, with its double quantization feature, performs nearly as well as BF16, with a difference of only 1 point. However, while memory efficiency is greatly improved, recent studies highlight significant trade-offs in specific applications. Experiments show that using BNB-NF4 results in an average accuracy drop of 6.9% for standard tasks and up to 23% for tasks involving long-text retrieval [26]. Therefore, it is essential to thoroughly test NF4 models, and if setting up a Turkish RAG system, additional tuning for Turkish may be necessary. Nevertheless, NF4-QLoRA remains the most logical and efficient choice for systems utilizing a large number of diverse, yet not particularly powerful, hardware components.

5.1.1. NF4 Mathematical Formulation

In order for a language model to run using less memory, its weights must be represented by a more limited number of values. One of the most common ways to achieve this is through quantization. Quantization stores the continuous weight values learned by the model during training by rounding them to a smaller number of "permitted" values. The aim is to significantly reduce memory usage.

In NF4 normalization, weights are first scaled appropriately to approximate $[-1,1]$ representing within a range and then mapping each value in that range to one of a limited number of levels. In a k-bit quantization, the total representation consists of 2^k representation levels. For example, in NF4 (k=4), weights are represented by only 16 different values. The distinguishing feature that sets NF4 apart from the classic 4-bit approach is that these 16 levels are not selected at equal intervals. This is because in large language models, weights often resemble a roughly normal (Gaussian) distribution. In a Gaussian distribution, most values cluster around 0, with very few values at the extremes. Equal-interval quantization can cause unnecessary information loss precisely in the most dense region around 0.

For this purpose, NF4 utilizes the statistical structure of the standard normal distribution when determining quantile levels. To do this, the quantile function $Q_X(\cdot)$, which is the inverse of the CDF function of the standard normal distribution, is used.

Furthermore, it is particularly important that the value 0 can be fully represented. Since "being 0" sometimes carries a special meaning in operations such as masking and padding, NF4 practically arranges the levels to allow for this [25].

The theoretical basis of NF4 quantization levels is mathematically defined as in Equation 5.

$$q_i = \frac{1}{2}\left[Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)\right] \quad (5)$$

Here $Q_X(\cdot)$, the inverse CDF function of the standard normal distribution, $0 \le i < 2^k$ represents the level index within the interval. This approach reduces quantization error by ensuring that each quantization level represents the center of its own quantile interval.

Block-Based Quantization:

Instead of quantizing the entire weight tensor as a single piece, NF4 typically applies it by dividing the weights into small blocks (e.g., 64-element blocks) [25]. The reason for this is that if there are outlier values (very large/small) in a block, their effect should not spread to the entire tensor but remain limited to that block.

For each block, the scaling factor for that block is calculated by taking the weight with the largest absolute value, as shown in Equation 6:

$$c = max(|W|) \quad (6)$$

Then the weights within the block are first normalized by dividing them by this value, then rounded to NF4 levels as expressed in Equation 7:

$$W_{quant} = NF4\left(\frac{W}{max(|W|)}\right) \quad (7)$$

We can intuitively think of this as "scale first, then convert to 4-bit." During computation,

dequantization is performed to convert back from quantized values to approximate actual values, and the scale is reapplied as shown in Equation 8:

$$W_{dequant} = c.W_{quant}^{NF4} \quad (8)$$

Here $\left(W_{quant}^{NF4}\right)$ The quantized weights selected from the NF4 levels represent the scale factor of the relevant block.

5.2. Double Quantization

QLoRA does not merely reduce weights to 4 bits using NF4 to further reduce memory usage; it also focuses on the space occupied in memory by the constant values required during quantization. Block-based quantization requires storing additional information such as a scaling factor for each block. When these factors are stored with high precision (e.g., with more bits), the total memory load becomes significant as the number of blocks increases with model size. This is where the idea of Double Quantization comes in: "We quantized the weights, so what if we also compress these constants we store to scale the weights back correctly into a smaller form?" Thus, the constants are quantized a second time, reducing the storage cost.

This process is mathematically expressed as in Equation 9:

$$W_{final} = dequant\left(c_2, c_1^{quant}\right).W_{quant}^{NF4} \quad (9)$$

Here c_1^{quant}, The quantized form of the constants (e.g., block scales) that emerge during the initial quantization stage; (c_2) is the second-level quantization information used to restore these constants to their approximate values when needed. In summary, the system significantly reduces the average memory cost per parameter by both storing weights at the 4-bit level and additionally compressing the "side information required to scale these weights back to the correct scale" [25].

5.3. Paged Optimizers and Memory Management

Paged optimizers are a memory management approach developed to reduce OOM (out-of-memory) issues that arise when GPU memory fills up quickly due to the additional memory held by the optimizer, especially in methods such as Adam/AdamW. The logic is quite practical: when GPU memory is nearing capacity, instead of keeping all optimizer states continuously in VRAM, the unused portions are automatically moved to RAM using paging logic; when needed, they are brought back to the GPU.

This allows the training or fine-tuning process to proceed more stably, even when memory usage spikes suddenly. This approach offers practical advantages, particularly in long-context scenarios (where cache pressure increases due to long sequences) and in federated learning, where memory load accumulates during local training steps. Of course, there is a cost: some speed loss may be observed due to the additional data transfer between the GPU and RAM. Nevertheless, when properly configured, it significantly facilitates experimentation by enabling larger batch/sequence trials [25].

5.4. Fine-Tuning the Mathematical Model with QLoRA

The starting point for QLoRA is quite simple: Instead of retraining all the parameters of a large language model, the model's main weights are kept fixed and low-order adapters are added to the model. During training, only these small adapter layers are updated. This significantly reduces memory usage, while the resulting performance is often very close to full fine-tuning [27].

This approach is based on the idea of "adding a low-order correction on top of frozen weights." Let the pre-trained weight matrix in a layer be expressed by Equation 10:

$$W_0 \in \mathbb{R}^{d\times k} \quad (10)$$

When LoRA is used, the model's effective weights take the form given in Equation 11:

$$W = W_0 + BA \quad (11)$$

Here, W_0 represents the frozen weights of the base model. In the QLoRA context, these weights are stored in memory in NF4 format, i.e., compressed with 4-bit precision. The $B \in \mathbb{R}^{d\times k}$ matrices are trainable LoRA adapters and are typically stored in a higher-precision format such as BF16. The rank value r determines the size of these matrices and is usually quite small compared to the values of d and k. As the rank

decreases, the number of trained parameters also decreases significantly.

Thanks to this structure, the weights of the base model remain unchanged. The model's adaptation to new tasks is learned solely through low-rank updates, expressed by the *BA* term. This keeps both computational and memory costs under control.

During forward propagation, the output calculated for an input x can be written in general terms as in Equation 12:

$$h = W_0 x + BAx \tag{12}$$

This process occurs step by step as follows. First, the input matrix X is typically in BF16 format. Since the frozen base weights W_0 are stored in memory in NF4 format, they are converted from NF4 to BF16 during computation (dequantization). Then, the XW_0 multiplication is computed via the main path. In parallel, the XBA operation is performed on the adapter path; the adapter's contribution is multiplied by a scaling factor defined by the α/r ratio, and finally, the outputs from the main path and the adapter path are summed. The final operation for QLoRA is shown in Equation 13:

$$Y = dequant(W_{NF4})X + \frac{\alpha}{r} \cdot XBA \tag{13}$$

There are several key reasons why this approach works in practice. First, storing the core weights in 4-bit NF4 format dramatically reduces VRAM usage, especially in large models. Second, since only the A and B matrices are updated during training, learning is performed with far fewer parameters compared to full fine-tuning. This enables both a faster and more accessible training process.

In summary, QLoRA enables adapting large language models to new tasks without the need for complete retraining. While the base model remains compressed and fixed, the learning process is executed through LoRA adapters. With these features, QLoRA offers an extremely suitable fine-tuning approach for federated or local systems with limited hardware resources [25,27].

6. QUANTITATIVE SYNTHESIS OF KEY FINDINGS

This section synthesizes quantitative findings related to the Federated RAG architecture, memory efficiency, and encryption.

6.1. Literature Review on NF4-QLoRA Memory Efficiency and Privacy

NF4 quantization (QLoRA) has been shown to enable the use of large models on consumer-grade hardware by reducing GPU VRAM usage for a 7-billion-parameter model from approximately 14 GB to 3.5 GB a reduction of nearly fourfold thanks to NF4 compression [25]. However, as a result of this compression, 4-bit quantization causes an average accuracy drop of 6.9% in languages other than English, including Turkish, and this drop emerges as a disadvantage, reaching up to 23% in long-context search tasks [26]. The second option, selective homomorphic encryption strategies, reduces the computational load by up to 90% compared to full FHE, making them a practical choice for federated RAG. Here, the architectural privacy advantage stems from the structure itself. Raw data never leaves the local node. Thus, natural compliance with GDPR/KVKK is achieved without the need for procedural controls [20].

6.2. Search Accuracy Between Hybrid Search and Vector-Only Search in Turkish

Looking at Table 3 in Section 3.2, we can see how well results were obtained when comparing a hybrid search that understands Turkish word structure with vector or exact-match searches. From these comparisons, it is clear that the hybrid search that understands Turkish performs much more reliably and effectively than the vector approach. In particular, in the T2-RAGBench tests, the MRR@3 score improved by 23.4% from 0.351 to 0.433, while the Recall@5 score improved by 18.4% from 0.587 to 0.695. For NDCG@10, the results have also improved from 0.466 to 0.551. This represents an 18.2% increase compared to vector search alone. Since Turkish is a language where numerous suffixes are added to words to change their meaning, typical vector models require exact matches of word parts to identify documents that would otherwise become meaningless when broken down into individual words. Therefore, these results demonstrate that for Turkish, a search system designed with Turkish in mind is not just a nice-to-have but an

absolute necessity in a RAG system that pulls information from multiple sources.

7. CONCLUSIONS AND RECOMMENDATIONS

While artificial intelligence technologies are advancing rapidly, large language models are delivering impressive results in many areas, from text generation to solving complex problems. However, these models have two fundamental limitations. First, they cannot naturally access current information released after their training date. Second, in some cases, they can produce factual errors or baseless answers. These two issues are critical for developing reliable artificial intelligence systems.

The RAG approach discussed in this study comes into play precisely at this point. RAG dynamically feeds the language model with external knowledge sources, combining what the model knows with the latest information it can access. This allows the system to generate both more accurate and more up-to-date answers. In short, RAG bridges the gap between the LLM's parametric memory and verifiable knowledge in the outside world. However, the weakest point of traditional RAG architectures is this: In most applications, all documents must be collected in a central vector database. In sensitive areas such as finance, healthcare, and law, this situation poses serious risks in practice. Moving data to a central location increases the attack surface; it can also create compliance issues in terms of regulations such as KVKK/GDPR. Therefore, the question “RAG is good, but how can we do it without centralizing data?” becomes critical. In this context, combining federated learning with RAG offers a strategic solution. The fundamental principle of the federated approach is clear: data stays where it is, computation goes to the data. Organizations can contribute to a shared learning process without sharing their raw data. This preserves privacy while enabling the creation of a more powerful system fueled by diverse data sources. The concept of “Federated RAG” combines this collective learning advantage with RAG's information access capability.

In the case of Turkish, there is an additional layer of difficulty. Since Turkish is a morphologically rich and agglutinative language, tokenization and embedding tools developed with English in mind do not always deliver sufficient performance. Standard tokenization methods may split words in a way that does not conform to Turkish's morphological structure; this can reduce embedding quality and lead to performance loss during retrieval. Furthermore, the dense inflectional suffixes in Turkish can cause words derived from the same root to be unnecessarily distant from each other in the vector space. Therefore, in Turkish RAG design, morphological analyzer integration, hybrid search (sparse + dense) strategies, and Turkish-specific embedding solutions become particularly important.

Another practical requirement is this: It is not always possible to run such systems on powerful servers. Quantization techniques play a critical role in enabling the use of LLMs in resource-constrained environments. Methods such as QLoRA and NF4 reduce memory requirements by reducing weights to 4-bit levels, enabling fine-tuning processes even on single consumer-grade hardware. This is even more important in federated scenarios, as clients' hardware is often heterogeneous and limited. Memory management approaches also stabilize the process by reducing potential OOM issues during training.

The security component is indispensable for the industrial applicability of Federated RAG. Techniques such as homomorphic encryption, secure aggregation, and differential privacy reduce the risk of data leakage that could occur through shared updates or intermediate representations. However, these techniques often introduce additional computational cost and latency. Therefore, in future work, “hybrid” designs that balance performance without compromising security emerge as a critical area of research.

Another aspect that increases system efficiency is adaptive RAG mechanisms. Not every query has to be processed at the same cost. Skipping the access step for simple queries, using the standard RAG flow for medium-level queries, and engaging iterative access for queries requiring multi-step reasoning increases scalability and cost-effectiveness. Additionally, routing approaches such as router modules and MoE can further improve performance by ensuring the query goes to the most appropriate expert or data source.

One of the significant gaps in the development of Turkish RAG systems lies in the evaluation aspect. While existing datasets are useful for measuring the general language proficiency of models, they are not sufficient on their own to measure the end-to-end performance of a RAG system (access quality, context integration, hallucination control, etc.). Therefore, the development of Turkish-specific RAG benchmarks, metrics, and evaluation protocols is a critical need for the field to mature.

Furthermore, hallucination detection remains important even in RAG systems. RAG reduces hallucinations but does not eliminate them entirely. Hallucination detection models developed for Turkish should be positioned not only as a final check but as a natural part of the production process. For example, when the model detects a suspicious situation, alerting the user, triggering re-access/re-generation, or providing output with a confidence score opens the door to more reliable systems. How these detection models will be trained and kept up-to-date in a federated environment is also a topic that needs further research. One of the most significant technical bottlenecks of federated RAG is the latency problem, particularly due to encryption and network communication. Strong security approaches such as homomorphic encryption can incur high processing costs. To mitigate this, methods such as vector quantization at the access stage, asynchronous communication protocols, and edge computing integration can be evaluated. Vector quantization can both reduce communication costs by decreasing data size and speed up access.

On the other hand, the Modular RAG approach transforms the system from a single fixed pipeline into an expandable structure that can be scaled according to need. Knowledge graphs, multi-document synthesis, tool usage, and agent-based components can be added in a modular fashion. The development of modules specific to Turkish and how these modules will be coordinated in a federated environment is a natural direction for future work.

From an industrial perspective, technical infrastructure alone is not enough. Organizational and legal frameworks are also required. Clear standards and protocols are needed for issues such as inter-institutional update sharing, data ownership, fair evaluation of contributions, performance monitoring, and auditing. It is critically important to design federated RAG in compliance with regulations, especially in areas with a high regulatory burden.

The open-source ecosystem is also decisive in this area. The increase in open source tools, datasets, and pre-trained models for Turkish NLP and RAG both accelerates academia and facilitates industrial adoption. The open sourcing of Turkish tokenization, morphological analysis, embedding, and hallucination detection components is an important lever for the growth of the ecosystem.

In conclusion, the approaches discussed in this study demonstrate that addressing privacy requirements and the need for access to up-to-date and verifiable information simultaneously is crucial for developing reliable artificial intelligence systems. Combining distributed learning principles, memory-efficient quantization techniques, Turkish morphology-aware retrieval methods, and adaptive query routing mechanisms yields solutions that are both more secure and more operationally efficient. For this to be implemented sustainably in the field, however, collaboration between academia, industry, and the public sector, the establishment of common standards, the maturation of evaluation frameworks, and the development of application practices compliant with data protection legislation are of critical importance.

REFERENCES

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I., "Attention is all you need", Advances in Neural Information Processing Systems, Vol. 30, 2017.

2. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I., "Language models are unsupervised multitask learners", OpenAI Blog, Vol. 1, Issue 8, Page 9, 2019.

3. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A. and Fung, P., "Survey of hallucination in natural language generation", ACM Computing Surveys, Vol. 55, Issue 12, Pages 1-38, 2023.

4. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S. and Kiela, D., "Retrieval-augmented generation for knowledge-intensive nlp tasks", arXiv preprint arXiv:2005.11401, 2020.

5. Yang, Q., Liu, Y., Chen, T. and Tong, Y., "Federated machine learning: Concept and applications", ACM Transactions on Intelligent Systems and Technology, Vol. 10, Issue 2, Pages 1-19, 2019.

6. Eryiğit, G., Nivre, J. and Oflazer, K., "Dependency parsing of Turkish", Computational Linguistics, Vol. 34, Issue 3, Pages 357-389, 2008.

7. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J. and Wang, H., "Retrieval-augmented generation for large language models: A survey", arXiv preprint arXiv:2312.10997, 2024.

8. Gumaan, E., "ExpertRAG: Efficient RAG with mixture of experts -- Optimizing context retrieval for adaptive LLM responses", arXiv preprint arXiv:2504.08744, 2025.

9. Jeong, S., Baek, J., Cho, S., Hwang, S. J. and Park, J. C., "Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity", arXiv preprint arXiv:2403.14403, 2024.

10. Reimers, N. and Gurevych, I., "Sentence-BERT: Sentence embeddings using Siamese BERT-networks", arXiv preprint arXiv:1908.10084, 2019.

11. Sennrich, R., Haddow, B. and Birch, A., "Neural machine translation of rare words with subword units", arXiv preprint arXiv:1508.07909, 2016.

12. Şahin, G. G. and Steedman, M., "Data augmentation via dependency tree morphing for low-resource languages", arXiv preprint arXiv:1903.09460, 2018.

13. Eryiğit, G., "ITU Turkish NLP web service", Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, Pages 1-4, 2014.

14. Akarsu, M., Karaman, R. K., & Mierbach, C., “From BM25 to Corrective RAG: Benchmarking Retrieval Strategies for Text-and-Table Documents”. Retrieved from https://arxiv.org/abs/2604.01733, 2025.

15. Taş, S., El Huseyni, M., Ezerceli, Ö., Bayraktar, R. and Terzioğlu, F. B., "Turk-LettuceDetect: A Hallucination Detection Models for Turkish RAG Applications", arXiv preprint arXiv:2509.17671, 2025.

16. Mao, Q., Yuan, Z., Yin, J., Jiang, W., Wang, L., Liang, J., Dong, S., Qian, S. and Zhao, D., "Privacy-Preserving Federated Embedding Learning for Localized Retrieval-Augmented Generation", arXiv preprint arXiv:2504.19101, 2025.

17. Addison, P., Nguyen, M. H., Medan, T., Shah, J., Manzari, M. T., Shen, S., Hashemi, M., Gupta, U. and Reddi, V. J., "C-FedRAG: A Confidential Federated Retrieval-Augmented Generation System", arXiv preprint arXiv:2412.13163, 2024.

18. Zhang, Y., Liu, Y., Tan, J. and Huang, J., "RAFFLE: Retrieval augmented few-shot learning with language models for federated learning", arXiv preprint arXiv:2307.09284, 2023.

19. Gentry, C., "A fully homomorphic encryption scheme", Doctoral Dissertation, Stanford University, Stanford, 2009.

20. Korkmaz, A. and Rao, P., "A Selective Homomorphic Encryption Approach for Faster Privacy-Preserving Federated Learning", arXiv preprint arXiv:2501.12911, 2025.

21. Hong, C., “Recent advances of privacy-preserving machine learning based on (Fully) Homomorphic Encryption.” Security and Safety, 4, 2024012,2025.

22. Ao, W. et al., "AutoFHE: Automated Adaption of CNNs for Efficient Evaluation over FHE", IACR ePrint, 2023.

23. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A. and Seth, K., "Practical secure aggregation for privacy-preserving machine learning", Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Pages 1175-1191, 2017.

24. Dwork, C. and Roth, A., "The algorithmic foundations of differential privacy", Foundations and Trends in Theoretical Computer Science, Vol. 9, Issue 3-4, Pages 211-407, 2014.

25. Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L., "QLoRA: Efficient finetuning of quantized LLMs", arXiv preprint arXiv:2305.14314, 2023.

26. Mekala, A., Atmakuru, A., Song, Y., Karpinska, M., & Iyyer, M.. “Does quantization affect models’ performance on long-context tasks?”, Retrieved from https://arxiv.org/abs/2505.20276, 2025.

27. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W., "LoRA: Low-rank adaptation of large language models", arXiv preprint arXiv:2106.09685, 2025.

28.Toy, M. T., & Süzen, A. A. “Advantages And Disadvantages of Retrieval Augment-Ed Generation (Rag) In Generative AI”. International Journal of 3D Printing Technologies and Digital Industry, Vol. 9, Issue 3, Pages 462-475, 2025.