

Tokenization Risk as a Control Signal for Adaptive RAG

Norbert Braunschweiler¹, Rama Doddipatla¹, Nigel Collier²,

¹Toshiba Europe Limited, Cambridge, UK, ²University of Cambridge,

{norbert.braunschweiler, rama.doddipatla}@toshiba.eu, nhc30@cam.ac.uk

Abstract

Retrieval-Augmented Generation (RAG) systems can degrade on inputs containing specialized, low-frequency vocabulary, where subword tokenization introduces fragmentation that disrupts both retrieval and generation. We introduce *tokenization risk*, a lightweight metric that predicts these failures. We show that (i) tokenization risk strongly correlates with degradation in retrieval and answer quality, and (ii) static mitigations (e.g., hybrid retrieval) provide only partial gains. To address this, we propose risk-based routing, which adaptively switches retrieval and generation strategies based on risk. Experiments on *SpecialVocabBench*, a risk-stratified benchmark of technical documents, demonstrate that this approach consistently improves retrieval accuracy and answer correctness with small system changes, establishing tokenization risk as an effective control signal for adaptive RAG.

1 Introduction

Industrial settings such as appliance installation, maintenance, and repair rely on extensive technical documentation filled with domain-specific terminology and tightly coupled procedures. Harnessing these materials via Retrieval-Augmented Generation (RAG) to support Large Language Models (LLMs) remains challenging, especially when precision and reliability are required to guide technicians in real-world operations (Figure 1).

A key difficulty is the handling of specialized identifiers. Such strings (e.g., part numbers and encoded specifications) are often split into subword fragments that distort representations and break lexical identity. This fragmentation can degrade retrieval through embedding mismatch for rare entities and degrade generation through incorrect reproduction or interpretation of terminology, even when the relevant information is present in the corpus (Sennrich et al., 2016; Pawar et al., 2025; Chai

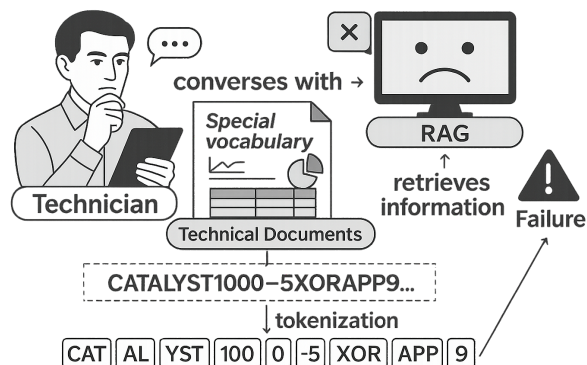


Figure 1: Illustration showing how specialized vocabulary in technical documents can impact the conversation between an information-seeking technician and an information-providing Retrieval-Augmented Generation (RAG) system, highlighting how complex terms like *CATALYST1000-5XORAPP9* can be incorrectly tokenized, leading to retrieval failures.

et al., 2024; Ram et al., 2023; Haslett, 2025; Wang et al., 2025). While prior work has documented the sensitivity of language models to tokenization (Chai et al., 2024), most RAG pipelines remain tokenization-agnostic, applying mitigation strategies uniformly and often incurring unnecessary noise or inefficiency.

We treat tokenization as an actionable control signal. We introduce *tokenization risk*, a lightweight score derived from tokenizer behavior, and show that it strongly predicts degradation in retrieval and answer quality. Building on this signal, we propose risk-based routing: low-risk inputs use standard dense retrieval and LLM generation, while high-risk inputs switch to lexical retrieval (BM25, Robertson and Zaragoza, 2009) and, when needed, vision-language generation over document images. We evaluate this approach on *SpecialVocabBench*, a risk-stratified benchmark constructed from industrial manuals, and find that:

1. Tokenization risk correlates strongly with degradation in retrieval and generation,

2. Static mitigations (e.g., hybrid retrieval) provide partial improvements,
3. Risk-based routing adaptively improves retrieval accuracy and answer correctness with minimal system changes.

1.1 Related work

A growing body of evidence shows that tokenization failures propagate through RAG, degrading retrieval, corrupting grounding, and ultimately lowering answer correctness. Wang et al. (2025) demonstrates that carefully constructed inputs which challenge subword segmentation (the ADT benchmark) reliably degrade state-of-the-art LLMs, directly implicating tokenization as a failure point for tasks that depend on accurate lexical semantics and span selection, such as RAG. Complementary analyses document subword brittleness: typos, boundary shifts, and segmentation variance alter model behavior and reduce robustness, even when relevant context exists Chai et al. (2024). Crucially, Haslett (2025) shows that segmentation alone can shift semantic interpretation, providing causal evidence that tokenization can change meaning without any surface change, a mechanism that naturally explains embedding mismatch and retrieval errors in RAG. Recent work on adversarial tokenization further shows that alternative valid segmentations can evade guardrails or distort model interpretation without altering the surface string, exposing a new axis of input-space attack that RAG pipelines may inadvertently retrieve and amplify Geh et al. (2025).

From a design standpoint, empirical and theoretical studies establish that tokenizer choice materially affects accuracy, efficiency, and learning dynamics, and that suitable tokenization is often a precondition for reliable modeling which implies that retrieval correctness and generation fidelity in RAG depend on the tokenizer Ali et al. (2024); Rajaraman et al. (2025). In parallel, tokenization-light and token-free encoders at the byte or character level reduce reliance on brittle subword boundaries and are more robust to noisy inputs. These properties are likely to improve retrieval matching and context fidelity in RAG Xue et al. (2022); Clark et al. (2022).

Xu et al. (2025) propose a theoretical framework for token-level harmonization in RAG and show how misaligned token fusion between retrieved content and LLM predictions can distort next-token

probabilities and degrade answer quality. A systematic risk assessment identifies tokenization and embedding inconsistencies as critical factors amplifying retrieval errors and semantic drift Ammann et al. (2025). Likewise, adversarial studies reveal structural weaknesses in RAG by exploiting tokenization biases: poisoned or strategically segmented documents can manipulate retrievability and induce misleading responses Choi et al. (2025). Collectively, these results underscore the need for tokenization-aware dataset curation, retriever design (e.g., hybrid lexical-dense or segmentation-robust retrieval), and evaluation, as well as RAG architectures that account for tokenization effects across ingestion, retrieval, and generation to prevent cascading errors from segmentation artifacts.

In contrast to prior work that primarily proposes static mitigations, we focus on making tokenization effects explicitly measurable and actionable. To this end, we introduce a tokenization risk scoring method that quantifies the likelihood of tokenization-induced degradation and use it as a control signal for adaptive RAG behavior.

2 Method

Specialized, low-frequency strings (e.g., model identifiers, diagnostic codes, and number-unit expressions) are frequently fragmented by subword tokenizers, which can distort both dense retrieval representations and the faithful reproduction of terminology during generation. Rather than treating tokenization as a static preprocessing detail, we treat it as an *actionable signal* within the system: we compute a tokenization risk score and use it to adapt the RAG pipeline on a per-query basis.

Concretely, we introduce (i) a lightweight *tokenization risk scoring method* that estimates whether a query or passage is likely to be harmed by fragmentation; (ii) a *tokenization-risk-based routing* mechanism (*RouteRAG*) that uses these scores to switch retrieval and, when needed, generation modality; and (iii) a compact *framework for diagnosing tokenization-linked RAG failures*, instantiated through the construction of *SpecialVocabBench*, a risk-stratified, passage-grounded QA dataset that enables attribution of errors to retrieval versus generation.

2.1 Tokenization risk scoring method

The tokenization-risk-scoring method audits how a given text is split by a tokenizer¹. Prior to tokenization, strings are normalized to reduce avoidable fragmentation (e.g., curly quotes, em-dashes, non-breaking spaces, soft hyphens). Two main types of scores are produced:

1. **Global (composite) risk scores** that summarize the whole sentence/passage.
2. **Local (per-word) risk scores** that expose the most-risky words.

Both types are combined in a user-configurable fashion. Below we give a thorough description of every metric that contributes to these scores. The following notations are used:

- S : the input string (after normalization) with length $|S|$ characters.
- T : total number of Llama-3 BPE tokens for S .
- For each character c_i we denote its Unicode grapheme cluster by g_i .
- $\mathcal{W} = \{w_k\}$: the set of *lexical words* extracted by the regular expression $[A-Za-z0-9_]+$. For a word w_k we write $|w_k|$ for its character length.
- The token-spans returned by the tokenizer are denoted $\mathcal{P} = \{(p_j^{\text{start}}, p_j^{\text{end}})\}_{j=1}^T$.

2.1.1 Global (sentence-level) indicators

The tokenization risk auditor first computes a vector of six elementary indicators $\mathbf{I} = (I_1, \dots, I_6)$. Each indicator is a ratio or a count that captures a distinct source of tokenization risk.

1. **Token density:** A high value of I_1 indicates that the tokenizer is producing many sub-word units for a relatively short string:

$$I_1 = \frac{T}{|S|} \times 100 \quad (\text{tokens per 100 characters}).$$

Range of I_1 : $\approx 0-\infty$ (typically 5–30); for smooth English $\approx 15-20$; for technical language $\approx 20-25$; alphanumeric identifiers $\approx 35+$, etc.

2. **Grapheme split rate:** A grapheme split occurs when a single Unicode grapheme cluster is broken across two adjacent tokens. Let G be the total number of such splits; then

$$I_2 = \frac{G}{T} \quad (\text{grapheme-split rate}).$$

Values close to 0 mean that the tokenizer respects grapheme boundaries. Example: An emoji like 🍕 (family emoji) consists of several characters but appears as a single emoji.

¹We used the Hugging-Face (HF) fast-tokenizer (https://huggingface.co/docs/transformers/main_classes/tokenizer) that emulates the Llama-3 tokenizer

3. **Number-unit split rate:** The auditor looks for patterns of the form “<number><unit>”, e.g. 12kg. If a split separates the numeric and the alphabetic part, it is counted as a *number-unit split*. Let U be the number of such splits; then

$$I_3 = \frac{U}{T} \quad (\text{number-unit split rate}).$$

4. **Hyphen-compound fragmentation rate:** Hyphenated compounds (e.g. well-known) are undesirable when the hyphen is tokenized separately from its neighbours. Let H be the number of hyphen-induced fragmentations detected by the auditor; then

$$I_4 = \frac{H}{T} \quad (\text{hyphen-fragmentation rate}).$$

5. **Capitalized entity fragmentation rate:** Capitalized entity fragmentation measures how often capitalized entities in a text (e.g., model names, part identifiers, acronyms) are broken across multiple tokens during tokenization. If E denotes the number of entities split into ≥ 3 tokens, then

$$I_5 = \frac{E}{T} \quad (\text{entity-fragmentation rate}).$$

6. **Long-word fertility:** Long words (by default $|w_k| \geq 8$ characters) are expected to be represented by a single token. For each long word we compute its *fertility* f_k , i.e. the number of tokens covering the word (cf. Rust et al., 2021). Let $F = \frac{1}{|\mathcal{W}_{\text{long}}|} \sum_{w_k \in \mathcal{W}_{\text{long}}} f_k$ be the average fertility; then

$$I_6 = \frac{F}{1} \quad (\text{long-word fertility}).$$

Values $I_6 > 1$ reveal that the tokenizer tends to over-segment long lexical items.

Composite global risk score The six indicators are linearly combined (with optional user-supplied weights $\mathbf{w} = (w_1, \dots, w_6)$) to obtain a scalar *global risk* R_{global} :

$$R_{\text{global}} = \frac{\sum_{i=1}^6 w_i I_i}{\sum_{i=1}^6 w_i}, \quad \mathbf{w} \geq 0, \mathbf{w} \neq \mathbf{0}. \quad (1)$$

If no explicit weights are supplied the auditor uses the default uniform weighting $w_i = 1$.

2.1.2 Local (per-word) risk scores

For each lexical word w_k the auditor computes a vector of six risk components $\mathbf{r}_k = (r_k^{(1)}, \dots, r_k^{(6)})$:

1. **Length-inflation risk:** penalises words whose token-to-character ratio exceeds the sentence-wide average (clip = clipping to $[0, 1]$):

$$r_k^{(1)} = \text{clip}\left(\frac{t_k}{|w_k|} - \frac{T}{|S|}, 0, 1\right),$$

where t_k is the number of tokens covering w_k .

2. **Grapheme-split risk:** 1 if any grapheme inside w_k is split across tokens, 0 otherwise:

$$r_k^{(2)} = \begin{cases} 1 & \text{if a split occurs inside } w_k, \\ 0 & \text{otherwise.} \end{cases}$$

3. **Hyphen-fragment risk:** 1 if a hyphen adjacent to w_k is tokenized separately:

$$r_k^{(3)} = \begin{cases} 1 & \text{if a hyphen split involves } w_k, \\ 0 & \text{otherwise.} \end{cases}$$

4. **Capitalized entity-fragment risk:** 1 if w_k is split into more than one token:

$$r_k^{(4)} = \begin{cases} 1 & \text{if } t_k > 1, \\ 0 & \text{otherwise.} \end{cases}$$

5. **Fertility risk:** the deviation of the word’s fertility from the ideal value 1:

$$r_k^{(5)} = \text{clip}(f_k - 1, 0, 1).$$

6. **Number-unit split risk:** 1 if w_k matches a “<number><unit>” pattern that is split:

$$r_k^{(6)} = \begin{cases} 1 & \text{if a number-unit split occurs inside } w_k, \\ 0 & \text{otherwise.} \end{cases}$$

The per-word risk R_k is obtained by a weighted sum of the six components (the default component weights sum to 1):

$$R_k = \sum_{j=1}^6 \alpha_j r_k^{(j)}, \quad \alpha_j = \frac{1}{6} \text{ (default),}$$

and is finally clipped to the interval $[0, 1]$.

2.1.3 Combination of global and local risks

The auditor provides five strategies to collapse the global risk R_{global} and the worst local risk $R_{\text{max}} = \max_k R_k$ into a single *final score* R_{final} :

1. **Aggregate** – use only the global component: $R_{\text{final}} = R_{\text{global}}$.
2. **Max** – use only the most dangerous word: $R_{\text{final}} = R_{\text{max}}$.
3. **Blend** – a convex combination controlled by $\alpha \in [0, 1]$ (default $\alpha = 0.7$): $R_{\text{final}} = \alpha R_{\text{global}} + (1 - \alpha) R_{\text{max}}$.
4. **Max-of** – the larger of the global risk and a scaled local risk, with scaling factor $\beta > 0$ (default $\beta = 1$): $R_{\text{final}} = \max(R_{\text{global}}, \beta R_{\text{max}})$.
5. **Both** – report *all three* numbers ($R_{\text{global}}, R_{\text{max}}, R_{\text{final}}$) and compute R_{final} using the *Blend* rule.

The user can select any of the above modes via the command-line options. By linearly aggregating these indicators with configurable weights,

the method can be tuned to emphasise any subset of failure modes (e.g. a production system that cares most about grapheme integrity may set $w_{\text{grapheme_split}}$ larger than the others).

The per-word risk decomposition further enables fine-grained diagnostics: researchers can retrieve the k most problematic tokens and inspect which component (length inflation, grapheme split, ...) dominates the risk for each token.

Finally, the flexible score mode selection lets users decide whether a single global risk is sufficient (aggregate), whether the worst offending token should dominate (max), or whether a balanced compromise is desired (blend or maxof).

The auditor² implements a transparent, mathematically-grounded suite of metrics for evaluating how a Llama-3-compatible BPE tokenizer treats a piece of text. By exposing both sentence-level composite risks and token-level contributions, the tool supports systematic auditing of dataset curation pipelines and downstream applications where token-level fidelity is critical.

2.2 Tokenization-risk-based routing

We propose *RouteRAG*, which uses tokenization risk to adapt RAG behavior. As shown in Figure 2, we compute risk for the query and retrieved passages. If risk is low, we use standard dense retrieval followed by LLM generation. If risk exceeds a threshold, we switch retrieval from dense (vector) search to lexical BM25, which is less sensitive to subword fragmentation; if risk remains high, we optionally switch generation from a text-only LLM to a VLM operating on the source page images associated with the retrieved passages (stored via metadata pointers). Thresholds are set using simple corpus statistics (average passage risk) to keep routing lightweight and production-feasible. Algorithm 1 outlines the method.

To systematically evaluate the effect of tokenization risk and the proposed routing strategy, we require a benchmark that isolates tokenization-sensitive failure modes. This motivates the construction of *SpecialVocabBench*.

2.3 Risk-stratified QA generation framework and SpecialVocabBench

We use tokenization risk score (Section 2.1) as a *selection signal* to construct a compact, passage-grounded evaluation set (henceforth called

²<https://github.com/Norbobert/TokenizationRiskAudit>

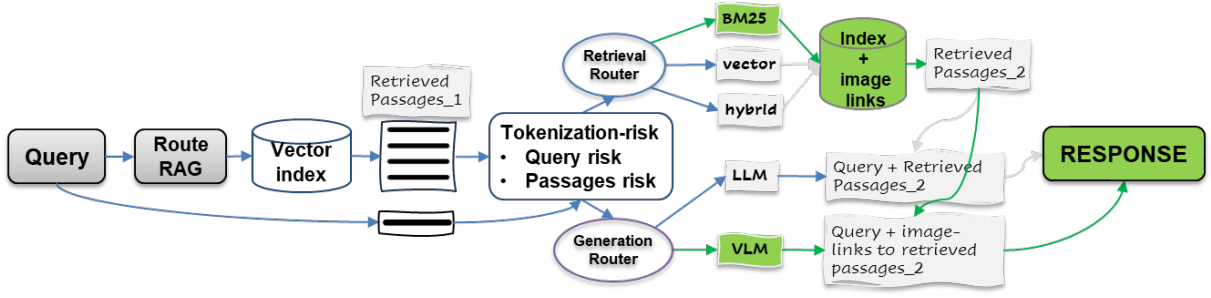


Figure 2: Tokenization-risk-based routing of retrieval and generation method in *RouteRAG*. The path in green shows a route via BM25 retrieval and VLM generation.

Algorithm 1 ROUTERAG: tokenization-risk-based routing for adaptive RAG

Require: Query q , dense retriever $\mathcal{R}_{dense}$, BM25 retriever $\mathcal{R}_{bm25}$, LLM generator $\mathcal{G}_{llm}$, optional VLM generator $\mathcal{G}_{vlm}$, risk scorer $\text{Risk}(\cdot)$, thresholds τ_q, τ_c , top- k

- 1: $r_q \leftarrow \text{Risk}(q)$ // query risk
- 2: $C \leftarrow \mathcal{R}_{dense}(q, k)$ // initial dense retrieval
- 3: $r_C \leftarrow \max_{c \in C} \text{Risk}(c)$ // aggregate context risk
- 4: **if** $r_q > \tau_q$ **or** $r_C > \tau_c$ **then**
- 5: $C \leftarrow \mathcal{R}_{bm25}(q, k)$ // retrieval routing: dense $\rightarrow$ BM25
- 6: **end if**
- 7: $r_C \leftarrow \max_{c \in C} \text{Risk}(c)$ // recompute after routing (if applied)
- 8: **if** $r_C > \tau_c$ **and** $\mathcal{G}_{vlm}$ available **then**
- 9: Retrieve page images linked to C via metadata pointers
- 10: $a \leftarrow \mathcal{G}_{vlm}(q, \text{images}(C))$ // generation routing: LLM $\rightarrow$ VLM
- 11: **else**
- 12: $a \leftarrow \mathcal{G}_{llm}(q, C)$ // standard generation
- 13: **end if**
- 14: **return** a

SpecialVocabBench) that enables us to isolate tokenization-linked failure modes in retrieval and generation. Concretely, we (i) build a corpus of retrieval-sized passages from real technical documents, (ii) score each passage for tokenization risk, and (iii) generate context-grounded QA pairs preferentially from high-risk passages while retaining a subset of low-risk passages for controlled comparison.

2.3.1 Document selection

To ensure realism, relevance, and domain diversity, we select three industrial manuals spanning air-conditioning, printers, and heat pumps. These documents are representative of real-world technical support scenarios and contain dense specialized vocabulary (e.g., model identifiers, diagnostic codes, number-unit expressions) as well as heterogeneous layouts (prose interleaved with tables, figures, diagrams, and procedural instructions), making them

well suited for studying tokenization-related RAG failures. Table 1 summarizes the selected documents, including their size and type.

The source manuals are internal (in-house) documents and cannot be publicly released. However, comparable technical manuals with similar structure and vocabulary are widely available online, and we expect our findings to generalize to these publicly accessible sources.

Domains	Aircon	Printer	Heatpump
# pages	155	78	47
Type	Service manual	Quickstart guide	Installation manual

Table 1: Domains and document-types in *SpecialVocabBench*.

2.3.2 PDF to text conversion

To obtain a high-fidelity textual representation suitable for indexing and passage selection, we convert PDFs to markdown using *LlamaCloud*³, using the *AgenticPlus* preset, chosen for robust handling of complex layouts (including multi-page tables, diagrams, and images). The resulting markdown is used for both retrieval indexing and risk scoring.

2.3.3 Chunking

We segment the markdown into retrieval-sized passages (“nodes”) using *LlamaIndex SentenceSplitter* with default settings (1024-token target; 200-token overlap). Chunks are typically page-aligned, depending on page density and splitting behavior. These nodes are the atomic units for retrieval and for passage-level tokenization risk scoring.

2.3.4 Generating context-grounded QA pairs

For each node, we compute a composite tokenization risk score (blend option; Section 2.1.3) and

³<https://cloud.llamaindex.ai>

rank nodes by risk. We then generate *context-grounded* questions with an LLM (GPT-5⁴; prompt in Appendix A), encouraging inclusion of specialized strings present in the passage to stress retrieval (matching rare/fragmented terms) and generation (faithful reproduction of terminology). A subset of question–context pairs is manually checked and corrected where necessary, focusing on both top-risk and bottom-risk passages (e.g., top-20 and bottom-20 by risk) to support controlled comparisons. Reference answers are generated with llama3.3-70b⁵.

2.3.5 Resulting corpus: SpecialVocabBench

The resulting dataset, *SpecialVocabBench*, contains 116 passage-grounded QA pairs across three domains (Table 2). Its risk-stratified design enables diagnostic analysis of RAG errors and evaluation of tokenization-risk-based routing (Sections 3 & 2.2).

The questions span a wide range, from factoid queries (e.g., *What is the maximum number of sheets that can be stapled for A4?*) to multi-step procedural requests (e.g., *Tell me the regular maintenance protocol for printer PRNT8-512SE*), and table-based queries requiring structured interpretation (e.g., *What is the part number of the 4-way valve in model ACON-9150LWSH-P7?*). These cover both short factual answers and longer instruction-style responses.

SpecialVocabBench is designed to isolate a specific and underexplored failure mode of RAG: errors arising from tokenization and specialized vocabulary (e.g., product identifiers and alphanumeric terms). While not covering all RAG challenges, this focused design enables controlled analysis of tokenization-sensitive failures, complementing broader benchmarks and providing a practical diagnostic tool for technical domains.

Domains	Aircon	Printer	Heatpump
# nodes	152	76	47
# QA pairs	50	21	45

Table 2: Number of nodes and QA pairs for each domain in *SpecialVocabBench*.

⁴OpenAI. (2025). GPT-5: <https://openai.com/gpt-5>

⁵<https://ollama.com/library/llama3.3:70b>

3 Experiments and Results

We evaluate (i) whether tokenization risk predicts downstream RAG failures and (ii) whether using risk as a control signal improves RAG via *RouteRAG* (Section 2.2). Experiments use *SpecialVocabBench* (Section 2.3.5), comprising 116 passage-grounded QA pairs from three industrial domains.

3.1 Setup

Systems. We compare (a) retrieval-free upper/lower bounds and (b) RAG variants. RAG uses top- $k=5$ contexts. *BaseRAG* performs dense vector retrieval over markdown passages; *BM25RAG* conducts sparse retrieval with BM25 over markdown; *HybridRAG* combines dense retrieval with BM25 via reciprocal rank fusion (Cormack et al., 2009). We also evaluate an index augmented with VLM-generated page descriptions (“MD+VLM”) in addition to text-only markdown (“MD”). *RouteRAG* applies tokenization-risk-based routing of retrieval (dense $\rightarrow$ BM25) and optionally generation (LLM $\rightarrow$ VLM on page images) as described in Section 2.2.

Models. Generation uses llama3.3-70b. VLM page descriptions and VLM answering use qwen3-v1-32b. Answer correctness is judged by gpt-oss:120b (OpenAI et al., 2025). All models called via ollama⁶.

Metrics. Retrieval quality is measured as semantic similarity between retrieved and reference contexts (higher is better). Answer quality is measured by *LLM-as-judge* (Zheng et al., 2023; Li et al., 2024, 2025; Gu et al., 2026) on a 1–5 scale (1 = worst, 5 = best); we report average values across three runs of the same system and provide standard deviations.

3.2 Tokenization risk distribution

We compute composite passage-level tokenization risk scores (blend-method; Section 2.1.3) for all nodes in each domain. Average risk is similar across domains, but the high-risk tail is clearly separated from the low-risk tail (Table 3), indicating substantial within-domain variation in tokenization fragility.

⁶<https://ollama.com/library/>

Nodes	Aircon	Printer	Heatp
all	0.273 $\pm$ 0.10	0.263 $\pm$ 0.07	0.273 $\pm$ 0.10
top-20	0.444 $\pm$ 0.01	0.371 $\pm$ 0.04	0.365 $\pm$ 0.06
last-20	0.114 $\pm$ 0.01	0.174 $\pm$ 0.04	0.185 $\pm$ 0.05

Table 3: Average tokenization risk scores (from no risk (0) to highest risk (1) with standard deviations) comparison for different node selections across the three domains in *SpecialVocabBench*.

3.3 Risk predicts RAG degradation

To test whether tokenization risk is predictive of failure, we evaluate *BaseRAG* on two aircon subsets: QA pairs grounded in the top-20 highest-risk nodes (Set A) versus the bottom-20 lowest-risk nodes (Set B). High-risk questions cause markedly lower retrieval similarity and lower answer correctness (Table 4), showing that tokenization risk correlates with retrieval mismatch and downstream answer degradation.

Metric	Set A (highrisk)	Set B (lowrisk)
avg_ctxtsimil $\uparrow$	0.835 $\pm$ 0.00	0.933 $\pm$ 0.03
avg_correctness $\uparrow$	2.79 $\pm$ 0.03	3.55 $\pm$ 0.10

Table 4: Comparison of BaseRAG on top-20 high tokenization risk QA pairs vs. bottom-20 lowest tokenization risk QA pairs.

3.4 Static baselines on *SpecialVocabBench*

Table 5 summarizes performance across the full benchmark. Retrieval-free baselines establish bounds: zero-shot LLM performs poorly, while providing the gold reference context (*Query+refCntxt*) or gold reference image (*VLM+ref-image*) yields high correctness (≈ 4.1), confirming that errors under RAG largely stem from retrieval/grounding rather than missing knowledge.

When retrieval is required, *BaseRAG* degrades substantially (avg. 3.11 correctness). Adding lexical signals helps: *BM25RAG* is strongest among text-only RAG variants (3.72-3.77) depending on index, while *HybridRAG* (MD+VLM) yields the best static RAG result (3.89), indicating that adding VLM page descriptions to the index further improves correctness.

3.5 RouteRAG: risk-based routing improves robustness

We next isolate the effect of adaptive routing compared to the static baselines in Table 5. We evalu-

ate *RouteRAG* (Section 2.2), which uses tokenization risk as a trigger for retrieval routing (dense $\rightarrow$ BM25) and optionally generation routing (LLM $\rightarrow$ VLM). As shown in Table 6, retrieval routing improves both retrieval accuracy and correctness relative to *BaseRAG* (0.89 $\rightarrow$ 0.94 retrieval; 3.11 $\rightarrow$ 3.32 correctness). Adding generation routing yields the largest gain, improving correctness to 3.78 while keeping retrieval accuracy unchanged. These results support the central claim: tokenization risk functions effectively as a lightweight control signal for adaptive RAG. Across the evaluation set, the routing mechanism selected vector retrieval in 44 cases and BM25 in 72 cases, indicating that a majority of queries were identified as tokenization-sensitive. For generation, the system used the text-only LLM in 34 cases and switched to the VLM in 82 cases, highlighting the prevalence of high-risk contexts where visual grounding is beneficial.

System	Retrieval $\uparrow$	Response $\uparrow$
LLM-direct		
Zero-shot	na	1.92 $\pm$ 0.08
Query+refCntxt	1.0	4.16 $\pm$ 0.02
VLM+ref-image	1.0	4.13 $\pm$ 0.03
RAG from MD		
BaseRAG	0.89	3.11 $\pm$ 0.05
BM25RAG	0.94	3.72 $\pm$ 0.01
HybridRAG	0.94	3.63 $\pm$ 0.09
RAG from MD + VLM description		
BaseRAG	0.86	3.48 $\pm$ 0.08
BM25RAG	0.88	3.77 $\pm$ 0.02
HybridRAG	0.87	3.89 $\pm$ 0.02

Table 5: Comparison of various response generation methods evaluated on *SpecialVocabBench*.

System	Retrieval $\uparrow$	Response $\uparrow$
Retrieval-routing: vector [44] BM25 [72]		
RouteRAG	0.94	3.32 $\pm$ 0.02
+Generation routing: LLM [34] VLM [82]		
RouteRAG	0.94	3.78 $\pm$ 0.02

Table 6: RouteRAG performance with routing frequencies for retrieval (vector vs. BM25) and generation (LLM vs. VLM) evaluated on *SpecialVocabBench*.

3.6 Model sensitivity under oracle grounding

To complement the system-level analysis, we examine how answer quality varies across model families when retrieval is not a limiting factor. We evaluate several LLMs and VLMs under oracle

grounding, where the reference passage or image is provided directly (retrieval = 1.0).

LLMs. Table 7 reports results for multiple LLMs given the reference context. All models achieve high correctness, but performance scales with model size. Smaller models (e.g., Mistral-small-22B) reach 3.85, while larger models such as Qwen3-next-80B achieve up to 4.32, with more answers rated ≥ 4.0 . This shows that model capacity remains important even without retrieval noise.

VLMs. Table 8 shows results for VLMs given reference images. Performance is comparable to or slightly stronger than text-only LLMs, with Qwen3.6-36B-VL reaching 4.27 correctness, indicating that visual grounding helps with structured or tokenization-sensitive content.

Implications. These results provide two insights: (i) they establish an upper bound on *SpecialVocabBench* (≈ 4.3 correctness), quantifying the gap from retrieval and tokenization errors; and (ii) they support *RouteRAG*, as VLM routing in high-risk cases is a principled design choice rather than a heuristic.

LLM	Response $\uparrow$	$\# \geq 4.0$
Mistral-small-22b	3.85 (0.07)	79.0 (1.4)
Mistral-large-123b	4.07 (0.03)	85.0 (0.8)
Llama3.3-70b	4.16 (0.02)	88.0 (1.6)
Qwen3-32b	4.21 (0.01)	89.6 (0.9)
Qwen3.6-36b	4.27 (0.04)	92.0 (0.8)
Qwen3-next-80b	4.32 (0.07)	93.3 (2.1)

Table 7: LLM performance with oracle reference context (retrieval = 1.0) evaluated on *SpecialVocabBench*.

VLM	Response $\uparrow$	$\# \geq 4.0$
Qwen3-vl-32b	4.13 (0.03)	80.0 (1.4)
Qwen3.6-36b	4.27 (0.04)	92.0 (0.8)

Table 8: VLM performance with oracle reference images (retrieval = 1.0) evaluated on *SpecialVocabBench*.

Notably, the gap between oracle performance (≈ 4.3) and standard RAG (≈ 3.1 – 3.9) quantifies the error introduced by retrieval and tokenization effects, highlighting the importance of adaptive mitigation.

3.7 Discussion: Tokenization risk as a practical control signal

Our results suggest the following interpretation: tokenization risk serves not only as a diagnostic metric but as a *control signal* indicating when dense-retrieval RAG is likely to fail on specialized strings. High-risk content consistently yields lower retrieval similarity and answer correctness, suggesting that subword fragmentation disrupts both embedding-based matching and downstream terminology handling. *RouteRAG* operationalizes this signal by applying mitigations only when needed. Retrieval routing (dense $\rightarrow$ BM25) improves both accuracy and correctness, showing that lexical matching is a robust fallback when dense similarity is less effective. Adding generation routing (LLM $\rightarrow$ VLM over linked images) further boosts correctness, indicating that visual grounding can bypass tokenization-induced distortions. Overall, tokenization risk acts as a lightweight gating mechanism for adaptive RAG, enabling targeted improvements without retraining or major architectural changes.

4 Conclusions

We studied a common failure mode of retrieval-augmented generation (RAG) in technical domains: specialized, low-frequency strings are often fragmented by subword tokenizers, degrading dense retrieval matching and the faithful reproduction of terminology during generation.

To address this, we introduced a **tokenization risk score** that quantifies tokenization fragility for both queries and passages, and showed that high-risk content is associated with lower retrieval similarity and reduced answer correctness. Building on this signal, we proposed *RouteRAG*, which uses tokenization risk as a lightweight control variable to adapt the pipeline: high-risk cases trigger routing from dense retrieval to BM25 and, when needed, from text-only generation to a VLM operating on linked page images.

Across *SpecialVocabBench*, a risk-stratified set of 116 passage-grounded QA pairs from industrial manuals, static baselines confirm that specialized vocabulary stresses standard RAG, while hybrid retrieval and VLM-augmented indexing offer partial mitigation. Crucially, risk-based routing yields the most consistent gains, improving both retrieval accuracy and answer correctness without retraining models.

Limitations

Tokenization risk scores are computed with respect to a specific tokenizer (Llama-3-compatible BPE). While the absolute values may vary across tokenizers or model families, the underlying phenomena (e.g., fragmentation of specialized strings) are intrinsic to subword tokenization. In practice, applying the method to new settings primarily requires lightweight recalibration of routing thresholds.

SpecialVocabBench is designed to isolate a targeted and underexplored class of RAG failures related to specialized vocabulary and tokenization sensitivity. It does not aim to cover the full spectrum of RAG challenges (e.g., open-domain reasoning or long-context synthesis). Rather than a limitation, this focused design enables controlled analysis of a failure mode that is often confounded in broader benchmarks. We view *SpecialVocabBench* as complementary to existing evaluations and as a diagnostic tool for developing tokenization-aware methods.

RouteRAG includes an optional multimodal routing component that assumes access to page images and document–page alignment metadata. While such inputs are available in many document-centric pipelines (e.g., PDF-based systems), they may not be present in purely text-based settings. In these cases, the retrieval-routing component (dense $\rightarrow$ BM25) remains fully applicable and accounts for a substantial portion of the observed gains.

Answer quality is evaluated using an *LLM-as-judge* protocol, which can introduce variability due to model and prompt choice. To mitigate this, we report averaged results across runs. Retrieval quality is measured via embedding-based similarity, which may not capture all notions of usefulness but provides a consistent proxy for grounding quality across systems.

Ethics Statement

This work uses technical manuals and automatically generated QA data; no personal or user-generated data was involved. The documents are internal technical materials and do not contain sensitive information.

AI assistants were used for question generation and development support. All generated outputs were reviewed and validated by the authors, who take full responsibility for the final content.

No crowdworkers or external annotators were used; all annotation and verification were per-

formed by the authors.

While our method improves robustness of RAG systems, care should be taken when deploying such systems in safety-critical domains.

Use of AI assistants. We used AI-based assistants to support parts of the development process, including code generation and drafting. All outputs were carefully reviewed and validated by the authors, and the final content and conclusions of this paper are solely the responsibility of the authors.

References

- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leueling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Schulze Buschhoff, Charvi Jain, Alexander Arno Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, and 2 others. 2024. Tokenizer Choice for LLM Training: Negligible or Crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.
- Lukas Ammann, Sara Ott, Christoph R. Landolt, and Marco P. Lehmann. 2025. [Securing RAG: A risk assessment and mitigation framework](#). In *Proceedings of the IEEE Swiss Conference on Data Science (SDS)*, pages 127–134.
- Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. 2024. [Tokenization Falling Short: On Subword Robustness in Large Language Models](#). In *Findings of EMNLP*, pages 1582–1599.
- Chanwoo Choi, Jinsoo Kim, Sukmin Cho, Soyeong Jeong, and Buru Chang. 2025. [The RAG Paradox: A Black-Box Attack Exploiting Unintentional Vulnerabilities in Retrieval-Augmented Generation Systems](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*.
- Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. [CANINE: Pre-Training an Efficient Tokenization-Free Encoder for Language Representation](#). *Transactions of the Association for Computational Linguistics*, 10:73–91.
- Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. [Reciprocal rank fusion outperforms condorcet and individual rank learning methods](#). In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759, Boston, MA, USA. ACM.
- Renato Geh, Zilei Shao, and Guy Van Den Broeck. 2025. [Adversarial Tokenization](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20738–20765, Vienna, Austria. Association for Computational Linguistics.

- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Zhouchi Lin, Bowen Zhang, Lionel Ni, Wen Gao, Yuanzhuo Wang, and Jian Guo. 2026. [A survey on LLM-as-a-Judge](#). *The Innovation*, page 101253.
- David A. Haslett. 2025. [Tokenization changes meaning in large language models: Evidence from Chinese](#). *Computational Linguistics*, 51(3):785–814.
- Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhat-tacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. 2025. [From Generation to Judgment: Opportunities and Challenges of LLM-as-a-Judge](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791, Suzhou, China. Association for Computational Linguistics.
- Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. [LLMs-as-Judges: A Comprehensive Survey on LLM-based Evaluation Methods](#). *Preprint*, arXiv:2412.05579.
- OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. [gpt-oss-120b & gpt-oss-20b Model Card](#). *Preprint*, arXiv:2508.10925.
- Sachin Pawar, Manoj Apte, Kshitij Jadhav, Girish Keshav Palshikar, and Nitin Ramrakhiyani. 2025. [Broken words, broken performance: Effect of tokenization on performance of LLMs](#). In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 372–385, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.
- Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. 2025. [Toward a Theory of Tokenization in LLMs](#). *Preprint*, arXiv:2404.08335.
- Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. [What are you token about? dense retrieval as distributions over the vocabulary](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2481–2498, Toronto, Canada. Association for Computational Linguistics.
- Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.
- Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. [How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural Machine Translation of Rare Words with Subword Units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Ziqin Luo, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2025. [Tokenization Matters! Degrading Large Language Models through Challenging Their Tokenization](#). *Preprint*, arXiv:2405.17067.
- Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2025. [A Theory for Token-Level Harmonization in Retrieval-Augmented Generation](#). In *The Thirteenth International Conference on Learning Representations (ICLR)*.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. [ByT5: Towards a Token-Free Future with Pre-Trained Byte-to-Byte Models](#). *Transactions of the Association for Computational Linguistics*, 10:291–306.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-bench and chatbot arena](#). In *Advances in Neural Information Processing Systems (NeurIPS 2023), Datasets and Benchmarks Track*. Neural Information Processing Systems Foundation.

Appendix

A Question generation prompt

We leveraged GPT-5 to generate questions for each text passage from the source documents using the following prompt:

You are an expert in formulating challenging questions which are based on a given text passage. Given the text passage, generate questions which are challenging for a retrieval-augmented generation system faced with any special words in the given text passage, such as specific error codes, product IDs, abbreviations, cap-change words or rare words and include these special words in your

question. Now generate *num_questions_per_node* questions (just the questions, no numbering or anything else) based on the following passage:

{text}

Questions:"""