RAGTurk: Best Practices for Retrieval Augmented Generation in Turkish

Süha Kağan Köse[1], Mehmet Can Baytekin[1], Burak Aktaş[1], Bilge Kaan Görür[1], Evren Ayberk Munis[2], Deniz Yılmaz[3], Muhammed Yusuf Kartal[4], Çağrı Toraman[3]
[1]Roketsan Inc., Artificial Intelligence Technologies Unit, Turkey
[2]Politecnico di Torino, Italy
[3]Middle East Technical University, Computer Engineering Department, Turkey
[4]TOBB University of Economics and Technology, AI Engineering Department, Turkey
kagan.kose@roketsan.com.tr, can.baytekin@roketsan.com.tr
burak.aktas@roketsan.com.tr, kaan.gorur@roketsan.com.tr
evrenayberk.munis@studenti.polito.it, deniz.yilmaz_12@metu.edu.tr
m.kartal@etu.edu.tr, ctoraman@metu.edu.tr

Abstract

Retrieval-Augmented Generation (RAG) enhances LLM factuality, yet design guidance remains English-centric, limiting insights for morphologically rich languages like Turkish. We address this by constructing a comprehensive Turkish RAG dataset derived from Turkish Wikipedia and CulturaX, comprising question–answer pairs and relevant passage chunks. We benchmark seven stages of the RAG pipeline—from query transformation and reranking to answer refinement—without task-specific fine-tuning. Our results show that complex methods like HyDE maximize accuracy (85%) that is considerably higher than the baseline (78.70%). Also a Pareto-optimal configuration using Cross-encoder Reranking and Context Augmentation achieves comparable performance (84.60%) with much lower cost. We further demonstrate that over-stacking generative modules can degrade performance by distorting morphological cues, whereas simple query clarification with robust reranking offers an effective solution.[1]

1 Introduction

Large Language Models (LLMs) perform strongly across many NLP tasks, yet they struggle when queries require current, domain-specific, or verifiable information. Retrieval-Augmented Generation (RAG) mitigates these limitations by incorporating external evidence during generation (Lewis et al., 2020). Over time, RAG has developed into a modular pipeline—spanning from query transformation to answer refinement—whose components collectively shape system performance (Gupta et al., 2024; Zhao et al., 2024; Liu et al., 2025a). Many studies have improved individual stages of this pipeline: dense retrievers (Karpukhin et al., 2020), late-interaction models (Khattab and Zaharia, 2020), LLM-based query expansion (Gao et al., 2022; Li et al., 2025), cross-encoder reranking (Nogueira and Cho, 2019), and hierarchical retrieval (Tao et al., 2025). Recent work also emphasizes iterative reasoning in RAG (Asai et al., 2024; Jiang et al., 2023b) and holistic evaluation metrics beyond answer accuracy alone (Yu et al., 2024; Es et al., 2024).

However, nearly all prior RAG research targets English. For morphologically rich and moderately resourced languages like Turkish, the behavior of RAG systems remains largely unknown. Turkish morphology, flexible word order, and variation across sources introduce retrieval and grounding challenges not reflected in English benchmarks. Existing work examines isolated components—retrievers (Bikmaz et al., 2025), cultural QA (Simsek, 2025), embeddings (Ezerceli et al., 2025), and hallucination detection (Taş et al., 2025)—but not full pipelines evaluated end-to-end on curated Turkish benchmarks. Relevant background on Turkish NLP challenges and benchmark evaluation appears in (Hakkani-Tür et al., 2002; Oflazer, 2014; Umutlu et al., 2025).

Meanwhile, the broader RAG ecosystem is shifting toward pipeline-level optimization. Frameworks such as AutoRAG (Kim et al., 2024), DSPy-RAG (Khattab et al., 2024), GraphRAG (Edge et al., 2024) and RAGSmith (Kartal et al., 2025) show that performance depends on coordinated component interaction rather than any single module. Yet these systems are also English-centric, leaving open questions about cross-linguistic transferability.

To address this gap, we present the first systematic, end-to-end evaluation of RAG pipeline components for Turkish on a dataset consisting of two parts

[1]Links to our datasets and source code are available at: https://github.com/metunlp/ragturk

with similar properties (well-formed Turkish text and grounded question–answer pairs with verifiable evidence passages) but different sources. Using 4,891 Turkish Wikipedia articles and 6,305 Turkish web articles derived from CulturaX (Nguyen et al., 2024), we assess seven core pipeline stages under a unified protocol.

Our contributions are:

- **A comprehensive Turkish RAG benchmark:** we construct a unified dataset sourced from Turkish Wikipedia and CulturaX, generating grounded question–answer pairs with gold evidence passages across factual and interpretive question types.
- **A systematic, end-to-end pipeline study for Turkish:** we benchmark seven core stages of a modern RAG stack (query transformation, candidate re-ranking, filtering & selection, context augmentation, condensation, prompt composition, and answer refinement) under a unified protocol.
- **Optimized recipes and reproducible release:** we distill actionable strategies—identifying a Pareto-optimal configuration that balances accuracy and efficiency while highlighting the risks of over-stacking LLM modules—and release the datasets, prompts, configuration files, and evaluation scripts to support reproducible Turkish-RAG research.

2 Related Work

Retrieval-Augmented Generation (RAG) strengthens Large Language Models (LLMs) with grounded, domain-specific information. Since Lewis et al. (Lewis et al., 2020), RAG has developed into a modular pipeline where choices across components (from query transformation to retrieval, reranking/selection, context construction/condensation, prompting, and answer refinement) jointly determine performance (Gupta et al., 2024; Zhao et al., 2024; Liu et al., 2025a). Recent evaluation work correspondingly argues for holistic assessment beyond generation quality, emphasizing retrieval effectiveness, grounding, attribution, and factual reliability (Yu et al., 2024; Es et al., 2024).

Retriever Architectures and Query Transformation. Dense retrieval replaces sparse lexical matching with neural representations; DPR (Karpukhin et al., 2020) demonstrates strong gains over BM25 on knowledge-intensive tasks, and hybrid dense+lexical approaches further improve robustness across query types (Lin et al., 2021). Complementary work improves recall via query reformulation/expansion: HyDE (Gao et al., 2022) generates hypothetical documents to bridge lexical gaps, while surveys of query expansion and multi-query strategies show consistent benefits for ambiguous or underspecified queries (Li et al., 2025).

Candidate Re-ranking, Context Selection, and Condensation. Reranking provides finer-grained relevance estimates after retrieval; cross-encoders (often BERT-based) remain the standard for high-precision ranking (Nogueira and Cho, 2019). Hierarchical/structured approaches such as TreeRAG (Tao et al., 2025) and related context-selection/condensation methods (e.g., selective extraction, compression, ordering) aim to pack the most useful evidence into limited context windows.

Answer Refinement and Self-Reflective RAG. Beyond "retrieve-then-generate," self-reflective methods such as Self-RAG (Asai et al., 2024) and FLARE (Jiang et al., 2023b) let models check support, trigger additional retrieval, and revise answers, improving factuality and robustness via iterative retrieval–generation.

System-Level Optimization: AutoRAG, DSPy-RAG, and GraphRAG. Recent work optimizes RAG end-to-end rather than tuning single components: AutoRAG (Kim et al., 2024) automates configuration over retrievers, chunking, query expansion, rerankers, prompts, and post-generation modules; DSPy-RAG (Khattab et al., 2024) treats RAG assembly as optimization over declarative modules; and GraphRAG (Edge et al., 2024) uses graph-based indexing and hierarchical retrieval to exploit document structure. These systems highlight strong component interactions, but are evaluated largely on English, leaving open questions for morphologically rich and underrepresented languages.

RAG and Retrieval in Turkish. A comprehensive survey of Turkish NLP resources (Çöltekin et al., 2023) provides essential background on corpora and lexical resources for the language. RAG research for Turkish is emerging: Bıkmaz et al. (Bikmaz et al., 2025) analyze retrievers and rerankers for Turkish QA, and Şimşek (Simsek,

2025) compares RAG against fine-tuning for culturally grounded QA. Turkish retrieval resources such as TurkEmbed4Retrieval (Ezerceli et al., 2025) emphasize language-tailored embeddings, while Turk-LettuceDetect (Taş et al., 2025) highlights the need for grounded, verifiable outputs. In addition, recent Turkish LLM benchmarking efforts such as TurkBench (Toraman et al., 2026) include evaluations of retrieval-augmented generation (RAG); however, they remain limited to general-purpose benchmarking and do not target the task-specific retrieval and reasoning challenges considered here.

However, most Turkish studies focus on isolated components rather than full pipelines, and do not systematically test how design choices transfer to Turkish datasets (e.g., Wikipedia articles and broad-coverage web text such as CulturaX) under rich morphology and source-dependent variation.

Positioning our work. To address the lack of non-English RAG benchmarks, we present the first systematic, end-to-end study of Turkish RAG pipelines. We construct a two-part dataset sourced from Turkish Wikipedia and CulturaX, and evaluate seven core pipeline stages—from query transformation to answer refinement—under a unified protocol. Our analysis identifies Pareto-optimal configurations that balance accuracy with efficiency, offering actionable "recipes" for Turkish retrieval. We release these resources and findings to support reproducible research in morphologically rich languages.

3 Dataset Construction

To ensure broad coverage of real-world retrieval scenarios, we construct a unified Turkish RAG benchmark comprising two complementary parts that share high standards for text quality and answerability but differ in source characteristics: the *Web Part (CulturaX)* is a diverse collection of Turkish web pages derived from CulturaX (Nguyen et al., 2024), filtered to retain contentful text across a wide range of topics (e.g., everyday life, entertainment, news), while the *Wikipedia Part* consists of Turkish Wikipedia articles providing encyclopedic reference text with stronger emphasis on biography, STEM, and history.

3.1 Corpus Acquisition and Filtering

To ensure valid evaluation, we filter raw crawls to keep only contentful, answerable documents. For the Web Part, we start from CulturaX and sample candidate Turkish pages. We then apply a two-stage filtering procedure guided by LLM-based judgments to ensure high retrieval utility:

1. *URL-only filtering (triage).* Given only the base website URL, an LLM estimates whether the site likely contains *valuable, informational* content—operationally, pages with factual statements that could answer user queries (*valuable*) and substantive prose rather than navigation menus, link lists, or boilerplate (*informational*). This stage acts as a low-cost pre-filter to exclude low-value page types such as pure landing pages, navigation hubs, or aggregator farms.

2. *Content-based filtering (page-level quality).* For pages passing stage (1), we fetch the page text and apply a second LLM filter that checks *coherence*, *content depth*, and *utility*. We retain pages that are understandable and content-rich (e.g., blog posts, forum discussions, news articles) with *good quality*. We reject pages that are *spam* (keyword stuffing, bots), *thin* (insufficient content), or *boilerplate* (largely non-textual content).

Prompt design for filtering. Filtering prompts are designed to retain answerable passages while excluding content that trivially breaks evaluation. We use gemini 2.5 flash (Google, 2025) for both stages; full prompts are in Appendix A (see Prompt A.1 and Prompt A.2).

Final Web corpus. After filtering, we retain 6,305 web pages. We convert each page to Markdown by preserving the main textual sections and mapping prominent HTML headings to Markdown headers.

Website Frequency Analysis. We report the base-domain frequency to ensure the Web Part is not dominated by a single source. The top domains include popular sites like sikayetvar.com and haberler.com, but the top 10 domains cover only ∼19.6% of documents, ensuring diversity (see Appendix B, Table 5 for full list).

Wikipedia Articles. For the Wikipedia articles, acquisition is straightforward given the structured nature of the source. We randomly sample Turkish Wikipedia pages, exclude short articles (< 300 chars), and retrieve plain-text sections. After filtering, we retain 4,891 articles and convert them to Markdown, mapping sections to headers.

Table 1: Topic statistics per dataset part and overall totals. Web/Wikipedia percentages are computed w.r.t. the full dataset (N=11,196).

Topic	Web	Wikipedia	Total	
	%	%	#	%
Entertainment	10.7	8.1	2,104	18.8
Biography	1.4	13.9	1,712	15.3
Everyday Life	12.4	0.2	1,414	12.6
STEM	5.2	6.6	1,322	11.8
Politics	9.9	1.6	1,298	11.6
Professional	7.4	1.0	939	8.4
History	3.8	4.5	925	8.3
Organizations	3.4	2.2	627	5.6
Geography	1.0	3.7	525	4.7
Humanities	0.9	1.9	309	2.8
Uncategorized	0.1	0.0	21	0.2
Total	**56.3**	**43.7**	**11,196**	**100.0**

3.2 Header-Aware Chunking

We apply a header-aware chunking strategy. Each chunk inherits document and section context (e.g., page title and section path). Long segments are split if they exceed 1,000 characters. We tried different thresholds and found this character limit provides the best balance between (i) preserving enough local context for answerability and (ii) limiting topic drift. This tokenizer-agnostic limit is particularly stable for Turkish, where agglutination packs more information into fewer whitespace-delimited tokens.

3.3 Topic Categorization

We annotate each document with a unified topic label using an LLM-based classifier using again gemini 2.5 flash (Google, 2025) as the LLM provider. We adapt our categorization approach and prompt design considerations from prior work on large-scale dataset construction (Gao et al., 2020; Soldaini et al., 2024; Weber et al., 2024; Penedo et al., 2024; Wenzek et al., 2020; Elazar et al., 2023). Full prompts are in Appendix A (see Prompt A.3).

Topic taxonomy. We define 10 broad topic categories (Table 1). The distribution highlights the complementary nature of the dataset: The *Web Part* leans towards *Everyday Life*, *Entertainment*, and *Politics*, reflecting the conversational web. The *Wikipedia Part* has higher coverage of *Biography*, *STEM*, and *History*. Taken together, the dataset spans the full range of user queries, from checking facts on public figures to navigating forum advices.

Table 2: Corpus and QA statistics for the complementary parts.

Statistic	Web Part	Wikipedia Part	Total
Articles	6,305	4,891	11,196
Characters	9,933,523	22,821,895	32,755,418
Char./Article	1,575.50	4,666.10	2,925.46
Chunks	15,985	42,304	58,289
Chunks/Article	2.54	8.65	5.21
Char./Chunk	695.61	598.81	561.97
Questions	10,682	9,777	20,459
Questions/Article	1.69	2.00	1.83
Factual	6,522	5,196	11,718
Interpretation	4,160	4,581	8,741

3.4 Question–Answer Pair Generation

We use gpt-oss:120B (OpenAI, 2025) for generation and validate with gemini-2.5-flash (Google, 2025) as an auxiliary consistency check. This cross-model verification reduces obvious hallucinations and off-topic generations, though it does not guarantee perfect grounding. We adapt our QA generation approach and prompt design considerations from prior work on large-scale dataset construction (Rajpurkar et al., 2016; Yang et al., 2018; Bloom et al., 1956; Anderson and Krathwohl, 2001; Zheng et al., 2023b; Liu et al., 2023). Full prompts are in Appendix A (see Prompt A.4). Table 2 reports the final statistics.

Interpreting corpus and QA statistics. The statistics in Table 2 confirm that the two parts offer complementary structural challenges. The *Web Part* contains more documents but with shorter average length, emphasizing precision in a broad search space. The *Wikipedia Part* contains fewer but much longer documents, requiring effective passage retrieval within dense, sectioned text. By covering both, and maintaining a balanced mix of *Factual* and *Interpretation* questions, the benchmark provides a robust testbed for Turkish RAG systems across the spectrum of quality Turkish text.

4 Methodology and Optimization

Optimizing RAG pipelines requires navigating a vast combinatorial space of design choices. To address this, we adopt a two-step approach: first, we define a comprehensive design space of candidate methods (Section 4.1); second, we employ a budgeted genetic search (Section 4.2) to efficiently identify high-performing configurations without exhaustive enumeration.

4.1 RAG Design Space

We evaluate a modular RAG pipeline (Figure 1) and vary methods within seven technique families while holding constant the rest of the system (chunking policy, index configuration, and prompt structure) to isolate which design choices drive performance.

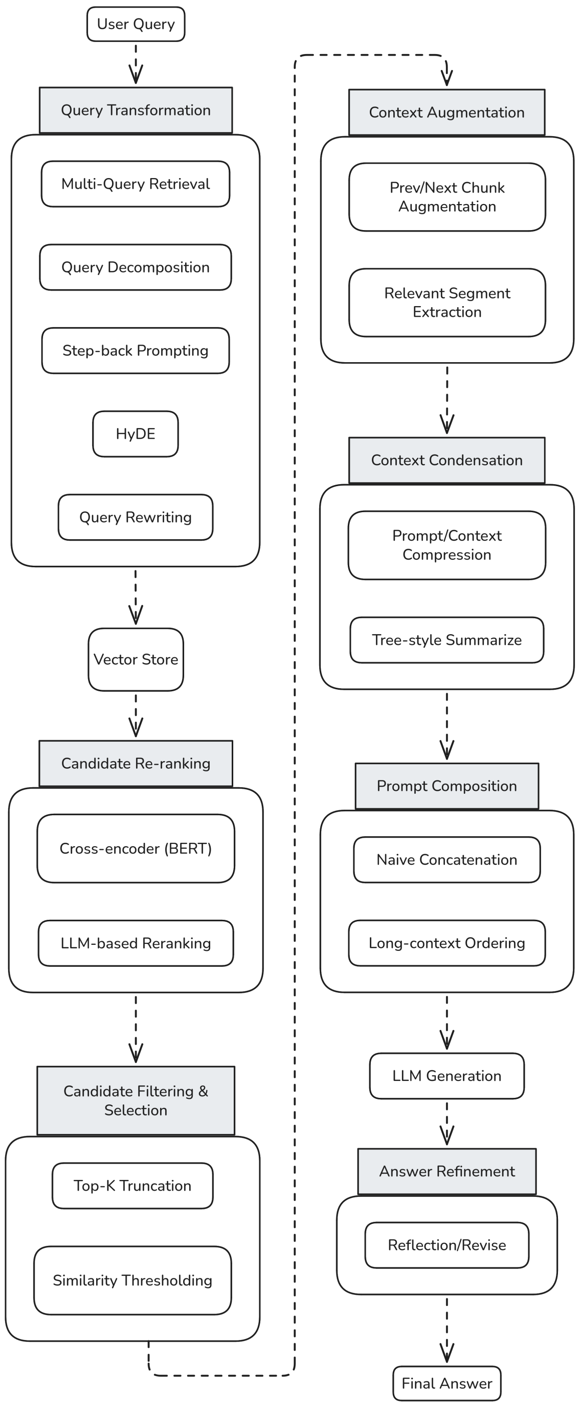

Figure 1: Overview of the RAG design space.

Selection rationale. Because the RAG literature is rapidly expanding, we focus on a representative set of techniques that cover the most common control points in end-to-end RAG (query-side, retrieval/reranking, context selection/compression, and generation-time refinement), are widely used or frequently reported as effective, and can be implemented reproducibly as modular components under a consistent evaluation protocol. A consistent protocol is vital for emerging ecosystems like Turkish (Bikmaz et al., 2025; Simsek, 2025; Ezerceli et al., 2025; Taş et al., 2025; Hakkani-Tür et al., 2002; Ezerceli et al., 2025; Umutlu et al., 2025), where rigorous standards are often lacking (Umutlu et al., 2025). For techniques that require prompts (e.g., rewriting/decomposition, HyDE-style generation, reflection/revision, LLM reranking), we use the default prompts and hyperparameters recommended in the corresponding papers.

Query transformation. *Multi-Query Retrieval:* Generate multiple semantically diverse rewrites of the user query and retrieve for each, then merge results to improve recall (Rackauckas, 2024). *Query Decomposition:* Break a complex question into simpler sub-questions, retrieve for each part, and aggregate evidence before answering (Zheng et al., 2023a). *Step-back Prompting:* Ask a more general "step-back" question to retrieve high-level background context that helps answer the original query (Zheng et al., 2023a). *HyDE:* Synthesize a hypothetical answer document from the query and use it as a retrieval query to better match relevant passages (Gao et al., 2022). *Query Rewriting / LLM-based expansion:* Rewrite and/or expand the query with additional keywords and paraphrases to reduce lexical mismatch and improve retrieval (Wang et al., 2023; Mao et al., 2021).

Candidate re-ranking. *Cross-encoder reranking (BERT):* Score each (query, passage) pair jointly with a cross-encoder and reorder retrieved passages by predicted relevance (Nogueira and Cho, 2019). *LLM-based reranking:* Use an instruction-tuned LLM to judge relevance of candidates and reorder/retain the most useful passages for answering (Lewis et al., 2020).

Candidate filtering & selection. *Top-K truncation:* Keep only the K highest-ranked retrieved passages to control context budget and reduce noise (Karpukhin et al., 2020; Lewis et al., 2020). *Similarity thresholding:* Discard candidates below a similarity cutoff to avoid injecting weakly related context into generation (Karpukhin et al., 2020; Lewis et al., 2020).

Context augmentation. *Prev/next chunk augmentation (multi-granular context):* Expand a retrieved chunk with its neighbors or multi-scale spans to recover lost local coherence from segmentation (Liu et al., 2025b). *Relevant segment extraction:* Extract only the most relevant spans within retrieved documents to maximize signal per token in the context window (Liu et al., 2025b).

Context condensation. *Prompt/context compression & redundancy pruning:* Compress passages and remove redundant content so more unique evidence fits within the model's context length (Jiang et al., 2023a; Li et al., 2023). *Tree-style summarize / iterative refine:* Summarize evidence hierarchically or refine a running summary over multiple steps to preserve key facts under tight budgets (Madaan et al., 2023).

Prompt composition. *Naive concatenation (RAG baseline):* Concatenate the selected passages into a single context block and prompt the model to answer grounded in that context (Lewis et al., 2020). *Long-context ordering ("lost in the middle"):* Vary the ordering/placement of evidence in long prompts to study position effects where mid-context facts are under-attended (Liu et al., 2024).

Answer refinement. *Reflection/Revise:* Generate an initial answer, critique it against the retrieved evidence, and revise to fix omissions or inconsistencies (Shinn et al., 2023; Madaan et al., 2023).

4.2 Genetic Algorithm to Determine Effective Combinations

The modular RAG design space is highly combinatorial, yielding ∼1,296 possible pipelines in our setting—too many for exhaustive evaluation. To efficiently navigate this space, we employ a constrained genetic algorithm (GA) inspired by Kartal et al. (2025). GAs are well-suited for combinatorial optimization (Holland, 1975; Goldberg, 1989), allowing us to identify near-optimal solutions by evaluating only ∼200 pipelines (approx. 15% of the search space). We supplement this automated search with manual evaluation of established baselines. The GA evolves a population of pipelines by (i) evaluating them on a small query set, (ii) selecting top performers, and (iii) generating new candidates via crossover and mutation, enforcing compatibility constraints.

Genome encoding. We encode a RAG pipeline as a discrete genome:

$$g = (m_1, m_2, \ldots, m_F)$$

where F is the number of technique families. Each gene $m_f \in \mathcal{M}_f$ selects exactly one method from family f (optionally including a `None` choice to disable that slot).

Fitness function. We optimize a composite objective that balances retrieval effectiveness and answer quality on an evaluation subset $\mathcal{Q}$. Let $R(g)$ denote a retrieval metric (e.g., nDCG@k or Recall@k) and $G(g)$ a generation metric (e.g., judged faithfulness/accuracy). We compute both metrics on $\mathcal{Q}$ and normalize them to comparable scales (denoted by $\widetilde{\cdot}$). The fitness of genome g is:

$$\mathrm{Fit}(g) = \alpha \cdot \widetilde{R}(g) + (1 - \alpha) \cdot \widetilde{G}(g). \quad (1)$$

We set $\alpha = 0.5$ to weight retrieval and generation equally, reflecting that strong RAG performance requires both (i) retrieving relevant evidence and (ii) producing a faithful and accurate response conditioned on that evidence. This equal weighting also avoids over-specializing the search toward pipelines that optimize retrieval quality at the expense of answer quality (or vice versa).

GA Procedure and evaluation budget. We run the GA for a small number of generations to identify strong pipelines under a fixed evaluation budget. Concretely, we use population size $P = 20$ and $G = 10$ generations. Each candidate genome is evaluated on a randomly sampled set of $|\mathcal{Q}| = 100$ questions per domain. This yields a total of $P \times G$ candidate evaluations while keeping per-candidate evaluation lightweight. This GA evaluation serves as the primary empirical basis for the paper's best-practice conclusions (Section 5); the full algorithm is detailed in Appendix B (Algorithm 1).

Reproducibility. All parameter values used in the genetic search (including selection and elitism settings, crossover/mutation operators and rates, constraint checks, and random seeds) as well as the full Python implementation of the algorithm used in our experiments are shared in the code repository.

4.3 Experimental Setup and Metrics

We select metrics to (i) capture complementary failure modes, (ii) align with common practice to ease comparison across RAG systems, and (iii) reduce sensitivity to any single noisy automatic signal.

Concretely, for retrieval we report a mix of *coverage*-oriented metrics (Recall@5) and *ranking-quality* metrics (mAP, nDCG@5, MRR), since downstream generation depends both on whether evidence is retrieved at all and on how highly it is ranked. For generation, we combine an embedding-based semantic similarity signal with an LLM-as-a-judge score to balance paraphrase-tolerant matching with a more holistic assessment of correctness and answer quality; this choice follows the recommendation to use multiple complementary evaluation criteria, including LLM-judge style assessments, when analyzing benchmark and system outputs (Umutlu et al., 2025).

4.3.1 Retrieval Metrics

We evaluate retrieval against the ground-truth evidence passages used to generate each Q&A. For a query q with relevant passages $\mathcal{D}^\star$ and the ranked list $\pi_k(q)$ of top-k retrieved passages: We compute an overall *retrieval score* as an equally weighted aggregate of Recall@5, mAP, nDCG@5, and MRR (we report the component metrics alongside the aggregate).

$$\begin{aligned}\text{Retrieval}(q) = \alpha\Big(&\text{Recall@5}(q) + \text{mAP}(q) \\ &+ \text{nDCG@5}(q) + \text{MRR}(q)\Big)\end{aligned} \tag{2}$$

where we set $\alpha = 0.25$ to equally weight the four complementary retrieval metrics (coverage and ranking quality) without privileging any single component, consistent with our equal-weight aggregation choices elsewhere.

4.3.2 Generation Metrics

We measure end-to-end answer quality with two complementary signals and compute an overall *generation score* as their equally weighted aggregate (we also report each component): (i) *Semantic similarity* (embedding-based similarity between the model answer and the reference answer), and (ii) *LLM-Judge* (an LLM-based judgment score for answer quality/correctness). We aggregate them as

$$\text{Generation}(q) = \alpha\,\text{Sim}(q) + (1-\alpha)\,\text{Judge}(q). \tag{3}$$

In our experiments we set $\alpha = 0.5$ to give equal weight to semantic similarity and the judge signal, reflecting a conservative choice that avoids over-optimizing to either an embedding proxy or a single LLM-based evaluator (analogous to the equal-weight aggregation used in our GA fitness objective).

Table 3: Compute and implementation details.

Component	Setting
Embedding model	embeddinggemma
Generator model	gpt-oss:120B
Evaluator model	gemini-2.5-flash
Reranker	ms-marco-MiniLM-L-12-v2
Hardware	M3 Ultra 80-core GPU

4.3.3 Latency and Practicality

We report total token usage for each configuration but do not use this metric when optimizing with the GA; we report it to take into account the practicality of the configuration when suggesting best practices. Table 3 records model versions and system settings.

5 Best Practices

5.1 Evaluation Protocol

Our evaluation is performed within the GA procedure (Section 4.2). We sample a stratified random subset of $n = 100$ questions (balanced across the Web and Wikipedia parts) from the unified benchmark and use this set both to score candidate genomes during the search and to report the final performance of the best GA-selected configurations. The full benchmark contains $N \approx 20{,}459$ grounded QA pairs; subsampling keeps evaluation tractable while enabling controlled comparison across configurations. This evaluation procedure constitutes the empirical basis for our results.

Why $n = 100$ is sufficient. In pilot runs, we varied the subset size and observed that performance estimates saturated around $n = 100$. Beyond this point, variance reduction was marginal compared to the linear increase in evaluation cost. Thus, $n = 100$ serves as an efficient saturation point that yields stable mean estimates and consistent relative rankings of candidate configurations, while keeping the per-candidate evaluation cost low enough for the GA to explore many pipelines under a fixed budget.

5.2 Overall Performance

Table 4 summarizes the performance of the top configurations identified by the genetic search on the unified Turkish RAG benchmark ($n = 100$). We compare the baseline against three distinct optimal points found by the GA: a maximum-accuracy configuration, a Pareto-optimal “best value” configuration, and a production-friendly configuration.

The complete performance results for all noteworthy configurations are provided in Appendix B (Table 6).

5.3 Component Analysis and Inferences

Our results highlight several key trade-offs in the design space of Turkish RAG systems.

Maximizing Accuracy. If the goal is to maximize the end-to-end score, the winning recipe is a combination of strong query expansion (HyDE), strong reranking (Cross-encoder Reranking), aggressive context compression (Tree-style Summarize), and Long-context Ordering. This configuration achieves the top score of 85.00%, driven by high generation quality (0.823). However, this comes at a significant cost: approximately $3.7\times$ the token usage of the baseline. HyDE and summarization are computationally expensive, making this approach suitable only when accuracy is paramount and resources are unconstrained.

The Pareto Winner. The “Best Value” configuration (Cross-encoder Reranking + Previous/Next Chunk Augmentation + Long-context Ordering) achieves 84.60% overall score, which is only 0.4 points behind the top performer, but with significantly lower overhead. Compared to the baseline, it yields a +5.9 point improvement for $\sim 2\times$ tokens. This represents the cleanest recommendation for practical applications: prioritize Cross-encoder Reranking and local context enrichment (Previous/Next Chunk Augmentation) before adopting expensive methods like HyDE.

Lightweight Best Option (Production-Friendly). The “Production-Friendly” configuration (Query Clarification + Cross-encoder Reranking + Previous/Next Chunk Augmentation) achieves 80.20% overall score, with Retrieval 0.901 and Generation 0.704, consuming $\sim$1,738 tokens ($\approx 1.74\times$ baseline tokens). A short clarification / rewrite step plus cross-encoder reranking and adjacent-context augmentation delivers a strong accuracy gain at modest cost. This likely helps Turkish by reducing morphology-driven ambiguity and improving matching for entity/surface-form variation, while avoiding extra overhead from context reordering.

HyDE and LLM Reranking Costs. We find that HyDE is not a “free lunch” on Turkish datasets. While it helps bridge semantic gaps, it often balloons costs and can underperform if it introduces noise. For instance, HyDE combined with Tree-style Summarize led to high computational overhead. Similarly, LLM-based Reranking was outperformed by Cross-encoder Reranking in terms of score-per-cost. Cross-encoder Reranking proved to be a strong, reliable default, whereas LLM-based Reranking should be reserved for niche reasoning-heavy cases.

Risks of Over-Stacking. Stacking multiple LLM-based modules often degrades efficiency without guaranteeing better performance. A maximalist pipeline combining six complex modules achieved only 79.60% accuracy, underperforming the simpler “High Accuracy” configuration (85.00%) while consuming heavy token usage. Excessive LLM post-processing in Turkish may distort morphological cues or accumulate errors. From these results, we can recommend adding at most one LLM-heavy stage (e.g., summarization or reflection) only when necessary, avoiding “stacking everything” without proven gain.

Over-Filtering Problems. Strict filtering combined with segment extraction can severely harm retrieval in Turkish. A pipeline using strict thresholds and segment extraction dropped to 78.30% overall score (Retrieval 0.755), well below baseline. Morphology and paraphrase variance in noisy web data make hard thresholds risky, causing the system to miss evidence. We recommend preferring reranking over strict filtering unless thresholds are carefully calibrated per domain.

5.4 Recommendations

Based on these findings, we propose the following best practices for Turkish RAG:

- *Recommended Default:* Use *Cross-encoder Reranking + Previous/Next Chunk Augmentation + Long-context Ordering*. This pipeline offers a strong balance of accuracy and efficiency and should serve as the standard baseline for Turkish RAG experiments.
- *High-Accuracy:* For leaderboards or applications where score is critical, use *HyDE + Cross-encoder Reranking + Tree-style Summarize + Long-context Ordering*. This requires a higher budget for latency and tokens.
- *Production-Friendly:* For latency-constrained applications, use *Query Clarification + Cross-encoder Reranking + Previous/Next Chunk*

Table 4: Performance of GA-selected configurations on the unified benchmark ($n = 100$). We report Overall Score, Retrieval Score, Generation Score, and estimated Token usage per query. The "High Accuracy" model achieves the best scores but at high cost, while the "Pareto Optimal" model offers the best balance.

Configuration	**Overall Score**	**Retrieval Score**	**Generation Score**	**Tokens (est.)**
High Accuracy (HyDE + Cross-encoder Reranking + Tree-style Summarize + Long-context Ordering)	85.00%	0.876	0.823	$\sim$3664
Pareto Optimal (Cross-encoder Reranking + Previous/Next Chunk Augmentation + Long-context Ordering)	84.60%	0.870	0.823	$\sim$1987
Production-Friendly (Query Clarification + Cross-encoder Reranking + Previous/Next Chunk Augmentation)	80.20%	0.901	0.704	$\sim$1738
Baseline (Dense Retrieval + Similarity Thresholding + Naive Concatenation)	78.70%	0.872	0.702	$\sim$1000

Augmentation. This provides meaningful improvements over naive RAG while maintaining fast response times and low token usage.

These recommendations are specific to Turkish and were derived under our experimental setup. While other morphologically rich languages such as Finnish, Hungarian, and Korean face similar challenges (e.g., morphological richness, agglutination, surface-form variation) (Tsarfaty et al., 2014; Gerz et al., 2018), we do not claim that our findings transfer directly; validating generalizability to other languages requires dedicated experiments.

6 Conclusion and Future Work

We presented an end-to-end, domain-aware study of Turkish RAG across informal web text and Turkish Wikipedia. By benchmarking modular choices across the RAG pipeline, we distill practical, domain-specific configuration guidance and provide resources to support reproducible Turkish-RAG research.

Future Work. We plan to: (i) explore hybrid RAG within a family instead of using single method from each, (ii) incorporate graph structure for retrieval and query expansion (entity graphs, hyperlink graphs), (iii) scale to larger and more diverse Turkish corpora (news, technical documentation, legal text), (iv) study Turkish morphology-aware retrieval features (e.g., lemma-aware sparse retrieval, morphological analyzers), (v) incorporate more noisy, real-world-like data and unanswerable questions to better assess system robustness (moving beyond our current evaluation on relatively clean questions), and (vi) examine document-side, index-time methods—including pre-embeddings and related pre-computation/caching techniques. We also plan controlled technique-family ablations per domain to quantify marginal gains and interactions (retrieval vs. generation) under a fixed evaluation and budget. Additionally, we plan to conduct a structured error analysis of end-to-end outputs, focusing on Turkish-specific failure modes such as inflection-driven mismatch, over-normalization of informal language, entity drift, and missing evidence in multi-passage questions. A systematic taxonomy and annotated error set will help separate retrieval versus generation errors and guide targeted improvements.

7 Limitations and Ethical Considerations

Limitations. All recommendations in this paper reflect our specific experimental setup (models, prompts, tokenization, corpus preprocessing, hardware, and context limits). In practice, the best-performing settings can shift across Turkish corpora due to differences in domain, document length distribution, content quality, noise/boilerplate, and latency constraints. We therefore position our findings as *best practices for generic Turkish text retrieval*, and encourage practitioners to re-run a small sweep on their own data to identify the best point on the quality–latency tradeoff. While the dataset reflects realistic web content, it is cleaner than typical production RAG pipelines and does not cover specialized domains such as legal or technical documentation; extending the benchmark to such domains is left for future work. As a defense-industry organization, we cannot release or fully describe some proprietary data sources used during development (e.g., internal enterprise documents); therefore, this paper and the released resources focus on openly available data.

Reproducibility and releases. We release the evaluation datasets, the full QA set with evidence spans, all RAG configuration files, and scripts to reproduce metrics in the code repository. All recommendations in this paper reflect our specific experimental setup (models, prompts, tokenization, corpus preprocessing, hardware, and context limits). In practice, the best-performing settings can shift across Turkish corpora due to differences in domain (formal vs. informal), document length distribution, noise/boilerplate, and latency constraints. We therefore position our findings as *best practices for generic Turkish text retrieval*, and encourage practitioners to re-run on their own data to identify the best point on the quality–latency tradeoff.

Ethics. Informal web data can contain sensitive or personal content; we recommend careful filtering, redaction, and license-aware release. Because the data are collected from the public internet, they may reflect societal biases and other problematic content; any such content is included for research purposes only and does not reflect the authors' opinions or endorsements. LLM-based filtering can itself introduce bias; we therefore document filtering criteria and provide audit samples where feasible.

Data Provenance and Copyright. The Web Part of our dataset is derived from CulturaX (Nguyen et al., 2024), a publicly available multilingual corpus; we do not perform independent scraping of websites. Source domains (e.g., haberler.com, sikayetvar.com) were included in CulturaX due to their topical diversity and public accessibility. We release only derived annotations—question–answer pairs, topic labels, and chunk boundaries—rather than redistributing full original articles. This approach aligns with standard research practices for web-derived corpora and respects the original data providers.

Use of Generative AI. Generative AI was used solely to assist with language editing. All scientific contributions, data construction, analysis, and interpretations presented in this work are original and were conducted entirely by the authors.

Acknowledgments

We gratefully acknowledge support from Roketsan Inc. and the Google Gemini Academic Reward Program, which helped enable the experiments and computing resources used in this study.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

Lorin W. Anderson and David R. Krathwohl, editors. 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York, NY.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*. Oral.

Erdoğan Bikmaz, Mohammed Briman, and Serdar Arslan. 2025. Bridging the language gap in RAG: A case study on Turkish retrieval and generation. *Researcher*, 5(1):38–49.

Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. David McKay Company, Inc., New York, NY.

Çağrı Çöltekin, A. Seza Doğruöz, and Özlem Çetinoğlu. 2023. Resources for Turkish natural language processing: A critical survey. *Language Resources and Evaluation*, 57:449–488.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. arXiv preprint. *Preprint*, arXiv:2404.16130.

Yanai Elazar, Akshita Bhagia, Ian Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hanna Hajishirzi, Noah A. Smith, and Jesse Dodge. 2023. What's in my big data? arXiv:2310.20707.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Özay Ezerceli, Gizem Gümüşçekiçci, Tuğba Erkoç, and Berke Özenç. 2025. TurkEmbed4Retrieval: Turkish embedding model for retrieval task. arXiv preprint. *Preprint*, arXiv:2511.07595.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. arXiv:2101.00027.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 316–327, Brussels, Belgium. Association for Computational Linguistics.

David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

Google. 2025. Gemini api: Model information. Accessed: 2025-12-24.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. A comprehensive survey of retrieval-augmented generation (RAG): Evolution, current landscape and future directions. arXiv preprint. *Preprint*, arXiv:2410.12837.

Dilek Z Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410.

John H. Holland. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. LLMLingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*.

Muhammed Yusuf Kartal, Suha Kagan Köse, Korhan Sevinç, and Burak Aktas. 2025. RAGSmith: A framework for finding the optimal composition of retrieval-augmented generation methods across datasets. *arXiv preprint arXiv:2511.01386*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*, pages 39–48. ACM.

Dongkyu Kim, Byoungwook Kim, Donggeon Han, and Matouš Eibich. 2024. AutoRAG: Automated framework for optimization of retrieval augmented generation pipeline. arXiv preprint. *Preprint*, arXiv:2410.20878.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2005.11401*.

Minghan Li, Xinxuan Lv, Junjie Zou, Tongna Chen, Chao Zhang, Suchao An, Ercong Nie, and Guodong Zhou. 2025. Query expansion in the age of pre-trained and large language models: A comprehensive survey. arXiv preprint. *Preprint*, arXiv:2509.07794.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of EMNLP*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*, pages 2356–2362, Virtual Event, Canada. ACM.

Charles Z. Liu, Imani Abayakoon, and Farookh Khadeer Hussain. 2025a. Retrieval-augmented generation: A survey of methodologies, techniques, applications, and future directions. Preprint.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Zuhong Liu, Charles-Elie Simon, and Fabien Caspani. 2025b. Passage segmentation of documents for extractive question answering. *arXiv preprint arXiv:2501.09940*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of ACL*.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2024. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4226–4237, Torino, Italia. ELRA and ICCL.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

Kemal Oflazer. 2014. Turkish and its challenges for language processing. *Language resources and evaluation*, 48(4):639–653.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925. `https://openai.com/research/gpt-oss-model-card/`.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb datasets: Decanting the web for the finest text data at scale. arXiv:2406.17557.

Zackary Rackauckas. 2024. RAG-fusion: a new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*. NeurIPS 2023.

Murat Simsek. 2025. Retrieval-augmented generation versus fine-tuning for Turkish cultural question answering: A comprehensive evaluation and analysis. Research Square preprint.

Luca Soldaini and 1 others. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wenyu Tao, Xiaofen Xing, Yirong Chen, Linyi Huang, and Xiangmin Xu. 2025. TreeRAG: Unleashing the power of hierarchical storage for enhanced knowledge retrieval in long documents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 356–371, Vienna, Austria. Association for Computational Linguistics.

Selva Taş, Mahmut El Huseyni, Özay Ezerceli, Reyhan Bayraktar, and Fatma Betül Terzioğlu. 2025. Turklettucedetect: A hallucination detection models for Turkish RAG applications. arXiv preprint. *Preprint*, arXiv:2509.17671.

Çağrı Toraman, Ahmet Kaan Sever, Ayse Aysu Cengiz, Elif Ecem Arslan, Görkem Sevinç, Mete Mert Birdal, Yusuf Faruk Güldemir, Ali Buğra Kanburoğlu, Sezen Felekoğlu, Osman Gürlek, Sarp Kantar, Birsen Şahin Kütük, Büşra Tufan, Elif Genç, Serkan Coşkun, Gupse Ekin Demir, Muhammed Emin Arayıcı, Olgun Dursun, Onur Gungor, and 3 others. 2026. Turkbench: A benchmark for evaluating turkish large language models. *arXiv preprint arXiv:2601.07020*.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2014. SPMRL-SANCL 2014 shared task on parsing morphologically rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland. Dublin City University.

Elif Ecem Umutlu, Ayse Aysu Cengiz, Ahmet Kaan Sever, Seyma Erdem, Burak Aytan, Busra Tufan, Abdullah Topraksoy, Esra Darıcı, and Cagri Toraman. 2025. Evaluating the quality of benchmark datasets for low-resource languages: A case study on Turkish. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pages 471–487, Vienna, Austria and virtual meeting. Association for Computational Linguistics.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of EMNLP*.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. RedPajama: an open dataset for training large language models. arXiv:2411.12372.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. arXiv preprint. *Preprint*, arXiv:2405.07437.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for AI-generated content: A survey. arXiv preprint. *Preprint*, arXiv:2402.19473.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2023a. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*. ICLR 2024.

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. 2023b. Judging LLM-as-a-judge with MT-bench and chatbot arena. arXiv preprint. *Preprint*, arXiv:2306.05685.

A Prompts and Validation Rubrics

A.1 URL-only Filtering Prompt

URL Filtering

```
Analyze the following website (base URL) and determine its eligibility:

Website: {url}

Evaluate:
1. Does this website likely contain valuable information (educational,
↪  informative, useful content)?
2. Is the content on this website likely written in proper language (casual,
↪  conversational)?

Based on your analysis of the website domain and typical content, provide:
- Status: "ELIGIBLE" if BOTH conditions are true, otherwise "NOT ELIGIBLE"
- Reason: Brief explanation (1-2 sentences)

Response format:
Status: [ELIGIBLE/NOT ELIGIBLE]
Reason: [Your explanation]
```

A.2 Content Filtering Prompt

Content Filtering

```
You are a data quality and style evaluator. You will be given TURKISH text taken
↪  from a web page, along with the URL it came from.

TASK 1 -- EVALUATION
Evaluate whether the text is:
- suitable for a RAG system,
- understandable,
- and "CLEAN" (everyday language; not nonsense/trash).

Definitions:

1) Informality level:
- "clean": Everyday language (blog/forum/social media), but:
  * understandable
  * slightly relaxed yet still structured
  * similar to news-site tone
  * sentences mostly well-formed
  * no heavy slang, no spam
- "nonsense_or_spam": incoherent, random words, bot/spam, only links/hashtags,
↪  etc.

2) Quality:
- "good": clear, coherent, not full of spelling errors, topic is followable,
↪  usable for RAG
- "bad": too short, messy, major spelling/spam issues, topic not followable

3) ELIGIBLE criteria:
- mostly Turkish
- "Clean"
- Quality must be "good"
- definitely NOT "formal"
- definitely NOT "nonsense_or_spam"
- text length > 100 characters
- mostly about a single topic/theme

IMPORTANT: First do the evaluation and determine Status.

TASK 2 -- MARKDOWN CONVERSION (ONLY IF ELIGIBLE)
WARNING: Do this step ONLY if Status: ELIGIBLE. If NOT ELIGIBLE, do NOT convert
↪  to markdown.
```

```
If Status: ELIGIBLE:
1) Detect headings and use markdown headings (#)
2) Split paragraphs
3) Remove unnecessary whitespace
4) Do not change content beyond that; do not add new content

OUTPUT FORMAT:

Status: [ELIGIBLE/NOT ELIGIBLE]
Reason: [Short explanation]

---MARKDOWN_START---
[ONLY if Status: ELIGIBLE, put the markdown-converted text here]
[If Status: NOT ELIGIBLE, leave this section COMPLETELY EMPTY]
---MARKDOWN_END---

URL: {url}
Text: {text}
```

A.3 Topic Classification Prompt

Topic Classification

```
You are labeling Turkish text for an LLM dataset. You must not use or infer any
↪  "source type" (Wikipedia vs web) in your decision. Treat every document the
↪  same.
You will be given one document: title (optional), url (optional), and text (may
↪  be truncated).
Task:

1) Assign exactly ONE topic category: topic_l1
2) Assign exactly ONE safety category: safety_label

Allowed values:
topic_l1 (choose exactly one):
- STEM
- Humanities
- Social_Sciences
- Professional_Applied
- Culture_Entertainment
- Everyday_Life
- Geography_Places
- Biography_People
- Organizations_Institutions
- Events_History
- Meta_Content

safety_label (choose exactly one):
- Safe
- Needs_Filtering
- Exclude
Safety guidelines:
- Safe: ordinary content with no clear policy risks.
- Needs_Filtering: contains potentially sensitive/age-restricted/controversial
↪  material or advisory content (e.g., medical or financial advice, explicit
↪  profanity/hate slurs contextually used, graphic descriptions) but not
↪  clearly disallowed.
- Exclude: clearly disallowed or high-risk content, such as explicit
↪  instructions for wrongdoing (e.g., making weapons, fraud), explicit sexual
↪  content involving minors, actionable self-harm instructions, doxxing/PII,
↪  extremist recruitment/praise, or pervasive hate/harassment.

Output rules:
- Output JSON only. No markdown. No extra keys.
- Keep rationale <= 200 characters, grounded only in the given text.

JSON format:
```

```
{
  "topic_l1": "...",
  "safety_label": "...",
  "rationale": "..."
}

Now label this document:
TITLE: {{title}}
URL: {{url}}

TEXT: {{text}}
```

A.4 QA Generation Prompts

We employ two types of prompts for question generation depending on the context: single-chunk and multi-chunk generation.

A.4.1 Single Chunk Generation

Single Chunk Generation

```
Generate exactly {num_questions} question-answer pair(s) that can be answered
↪  from this text chunk:

Chunk ID: {chunk_id}{context_info}

Text:
{chunk_content}

Each question must be categorized into one of these two categories:
1. **FACTUAL**: Questions that test direct recall of specific details. The
↪  answer is a specific name, date, number, or short verbatim phrase found
↪  directly in the text.
2. **INTERPRETATION**: Questions that test comprehension by asking for
↪  explanations of causes, effects, or relationships between concepts in the
↪  text. The answer requires synthesizing information rather than just quoting
↪  it.

Requirements:
- Only ask about information explicitly stated in this text
- Make questions specific and factual
- Each question should be answerable from this chunk alone
- Provide complete, accurate answers based solely on the chunk content
- Categorize each question appropriately based on the type of cognitive task
↪  required
- Return valid JSON with the specified structure
- Do NOT use markdown code blocks (like)
- Return ONLY the JSON object, no other text
```

A.4.2 Multi-Chunk Generation

Multi-Chunk Generation

```
Generate exactly {num_questions} question-answer pair(s) that require
↪  information from multiple chunks below.

These chunks are related. Generate questions that:
1. Require information from at least 2 of the provided chunks
2. Are about connections, relationships, comparisons, or broader concepts across
↪  chunks
3. Cannot be answered from any single chunk alone{context_info}

Each question must be categorized into one of these two categories:
```

```
1. **FACTUAL**: Questions that test direct recall of specific details. The
↪  answer is a specific name, date, number, or short verbatim phrase found
↪  directly in the text.
2. **INTERPRETATION**: Questions that test comprehension by asking for
↪  explanations of causes, effects, or relationships between concepts in the
↪  text. The answer requires synthesizing information rather than just quoting
↪  it.

Chunks:
{chunks_text}

Requirements:
- Focus on relationships and connections between the chunks
- Make questions that require synthesis of information
- Provide complete answers that synthesize information from multiple chunks
- Categorize each question appropriately based on the type of cognitive task
↪  required
- Return valid JSON with chunk IDs {chunk_ids} in related_chunk_ids
- Do NOT use markdown code blocks (like)
- Return ONLY the JSON object, no other text
```

A.5 QA Validation Rubric

QA Validation System Prompt

```
You evaluate question-answer pairs for accuracy.

Check if:
- Question is clear
- Answer is accurate based on provided text chunks
- Answer fully addresses the question
- Chunks contain all necessary information

Return JSON: {"is_correct": boolean, "reason": "brief explanation"}

Keep reason concise (max 50 words). Return ONLY valid JSON.
```

B Full Experimental Results and Algorithms

Algorithm 1 Genetic search over modular RAG pipelines

```
Require: Families $\{\mathcal{M}_f\}_{f=1}^F$, population size $P$, generations $G$, mutation rate $\mu$, evaluation set $\mathcal{Q}$
1: Initialize population $\mathcal{P}_0 = \{g_i\}_{i=1}^P$ by sampling valid genomes
2: for $t = 1$ to $G$ do
3:     Evaluate $\mathrm{Fit}(g)$ for all $g \in \mathcal{P}_{t-1}$ on $\mathcal{Q}$
4:     Select elites $\mathcal{E}$ and parents $\mathcal{S}$ (e.g., tournament selection)
5:     Create offspring via crossover over genomes in $\mathcal{S}$
6:     Mutate genes with probability $\mu$
7:     Form $\mathcal{P}_t \leftarrow \mathcal{E} \cup \mathcal{O}$
8: end for
9: return best genomes from $\mathcal{P}_G$
```

Table 5: Top base domains by document frequency with cumulative coverage (Web Part).

Domain	**Docs**	**Cumulative %**
sikayetvar.com	227	3.6
haberler.com	158	6.1
posta.com.tr	134	8.2
mynet.com	132	10.3
donanimhaber.com	108	12.0
webtekno.com	100	13.6
onedio.com	98	15.2
sondakika.com	98	16.7
fanatik.com.tr	91	18.2
haberaktuel.com	89	19.6

Table 6: Complete performance results for **noteworthy** evaluated RAG configurations.

RAG Methods Combination	**Overall Score**	**Retrieval**	**Generation**	**Total Token Usage**
hyde + ce_rerank + tree_summarize + long_context_reorder	85.00%	0.876	0.823	3,663.8
hyde + ce_rerank + llm_summarize + reflection_revising	84.90%	0.876	0.822	3,118.4
hyde + ce_rerank + tree_summarize + reflection_revising	84.80%	0.876	0.819	3,966.2
ce_rerank + adjacent_augmenter + long_context_reorder	84.60%	0.87	0.823	1,987.2
hyde + tree_summarize	84.50%	0.892	0.798	5,260.4
hyde + ce_rerank + tree_summarize + long_context_reorder + reflection_revising	84.50%	0.876	0.814	3,964.1
hyde + ce_rerank + adjacent_augmenter + tree_summarize + long_context_reorder	84.40%	0.876	0.812	4,906.3
ce_rerank + adjacent_augmenter + long_context_reorder	84.40%	0.865	0.822	2,036.8
hyde + tree_summarize + long_context_reorder	84.30%	0.892	0.794	5,276.3
ce_rerank + adjacent_augmenter + llm_summarize + long_context_reorder	83.40%	0.87	0.799	2,703.2
hyde + long_context_reorder + reflection_revising	83.10%	0.896	0.765	2,339.3
hyde + llm_rerank + tree_summarize	83.10%	0.868	0.795	4,295.2
hyde + adjacent_augmenter + long_context_reorder	82.90%	0.896	0.761	3,138.8
adjacent_augmenter + long_context_reorder	82.70%	0.896	0.758	2,147.0
llm_rerank + adjacent_augmenter + llm_summarize	82.70%	0.863	0.792	2,973.8
ce_rerank + llm_summarize + long_context_reorder	82.40%	0.87	0.778	2,167.4
ce_rerank + long_context_reorder + reflection_revising	82.40%	0.865	0.783	1,773.4
hyde + relevant_segment_extractor + llm_summarize + long_context_reorder	82.00%	0.891	0.75	2,685.5
ce_rerank + adjacent_augmenter + tree_summarize + long_context_reorder	81.90%	0.865	0.772	4,762.7
hyde + llm_summarize + long_context_reorder	81.50%	0.896	0.733	3,437.9
simple_query_refinement_clarification + ce_rerank + adjacent_augmenter + long_context_reorder	81.10%	0.904	0.719	1,928.6
adjacent_augmenter + llm_summarize + long_context_reorder	81.10%	0.896	0.726	3,073.2
hyde + llm_summarize	81.10%	0.896	0.726	3,409.4
ce_rerank + relevant_segment_extractor + llm_summarize + long_context_reorder + reflection_revising	81.10%	0.87	0.753	2,828.2
hyde + llm_summarize	80.90%	0.896	0.723	3,261.5
hyde + adjacent_augmenter + llm_summarize + long_context_reorder	80.70%	0.896	0.717	3,994.5
hyde + llm_summarize + long_context_reorder	80.60%	0.896	0.715	3,472.5
hyde + llm_summarize + long_context_reorder + reflection_revising	80.50%	0.896	0.714	3,806.4
ce_rerank + relevant_segment_extractor + tree_summarize + long_context_reorder	80.50%	0.87	0.74	3,886.9
simple_query_refinement_clarification + ce_rerank + adjacent_augmenter	80.20%	0.901	0.704	1,738.0
query_expansion_simple_multi_query_borda + ce_rerank + adjacent_augmenter	80.00%	0.887	0.712	1,431.4
llm_rerank + llm_summarize	80.00%	0.885	0.715	2,707.5
simple_query_refinement_clarification + llm_rerank + adjacent_augmenter	79.80%	0.903	0.693	2,291.5
simple_query_refinement_clarification + ce_rerank + adjacent_augmenter + reflection_revising	79.80%	0.9	0.697	1,991.6
simple_query_refinement_clarification + ce_rerank + adjacent_augmenter + tree_summarize + long_context_reorder + reflection_revising	79.60%	0.836	0.756	4,524.5
query_expansion_simple_multi_query_borda + ce_rerank + similarity_threshold + adjacent_augmenter + llm_summarize + reflection_revising	79.40%	0.882	0.706	2,155.3
hyde + ce_rerank + llm_summarize + long_context_reorder	79.40%	0.877	0.711	3,197.8
hyde + tree_summarize + long_context_reorder	79.30%	0.896	0.689	7,137.7
hyde + ce_rerank + adjacent_augmenter + llm_summarize	79.30%	0.877	0.708	3,343.6
vector_simple + simple_threshold + simple_listing (Baseline)	78.70%	0.872	0.702	1,000.4
adjacent_augmenter + tree_summarize + long_context_reorder	78.40%	0.896	0.672	7,216.6
ce_rerank + similarity_threshold + relevant_segment_extractor + tree_summarize + reflection_revising	78.30%	0.755	0.81	2,717.9