

Comparing Chunking and Embedding Strategies for Turkish RAG Systems

Mustafa Sertac Türkel
Data Science and Innovation
Ata Technology Platforms
Istanbul, Turkey
sertac.turkel@atptech.com

Fatma Nur Korkmaz
Data Science and Innovation
Ata Technology Platforms
Istanbul, Turkey
fatmanur.korkmaz@atptech.com

Ahmet Tuğrul Bayrak
Data Science and Innovation
Ata Technology Platforms
Istanbul, Turkey
tugrul.bayrak@atptech.com

Accepted to INTCEC 2026. This is the author's pre-print version.
The final authenticated version will be available through the
conference proceedings.

Abstract—How documents are segmented into retrievable chunks and how those chunks are embedded strongly affect Retrieval-Augmented Generation (RAG) quality, yet neither has been systematically studied for morphologically rich languages such as Turkish. We compare Turkish document question answering across three chunking strategies (fixed-length, semantic, and layout-aware Docling), five embedding models, and two generator LLMs, over three documents with contrasting layouts. The fully crossed design yields 9,000 graded question-answer evaluations, each scored by an independent judge model, and component comparisons are tested by paired McNemar tests under Holm correction. Four findings follow. The chunking strategy determines how much the embedding choice matters: layout-aware chunking compresses the spread between the modern embedding models to about a point. The three leading embedding models are statistically indistinguishable, so language specialization yields no measurable retrieval advantage. The faster generator is not the more accurate one. And the preferred configuration depends on content type, since layout-aware chunking helps documents containing tables far more than prose. The best individual components therefore do not compose into the best complete configuration, which reaches 87.0%.

Index Terms—retrieval-augmented generation, large language models, text chunking, embedding models, Turkish NLP, question answering

I. INTRODUCTION

Retrieval-Augmented Generation (RAG) combines a retrieval component that selects relevant passages from a document collection with a generative large language model (LLM) that answers conditioned on those passages [1], [2]. Because the LLM answers from retrieved context, quality depends on decisions made before generation: how documents are split into retrievable units (*chunking*), and how those units and the query are mapped into a vector space for similarity search (*embedding*) [3]. These choices interact with each other and with document properties, yet in practice are often made by convention rather than measurement [4].

This matters for Turkish. As an agglutinative language, Turkish encodes grammatical information through extensive

suffixation, so a single lemma appears in many inflected forms [5]. The effect manifests first at tokenization: subword vocabularies poorly matched to Turkish morphology fragment inflected forms inconsistently, degrading downstream Turkish model performance [6], and morphological sparsity is a recurring obstacle for morphologically rich languages [7]. An embedding model inheriting such a mismatch may place semantically similar passages far apart. Document layout adds a second factor: institutional documents mix prose with tables, and a chunking method that splits a table across boundaries makes the corresponding facts harder to retrieve regardless of the embedding.

This paper identifies, by direct measurement, an effective RAG configuration for Turkish document question answering. We treat this as a controlled comparison across chunking strategy, embedding model, and generator LLM, evaluating every combination on the same question set, and report operational metrics alongside accuracy because a deployed system must balance quality against cost and latency. Our contributions are fourfold. First, we describe a reproducible harness that grades 9,000 question-answer pairs across a fully crossed design, reporting all fourteen planned component comparisons with a paired significance test under Holm correction. Second, we show that the three leading embedding models cannot be statistically separated, so language specialization does not by itself confer a retrieval advantage. Third, we analyse the chunking-by-embedding interaction and show that the chunking strategy governs how much the embedding choice matters. Fourth, we show that the preferred configuration depends on content type.

II. RELATED WORK

RAG combines the parametric knowledge of a language model with non-parametric retrieval over an external corpus, allowing use of up-to-date or domain-specific content without retraining [1]. Generative readers such as Fusion-in-Decoder showed that conditioning generation on several retrieved passages jointly improves open-domain question answering [2]. Retrieval has shifted from sparse lexical matching to dense retrieval, in which queries and passages are embedded into

a shared space and compared by inner product or cosine similarity [3]; at scale this uses approximate nearest-neighbour indexes such as FAISS [8]. Embedding quality is therefore central to end-to-end performance.

Sentence-level encoders trained with Siamese or contrastive objectives, such as Sentence-BERT [9] and language-agnostic models such as LaBSE [10], suit semantic similarity better than token-level masked language models [11] or classical subword word embeddings [12]. Large multilingual encoders such as multilingual-e5 [13] and commercial embedding services [14] extend dense representations to many languages at once, but strong average multilingual performance does not guarantee strong results for any single language. For Turkish, dedicated pretrained models such as BERTurk [5] outperform multilingual models on downstream tasks such as classification and offensive language detection [15], and tokenization studies confirm that the match between subword vocabulary and Turkish morphology measurably affects model quality [6]. Whether this transfers from classification to retrieval, where the objective is relative ranking rather than label prediction, is an open question we examine directly.

Chunking has received less systematic attention, though it determines which units the retriever can see at all. The problem predates RAG as *text segmentation*: lexical-cohesion methods such as TextTiling place boundaries where vocabulary overlap between adjacent windows drops [16], and supervised neural segmenters learn boundary placement from labelled documents [17]. Fixed-length splitting remains the common baseline in deployed systems, semantic segmentation is the practical descendant of the cohesion-based line, and layout-aware parsing preserves structural elements such as tables and headings, the role played by toolkits such as Docling [18]. For evaluation, model-based protocols such as RAGAS assess RAG outputs at scale [19], and surveys review the design space [4]. We contribute a controlled, fully crossed comparison for Turkish documents mixing prose and tabular content, with operational metrics and significance tests.

III. DATA

The corpus consists of three documents spanning layout characteristics representative of institutional Turkish content, with a fixed question set per document whose reference answers are known in advance. The documents are distinguished by the proportion of content presented as tables rather than flowing text and are referred to by document type: *table-heavy*, *balanced* (tables and text in comparable proportion), and *text-heavy*. Since identical chunking and embedding methods are applied to all three, this isolates the effect of layout.

Each document has 100 distinct questions, 300 in total, and every question is labelled according to whether answering it requires information located in a table (*is_in_table*), which supports the separate analysis in Section V. Of the 300 questions, 105 are table-grounded and 195 text-grounded, distributed as 80, 22, and 3 of 100 in the table-heavy, balanced, and text-heavy documents respectively. This uneven distribution is intentional: it allows table-grounded accuracy to be

examined under three different densities of tabular content. It also means that the table-question figure for the text-heavy document rests on only three questions and is reported for completeness rather than interpretation.

Crossing 3 chunking strategies, 5 embedding models, and 2 generators gives 30 pipeline configurations, and answering all 300 questions under each yields $300 \times 30 = 9,000$ graded evaluations. Each configuration-level accuracy is a mean over 300 questions, each embedding-level figure a mean over 1,800 evaluations, each chunking-level figure a mean over 3,000, and each generator-level figure a mean over 4,500. Because the design is fully crossed, component comparisons are not confounded by uneven allocation of questions or documents, and because the same 300 questions underlie every configuration, the paired tests of Section IV-F apply.

IV. METHOD

A. RAG Pipeline

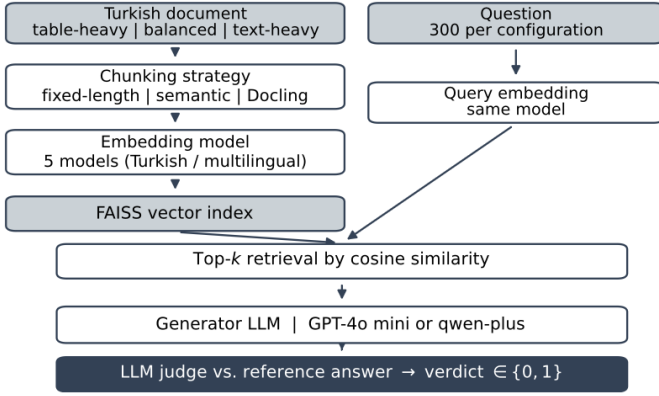
Every configuration follows the same pipeline, differing only in the components under comparison. Each document is segmented by the selected chunking strategy; every chunk is encoded into a dense vector by the selected embedding model and stored in a FAISS index [8]. At query time the question is embedded with the same model, the most similar chunks are retrieved by cosine similarity, and they are inserted into a fixed prompt template with the question. The selected generator produces the answer, which is graded against the reference as described in Section IV-E. Fig. 1 shows the architecture of a single configuration together with the corresponding procedure. The number of retrieved chunks, the prompt template, and the similarity metric were held constant, so the differences in Section V are attributable to the three factors under study rather than to incidental changes in retrieval or prompting; this allows the $3 \times 5 \times 2$ design to be read as a controlled experiment rather than a collection of separately tuned systems.

B. Chunking Strategies

Fixed-length chunking splits text into segments of fixed target character length regardless of content, serving as a baseline. *Semantic* chunking places boundaries where topical similarity between adjacent sentences drops; it is the practical descendant of lexical-cohesion segmentation [16], [17] and typically produces shorter chunks. *Docling* chunking is layout-aware, parsing document structure and keeping structurally coherent elements, in particular tables, together within a chunk [18], which yields larger chunks. The three strategies thus represent increasing awareness of structure, from none, through topical structure, to explicit layout.

C. Embedding Models

FastText [12] is a classical subword-based word-embedding method included as a lightweight baseline. *TurkEmbed* [20] and *Mursit-large* [21] are publicly available embedding models oriented toward Turkish, the former trained on Turkish natural language inference and sentence-similarity data and the latter retrieval-tuned on a Turkish-dominant pretraining



```

1: index ← FAISS(embedding model(chunker(document)))
2: for each question do
3:   answer ← generator(prompt(question, top-k(index, question)))
4:   verdict ← judge(question, answer, reference) ∈ {0,1}
5: end for
  
```

Fig. 1. Architecture and procedure of one RAG configuration. Only chunking strategy, embedding model, and generator vary; retrieval depth, prompt template, similarity metric, and judge are held constant. Repeating it for all $3 \times 5 \times 2 = 30$ configurations over 300 questions gives 9,000 verdicts.

corpus. *multilingual-e5-large* [13] is a large multilingual text-embedding model, and *text-embedding-3-small* [14] a general-purpose commercial model accessed through a hosted API. All five were evaluated in the identical pipeline, so only the embedding step varied. Dense sentence encoders of this kind [9], [10] have largely replaced word-level representations [11] for retrieval; the selection lets us test whether that holds for Turkish and whether language-specialized models add further benefit.

D. Generator LLMs

Two generators were compared: *GPT-4o mini* [22] and *qwen-plus* [23], compact commercial models from different providers. Both received the same template and retrieved context, so the comparison reflects generation quality and latency rather than retrieval.

E. Evaluation Protocol and Operational Metrics

Each answer was graded correct or incorrect against the reference for its question, and accuracy is the percentage correct over the relevant set. Grading was automatic, performed by an LLM judge, Llama 3.3 70B Instruct [24], prompted with the question, the reference answer, and the candidate answer and instructed to return a binary verdict; this reference-guided model-based protocol follows practice established for scalable RAG evaluation [19]. The judge is deliberately drawn from a third provider, distinct from both generators, so grading cannot favour either generator through self-preference. The judge’s verdicts were not validated against human annotation in this study, so every accuracy reported below is an accuracy with respect to this judge rather than to human adjudication; a stratified human validation is left to future work.

Three operational metrics were recorded per evaluation: *average chunk size*, the mean character length of retrieved

chunks and a proxy for context tokens and therefore cost per query; *average answer length*, the mean character length of generated answers; and *average response time*, mean end-to-end API latency. A configuration that is slightly more accurate but slower or more expensive may not be the preferred production choice.

F. Statistical Analysis

Component comparisons exploit the fact that all configurations answer the same 300 questions: for two levels of a factor the evaluations match one-to-one on the remaining factors, so differences are assessed with the paired McNemar test with continuity correction rather than a test for independent proportions, which would ignore the pairing. Fourteen comparisons are planned, so *p*-values are adjusted by Holm–Bonferroni and the adjusted values interpreted. To test whether the effect of the embedding model depends on the chunking strategy, a logistic regression of correctness on chunking, embedding, and generator main effects is compared by likelihood-ratio test against the same model extended with the chunking-by-embedding interaction.

V. RESULTS

Across all 9,000 evaluations, overall accuracy is 68.10%, a reference point only, since accuracy varies widely with configuration.

Table I reports accuracy and operational metrics by embedding model, each over 1,800 evaluations. *multilingual-e5-large* is strongest at 80.00%, ahead of the Turkish-oriented models and *text-embedding-3-small*, which fall between 75.78% and 78.67%; FastText is substantially lower at 27.56%, so a subword word-embedding method is not competitive here. The four competitive models differ only modestly on operational metrics, with chunk sizes between 1,836 and 1,909 characters and response times near 4.0 s, so their accuracy differences are not explained by context length.

TABLE I
PERFORMANCE BY EMBEDDING MODEL

Embedding	Acc. (%)	Chunk size	Ans. len.	Time (s)
multilingual-e5-large	80.00	1836.39	72.63	3.99
TurkEmbed	78.67	1865.79	73.34	4.04
text-embedding-3-small	78.50	1869.44	77.54	4.10
Mursit-large	75.78	1909.23	74.54	4.09
FastText	27.56	1960.40	86.28	4.33

By generator, *GPT-4o mini* is more accurate than *qwen-plus* (69.58% versus 66.62%), while *qwen-plus* is faster, averaging 2.60 s against 5.62 s; mean answer lengths are 79.55 and 74.18 characters. The accuracy gap is small in absolute terms but statistically reliable (Table II). This is a cost-quality trade-off: *qwen-plus* responds in under half the time at a cost of about three points of accuracy, acceptable in latency-sensitive deployments and poor where accuracy is the binding constraint.

Fig. 2 reports accuracy by chunking strategy, decomposed by question type. Docling is strongest overall at 74.37%, and

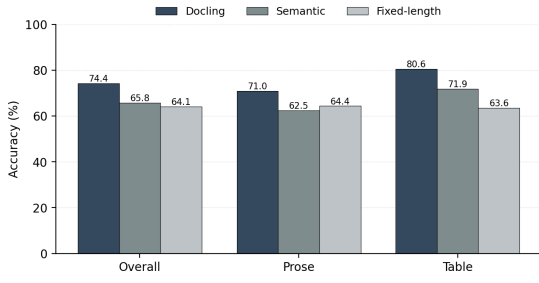


Fig. 2. Accuracy by chunking strategy, overall and by question type. The layout-aware (Docling) advantage is concentrated in table-grounded questions.

its advantage is largest on table-grounded questions (80.57%), consistent with keeping tables intact. Semantic and fixed-length chunking are close overall (65.80% and 64.13%), but their relative order reverses by question type: semantic is better on table questions (71.90% versus 63.62%), fixed-length slightly better on text questions (64.41% versus 62.51%). Docling achieves this with chunks averaging 3,195 characters, roughly double the 1,451 of fixed-length and the 1,019 of semantic chunking, which raises per-query context cost and is the principal cost of this advantage; mean response times are close across the three strategies, between 3.72 and 4.34 s.

A. Significance of Component Differences

Table II reports the paired McNemar comparisons of Section IV-F with Holm-adjusted p -values. Ten of the fourteen planned comparisons remain significant, including all four comparisons of the dense embedding models against FastText (48.22 to 52.44 points, all $p_{\text{Holm}} < 0.001$), omitted from the table for space. The four that are not significant are informative: multilingual-e5-large cannot be separated from TurkEmbed (+1.33 points, $p_{\text{Holm}} = 0.323$) or text-embedding-3-small (+1.50, 0.315), TurkEmbed and text-embedding-3-small are indistinguishable (+0.17, 0.900), and semantic and fixed-length chunking cannot be separated (+1.67, 0.211). In contrast, both comparisons of Docling against the alternatives, the generator comparison, and all three comparisons against Mursit-large are significant. The ranking of the top three embedding models should therefore not be over-interpreted, whereas the chunking and generator effects are statistically reliable.

TABLE II
PAIRED MCNEMAR COMPARISONS WITH HOLM CORRECTION

Comparison	Diff. (pts)	p	p_{Holm}
Docling vs. Fixed-length	+10.23	8×10^{-29}	< 0.001
Docling vs. Semantic	+8.57	1×10^{-23}	< 0.001
GPT-4o mini vs. qwen-plus	+2.96	7×10^{-11}	< 0.001
mE5-large vs. Mursit-lg	+4.22	1.5×10^{-5}	0.0001
TurkEmbed vs. Mursit-lg	+2.89	0.0034	0.021
te3-small vs. Mursit-lg	+2.72	0.0065	0.033
Semantic vs. Fixed-length	+1.67	0.053	0.211
mE5-large vs. te3-small	+1.50	0.105	0.315
mE5-large vs. TurkEmbed	+1.33	0.162	0.323
TurkEmbed vs. te3-small	+0.17	0.900	0.900

B. Chunking–Embedding Interaction

Because the design is fully crossed, the two retrieval-side factors can be examined jointly. Table III reports accuracy for each of the fifteen chunking-by-embedding pairs, each over 600 evaluations, and the pattern is not additive. Under Docling chunking the four modern embedding models fall within 1.33 points of one another (82.50–83.83), so the differences among them are negligible in this setting. Under semantic and fixed-length chunking the same four span 7.67 and 7.17 points, and the best embedding model changes: Mursit-large under Docling, multilingual-e5-large under semantic, and TurkEmbed under fixed-length. Along the other axis, sensitivity to the chunking strategy ranges, among these four embedding models, from 7.00 points for TurkEmbed to 15.00 for Mursit-large, while FastText varies by 22.17 points.

TABLE III
ACCURACY (%) BY CHUNKING STRATEGY AND EMBEDDING MODEL

Chunking	FastText	mE5-large	Mursit-lg	te3-small	TurkEmbed
Docling	39.50	82.50	83.83	83.00	83.00
Semantic	17.33	82.33	74.67	77.67	77.00
Fixed-length	25.83	75.17	68.83	74.83	76.00

The chunking strategy therefore determines how much the embedding choice matters: where segmentation preserves structure a strong embedding model can be chosen on cost, licensing, or latency grounds with little penalty, whereas where it is structure-agnostic the choice among the same embedding models costs up to 7.17 points, and the weakest of them, Mursit-large, falls fifteen points below its own best case. This also qualifies the ranking in Table I, which averages over chunking strategies and is not stable within them.

C. Performance by Document Type

Table IV reports accuracy by document type. Three features stand out. The balanced document is markedly the most difficult, 57.73% against 74.57% and 72.00%. The benefit of layout-aware chunking is uneven: on the text-heavy document the three strategies lie within 2.1 points (71.00–73.10), so segmentation has little influence there, whereas on the table-heavy and balanced documents Docling leads the best alternative by 6.0 and 12.3 points (82.30 versus 76.30, and 67.70 versus 55.40). The claim that chunking is the most influential factor is therefore conditional on the document containing tabular structure to preserve.

Third, one figure runs against the aggregate pattern: in the table-heavy document table questions score below text questions (73.58% versus 78.50%), whereas over the corpus as a whole they score above them (72.03% versus 65.98%). The reversal is confined to fixed-length chunking, which reaches 61.50% on that document’s table questions against 79.50% on its text questions; under Docling and semantic chunking table questions score slightly higher there (82.62 versus 81.00, and 76.62 versus 75.00). The aggregate figure therefore reflects the cost of splitting tables under one strategy, not an intrinsic difficulty of table-grounded questions.

TABLE IV
ACCURACY (%) BY DOCUMENT TYPE

Document type	Overall (%)	Text Q.	Table Q.
Table-heavy	74.57	78.50	73.58
Balanced	57.73	54.49	69.24
Text-heavy	72.00	72.65	51.11

D. Best Complete Configurations

Table V reports the five best configurations overall and on table-grounded questions. The best overall, Docling chunking with Mursit-large and GPT-4o mini, reaches 87.00%, about nineteen points above the 68.10% average. Docling appears in eight of the ten best overall configurations and GPT-4o mini in five; the five configurations ranked sixth to tenth all use qwen-plus, spanning 80.67–81.33% at 2.04–3.63 s, so they remain within 6.5 points of the best configuration at roughly half its latency. Because each configuration is evaluated on 300 questions rather than the 1,800 evaluations behind each embedding-level figure, its accuracy carries a 95% confidence interval of roughly ± 4 points, and the maximum over thirty configurations is upward-biased by selection; the identity of the single best configuration is therefore indicative rather than established. The top of the ranking is nonetheless consistent in its chunking strategy while varying in embedding model and generator, which is useful when one component is constrained by cost or availability.

The subset rankings differ: the configuration best on average is not best on every kind of question. On table questions the leaders reach 93.33% and are dominated by Docling and semantic chunking, and several qwen-plus configurations rank among the strongest, so the latency-accuracy gap narrows. On text questions accuracy is lower, the best configuration reaching 84.10%, and a fixed-length configuration enters the ten best there while none does so on table questions, consistent with fixed-length chunking being less damaging for flowing text than for tables, where an ill-placed boundary can separate a value from its row or column header.

TABLE V
TOP FIVE CONFIGURATIONS OVERALL AND ON TABLE-GROUNDED QUESTIONS

Chunking	Embedding	LLM	Acc. (%)	Time (s)
<i>Overall</i>				
Docling	Mursit-lg	GPT-4o mini	87.00	5.17
Docling	te3-small	GPT-4o mini	85.00	4.98
Docling	TurkEmbed	GPT-4o mini	84.67	4.86
Docling	mE5-large	GPT-4o mini	84.00	4.91
Semantic	mE5-large	GPT-4o mini	83.33	5.14
<i>Table-related questions</i>				
Docling	TurkEmbed	GPT-4o mini	93.33	4.29
Docling	mE5-large	GPT-4o mini	93.33	4.26
Docling	Mursit-lg	GPT-4o mini	92.38	4.50
Semantic	mE5-large	GPT-4o mini	92.38	4.61
Semantic	te3-small	qwen-plus	91.43	2.28

VI. DISCUSSION

First, the chunking strategy is the most influential factor for content containing tabular structure: Docling chunking improves table-question accuracy by 17.0 points over the

fixed-length baseline and by 8.7 points over semantic chunking because it retrieves whole tables rather than fragments, and both comparisons against the alternatives survive correction for multiple testing. The gain comes with larger chunks and therefore higher context-token consumption, and it is conditional: on the text-heavy document the three strategies differ by at most 2.1 points, so predominantly prose content gains little from structure-aware segmentation.

Second, the effect of embedding quality is concentrated among weak embedding models, and how much it matters depends on the chunking strategy. The four modern embedding models differ by about four points overall and the top three cannot be statistically separated, whereas FastText is much lower at 27.56%; the important decision is to avoid an inadequate embedding model rather than to choose among the strong ones. Table III refines this observation: under Docling the four span 1.33 points and the embedding models perform comparably, while under fixed-length chunking the spread grows to 7.17 points and Mursit-large falls fifteen points below its own best case.

A. Does Language Specialization Help Retrieval?

This study assumed that Turkish morphology would favour Turkish-oriented embedding models. The results do not support that assumption, and the appropriate conclusion is stronger than a simple reversal. multilingual-e5-large is nominally most accurate but not significantly ahead of TurkEmbed (+1.33, $p_{\text{Holm}} = 0.323$) or text-embedding-3-small (+1.50, 0.315), which are themselves indistinguishable (+0.17, 0.900). The defensible statement is not that a multilingual embedding model outperforms Turkish-oriented ones, but that *language specialization confers no measurable retrieval advantage on this corpus*: a strong multilingual embedding model, a Turkish-oriented one, and a commercial multilingual service perform equivalently, and only Mursit-large is reliably behind. For practitioners this is a useful negative result, since the embedding model can be chosen on cost, latency, or licensing grounds without an accuracy penalty.

The morphological motivation is nonetheless supported, but at a different level of the pipeline than anticipated. FastText, a word-level model built from subword n -grams without contextual composition, falls to 27.56% and is separated from every dense embedding model by roughly fifty points at $p_{\text{Holm}} < 0.001$ — consistent with the failure mode agglutinative morphology would predict, since representations that do not compose morphology contextually are less able to bring inflected variants of a lemma together. Because the design contains no non-agglutinative control language, however, this gap cannot be attributed to Turkish morphology specifically rather than to the absence of contextual composition in general. What large multilingual embedding models appear to provide is sufficient subword coverage and contextual composition to absorb Turkish morphology without language-specific pre-training, consistent with tokenization work locating the effect in the match between subword vocabulary and morphology rather than in the language identity of the training corpus [6].

Transfer from classification, where Turkish-specific pretraining does help [5], [15], to retrieval is thus not automatic.

B. Generator Choice and Trade-offs

Turning to generation, the faster generator is not the more accurate one. qwen-plus answers in under half the time of GPT-4o mini while giving up 2.96 points, small but statistically reliable ($p_{\text{Holm}} < 0.001$), and several qwen-plus configurations rank among the strongest on table questions. The generator choice therefore depends on whether a deployment prioritizes accuracy or latency, and can be made largely independently of chunking and embedding, whose leading settings are stable across both generators. For Turkish document question answering over mixed content, these results favour, on this corpus, layout-aware chunking with any of the three leading embedding models, reserving the faster generator for latency-sensitive settings and the more accurate one where correctness dominates; for predominantly prose documents the penalty for fixed-length chunking is modest.

VII. CONCLUSION

We compared chunking and embedding strategies for Turkish-language Retrieval-Augmented Generation over a fully crossed design of three chunking methods, five embedding models, and two generator LLMs, yielding 9,000 graded question-answer pairs from documents of three contrasting layout types. Docling chunking, multilingual-e5-large, and GPT-4o mini were the best individual components, but they did not compose into the best complete configuration: the highest-scoring pipeline paired Docling with Mursit-large and GPT-4o mini at 87.00%, about nineteen points above the average, even though Mursit-large ranked last of the four modern embedding models in the marginal comparison. This is what the interaction predicts, since under layout-aware chunking the embedding models are indistinguishable and the ranking between them is not stable. Paired tests show which differences are reliable: the chunking and generator effects are, whereas the ranking of the three leading embedding models is not, from which we conclude that language specialization confers no measurable retrieval advantage on this corpus. Analysing the two retrieval-side factors jointly shows further that the chunking strategy determines how much the embedding choice matters, since structure-preserving segmentation compresses the spread between strong embedding models to little more than one point. The preferred configuration also depends on content type, and accuracy should be weighed against cost and latency. Future work includes a larger and more varied document collection, validation of the automatic grades against human annotation on a stratified sample, and hybrid chunking that routes tabular and prose content to different strategies within one document.

REFERENCES

- [1] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.
- [2] G. Izacard and E. Grave, “Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering,” in *Proc. Conf. Eur. Chapter Assoc. Comput. Linguist. (EACL)*, 2021, pp. 874–880.
- [3] V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 6769–6781.
- [4] Y. Gao et al., “Retrieval-Augmented Generation for Large Language Models: A Survey,” arXiv preprint arXiv:2312.10997, 2023.
- [5] S. Schweter, “BERTurk: BERT Models for Turkish,” Zenodo, 2020, doi: 10.5281/zenodo.3770924.
- [6] C. Toraman, E. H. Yilmaz, F. Şahinuç, and O. Özcelik, “Impact of Tokenization on Language Models: An Analysis for Turkish,” *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 22, no. 4, pp. 1–21, 2023.
- [7] G. G. Şahin and M. Steedman, “Data Augmentation via Dependency Tree Morphing on Low-Resource Languages,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2018, pp. 5004–5009.
- [8] J. Johnson, M. Douze, and H. Jégou, “Billion-Scale Similarity Search with GPUs,” *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2021.
- [9] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2019, pp. 3982–3992.
- [10] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, “Language-agnostic BERT Sentence Embedding,” in *Proc. Annu. Meeting Assoc. Comput. Linguist. (ACL)*, 2022, pp. 878–891.
- [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol. (NAACL-HLT)*, 2019, pp. 4171–4186.
- [12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017.
- [13] L. Wang et al., “Multilingual E5 Text Embeddings: A Technical Report,” arXiv preprint arXiv:2402.05672, 2024.
- [14] OpenAI, “Embeddings,” OpenAI API Documentation, 2024. [Online]. Available: <https://platform.openai.com/docs/guides/embeddings>
- [15] A. Safaya, M. Abdullatif, and D. Yuret, “KUISAIL at SemEval-2020 Task 12: BERT-CNN for Offensive Speech Identification in Social Media,” in *Proc. 14th Workshop Semantic Eval. (SemEval)*, 2020, pp. 2054–2059.
- [16] M. A. Hearst, “TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages,” *Comput. Linguist.*, vol. 23, no. 1, pp. 33–64, 1997.
- [17] O. Koshorek, A. Cohen, N. Mor, M. Rotman, and J. Berant, “Text Segmentation as a Supervised Learning Task,” in *Proc. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol. (NAACL-HLT)*, 2018, pp. 469–473.
- [18] C. Auer et al., “Docling Technical Report,” arXiv preprint arXiv:2408.09869, 2024.
- [19] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “RAGAS: Automated Evaluation of Retrieval Augmented Generation,” in *Proc. Conf. Eur. Chapter Assoc. Comput. Linguist.: Syst. Demonstrations (EACL)*, 2024, pp. 150–158.
- [20] Ö. Ezerceli, G. Gümüşçekiççi, T. Erkoç, and B. Özenc, “TurkEmbed: Turkish Embedding Model on Natural Language Inference and Sentence Text Similarity Tasks,” in *Proc. IEEE 11th Int. Conf. Adv. Softw., Hardware Syst. Eng. (ASYU)*, 2025, doi: 10.1109/ASYU67174.2025.11208511.
- [21] Ö. Uğur et al., “Mecellem Models: Turkish Models Trained from Scratch and Continually Pre-trained for the Legal Domain,” arXiv preprint arXiv:2601.16018, 2026.
- [22] OpenAI, “GPT-4o mini,” OpenAI Model Documentation, 2024. [Online]. Available: <https://platform.openai.com/docs/models>
- [23] Alibaba Cloud, “Qwen-Plus,” Alibaba Cloud Model Studio Documentation, 2024. [Online]. Available: <https://www.alibabacloud.com/help/en/model-studio>
- [24] Meta AI, “Llama 3.3 70B Instruct,” Model card, 2024. [Online]. Available: <https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct>