How Chunk Size and Embedding Model Choice Affect Retrieval-Augmented Generation Quality and Cost: A Controlled Comparison on Consumer Hardware

Ason Gautam
Independent Researcher, Nepal
ason.gautam02@gmail.com
ason.gautam@deerwalk.edu.np
DOI: 10.5281/zenodo.21863445

Abstract

Retrieval-Augmented Generation (RAG) pipelines depend heavily on two design choices that are often set by habit rather than evidence. How large each document chunk should be, and which embedding model should encode those chunks. We run a controlled comparison of three chunk sizes (128, 256, and 512 words) crossed with three compact embedding models (BGE-small, E5-small, and MiniLM). This created a total of nine setups. On a fixed 800-question SQuAD 1.1 subset (54 passages), we measure indexing cost, retrieval quality (Recall@5), and end-to-end answer quality after a local Llama 3.2 3B model generates answers from the top-5 retrieved chunks (F1 and Exact Match).

Contrary to the common belief that smaller chunks retrieve better, Recall@5 rose with chunk size (r = +0.37), while answer F1 did not follow the same pattern (r = -0.31 with chunk size; only weakly related to Recall@5, r = +0.12). The strongest overall setup was 512-word chunks with E5-small (Recall@5 = 0.946; F1 = 0.415); the most indexing-efficient setup was 256-word chunks with MiniLM. Significance tests show that embedding-model gaps in Recall@5 are real, but most F1 gaps are not statistically clear at our generation sample size. On short passages like SQuAD, coarser chunks and retrieval-tuned embeddings tend to work best, and high retrieval scores alone are not a reliable stand-in for answer quality.

Keywords*: Retrieval-Augmented Generation; chunk size; sentence embeddings; dense retrieval; question answering; SQuAD; local LLM; cost–quality tradeoff*

1. Introduction

Retrieval-Augmented Generation (RAG) has become a common way to ground LLM outputs in external text, retrieving relevant chunks from a document index and letting the model condition its answer on them (Lewis et al., 2020). Two of the most important design choices in this pipeline, chunk size and embedding model selection, are still not well standardized. In practice, they're often decided using anecdotal guidance or leaderboard rankings like MTEB (Muennighoff et al., 2023), which don't account for downstream generation quality or indexing cost on limited hardware.

In this paper, we run a fully factorial study of chunk size and embedding model choice, keeping the dataset, question set, retrieval depth, generation model, and prompt template fixed across nine configurations. We evaluate each configuration along three axes: indexing cost, retrieval quality (Recall@5), and end-to-end answer quality.

Our starting hypothesis was that smaller chunks would improve retrieval accuracy, but at the cost of higher indexing time and storage, with some middle ground offering the best balance. We found this to be only partially true. Smaller chunks do reduce storage, and indexing cost turns out to be driven mainly by the embedding model rather than chunk size. But smaller chunks did not improve retrieval accuracy. If anything, we saw the opposite. We report the full results, including a quality-cost efficiency metric, a correlation analysis across factors, and a qualitative look at retrieval failures.

2. Related Work

Lewis et al. (2020) were the ones who really put Retrieval-Augmented Generation on the map, combining a dense passage retriever with a sequence-to-sequence generator. They showed that grounding generation in retrieved evidence leads to better factual accuracy than relying on the model's parametric memory alone. Around the same time, Karpukhin et al. (2020) introduced Dense Passage Retrieval (DPR) and showed that dual-encoder dense retrieval clearly beats sparse methods like BM25. This is largely why dense embedding similarity has become the go-to retrieval mechanism, and it's what we build on here.

Sentence embedding models have come a long way since Sentence-BERT (Reimers and Gurevych, 2019), which showed that contrastively fine-tuned encoders can produce embeddings where cosine similarity actually reflects semantic relatedness, making fast nearest-neighbor search over large collections possible. Two of the embedding families we test, E5 (Wang et al., 2022) and BGE (Xiao et al., 2023), grew out of this work. Both are trained with large-scale contrastive learning and hard-negative mining, and both sit near the top of MTEB (Muennighoff et al., 2023) among small encoders under 150M parameters. MiniLM (Wang et al., 2020) is older and wasn't built with retrieval in mind at all; it was distilled mainly for general sentence similarity. That's actually why we kept it in the study, as a baseline that isn't retrieval-tuned.

Chunking strategy hasn't gotten nearly the same attention. Most production RAG guides suggest chunk sizes somewhere between 200 and 500 tokens, often with some overlap, but these numbers are rarely tested against a fixed QA benchmark with matched retrieval depth and generator. Our study tries to fill that gap in three ways: we vary chunk size and embedding model together in a full factorial design rather than one at a time, we measure indexing cost (time, GPU memory, storage) alongside retrieval and generation quality, and we run everything on consumer-grade hardware (a single 4GB-class GPU) rather than data-center GPUs.

3. Method

3.1 Dataset and Task

We used SQuAD 1.1 for our dataset (Rajpurkar et al., 2016), Wikipedia-based, with each question tied to a paragraph and a specific answer span. The nice thing about it is you can check whether retrieval worked without needing a human to judge it, since the answer is always an exact span in the text. We pulled the first 800 questions out of the validation split. Once duplicate passages were dropped, 54 remained. These ranged pretty widely in length, from as short as 25 words up to 337, averaging somewhere around 105.

3.2 Chunking Strategy

Each unique passage is segmented into non-overlapping, fixed-length chunks by whitespace-delimited word count, using three chunk sizes: 128, 256, and 512 words. The final chunk of a passage retains whatever words remain after the preceding full-size windows are removed, so chunk length is variable at passage boundaries but capped at the target size elsewhere. Each chunk records the character offset span it occupies in the original passage, which is used at evaluation time to determine, for a given gold answer span, which chunk (if any) fully contains it. This chunk is treated as the retrieval ground truth for that question.

Table 1. Chunk count by target chunk size across the 54 unique passages.

Chunk size (words)	Resulting chunks	Mean chunks per passage
128	70	1.30
256	55	1.02
512	54	1.00

Because SQuAD passages are relatively short, the 256- and 512-word settings frequently retain each passage as a single chunk, while the 128-word setting is the only condition that reliably splits longer passages into multiple retrievable units. This property of the dataset is directly relevant to interpreting The retrieval results in Section 4.2.

3.3 Embedding Models

We evaluate three open-weight sentence embedding models spanning a range of retrieval-specific pre-training objectives, all in the same small parameter-count tier (≈22M–110M parameters) to keep the comparison relevant to constrained hardware:

- BAAI/BGE-small-en-v1.5 (BGE-small) — 384-dimensional embeddings; contrastively pre-trained with retrieval-specific hard-negative mining (BAAI General Embedding family).
- intfloat/E5-small-v2 (E5-small) — 384-dimensional embeddings; trained with weakly supervised contrastive pre-training followed by supervised fine-tuning explicitly targeting retrieval.

- sentence-transformers/all-MiniLM-L6-v2 (MiniLM-L6) — 384-dimensional embeddings; a distilled general-purpose sentence encoder not specifically optimized for retrieval, included as a non-retrieval-tuned baseline.

All three models are loaded via the sentence-transformers library with identical batch size (32) and default pooling, so that any measured difference reflects the pre-trained weights and objective rather than an inference configuration confound.

3.4 Indexing and Retrieval Pipeline

For each of the nine (chunk size × embedding model) configurations, every chunk of the corresponding size is encoded and inserted into a dedicated persistent Chroma vector collection, yielding nine independent vector indexes. At query time, a question is encoded with the same embedding model used to build the index being queried, and the five nearest chunks by vector similarity (Recall@5 depth, k = 5) are retrieved. Using a fixed k = 5 across all configurations isolates the effect of chunk size and embedding model from the effect of retrieval depth.

3.5 Answer Generation

The five retrieved chunks are concatenated into a fixed prompt template along with the question, and passed to a locally hosted Llama 3.2 (3B, instruction-tuned) via Ollama to produce a free-text answer. The prompt template, retrieval depth, generation model, and decoding settings are held constant across all nine configurations. The exact template, with the five retrieved chunks space-joined into a single context block, is shown below verbatim:

```
Answer the question using only the context below.
Context: {retrieved chunks}
Question: {question}
Answer:
```

No system prompt, few-shot examples, or answer-length instruction is used, only this template with default Ollama decoding settings. Retrieval evaluation (Recall@5) is computed over the full 800-question set; end-to-end generation evaluation, which is more expensive due to autoregressive decoding, is computed over a fixed 100-question subsample (the first 100 questions, held identical across all nine configurations), for a total of 900 scored generation trials.

3.6 Evaluation Metrics

- Indexing time (s): wall-clock time to encode all chunks of a given size with a given model and insert them into the vector index.
- Peak GPU memory (VRAM, MB): peak CUDA memory allocated during embedding model loading and encoding for that configuration.
- Index storage (MB): on-disk size of the persisted vector index directory.
- Recall@5: fraction of questions for which the chunk containing the gold answer span appears among the top-5 retrieved chunks.

- F1: token-level overlap F1 between the generated answer and the gold answer span, after lowercasing, punctuation removal, and article removal (standard SQuAD normalization).
- Exact Match (EM): fraction of generated answers that are identical to the gold answer span after the same normalization.
- Query latency and generation latency (ms): wall-clock time per retrieval call and per LLM generation call, respectively.

3.7 Statistical Testing Procedure

Numeric differences between configurations can arise from genuine effects or from sampling noise in a finite question set, so every headline metric in Section 4 is reported with a 95% bootstrap confidence interval, and key configuration comparisons are backed by a paired significance test rather than a raw ranking. Confidence intervals for Recall@5, F1, and Exact Match are computed by resampling the per-question scores for a given configuration with replacement 10,000 times and taking the 2.5th and 97.5th percentiles of the resulting distribution of means; this requires no additional model inference, only re-aggregation of scores already recorded per question.

For paired comparisons between two specific configurations on the same question set, binary outcomes (Recall@5 hit/miss, Exact Match) are compared with an exact two-sided McNemar test on the discordant pairs — questions where one configuration succeeds and the other fails — which tests whether one configuration is significantly better rather than merely numerically higher. The continuous F1 metric is compared with a paired bootstrap test: the difference in mean F1 is recomputed over 10,000 paired resamples of the shared question indices, and the two-sided p-value is the proportion of the bootstrap distribution that crosses zero. All tests use a significance threshold of $\alpha = 0.05$ and a fixed random seed (42) for reproducibility. Full pairwise results are given in Section 4.5 and Appendix B.

3.8 Hardware and Software Environment

All indexing, retrieval, and embedding computation was executed on an NVIDIA GeForce GTX 1650 Ti (4 GB VRAM) using PyTorch with CUDA acceleration. Answer generation used Llama 3.2 3B served locally through Ollama. The vector store is ChromaDB in persistent (on-disk) mode. This hardware profile is representative of a single consumer laptop GPU rather than a data-center accelerator, which is a deliberate choice: the cost measurements in this study are intended to reflect the constraints faced when deploying RAG pipelines on limited local hardware.

4. Results

We report results for all nine (chunk size × embedding model) configurations across three stages of the pipeline: indexing cost, retrieval quality (Recall@5, evaluated over all 800 questions), and end-to-end generation quality (F1 and Exact Match, evaluated over the fixed 100-question subsample). Table 2 presents the complete set of measured metrics; Sections 4.1–4.4 discuss each stage in turn, Section 4.5 reports statistical significance tests, and Section 4.6 presents a qualitative failure analysis.

Table 2. Complete results across all nine configurations. Recall@5 is computed over all 800 questions; F1 and EM are computed over the fixed 100-question generation subsample.

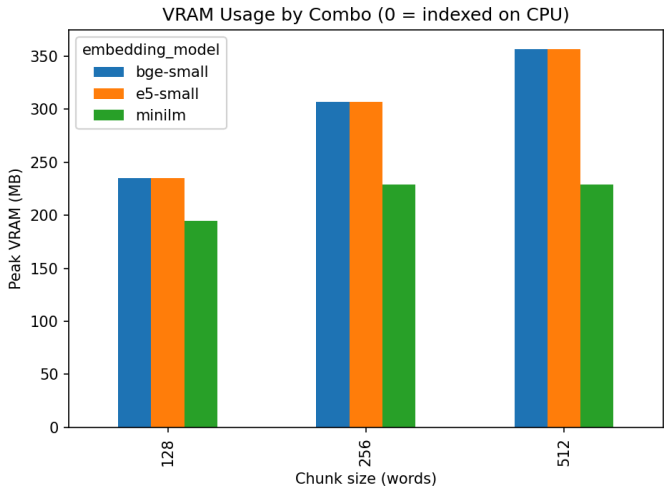
Chunk	Embedding models	n chunks	Index(s)	VRAM (MB)	Storage (MB)	Recall@5	F1	EM
128	BGE-small	70	1.07	235.4	0.88	0.871	0.402	0.18
128	E5-small	70	0.80	235.4	0.88	0.922	0.379	0.16
128	MiniLM	70	0.48	194.8	0.88	0.864	0.382	0.19
256	BGE-small	55	0.79	307.1	0.83	0.901	0.364	0.16
256	E5-small	55	0.78	301.1	0.83	0.945	0.372	0.18
256	MiniLM	55	0.46	229.0	0.83	0.886	0.388	0.22
512	BGE-small	54	0.92	357.0	0.83	0.905	0.354	0.14
512	E5-small	54	0.93	357.0	0.83	0.946	0.415	0.20
512	MiniLM	54	0.47	229.0	0.83	0.891	0.343	0.16

4.1 Indexing Cost

Indexing cost is driven primarily by the embedding model rather than chunk size. Averaged across chunk sizes, MiniLM required 0.47 s for indexing, substantially faster than E5-small (0.84 s) and BGE-small (0.93 s), consistent with its smaller parameter count. Peak GPU memory followed the same ordering and increased with chunk size for the retrieval-tuned models. For BGE-small and E5-small, peak VRAM increased from approximately 235 MB at 128-word chunks to 358 MB at 512-word chunks, reflecting longer token sequences per batch. Index storage was primarily determined by the number of chunks, making the 128-word configuration (70 chunks) the largest despite its shorter chunks.

The cheapest configuration was MiniLM with 256-word chunks, requiring 0.46 s for indexing. To reduce measurement noise from GPU warm-up, OS scheduling, and caching, indexing time and peak VRAM were measured three times per configuration using independent subprocesses. Reported values are the means of these measurements. Variability was low: the largest standard deviation was 0.355 s for

indexing time, while peak VRAM had a standard deviation of 0.0 MB across all configurations. The 256-word MiniLM configuration measured 0.460 ± 0.008 s, while the best end-to-end configuration, 512-word E5-small, measured 0.929 ± 0.008 s.



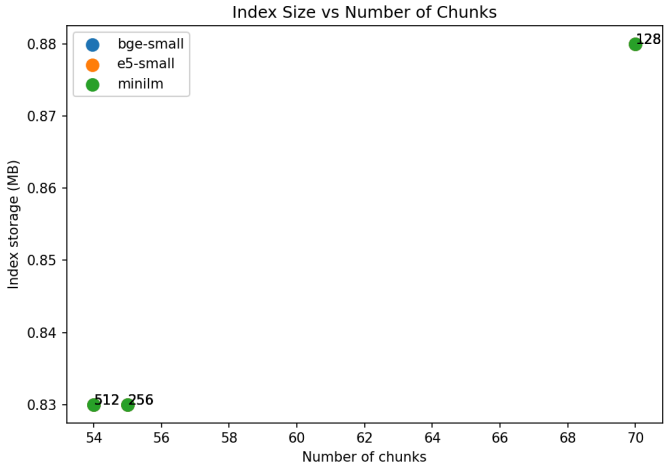


Figure 1. Peak GPU memory (VRAM) by chunk size and embedding model.

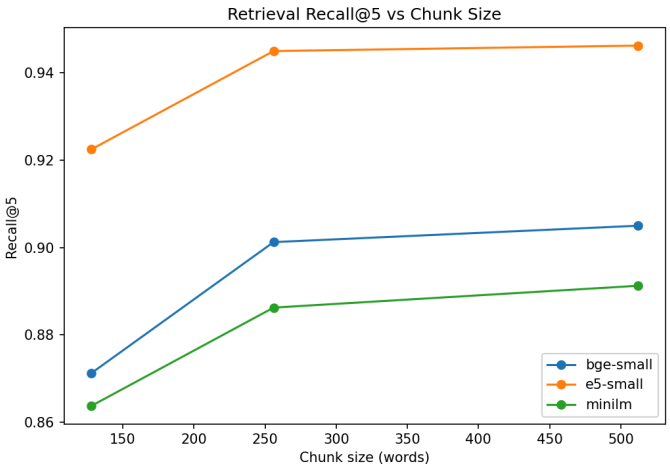
Figure 2. On-disk index storage as a function of the number of indexed chunks.

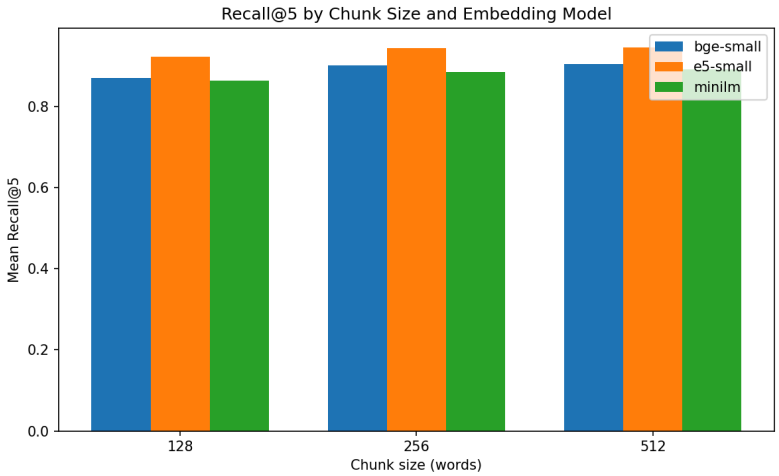
4.2 Retrieval Quality

Recall@5 ranged from 0.864 to 0.946 across the nine configurations. Two effects are clearly separable. First, embedding model choice has the larger effect: E5-small achieved the highest mean Recall@5 (0.938), followed by BGE-small (0.892) and MiniLM (0.880), matching the expectation that retrieval-specific contrastive pre-training (E5, BGE) outperforms a general-purpose sentence encoder (MiniLM) on a retrieval task.

Second, and contrary to our initial hypothesis, Recall@5 increases with chunk size rather than decreasing: mean Recall@5 rises from 0.886 at 128 words to 0.911 at 256 words to 0.914 at 512 words ($r = +0.37$ between chunk size and Recall@5). The most likely explanation is dataset-specific: because SQuAD passages average only 105 words, the 256- and 512-word settings retain most passages as a single chunk, so the retrieval target for most questions is effectively "find this one passage," a comparatively easy discrimination task among 54–55 candidates. The 128-word setting instead splits longer passages into multiple partial chunks, occasionally separating the gold answer span from surrounding disambiguating context and increasing competition among near-duplicate partial chunks from the same passage.

The best single configuration for retrieval was 512-word chunks with E5-small (Recall@5 = 0.946, 95% CI [0.930, 0.961], n = 800 questions, bootstrap-resampled). Section 4.5 shows that the embedding-model gap in Recall@5 is statistically significant, while the chunk-size gap, though numerically consistent, is only significant between the extremes (128 vs. 512 words) and not between adjacent sizes.





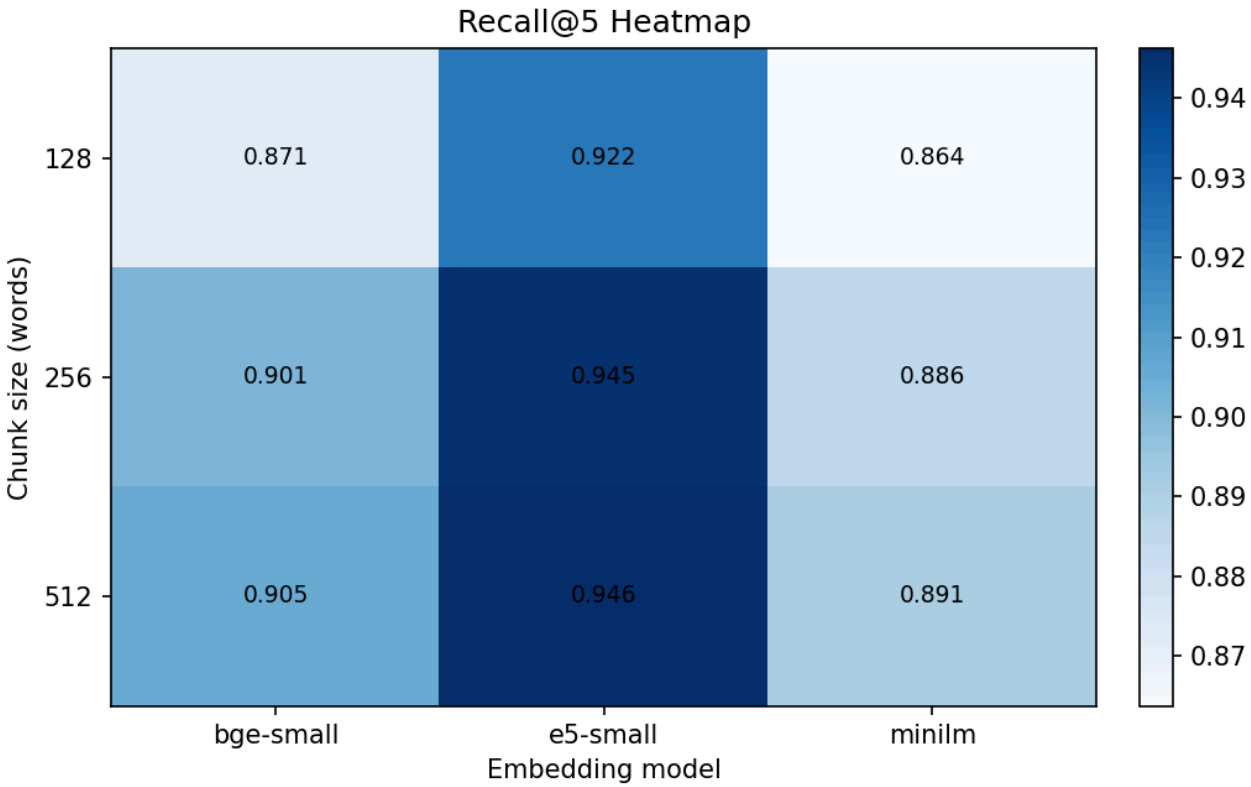


Figure 3. Mean Recall@5 as a function of chunk size, one line per embedding model.

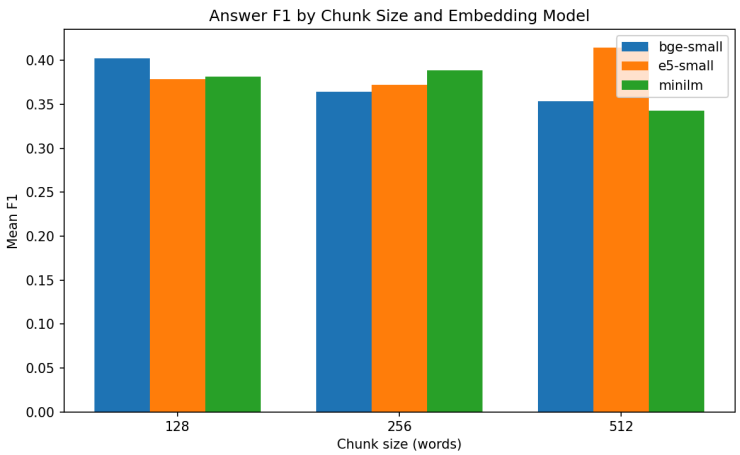
Figure 4. Recall@5 by chunk size and embedding model (grouped bar view).

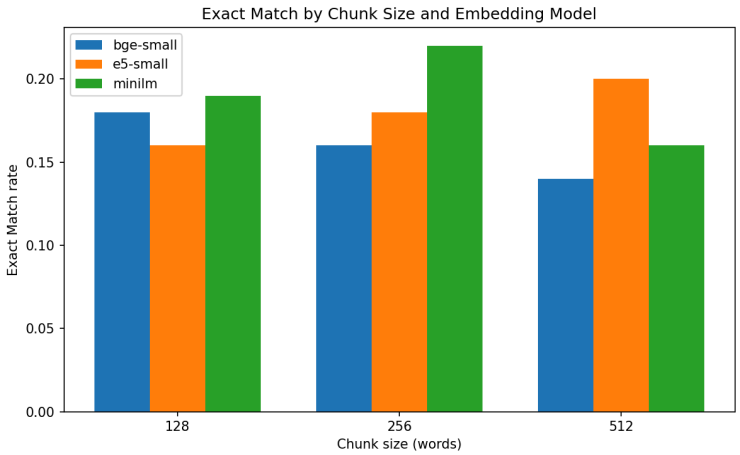
Figure 5. Recall@5 heat map across all nine configurations.

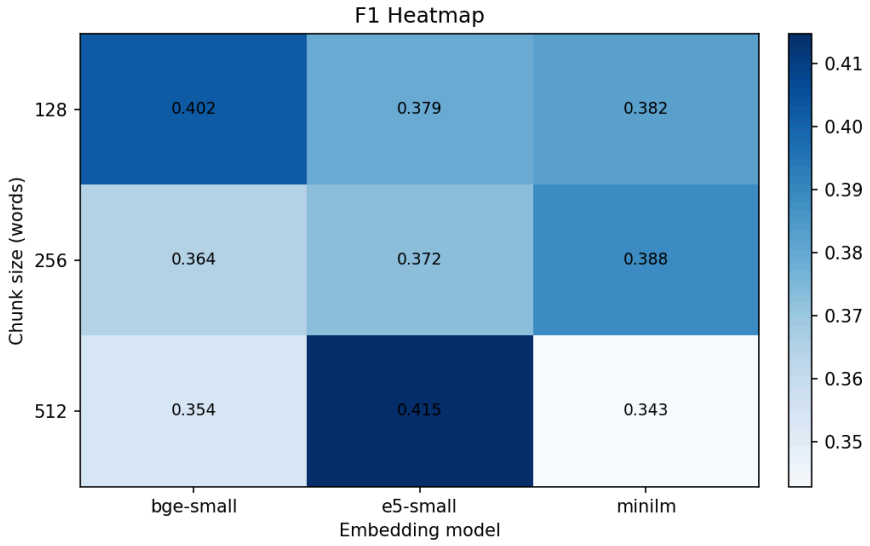
4.3 End-to-End Answer Quality

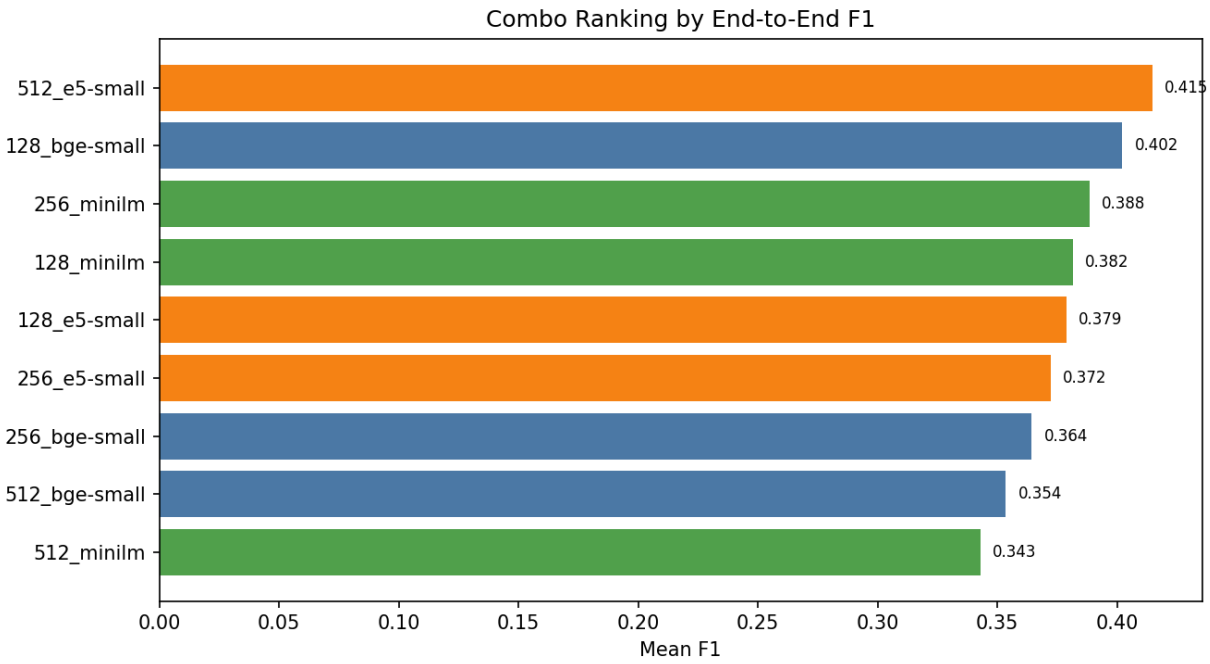
F1 ranged from 0.343 to 0.415 across configurations — a narrower relative spread than Recall@5, and the ranking by F1 does not track the ranking by Recall@5 (r = +0.12 across the nine configurations, i.e. a weak positive but far from perfect relationship). The best end-to-end configuration was 512-word chunks with E5-small (F1 = 0.415, 95% CI [0.345, 0.486], n = 100; Recall@5 = 0.946), while the highest Exact Match rate was achieved by 256-word chunks with MiniLM (EM = 0.22, 95% CI [0.14, 0.30]), despite
this configuration not having the highest F1 or Recall@5 — indicating that MiniLM-based retrieval, though weaker at finding the exact gold chunk, tends to surface chunks that lead the generator to produce short, precisely-matching answers when it does succeed. The wide, overlapping confidence intervals on F1 across configurations (visible in Table 5, Appendix A) are themselves informative: at n = 100 questions, the generation-stage sample is not large enough to distinguish most configurations' F1 scores from one another with statistical confidence, a point developed further in Section 4.5.

Chunk size correlates weakly negatively with F1 (r = -0.31), the opposite direction from its correlation with Recall@5. This divergence is the central empirical finding of this study: improving retrieval recall by increasing chunk size does not translate into proportionally better generated answers, because larger chunks also inject more irrelevant surrounding text into the generator's context window, which a 3B-parameter model appears to have some difficulty filtering when composing a concise answer.









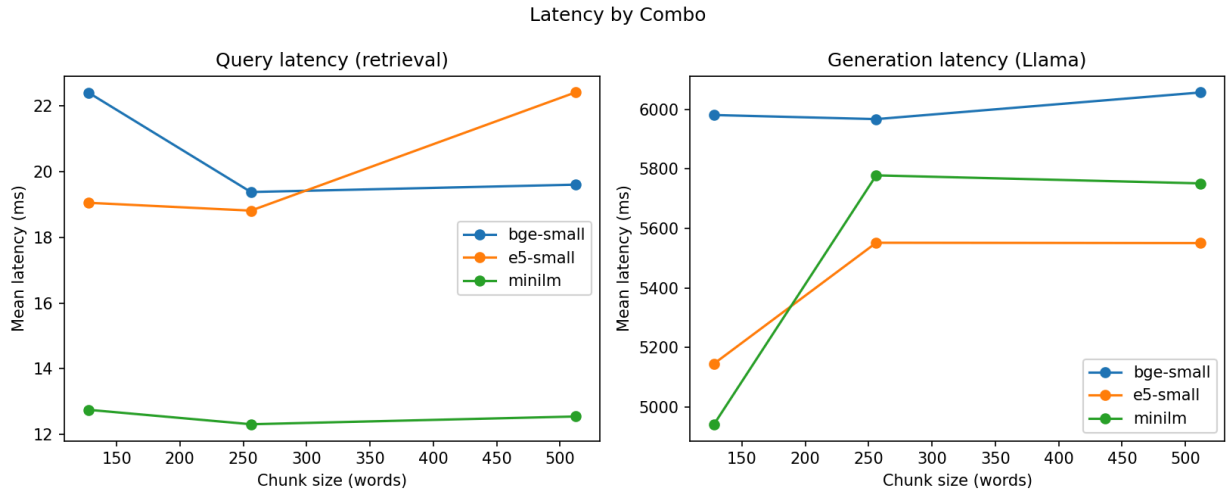


Figure 6. Mean end-to-end F1 by chunk size and embedding model.

Figure 7. Exact Match rate by chunk size and embedding model.

Figure 8. End-to-end F1 heat map across all nine configurations.

Figure 9. All nine configurations ranked by mean F1.

Figure 10. Mean retrieval query latency and generation latency by configuration.

4.4 Quality-Cost Tradeoff

To jointly summarize quality and indexing cost, we define efficiency as F1 divided by indexing time (F1 per indexing-second). By this measure, 256-word chunks with MiniLM was the most efficient configuration (efficiency = 0.845), driven by MiniLM's substantially lower indexing cost rather than by superior answer quality in absolute terms. This distinction matters for deployment decisions under a fixed compute budget: the highest-quality configuration and the most cost-efficient configuration are not the same, and the appropriate choice depends on whether the deployment is index-build-bound (favoring MiniLM) or quality-bound (favoring E5-small).

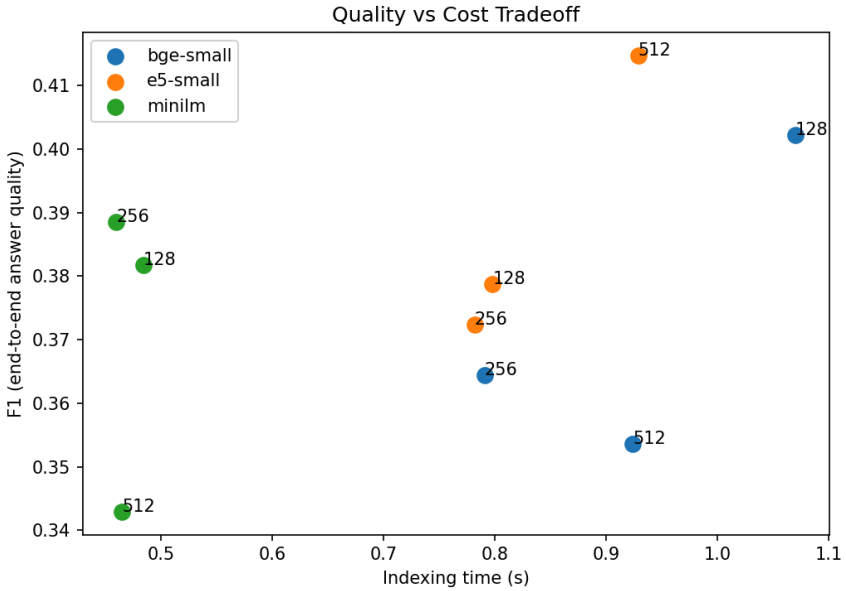


Figure 11. Quality-cost tradeoff: mean F1 versus indexing time, annotated by chunk size.

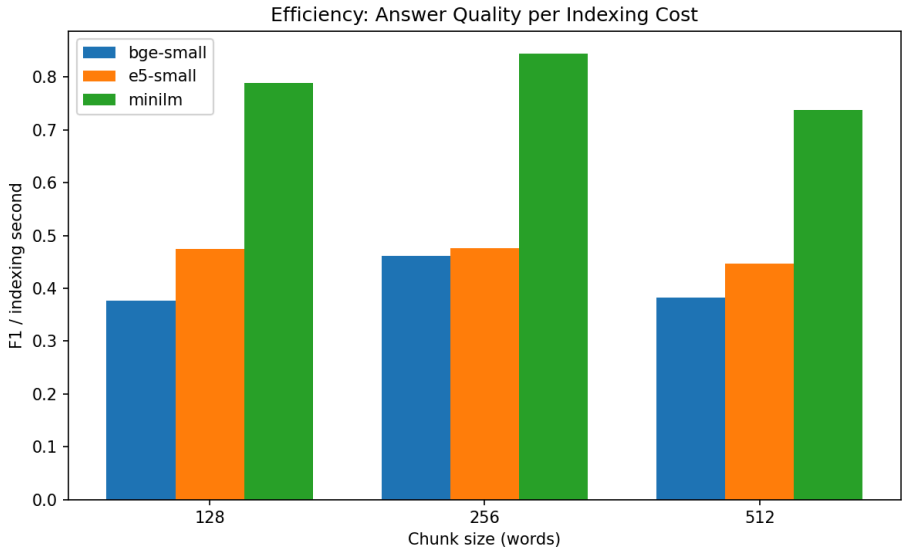


Figure 12. Efficiency (F1 per indexing-second) by chunk size and embedding model.

4.5 Statistical Significance

To move beyond ranking configurations by raw numbers, we ran the paired significance tests described in Section 3.7 on five configuration pairs chosen to isolate one factor at a time: the chunk-size effect within the strongest embedding model (E5-small), the embedding-model effect at the strongest chunk size (512 words), and the overall #1-vs-#2 configuration by F1. Table 3 summarizes the results; full contingency counts are in Appendix B.

Table 3. Pairwise significance tests between key configurations. * denotes $p < 0.05$. Recall@5 and EM p-values are from an exact two-sided McNemar test on discordant question pairs; F1 p-values are from a paired bootstrap test on the mean difference (10,000 resamples).

Comparison	Recall@5 (McNemar)	F1 (paired bootstrap)	EM (McNemar)
512_E5-small vs 128_e5-small	0.946 vs 0.922 (p=0.0002) *	0.415 vs 0.379 (p=0.1908)	0.200 vs 0.160 (p=0.3877)
512_E5-small vs 256_e5-small	0.946 vs 0.945 (p=1.0000)	0.415 vs 0.372 (p=0.2238)	0.200 vs 0.180 (p=0.8318)
512_E5-small vs 512_bge-small	0.946 vs 0.905 (p=0.0000) *	0.415 vs 0.354 (p=0.0890)	0.200 vs 0.140 (p=0.2863)
512_E5-small vs 512_MiniLM	0.946 vs 0.891 (p=0.0000) *	0.415 vs 0.343 (p=0.0304) *	0.200 vs 0.160 (p=0.5034)
512_E5-small vs 128_BGE-small	0.946 vs 0.871 (p=0.0000) *	0.415 vs 0.402 (p=0.7326)	0.200 vs 0.180 (p=0.8318)

The pattern is consistent across all five comparisons: Recall@5 differences are statistically significant in every case, including the embedding-model comparisons at 512 words (E5-small vs. BGE-small and E5-small vs. MiniLM, both $p < 0.001$) and the chunk-size extremes comparison (128 vs. 512 words with E5-small fixed, $p < 0.001$), because the 800-question retrieval evaluation has ample statistical power to detect even moderate effect sizes. F1 differences are smaller in absolute terms and, at $n = 100$ generation trials per configuration, only one of the five comparisons reaches significance at $\alpha = 0.05$ (E5-small vs. MiniLM at 512 words). Exact Match differences do not reach significance in any of the five comparisons at this sample size. This is an important qualifier on the ranking presented in Table 2 and Section 4.3: the ordering of configurations by mean F1 is our best point estimate, but for most pairs of configurations we cannot rule out that the true F1 difference is zero given only 100 generation trials, whereas we can state with high confidence that the retrieval-quality ordering is real.

4.6 Failure Case Analysis

To understand the residual error in the strongest configuration (512-word chunks, E5-small), we inspected questions for which the gold-answer chunk was absent from the top-5 retrieved results.

Representative failures fall into two categories. First, questions requiring aggregation or comparison across multiple passages (e.g. counting how many teams share a record, or comparing two teams' season records) have no single chunk that contains the full answer, so chunk-level retrieval is structurally unable to succeed regardless of chunk size or embedding model. Second, several failures involve near-duplicate passages describing the same event (e.g. multiple passages about the same game from different narrative angles), where the embedding models correctly identify the general topic but retrieve a topically similar sibling passage instead of the one containing the specific gold answer span. Table 4 lists representative examples.

Table 4. Representative retrieval failures for the best-performing configuration (512-word chunks, E5-small).

Question	**Gold answer (not retrieved in top 5)**
Who did the Broncos prevent from going to the Super Bowl?	New England Patriots
How many teams have been in the Super Bowl eight times?	four
What was the win/loss ratio in 2015 for the Carolina Panthers during their regular season?	15–1
What were the win/loss game stats for the Denver Bronco's regular season in 2015?	12–4

5. Discussion

This study set out to test whether smaller retrieval chunks improve accuracy at the cost of higher indexing overhead. The evidence only partially supports this hypothesis, and in one important respect contradicts it directly.

Hypothesis verdict

The cost side of the hypothesis is qualified support: indexing cost is driven far more strongly by embedding model choice than by chunk size on this dataset, though peak GPU memory does increase with chunk size for the retrieval-tuned models, and storage cost is highest for the smallest chunk size purely due to the larger number of vectors, as expected. The accuracy side of the hypothesis is not supported: smaller chunks did not improve Recall@5; larger chunks did. We attribute this to the specific structure of SQuAD, whose passages are short enough that finer chunking fragments otherwise-coherent evidence rather than isolating it. This is an important caveat for generalizing chunk-size guidance across corpora — for longer source documents where a single passage would span thousands of words, the fragmentation argument for smaller chunks may hold with the opposite sign, since our over-512-word setting here essentially degenerates to “one chunk per document” for most passages.

Retrieval quality is not a reliable proxy for answer quality

The weak recall–F1 correlation ($r = +0.12$) is arguably the more actionable finding for practitioners than the chunk-size result itself. Optimizing a RAG pipeline purely against Recall@k, as is common practice, does not guarantee a proportional improvement in the quality of the final generated answer, because the generator's ability to extract and phrase a concise answer from a noisy or overlong context is an independent bottleneck. This argues for evaluating RAG systems end-to-end whenever the downstream deployment is a QA or chat application, rather than relying solely on retrieval-stage metrics.

Statistical confidence differs sharply between the retrieval and generation stages

The significance testing in Section 4.5 adds an important qualifier to the above claims: it is not merely that recall and F1 are weakly correlated, but that we have very different levels of statistical confidence in the two stages' results. With 800 questions, every embedding-model comparison on Recall@5 is significant at $p < 0.001$, so we can state with confidence that E5-small genuinely retrieves better than BGE-small and MiniLM on this corpus. With only 100 generation trials per configuration, the corresponding F1 gaps — while pointing in the same direction — are mostly indistinguishable from noise. Only the E5-small-vs-MiniLM comparison at 512 words reached significance on F1 ($p = 0.030$); the remaining four factor-isolating comparisons did not. We therefore treat the Recall@5 ranking as a well-supported empirical finding and the F1 ranking as a directionally suggestive but statistically underpowered point estimate, and we report both framings explicitly rather than presenting only the more impressive-looking numeric ranking.

Embedding model choice generalizes better than chunk size

Unlike the chunk-size effect, the ranking of embedding models by Recall@5 (E5-small > BGE-small > MiniLM) matches their public retrieval-benchmark rankings and is unlikely to be an artifact of this particular dataset, since it reflects each model's pre-training objective rather than corpus-specific passage length. Practitioners with a fixed indexing-time budget should treat embedding model selection as the higher-leverage decision relative to chunk-size tuning, at least for corpora with passage lengths comparable to SQuAD.

6. Limitations

- Dataset scope: the study uses a single benchmark (SQuAD) whose passages are short (mean 105 words); the chunk-size findings may not transfer to corpora with much longer source documents (e.g. technical manuals or legal filings), where finer chunking could behave differently.
- Chunking method: chunks are formed by whitespace word count with no overlap and no respect for sentence or paragraph boundaries; semantic or sentence-aware chunking, and chunk overlap, were not evaluated and could change the retrieval-quality trend.
- Generation sample size and statistical power: end-to-end generation metrics (F1, EM) are computed on a fixed 100-question subsample per configuration rather than the full 800-question set, to keep local LLM inference cost tractable. As shown directly by the significance tests in Section 4.5, this sample size gives the retrieval stage ($n = 800$) ample power to detect real

differences but leaves the generation stage underpowered: four of five factor-isolating F1 comparisons could not be distinguished from noise at $\alpha = 0.05$. Scaling generation evaluation to the full question set, as proposed in Section 7, would directly address this.

- Indexing cost repeatability: indexing time and peak VRAM were each measured 3 times per configuration in independent subprocesses to report a mean and standard deviation rather than a single sample (Section 4.1). Three repeats is enough to characterize measurement noise on this single, largely idle GPU, but is still a small sample; on shared or thermally-throttled hardware, additional repeats would give a tighter estimate of the true variance.
- Single generator model: only one local LLM (Llama 3.2, 3B parameters) was used for generation; results may differ with larger or differently instruction-tuned generators, which could be more robust to longer or noisier retrieved context.
- Fixed retrieval depth: only $k = 5$ was evaluated; the interaction between chunk size and retrieval depth (e.g. whether smaller chunks benefit from a larger k) was not explored.
- Single hardware configuration: all cost measurements (time, VRAM) reflect one GPU (NVIDIA GTX 1650 Ti, 4 GB); absolute costs will differ on other hardware, though the relative ordering between embedding models is expected to be stable.

7. Conclusion and Future Work

Across a fully factorial evaluation of three chunk sizes and three embedding models on a SQuAD-based question-answering task, we find that embedding model choice dominates chunk-size choice as a driver of retrieval quality, that larger chunks (256–512 words) achieve higher Recall@5 than smaller chunks on this short-passage dataset, and that Recall@5 is only weakly predictive of end-to-end answer quality once a language model generates the final response. The strongest overall configuration was 512-word chunks with E5-small, while the most indexing-efficient configuration was 256-word chunks with MiniLM,
illustrating that “best quality” and “lowest cost to build” are distinct optimization targets that practitioners should evaluate separately rather than assuming they coincide.
Future work should extend this design along three axes: (1) evaluating on longer-document corpora where finer chunking is more likely to matter, (2) testing sentence-aware and overlapping chunking strategies against the fixed-word-count baseline used here, and (3) scaling the generation evaluation to the full question set and to larger generator models to test whether the weak recall–F1 correlation observed here persists as generation capacity increases.

References

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of EMNLP 2020.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems (NeurIPS) 33.

Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2023). MTEB: Massive Text Embedding Benchmark. Proceedings of EACL 2023.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. Proceedings of EMNLP 2016.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP-IJCNLP 2019.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., & Wei, F. (2022). Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv preprint arXiv:2212.03533.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. Advances in Neural Information Processing Systems (NeurIPS) 33.

Xiao, S., Liu, Z., Zhang, P., & Muennighoff, N. (2023). C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv preprint arXiv:2309.07597. (BGE model family.)

Appendix A. Full Per-Configuration Results

This appendix reports the complete set of measured statistics per configuration, including bootstrap 95% confidence intervals for Recall@5 and F1 plus mean query and generation latency, supplementing the summary in Table 2.

Table 5. Full statistics with bootstrap 95% confidence intervals (10,000 resamples) for all nine configurations. Recall@5 CIs are computed over n=800 questions; F1 CIs over n=100.

Configuration	Recall@5	Recall@5 95% CI	Query ms	F1	F1 95% CI	EM	Gen ms
128_BGE-small	0.871	[0.848, 0.895]	22.4	0.402	[0.338, 0.468]	0.18	5981
128_E5-small	0.922	[0.904, 0.940]	19.1	0.379	[0.312, 0.447]	0.16	5146
128_MiniLM	0.864	[0.840, 0.887]	12.8	0.382	[0.312, 0.454]	0.19	4942
256_BGE-small	0.901	[0.880, 0.921]	19.4	0.364	[0.300, 0.433]	0.16	5967
256_E5-small	0.945	[0.929, 0.960]	18.8	0.372	[0.305, 0.440]	0.18	5552
256_MiniLM	0.886	[0.864, 0.907]	12.3	0.388	[0.317, 0.463]	0.22	5778
512_BGE-small	0.905	[0.884, 0.925]	19.6	0.354	[0.291, 0.418]	0.14	6057
512_E5-small	0.946	[0.930, 0.961]	22.4	0.415	[0.345, 0.486]	0.20	5551
512_MiniLM	0.891	[0.869, 0.912]	12.5	0.343	[0.277, 0.411]	0.16	5751

Correlation matrix summary (Pearson r across the nine configuration-level means): chunk size vs. Recall@5 = +0.373; chunk size vs. F1 = -0.305; Recall@5 vs. F1 = +0.124; indexing time vs. F1 = +0.376.

Appendix B. Pairwise Significance Test Details

This appendix reports the full contingency counts and bootstrap statistics underlying the pairwise significance tests summarized in Table 3 (Section 4.5), for readers who wish to verify or recompute the tests. For each comparison, b is the number of questions where configuration A succeeded and B failed, and c is the number where B succeeded and A failed; the McNemar test is computed on these discordant pairs only. F1 statistics are from the paired bootstrap described in Section 3.7.

Table 6. Full contingency counts (McNemar) and bootstrap confidence intervals on the mean difference (F1) for each pairwise comparison in Section 4.5.

Comparison	Recall b,c	Recall p	F1 diff 95% CI	F1 p	EM b,c	EM p
512_E5-small vs 128_e5-small	b=22, c=3	0.0002	[-0.019, 0.091]	0.1908	b=8, c=4	0.3877
512_E5-small vs 256_e5-small	b=1, c=0	1.0000	[-0.027, 0.112]	0.2238	b=12, c=10	0.8318
512_E5-small vs 512_bge-small	b=47, c=14	0.0000	[-0.009, 0.133]	0.0890	b=14, c=8	0.2863
512_E5-small vs 512_MiniLM	b=63, c=19	0.0000	[0.007, 0.136]	0.0304	b=12, c=8	0.5034
512_E5-small vs 128_bge-small	b=76, c=16	0.0000	[-0.056, 0.082]	0.7326	b=12, c=10	0.8318

Method note: all bootstrap resampling in this study (10,000 resamples per estimate) used NumPy's default bit generator with a fixed seed (42) for exact reproducibility. McNemar p-values are exact two-sided binomial test p-values on the discordant pairs, not the chi-square approximation, since several discordant counts in this study are small enough that the chi-square approximation would be unreliable.