Index-Time Prefix Injection for Multi-Tenant Retrieval: Improving Search Relevance Without Model Fine-Tuning

Vaibhav Varshney
ServiceNow
vaibhav.varshney@servicenow.com

Manjunatha Naik MC
ServiceNow
manjunathanaik.mc@servicenow.com

Abstract

Multi-tenant enterprise search platforms serve hundreds of customers through a single shared retrieval model. Fine-tuning on individual customer data is typically prohibited by contractual and regulatory constraints, and maintaining per-customer models does not scale. We present *index-time prefix injection*, a training-free method that improves retrieval relevance by prepending domain-descriptive natural-language prefixes to documents during indexing. For example, prepending "IT service management knowledge article:" to an IT knowledge base shifts its embeddings into a tighter, more domain-coherent region of the vector space. Prefixes are discovered through a tiered strategy: LLM-based generation from document samples when data policies allow, domain-expert curation when they do not, and a standardized prefix library as fallback. Deployed across 18 languages and 400+ customer instances, the approach yields 3–8% Hit@5 improvements with zero model training. A/B tests confirm a 4.2% CTR lift. We describe the system design, evaluation at scale, and deployment lessons including failure modes.

1 Introduction

We operate the search stack for a large enterprise platform serving hundreds of customers across industries and geographies. A single multilingual bi-encoder handles all retrieval, with datasources ranging from IT incident knowledge bases in Japanese to German legal compliance handbooks. Domain-adaptive fine-tuning is impractical here: customer data is contractually isolated, per-domain model variants do not scale, and new customers expect search to work on day one.

This raises a practical question: given a frozen encoder, what can we do at the *data preparation* stage to improve retrieval per datasource? Instruction-tuned models like E5 (Wang et al., 2022b) and INSTRUCTOR (Su et al., 2023) already use natural-language prefixes to shape encoder behavior, but these are task-generic (e.g., "Represent this sentence for retrieval: ...") and domain-agnostic. We investigated replacing them with domain-specific prefixes tailored to each datasource.

Prepending "IT service management knowledge article:" to an IT knowledge base shifts its embeddings into a tighter, domain-coherent region of the vector space, reducing cross-domain contamination—where documents from one datasource leak into top results for queries targeting another—from 12–18% to 5–8% of top-5 results (measured on a stratified sample of 2K queries across 15 customer instances; see Section 6). The implementation is minimal—string concatenation in the indexing pipeline—but reliably discovering effective prefixes across thousands of datasources in 18 languages required a more systematic approach.

This paper makes the following contributions:

1. A document-side prefix injection scheme applied at index time, requiring no model training, compatible with any instruction-tuned bi-encoder.
2. A tiered prefix discovery pipeline: LLM-generated from document samples (where data policies permit), domain-expert curated, or selected from a standardized library.
3. Production evaluation at scale (18 languages, 400+ customer instances) with both offline metrics and online A/B test results.
4. Analysis of failure modes and practical deployment considerations.

2 Related Work

Instruction-tuned embeddings. E5 (Wang et al., 2022b) introduced `query:`/`passage:` tags; E5-

Instruct (Wang et al., 2024a) and INSTRUCTOR (Su et al., 2023) generalized this to natural-language task instructions. BGE (Xiao et al., 2024), GTE (Li et al., 2023), NV-Embed (Lee et al., 2024), and GritLM (Muennighoff et al., 2024) follow similar paradigms, but in all cases prefixes remain *task-generic* and domain-agnostic. TART (Asai et al., 2023) and InstructIR (Oh et al., 2024) prepend task descriptions to *queries*, requiring model training on instruction-augmented data. We instead apply *domain-specific* prefixes to *documents* at index time with a frozen encoder.

Domain adaptation for retrieval. Standard approaches—continued pretraining (Gururangan et al., 2020), synthetic data generation (Wang et al., 2022a), domain-specific models (Chalkidis et al., 2020; Gu et al., 2021)—all require training on target-domain data. Parameter-efficient alternatives like prefix-tuning (Li and Liang, 2021), prompt tuning (Lester et al., 2021), and DPTDR (Tang et al., 2022) still need gradient updates, which multi-tenant data isolation prohibits. The closest prior work is Hashemi et al. (2023), who prepend domain descriptions to queries without requiring a target corpus. Their approach differs in operating on the query side (requiring per-query domain knowledge at inference), targeting single-domain retrieval, and being evaluated on public benchmarks. We apply prefixes to documents at index time, avoiding query-time overhead and naturally supporting cross-datasource search.

Dense and multilingual retrieval. We build on the bi-encoder paradigm (Karpukhin et al., 2020; Khattab and Zaharia, 2020; Izacard et al., 2022) with multilingual models like E5 (Wang et al., 2024b) as backbone. Alternatives include mGTE (Zhang et al., 2024) and BGE-M3 (Chen et al., 2024). HyDE (Gao et al., 2023) reshapes the query embedding space via hypothetical document generation but requires LLM inference per query. None address per-datasource domain adaptation with a frozen encoder.

Positioning. Table 1 summarizes these distinctions. Prior work either trains models with task-level instructions or uses fixed generic prefixes. We occupy a different point: *domain-specific* prefixes discovered per datasource, applied to documents at index time, with no model training.

System	Prefix	Applied	Training	Scope
E5 / BGE	Generic	Q + D	Required	Task
INSTRUCTOR	Generic	Q + D	Required	Task
TART	Per-task	Q only	Required	Task
GritLM	Generic	Q + D	Required	Task
Hashemi et al.	Per-domain	Q only	None	Domain
Ours	Per-DS	D only	None	DS

Table 1: Comparison with instruction-based retrieval approaches. Q = query side, D = document side, DS = datasource. "Training" indicates whether model training on task/domain data is required.

3 System Overview

Figure 1 shows the two halves of the system: offline prefix discovery and indexing, and online query processing.

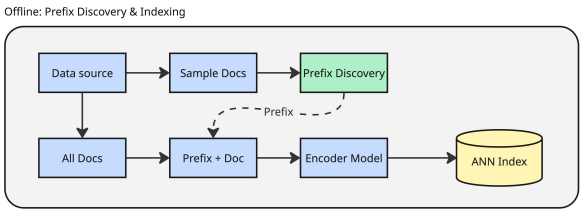

Figure 1: System overview. Offline: a prefix p_s is discovered per datasource from document samples, then prepended to all documents before encoding. Online: queries go through the standard encoding path, unmodified. Dashed arrow shows prefix flow.

3.1 How Prefix Injection Works

Each customer in our deployment maintains multiple datasources. The retrieval model is a shared frozen encoder; the prefix is stored as a metadata field on the datasource configuration and prepended during the indexing pipeline. The approach is encoder-agnostic: any instruction-tuned bi-encoder trained with prefix conventions (e.g., the E5, INSTRUCTOR, or BGE families) can be used. No changes to the ML serving infrastructure were required.

For a datasource s, we first discover a prefix string p_s (Section 4). At indexing time, every document $d \in s$ is encoded as:

$$\mathbf{v}_d = \text{Enc}([p_s \ ; \ d]) \tag{1}$$

where $[;]$ denotes string concatenation and Enc is a frozen instruction-tuned bi-encoder. We view this as *domain-conditioning*: the prefix provides domain context that biases the encoder's internal representations toward a domain-specific subspace, rather than issuing an explicit task instruction.

Queries are *not* prefixed with the datasource prefix—they pass through the encoder with the model's standard query prefix only. This asymmetry is deliberate: a single query frequently searches across multiple datasources simultaneously. Applying a datasource-specific prefix to the query would require either selecting one prefix (losing coverage for other datasources) or encoding the query N times (one per datasource), increasing latency linearly. Document-side prefixing avoids both issues—the prefix is baked into document embeddings at index time, and the query is encoded once regardless of how many datasources are searched.

Because the prefix is a plain text string prepended before encoding, the approach has transferred across two major encoder upgrades in production without requiring prefix redesign, though we have not conducted a systematic cross-model evaluation.

4 Prefix Discovery

We adopt a three-tier prefix discovery approach, reflecting the reality that datasources differ in data access constraints and available expertise.

4.1 Tier 1: LLM-Generated Prefixes

When data policies allow LLM processing, we sample $K{=}5$ documents from the datasource and prompt a language model for a short descriptive phrase (8–15 words) in the pattern "[phrase]: [document text]". We generate $N{=}5$ candidates and select one for deployment. Three selection strategies—synthetic query evaluation, real click-based evaluation, and random selection—yielded similar Hit@5 (Appendix B.4), so the deployed system uses random selection to avoid dependence on query traffic and evaluation circularity. Representative outputs:

- IT incident KB → "IT service management article for resolving technical issues:"
- HR policies → "Employee human resources policy and guidelines document:"
- Product docs → "Enterprise software product description and feature specification:"

LLM-generated prefixes outperform manually written ones, likely because the model captures vocabulary patterns specific to the sampled documents. The choice of LLM has minimal impact: four models (GPT-4o, GPT-4o-mini (OpenAI, 2024), Claude Sonnet, Claude Haiku (Anthropic, 2024)) all produced prefixes within 0.3% Hit@5 of each other (Appendix B.5), so we use a smaller model for cost efficiency. The pipeline requires $N{=}5$ calls per datasource (∼1,700 input tokens each), totaling ∼8,600 tokens per datasource (Appendix B.7).

4.2 Tier 2: Expert-Written Prefixes

When LLM access is prohibited by data residency or contractual restrictions, we request a short datasource description from the customer's search administrator using a template: `[Domain] [content type] document relevant to [topic area]:`

4.3 Tier 3: Standardized Prefix Library

When neither LLM access nor expert input is available, we fall back to a library of 30 standardized prefixes covering common enterprise domains, each in 18 languages. A fastText classifier assigns datasources to the closest prefix (87% top-1 accuracy; misclassifications confuse adjacent domains like IT Security and Security Compliance, yielding suboptimal but not harmful prefixes). As we show in Section 6, even these coarse-grained prefixes yield meaningful gains.

4.4 Prefix Design Observations

Length. Prefixes of 8–15 tokens work best; shorter prefixes provide insufficient signal, while those exceeding 25 tokens consume too much of the encoder's 512-token window. *Punctuation.* Colon-terminated prefixes outperform period or no punctuation by 0.5–1%. *Language matching.* For non-English datasources, prefixes in the document's language outperform English-language prefixes.

5 Experimental Setup

5.1 Production Environment

Our platform serves hundreds of enterprise customers, each operating an independent search instance. A typical customer maintains 5–15 active datasources, though the range is wide: some have a single knowledge base, others operate 50+ specialized collections spanning IT support, HR policies, legal documents, product catalogs, and internal wikis. Datasource sizes are equally heterogeneous, from fewer than 100 documents (e.g., a company's travel policy handbook) to over 500K (e.g., a global IT incident knowledge base accumulated over several years), with a median of approximately 8K

documents.

5.2 Baselines

All experiments use the same frozen instruction-tuned bi-encoder. The only variable is the prefix string prepended to documents at index time.

1. **No Prefix**: documents are encoded with no prefix at all (raw text).
2. **Generic**: documents are encoded with the model's default trained prefix "`passage:`" only.
3. **Domain-Only**: e.g., "IT support knowledge article:" (always in English).
4. **Datasource Prefix**: our tiered approach (Tier 1/2/3 depending on what is available for each datasource).

5.3 Metrics

We evaluate on two axes: offline retrieval quality and online user impact.

Offline evaluation. We sampled 42 customer instances spanning 187 datasources across 9 domain categories and 18 languages. Annotators labeled query–document relevance (binary) for 10K queries total: 5K queries drawn from the five highest-traffic (language, domain) pairs, and 500 queries each from 10 additional pairs selected to cover tail languages and domains. We report MRR@5, MRR@10, Hit@5, and Hit@10.

Online evaluation. A 4-week A/B test (50/50 split at the customer-instance level) across all 400+ active instances spanning 1,200+ datasources. We measure CTR, null result rate, MRR of clicked results, and clicks per session.

We evaluate on production data with real user queries and click signals rather than public benchmarks, as the approach targets deployed multi-tenant settings where datasource characteristics vary substantially.

6 Results

Table 2 shows offline metrics, macro-averaged across language-domain pairs.

Tier 1 (LLM-generated) prefixes yield +5.5% absolute Hit@5 over the unprefixed baseline—entirely from prefix choice, with no model training. Tier 3 (standardized library) and Domain-Only perform comparably (+3.0 and +2.7 Hit@5), indicating that even coarse domain labels capture a meaningful share of the gain. The additional lift from Tier 1 (+2.5% over Tier 3) comes from prefixes tailored to actual document content. Per-datasource variance is substantial: the interquartile range of Tier 1 Hit@5 improvement spans +2.1 to +8.3 percentage points. Notably, no datasource with more than 50 documents showed a statistically significant degradation from prefixing.

Config.	MRR@5	MRR@10	Hit@5	Hit@10
No Prefix	.709	.736	.796	.854
Generic	.721	.748	.810	.866
Domain-Only	.731	.758	.823	.877
DS Pfx (Tier 3)	.733	.760	.826	.879
DS Pfx (Tier 2)	.744	.771	.839	.890
DS Pfx (Tier 1)	**.757**	**.783**	**.851**	**.902**

Table 2: Offline results on production data. DS = datasource-specific. All use the same frozen encoder; only the prefix differs.

Ablations in Appendix B inform several design choices: document-only prefixing outperforms dual (query+document) prefixing by 1.5% Hit@5 (Appendix B.1); the 8–15 token range is optimal for prefix length (Appendix B.2); and content-based document sampling outperforms title-based by 1.6% for LLM-generated prefixes (Appendix B.3). We also evaluated prefix injection with BM25 and hybrid (dense + BM25) retrieval on English data: BM25 sees minimal gain (+0.6% Hit@5) since the prefix tokens are shared across all documents in a datasource and receive low IDF weight, while hybrid retrieval benefits (+2.6%) primarily through its dense component (Appendix B.6).

6.1 Where Prefixes Help Most

Table 3 breaks down results by domain. Domains with distinctive vocabulary see the largest gains.

Domain	No Pfx	Generic	DS Pfx	Δ
IT Support	.839	.851	.892	+5.3
Legal	.779	.792	.838	+5.9
HR	.818	.831	.863	+4.5
Customer Service	.794	.806	.834	+4.0
Finance	.786	.799	.833	+4.7
Security	.803	.815	.844	+4.1
Engineering	.812	.824	.853	+4.1
Product Catalog	.834	.844	.878	+4.4
General KB	.808	.816	.827	+1.9

Table 3: Hit@5 by domain (best available prefix tier). Δ = absolute gain over no prefix.

Legal (+5.9) and IT Support (+5.3) benefit the most, likely because their terminology ("change request", "CMDB", "data processing agreement")

is highly domain-specific and the prefix helps the encoder contextualize it. General knowledge bases gain the least (+1.9), as expected—their content lacks a coherent domain signal for the prefix to reinforce.

6.2 Across Languages

Lang. Group	No Pfx	DS Pfx	Δ
English	.843	.878	+3.5
W. Euro. (DE, FR, ES, PT, IT, NL)	.804	.853	+4.9
E. Euro. (PL, RU, CS)	.781	.831	+5.0
CJK (JA, KO, ZH)	.762	.826	+6.4
Other (TH, VI, ID, AR, HI)	.738	.775	+3.7

Table 4: Hit@5 by language group (averaged across domains).

CJK languages show the largest gain (+6.4), possibly because the embedding space for these languages has wider semantic spread, making the prefix's disambiguation signal proportionally more valuable. The "Other" group gains less (+3.7), partly because several of these languages fall back to Tier 3 standardized prefixes that are less well-calibrated (Section 6.3).

The gap between Tier 1 and the Generic baseline (`passage:`) varies substantially across languages. For English, the improvement is modest (+2.0% Hit@5), likely because the encoder's pretrained representations are already well-calibrated for English enterprise content. For languages with less representation in pretraining data, the domain prefix provides a much stronger signal: Japanese sees +8.0% and Korean +6.2% over the Generic baseline (Appendix A). This pattern suggests that prefix injection is especially valuable in multilingual deployments where the encoder's out-of-the-box performance is weakest.

6.3 Failure Modes

We document several failure modes encountered during development and deployment.

Non-instruction-tuned encoders. We evaluated prefix injection with BERT-base and RoBERTa-large bi-encoders that were not trained with prefix conventions. Both showed no improvement or slight degradation (-0.3 to -0.8% Hit@5). Because these encoders never learned to interpret a leading natural-language instruction as a semantic steering signal, the prefix is treated as ordinary text that dilutes the document representation. This confirms that the approach depends on instruction-tuned encoders (E5, INSTRUCTOR, BGE families) whose pretraining taught them to condition on prefix semantics.

Overly specific LLM prefixes. In about 8% of Tier 1 datasources, the LLM generated a prefix that was too narrow. If the sample happened to contain many VPN troubleshooting articles, the LLM might produce "VPN and network connectivity troubleshooting guide:" even though the datasource covers all of IT. Increasing the sample size to 20 and adding an explicit instruction to "describe the collection broadly, not the specific sampled documents" reduced this to about 3%.

Tiny datasources. For datasources with fewer than 50 documents, prefixes have negligible effect ($<$1% improvement). The search space is small enough that the ANN index returns everything relevant regardless of prefix.

6.4 Latency

Since the prefix is applied at index time only, the query-time latency overhead is negligible.

Phase	Without	With Prefix
Doc encoding (per doc)	8.3 ms	8.9 ms (+7%)
Indexing (per 10K docs)	94 s	101 s (+7%)
Query encoding (P50)	4.6 ms	4.6 ms (+0%)
End-to-end query (P50)	46.2 ms	46.4 ms
End-to-end query (P99)	138.7 ms	139.1 ms

Table 5: Latency impact. Indexing is 7% slower (one-time cost). Query path is unchanged.

The query path is completely unmodified—the prefix is already baked into document embeddings at query time. The 7% indexing overhead is a one-time cost incurred during datasource setup or the weekly index refresh.

7 Production Deployment

We rolled this out over 8 weeks in three phases: shadow mode (weeks 1–2, logging prefixed results without serving them), staged A/B test (weeks 3–6, gradually increasing traffic from 10% to 50%), and full rollout (weeks 7–8).

7.1 A/B Test Results

The 4.2% CTR lift and 5.7% reduction in null result rate were consistent across customer segments and statistically significant. Randomization was at the customer-instance level to avoid within-session inconsistency. Weekly CTR lifts of 4.0%, 4.1%, 4.3%, and 4.4% across weeks 1–4 show no novelty

Metric	Control	With Prefixes
CTR	62.3%	64.9% (+4.2%)
Null Result Rate	8.7%	8.2% (−5.7%)
MRR_{click}	0.548	0.571 (+4.2%)
Clicks per Session	1.41	1.52 (+7.8%)

Table 6: A/B test results (4 weeks, 400+ customer instances). All significant at $p < 0.01$.

effect or decay. The null result rate improvement is particularly noteworthy: these are cases where the search system returns nothing at all, providing no value to the user.

7.2 Downstream RAG Evaluation

We also evaluated how prefix injection affects a downstream question-answering (QA) system built on retrieval-augmented generation (RAG). The QA pipeline retrieves the top-5 documents using our bi-encoder and feeds them as context to an LLM for answer generation. We tested on two datasource types: IT knowledge base (KB) articles (troubleshooting guides, how-to documents) and product catalog items (feature descriptions, specifications).

Datasource	No Pfx	DS Pfx	Δ
Answer correctness (human-judged, %)			
KB Articles	68.4	73.1	+4.7
Product Catalog	61.2	66.8	+5.6
Retrieval Hit@5			
KB Articles	.841	.889	+4.8
Product Catalog	.803	.854	+5.1
Answer attribution rate (%)			
KB Articles	74.3	79.6	+5.3
Product Catalog	69.1	75.4	+6.3

Table 7: RAG evaluation on KB articles and product catalog datasources. DS Pfx uses best available tier. Answer correctness and attribution are human-judged on 500 questions per datasource.

Prefix injection improves answer correctness by 4.7–5.6 percentage points, driven primarily by better retrieval: when more relevant documents enter the LLM's context window, the generated answer is more likely to be correct and grounded. The attribution rate—the fraction of answers whose key claims can be traced to specific retrieved passages, judged by two annotators per question ($\kappa = 0.72$)—also improves, suggesting that prefixed retrieval reduces hallucination by providing more on-topic context. Product catalog items see a slightly larger gain, likely because catalog terminology (product names, feature codes) benefits more from domain-specific disambiguation than the relatively self-contained KB articles.

Limitations

The approach requires instruction-aware encoders; a vanilla BERT bi-encoder not trained with prefix conventions would likely ignore the prepended text entirely. The prefix is static per datasource—it cannot adapt to individual queries or account for drift in datasource content over time, and we do not yet have automated staleness detection to flag when a prefix no longer reflects evolving content. Our evaluation is within a single platform, and we have not tested generalizability across very different retrieval architectures. The Tier 1 strategy depends on LLM access, which is constrained by customer data policies. Finally, while we present an initial RAG evaluation (Section 7.2), a more comprehensive study across diverse RAG configurations and generation models remains future work.

8 Conclusion

We presented index-time prefix injection, a training-free method that improves multi-tenant retrieval by prepending domain-descriptive prefixes to documents during indexing. Leveraging the domain-conditioning capability of instruction-tuned encoders without modifying model parameters, the approach yields 3–8% Hit@5 improvements across 18 languages and a 4.2% CTR lift in production A/B tests, with negligible latency cost.

Future directions include automated monitoring of prefix quality as datasource content evolves, extending the approach to generative retrieval settings, and distilling the prefix effect into lightweight adapter layers for encoders that do not natively support instruction prefixes.

Ethics Statement

Our system processes anonymized search queries and documents within enterprise instances; no personally identifiable information is used. LLM-based prefix generation (Tier 1) processes only small document samples ($\leq$20 documents) and only when the customer's data-handling policy permits. The prefix mechanism does not introduce biases beyond those already present in the pretrained encoder. All annotators for offline evaluation were compensated at or above market rates.

References

Anthropic. 2024. The Claude 3 model family: A new standard for intelligence.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3650–3675, Toronto, Canada.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Helia Hashemi, Hamed Zamani, and W. Bruce Croft. 2023. Dense retrieval adaptation using target domain description. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 95–104, Taipei, Taiwan.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research (TMLR)*.

Vladimir Karpukhin, Barlas Oğuz, 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 the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

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*, pages 39–48.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved techniques for training LLMs as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Hanseok Oh, Hyunji Seo, Minjin Zhong, and 1 others. 2024. InstructIR: A benchmark for instruction following of information retrieval models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

OpenAI. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada.

Zhengyang Tang, Benyou Wang, and Ting Yao. 2022. DPTDR: Deep prompt tuning for dense passage retrieval. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1193–1202.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022a. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360, Seattle, USA.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022b. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024b. Multilingual E5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2024. C-Pack: Packaged resources to advance general Chinese embedding. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Washington, DC, USA.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. 2024. mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412.

A Per-Language Results

Table 8 reports Hit@5 for all 18 languages using Tier 1 prefixes. The benefit of prefix injection varies widely: English sees +3.1%, while Japanese gains +8.9%.

Language	No Pfx	Generic	DS Pfx	Δ
English	.851	.862	.882	+3.1
German	.814	.824	.862	+4.8
French	.808	.818	.856	+4.8
Spanish	.811	.821	.858	+4.7
Portuguese	.799	.809	.847	+4.8
Italian	.803	.813	.849	+4.6
Dutch	.806	.815	.851	+4.5
Polish	.789	.798	.839	+5.0
Russian	.778	.787	.829	+5.1
Czech	.776	.785	.825	+4.9
Japanese	.739	.748	.828	+8.9
Korean	.754	.764	.826	+7.2
Chinese	.793	.803	.841	+4.8
Thai	.718	.726	.772	+5.4
Vietnamese	.741	.749	.783	+4.2
Indonesian	.756	.764	.793	+3.7
Arabic	.732	.740	.769	+3.7
Hindi	.744	.752	.781	+3.7

Table 8: Per-language Hit@5 (Tier 1 prefixes). Δ = absolute gain over No Prefix.

B Ablation Studies

We report ablations for design decisions discussed in the main text: prefix application strategy, prefix length, document sampling, prefix selection strategy, LLM choice, retrieval method (BM25 and hybrid), and LLM cost. Unless otherwise noted, all ablations use Tier 1 prefixes on the same evaluation set as Table 2.

B.1 Prefix Application Strategy

Table 9 compares four strategies for where to apply the prefix.

Strategy	Hit@5	Δ
No Prefix	.796	—
Query-only	.821	+2.5
Document-only (ours)	.851	+5.5
Both (dual)	.836	+4.0

Table 9: Hit@5 by prefix application strategy. Dual prefixing (applying the datasource prefix to both queries and documents) underperforms document-only by 1.5%.

Document-only prefixing outperforms all alternatives. Query-only prefixing provides a modest gain but falls short, likely because it locks the query to one datasource's prefix and cannot benefit from cross-datasource search. Dual prefixing underperforms document-only by 1.5% Hit@5: we hypothesize that prefixing both sides over-constrains the alignment, particularly for short or ambiguous queries where the prefix dominates the query representation.

B.2 Prefix Length

Table 10 shows performance as a function of prefix length in tokens.

Prefix Length	Hit@5	Δ
No Prefix	.796	—
1–4 tokens	.825	+2.9
5–7 tokens	.837	+4.1
8–15 tokens	.851	+5.5
16–25 tokens	.840	+4.4
>25 tokens	.823	+2.7

Table 10: Hit@5 by prefix length (Tier 1 prefixes, bucketed). The 8–15 token range yields the best results.

Very short prefixes (1–4 tokens, e.g., "IT support:") provide some gain but insufficient domain signal. The 8–15 token range is optimal. Beyond 25 tokens, the prefix consumes too much of the

encoder's 512-token input window, particularly for shorter documents, and performance degrades toward baseline levels.

B.3 Document Sampling for Prefix Generation

Table 11 compares different input strategies for Tier 1 LLM-based prefix generation.

LLM Input	Hit@5	Δ
Datasource name only	.829	+3.3
Document titles (K=5)	.835	+3.9
Document content (K=5)	.851	+5.5
Document content (K=10)	.853	+5.7
Document content (K=20)	.854	+5.8

Table 11: Hit@5 by LLM input strategy for Tier 1 prefix generation. Content-based sampling substantially outperforms title-based or name-based approaches. Returns diminish beyond K=5.

Providing actual document content to the LLM yields substantially better prefixes than using only the datasource name (+2.2%) or document titles (+1.6%). Increasing the sample size beyond K=5 provides diminishing returns: K=20 gains only +0.3% over K=5, though larger samples reduce the risk of overly specific prefixes (Section 6.3).

B.4 Prefix Selection Strategy

Table 12 compares three strategies for selecting the final prefix from N=5 LLM-generated candidates, evaluated on English datasources.

Selection Strategy	Hit@5	Δ
No Prefix	.851	—
Random	.882	+3.1
Synthetic queries (LLM-generated)	.884	+3.3
Real queries (click-as-relevance)	.883	+3.2

Table 12: Hit@5 on English datasources by prefix selection strategy (N=5 candidates). The three selection methods perform within 0.2% of each other, indicating that the variance across candidates is small. The deployed system uses random selection.

The differences between selection strategies are not statistically significant ($p > 0.3$, paired bootstrap). The deployed system uses random selection since it requires no query data and avoids the circularity of evaluating with LLM-generated queries drawn from the same document sample.

B.5 LLM Choice for Prefix Generation

Table 13 compares four LLMs used for Tier 1 prefix generation, evaluated on English datasources with the same evaluation set as Table 2.

LLM	Hit@5	Δ
No Prefix	.851	—
Claude Haiku	.880	+2.9
Claude Sonnet	.882	+3.1
GPT-4o-mini	.881	+3.0
GPT-4o	.883	+3.2

Table 13: Hit@5 on English datasources by LLM used for Tier 1 prefix generation. All four models produce prefixes of comparable quality.

All four models perform within 0.3% of each other. Smaller models (Claude Haiku, GPT-4o-mini) match their larger counterparts, suggesting that producing a short domain-descriptive phrase from a handful of documents does not require frontier-scale reasoning. The deployed system uses a smaller model for cost efficiency.

B.6 Retrieval Method: BM25 and Hybrid

Table 14 evaluates prefix injection across three retrieval methods on English datasources. Hybrid retrieval uses reciprocal rank fusion (RRF) of dense bi-encoder and BM25 scores.

Retrieval Method	Prefix	Hit@5	Δ
Dense (bi-encoder)	No Prefix	.851	—
Dense (bi-encoder)	DS Prefix	.882	+3.1
BM25	No Prefix	.762	—
BM25	DS Prefix	.768	+0.6
Hybrid (Dense + BM25)	No Prefix	.872	—
Hybrid (Dense + BM25)	DS Prefix	.898	+2.6

Table 14: Hit@5 by retrieval method on English datasources (Tier 1 prefixes). BM25 gains are minimal because the prefix tokens are identical across all documents in a datasource and thus receive near-zero IDF weight. Hybrid gains are driven primarily by the dense component.

Prefix injection is most effective with dense retrieval, where it directly shifts embedding representations. BM25 is largely unaffected: because the same prefix string is prepended to every document in the datasource, the prefix tokens appear in all documents and receive near-zero IDF scores, contributing negligibly to ranking. Hybrid retrieval benefits (+2.6%) but the gain is attenuated relative to dense-only (+3.1%) since the BM25 component's rankings remain unchanged.

B.7 LLM Cost for Tier 1 Prefix Generation

Table 15 breaks down the LLM cost for Tier 1 prefix generation per datasource and at deployment scale.

Component	Value
Per datasource	
LLM calls	5 (N=5 candidates)
Input tokens per call (prompt + K=5 doc excerpts)	∼1,700
Output tokens per call (1 candidate prefix)	∼20
Total tokens per datasource	∼8,600
Typical customer instance (50 datasources)	
Total LLM calls	∼250
Total tokens	∼430,000

Table 15: LLM token usage for Tier 1 prefix generation. Each datasource requires N=5 independent LLM calls (one per candidate) with K=5 document excerpts (truncated to ∼300 tokens each) as input. A typical customer with 50 datasources requires ∼430K tokens total.

At ∼8,600 tokens per datasource, prefix generation is not a meaningful cost driver—a typical customer instance with 50 datasources requires fewer than 250 LLM calls and ∼430K tokens total. The dominant operational cost is re-encoding and re-indexing documents with the new prefix, which uses the same infrastructure already required for periodic index refreshes.