Towards Budget-Aware Dense Retrieval for Tables: Trade-offs, Alternatives and Future Directions

Inwon Kang
Rensselaer Polytechnic Institute
kangi@rpi.edu

Kavitha Srinivas
IBM Research
kavitha.srinivas@ibm.com

Sola Shirai
IBM Research
solashirai@ibm.com

Nandana Mihindukulasooriya
IBM Research
nandana@ibm.com

Niharika D'Souza
IBM Research
Niharika.DSouza@ibm.com

Horst Samulowitz
IBM Research
samulowitz@us.ibm.com

Oshani Seneviratne
Rensselaer Polytechnic Institute
senevo@rpi.edu

ABSTRACT

Dense retrieval is an effective method to enable table search for downstream tasks, but the process of serializing, embedding, and indexing tables can be costly. This cost of producing tables embeddings is exacerbated in large data lakes, where many tables may never be queried but still require expensive document-side encoding. In this work, we examine the trade-offs between existing retrieval methods, with a focus on reducing the startup cost of building a dense index. Using the TARGET benchmark, we compare sparse and dense retrieval with different cost profiles, and lightweight adapters that map cheap table embeddings into the latent space of a stronger model. We find that existing embedding models do not offer a smooth cost/quality frontier, and a simple adapter model to serve as a first-stage retriever shows promising results. However, we also find that training such an adapter to *generalize* to unseen table distributions is much harder. Our findings suggest that there is no simple way to avoid the startup cost of dense retrieval, but that a calibrated cheap representation can help defer some of the expensive embedding work until query time.

VLDB Workshop Reference Format:
Inwon Kang, Kavitha Srinivas, Sola Shirai, Nandana Mihindukulasooriya, Niharika D'Souza, Horst Samulowitz, and Oshani Seneviratne. Towards Budget-Aware Dense Retrieval for Tables: Trade-offs, Alternatives and Future Directions . VLDB 2026 Workshop: Tabular Data Analysis (TaDA).

1 INTRODUCTION

As large language models (LLMs) are applied to tabular data tasks such as question answering, fact verification, data analysis, and text-to-SQL, it is crucial to have high-quality retrieval of relevant tables to provide useful context. This is especially important for enterprise data lakes, which may contain a large number of candidate tables to consider for retrieval. A table can be represented through its schema, cell values, rows, columns, metadata, sampled views, or summaries, and each choice changes both the retrieval signal and the amount of document-side work required before the system can serve queries.

Recent work has shown that retrieval quality matters for generative tasks over tables. TARGET [6] benchmarks table retrieval for downstream generation tasks such as question answering, fact verification, and text-to-SQL, and finds that sparse retrieval is generally weak while dense retrieval over table content is a strong default. This result clarifies the quality side of the problem. But it also raises a question about the cost of building such a retrieval system – how good is good enough? Typically, larger models with more complex internal representations yield better retrieval quality. However, these larger models also require more time and computational resources. Thus, it is not entirely clear how practitioners can control the cost/quality trade-off when building a dense retrieval system, and whether this trade-off can be managed gracefully.

In this work, we focus on the start-up cost of building a dense retrieval system for tabular data. This cost is a combination of multiple steps. A table must be serialized, passed through an embedding model, and inserted into an approximate nearest neighbor (ANN) index. For a small benchmark corpus, this cost may be trivial. However, for a large data lake with millions of tables, this cost can be prohibitive: every table may need to be embedded before the workload is known, even though only a small fraction may be relevant to early user queries. The cost is amplified by table-specific representation choices, since a system may choose to embed full tables, row subsets, column subsets, metadata, summaries, or multiple sampled views of the same underlying table. In order to understand how to manage this cost, we break down the problem into three research questions. First, can sparse retrieval replace dense retrieval for a meaningful subset of table queries, allowing the system to avoid dense indexing for those cases? Second, do existing embedding models provide a useful cost/quality frontier, so that a user may simply choose a cheaper model when indexing time is limited? Third, can we learn an intermediate representation that uses a cheap embedder to approximate the neighborhood structure of a stronger, more expensive embedder?

We answer these questions using the TARGET benchmark and additional distractor tables sampled from NQTables [4]. Our analysis shows that BM25 rarely retrieves tables that a strong dense model

Proceedings of the VLDB Endowment. ISSN 2150-8097.

misses, which limits its usefulness as a substitute for dense retrieval. We then compare a range of embedding models and find that model choice alone does not provide a graceful trade-off: cheaper models are much faster but substantially weaker, while the stronger models require much larger indexing times. Finally, we train lightweight adapters from MiniLM table embeddings into STELLA's embedding space. These adapters do not replace the expensive model, but they are useful as first-stage retrievers that identify which tables should be embedded by the expensive model next.

Overall, our findings suggest that the central optimization problem is not whether dense table retrieval is necessary, but how to schedule dense document-side work. Sparse methods do not provide enough coverage, and current embedders do not offer enough granularity in the cost/quality trade-off. A calibrated cheap representation, however, can help defer expensive embedding calls until they are likely to matter. This points toward retrieval systems that are dense by default, but budget-aware in how they build and expand their indexes.

2 RELATED WORK

Table retrieval. Table retrieval covers several related but distinct problems. In data management, the query is often a table, and the goal is to find tables that can be joined, unioned, or used for augmentation. Semantic table union search, such as SANTOS, retrieves tables whose schemas, values, and inter-column relationships make them unionable [7]. Table augmentation systems retrieve joinable candidates before merging them into a downstream prediction task [2]. These settings share our data-lake motivation, but their retrieval signal is table-to-table compatibility rather than a natural-language information need.

Natural-language table retrieval is closer to retrieval-augmented QA, fact verification, and text-to-SQL: the query is a question, claim, or task description, and the retriever must identify evidence tables for a downstream model. Open-domain table QA introduced dense retrievers for answer-bearing tables [4], while FeTaQA and TabFact stress generative QA and fact verification over tables [3, 11]. TARGET builds on this line and shows that dense embedders over table and row serializations are generally stronger than sparse retrieval under weak metadata [6]. This result motivates our question: *if dense representations are the strongest default, how much dense document-side work must be paid before the first query can be served?*

Distillation and query generation. Dense retriever distillation transfers behavior from a stronger retriever to a cheaper model or representation. LEAD, for example, studies feature-based distillation and argues that direct feature matching should be guided by retrieval behavior [15]. This is relevant to our projection and adapter experiments, but much of the literature assumes training queries or teacher rankings, and optimizes final quality or online latency rather than cold-start document encoding cost.

Query generation improves retrieval without manually labeled target queries. Document expansion predicts likely queries for each document and appends them to the indexed text [12]; GPL generates synthetic queries and pseudo-labels for unsupervised dense-retriever adaptation [17]; and table-specific work generates questions from partial table views [9]. These methods inform our synthetic-query experiments, but their usefulness depends on domain alignment and negative-label quality. Our target constraint is different: reducing dense document-side work before the workload is known.

3 TASK SETUP

We study the cold-start setting for natural-language table retrieval, where our task is take a natural language query and retrieve relevant tables. Given a collection of tables $\mathcal{T}$ and a set of queries $\boldsymbol{Q}$, the system must build an index over $\mathcal{T}$ before it can perform retrieval. Each table $t \in \mathcal{T}$ is first converted into text using a serialization function $s(t)$, encoded by an embedding model f, and then inserted into an approximate nearest neighbor (ANN) index. At query time, a query q is encoded by the same model and the index returns the most relevant tables, ranked by the cosine similarity between $f(q)$ and $f(s(t))$. In our experiments, we use markdown syntax of tables as the default serialization scheme with a single view per table (sampling up to 100 rows), following the Dense Table Embedding baseline from TARGET.

We evaluate retrieval systems along two axes. The first is retrieval quality, measured by Recall@10 – a query is successful if at least one gold table appears among the top 10 retrieved tables. Following the TARGET benchmark, we compute the average Recall@10 for each benchmark to get the overall score. The second axis is index creation time, which captures the amount of document-side work required before the system can serve queries. This cost can be decomposed as

$$T_{\text{index}} = T_{\text{serialize}} + T_{\text{embed}} + T_{\text{ANN}},$$

where $T_{\text{serialize}}$ is the time to construct table text, T_{embed} is the embedding model forward-pass time over those texts, and T_{ANN} is the time to build or update the ANN index. In this work, we focus primarily on T_{embed}, which can be controlled by picking different model sizes, unlike the near-static cost of $T_{\text{serialize}}$ and T_{ANN}.

The main question we wish to answer is: Is it possible to maintain reasonable Recall@10 while reducing T_{embed}? In other words, can we find a method that allows us to serve queries with high recall without having to pay the full cost of embedding every table before the first query is served? This is especially important in large data lake settings, where the number of tables can be very large and the workload may be unknown at startup.

A sparse method may reduce T_{embed} by avoiding dense encodings, a smaller embedder may reduce the forward-pass cost, and a learned surrogate model may defer calls to a stronger embedder until the system has narrowed the candidate set. The following experiments test these possibilities under the same retrieval benchmark and runtime environment.

4 EXPERIMENTS AND ANALYSIS

To better understand how we can gracefully control the cost/quality trade-off, we break down the problem into three research questions, leading up to our prototype implementation

- RQ1. Can sparse retrieval replace dense retrieval for some table/queries?
- RQ2. How good are existing dense retrievals? Can we budget by simply picking a different embedder?

RQ3. Can we learn to *approximate* the expensive embedder's latent space using a cheaper embedder?

The first question is a sanity check for the necessity of dense retrieval. If sparse retrieval already covers most of the same queries, then the startup cost of dense retrieval may not be justified. The second question is whether we can simply pick a cheaper embedder to reduce startup cost – i.e. if the cost/quality trade-off of existing embedding models have enough granularity/variety to allow the user to balance the startup cost against the final recall, without needing to change the retrieval architecture. The third question is whether we can train a cheap surrogate to approximate the expensive embedding space, which would allow us to use the surrogate for candidate selection before the expensive index is fully built.

We use the TARGET benchmark [6] to explore these research questions. TARGET provides a set of 5 benchmarks for table retrieval, covering question answering, fact verification, and text-to-SQL tasks. Each benchmark has a different set of tables and queries, but they share the same retrieval pipeline: given a query, the retriever must identify the relevant table(s) from a large candidate pool, and then a downstream model uses the retrieved tables to produce an answer or prediction. In order to simulate a larger data lake setting, we also inject additional tables sampled from NQTables [4]. While TARGET used Gittables [5] as its source of distractor tables, we find that the type of tables present in Gittables (mostly related to downstream learning tasks) are distinct enough from the existing benchmark corpus (small tables sampled from Wikipedia) and are easily distinguished. Thus, we opt to instead use NQTables to provide a more challenging setting. To fairly measure index creation time across methods, we run all experiments using single-thread execution on a server with a H100 GPU and 512GB of RAM.

4.1 RQ1. When is sparse embedding enough?

While the results from TARGET have shown that dense embeddings are generally much stronger than sparse retrieval methods [3], they do not completely rule out the possibility that sparse retrieval methods may be effective in certain cases. To investigate the usefulness of sparse retrieval on TARGET, we compare how often a query achieves perfect recall when using a dense or sparse embedding.

Dataset	Dense-only	BM25-only	Both
BIRD	1277	2	151
FeTaQA	1399	8	84
OTT-QA	899	21	1234
Spider	508	9	193
TabFact	6852	154	3680

Table 1: Retrieval comparison for STELLA [20] and BM25. Each column shows the number of queries where only one method retrieved the golden table.

Table 1 shows results comparing retrieval using a dense embedding versus BM25, comparing the number of queries which achieved perfect recall on only the dense embedding, only BM25, or on both embeddings. We find that BM25 rarely retrieves tables that a strong dense model misses, which limits its usefulness as a substitute for dense retrieval. This suggests that sparse retrieval methods like BM25 are not sufficient for serving a significant portion of the queries in the TARGET benchmark, and that dense retrieval methods are necessary to achieve high recall. This also means that the startup cost of dense retrieval cannot be avoided by simply relying on sparse retrieval for some subset of the queries, and that we need to explore other ways to manage the cost/quality trade-off in dense retrieval systems.

Model	FeTaQA	OTT-QA	TabFact
MiniLM (60M) [16]	$0.2551_{/762.24}$	$0.3198_{/655.54}$	$0.3583_{/712.59}$
F2LLM-80M (80M) [22]	$0.2126_{/605.27}$	$0.3089_{/576.88}$	$0.4532_{/639.63}$
E5-Base (100M) [18]	$0.5706_{/796.56}$	$0.6639_{/730.36}$	$0.6601_{/818.35}$
F2LLM-160M (160M) [22]	$0.2941_{/737.50}$	$0.3839_{/719.68}$	$0.4867_{/693.11}$
EmbGemma (300M) [13]	$\mathbf{0.6286}_{/3605.29}$	$\underline{0.7312}_{/3360.58}$	$\mathbf{0.6831}_{/3363.41}$
E5-Large (300M) [18]	$0.5571_{/1230.23}$	$0.5528_{/1206.36}$	$0.6189_{/1705.59}$
STELLA (400M) [20]	$\underline{0.6156}_{/2930.64}$	$\mathbf{0.7651}_{/2391.12}$	$\underline{0.6810}_{/1843.49}$
Qwen3-0.6B (600M) [21]	$0.4398_{/1746.19}$	$0.5149_{/1718.80}$	N/A*
Approx. Adapter	$0.5921_{/360.45}$	$0.7425_{/355.38}$	$0.6763_{/333.03}$

Table 2: Recall@10 and indexing runtime for different embedding models (parameter size shown in parentheses) on the FeTaQA, OTT-QA, and TabFact datasets from TARGET. Approx. Adapter results are shown in the bottom, where runtime excludes the MLP training time.

*** Some inputs exceeded our environment's memory limits.**

4.2 RQ2. Can we budget our embedder?

Having verified the gap between sparse and dense retrieval methods, we next ask whether it is possible to "budget" between the startup cost and the downstream recall performance – i.e., can the cost/quality trade-offs of embedding models allow the user to pick the appropriate model for their use case? Or is it the case that the most models have a similar trade-off curve, meaning the user would not be able to pick some ideal model for their intended budget?

To test whether this hypothesis is true, we compare the recall performance of several embedding models by picking top-ranking embedding models from the MTEB benchmark [10] with less than 1 billion parameters. For consistency's sake, we use the same markdown-style serialization of each table (up to 100 rows) as used in TARGET. Table 2 shows the recall@10 and the runtime for indexing the tables in the same environment with equal resource constraints. The results show that there is a large gap between the cheaper models (MiniLM, F2LLM-80M, E5-Base, F2LLM-160M) and the medium/larger models (EmbGemma, E5-Large, STELLA). This suggests that existing embedding models do not show a graceful cost/quality trade-off, and that simply picking a different model may not be sufficient to achieve the desired balance between startup cost and final recall performance.

It is also worth noting that the embedding models also display an interesting trend with ANN-related operations. We note that some models (STELLA, EmbeddingGemma) consistently show a higher query time. This suggests that something about the geometry of the embeddings produced by these models makes them more expensive to index and search, leaving room for future works to investigate further. We also note that some models have a much higher context

length (Qwen3 has a context limit of 22K tokens, while STELLA only handles 512), meaning they view more of the "row data" than a smaller context length model. Interestingly, we find that larger context length *does not* appear to help recall while it hurts runtime, which also suggests a need for a better understanding of efficient context usage while encoding tables.

4.3 RQ3. Can we learn to approximate the expensive embedding space?

Our results in RQ2 suggest that there is no existing solution that allows the user to adaptively pick a model for their intended budget. Thus, we then ask the following question: Is it possible to learn a *middle ground* solution, where we leverage the speed of a cheap embedder while *approximating* the expensive embedder's latent space? If this is possible, then we can use the cheap surrogate for candidate selection before the expensive index is fully built, which would allow us to reduce the startup cost of dense retrieval without sacrificing recall.

We test this hypothesis by training a projector model to map the cheap embedding space to the expensive embedding space. Specifically, given a cheap embedding e_{cheap} of a table view, we train a multi-layer perceptron (MLP) to predict the corresponding expensive embedding e_{exp}:

$$\hat{e}_{\text{exp}} = \text{Adapter}(e_{\text{cheap}}),$$

where $\hat{e}_{\text{exp}}$ is the predicted expensive embedding. We test two variants of this approach: one where we sample a fraction of tables from each benchmark's corpus to train the projector, and another where we attempt to pre-train a generalizable version of the adapter by training it on a large set of tables gathered from the Web, government and financial sources. Following the results from TARGET and our observations from RQ2, we use STELLA as our target embedder and MiniLM as our cheap embedder.

The per-corpus adapter improves recall over the original cheap embedder, which shows that the cheap table representation still contains useful information that can be aligned with the expensive embedding space. More importantly, we find that the adapted representation is effective as a first-stage retriever. We test this by building a hybrid retrieval system where the cheap embedder retrieves a large K tables as a candidate set, and the expensive embedder re-ranks those candidates by embedding them. The bottom row of Table 2 shows the performance of a per-corpus adapter trained on 30% of the tables from each benchmark's corpus, including the distractor tables. In this setting, we find that the hybrid model can offer comparable recall to the expensive model over the full corpus, while only applying the expensive model to a fraction of the tables before the first query is served. This suggests that the adapter is effective at preserving enough neighborhood structure to decide which tables should be densified next, even if it does not reproduce the expensive embedding well enough to be the final retrieval model.

However, it should be noted that the hybrid system does not remove the cost of the expensive embedder entirely. In fact, we find that the adapter alone does *not* make a good retriever, performing slightly better than MiniLM. While the hybrid system can indeed reduce the startup cost, it does so by deferring some of the expensive embedding calls until query time, rather than eliminating them. This makes the system cheaper to initialize, but it can make individual queries slower because the reranking candidates still need to be embedded by STELLA. On the other hand, we find that inserting newly embedded candidates into the ANN index is not a major bottleneck, showing that the main cost we pay in this hybrid approach is the expensive embedding calls at query time.

Finally, we test a pre-trained variant of this adapter by gathering 1 million tables sampled from the Web [8], goverment [1, 14], financial [19] and general tabular ML tasks [5], following the same training scheme we use for the per-corpus adapter. The motivation is to now train an adapter that works for any corpus without needing to sample/fine-tune on it, which would make this approach more practical for real-world use cases. However, we find that the pre-trained adapter does not transfer well. We tested several variants, including transformer-based adapters and contrastive training with InfoNCE instead of MSE, but the downstream recall remains weak. This suggests that the cheap-to-expensive mapping is sensitive to the benchmark corpus and cannot be learned once from a broad table collection in a way that immediately generalizes. On the other hand, the simple per-corpus MSE adapter works surprisingly well as a first-stage retriever. This indicates that even a lightweight calibration step can make a cheap off-the-shelf embedder more useful, as long as the goal is candidate selection rather than replacing the expensive retriever completely.

5 DISCUSSION

In this work, we explored the dynamics between different table representation schemes for retrieval with a focus on the startup cost of building a dense retrieval index. We first establish that sparse retrieval methods are not sufficient to cover a significant portion of the queries in the TARGET benchmark, and that dense retrieval methods are necessary to achieve high recall. We also found that existing embedding models do not show a graceful cost/quality trade-off, and that simply picking a different model may not be sufficient to achieve the desired balance between startup cost and final recall performance. Finally, we found that it is possible to learn a cheap surrogate to approximate the expensive embedding space, which can be effective for candidate selection before the expensive index is fully built.

Our findings suggest a few directions for future work. First, there is room for a "middle ground" solution in the dense retrieval landscape – even the sub-billion parameter models can start to take hours to embed a large data lake, and a mechanism for leveraging cheaper models can potentially save a great amount of resources at a large scale operation. Second, the interaction between the embedding geometry and the ANN indexing/searching process is an interesting area for further investigation, as it can have a significant impact on the runtime performance of the retrieval system. Finally, our results also show that short-context models are competitive (if not better) to long-context models. This suggests that there is a need for a better understanding of how to efficiently use the context when encoding tables, and whether there are ways to optimize the context usage to improve recall without increasing runtime.

ACKNOWLEDGMENTS

This work was supported by IBM through the IBM-Rensselaer Future of Computing Research Collaboration.

REFERENCES

[1] U.S. General Services Administration. 2026. Data.gov. https://www.data.gov/.

[2] Riccardo Cappuzzo, Aimée Coelho, Felix Lefebvre, Paolo Papotti, and Gaël Varoquaux. 2024. Retrieve, Merge, Predict: Augmenting Tables with Data Lakes. *arXiv preprint arXiv:2402.06282* (2024). https://doi.org/10.48550/arXiv.2402.06282

[3] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *International Conference on Learning Representations*. OpenReview.net, Online.

[4] Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open Domain Question Answering over Tables via Dense Retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 512–519. https://doi.org/10.18653/v1/2021.naacl-main.43

[5] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data* 1, 1 (2023), 30:1–30:17. https://doi.org/10.1145/3588710

[6] Xingyu Ji, Parker Glenn, Aditya G. Parameswaran, and Madelon Hulsebos. 2025. TARGET: Benchmarking Table Retrieval for Generative Tasks. *arXiv preprint arXiv:2505.11545* (2025).

[7] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2022. SANTOS: Relationship-based Semantic Table Union Search. *arXiv preprint arXiv:2209.13589* (2022). https://doi.org/10.48550/arXiv.2209.13589

[8] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A Large Public Corpus of Web Tables Containing Time and Context Metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*. ACM Press, Montréal, Québec, Canada, 75–76. https://doi.org/10.1145/2872518.2889386

[9] Hsing-Ping Liang, Che-Wei Chang, and Yao-Chung Fan. 2025. Improving Table Retrieval with Question Generation from Partial Tables. In *Proceedings of the 4th Table Representation Learning Workshop*. Association for Computational Linguistics, Vienna, Austria, 217–228. https://doi.org/10.18653/v1/2025.trl-1.19

[10] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Dubrovnik, Croatia, 2014–2037. https://doi.org/10.18653/v1/2023.eacl-main.148

[11] Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2022. FeTaQA: Free-form Table Question Answering. *Transactions of the Association for Computational Linguistics* 10 (2022), 35–49. https://doi.org/10.1162/tacl_a_00446

[12] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv preprint arXiv:1904.08375* (2019). https://doi.org/10.48550/arXiv.1904.08375

[13] Henrique* Schechter Vera, Sahil* Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, Weiyi Wang, Zhe Li, Gus Martins, Jinhyuk Lee, Mark Sherwood, Juyeong Ji, Renjie Wu, Jingxiao Zheng, Jyotinder Singh, Abheesht Sharma, Divya Sreepat, Aashi Jain, Adham Elarabawy, AJ Co, Andreas Doumanoglou, Babak Samari, Ben Hora, Brian Potetz, Dahun Kim, Enrique Alfonseca, Fedor Moiseev, Feng Han, Frank Palma Gomez, Gustavo Hernández Ábrego, Hesen Zhang, Hui Hui, Jay Han, Karan Gill, Ke Chen, Koert Chen, Madhuri Shanbhogue, Michael Boratko, Paul Suganthan, Sai Meher Karthik Duddu, Sandeep Mariserla, Setareh Ariafar, Shanfeng Zhang, Shijie Zhang, Simon Baumgartner, Sonam Goenka, Steve Qiu, Tanmaya Dabral, Trevor Walker, Vikram Rao, Waleed Khawaja, Wenlei Zhou, Xiaoqi Ren, Ye Xia, Yichang Chen, Yi-Ting Chen, Zhe Dong, Zhongli Ding, Francesco Visin, Gaël Liu, Jiageng Zhang, Kathleen Kenealy, Michelle Casbon, Ravin Kumar, Thomas Mesnard, Zach Gleicher, Cormac Brick, Olivier Lacombe, Adam Roberts, Yunhsuan Sung, Raphael Hoffmann, Tris Warkentin, Armand Joulin, Tom Duerig, and Mojtaba Seyedhosseini. 2025. EmbeddingGemma: Powerful and Lightweight Text Representations. (2025). https://arxiv.org/abs/2509.20354

[14] Government Digital Service. 2026. Data.gov.uk. https://data.gov.uk/.

[15] Hao Sun, Xiao Liu, Yeyun Gong, Anlei Dong, Jian Jiao, Jingwen Lu, Yan Zhang, Daxin Jiang, Linjun Yang, Rangan Majumder, and Nan Duan. 2024. LEAD: Liberal Feature-based Distillation for Dense Retrieval. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, New York, NY, USA, 1614–1623. https://doi.org/10.1145/3616855.3635774

[16] Sentence Transformers. 2021. sentence-transformers/all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

[17] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. 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*. Association for Computational Linguistics, Seattle, United States, 2345–2360. https://doi.org/10.18653/v1/2022.naacl-main.168

[18] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv preprint arXiv:2212.03533* (2022).

[19] Elias Zavitsanos, Dimitris Mavroeidis, Eirini Spyropoulou, Manos Fergadiotis, and Georgios Paliouras. 2024. ENTRANT: A Large Financial Dataset for Table Understanding. https://doi.org/10.5281/ZENODO.10667088

[20] Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and Stella: distillation of SOTA embedding models. arXiv:2412.19048 [cs.IR] https://arxiv.org/abs/2412.19048

[21] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176* (2025).

[22] Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. 2026. F2LLM-v2: Inclusive, Performant, and Efficient Embeddings for a Multilingual World. arXiv:2603.19223 [cs.CL] https://arxiv.org/abs/2603.19223