

Effect of Text Embedding Scale on GraphRAG Accuracy

Stanford CS224N Custom Project

Jon Valur Bjornsson
Department of Statistics
Stanford University
jonvb@stanford.edu

Abstract

Contributions of individual components in GraphRAG pipelines remain poorly understood. We investigate the impact of text encoder scale on downstream accuracy in the G-Retriever framework [1], evaluating 6 encoders spanning 23M to 8B parameters on the STaRK-Prime biomedical benchmark [2]. Using a modular pipeline that decouples retrieval from GNN training, we run 36 experiments across 3 random seeds. Encoder-only (mean-pooled) models consistently outperform decoder-only (last-token-pooled) models of comparable or larger size, while within-family scaling trends are inconclusive. An ablation isolating the GNN feature encoder from retrieval shows no clear effect of encoder scale on message passing, and the graph token provides only marginal improvement over a no-GNN baseline.

1 Key Information

- Mentor: Caroline Choi
- External Collaborators: N/A
- Sharing project: N/A
- Late Days (if you are using any):

Team member	Late days remaining
Jon Valur Bjornsson	2
Total pooled	2
Team size	1
Late days applied for the report	2

2 Introduction

Graph-based retrieval-augmented generation (GraphRAG) has emerged as a promising approach for grounding LLM responses in structured knowledge, enabling more accurate and interpretable question answering over graph-structured data. Unlike standard RAG over unstructured text, GraphRAG must contend with rich relational structure, variable-length textual attributes, and multi-hop reasoning, all of which place particular demands on the text encoder used to represent nodes and edges. The design space for each component is large, and controlled studies that isolate the effect of a single component remain valuable for understanding what drives end-to-end performance.

In this work, we systematically vary the text encoder in the G-Retriever framework [1] while holding all other components fixed, evaluating 6 encoders from 23M to 8B parameters on the STaRK-Prime biomedical dataset [2]. Using a custom modular pipeline, we isolate the encoder’s effect on both subgraph retrieval and GNN message passing. To our knowledge, no prior work has performed

this kind of controlled encoder scaling study within a GraphRAG framework. Our main finding is that encoder architecture family (encoder-only vs. decoder-only) matters more than parameter count: the three encoder-only models consistently outperform the three decoder-only models, while within-family scaling shows no clear trend.

3 Related Work

The G-Retriever method [1] enables retrieval-augmented generation over textual graphs by combining Prize-Collecting Steiner Tree (PCST) subgraph retrieval with a GNN and an LLM. The original paper evaluates on SceneGraphs, ExplaGraphs, and WebQSP, settings where benchmark answers may overlap with LLM pretraining data.

A collaboration between NVIDIA and Neo4j [3] evaluated a modified G-Retriever pipeline on the STaRK-Prime biomedical knowledge base [2], a more challenging setting with 129K entities and 8.1M relations across heterogeneous biomedical types. Using OpenAI’s text-embedding-ada-002 and Llama-3.1-8B-Instruct [4], they achieved 32% Hit@1. Building on a reproduction of those experiments¹, this work investigates how replacing the text encoder with models of varying scale affects end-to-end accuracy.

4 Approach

We adapt the PyG [5] G-Retriever pipeline into a four-phase workflow (Figure 1) for systematic encoder scaling studies. Our implementation uses PyG’s `GRetriever` model class and PCST retrieval utilities; the pipeline orchestration, parallelized subgraph construction, and experiment configuration were developed for this project and deployed on Modal, enabling full reproducibility of all 36 runs.

The key design choice is decoupling subgraph retrieval from GNN training. Pre-encoding all node and edge texts per encoder allows independent variation of the retrieval encoder and the feature encoder used for GNN message passing, and enables parallelized PCST subgraph construction across CPU-only containers. Following [3], we fully fine-tune the LLM decoder rather than freezing it as in the original G-Retriever formulation. PyG’s default subgraph textualization uses a CSV-style format that repeats the full text of each node and relation in every edge triple, which caused out-of-memory failures when encoded by the LLM on a single GPU. We instead use a compact ID-based format where node and relation texts each appear once in lookup tables and edges reference them by index (e.g., [0] - (1) -> [2]).

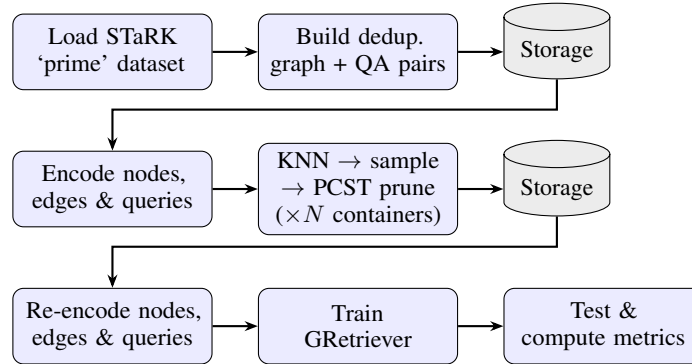


Figure 1: Four-phase pipeline. Phase 1 (top) loads and deduplicates the graph. Phase 2 encodes texts and runs parallelized PCST retrieval. Phase 3 optionally re-encodes with a different feature encoder. Phase 4 trains and evaluates.

Figure 2 illustrates the G-Retriever architecture. Given PCST-pruned subgraphs, a Graph Attention Network (GAT) [6] processes node features to produce graph-level representations, which are projected into the LLM embedding space via an MLP. The projected graph tokens are concatenated with the query embedding and passed to the LLM decoder.

¹<https://github.com/puririshi98/gretriever-stark-prime/>

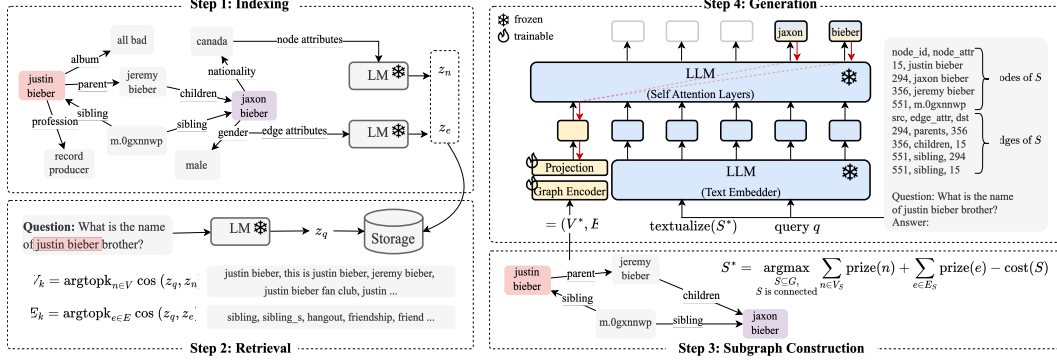


Figure 2: G-Retriever architecture (figure from [1]). In the original formulation, the LLM is frozen and only the GNN and projection MLP are trained. In our experiments, we fully fine-tune all components including the LLM decoder.

The model is trained with cross-entropy loss over the LLM decoder’s generated tokens, where the target is “|”-separated ground-truth entity names:

$$\mathcal{L} = - \sum_{i=1}^T \log p_{\theta, \phi_1, \phi_2}(y_i | y_{<i}, \hat{h}_g, h_t), \quad \hat{h}_g = \text{MLP}_{\phi_2}(\text{POOL}(\text{GNN}_{\phi_1}(S^*))) \quad (1)$$

where S^* is the PCST-pruned subgraph, $h_t = \text{TextEmbed}([x_q])$, and θ, ϕ_1, ϕ_2 parameterize the LLM, GNN, and projection MLP respectively.

During training, passing graph embeddings concatenated with the query into the LLM decoder introduces padding mask instabilities for specific sequence lengths. The doubly-masked attention computes $\text{softmax}(QK^\top / \sqrt{d_k} + M_{\text{causal}} + M_{\text{pad}})V$, where entirely masked rows produce NaN under standard softmax. We resolved this by enforcing the `flash_attention_2` backend in HuggingFace, which handles fully-masked rows gracefully.

5 Experiments

5.1 Data

We use STaRK-Prime [2], a large-scale semi-structured knowledge base for precision medicine. It contains 129,375 entities across 10 types (disease, gene/protein, drug, pathway, etc.), 8,100,498 relations across 18 types, and 31.8M tokens of textual attributes. The QA dataset comprises 11,204 synthesized queries and 109 human-generated queries. The input is a question paired with a retrieved subgraph, and the output is a string of entity names.

5.2 Evaluation method

The primary metric is Hit@k:

$$\text{Hit}@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \mathbf{1}[\hat{Y}_i^{(k)} \cap Y_i \neq \emptyset] \quad (2)$$

where $\hat{Y}_i^{(k)}$ is the set of the first k predicted entities and Y_i is the ground-truth set. We report Hit@1 as the primary metric.

Our reference point is 32% Hit@1, reported by [3] for G-Retriever on STaRK-Prime using OpenAI’s text-embedding-ada-002 and Llama-3.1-8B-Instruct [4]. We note that our experiments use Llama-3.2-1B-Instruct, an $8\times$ smaller decoder, so the comparison is not direct; our focus is on relative differences across encoders rather than matching absolute performance.

5.3 Experimental details

We evaluate 6 text encoders spanning two architectural families: three encoder-only models from the sentence-transformers library [7] (all-MiniLM-L6-v2 [8], all-distilroberta-v1 [9], and all-roberta-large-v1 [10]; 23M–355M parameters, mean-pooled; hereafter MiniLM-L6, DistilRoBERTa, and RoBERTa-L), all contrastively fine-tuned for semantic similarity; and three decoder-only models from the Qwen3-Embedding family [11] (0.6B, 4B, 8B parameters; last-token-pooled; hereafter Qwen3-0.6B, Qwen3-4B, and Qwen3-8B), trained for text embedding and retrieval tasks.

In the **main experiments**, both retrieval and feature encoders use the same model. In the **ablation experiments**, the retrieval encoder is fixed to MiniLM-L6 while only the GNN feature encoder varies, isolating the encoder’s effect on message passing from its effect on retrieval. We also include a **no-GNN baseline** where graph tokens are omitted entirely. Each configuration is run across 3 random seeds (0, 1, 2), totaling 36 runs. All other hyperparameters are held fixed (Table 1).

Because the textualized subgraph must fit in the LLM’s context, larger subgraphs or larger decoders increase the risk of out-of-memory errors. Since different encoders produce different subgraph sizes (Table 3), OOM failures that occur only for certain encoders would confound cross-encoder comparisons. To eliminate this, we both reduced the LLM decoder to 1B parameters (from the 8B model used by [3]) and tuned PCST hyperparameters to produce smaller subgraphs, ensuring no configuration triggers OOM errors. All training runs used a single NVIDIA H200 GPU, with each run completing in approximately 22 minutes. Including encoding and subgraph retrieval, each full configuration took 25–30 minutes.

Table 1: Hyperparameter configuration (fixed across all 36 runs).

Subgraph (PCST)		GNN (GAT)		LLM Decoder	
Top- k nodes	3	Hidden dim	512	Model	Llama-3.2-1B-Instruct
Top- k edges	5	Layers	3	Fine-tuning	Full
Edge cost	0.5	Attention heads	4		
Num. clusters	1	Dropout	0.5		
Seed nodes	4				
Neighbor hops	1				
Fanout	10				

Training	
Optimizer	AdamW [12] ($\beta = (0.9, 0.95)$)
Learning rate	1×10^{-5}
Min learning rate	5×10^{-6}
Warmup epochs	1
Total epochs	3
Batch size	4
Weight decay	0.05
Gradient clipping	0.1
Data split	70 / 20 / 10

5.4 Results

Because absolute Hit@1 varies across random seeds due to initialization, we report Δ Hit@1: for each seed, we subtract the mean Hit@1 across all 6 encoders, isolating encoder effects from seed-level variance. Figures show the median and min–max range of these deviations across 3 seeds. Table 2 reports the absolute median Hit@1 for reference.

6 Analysis

The dominant pattern in Figure 3 is a gap between encoder families: all three encoder-only models score above the per-seed mean, while all three decoder-only models score below it. Although the aggregate plot suggests a negative trend with encoder scale, this is an artifact of the family-level gap. When examining each family separately (Figures 5 and 6), no conclusive within-family scaling trend emerges, with min-max bars overlapping across models.

Table 2: Median Hit@1 (%) per encoder in the main experiments (3 seeds).

Encoder	Hit@1 (%)
MiniLM-L6 (23M)	19.5
DistilRoBERTa (82M)	19.4
RoBERTa-L (355M)	18.8
Qwen3-0.6B (600M)	17.0
Qwen3-4B (4B)	16.6
Qwen3-8B (8B)	16.2

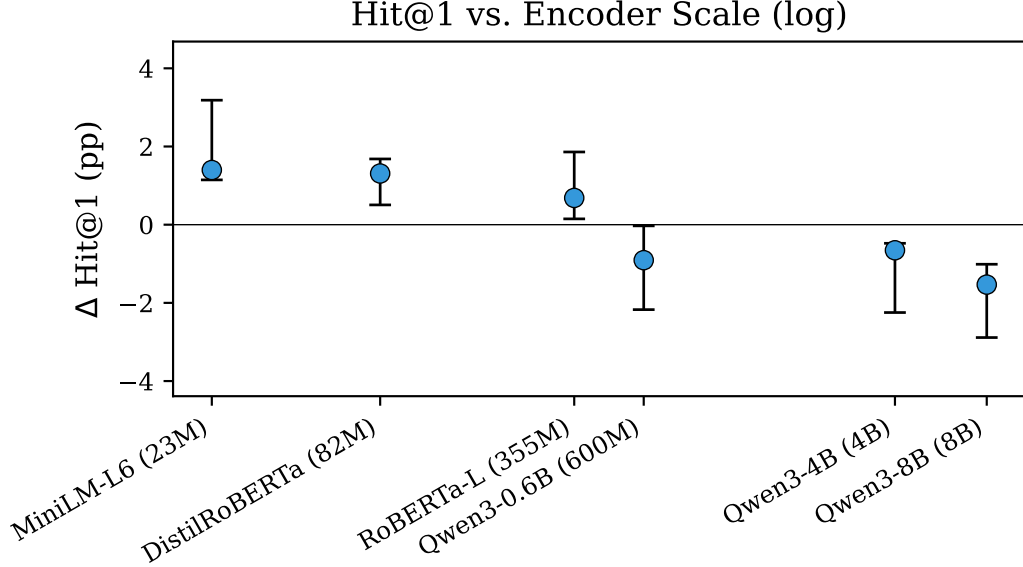


Figure 3: Main experiments: Hit@1 deviation from per-seed mean vs. encoder scale. Dots show median deviation; bars show min-max range across 3 seeds.

This family-level gap likely reflects architectural differences rather than scale. The encoder-only models use mean pooling over all token representations and are trained with contrastive objectives for semantic similarity. The decoder-only Qwen models use last-token pooling, a strategy that may be less suited to the short, attribute-heavy textual descriptions in STaRK-Prime. These encoder-only models are also specifically fine-tuned for sentence similarity, which aligns well with the KNN retrieval step in the pipeline.

Table 3 reveals that encoder choice also affects retrieved subgraph structure: encoder-only models produce smaller, tighter subgraphs (8–9 nodes) compared to decoder-only models (10–12 nodes for Qwen3-0.6B and Qwen3-4B, though Qwen3-8B produces subgraphs comparable in size to the encoder-only models). This suggests that the decoder-only embeddings yield less precise nearest-neighbor retrieval, pulling in more loosely related nodes that may dilute the signal for downstream reasoning.

The ablation experiment (Figure 4), which fixes the retrieval encoder to MiniLM-L6 and varies only the GNN feature encoder, shows no clear trend across encoder scales. This suggests that the encoder’s primary impact on performance is through retrieval quality rather than GNN message-passing representations. The no-GNN baseline (red dashed line at -0.5pp) sits slightly below most GNN-equipped configurations, indicating that the graph token provides a small but not definitive benefit. Given the small margin and variance across seeds, we cannot conclusively confirm that GNN message passing meaningfully contributes beyond what the textualized subgraph alone provides.

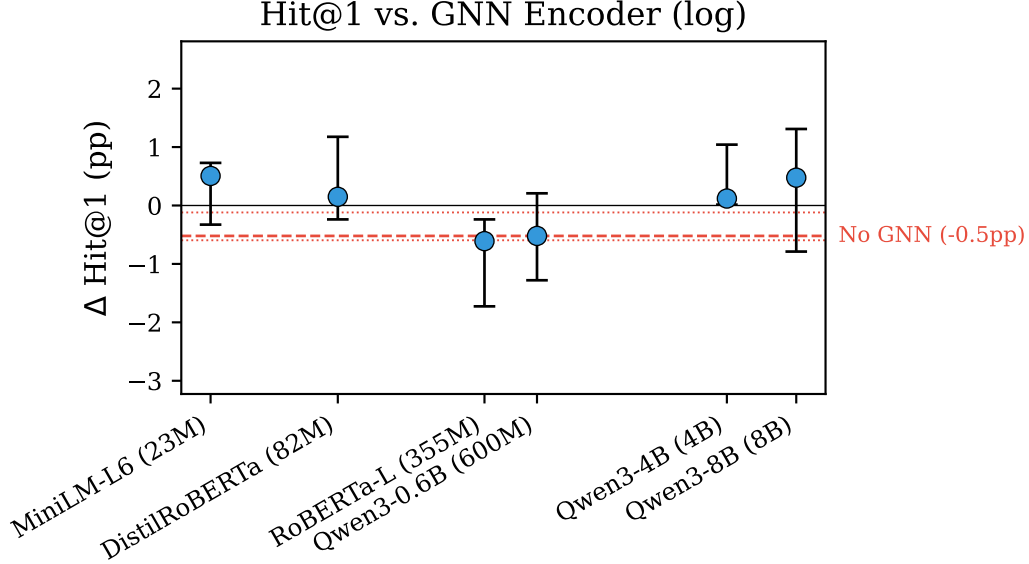


Figure 4: Ablation: Hit@1 vs. GNN feature encoder scale with retrieval fixed to MiniLM-L6. Dashed red line shows the no-GNN baseline (median and min-max).

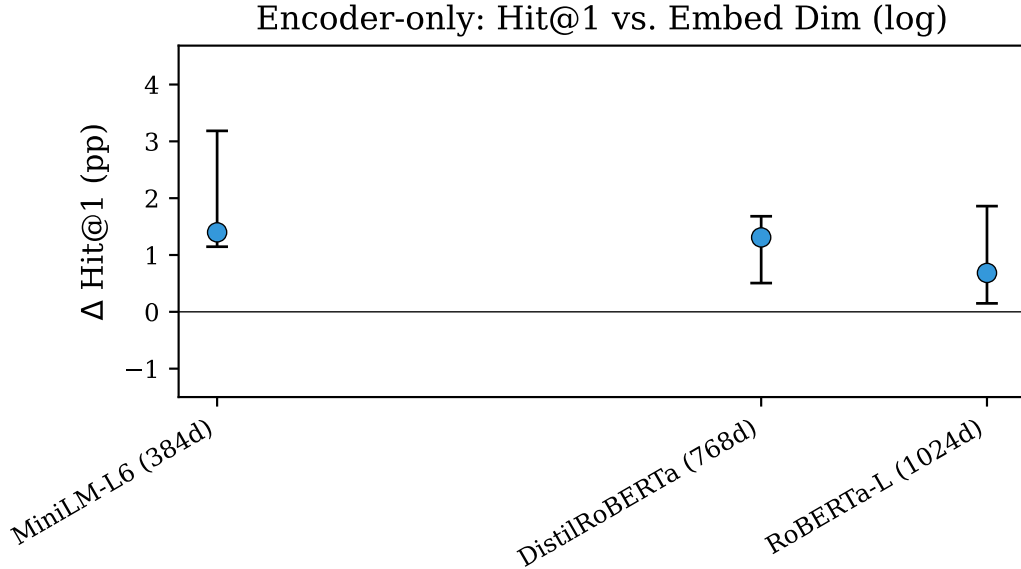


Figure 5: Encoder-only (sentence-transformers) family: Hit@1 vs. embedding dimension. All three models score above the per-seed mean.

7 Conclusion

We presented a systematic study of text encoder scale in the G-Retriever GraphRAG framework, evaluating 6 encoders from 23M to 8B parameters on STaRK-Prime. Our main finding is that the apparent negative scaling trend is driven by a family-level gap: encoder-only (mean-pooled) models consistently outperform decoder-only (last-token-pooled) models, while within-family scaling is inconclusive. The encoder’s impact operates primarily through retrieval quality, with GNN message passing showing no clear dependence on encoder scale.

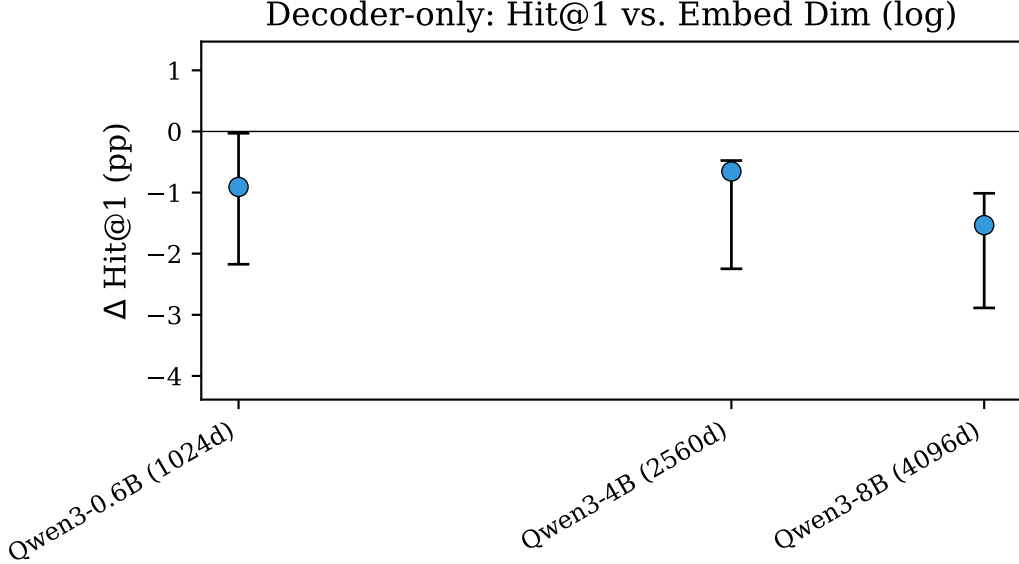


Figure 6: Decoder-only (Qwen) family: Hit@1 vs. embedding dimension. All three models score below the per-seed mean.

Table 3: Median subgraph size per encoder ($\pm$ max deviation, 3 seeds).

Encoder	Nodes	Edges
MiniLM-L6 (23M)	9.4 ± 0.1	16.6 ± 0.3
DistilRoBERTa (82M)	9.2 ± 0.1	16.4 ± 0.2
RoBERTa-L (355M)	8.1 ± 0.3	14.2 ± 0.7
Qwen3-0.6B (600M)	11.8 ± 0.3	21.6 ± 0.6
Qwen3-4B (4B)	11.0 ± 0.6	20.0 ± 1.2
Qwen3-8B (8B)	9.7 ± 0.1	17.4 ± 0.1

Several limitations constrain our conclusions. Memory requirements for encoding textualized subgraphs restricted the LLM decoder to 1B parameters, well below the 8B model used in the reference configuration [3]. PCST hyperparameters were similarly constrained to produce smaller subgraphs, limiting the graph context available to the model. The large hyperparameter space for subgraph retrieval (top- k , edge cost, hops, fanout) remains underexplored and may interact with encoder choice in ways our fixed configuration does not capture. Finally, the marginal contribution of the graph token suggests that either the subgraphs are too small to provide useful structural information, or the GNN and projection layers require further tuning.

Future work could address these limitations by scaling to additional hardware to support larger LLM decoders and richer PCST subgraph configurations, and by using a GNN architecture that incorporates relation embeddings during message passing.

References

- [1] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc., 2024.
- [2] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. In *NeurIPS Datasets and Benchmarks Track*, 2024.

- [3] NVIDIA and PyG Team. Boosting QA accuracy with GraphRAG using PyG and graph databases. <https://developer.nvidia.com/blog/boosting-qa-accuracy-with-graphrag-using-pyg-and-graph-databases/>, 2024. NVIDIA Developer Blog.
- [4] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kam-badur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shao-liang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn,

Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippas Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabza, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaoqian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

- [5] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.
- [6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.
- [7] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.
- [8] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.
- [9] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.
- [10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

- [11] 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. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.
- [12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.