Contextual Tokenization for Graph Inverted Indices

Pritish Chakraborty IIT Bombay
Indradyumna Roy IIT Bombay
Soumen Chakrabarti IIT Bombay
Abir De IIT Bombay

Emails: {pritish, indraroy15, soumen, abir}@cse.iitb.ac.in

Abstract

Retrieving graphs that contain a query subgraph is a core operation in molecular search, program analysis, and scene graph retrieval. Existing methods either rely on single-vector dense embeddings, which are efficient but coarse, or multi-vector neural alignments, which are accurate but require exhaustive corpus scoring. We propose CORGII (Contextual Representation of Graphs for Inverted Indexing), a framework that bridges these extremes by learning discrete, contextual graph tokens that can be indexed with classical inverted indices. Our contributions include (i) a differentiable graph tokenizer that discretizes node embeddings, (ii) a query-aware, trainable impact weighting mechanism, and (iii) co-occurrence based multi-probing for balancing recall and efficiency. Extensive experiments show that CORGII provides better trade-offs between accuracy and efficiency, compared to several baselines. The extended version is given in [1].

1 Introduction

Given a query graph G_q, the retrieval task is to find, from a large corpus of C graphs, those G_c that contain a subgraph isomorphic to G_q [2]. A ranking relaxation returns the top-K corpus graphs that best contain G_q under an approximate subgraph-containment score. Applications include functional group search in molecular databases [3], control-flow pattern detection in program analysis [4], and semantic search in scene graphs [5].

Two challenges arise. Locally, the exact subgraph isomorphism decision problem is NP-complete [6], motivating approximate, often neural, surrogates. Globally, the most accurate approximations rely on *early cross-interaction* between G_q and G_c, which precludes precomputation and forces $\Omega(C)$ query-time scoring. We seek an indexing framework that preserves strong containment signals while avoiding exhaustive scoring.

Single vs. multi-vector representations Neural containment surrogates fall into two families. Early methods use a *single vector* per graph [7–9], enabling ANN-style indices but losing structural fidelity. Later methods represent each graph as a *set of node embeddings* and compute transport/alignment based scores [10–12], improving accuracy at the cost of query-time interaction. This mirrors dense text retrieval's move from bi-encoders [13–15] to multi-vector late interaction (e.g., ColBERT [16]).

Lessons from text retrieval Classical IR uses inverted indices over discrete tokens [17–19] with highly optimized engines [20–22]. With neural encoders, single-vector systems adopted ANN structures [23–35]. Hybrid approaches contextualize words separately and aggregate via indexable surrogates such as a *Chamfer-like* score [36] (e.g., ColBERT [16], PLAID [37]), but incur decompression/bookkeeping overhead. SPLADE [38] improves efficiency by expanding documents before using a standard inverted index. For graphs, however, the lack of a canonical, finite vocabulary and of clear impact weights complicates a direct port of text pipelines.

1.1 Our Contributions

We propose CORGII (Contextual Representation of Graphs for Inverted Indexing), which transfers dense-text retrieval "wisdom" to graph retrieval by (i) learning a contextual *discrete* tokenization of

Chakraborty et al., Contextual Tokenization for Graph Inverted Indices. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 269, Hybrid Event, December 10–12, 2025.

nodes so graphs admit inverted indexing, and (ii) retaining accurate late-interaction signals for final reranking.

Differentiable graph tokenization We introduce a graph tokenizer network (GTNet) that maps each node to a structure-aware token from a latent vocabulary. GTNet first produces *binary-like* node codes that act as discrete tokens, yielding a multi-vector, *sparse* representation compatible with inverted indices. Unlike continuous graph embeddings [7–12], our design uses separate tokenizer heads for queries and corpus (asymmetry helps subgraph containment), while scoring matched nodes via a symmetric distance. Instead of injective relaxations, we rely on a Chamfer-style matching over discrete representations, then index tokens into posting lists of corpus graphs.

Query-aware trainable impact Pure set membership of tokens is weak. Inspired by TF–IDF/BM25, we learn a *token impact* function that conditions on the discrete token and the underlying continuous node embedding, enabling query-aware weighting while remaining index-compatible.

Recall-enhancing multiprobing Independent per-node matches can miss true positives due to discretization noise. We introduce a *co-occurrence multiprobing* strategy: for each query token, we also probe tokens with large posting-list overlap, and aggregate with a threshold, enabling smooth accuracy–efficiency trade-offs prior to final reranking.

Across benchmarks, CORGII yields superior accuracy–efficiency trade-offs and supports tunable latency/quality behavior.

2 Preliminaries

Notation Let $G_q = (V_q, E_q)$ be a query graph and $G_c = (V_c, E_c)$ a corpus graph; the corpus is $\{G_1, \ldots, G_C\}$. A pair (G_q, G_c) has label $y_{qc} \in \{0, 1\}$ with $y_{qc} = 1$ iff $G_q \subseteq G_c$. Denote relevant and non-relevant sets by $\mathcal{C}_{q\oplus} = \{c : y_{qc} = 1\}$ and $\mathcal{C}_{q\ominus} = [C] \setminus \mathcal{C}_{q\oplus}$. After padding to $|V_q| = |V_c| = m$, let $\boldsymbol{A}_q, \boldsymbol{A}_c \in \{0, 1\}^{m\times m}$ be adjacencies; $[\cdot]_+ = \max\{0, \cdot\}$ and $[\![\cdot]\!]$ is the indicator.

Subgraph isomorphism and neural surrogate The exact check $G_q \subseteq G_c$ is equivalent to the existence of a permutation $\boldsymbol{P}$ with $\boldsymbol{A}_q \leq \boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top$, inducing a coverage-loss distance $\min_{\boldsymbol{P}\in\mathcal{P}_m}[\boldsymbol{A}_q - \boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top]_+$. Solving the associated QAP is NP-hard, so neural surrogates [10, 11] compute node embeddings $\boldsymbol{h}_q(u), \boldsymbol{h}_c(v) \in \mathbb{R}^{\dim_h}$ and a soft alignment $\boldsymbol{P}$ (doubly stochastic via Gumbel–Sinkhorn [39, 40]), yielding the hinge-style distance

$$\Delta(G_q, G_c) = \sum_{i\in[\dim_h],\, u\in[m]} \big[\boldsymbol{H}_q - \boldsymbol{P}\boldsymbol{H}_c\big]_+[u, i], \quad \boldsymbol{P} \in [0, 1]^{m\times m}. \tag{1}$$

Pretrained backbone We use IsoNet [10] only for final reranking: it computes the alignment-based distance $\Delta(G_q, G_c)$ (Eq. 1) for each (G_q, G_c) and exposes $\boldsymbol{H}_q, \boldsymbol{H}_c$ and the soft alignment $\boldsymbol{P}$, while our indexing and candidate generation remain independent.

Inverted index over graph tokens Let ω be a token vocabulary. Each graph c is represented by a multiset $\omega(c) = \{\tau^{(1)}, \tau^{(2)}, \ldots\}$ with $\tau^{(\bullet)} \in \omega$. The inverted index maps $\tau \in \omega$ to its posting list $\mathrm{PostingList}(\tau) = \{c : \tau \in \omega(c)\}$. Impact-ordered posting lists (as in TF–IDF) sort by a token's importance to each item. Given a query instance q, we form its token set $\omega(q)$, traverse posting lists for $\tau \in \omega(q)$, aggregate scores, and keep the top candidates for subsequent reranking by $\Delta(G_q, G_c)$.

3 Proposed approach

CORGII is a scalable graph retrieval system that marries accurate late-interaction scoring with the efficiency of classic inverted indices. Starting from the alignment-inspired hinge distance in Eq. (1), we (i) learn *contextual* yet *discrete* node tokens suitable for indexing, (ii) score candidates via a *query-aware impact* aggregator over posting lists, and (iii) use *multiprobing* to balance recall and efficiency before a final reranking by $\Delta(G_q, G_c)$. Figure 1 summarizes the pipeline; detailed descriptions are in the Appendix D.

Contextual node embeddings: A shared GNN GNN_θ contextualizes nodes of G_q and G_c, producing embeddings $\{\boldsymbol{x}_q(u)\}$ and $\{\boldsymbol{x}_c(v)\}$. To preserve the asymmetry of subgraph containment while enabling exact token matches, we attach *separate* MLP heads for query and corpus: $\boldsymbol{z}_q(u) = \sigma(\mathrm{MLP}_{\phi_1}(\boldsymbol{x}_q(u)))$, $\boldsymbol{z}_c(v) = \sigma(\mathrm{MLP}_{\phi_2}(\boldsymbol{x}_c(v)))$, driving both toward *near-binary* codes.

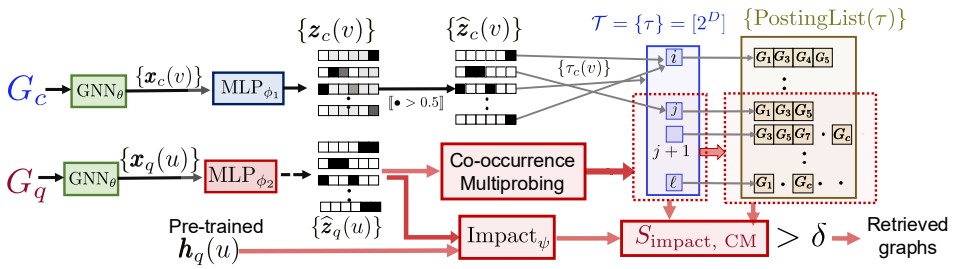


Figure 1: CORGII block diagram. Each (query, corpus) graph pair (G_q, G_c) is encoded using a shared GNN_θ, followed by separate MLPs (MLP_{ϕ_1} and MLP_{ϕ_2}) to compute soft binary node embeddings $\boldsymbol{z}_c(v), \boldsymbol{z}_q(u) \in (0,1)^D$. These are thresholded to obtain discrete binary codes $\widehat{\boldsymbol{z}}_c(v), \widehat{\boldsymbol{z}}_q(u) \in \{0,1\}^D$, mapped to integer-valued latent tokens $\tau \in \mathcal{T} = [2^D]$. Corpus tokens are indexed into posting lists $\text{PostingList}(\tau)$, enabling sparse inverted indexing. During retrieval, query tokens $\tau_q(u)$ are expanded via co-occurrence–based multi-probing (CM) to select similar tokens $\mathcal{N}_b(\tau_q(u))$. Each expanded token τ contributes to the corpus score through an impact score $\text{Impact}_\psi(\tau, \boldsymbol{h}_q(u))$, producing the overall retrieval score $S_{\text{impact,CM}}(G_q, G_c)$. Graphs with score exceeding a threshold δ are shortlisted and reranked using the alignment distance $\Delta(G_q, G_c)$ (Eq. (1)).

Chamfer-driven differentiable tokenization: Rather than an injective soft permutation, we train with a Chamfer distance that independently matches each query node to its best corpus node: $\text{Chamfer}(G_q, G_c) = \sum_{u \in V_q} \min_{v \in V_c} \|\boldsymbol{z}_q(u) - \boldsymbol{z}_c(v)\|_1$. A margin-based pairwise ranking loss encourages relevant pairs to have strictly smaller Chamfer than non-relevant pairs.

Discretization and inverted index: Post-training, we threshold to hard bit-vectors $\widehat{\boldsymbol{z}}_q(u), \widehat{\boldsymbol{z}}_c(v) \in \{0,1\}^D$, inducing a latent token space $\mathcal{T} = [2^D]$. Each graph becomes a *multiset* of tokens $\omega(G) = \{\widehat{\boldsymbol{z}}(\cdot)\}$; the inverted index stores posting lists $\text{PostingList}(\tau) = \{c : \tau \in \omega(G_c)\}$.

Candidate scoring: from uniform counts to impacts: A basic score counts matched tokens, $S_{\text{unif}}(G_q, G_c) = \sum_{u \in V_q} [\![\widehat{\boldsymbol{z}}_q(u) \in \omega(G_c)]\!]$, but ignores structure and frequency selectivity. We introduce a lightweight MLP impact function $\text{Impact}_\psi(\tau, \boldsymbol{h}) = \text{MLP}_\psi(\text{concat}(\tau, \boldsymbol{h}))$, conditioning on the discrete token *and* the continuous node embedding (query-aware). The impact-weighted score is $S_{\text{impact}}(G_q, G_c) = \sum_{u \in V_q} \text{Impact}_\psi(\widehat{\boldsymbol{z}}_q(u), \boldsymbol{h}_q(u))\, [\![\widehat{\boldsymbol{z}}_q(u) \in \omega(G_c)]\!]$, trained with a pairwise margin so relevant graphs outrank non-relevant ones. Impacts are computed on-the-fly at query time and are not stored in the index.

Multiprobing for recall: Exact token matches can be brittle after discretization. We therefore probe additional tokens per query token: (i) *Hamming multiprobing* probes a radius-r Hamming ball in $\{0,1\}^D$; (ii) *Co-occurrence multiprobing (CM)* selects the top-b tokens whose posting lists have the largest overlap with the seed token's posting list, weighting contributions by the overlap.

Query-time pipeline and reranking: For a query G_q: (a) compute $\widehat{\boldsymbol{z}}_q(u)$ and impacts $\text{Impact}_\psi(\cdot)$; (b) probe the index for each token with CM; (c) aggregate per-graph scores and keep the shortlist $\mathcal{R}_q(\delta) = \{c : S_\star(G_q, G_c) \geq \delta\}$; (d) *rerank* candidates via the pretrained alignment distance $\Delta(G_q, G_c)$ (Eq. (1)). The threshold δ directly controls latency/quality.

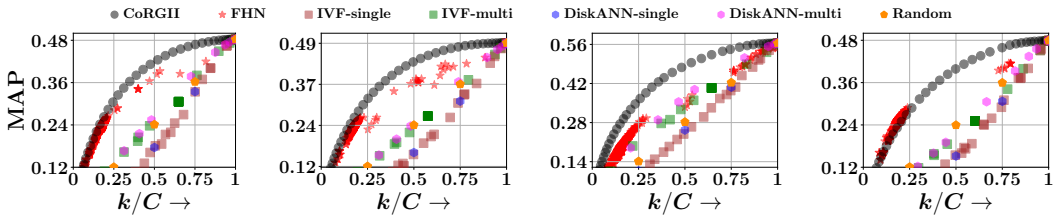
Training recipe: Stage 1: train GTNet $= (\theta, \phi_1, \phi_2)$ with the Chamfer ranking loss to produce discretization-friendly codes. Stage 2: freeze GTNet and train ψ with the impact ranking loss.

4 Experiments

We evaluate CORGII against strong baselines on real-world graph benchmarks and report the main accuracy–efficiency trade-off; extended analyses are deferred to Appendix G.

Datasets: We use four TU datasets [41]: PTC-FR, PTC-FM, COX2, and PTC-MR, commonly used in subgraph matching [10, 11].

Baselines: We compare to (1) **FourierHashNet** (FHN) [42] using single-vector graph embeddings; (2) **IVF-single** (single-vector IVF) [26]; (3) **IVF-multi** (multi-vector IVF akin to ColBERT [16]);



(a) PTC-FR **(b)** PTC-FM **(c)** COX2 **(d)** PTC-MR

Figure 2: Accuracy–efficiency trade-off: MAP vs. fraction of corpus retrieved (k/C) on test queries. CoRGII consistently dominates baselines across datasets.

(4) **DiskANN-single** (single-vector HNSW-based ANN) [30]; (5) **DiskANN-multi** (multi-vector variant); and (6) **Random**.

Protocol: Queries $\mathcal{Q}$ are split 60/20/20 into train/dev/test. For each test query, each method retrieves a candidate set $\mathcal{R}_q$; we *rerank* candidates with the pretrained alignment scorer $\Delta(G_q, G_c)$ (Eq. (1)). We report mean average precision (MAP) versus retrieval budget, measured as the fraction of the corpus retrieved k/C, where $k = \frac{1}{|\mathcal{Q}_{\text{test}}|}\sum_q |\mathcal{R}_q|$. For FHN we sweep hash/table settings; for IVF/DiskANN we vary k directly. Hyperparameters: token bits D=10 ($|\mathcal{T}|$=2^{10}); co-occurrence multiprobing size b=32 (when used).

4.1 Main result: CoRGII vs. baselines

(1) Best trade-offs. Across all datasets, CoRGII achieves the strongest MAP at a given retrieval budget (Fig. 2). On COX2, for example, CoRGII reaches ∼0.50 MAP at k/C=0.5, whereas FHN saturates near 0.35.

(2) Early gains at low budgets. CoRGII attains competitive MAP while retrieving a much smaller fraction of the corpus. On PTC-FR, CoRGII achieves ∼0.36 MAP with k/C<0.33, while most baselines require k/C>0.75 to match.

(3) Multi- vs. single-vector. Multi-vector IVF/DiskANN variants improve upon their single-vector counterparts, confirming the value of node-level granularity; nevertheless they trail CoRGII, reflecting the limitations of symmetric distances for inherently asymmetric subgraph containment.

4.2 Further analyses (Appendix)

We provide detailed studies in Appendix G: (i) co-occurrence vs. Hamming multiprobing and the effect of the expansion budget b; (ii) the impact-weight network vs. uniform counts; (iii) asymmetric vs. Siamese heads; (iv) Chamfer-based training vs. injective alignment; and (v) token co-occurrence insights (posting-list–overlap similarity, Eq. (13), enabling robust multiprobing and smoother recall–latency trade-offs). These support the design choices behind CoRGII and explain the observed gains in Fig. 2.

5 Conclusion

We proposed CoRGII, a scalable graph retrieval framework that bridges highly accurate late interaction query containment scoring with the efficiency of inverted indices. By discretizing node embeddings into structure-aware discrete tokens, and learning contextual impact scores, CoRGII enables fast and accurate retrieval. Experiments show that CoRGII consistently outperforms several baselines. Our work opens up several avenues of future work. It would be interesting to incorporate richly attributed graphs, capturing temporal dynamics in evolving corpora, learning adaptive token vocabularies, and exploring differentiable indexing mechanisms for end-to-end training. Another avenue is to integrate CoRGII into large retrieval-augmented systems that require structured subgraph reasoning at scale.

References

[1] Pritish Chakraborty, Indradyumna Roy, Soumen Chakrabarti, and Abir De. Contextual tokenization for graph inverted indices (corgii). In *Proceedings of the Neural Information Processing Systems (NeurIPS) 2025*, 2025. 1

[2] Bingqing Lyu, Lu Qin, Xuemin Lin, Lijun Chang, and Jeffrey Xu Yu. Scalable supergraph search in large graph databases. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 157–168. IEEE, 2016. 1

[3] Robert P Sheridan and Simon K Kearsley. Why do we need so many chemical similarity search methods? *Drug discovery today*, 7(17):903–911, 2002. 1

[4] Xiang Ling, Lingfei Wu, Saizhuo Wang, Gaoning Pan, Tengfei Ma, Fangli Xu, Alex X Liu, Chunming Wu, and Shouling Ji. Deep graph matching and searching for semantic code retrieval. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5):1–21, 2021. 1

[5] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015. 1

[6] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004. 1

[7] Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020. 1, 2, 9

[8] Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. Greed: A neural framework for learning graph distance functions. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, November 29-Decemer 1, 2022*, 2022. 9, 10

[9] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019. URL https://arxiv.org/abs/1904.12787. 1, 9

[10] Indradyumna Roy, Venkata Sai Velugoti, Soumen Chakrabarti, and Abir De. Interpretable Neural Subgraph Matching for Graph Retrieval. *AAAI*, 2022. 1, 2, 3, 9, 13, 15

[11] Ashwin Ramachandran, Vaibhav Raj, Indradyumna Roy, Soumen Chakrabarti, and Abir De. Iteratively refined early interaction alignment for subgraph matching based graph retrieval. *Advances in Neural Information Processing Systems*, 37:77593–77629, 2024. 2, 3, 9, 10

[12] Wei Zhuo and Guang Tan. Efficient graph similarity computation with alignment regularization. *Advances in Neural Information Processing Systems*, 35:30181–30193, 2022. 1, 2, 9

[13] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 373–382, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336215. doi: 10.1145/2766462.2767738. URL https://doi.org/10.1145/2766462.2767738. 1

[14] Zhe Dong, Jianmo Ni, Dan Bikel, Enrique Alfonseca, Yuan Wang, Chen Qu, and Imed Zitouni. Exploring dual encoder architectures for question answering. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9419, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.640. URL https://aclanthology.org/2022.emnlp-main.640/.

[15] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/. 1

[16] Omar Khattab and Matei Zaharia. 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, 2020. 1, 3, 9

[17] Gerard Salton. Modern information retrieval. *(No Title)*, 1983. 1

[18] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. An Introduction to Information Retrieval. Cambridge University Press, 2008. ISBN 9780521865715. URL https://books.google.co.in/books?id=GNvtngEACAAJ. 11

[19] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 2nd edition, 1999. 1

[20] Otis Gospodnetic, Erik Hatcher, and Michael McCandless. *Lucene in action*. Simon and Schuster, 2010. 1, 9

[21] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.", 2015. 9

[22] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2356–2362, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3463238. URL https://doi.org/10.1145/3404835.3463238. 1

[23] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014. 1, 9

[24] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018. 9

[25] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/1908.10396. 9

[26] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024. 3, 9, 16

[27] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[28] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990. 21

[29] Christos Faloutsos and Douglas W Oard. *A survey of information retrieval and filtering methods*. Citeseer, 1995.

[30] Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subramanya, Andrija Antonijevic, Dax Pryce, David Kaczynski, Shane Williams, Siddarth Gollapudi, Varun Sivashankar, Neel Karia, Aditi Singh, Shikhar Jaiswal, Neelam Mahapatro, Philip Adams, Bryan Tower, and Yash Patel. DiskANN: Graph-structured indices for scalable, fast, fresh and filtered approximate nearest neighbor search, 2023. URL https://github.com/Microsoft/DiskANN. 4, 16

[31] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[32] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801, 2015.

[33] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. *Advances in neural information processing systems*, 28, 2015.

[34] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *arXiv preprint arXiv:1707.00143*, 2017.

[35] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488, 2019. 1

[36] Ainesh Bakshi, Piotr Indyk, Rajesh Jayaram, Sandeep Silwal, and Erik Waingarten. A near-linear time algorithm for the chamfer distance, 2023. URL https://arxiv.org/abs/2307.03043. 1, 10

[37] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. Plaid: an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1747–1756, 2022. 1, 9

[38] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292, 2021. 1, 9

[39] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013. 2, 14

[40] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018. URL https://arxiv.org/pdf/1802.08665.pdf. 2

[41] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020. 3

[42] Indradyumna Roy, Rishi Agarwal, Soumen Chakrabarti, Anirban Dasgupta, and Abir De. Locality sensitive hashing in fourier frequency domain for soft set containment search. *Advances in Neural Information Processing Systems*, 36:56352–56383, 2023. 3, 15

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013. 9

[44] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 9

[45] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019. 9

[46] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010. 9

[47] Number of simple connected graphs on n unlabeled nodes. Online. URL https://oeis.org/A001349. 11

[48] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018. 16

[49] Antonella Falini. A review on the selection criteria for the truncated svd in data science applications. *Journal of Computational Mathematics and Data Science*, 5:100064, 2022. 21

[50] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020. 21

Appendix

A Broader Impact

Graph retrieval is a key enabler in many real-world domains where structured relationships are central. Our work on CORGII contributes to subgraph-based retrieval, offering benefits across a range of applications:

- **Drug discovery and molecular search:** Efficient subgraph containment enables rapid screening of compounds containing functional motifs, aiding in virtual screening pipelines.
- **Program analysis and code intelligence:** Retrieval over control-flow or abstract syntax graphs can improve vulnerability detection and semantic code search.
- **Scene understanding and vision-language systems:** Graph-based representations of scenes or object relationships benefit from scalable matching of structured queries.
- **Scientific knowledge extraction:** Structured retrieval over citation or concept graphs supports discovery in large corpora of scientific knowledge.

By enabling fast and accurate retrieval of graphs under substructure containment, our work has the potential to improve the scalability and responsiveness of systems that rely on structured search over large graph collections. The proposed method, CORGII, contributes toward democratizing structure-aware search by bridging discrete indexing methods with neural representations, making such systems more accessible to low-resource settings where dense model inference may be prohibitive.

While CORGII offers efficiency and interpretability advantages, like any retrieval system, it may raise concerns when applied to sensitive graph-structured data—such as personal social networks or proprietary molecular datasets—potentially risking privacy or intellectual property leakage. Moreover, since training relies on learned embeddings, there remains a possibility of inherited biases from the underlying data. Practitioners are advised to apply appropriate safeguards, including privacy-preserving techniques and fairness auditing, when deploying the system in sensitive domains.

B Limitations

While CORGII demonstrates strong performance in scalable graph retrieval, several aspects offer room for further improvement. We outline them below as avenues for future exploration:

- **Static token vocabulary:** Our framework relies on a fixed latent vocabulary defined by binary token length. Learning adaptive or dynamic vocabularies could improve representation flexibility and efficiency.
- **Lack of support for attributed or heterogeneous graphs:** CORGII currently operates on purely structural information. Extending the framework to incorporate rich node/edge attributes and heterogeneous graph types is a natural next step.
- **Limited handling of evolving corpora:** The inverted index assumes a static corpus. Incorporating update-friendly indexing or continual learning mechanisms would enable deployment in dynamic settings, such as codebases or scientific repositories.

Addressing these limitations can further improve the adaptability, expressivity, and deployment readiness of CORGII in diverse graph-based retrieval settings.

C Related Work

C.1 A Brief History of Text-based Retrieval Architectures

Early information retrieval (IR) systems for text, relied on sparse lexical matching using bag-of-words (BoW) representations. Documents were encoded as high-dimensional sparse vectors over a fixed vocabulary, with inverted indices mapping each term to its corresponding document sets. Statistical term-weighting schemes like TF-IDF and BM25 were used to estimate relevance, prioritizing terms that were both frequent within a document yet discriminative across the corpus. Decades of research culminated in highly optimized sparse retrieval systems, such as Lucene [20] and Elasticsearch [21], which remain industry standards for lexical search.

Despite their efficiency, lexical methods are fundamentally limited by their reliance on surface-level token matching, failing to capture deeper semantic similarity. To address this, dense neural IR models were proposed, using learned embeddings–initially static (e.g., Word2Vec [43], GloVe [44]) and later contextual (e.g., BERT [45])– to encode text into compact, low-dimensional vector spaces that support semantic retrieval. As these dense representations are incompatible with inverted indices, retrieval relies on Approximate Nearest Neighbor (ANN) search techniques such as LSH [23], HNSW [24], and IVF [46], with efficient implementations provided by libraries like FAISS [26] and ScaNN [25].

Early dense retrieval systems typically compressed entire texts into single-vector representations, but this proved suboptimal for longer inputs due to over-compression, which obscures token-level distinctions. This limitation motivated a shift toward multi-vector representations, which preserve token-level information and enable more precise semantic alignment. Architectures such as ColBERT [16] and PLAID [37] adopt late interaction mechanisms that allow scalable token-wise retrieval. However, these methods still rely on ANN search as a subroutine, which—despite its effectiveness in dense settings—is slower than inverted indexing in practice.

To bridge the gap between dense semantic modeling and efficient retrieval, recent work has revisited sparse representations through a neural lens. Sparse neural IR models seek to combine the semantic expressiveness of dense models with the scalability and efficiency of traditional inverted indices. Approaches like SPLADE [38] learn document-specific, term-weighted sparse vectors by projecting inputs onto a high-dimensional vocabulary space. Crucially, this space is not limited to surface-level input tokens; the model can activate latent or implicitly relevant terms through learned expansions, effectively enriching the document representation beyond what is explicitly present in the text. This allows for semantically-informed exact matching within classical IR frameworks, effectively narrowing the gap between dense retrieval and sparse indexing.

C.2 A Briefer History of Neural Graph-Containment Scoring Models

Graph containment—determining whether a query graph G_q is (approximately) embedded within a corpus graph G_c—has long been a central problem in graph-based search. Traditional methods rely on combinatorial algorithms and subgraph isomorphism solvers, which are computationally expensive and scale poorly to large graph corpora. To address this, recent neural methods propose differentiable surrogates for containment using graph neural networks (GNNs).

NeuroMatch [7] introduced a Siamese GNN with a hinge loss over aggregated node embeddings, but its global pooling loses fine structural detail. IsoNet [10] addresses this by retaining node-level embeddings $\boldsymbol{H}_q, \boldsymbol{H}_c$, computing soft alignments via a Gumbel-Sinkhorn network to produce a doubly stochastic matrix $\boldsymbol{P}$, and scoring relevance using an asymmetric hinge loss $[\boldsymbol{H}_q - \boldsymbol{P}\boldsymbol{H}_c]_+$. This better models subgraph containment and achieves improved empirical performance. IsoNet++ [11] extends this with early-interaction message passing for richer local-global representation. However, both IsoNet and IsoNet++ require dense, pairwise alignment across the corpus, limiting scalability.

Other recent works, such as [8, 9, 12], model graph similarity via node-level interactions using matching networks or soft attention. While these approaches capture structural alignment to some extent, they are tailored for general-purpose similarity tasks, making them less suited for subgraph containment and less scalable to large graph corpora.

D Detailed Description of Proposed Approach

We now present CORGII: a scalable retrieval system for graph retrieval that takes advantage of decades of optimization of inverted indices on discrete tokens, and yet supports scoring and ranking using continuous node embeddings. Starting with the hinge distance (1), we propose a series of steps that adapt GNN-based contextual node embeddings toward a discrete token space, enabling us to use inverted indices. Before describing the modules of CORGII, we outline these adaptation steps.

GNN-based node embeddings: As described in Section 2, a (differentiable) GNN contextualizes nodes in their graph neighborhood to output $\boldsymbol{H}_q$ and $\boldsymbol{H}_c$. The (transportation-inspired) hinge distance between them, found effective for ranking in earlier work, is asymmetric and based on a (soft) permutation $\boldsymbol{P}$. These introduce two major hurdles in the way of deploying inverted indices. CORGII approximates the asymmetric, early-interaction distance (1) with an asymmetric dual encoder (late interaction) network, but based on a non-injective granular scoring function.

Efficient differentiable (near-)tokenization: As a first step toward tokenization, we apply two (still differentiable, but distinct) networks to $\boldsymbol{h}_q(u), \boldsymbol{h}_c(v)$, ending with sigmoid activations, which take the outputs $\boldsymbol{z}_q(u), \boldsymbol{z}_c(v)$ closer to bit-vector representation of tokens. The GNN, together with these networks, are trained for *retrieval accuracy* (and not, *e.g.*, any kind of reconstruction). We also replace the permutation with a Chamfer distance [36] which brings us closer to inverted indices.

Token discretization and indexing: Finally, we round $\boldsymbol{z}_q(u), \boldsymbol{z}_c(v)$ to 0/1 bit vectors $\widehat{\boldsymbol{z}}_q(u), \widehat{\boldsymbol{z}}_c(v)$, assigning a bit-vector token to each node. Much like text documents, a graph is now represented as a multiset of discrete tokens. With this step, we lose differentiability, but directly use an inverted index.

Impact weights and multi-probing: All tokens should not contribute equally to match scores. Based on corpus and query workloads, we learn suitable impact weights of these (manufactured) tokens. We further optimize the performance of CORGII by designing a suitable aggregation mechanism to prune the posting lists obtained from all the query tokens. Finally, we consider one folklore and one novel means to explore the 'vicinity' of a query token, to provide a smooth trade-off between query latency and ranking accuracy.

D.1 Graph tokenizer network GTNet

We now proceed to describe the major components of CORGII.

The first stage of GTNet is a standard GNN similar to that described in Section 2, but here we will train it exclusively for indexing and retrieval. The GNN will share the same parameters θ across query and corpus graphs. After the GNN, we will append a multi-layer perceptron (MLP) layer with different parameters ϕ_1 and ϕ_2 for the query and corpus graphs.

$$\boldsymbol{z}_q(u) = \sigma(\text{MLP}_{\phi_1}(\boldsymbol{x}_q(u))) \text{ for } u \in V_q \quad \text{where, } \{\boldsymbol{x}_q(u)\} = \text{GNN}_\theta(G_q); \tag{2}$$

$$\boldsymbol{z}_c(v) = \sigma(\text{MLP}_{\phi_2}(\boldsymbol{x}_c(v))) \text{ for } v \in V_c \quad \text{where, } \{\boldsymbol{x}_c(v)\} = \text{GNN}_\theta(G_c). \tag{3}$$

Rationale behind different MLP networks Unlike exact graph matching, the subgraph matching task is inherently asymmetric, where $G_q \subset G_c$ does not mean $G_c \subset G_q$. To model this asymmetry, existing works [8–11] employs hinge distance $\Delta(G_q, G_c)$ (1), while sharing a a Siamese network with the same parameters for query and corpus pairs. However, such approach will preserve subgraph matching through order embeddings: $\boldsymbol{Z}_q \leq \boldsymbol{S}\boldsymbol{Z}_c$. But inverted indexing requires exact token matching, making order embeddings incompatible with token-based indexing. Therefore, we retain asymmetry through separate MLPs for queries and corpus.

Introducing Chamfer Distance between graphs An asymmetric Siamese network lets us replace hinge distance $[\boldsymbol{H}_q - \boldsymbol{P}\boldsymbol{H}_c]_+$ with the normed distance $\|\boldsymbol{Z}_q - \boldsymbol{P}\boldsymbol{Z}_c\|_1$, but, for the sake of indexing, we need to avoid the permutation $\boldsymbol{P}$ (whose best choice depends on both $\boldsymbol{H}_q$ and $\boldsymbol{H}_c$), so that 'document' graphs can be indexed independent of queries. Moreover, training from relevance labels require $\boldsymbol{P}$ to be modeled as a doubly stochastic soft permutation matrix (see Eq. (1)). However, its continuous nature of $\boldsymbol{P}$ smears the values in $\boldsymbol{Z}_q$ and $\boldsymbol{Z}_c$, leading to poor discretization. Due to these reasons, we avoid the permutation, and for each query node u, match $\boldsymbol{z}_q(u)$ to $\boldsymbol{z}_c(v)$ for some corpus node v, *independently of other query nodes*, as opposed to the joint matching of all nodes in query corpus pairs, thus permitting non-injective mappings via the *Chamfer distance* [36]:

$$\text{Chamfer}(G_q, G_c) = \textstyle\sum_{u \in V_q} \min_{v \in V_c} \|\boldsymbol{z}_q(u) - \boldsymbol{z}_c(v)\|_1, \tag{4}$$

Ideally, relevant graphs yield $\mathrm{Chamfer}(G_q, G_c) = 0$, but GTNet produces approximate binary representations, making exact matches unlikely. To ensure a robust separation between the relevant and non-relevant query-corpus pairs, we seek to impose a margin of separation m: non-relevant graphs should differ by at least one additional bit per node compared to the relevant graphs, corresponding to a total Chamfer distance separation of m.

Formally, for a query graph G_q, and the set of relevant and irrelevant (or less relevant) corpus graphs $c_\oplus \in \mathcal{C}_{q\oplus}$ and $c_\ominus \in \mathcal{C}_{q\ominus}$, we require $\mathrm{Chamfer}(G_q, G_{c_\oplus}) + m < \mathrm{Chamfer}(G_q, G_{c_\ominus})$. This yields the following ranking loss optimized over parameters of GTNet, *i.e.*, θ, ϕ_1, ϕ_2:

$$\min_{\theta,\phi_1,\phi_2} \sum_q \sum_{c_\ominus \in \mathcal{C}_{q\ominus}, c_\oplus \in \mathcal{C}_{q\oplus}} [\mathrm{Chamfer}(G_q, G_{c_\oplus}) - \mathrm{Chamfer}(G_q, G_{c_\ominus}) + m]_+ \tag{5}$$

Note that node embeddings $\boldsymbol{Z}_q, \boldsymbol{Z}_c$ still allow backprop, but are closer to "bit vectors". Moreover, the training of θ, ϕ_1, ϕ_2 is guided not by reconstruction considerations, but purely by retrieval efficacy.

D.2 Discretization and inverted index

Once GTNet is trained, we compute, for each corpus graph and node therein, $\boldsymbol{z}_c(v)$, and, from there, a bit vector $\widehat{\boldsymbol{z}}_c(v) = [\![\boldsymbol{z}_c(v) > 0.5]\!]$ as a 'hard' representation of each corpus node (and similarly from $\boldsymbol{z}_q(u)$ to $\widehat{\boldsymbol{z}}_q(u)$ for query nodes u). Given $\boldsymbol{z}_c(v) \in (0,1)^D$, this means $\widehat{\boldsymbol{z}}_c(v) \in \{0,1\}^D$, *i.e.*, each node gets associated with a D-bit integer. Let us call this the *token space* $\mathcal{T} = [2^D]$. Note that multiple nodes in a graph may get assigned the same token. Thus, each query graph G_q and corpus graph G_c are associated with *multisets* of tokens, denoted

$$\omega(G_q) = \{\!\{\widehat{\boldsymbol{z}}_q(u) : u \in V_q\}\!\} \quad \text{and} \quad \omega(G_c) = \{\!\{\widehat{\boldsymbol{z}}_c(v) : v \in V_c\}\!\}. \tag{6}$$

(If a graph is padded for efficient tensor operations, the tokens corresponding to padded nodes are logically excluded from the multisets. We elide this code detail for clarity.)

Conceptually, a basic inverted index is a map where the keys are tokens. Each token $\tau \in \mathcal{T}$ is mapped to the set (without multiplicity) of corpus graphs (analog of 'documents') in which it appears: $\mathrm{PostingList}(\tau) = \{c \in \mathcal{C} : \tau \in \omega(G_c)\}$. Intuitively, the goal of minimizing the Chamfer distance (4) in the pre-discretized space corresponds, in the post-discretized space, to locating documents that have large token overlap with the query, which finally enables us to plug in an inverted index.

Candidate generation using uniform impact At query time, the query graph G_q is processed as in (6), to obtain $\omega(G_q)$. Given the inverted index, each token $\tau \in \omega(G_q)$ is used to retrieve $\mathrm{PostingList}(\tau)$. As a simple starting point (that we soon improve), a corpus graph can be scored as

$$S_{\text{unif}}(G_q, G_c) = \textstyle\sum_{u \in V_q} [\![\widehat{\boldsymbol{z}}_q(u) \in \omega(G_c)]\!]. \tag{7}$$

(If multiple nodes u have the same token $\widehat{\boldsymbol{z}}_q(u)$, they are counted multiple times. Belongingness in $\omega(G_c)$ is Boolean, without considering multiplicities.) These scores are used to select a subset of candidates from the whole corpus. These qualifying candidates are reranked using the (computationally more expensive) alignment-based distance $\Delta(G_q, G_c)$ (1).

D.3 Impact weight network

The crude unweighted score (7) has some limitations:

(1) Information is lost from $\boldsymbol{H}$ to $\boldsymbol{Z}$ to $\hat{\boldsymbol{Z}}$. Nodes with minor differences in neighborhood structure may be mapped to overlapping tokens, resulting in large candidate pools.

(1) Similar to IDF, we need to discriminate against common graph motifs with poor selectivity.

In our setting, the combinatorial explosion of motifs makes the estimation of motif frequencies intractable [47]. Moreover, unlike IDF in text retrieval [18], frequent structures cannot be down-weighted, as subgraph retrieval requires matching all query components, regardless of the frequency of the subgraphs.

To mitigate the above difficulties, we use a notion of token impact weight in the same spirit as in traditional text retrieval, although there are some crucial differences. We introduce an *impact weight network* $\mathrm{Impact}_\psi : \mathcal{T} \times \mathbb{R}^{\dim_h} \to \mathbb{R}$, parameterized with ψ, where $\dim_h$ is the dimension of the pre-trained continuous node embedding $\boldsymbol{h}_q(\bullet)$ or $\boldsymbol{h}_c(\bullet)$. We often substitute τ with $\widehat{\boldsymbol{z}}_q(u)$ in the input to Impact_ψ depending on context.

(a) 2-stage training of CORGII

input: graph corpus $\mathcal{C}$, training queries $\{G_q\}$ with relevance labels $\{y_{qc}\}$
// Train GTNet
for each query-corpus pair (G_q, G_c) **do**
    Compute $\boldsymbol{Z}_q = \text{GTNet}(G_q)$ (Eq. (2))
    Compute $\boldsymbol{Z}_c = \text{GTNet}(G_c)$ (Eq. (3))
    Compute $\text{Chamfer}(G_q, G_c)$ (Eq. (4))
end for
Train GTNet by minimizing margin-based ranking loss on $\text{Chamfer}(G_q, G_c)$ (Eq. (5))
// Train impact network
for each query-corpus pair (G_q, G_c) **do**
    Compute $\boldsymbol{Z}_q = \text{GTNet}(G_q)$
// Compute binary embeddings
    $\{\widehat{\boldsymbol{z}}_q(u)\} = [\![\boldsymbol{Z}_q > 0.5]\!]$
// compute impact scores of all query graphs
    compute $S_{\text{impact}}(G_q, G_c)$ (Eq. (9))
end for
Train Impact_ψ network by minimizing margin-based ranking loss on $S_{\text{impact}}(G_q, G_c)$ (Eq. (10))

(b) Retrieval and reranking

inputs: query G_q, threshold t, pre-trained embeddings $\{\boldsymbol{h}_q(u)\}$ for G_q.
// Obtain approximate binary representation
Compute $\boldsymbol{Z}_q = \text{GTNet}(G_q)$ (Eq. (2))
// Compute binary embeddings and tokens
$\{\widehat{\boldsymbol{z}}_q(u)\} = [\![\boldsymbol{Z}_q > 0.5]\!]$
$\omega(G_q) = \text{ObtainTokenSet}(\{\widehat{\boldsymbol{z}}_q(u)\})$
// Compute impact weights
for each node $u \in V_q$ **do**
    Compute $\text{Impact}_\psi(\widehat{\boldsymbol{z}}_q(u), \boldsymbol{h}_q(u))$ (Eq. (8))
end for
// Probe index using query node tokens and their impacts (with optional token expansion) and aggregate preliminary relevance scores
$S_{\text{impact},CM}(G_q, G_c)$ (Eq. (9))
// Shortlist candidates
$\mathcal{R}_q(\delta) = \{G_c : S_{\text{impact},CM}(G_q, G_c) \geq \delta\}$ (11)
rerank surviving candidates using $\Delta(G_q, G_c)$ (1)
return top-k corpus graphs

Figure 3: (a) preprocessing and (b) query-time components of CORGII.

Neural architecture of Impact_ψ Network Impact_ψ is implemented as a lightweight multi layer perceptron (MLP). Given input token τ, presented as a binary code from $\{0,1\}^D$, and input node embedding $\boldsymbol{h}$, we concatenate them and pass the result through a multi-layer perceptron, *i.e.*,

$$\text{Impact}_\psi(\tau, \boldsymbol{h}) = \text{MLP}_\psi\left(\text{concat}\left(\tau, \boldsymbol{h}\right)\right). \tag{8}$$

Rather than count all matched tokens uniformly (7), we compute an impact-weighted aggregate:

$$S_{\text{impact}}(G_q, G_c) = \sum_{u \in V_q} \text{Impact}_\psi(\widehat{\boldsymbol{z}}_q(u), \boldsymbol{h}_q(u))\, [\![\widehat{\boldsymbol{z}}_q(u) \in \omega(G_c)]\!] \tag{9}$$

Thus, token matches are weighted according to their learned structural importance rather than treated uniformly, enabling fine-grained, query-sensitive retrieval over the inverted index.

Training Impact_ψ Let $\mathcal{C}_{q\oplus}$ and $\mathcal{C}_{q\ominus}$ be the relevant and non-relevant graphs for query G_q. Similar to Eq. (5), we encourage that $\S_{\text{Impact}}(G_q, G_{c_\oplus}) > S(G_q, G_{c_\ominus}) + \gamma$ for $c_\oplus \in \mathcal{C}_{q\oplus}$ and $c_\ominus \in \mathcal{C}_{q\ominus}$, where $\gamma > 0$ is a margin hyperparameter. As before, this leads to a pairwise hinge loss to train ψ:

$$\underset{\psi}{\text{argmin}} \sum_{q} \sum_{c_\oplus \in \mathcal{C}_{q\oplus}} \sum_{c_\ominus \in \mathcal{C}_{q\ominus}} \left[S_{\text{impact}}(G_q, G_{c_\ominus}) - S_{\text{impact}}(G_q, G_{c_\oplus}) + \gamma\right]_+. \tag{10}$$

Note that the networks described earlier, with parameters θ, ϕ_1, ϕ_2 are frozen before training the impact parameters ψ. Unlike in classical inverted indices, impact weights are not associated with documents, or stored in the index. Figure 3(a) shows all the training steps of CORGII.

D.4 Query processing steps and multi-probing

At retrieval time, the query graph G_q is first embedded using the pretrained encoder $\mathcal{E}$ to obtain node embeddings $\boldsymbol{H}_q$. The graph tokenizer GTNet then discretizes $\boldsymbol{H}_q$ into soft binary codes $\boldsymbol{Z}_q$ and later hard binary codes $\hat{\boldsymbol{Z}}$, and Impact_ψ computes impact weights (if used).

Each query graph token is used to probe the inverted index. Candidate corpus graphs are retrieved by aggregating impact scores across matching tokens. Graphs with cumulative relevance scores above a tunable threshold δ form the shortlist:

$$\mathcal{R}_q(\delta) = \{c \in \mathcal{C} : S_\spadesuit(G_q, G_c) \geq \delta\}, \tag{11}$$

where $\spadesuit \in \{\text{unif}, \text{impact}\}$. Here, δ controls the trade-off between the query time and retrieval accuracy. High δ results in smaller size of $\mathcal{R}_q(\delta)$, yielding low query time, whereas a low δ gives high query time. Note that, $S_\spadesuit(G_q, G_c)$ is used only to obtain $\mathcal{R}_q$. Candidates in $\mathcal{R}_q$ are further reranked using the pretrained alignment-based 'true' distance $\Delta(G_q, G_c)$ (1). Details are in Figure 3(b).

Limitation of single probe per query node We have described how candidate corpus graphs are scored using uniform and impact-weighted aggregates. In both methods, each token $\widehat{\boldsymbol{z}}_q(u)$ from the query resulted in exactly one probe into the index. Preliminary experiments suggested that a single probe using each query token leads to lost recall, brought on partly by losing signal from continuous to bit-like node representations, and by replacing permutation-based node alignment with Chamfer score. We must discover and exploit affinities between tokens while accessing posting lists.

In the rest of this section, we explore two means to this end. The first, Hamming expansion, has already been used in the literature on locality-sensitive hashing. The second, co-occurrence expansion, is a proposal novel to CoRGII.

Term weighting	Single Probe	Hamming multiprobe (HM)	Co-occurrence multiprobe (CM)
Uniform	S_{unif}	$S_{\text{unif,HM}}$ $(r = \dots)$	$S_{\text{unif,CM}}$ $(b = \dots)$
Impact	S_{impact}	$S_{\text{impact,HM}}$ $(r = \dots)$	$S_{\text{impact,CM}}$ $(b = \dots)$ (CoRGII)

Table 1: Possible combinations of term weighting and probing strategies. Default CoRGII corresponds to $S_{\text{impact,CM}}$. r and b indicate Hamming radius for HM and number of tokens chosen for CM.

Hamming expansion multiprobe (HM) While exact token matches may be adequate when query and corpus graphs are locally near-isomorphic, discretization errors and structural noise can cause relevant corpus graphs to be missed if no exact token match is found. To improve recall, we "smooth the boundaries of token bit encodings" by introducing a lightweight *token expansion* mechanism: given a query token $\tau \in \mathcal{T}$, we probe the inverted index using not only τ, but also nearby tokens within a Hamming ball of radius r in the binary space $\{0,1\}^D$. Given $\widehat{\boldsymbol{z}}$ and $\widehat{\boldsymbol{z}}$ are the corresponding binary vectors of τ, τ' respectively, we write $B_r(\tau) = \{\tau' : \|\boldsymbol{z} - \boldsymbol{z}'\|_1 \leq r\}$. S_{impact} from (9) is extended, by summing over the ball, to

$$S_{\text{impact,HM}}(G_q, G_c) = \sum_{u \in V_q} \sum_{\tau \in B_r(\widehat{\boldsymbol{z}}_q(u))} \text{Impact}_\psi(\tau, \boldsymbol{h}_q(u)) \, [\![\tau \in \omega(G_c)]\!]. \tag{12}$$

This expansion allows retrieval of corpus graphs containing *approximate matches*, mitigating the brittleness of hard discretization without requiring dense alignment. Hamming expansion has the potential to improve recall, but there is a risk of too many false positive candidates to eliminate through expensive scoring later.

Co-occurrence expansion multiprobe (CM) In classical text indexing, a token is sometimes characterized by the set of documents that mention it. Two tokens can then be compared by comparing their respective posting lists. A large overlap in these posting lists hints that the tokens have high affinity to each other. Adapting this idea to graph indexing can provide an alternative to Hamming-based affinity, which can be used either by itself, or in conjunction with Hamming-based token expansion.

For each query token $\tau \in \mathcal{T}$, we identify additional tokens τ' whose posting lists overlap significantly with the posting list of τ, *i.e.*, $\text{PostingList}(\tau)$. Specifically, we define a similarity score between tokens τ and τ' as

$$\text{sim}(\tau, \tau') = \frac{|\text{PostingList}(\tau) \cap \text{PostingList}(\tau')|}{\sum_{\tau_\star \in \mathcal{T}} |\text{PostingList}(\tau) \cap \text{PostingList}(\tau_\star)|} \tag{13}$$

and expanded token set $\mathcal{N}_b(\tau) = \text{argmax}^{(b)}_{\tau' \in \mathcal{T}} \text{sim}(\tau, \tau')$, where b is the number of similar tokens. Similar to Impact_ψ, we overload the input notation for sim where necessary. S_{impact} from (9) is then updated to aggregate over this expanded neighborhood, weighted by similarity:

$$S_{\text{impact,CM}}(G_c, G_q) = \sum_{u \in V_q} \sum_{\tau \in \mathcal{N}_b(\widehat{\boldsymbol{z}}_q(u))} \text{sim}(\tau, \widehat{\boldsymbol{z}}_q(u)) \, \text{Impact}_\psi(\tau, \boldsymbol{h}_q(u)) \, [\![\tau \in \omega(G_c)]\!]. \tag{14}$$

This way, a corpus graph G_c can receive a non-zero score for a query node u, if any token τ in the expanded set $\mathcal{N}_b(\boldsymbol{z}_q(u))$ appears in $\omega(G_c)$ — not just $\boldsymbol{z}_q(u)$ itself. Table 1 lists different variants including CoRGII.

E Additional details about our model and training

E.1 Pre-trained backbone

We use Isonet [10] for final scoring mechanism. IsoNet has two components: (1) a GNN and (2) a permutation network. GNN comprises of feature initialization network F_1; a message computation

network F_2 and an embedding update (or combination) network F_3. Specifically, for the query graph G_q, we execute L mesage passing layers as follows:

$$\boldsymbol{h}_{q,0}(u) = F_1(\text{Feature}(u)) \quad \text{for all } u \in V_q \tag{15}$$

$$\boldsymbol{h}_{q,k+1}(u) = F_3\left(\boldsymbol{h}_{q,k}(u); \sum_{v:(u,v)\in E} F_2(\boldsymbol{h}_{q,k}(u), \boldsymbol{h}_{q,k}(v))\right), \quad \text{for all } u \in V_q, k \in \{0,..,L-1\} \tag{16}$$

We use the same procedure to compute the embeddings $\boldsymbol{h}_{c,k}$ for corpus graphs. We collect these embeddings in $\boldsymbol{H}_q, \boldsymbol{H}_c \in \mathbb{R}^{m\times \dim_h}$. These embeddings are finally used fed into multilayer perceptron, followed by dot product, to obtain an affinity matrix $\text{MLP}(\boldsymbol{H}_q)\,\text{MLP}(\boldsymbol{H}_c)^\top$ which is then provided as input into a node alignment network to obtain $\boldsymbol{P}$. Given a temperature hyperparameter temp, this network outputs a soft-permutation matrix using Sinkhorn iterations [39].

$$\boldsymbol{P} = \text{Sinkhorn}(\text{MLP}(\boldsymbol{H}_q)\,\text{MLP}(\boldsymbol{H}_c)^\top/\text{temp}) \tag{17}$$

Gumbel-Sinkhorn network consists of iterative row-column normalization as follows:

$$\boldsymbol{P}_0 = \exp(\text{MLP}(\boldsymbol{H}_q)\,\text{MLP}(\boldsymbol{H}_c)^\top/\text{temp}) \tag{18}$$

$$\boldsymbol{P}_{t+1} = \text{RowNormalize}\left(\text{ColumnNormalize}(\boldsymbol{P}_t)\right) \quad 0 \le t \le T-1. \tag{19}$$

As $T \to \infty$ $\boldsymbol{P}_T$ approaches as doubly stochastic matrix and as temp $\to 0, T \to \infty$, the matrix $\boldsymbol{P}_T$ approaches a permutation matrix.

In our work, we set $\dim_h = 10$. Here, F_1 is 10-dimensional encoder; F_2 consists of a combination of a propagator layer with a hidden dimension of 20 and a GRU layer at the output, with final dimension 10; and F_3 consists of an aggregator layer with hidden dimension 20 and output dimension 10. The MLPs used in Sinkhorn network, are linear-ReLU-linear networks. Each MLP has a hidden layer of 25 dimensions, and the output is of 25 dimensions. Finally, we minimize the ranking loss to obtain the parameters of F_1, F_2 and F_3 (Eq. (15)– (16)); and MLP used in Eq. (19)

$$\sum_{q} \sum_{c_\oplus \in \mathcal{C}_{q\oplus}} \sum_{c_\ominus \in \mathcal{C}_{q\ominus}} \left[\Delta(G_q, G_{c_\oplus}) - \Delta(G_q, G_{c_\ominus}) + \text{Margin}\right]_+. \tag{20}$$

We used a margin of 0.5. Note that Δ is *only* used in the final stage of ranking. In Sinkhorn network, we set the number of iterations $T = 10$ and temparature 0.1.

E.2 Details about CORGII

Architecture of GTNet **and** Impact_ψ The GNN in GTNet consists of same architecture as in Eqs. (15)– (16), with the same number of layers and hidden dimensions. Here, we set $\dim(\boldsymbol{x}_\bullet) = 10$. Each of the MLPs in GTNet, *i.e.*, MLP_{ϕ_1}, MLP_{ϕ_2} in Eqs. (2) and (3) consist of a linear-ReLU-linear network with input dimension 10, hidden layer of size 64 and output dimension 10. Note that GTNet does not share any components with the pre-trained backbone.

MLP_ψ used in Eq. (8) to model the impact scorer admits a similar architecture as $\text{MLP}_{\phi_1}, \text{MLP}_{\phi_2}$. It consists of a linear-ReLU-linear network with input dimension 10, hidden layer of size 64 and output dimension 10.

Optimization and Early Stopping. We train both models using the Adam optimizer with a learning rate of 1×10^{-3} and a batch size of 3000. During GTNet training, early stopping is performed at the sub-epoch level (i.e., across batches) with a patience of 30 steps and validation every 30 steps. For Impact_ψ, early stopping is applied at the epoch level with a maximum of 20,000 epochs and patience set to 50. Validation is conducted every epoch, with a default tolerance threshold of 5×10^{-3}. In both cases, the model is evaluated using the score function aligned with its training objective.

Margin Hyperparameter Tuning. For the Chamfer-based ranking loss in Eq. (5), we experiment with margin values of $\{0.01, 0.1, 1.0, 10, 30\}$. The best-performing margins are 10 for PTC-FR and PTC-FM, and 30 for COX2 and PTC-MR. For the impact network loss in Eq. (10), tested margins include $\{0.01, 0.1, 1.0\}$. Margins of 0.01, 0.01, 1.0, and 0.1 work best for PTC-FR, PTC-FM, COX2, and PTC-MR, respectively.

Training Under Co-Occurrence Expansion. During training with co-occurrence multiprobing, the token neighborhood $\mathcal{N}_b(\widehat{z}_q(u))$ in Eq. (14) is replaced with the full vocabulary $\mathcal{T}$. This allows Impact_ψ to learn a relevance-aware importance score for every token. At retrieval time, top-b tokens are selected based on sim, using the learned impact scores.

Reproducibility. All experiments are run with a fixed random seed of 42 across libraries and frameworks. We leverage PyTorch's deterministic execution setting and CuBLAS workspace configuration to ensure reproducible execution.

F Additional details about experiments

Dataset	$\min(\lvert V_C \rvert)$	$\max(\lvert V_C \rvert)$	$\mathbb{E}[\lvert V_C \rvert]$	$\min(\lvert E_C \rvert)$	$\max(\lvert E_C \rvert)$	$\mathbb{E}[\lvert E_C \rvert]$
PTC-FR	16	25	18.68	15	28	20.16
PTC-FM	16	25	18.70	15	28	20.13
COX2	16	25	19.65	15	26	20.23
PTC-MR	16	25	18.71	15	28	20.17

(a) Corpus graph statistics.

Dataset	$\min(\lvert V_Q \rvert)$	$\max(\lvert V_Q \rvert)$	$\mathbb{E}[\lvert V_Q \rvert]$	$\min(\lvert E_Q \rvert)$	$\max(\lvert E_Q \rvert)$	$\mathbb{E}[\lvert E_Q \rvert]$	$\mathbb{E}[\frac{\lvert y=1 \rvert}{\lvert y=0 \rvert}]$
PTC-FR	6	15	12.64	6	15	12.41	0.11
PTC-FM	7	15	12.58	7	15	12.34	0.12
COX2	6	15	13.21	6	16	12.81	0.11
PTC-MR	6	15	12.65	7	15	12.41	0.12

(b) Query graph statistics and average positive-to-negative label ratio ($\mathbb{E}[\frac{\lvert y=1 \rvert}{\lvert y=0 \rvert}]$).

Table 2: Statistics of sampled subgraph datasets used in our experiments. Each dataset consists of 500 query graphs and 100,000 corpus graphs.

F.1 Datasets

All experiments are performed on the following datasets: PTC-FR, PTC-FM, COX2 and PTC-MR. From each dataset, we extract corpus and query graphs using the sampling procedure outlined in [10], such that $|\mathcal{C}| = 100000$ and $|\mathcal{Q}| = 500$. The queryset is split such that $|\mathcal{Q}_{\text{train}}| = 300$, $|\mathcal{Q}_{\text{dev}}| = 100$ and $|\mathcal{Q}_{\text{test}}| = 100$. Each dataset has its relevant statistics outlined in Table 2, including the minimum, maximum and average number of nodes and edges in both the corpus set and the queryset. Additionally, the table also lists the per-query average ratio of positive ground truth relationships to negative ground truth relationships.

F.2 Baselines

We provide a detailed description of each of the baselines used in our experiments.

FourierHashNet: It is a Locality-sensitive Hashing (LSH) mechanism designed specifically for the set containment problem [42], applied to subgraph matching. In particular, it overcomes the weaknesses of symmetric relevance measures. Earlier work employed measures such as Jaccard similarity, cosine similarity and the dot product to compute similarity between a pair of items, which do not reflect the asymmetric nature of the problem. FHN, on the other hand, employs a hinge-distance guided dominance similarity measure, which is further processed using a Fourier transform into the frequency domain. The idea is to enable compatibility with existing fast LSH techniques by leveraging inner products in the frequency domain, while retaining asymmetric notion of relevance.

We adopt the original architecture and training settings of FourierHashNet [42] without modification. The model employs 10 sampled Fourier frequencies to compute learned asymmetric embeddings, which are then optimized using a binary cross-entropy loss over embedding vectors of dimension 10. The final hash representation consists of 64-bit codes. For training, we perform a full grid sweep over the hyperparameter configurations proposed in the original work, including all specified loss weights. To study the trade-off between retrieval accuracy and index efficiency, we vary the number of hash table buckets at query time, ranging from 2^1 to 2^{60}.

IVF: The FAISS library provides facilities for inverted file indexing (IVF) [26]. IVF clusters the corpus of vectors using a suitable quantization method. The quantization method produces centroid vectors for each corpus vector, and each centroid represents a cluster. Internally, the library stores the vectors assigned to each cluster in the form of (possibly compressed) contiguous postings lists. To search this construction, a query vector is transformed into its corresponding centroid to match with the given cluster. Depending on the number of probes argument given during search time, one may expand their search into multiple neighboring clusters. We implement single-vector and multi-vector variants of IVF.

Note that we use the `faiss.IndexFlatIP` as the quantizer and `faiss.IndexIVFFlat` as the indexer.

DiskANN: To tackle the challenge of having to store search indices in memory for strong recall, DiskANN introduces efficient SSD-resident indices for billion-scale datasets [30]. To this end, the authors develop a graph construction algorithm inspired by methods such as HNSW [48], but which produce more compact graphs (smaller diameter). They construct smaller individual indices using this algorithm on overlapping portions of the dataset, and then merge them into a single all-encompassing index. These disk-resident indices can then be searched using standard techniques. One of the key benefits of DiskANN is that it requires modest hardware for the construction and probing of their disk-resident indices. We implement single-vector and multi-vector variants. Note that we test against the memory-resident version of DiskANN.

We employ a graph degree of 16, complexity level of 32, alpha parameter of 1.2 during indexing. During search, we use an initial search complexity of 2^{21}.

F.3 Evaluation metric

We report Mean Average Precision (MAP) and the average number of retrieved candidates to characterize the efficiency–accuracy tradeoff. Retrieved candidates are reranked using the pretrained alignment model for consistent evaluation. For a query q with relevant corpus set $\mathcal{C}_{q\oplus}$ and a retrieved ranking π_q, we define the average precision (AP) as

$$\frac{1}{|\mathcal{C}_{q\oplus}|}\sum_{r=1}^{|\pi_q|}\text{Prec@}r\cdot\text{rel}_q(r) \tag{21}$$

where $\text{rel}_q(r)\in\{0,1\}$ indicates whether the r-th ranked item is relevant to q and Prec@r is the precision at rank r. MAP is the mean of AP over all queries. This formulation penalizes high precision with low recall, ensuring models are rewarded only when most number of relevant items are retrieved with high retrieval accuracy.

F.4 System configuration

All experiments were conducted on an in-house NAS server equipped with seven 48GB RTX A6000 GPUs respectively. All model training is done on GPU memory. Further, the server is equipped with 96-core CPU and a maximum storage of 20TB, and runs Debian v6.1. We found that this hardware was sufficient to train CoRGII.

F.5 Licenses

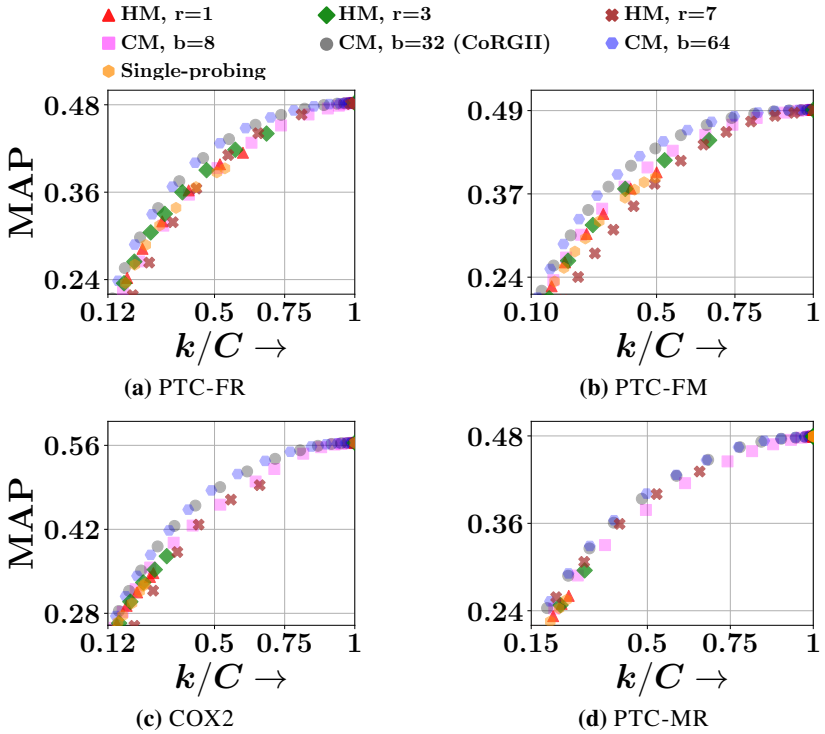
Our code will be released under the MIT license. DiskANN, FAISS and FourierHashNet are all released under the MIT license.

G Additional experiments

We present additional experimental results covering the comparison between co-occurrence-based multiprobing (CM) and Hamming multiprobing (HM), the ablation study on the impact scorer Impact_ψ, and the effect of using a Siamese versus asymmetric architecture in GTNet. We also include supporting analyses on posting list statistics—such as token frequency distributions and posting list co-occurrence patterns—as well as extended results for various CoRGII variants.

G.1 CM vs HM

Benefits of co-occurrence based multi-probing Here, we analyze the effect of our co-occurence based multiprobing (CM) strategy (14), by comparing it with a traditional Hamming distance-based multiprobing variant (HM) (12).



(a) PTC-FR **(b)** PTC-FM **(c)** COX2 **(d)** PTC-MR

Figure 4: Comparison of Hamming expansion multiprobing (HM) against co-occurrence expansion multiprobing (CM) across four real-world datasets and across several values of r (Hamming ball radius) and b (number of topmost co-occurring tokens). Each plot consists of tradeoffs between selectivity (k/C) and MAP, for different values of r and b. $b = 32$ is sufficient for CM to outperform HM variants. To deal with the crowding and overlapping problem in the plots, we have applied point sub-sampling on CoRGII.

Figure 4 shows the results for HM and CM, with different values r and b, and single probing strategy. We make the following observations.

(1) Single probing fails to span the full accuracy-efficiency trade-off curve across all datasets. The retrieved set is noticeably sparse at $k/C \geq 0.50$. These results highlight the necessity of multiprobing to achieve sufficient candidate coverage across varying levels of retrieval selectivity.

(2) CM consistently achieves better trade-off than the corresponding variant of HM and single-probing strategy, while smoothly spanning the full range of retrieval selectivity. As b increases, its performance improves consistently but with diminishing gains, saturating beyond $b = 32$. This indicates that a moderate number of co-occurrence-based token expansions suffices to approach near-exhaustive token expansion performance.

(3) HM retrieves a broader range of candidates and spans the selectivity axis more effectively than the base impact score. However, as the Hamming radius r increases, the expansion becomes increasingly data-agnostic, ignoring semantic alignment from Impact_ψ. This leads to degraded MAP at large r.

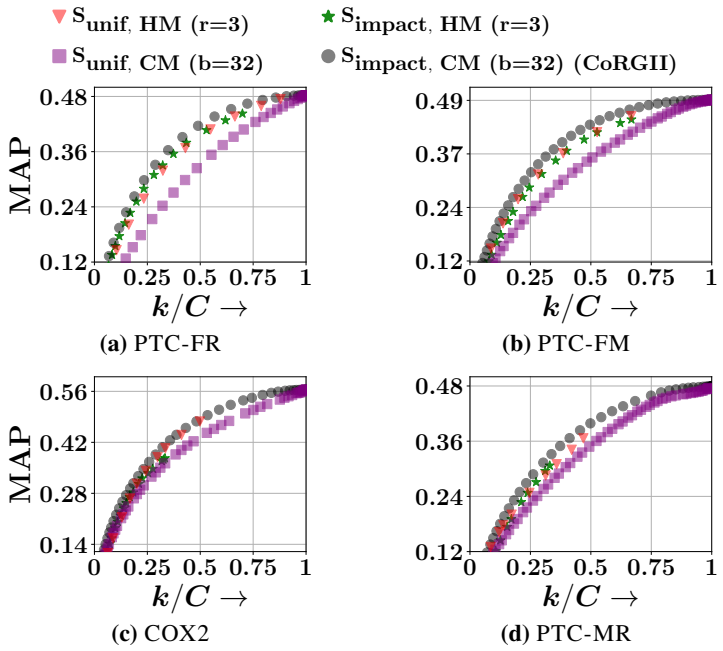
(4) On COX2, HM fails to sweep the entire selectivity axis; even with $r = 7$, it only reaches up to $k/C = 0.6$.

(5) CM steadily improves with increasing b, approaching exhaustive coverage, and saturates beyond $b = 32$.

(6) On PTC-MR, while the difference between HM ($r = 7$) and CM ($b = 32$) is less pronounced, HM still does not cover the full selectivity range.

G.2 Impact weight network ablation

Next, we analyze the effect of impact weight network, by comparing with the variants of our model for both co-occurrence based multiprobing (CM, Eq. (14)) and Hamming distance based multiprobing (HM, Eq. (12)). This results in four models whose scores are $S_{\text{unif,HM}}$, $S_{\text{Impact,HM}}$, $S_{\text{unif,CM}}$ and $S_{\text{Impact,CM}}$ (CoRGII).



(a) PTC-FR **(b)** PTC-FM

(c) COX2 **(d)** PTC-MR

Figure 5: Effect of ablation on the impact weight network across four datasets. Each subplot compares four retrieval variants: uniform aggregation with Hamming multiprobing ($S_{\text{unif, HM}}$), uniform aggregation with co-occurrence multiprobing ($S_{\text{unif, CM}}$), impact-weighted Hamming multiprobing ($S_{\text{impact, HM}}$), and impact-weighted co-occurrence multiprobing ($S_{\text{impact, CM}}$, the default CoRGII).

Figure 5 summarizes the results. We observe that addition of impact weighting network improves the quality of trade-off. The performance gain achieved by impact weighting network is significant, for co-occurrence based multiprobing. We make the following observations:

(1) CoRGII continues to outperform all other variants across the full retrieval budget spectrum on both datasets.

(2) On COX2, uniform aggregation with Hamming multiprobing (HM) briefly approaches CoRGII at low k/C values, but quickly falls behind as selectivity increases.

(3) Removing impact weights causes a significant drop in CM performance across both datasets, underscoring the value of learned token-level importance.

(4) Uniform aggregation under CM fails to deliver competitive trade-offs, confirming that context-aware impact scoring is essential for effective retrieval.

G.3 Siamese vs Asymmetric networks

As discussed in Section D.1, GTNet employs an asymmetric network architecture, which enables exact token matching while capturing the inherent asymmetry of subgraph matching. Here, we investigate its benefits by comparing against a Siamese variant of GTNet, which shares the same MLP across query and corpus graphs.

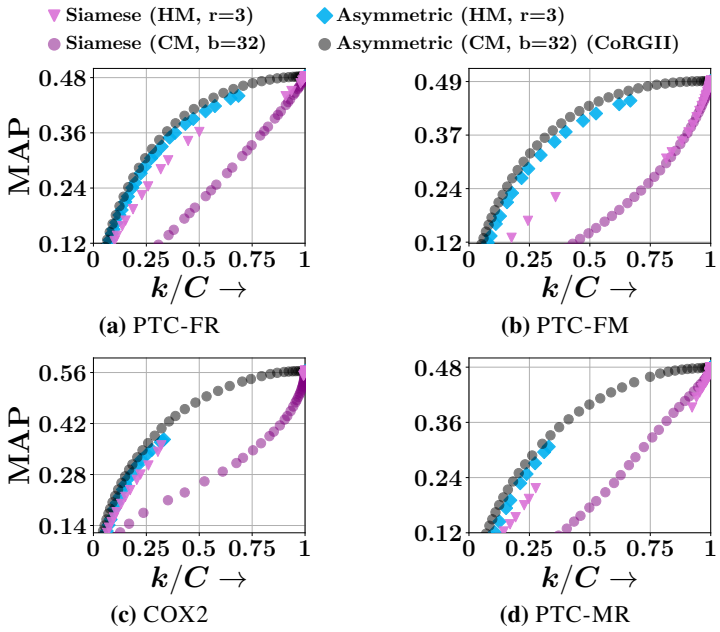


Figure 6: Ablation comparing Siamese and asymmetric network architectures under different probing strategies. Each variant combines one of two network architectures—**Siamese** (shared MLP for query and corpus) and **Asymmetric** (separate MLPs)—with one of two probing strategies: **HM** (Hamming multiprobing with radius $r = 3$) or **CM** (Co-occurrence multiprobing with $b = 32$). CoRGII corresponds to the **Asymmetric + CM** configuration, shown as black circles.

Figure 6 summarizes the results for both co-occurrence based multiprobing (Eq. (14)) and Hamming based multiprobing (Eq. (12)) We highlight the following key observations:

(1) The asymmetric variant of GTNet consistently outperforms the Siamese variant for both HM and CM. The performance boost is strikingly high for CM.

(2) When using the asymmetric network, CM gives notable improvements over HM. However, for the Siamese variant, CM performs poorly on both PTC-FR and PTC-FM, while HM also suffers significantly on PTC-FM. This contrast highlights the importance of architectural asymmetry, especially for effective co-occurrence-based token matching.

(3) CoRGII consistently outperforms both Siamese variants (with CM and HM probing), reaffirming the importance of architectural asymmetry for subgraph containment.

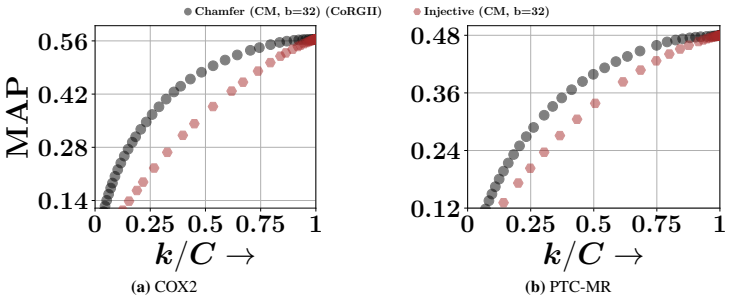
(4) Among the HM variants, the asymmetric network achieves a better tradeoff curve compared to the Siamese counterpart, particularly evident in the mid-selectivity range.

(5) Despite this, HM-based variants—both asymmetric and Siamese—fail to span the full selectivity axis, highlighting the limitations of Hamming multiprobing for recall.

(6) These results further validate the need for asymmetry in the encoder architecture to accurately reflect containment semantics under both probing schemes.

G.4 Chamfer distance vs injective mapping

Chamfer distance provides a non-injective mapping. Here, we compare its performance against traditional graph matching distance with injective mapping, *i.e.*, $\|\boldsymbol{Z}_q - \boldsymbol{P}\boldsymbol{Z}_c\|_1$, where $\boldsymbol{P}$ is a



(a) COX2 **(b)** PTC-MR

Figure 7: Injective vs. Chamfer

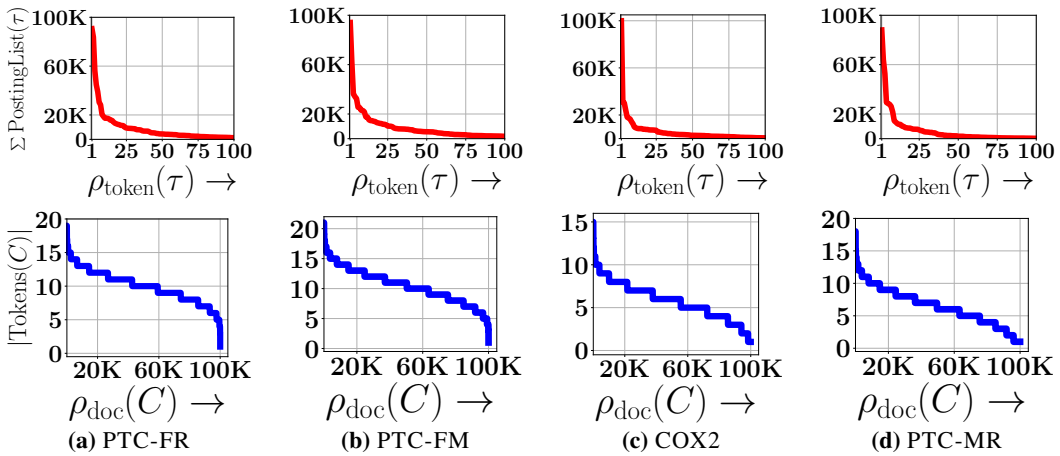
soft permutation (doubly stochastic) matrix. Figure 7 shows that Chamfer distance outperforms injective alignment-based graph matching. This is because injective mappings tightly couple corpus embeddings $\boldsymbol{Z}_c$ with query embeddings $\boldsymbol{Z}_q$, preventing effective inverted indexing.

G.5 Insights into token co-occurrence

Drawing parallels from natural language and retrieval systems, the structure of posting lists and corresponding token co-occurrence statistics are of key interest. In this section, we examine how these properties vary across datasets.

Token rank vs Document frequency We rank each token in the vocabulary by the length of its posting list, $\sum \text{PostingList}(\tau)$, in descending order. Similarly, we rank each document by the number of unique tokens it contains, i.e., $|\text{Tokens}(C)|$.

In the top row of Figure 8, we plot the posting list lengths of tokens by rank. A small number of high-frequency tokens are associated with nearly all documents in the whole corpus, while the vast majority of tokens have short posting lists. The distribution exhibits a steep drop-off with rank, reminiscent of a Zipfian pattern. Inverted indexes are expected to be efficient in precisely these settings. In the second row, we plot the 'fill' of documents against documents ranked by their fills. A similar decay trend is observed, showing that most documents have a small number of tokens with non-zero impacts.



(a) PTC-FR **(b)** PTC-FM **(c)** COX2 **(d)** PTC-MR

Figure 8: Top row: Posting list length vs descending token rank. Bottom row: Number of unique tokens vs descending document rank. $\rho_{\text{token}}(\tau)$ represents the rank of the token τ when sorted in descending order of posting list lengths. $\rho_{\text{doc}}(C)$ is the rank of the document C when sorted in descending order of unique token count.

Co-occurrence statistics Table 3 reports structural statistics of the posting list matrix across all datasets. Let the posting list matrix be $\mathbf{PL} \in \{0,1\}^{1024\times 100K}$, where each row represents

a token and each column a document. The corresponding co-occurrence matrix is defined as $\mathbf{C} = \mathbf{PL} \cdot \mathbf{PL}^\intercal \in \mathbb{Z}_+^{1024\times1024}$.

We list both the *actual rank* and the *effective rank* of $\mathbf{PL}$, the latter computed using the energy-preserving criterion from truncated singular value decomposition (SVD) [49, 50]. Let $\sigma_1, \ldots, \sigma_n$ denote the singular values of $\mathbf{PL}$. The effective rank is the smallest K such that $\frac{\sum_{i=1}^{K}\sigma_i^2}{\sum_{i=1}^{n}\sigma_i^2} > \gamma$, with $\gamma = 0.95$. Since $\text{rank}(\mathbf{PL}) = \text{rank}(\mathbf{C})$ but $\text{rank}^{\text{eff}}(\mathbf{PL}) \geq \text{rank}^{\text{eff}}(\mathbf{C})$, the effective rank of $\mathbf{PL}$ serves as an informative upper bound for that of $\mathbf{C}$.

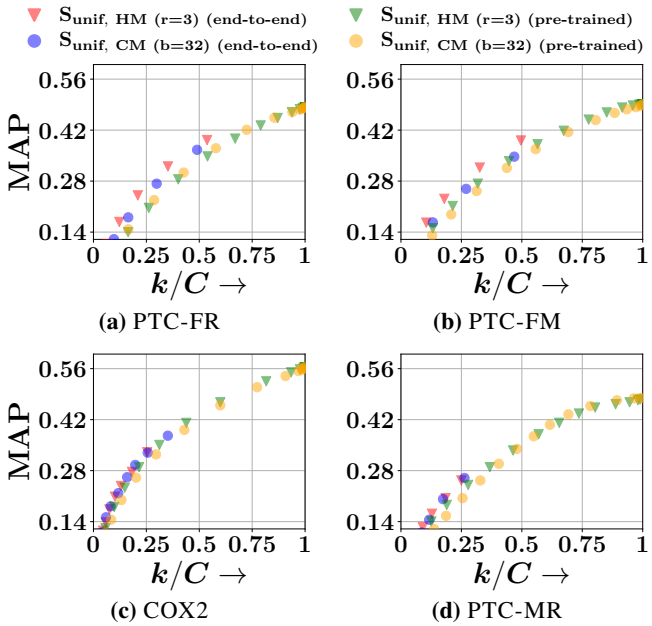
The large gap between the actual and effective rank across datasets—particularly the low effective rank—indicates that token co-occurrences lie on a low-dimensional manifold. This suggests that both $\mathbf{PL}$ and $\mathbf{C}$ are highly compressible, enabling projection onto a lower-dimensional subspace without significant loss of information. This again resembles the behavior of text corpora, where the discrete word space may be in the tens or hundreds of thousands, but a few hundred dense dimensions suffice to encode words and documents [28].

Dataset	Actual Rank	Effective Rank
PTC-FR	393	4
PTC-FM	498	9
COX2	250	7
PTC-MR	232	3

Table 3: Actual and effective rank (95% SVD energy threshold) of the posting list matrix $\mathbf{PL}$ for each dataset.

G.6 End-to-End Training vs Frozen Backbone for $\mathbf{S}_{\text{unif, CM}}$ and $\mathbf{S}_{\text{unif, HM}}$

A key design consideration is whether GTNet benefits from end-to-end training using its own GNN encoder, or whether comparable results can be obtained using frozen embeddings from a pretrained backbone. Figure 9 compares the performance of $\mathbf{S}_{\text{unif, CM}}$ and $\mathbf{S}_{\text{unif, HM}}$ under both configurations.



(a) PTC-FR **(b)** PTC-FM

(c) COX2 **(d)** PTC-MR

Figure 9: Comparison of end-to-end training versus using a frozen backbone for $\mathbf{S}_{\text{unif, CM}}$ and $\mathbf{S}_{\text{unif, HM}}$, with $r = 3$ and $b = 32$. End-to-end training refers to learning the GTNet encoder jointly, while the frozen variant reuses pretrained embeddings.

We observe that: **(1)** End-to-end training consistently yields better MAP–selectivity tradeoffs for both CM and HM variants, indicating that learning task-specific embeddings improves token discriminability. **(2)** The frozen backbone variant spans a wider range of selectivity values (k/C), suggesting

looser token matching and higher recall, but at the cost of reduced precision. **(3)** End-to-end models tend to retrieve fewer candidates for the same threshold, reflecting tighter, more precise tokenization.