ON THE TOKEN COLLAPSE PHENOMENON IN TEXTUAL GRAPH SOFT PROMPTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent methods for injecting graph structure into LLM prompts include *graph soft prompting*, in which a graph neural network (GNN) learns graph-specific tokens that capture relational information among entities in textual prompts. A major development in this line of work is G-Retriever, which combines graph soft prompts with a verbalized description of the same graph to strengthen the textual prompt signal. In this paper, we conduct an in-depth analysis of G-Retriever and uncover a pathological behavior we term *token collapse*. Across three benchmarks, three graph encoders, two LLM families, and both frozen and LoRA-adapted models, G-Retriever's graph tokens become almost indistinguishable across inputs. We provide evidence that the same failure recurs in subsequent methods to G-Retriever, even with multi-token pooling. Under the standard setup, swapping a token with the dataset mean, a token from another example, or one derived from a corrupted graph produces no measurable change in answer accuracy. Moreover, a learned constant prompt that never accesses the graph matches or surpasses G-Retriever on all three benchmarks. These findings indicate that, rather than conveying graph content as intended, the collapsed token functions primarily as a generic soft prompt. We examine conditions linked to this collapse and propose a two-stage procedure that yields non-collapsed tokens while maintaining competitive performance in most cases. However, token variability alone does not prove that the model is using graph information: via interventions and decoding probes, we show that depending on the configuration, a token can reflect the question, an answer prior, or graph-derived content that the LLM may or may not actually leverage. Overall, our results offer a practical approach to evaluating graph-token alignment rather than inferring it from task accuracy alone. Code to reproduce our experiments is available here.

1 INTRODUCTION

Large language models (LLMs) store knowledge in their parameters, making information outside or newer than their pre-training data hard to access or update (Lewis et al., 2020). Such external knowledge may be organised as text-attributed graphs (He et al., 2024). This information can be incorporated into an LLM either by *textualizing the graphs*, i.e., constructing verbalized descriptions of nodes and edges, or through *graph soft prompting*, i.e., providing a learned representation generated by a graph neural network (GNN). G-Retriever (He et al., 2024) supplies the same underlying graph via both channels. However, this design introduces an identifiability issue: because the graph content is already present in textual form, improvements in answer accuracy alone cannot distinguish whether the model relies on the graph token, the verbalized graph, or both. Furthermore, the training objective does not explicitly enforce that the graph-derived token

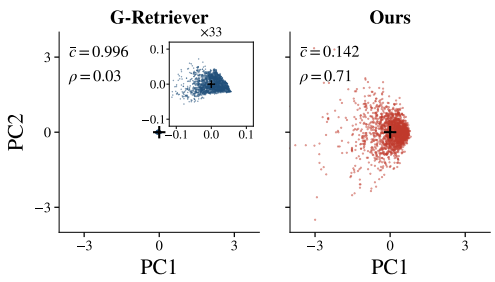


Figure 1: **Graph tokens of all 1,628 `WebQSP` test questions**, projected onto two principal components after removing the mean token μ, with pairwise cosines shown below.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

encodes graph-specific information when the textual channel suffices. Consequently, a token computed from the graph may degenerate into a graph-invariant soft prompt.

This study evaluates the proposed mechanism by quantifying both the information encoded in graph-derived tokens and the information that the large language model (LLM) causally exploits during inference. The experimental protocol integrates token-space statistical analyses with fixed-weight causal interventions: individual tokens are substituted with (i) the training-set mean token, (ii) a token sampled from a different example, or (iii) a token recomputed from perturbed (corrupted) or counterfactual graph inputs. In addition to answer accuracy, we assess gold-answer log-likelihood to capture subtler shifts in the model's output distribution, and we employ decoding-based probes to measure the recoverability of question, answer, and graph attributes from internal representations. Graph-free control conditions preserve the projector architecture and the full training pipeline while ablating the graph input itself.

Applying this protocol to G-Retriever uncovers an unanticipated phenomenon that we call *token collapse*. Specifically, across several graph encoders, benchmark datasets, LLM families, and fine-tuning regimes, the token representations generated by G-Retriever are nearly invariant with respect to the input graph (Figure 1). Moreover, substituting these input-dependent projected tokens with a fixed alternative yields no measurable change at the answer level under the default configuration; indeed, a learned constant token that never observes any graph matches or even surpasses the performance of the full model. These results indicate that the collapsed token operates primarily as a soft prompt rather than as a carrier of graph-specific information. Although likelihood-based analyses show that perturbing the token can modulate the output distribution, such sensitivity alone is insufficient to demonstrate input- or graph-conditioned utilization.

We associate collapse with two factors: a projector that suppresses variation in the encoder representations and an answer objective that does not require this variation to be preserved. Our ablations implicate both factors, although they do not isolate either as the sole cause. We address them with a two-stage pipeline that first aligns and freezes the graph encoder, then trains separate graph-token and prompt-token pathways. The resulting tokens no longer collapse, but non-collapse alone guarantees neither graph use nor improved accuracy. Depending on the benchmark and training configuration, they may encode the question, an answer prior, graph information made redundant by the text, or graph information that the LLM uses to answer. This distinction matters because task improvements may otherwise be attributed to graph–language alignment even when they arise from the verbalised graph or a graph-independent soft prompt. By separating token variation, graph content, and downstream use, our protocol makes alignment empirically testable and provides concrete criteria for developing graph–language models that preserve and use task-relevant graph information beyond what is already supplied by the textual channel.

Contributions. We identify *token collapse* across G-Retriever benchmarks, graph encoders, LLM families, and tuning regimes, and find the same failure in later graph-prompting interfaces. Using matched graph-free controls and fixed-weight interventions, we show that the collapsed token acts primarily as a soft prompt rather than as a graph representation used to answer. We associate collapse with the projector and end-to-end answer training, and propose a two-stage pipeline that mitigates it across the tested settings. Finally, we show that a non-collapsed token may encode *question* or *answer priors*, *unused graph information*, or *graph information* that changes the answer. Together, these results provide a practical protocol for testing graph-token alignment rather than inferring it from task accuracy.

2 Related work

Graph soft prompting. Soft prompting conditions an LLM with learned continuous vectors (Lester et al., 2021; Li & Liang, 2021); graph soft prompting instead projects graph-encoder outputs into one or more such tokens. GraphToken (Fatemi et al., 2024), GraphLLM (Chai et al., 2026), GNP (Tian et al., 2024), GraphGPT (Tang et al., 2024), and G-Retriever (He et al., 2024) differ in their encoders, token counts, retrieval, and LLM adaptation. In this paper, we test whether G-Retriever and subsequent interfaces preserve and use graph-specific information in their projected tokens.

Evaluating graph–language models. Recent work questions whether existing objectives and benchmarks require graph–language integration. Zhang et al. (2026) show that autoregressive train-

ing can favour a text-dominant solution, while Petkar et al. (2026) find that single-modality and raw soft-prompt baselines remain competitive and introduce CLEGR to test compositional graph–text reasoning. We complement these studies by auditing the graph token– LLM interface of graph soft prompting. Extended related work discussion is in Appendix S.

3 METHOD

3.1 PRELIMINARIES: GRAPH SOFT PROMPTING IN G-RETRIEVER

A *textual graph* $G = (\mathcal{V}, \mathcal{E}, \{x_v\}_{v\in\mathcal{V}}, \{x_e\}_{e\in\mathcal{E}})$ associates every node v with a text attribute x_v and every edge e with a text attribute x_e. Each example i consists of a graph, a question q_i, and an answer y_i.

Retrieval. Node, edge, and question texts are embedded by a frozen sentence encoder $\mathrm{LM}(\cdot)$ into $\mathbb{R}^d$ (Reimers & Gurevych, 2019). The respective representations are $\mathbf{z}_v = \mathrm{LM}(x_v)$, $\mathbf{z}_e = \mathrm{LM}(x_e)$, and $\mathbf{q}_i = \mathrm{LM}(q_i)$. When the complete graph exceeds the LLM's context length or a predefined budget, only a relevant subgraph can be supplied to the model. Following G-Retriever, similarities between $\mathbf{q}_i$ and the node and edge embeddings define retrieval relevance, and a prize-collecting Steiner tree is used to select a compact, connected subgraph (Bienstock et al., 1993). We denote the resulting question-specific subgraph by G_i. When the complete graph fits within the budget, no retrieval is needed and $G_i = G$.

Graph token. A graph encoder f_ϕ processes G_i using node features $\mathbf{z}_v$, edge features $\mathbf{z}_e$, and optionally the question representation $\mathbf{q}_i$. G-Retriever uses a graph transformer by default (Shi et al., 2020); we also evaluate GAT (Veličković et al., 2018) and GCN (Kipf & Welling, 2017). The node representations are mean-pooled into $\mathbf{h}_i \in \mathbb{R}^{d_g}$, and a projector P_ψ maps the result into the LLM embedding space:

$$\mathbf{t}_i = P_\psi(\mathbf{h}_i) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{h}_i + \mathbf{b}_1) + \mathbf{b}_2 \in \mathbb{R}^{d_\ell}, \tag{1}$$

where σ is the logistic sigmoid and the projector has hidden width d_p. We call $\mathbf{t}_i$ the *graph token*.

Generation. The retrieved subgraph is also verbalised as a list of nodes and edges, denoted by $\tau(G_i)$, and truncated to a budget of $L_{\max}$ tokens. In the default frozen-LLM setting, the model receives

$$[\,\mathbf{e}_{\mathrm{bos}};\ \mathbf{t}_i;\ \mathrm{Emb}(\tau(G_i));\ \mathrm{Emb}(q_i)\,].$$

The graph token therefore precedes the verbalised graph and the question. The LLM parameters θ remain fixed, while the graph encoder and projector are trained with the answer loss

$$\mathcal{L}_{\mathrm{QA}}(\phi,\psi) = -\sum_i \sum_k \log p_\theta\big(y_{i,k} \mid y_{i,<k}, \mathbf{t}_i, \tau(G_i), q_i\big). \tag{2}$$

The graph encoder does not receive the question directly, so in G-Retriever the token depends on the question only through retrieval.

3.2 THE GRAPH TOKEN COLLAPSE PHENOMENON

Task accuracy conflates three distinct questions: *Does the token vary across inputs? Does the LLM use the graph token? What information does it contain?* We address them with complementary measurements of token geometry, fixed-weight interventions, gold-answer likelihood, and decodability.

Does the token vary across inputs? We measure inter-sample cosine, the input-specific residual relative to the mean token, and token norm relative to ordinary word embeddings. Collapse appears as cosine near one and a small residual ratio. We also compute inter-sample cosine before projection, on the pooled encoder representations $\mathbf{h}_i$, to determine where collapse arises. Formal definitions are given in Appendix D.

Does the LLM use graph-specific token variation? At fixed weights, REAL uses the original token, MEAN replaces it with the training-set mean, and SWAP uses another example's token. Our main graph-specific intervention, GSWAP, recomputes the token from another graph while preserving the original question conditioning. NODE, EDGE, and REL corrupt individual graph components, while NOTXT removes the verbalised graph. Retrained no-entity and no-question variants isolate

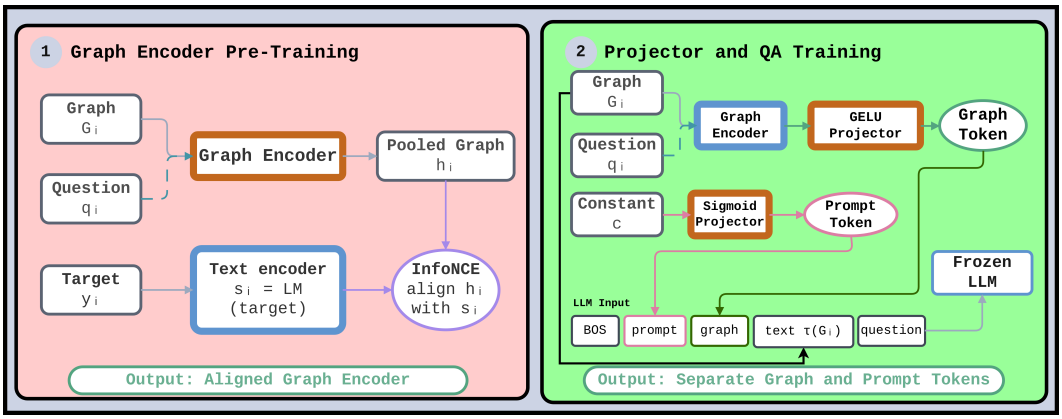


Figure 2: **Two-stage pipeline.** Stage 1 aligns the question-conditioned graph encoder. Stage 2 freezes it and trains separate graph-token and prompt-token pathways. Orange modules are trained; blue modules are frozen.

the information available during training. Because accuracy observes only changes in the decoded answer, we also measure the paired change in gold-answer negative log-likelihood under teacher forcing. This detects smaller changes in the output distribution. Full details on our interventions are in Appendix D.

What does the token contain? Recoverable content and downstream use are not equivalent: a token may contain information that the LLM ignores. This matters for our pipeline because Stage 1 is conditioned on the question and trained with an answer-related target. A non-collapsed token may therefore encode the **question**, an **answer** prediction or prior, the **graph**, or a **mixture**. The answer is a supervision target rather than an input, so it cannot be removed through a fixed-weight intervention; appendix Table 8 maps the information retained by the remaining conditions.

We complement the interventions with a post-hoc probe that maps stored tokens to frozen embeddings of the question, answer, or verbalised graph. The map is fitted on half of the test examples; on the held-out half, we report the median rank of each example's correct target among all candidates. A constant token provides the control rank. Estimation details and limitations are given in Appendix D.

Graph-free controls. **CONST** passes a learned graph-independent vector through G-Retriever's projector, while **MLPQ** replaces the graph encoder with an MLP over the question. **PT** is the raw prompt-tuning baseline and does not use the projector. These controls distinguish the contribution of graph input from that of token parameterisation.

3.3 Preventing Collapse: A Two-Stage Pipeline

Our ablations associate collapse with two factors: a Sigmoid projector that suppresses encoder variation and end-to-end answer training that does not require this variation to be preserved. We address both while retaining G-Retriever's data, prompt, answer loss, and optimisation setup (Figure 2).

Stage 1: preserve input-specific variation. Without the LLM, we train the graph encoder with an InfoNCE objective (van den Oord et al., 2019). Its pooled representation $\mathbf{h}_i$ is aligned with a frozen embedding of a sample-specific target and contrasted against targets in a memory bank. The question conditions each node through $\mathbf{x}_v = \mathbf{z}_v + \mathbf{W}_q\mathbf{q}_i$, with $\mathbf{W}_q$ initialised at zero. We use the answer as the target, or the verbalized graph, when the answer space is too small to distinguish samples. The encoder is then frozen.

Stage 2: separate graph representation from prompting. Using the original answer loss, we jointly train a GELU projector over the frozen encoder and a separate graph-independent prompt token. Freezing protects the variation learned in Stage 1, while the second pathway gives the shared prompting function its own token rather than forcing it to compete with input-specific information.

The full objective, targets, and design ablations are reported in Appendix N.

Table 1: **Collapse across datasets, graph encoders, LLMs, and tuning regimes.** Scores are accuracy for `ExplaGraphs` and `SceneGraphs` and Hit@1 for `WebQSP`.

Dataset	Model	Score	Encoder cosine	Token cosine
`ExplaGraphs`	G-Retriever (GT)	85.56	0.678	0.997
	G-Retriever (GAT)	85.20	0.457	0.998
	G-Retriever (GCN)	78.88	0.423	0.985
`SceneGraphs`	G-Retriever (GT)	81.35	0.630	0.993
	G-Retriever (GAT)	80.20	0.693	0.998
	G-Retriever (GCN)	80.55	0.428	0.998
`WebQSP`	G-Retriever (GT)	70.33	0.593	0.996
	+ LoRA	74.32	0.621	0.997
	+ Qwen2.5-3B	55.47	0.589	0.997

4 EXPERIMENTS

We investigate four questions. **RQ1:** Do graph tokens collapse, and how general is this behaviour? **RQ2:** What role does the collapsed token play when the same graph is also verbalised in the prompt? **RQ3:** What is collapse associated with, and can it be prevented without sacrificing task performance? **RQ4:** What do non-collapsed tokens encode, and which of that information does the LLM use?

4.1 SETUP

Benchmarks. We use the three GraphQA benchmarks introduced with G-Retriever (He et al., 2024): `ExplaGraphs`, `SceneGraphs`, and `WebQSP`. Following their official splits, retrieval procedures, and evaluators, we report accuracy on `ExplaGraphs` and `SceneGraphs` and Hit@1 on `WebQSP`. Dataset statistics and retrieval settings are in Appendix B.

Models. We reproduce G-Retriever with its default graph transformer (GT) and frozen Llama-2-7B (Touvron et al., 2023). Robustness experiments use GAT, GCN, Qwen2.5-3B (Qwen Team, 2025), and LoRA adaptation. Architecture, optimisation, and pipeline settings are in Appendix A.

Evaluation stability. Re-evaluating identical `WebQSP` weights in a separate process changes Hit@1 by 1.04 points. We therefore require an answer-level difference larger than 2 points with $p < .05$. Gold-answer likelihood is more stable; we require $|\Delta\mathrm{NLL}| > 0.03$ and a bootstrap interval excluding the replicate floor. Calibration results are in Appendix D.

Statistical testing. Accuracy and Hit@1 comparisons are paired with the exact McNemar test (McNemar, 1947). We denote $p < .10$, $.05$, $.01$, and $.001$ by †, *, **, and ***, respectively; unmarked comparisons have $p \geq .10$. Reported p-values are unadjusted, so claims rely on the effect-size criteria above and consistent evidence rather than isolated significant cells.

4.2 RQ1: DO GRAPH TOKENS COLLAPSE, AND HOW GENERAL IS COLLAPSE?

Collapse occurs in every tested configuration. Token cosine ranges from 0.985 to 0.998, while the input-specific residual is only 4–12% of the shared mean, whose norm is 22–97 times that of an ordinary word embedding (Appendix E). Figure 1 illustrates this geometry on `WebQSP`. Models with similarly collapsed tokens nevertheless obtain substantially different scores, so task performance does not reveal collapse.

The encoder representations are considerably more variable, with cosines between 0.423 and 0.693. This makes GNN oversmoothing (Rusch et al., 2023) an insufficient explanation for token collapse: even if node representations are smoothed within a graph, the pooled outputs remain distinct across inputs, and near-identity appears only after projection. Collapse also persists across three graph encoders, two LLM families, and frozen and LoRA-adapted LLMs. Moreover, a controlled `WebQSP` audit finds the same geometry in later single- and multi-token graph-prompting interfaces (Appendix F). This answers **RQ1**: collapse is a recurring failure mode rather than a peculiarity of

Table 2: **Role of G-Retriever's collapsed token.** REAL, NOTXT, and CONST are absolute scores; other columns report differences from their corresponding reference. Full results in Appendices I, J, and K.

Dataset	REAL	ΔMEAN	ΔSWAP	ΔGSWAP	NOTXT	ΔNOTXT-MEAN	CONST
`ExplaGraphs`	85.56	−0.18	−1.26	−1.08	66.43	−0.90	85.56
`SceneGraphs`	81.35	−0.35	−1.20†	−1.10†	30.20	+0.45	81.40
`WebQSP`	70.33	−1.84*	−0.92	−0.55	41.34	+1.04	71.25

Table 3: **Factors associated with collapse on `ExplaGraphs`.** Step 0 precedes projector training, and significance markers compare task score with the first row. Full projector diagnostics and intervention results are in Appendices H and I.

Encoder	Projector	Encoder cos.	Step-0 token cos.	Token cos.	Score
trained end to end	Sigmoid	0.695	1.000	0.997	85.56
trained end to end	GELU	0.787	0.992	0.914	73.29***
frozen after Stage 1	Sigmoid	0.024	0.987	0.992	84.48
frozen after Stage 1	GELU	0.024	0.091	0.323	57.76***

one encoder, LLM, or single-token implementation. We next ask what the collapsed token contributes to the answer.

4.3 RQ2: What role does the collapsed token play?

The input-specific component has no resolved answer-level effect. Across all three benchmarks, MEAN, SWAP, and GSWAP remain within the two-point threshold (Table 2); corrupting entities, edges, or relations gives the same result (more evidence in Appendix I). Yet replacing the token with zero costs 59.64–81.95 points (see Appendix J). The model therefore relies on the token's shared value, but not measurably on its graph-specific variation.

The verbalised graph is the dominant graph channel. Removing it costs 19–51 points, whereas replacing the remaining token with its mean changes performance by at most 1.04 points. Thus, in the default Llama-2 configuration, task-relevant graph information is supplied primarily through text.

The shared token acts as a soft prompt. CONST, a graph-independent vector passed through the same projector, matches or exceeds G-Retriever on every benchmark. Graph input is therefore unnecessary for the benefit attributed to the collapsed token. Raw prompt tuning, namely the existing ablation performed by He et al. (2024), performs worse (Appendix K), but changes the parameterisation and is not a controlled graph-versus-no-graph comparison.

Small distributional effects remain: on `WebQSP`, SWAP and GSWAP increase gold-answer NLL by 0.0183 and 0.0182, both below the 0.03 criterion (Appendix L). These results answer **RQ2**: the collapsed token functions primarily as a projector-parameterised soft prompt, while the verbalised graph is the dominant graph-information channel. **RQ3** asks what collapse is associated with and whether it can be prevented.

4.4 RQ3: What is collapse associated with, and can it be prevented?

Factors associated with collapse. Table 3 identifies two associations. With the aligned Stage 1 encoder frozen, its representations are highly distinct (cosine 0.024), yet the Sigmoid projector produces nearly identical tokens before and after training (0.9870 and 0.9918). GELU instead yields cosine 0.0910 before training and 0.3229 afterwards. Untrained-projector experiments reproduce this contrast on random inputs (Appendix H), linking the Sigmoid parameterisation to suppression of encoder variation.

Table 4: **Token geometry, performance, and intervention effects.** *Ours* denotes the complete two-token pipeline; Intervention columns report condition minus REAL. Bold scores mark the highest score within each dataset–LLM block; bold GSWAP effects satisfy $|\Delta| > 2$ and $p < .05$. Complete interventions and p-values are in Appendix I, and pipeline ablations are in Appendix O.

Dataset	Model	Score	Token cos.	ΔMEAN	ΔSWAP	ΔGSWAP
Llama-2-7B						
WebQSP	G-Retriever	70.33	0.996	-1.84^{*}	-0.92	-0.55
	Ours	69.90	0.142	-3.13^{***}	-2.15^{*}	$-1.97^{\dagger}$
	no entity features	**72.73**	0.125	-6.14^{***}	-5.65^{***}	-1.97^{*}
	no question	69.84	0.169	-0.06	-3.69^{***}	$\mathbf{-2.33}^{*}$
SceneGraphs	G-Retriever	**81.35**	0.993	-0.35	$-1.20^{\dagger}$	$-1.10^{\dagger}$
	Ours	72.50	0.133	-1.40	-3.40^{***}	$\mathbf{-5.30}^{***}$
	no entity features	79.65	0.125	-5.75^{***}	-9.60^{***}	$+0.35$
	no question	76.35	0.225	-1.15	-0.25	-0.25
ExplaGraphs	G-Retriever	85.56	0.997	-0.18	-1.26	-1.08
	Ours	85.74	0.099	$+0.36$	-0.54	-1.44
	no entity features	84.48	0.113	$+3.07^{**}$	0.00	-0.36
	no question	**87.55**	0.108	$+1.08$	$+0.90$	$+1.26$
Qwen2.5-3B, matched training budget						
WebQSP	G-Retriever	57.13	0.997	-0.55	-1.41	$-1.60^{\dagger}$
	Ours	**65.79**	0.200	-2.09^{**}	-1.90^{**}	$\mathbf{-2.09}^{**}$
SceneGraphs	G-Retriever	**76.05**	0.997	-0.30	$-1.05^{\dagger}$	-1.15^{*}
	Ours	72.65	0.139	-7.00^{***}	-5.75^{***}	$\mathbf{-6.50}^{***}$
ExplaGraphs	G-Retriever	74.37	0.998	-1.62	-2.17	$\mathbf{-4.51}^{*}$
	Ours	**83.94**	0.103	$+0.90$	0.00	$+0.36$

Changing the projector alone is insufficient: under end-to-end answer training, GELU still produces highly similar tokens (cosine 0.9139). Conversely, combining the aligned frozen encoder with GELU prevents collapse, but the single-token model falls to approximately majority-class performance (57.76 versus 58.8). Thus, both factors are associated with collapse, but this experiment does not identify either as the sole cause. The cross-method audit supports this qualification (Appendix F). GRAG (Hu et al., 2025) and GRETPool (Grover et al., 2026) inherit the same Sigmoid projector and also collapse, while GraSP (Wang et al., 2026) collapses despite using Tanh.

Preventing collapse while retaining performance. Table 4 shows that the complete two-token pipeline mitigates collapse in every tested setting. For the standard pipeline, token cosine falls from above 0.99 to 0.099–0.142 with Llama-2 and to 0.103–0.200 with Qwen2.5-3B.

With Llama-2, the pipeline retains performance on WebQSP (69.90 versus 70.33) and ExplaGraphs (85.74 versus 85.56), with both differences below our resolution threshold. On ExplaGraphs, the separate prompt pathway recovers the performance lost by the non-collapsed single-token model, raising its score from 57.76 to 85.74. On SceneGraphs, however, GSWAP produces a resolved -5.30-point effect while the score falls from 81.35 to 72.50.

The Qwen2.5-3B replication further separates performance from graph use. The Qwen rows in Table 4 use a matched training budget and are distinct from the robustness checkpoint in Table 1. On WebQSP, our pipeline improves the matched baseline from 57.13 to 65.79 ($p = 4.5 \times 10^{-20}$), while GSWAP produces a resolved -2.09-point effect ($p = .0027$). On ExplaGraphs, it improves from 74.37 to 83.94 ($p = 1.5 \times 10^{-5}$), but GSWAP has no effect ($+0.36$). Conversely, on SceneGraphs, it produces the strongest graph-replacement effect in the study (-6.50, $p < .001$) while scoring 3.40 points below G-Retriever. Qwen G-Retriever on ExplaGraphs provides the opposite exception: its tokens remain collapsed, yet GSWAP causes a resolved -4.51-point effect. Near-collapse therefore does not imply complete insensitivity to the remaining variation.

Table 5: **Token content and use with Llama-2.** Q, A, and G are median retrieval ranks (lower is better); constant controls are 407 for `WebQSP`, 500 for `SceneGraphs` (A: 476–500), and 139 for `ExplaGraphs` (A omitted). ΔScore is the with-text GSWAP effect; NLL reports SWAP/GSWAP. Bold denotes effects meeting the criteria in Section 4.1.

Dataset	Model	Q rank	A rank	G rank	ΔScore GSWAP	ΔNLL S/GS	Diagnosis
`WebQSP`	G-Retriever	293	317	369	-0.55	$+0.0183/+0.0182$	SOFT PROMPT
	Ours	65	60	315	$-1.97^{\dagger}$	$\mathbf{+0.0895}/\mathbf{+0.0332}$	MASKED
	no entity features	34	51	303	-1.97^{*}	$\mathbf{+0.1111}/+0.0262$	PRIOR
	no question	120	134	329	$\mathbf{-2.33}^{*}$	$\mathbf{+0.0848}/\mathbf{+0.0845}$	USED
`SceneGraphs`	G-Retriever	271	436	248	$-1.10^{\dagger}$	$+0.0169/+0.0167$	SOFT PROMPT
	Ours	47	187	427	$\mathbf{-5.30}^{***}$	$+0.0229/\mathbf{+0.1665}$	USED
	no entity features	8	59	369	$+0.35$	$\mathbf{+0.1553}/+0.0045$	PRIOR
	no question	116	395	185	-0.25	$+0.0214/+0.0213$	MASKED
`ExplaGraphs`	G-Retriever	138	–	139	-1.08	$+0.0014/+0.0022$	SOFT PROMPT
	Ours	54	–	41	-1.44	$+0.0010/+0.0032$	DECODABLE
	no entity features	67	–	67	-0.36	$+0.0026/+0.0056$	DECODABLE
	no question	84	–	70	$+1.26$	$-0.0063/-0.0073$	DECODABLE

These results answer **RQ3**. On `ExplaGraphs`, collapse is associated with both the projector and end-to-end answer training, without either being isolated as its sole cause. The two-stage pipeline mitigates collapse across both LLM families and can retain or improve performance. However, geometric collapse, graph-specific token use, and task performance remain distinct properties. **RQ4** examines what the tokens encode and which information the LLM uses.

4.5 RQ4: WHAT DOES A NON-COLLAPSED TOKEN CARRY?

Non-collapse shows that tokens differ across inputs, but not what they represent or whether the LLM uses those differences. Because Stage 1 receives the question and an answer-related target, the token may encode the question, an answer prior, the graph, or a mixture.

We separate recoverable content from downstream use. Table 5 reports question, answer, and graph retrieval ranks, answer-level effects under GSWAP, and likelihood changes under SWAP and GSWAP. Table 4 measures reliance on sample-specific token information through MEAN and SWAP, while the appendix Table 70 repeats the interventions without the retrieved verbalised graph. Complete cross-LLM results are reported in Appendix M.

The retrieval probe maps each token to the corresponding question, answer, or verbalised graph among held-out candidates. We report the median rank of the correct target: lower is better, and a constant token provides the control rank. Recoverability does not imply use. A low answer rank, for example, means that the answer is predictable from the token, not that it was computed from the graph or used by the LLM.

We use five diagnoses. USED means that GSWAP changes the task score by more than two points with $p < .05$. MASKED means that this effect resolves only after removing the verbalised graph. PRIOR identifies a no-entity model whose token carries question and answer information and is used under MEAN or SWAP, without resolved graph-specific use. DECODABLE means that graph information is recoverable without a resolved answer-level effect. SOFT PROMPT denotes a collapsed baseline without resolved graph-specific use; any information recoverable from its residual variation is reported separately by the retrieval ranks.

`WebQSP`: the token's role depends on training. The standard token makes the question and answer strongly recoverable, at ranks 65 and 60 against the control of 407. With the verbalised graph present, GSWAP affects likelihood but not the decoded score; after removing the text, it causes a resolved -2.27-point effect (Tables 5 and 70). Removing entity features further strengthens question and answer recovery and improves Hit@1 without resolved graph use, supporting the PRIOR diagnosis. Conversely, removing question conditioning produces resolved graph use. Thus, the same architecture can favour priors or graph information depending on its training inputs.

`SceneGraphs`: graph use and accuracy diverge. Our pipeline uses graph-specific token content: GSWAP lowers accuracy by 5.30 points and increases NLL by 0.1665, but it remains less accurate than G-Retriever. G-Retriever's collapsed token contains recoverable graph information, yet shows no resolved graph use, so its functional diagnosis remains SOFT PROMPT. The no-entity model nearly recovers its accuracy while eliminating graph use, confirming that performance and token use are distinct.

`ExplaGraphs`: graph information is decodable but unused. Our token makes the graph substantially more recoverable, at rank 41 against the control of 139, but GSWAP has no resolved effect with or without the verbalised graph. Because these graphs are small and supplied in full, their textual representation may already be sufficient. However, removing that text does not make the model rely on the token. The evidence therefore establishes recoverable graph information, not downstream use.

Together, these results answer **RQ4**: preventing collapse does not determine what a token represents or whether the LLM uses it. Depending on the benchmark and training inputs, a non-collapsed token may encode a question or answer prior, unused graph information, or graph information that changes the answer.

5 DISCUSSION

What standard evaluation misses. Task accuracy cannot isolate graph-token use, while raw prompt tuning changes the token parameterisation and no-text retraining changes the learning problem. A minimal audit should combine token geometry, a graph-free control using the same projector, and fixed-weight MEAN, SWAP, and GSWAP interventions. The controlled `WebQSP` audit in Appendix F finds collapse with query-conditioned and multi-token interfaces, extending the phenomenon beyond G-Retriever.

Graph use and accuracy are different outcomes. Our pipeline can make answers depend on graph-specific token variation without improving accuracy. The textual channel may be easier to exploit, while continuous graph tokens may be lossy or difficult for the LLM to interpret. The Qwen2.5-3B results show that this trade-off is not inevitable; alignment claims should therefore report graph use and performance separately.

Limitations and future directions. Our detailed analysis covers G-Retriever on three question-answering benchmarks; the cross-method audit is limited to `WebQSP`, runnable implementations, and controlled variants using G-Retriever's retrieved subgraphs. The mechanism evidence is associative and does not isolate the projector or end-to-end training as a sole cause.

The audited recipe trains from scratch with answer cross-entropy; collapse may differ with pre-trained encoders or explicit contrastive, reconstructive, or counterfactual objectives (Zhang et al., 2026; Xu et al., 2025). Stage 1 explores one alternative, but its conditioning and targets can introduce priors. GSWAP tests graph–question pairing rather than fact-level reasoning, and decodability measures recoverability rather than use. Our pipeline is therefore a diagnostic proof of concept. Future work should preserve task-relevant graph information and verify that the LLM uses it.

6 CONCLUSION

We uncover token collapse in G-Retriever across benchmarks, graph encoders, LLM families, and tuning regimes, and find the same geometry in controlled tests of later single- and multi-token interfaces. Despite competitive task scores, the projected tokens are nearly identical across inputs. We also find that graph-free controls and fixed-weight interventions match the performance of the original G-retriever model. These findings show that strong task performance can conceal a graph-token interface whose principal contribution is a graph-independent soft prompt, while the verbalised graph provides the dominant graph-information channel. Our two-stage pipeline mitigates this collapse, but also demonstrates that distinct tokens do not guarantee graph use or improved accuracy. Token variation, recoverable graph information, downstream use, and task performance must therefore be evaluated separately. Our protocol provides a practical way to distinguish these outcomes and turns alignment from an architectural assumption into an empirical claim that will foster the development of future graph–language models.

AI USE STATEMENT

Generative AI tools assisted with prose revision and parts of the analysis and plotting code. They did not generate experimental results: all newly reported experimental numbers were produced by our pipeline and checked against stored predictions, tokens, checkpoints, or logs. The authors reviewed all AI-assisted text and code, verified the experimental claims, and take full responsibility for the paper.

REPRODUCIBILITY STATEMENT

Appendix A specifies the repository and code modification, optimisation settings, model dimensions, LoRA configuration, Stage-1 hyperparameters, trainable parameter counts, hardware, and evaluation procedure. Appendix B reports dataset statistics, splits, retrieval settings, metrics, and intervention subsets. Appendix I provides the complete fixed-weight intervention results; the remaining appendices report the geometry, likelihood, decodability, controls, and pipeline measurements.

REFERENCES

Emily Alsentzer, Samuel G. Finlayson, Michelle M. Li, and Marinka Zitnik. Subgraph neural networks, 2020. URL https://arxiv.org/abs/2006.10538.

Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59(1):413–420, 1993.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. GraphLLM: Boosting graph reasoning ability of large language model. *IEEE Transactions on Big Data*, 12(2):475–483, 2026. doi: 10.1109/TBDATA.2025.3627488.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Hongyang Gao and Shuiwang Ji. Graph u-nets, 2019. URL https://arxiv.org/abs/1905.05178.

Ankit Grover, Lodovico Giaretta, Rémi Bourgerie, and Sarunas Girdzijauskas. Is one token all it takes? graph pooling tokens for LLM-based GraphQA, 2026.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4145–4157, 2025.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling, 2019. URL https://arxiv.org/abs/1904.08082.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190.

Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. Graphadapter: Tuning vision-language models with dual knowledge graph, 2023. URL https://arxiv.org/abs/2309.13625.

Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Soham Petkar, Hari Aakash K, Anirudh Vempati, Akshit Sinha, Ponnurangam Kumaraguru, and Chirag Agarwal. A graph talks, but who's listening? rethinking evaluations for graph-language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32441–32462, 2026. doi: 10.18653/v1/2026.findings-acl.1624.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410.

T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks, 2023. URL https://arxiv.org/abs/2303.10993.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V. Chawla, and Panpan Xu. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Runze Wang, Mingqi Yang, and Yanming Shen. Bridging molecular graphs and large language models, 2025. URL https://arxiv.org/abs/2503.03135.

Shuai Wang, Xixi Wang, and Yinan Yu. Topology-aware reasoning over incomplete knowledge graph with graph-based soft prompting, 2026. URL https://arxiv.org/abs/2604.12503.

Derong Xu, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Maolin Wang, Qidong Liu, Xiangyu Zhao, Yichao Wang, Huifeng Guo, Ruiming Tang, Enhong Chen, and Tong Xu. Align-GRAG: Reasoning-guided dual alignment for graph retrieval-augmented generation. *arXiv preprint arXiv:2505.16237*, 2025.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Zhongjian Zhang, Xiao Wang, Mengmei Zhang, Jiarui Tan, and Chuan Shi. Toward graph-tokenizing large language models with reconstructive graph instruction tuning. In *Proceedings of the ACM Web Conference 2026*, WWW '26, pp. 430–441, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400723070. doi: 10.1145/3774904.3792077. URL https://doi.org/10.1145/3774904.3792077.

Appendix Overview

Setup. A implementation and hyperparameters; B datasets and retrieval; C graph encoders.

Protocol. D condition taxonomy, likelihood and decodability estimators, evaluation stability and the statistical protocol.

Collapse. E complete collapse statistics and per-encoder interventions; F complete collapse statistics for methods subsequent to G-Retriever; G token space of every model; H projector diagnostics behind Section 4.4.

Causal audit. I full fixed-weight interventions; J magnitude and geometric conditions; K complete graph-free controls; L gold-answer likelihood.

Content and pipeline. M complete decodability results; N design choices and Stage-1 diagnostics; O pipeline and robustness ablations; P cross-fitting; Q initialisations and evaluation noise.

Context. R the verbalised graph is not enough; S extended related work.

A Implementation details

G-Retriever reproduction. We use the authors' public implementation at a pinned commit. For the baseline reproduction, our only modification is explicit device placement, allowing each run to fit on a single GPU. The frozen sentence encoder is all-roberta-large-v1 with output width $d = 1024$. The default graph encoder is a four-layer, four-head graph transformer with pooled width $d_g = 1024$. Its representation is mapped to the LLM embedding space by a two-layer projector with hidden width $d_p = 2048$. The default LLM is frozen Llama-2-7B, whose embedding width is $d_\ell = 4096$. The verbalised graph is truncated to $L_{\max} = 512$ tokens, and generation is limited to 32 tokens.

Optimisation follows the authors' defaults: AdamW with learning rate 10^{-5}, weight decay 0.05, and at most 10 epochs, with early stopping after two epochs without validation improvement. Every model in this paper is trained on a single 80 GB GPU.

Stage 1. InfoNCE with temperature 0.07 against a memory bank of about 2,000 targets, with no LLM loaded. Trainable: 20,994,048 parameters in the encoder and 1,049,600 in the question-conditioning layer $\mathbf{W}_q$.

Stage 2. The Stage-1 encoder and $\mathbf{W}_q$ parameters are frozen, while their running normalisation statistics remain in training mode. The projector's Sigmoid is replaced by GELU, and a learned constant of width d_g passes through its own Linear $\rightarrow$ Sigmoid $\rightarrow$ Linear projector to produce the prompt token. The two projector pathways are initialised and optimised jointly; checkpoint-derived trainable parameter counts are reported in Table 64. Interventions modify only the graph token, never the prompt token.

Controls and parameter counts. Table 64 reports checkpoint-derived trainable parameter counts for the evaluated models. Stage 1 separately trains $22{,}043{,}648$ parameters; because the two stages are trained sequentially, we do not report their sum as a simultaneous trainable count. Accuracy does not track capacity: CONST improves over PT by 19 points, while our larger Stage-2 model scores 1.35 points below CONST. Since PT and CONST differ in both parameterisation and capacity, their gap should not be attributed to the projector alone. The controlled graph-input comparison is instead between CONST and G-Retriever, which share the projector parameterisation.

B DATASETS

We use the GraphQA benchmark assembled by He et al. (2024), whose statistics are reproduced in Table 6. Retrieval is needed on two of the three: verbalising an average `WebQSP` graph takes 100,627 tokens, reduced to 610 after retrieval, and an average `SceneGraphs` graph 1,396 tokens, reduced to 235. `ExplaGraphs` graphs fit in the prompt, so the whole graph is used. Retrieval keeps the top 3 nodes and edges on `SceneGraphs` with edge cost 1, and the top 3 nodes and 5 edges on `WebQSP` with edge cost 0.5.

Table 6: GraphQA, textual graphs benchmark. Numbers are from He et al. (2024).

	`ExplaGraphs`	`SceneGraphs`	`WebQSP`
Graphs	2,766	100,000	4,737
Average nodes	5.17	19.13	1,370.89
Average edges	4.25	68.44	4,252.37
Node attributes	commonsense concepts	objects and attributes	Freebase entities
Edge attributes	commonsense relations	actions, spatial relations	Freebase relations
Task	stance prediction	scene-graph QA	knowledge-graph QA
Metric	accuracy	accuracy	Hit@1
Test questions	554	20,025	1,628

Metrics use the benchmark's own evaluators, applied to the stored per-question predictions: on `ExplaGraphs` the first "support" or "counter" in the output must equal the label; on `SceneGraphs` the gold label must appear in the output; on `WebQSP` a prediction counts as a hit when it contains one of the gold answers. Interventions use all test questions on `ExplaGraphs` and `WebQSP`, and the first 2,000 on `SceneGraphs`, of which 1,302 are not yes/no questions. The one comparison drawn on the full `SceneGraphs` test set, prompt tuning against the graph-free constant in Section 4.3, is stated as such where it appears.

C GRAPH ENCODERS

GAT (Veličković et al., 2018) and GCN (Kipf & Welling, 2017) replace the graph transformer (Shi et al., 2020) inside G-Retriever's pipeline, using the repository's own implementations with the same depth and hidden width. Their collapse statistics and interventions appear in Tables 1 and 2, their token spaces in Figures 4 and 5, and every condition measured for them in Appendix I. The three encoders differ in accuracy on `ExplaGraphs`, where GCN loses 6.7 points to the graph transformer, but not in the behaviour of their tokens.

D PROTOCOL DETAILS

Token-geometry statistics. Let $\mathbf{t}_1, \ldots, \mathbf{t}_n$ be the test tokens, $\boldsymbol{\mu} = \frac{1}{n}\sum_i \mathbf{t}_i$ their mean, and $\bar{e}$ the median norm of the LLM's word embeddings. We measure

$$\cos_{\text{inter}} = \frac{2}{n(n-1)} \sum_{i<j} \frac{\mathbf{t}_i^\top \mathbf{t}_j}{\|\mathbf{t}_i\|\|\mathbf{t}_j\|}, \qquad \text{RR} = \frac{1}{n} \sum_i \frac{\|\mathbf{t}_i - \boldsymbol{\mu}\|}{\|\boldsymbol{\mu}\|}, \qquad \text{NR} = \frac{\frac{1}{n}\sum_i \|\mathbf{t}_i\|}{\bar{e}}. \tag{3}$$

Inter-sample cosine measures whether tokens point in the same direction, the residual ratio (RR) measures their input-specific component relative to the shared mean, and the norm ratio (NR) compares their magnitude with an ordinary word embedding. We compute the same inter-sample cosine for the pooled encoder representations $\mathbf{h}_i$ to distinguish collapse in the encoder from collapse introduced by the projector.

GSWAP. The partner graph is drawn by a fixed derangement of the test set, a cyclic shift, and two further random derangements are reported as replicates in Appendix I. The partner's graph replaces the batch's graph before the encoder, while the question conditioning $\mathbf{W}_q\mathbf{q}_i$ still uses question i, which is what makes the condition specific to the graph.

The conditions. MEAN replaces the token by the mean token over the training graphs, identical for every question. SWAP replaces it by the token of another test question under a fixed permutation. NODE recomputes the token after replacing node features with real rows from other graphs, keeping the edges; EDGE after rewiring every edge uniformly at random inside its own graph, keeping features and edge count; REL after resampling relation attributes. The conditions on the token's geometry are the residual scaling $\boldsymbol{\mu} + a(\mathbf{t}_i - \boldsymbol{\mu})$ with $a \in \{0, 2\}$, the zero token $\mathbf{0}$, and rescaling the token to a chosen norm. NOTXT-X evaluates condition x with the verbalised graph deleted from the prompt, which puts every model out of its training distribution and is read as a comparison between models on the same items.

The measured floor. Table 7 is the calibration behind the thresholds of Appendix D. The null replicate reruns the identical condition on the identical weights in a second process and compares the generated strings; models with a GNN are not bit-reproducible across processes, because scatter reductions accumulate in a nondeterministic order, and a difference of order 10^{-7} in the prefix token rewrites most greedy decodings. MLPQ's tokens are identical to within 3×10^{-4}, so its SWAP carries no information and is still significant.

Table 7: **The measured floor**, `WebQSP`, $n = 1{,}628$. Churn is the fraction of generated strings that change; no perturbation here carries graph information.

Control	Perturbation	Churn	Accuracy	Exact McNemar p
Null replicate, same weights, same condition	none at all	63.2%	-1.04	.247
MLPQ, SWAP	3×10^{-4}, cosine ≥ 0.999999	38%	-1.72	**.015**
No entity features, NODE	zero, features already zeroed	43%	-0.61	.415
CONST, SWAP	zero, all token rows identical	0%	0.00	1

Gold-answer likelihood. With y_i the gold answer of question i, truncated to the generation budget and terminated by the end-of-sequence token, and $\mathbf{t}_i^c$ the graph token under condition c:

$$\text{NLL}_i(c) = -\frac{1}{|y_i|} \sum_{k=1}^{|y_i|} \log p_\theta\big(y_{i,k} \mid y_{i,<k},\, \mathbf{t}_i^c,\, \tau(G_i),\, q_i\big), \quad \Delta_{\text{NLL}}(c) = \frac{1}{n}\sum_{i=1}^{n} \big[\text{NLL}_i(c) - \text{NLL}_i(\text{REAL})\big]. \tag{4}$$

This is the token-averaged cross-entropy of the gold answer under teacher forcing, so nothing is generated and no decoding is involved; a positive Δ_{NLL} means the intervention made the gold answer less likely, that is, the token was helping. The gold answer is truncated to 32 tokens plus the end-of-sequence token, the batch size is 1, and paired differences are reported with a 2,000-sample bootstrap confidence interval and a Wilcoxon signed-rank test.

Table 8: **Information from the original sample retained under each condition.** ✓ indicates that the information remains available when constructing the token; × indicates that it is removed or replaced. The final three rows are separately trained ablations rather than fixed-weight interventions.

		information from the original sample retained			
Condition	**What it replaces**	question	entities	structure	relations
Interventions at fixed weights					
MEAN	token → training-set mean $\boldsymbol{\mu}$	×	×	×	×
SWAP	token → another question's token	×	×	×	×
GSWAP	graph → another question's graph	✓	×	×	×
NODE	node features → resampled features	✓	×	✓	✓
EDGE	edges → uniformly rewired edges	✓	✓	×	✓
REL	relation features → resampled features	✓	✓	✓	×
Pipelines retrained with one input removed					
no entity features	node features zeroed during training and testing	✓	×	✓	✓
no question	question conditioning removed from Stage 1	×	✓	✓	✓
question MLP	graph encoder replaced by an MLP on q	✓	×	×	×

Decodability. With $\mathbf{T} \in \mathbb{R}^{m \times d_\ell}$ the stored test tokens of the fitting half, $\mathbf{Y}$ the frozen sentence embeddings of the target texts and $\tilde{\mathbf{T}} = [\mathbf{T}, \mathbf{1}]$,

$$\mathbf{W}^\star = \left(\tilde{\mathbf{T}}^\top \tilde{\mathbf{T}} + \alpha \mathbf{I}\right)^{-1} \tilde{\mathbf{T}}^\top \mathbf{Y}, \tag{5}$$

solved in double precision with $\alpha = 10^3$. The held-out half is scored by retrieval: each question's predicted embedding is ranked against all held-out targets by cosine, and we report the median rank of its own target, with ties counted against us. The split is 50/50 with seed 0, the sentence encoder is the same frozen all-roberta-large-v1 used everywhere else, and the control replaces every token by the fitting half's mean, which lands at the median rank by construction.

Two caveats bound the graph column. The sentence encoder truncates at 512 wordpieces, and the verbalised subgraphs are post-retrieval: the median is 76 wordpieces on `ExplaGraphs` with nothing truncated, 260 on `WebQSP` with 14.5% truncated and 256 on `SceneGraphs` with 23%, so the instrument is intact on the benchmark where we lean on it and is a lower bound on the other two. Compressing a subgraph into one 1024-dimensional sentence vector is lossy in any case, so the column measures whether the token aligns with the gist of the verbalisation. Failing to decode is therefore not proof of absence, while decoding successfully is proof of presence, which is the direction every claim in the paper uses.

Likelihood floor. The same null replicate gives a paired likelihood difference of $+0.00026$ with a 95% confidence interval of $[-0.00027, +0.00080]$ and Wilcoxon $p = .785$.

E Complete collapse statistics and per-encoder interventions

Table 11 gives the two token statistics dropped from Table 1, and Table 12 the per-encoder interventions dropped from Table 2. Every remaining condition of every model is in Appendix I.

F Beyond G-Retriever: token collapse in subsequent graph-prompting interfaces

Motivation and scope. We test whether collapse is confined to G-Retriever or recurs in later graph-prompting interfaces. We consider methods evaluated on `WebQSP` that construct continuous prompts from graphs and have runnable public implementations. These criteria retain GRAG (Hu et al., 2025), GRETPool (Grover et al., 2026), and GraSP (Wang et al., 2026). We exclude Graph2Token (Wang et al., 2025), which targets molecular graphs, and GraphAdapter (Li et al., 2023), which adapts vision–language models. Align-GRAG (Xu et al., 2025) could not be reproduced because its anonymous repository was no longer available.

Table 9: **Cross-method configurations.** K is the number of graph-derived tokens and N the retained PCST node count. The projector column reports the final graph-to-LLM mapping.

Method	Configuration	Projector	K regime	Mean K
GRAG-PCST (Hu et al., 2025)	released GAT + LoRA	Linear–Sigmoid–Linear	1	1.000
GRETPool (Grover et al., 2026)	TopK (Gao & Ji, 2019), ratio 0.44	Linear–Sigmoid–Linear	$\lceil 0.44N \rceil$	7.848
–	SAG (Lee et al., 2019), ratio 0.44	Linear–Sigmoid–Linear	$\lceil 0.44N \rceil$	7.848
–	DiffPool (Ying et al., 2018)	Linear–Sigmoid–Linear	8	8.000
–	DiffPool (Ying et al., 2018)	Linear–Sigmoid–Linear	100	100.000
GraSP-PCST (Wang et al., 2026)	SUBGRAPHGNN (Alsentzer et al., 2020) + Llama	Linear–Tanh–Linear	N	16.719

Controlled setup. All interfaces use the 1,628-example `WebQSP` test split and the PCST subgraphs retrieved for G-Retriever. Holding retrieval fixed prevents graph size and relevance from explaining differences in token geometry, but makes GRAG-PCST and GraSP-PCST controlled adaptations rather than complete reproductions of the published systems. GRAG-PCST retains the released query-conditioned GAT, projector, and LoRA-tuned Llama configuration. GRETPool uses its reported TopK (Gao & Ji, 2019) and SAG (Lee et al., 2019) ratios of 0.44 and DiffPool (Ying et al., 2018) budgets of 8 and 100 tokens. GraSP-PCST uses the released SUBGRAPHGNN (Alsentzer et al., 2020) and GRAPH2PREFIX with a local Llama model, producing one token per retained node. Table 9 summarises the resulting cardinality regimes.

Diagnostics across cardinalities. For inter-sample comparison, we mean-pool each example's K prompt tokens and compute the geometry statistics of Appendix D. This pooling is used only for measurement and does not alter inference. For $K > 1$, we also report the mean pairwise cosine among tokens from the same graph. All interventions preserve cardinality: donor examples have the same K, and node randomisation and edge rewiring preserve node and edge counts.

Results. Collapse occurs in all six configurations and every tested cardinality: inter-sample cosine is 0.998–1.000, and the multi-token interfaces are also internally almost collinear. GRAG and GRETPool share G-Retriever's Linear–Sigmoid–Linear projector, so their results show that collapse persists under alternative encoders and pooling operators without changing this bottleneck. GraSP provides the independent case: despite using a Linear–Tanh–Linear projector and one token per retained node, it reaches inter-sample cosine 0.9978, within-example cosine 0.9995, and residual ratio 0.039. Its smaller norm ratio (4.27) shows that collapse need not involve the unusually large token norms observed with the Sigmoid projectors. Thus, collapse can persist with multiple tokens and with either Sigmoid or Tanh projectors.

With the verbalised graph present, none of the direct token-content controls resolves: SWAP changes Hit@1 by at most 0.921 points ($p = .298$), GSWAP by at most 0.676 ($p \geq .379$), and MEAN by at most 0.799 ($p = .263$). Node randomisation is likewise unresolved. Edge rewiring produces two marginal effects: -1.413 for GRAG-PCST ($p = .0616$) and $+2.088$ for DiffPool with $K = 8$ ($p = .0525$). Neither satisfies the paper's joint requirement of $|\Delta| > 2$ and $p < .05$, and the latter improves rather than degrades performance.

The no-text tests use NOTXT-REAL, rather than the text-present score, as their paired reference. They provide limited, configuration-dependent evidence: DiffPool with $K = 100$ decreases under the mean-token diagnostic (-1.904, $p = .0472$), while DiffPool with $K = 8$ under SWAP ($+2.334$, $p = .0530$) and GraSP-PCST under GSWAP ($+2.518$, $p = .0767$) are marginal. None satisfies both the effect-size and p-value criteria. Removing the verbalised graph itself lowers performance by 25.6–43.2 points, but this comparison measures removal of textual evidence rather than graph-token specificity.

These results extend the collapse observation beyond G-Retriever: it recurs under GRAG's query-conditioned single-token interface and under fixed, ratio-selected, and node-preserving multi-token pooling. They do not show that every graph-prompting method collapses or that graph information is universally unnecessary. The paired tests establish that token-content replacements are unresolved in every text-present configuration, while the no-text conditions show only limited and inconsistent sensitivity. Because this audit remains single-seed and lacks likelihood measurements, it supports a recurrence claim rather than the stronger conclusions established for G-Retriever in the main experiments.

Table 10: **Cross-method token-collapse audit on `WebQSP`.** Panel (a) reports geometry after mean-pooling each example's K tokens; within-example cosine is computed before pooling. Panels (b) and (c) report Hit@1 differences with exact McNemar p-values in parentheses. Panel (b) uses the text-present REAL condition as reference; Panel (c) uses each model's NOTXT-REAL condition. Significance symbols follow Section 4.1. Results are single-seed and p-values are unadjusted; NLL and multi-seed uncertainty are unavailable.

(a) Geometry and unmodified performance

Method	Mean K	Hit@1	Inter cos.	RR	Norm ratio	Within cos.	Eff. rank
GRAG-PCST	1.000	74.570	0.999171	0.028223	42.935	–	10.300
TopK	7.848	71.437	1.000000	0.000155	28.258	1.000000	–
SAG	7.848	72.727	1.000000	0.000531	25.802	0.999999	–
DiffPool ($K = 8$)	8.000	52.703	0.999712	0.009284	25.477	0.999856	–
DiffPool ($K = 100$)	100.000	61.364	0.999994	0.001291	27.170	1.000000	–
GraSP-PCST	16.719	81.450	0.997783	0.039442	4.271	0.999465	–

(b) With-text interventions: ΔHit@1 (p) relative to REAL

Method	SWAP	W-SHUF	GSWAP	MEAN	NODE	EDGE
GRAG-PCST	-0.246 (.791)	–	$+0.061$ (1.000)	$+0.676$ (.379)	-0.061 (1.000)	-1.413 (.0616)†
TopK	$+0.430$ (.659)	–	-0.676 (.514)	-0.184 (.877)	-0.799 (.390)	$+1.044$ (.254)
SAG	-0.921 (.298)	–	-0.676 (.462)	$+0.553$ (.543)	-0.369 (.685)	$+0.614$ (.493)
DiffPool ($K = 8$)	-0.614 (.612)	–	$+0.614$ (.610)	-0.184 (.909)	$+0.676$ (.556)	$+2.088$ (.0525)†
DiffPool ($K = 100$)	-0.430 (.689)	–	$+0.184$ (.894)	$+0.246$ (.839)	-0.553 (.569)	$+0.430$ (.674)
GraSP-PCST	-0.553 (.460)	$+0.123$ (.930)	-0.676 (.379)	$+0.799$ (.263)	-0.430 (.586)	-0.123 (.924)

(c) Without-text interventions: REAL Hit@1 and ΔHit@1 (p) relative to NOTXT-REAL

Method	NOTXT-REAL	NOTXT-SWAP	NOTXT-GSWAP	NOTXT-MEAN
GRAG-PCST	41.585	-1.044 (.303)	$+0.184$ (.904)	-0.369 (.756)
TopK	28.194	-0.184 (.904)	$+0.860$ (.519)	-0.921 (.395)
SAG	31.511	$+0.307$ (.802)	$+0.369$ (.798)	-0.369 (.758)
DiffPool ($K = 8$)	27.150	$+2.334$ (.0530)†	$+0.491$ (.719)	$+0.921$ (.463)
DiffPool ($K = 100$)	22.789	-1.167 (.224)	-0.553 (.605)	-1.904 (.0472)*
GraSP-PCST	38.391	-0.123 (.963)	$+2.518$ (.0767)†	$+1.658$ (.234)

Table 11: Residual and norm ratios of the trained tokens; norm ratio is relative to the median word embedding.

Dataset	Model	Residual ratio	Norm ratio
ExplaGraphs	G-Retriever (GT)	0.050	21.9×
	G-Retriever (GAT)	0.040	24.4×
	G-Retriever (GCN)	0.121	24.3×
SceneGraphs	G-Retriever (GT)	0.088	84.8×
	G-Retriever (GAT)	0.046	87.6×
	G-Retriever (GCN)	0.048	97.0×
WebQSP	G-Retriever (GT)	0.057	29.4×
	+ LoRA	0.050	37.8×
	+ Qwen2.5-3B	0.047	22.8×

Table 12: Interventions on the GAT and GCN encoders, as differences from REAL in accuracy points.

Dataset	Model	REAL	MEAN	SWAP	NODE	EDGE	NOTXT-REAL
ExplaGraphs	G-Retriever (GAT)	85.20	−0.36	0.00	−0.18	+0.18	76.90
	G-Retriever (GCN)	78.88	+0.90	+0.36	+1.08	+0.72	69.86
SceneGraphs	G-Retriever (GAT)	80.20	+0.60	+0.20	−0.20	+0.10	28.65
	G-Retriever (GCN)	80.55	−0.10	−1.05	$-1.25^{\dagger}$	$-1.25^{\dagger}$	29.80

G THE TOKEN SPACE OF EVERY MODEL

The figures show the graph tokens of every test question. Their top rows project the mean-centred tokens onto two principal components and scale them by $\|\boldsymbol{\mu}\|$, making models with different token magnitudes comparable. The reported mean radius is measured in this two-dimensional projection and is therefore smaller than the full-dimensional residual ratio in Table **??**. Panels with radius below 0.1 include a magnified inset. The bottom rows show inter-sample cosine distributions on a fixed axis.

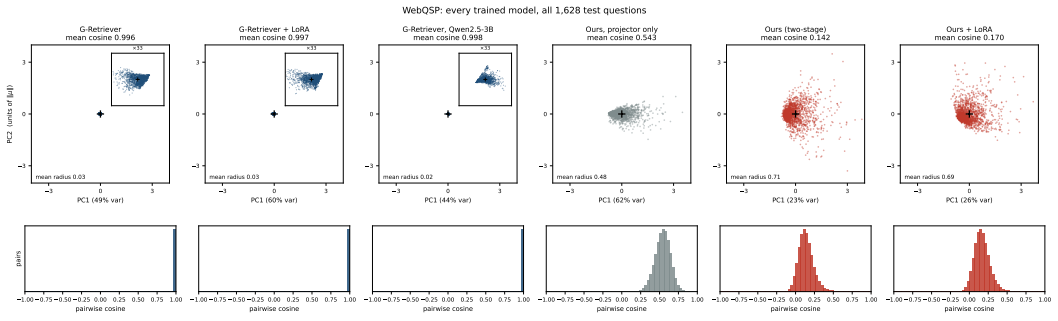


Figure 3: WebQSP, all 1,628 test questions, six models. G-Retriever 0.9963 cosine and 0.031 mean radius; with LoRA 0.9970 and 0.032; with Qwen2.5-3B 0.9975 and 0.022; our projector-only ablation 0.5425 and 0.479; our two-stage pipeline 0.1424 and 0.714; ours with LoRA 0.1702 and 0.686. Fine-tuning the LLM and changing the LLM family leave the token as collapsed as it was, and the projector-only ablation sits between the two regimes, which is the visual counterpart of the trade-off in Table 5.

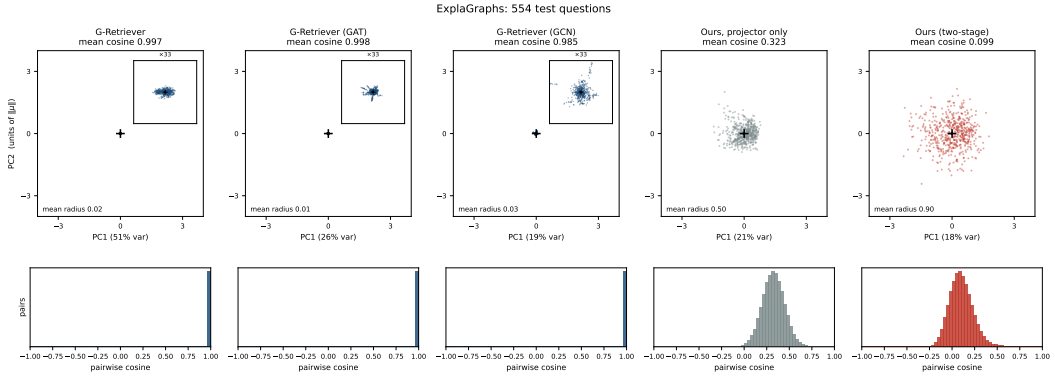


Figure 4: `ExplaGraphs`, all 554 test questions, five models: the three graph encoders of G-Retriever, with cosine 0.9974, 0.9983 and 0.9852 and mean radius 0.017, 0.012 and 0.026; our projector-only ablation, 0.3229 and 0.503; and our two-stage pipeline under the cross-fitted recipe, 0.0986 and 0.895. Collapse is the same for all three encoders.

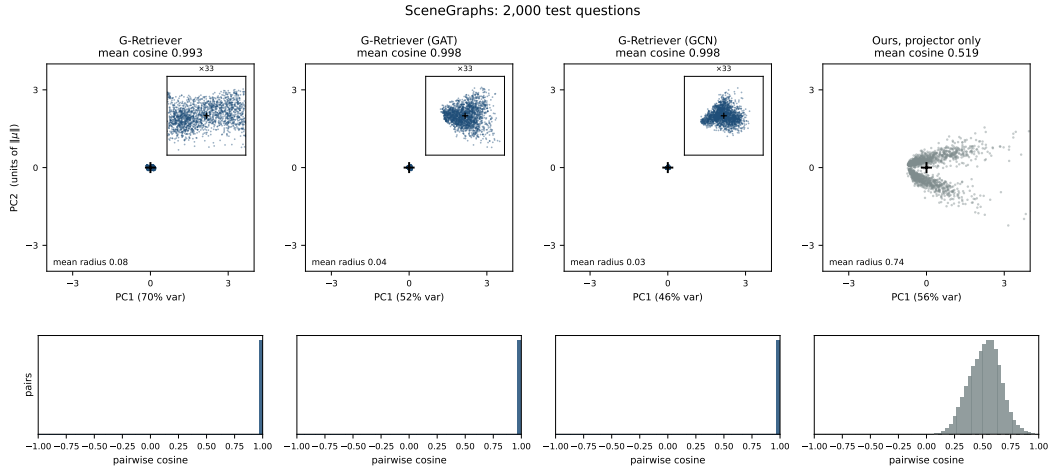


Figure 5: `SceneGraphs`, the first 2,000 test questions, four models: G-Retriever with a graph transformer (cosine 0.9934, mean radius 0.077), with GAT (0.9980, 0.040), with GCN (0.9979, 0.032), and our projector-only ablation (0.5194, 0.736).

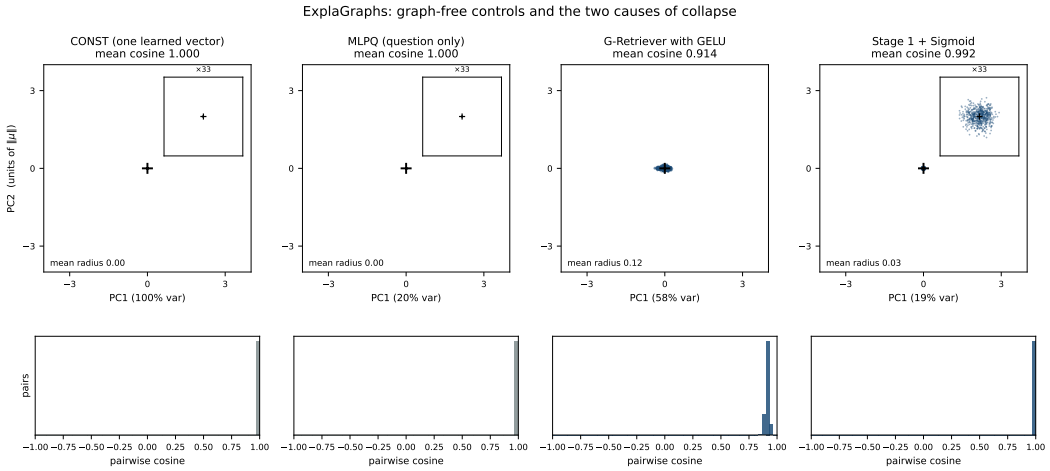


Figure 6: `ExplaGraphs`: the graph-free controls and the two causes of collapse. CONST (cosine 1.0000, radius 0.000) is a single point by construction and is the reference for a fully collapsed token; MLPQ (1.0000, 0.000) shows that a token computed from the question collapses just as completely; G-Retriever with GELU (0.9139, 0.116) shows that removing the saturating non-linearity alone is not enough while the encoder trains; Stage 1 with a Sigmoid projector (0.9918, 0.027) shows that an encoder whose outputs are nearly orthogonal still produces collapsed tokens. The last two panels are the visual form of Table 3.

H MECHANISM DIAGNOSTICS

Table 13 supports the first association in Section 4.4. An untrained projector, at its default initialisation, is applied to 500 random inputs of fixed norm, and the tokens it produces are almost identical to each other unless the input norm is two orders of magnitude larger than anything a trained encoder produces.

Table 13: Untrained projector (Equation 1, default initialisation) applied to 500 random inputs of fixed norm. Collapse precedes training and is a property of the parameterisation.

Input norm	Token norm	Token cosine	Residual ratio
1	18.4	0.9999	0.009
10	18.4	0.9919	0.090
100	21.4	0.7346	0.600

Why Sigmoid can create a shared direction. Let $\mathbf{x} = \mathbf{W}_1\mathbf{h}_i + \mathbf{b}_1$. If these preactivations are near zero, then $\sigma(\mathbf{x}) \approx \frac{1}{2}\mathbf{1} + \frac{1}{4}\mathbf{x}$, and the projector is approximately

$$P_\psi(\mathbf{h}_i) \approx \frac{1}{2}\mathbf{W}_2\mathbf{1} + \frac{1}{4}\mathbf{W}_2\mathbf{b}_1 + \mathbf{b}_2 + \frac{1}{4}\mathbf{W}_2\mathbf{W}_1\mathbf{h}_i.$$

The first three terms are input-independent, while the factor $\frac{1}{4}$ attenuates input-specific variation. This provides one possible explanation for the shared direction produced by the untrained projector. It is not a complete explanation: GraSP remains collapsed with a zero-centred Tanh projector (Appendix F), showing that Sigmoid is neither necessary nor the sole cause.

The approximation therefore motivates a mechanism consistent with Table 13, rather than establishing that the projector operates entirely in this local regime.

I FULL FIXED-WEIGHT INTERVENTIONS

The following tables report the available fixed-weight interventions for each model. Each table gives absolute accuracy, the difference from the reference condition, and the exact McNemar p-value. The reference is REAL, or NOTXT-REAL for conditions without the verbalised graph. The second and third SWAP rows, where present, use independent derangements. The reference is REAL, and NOTXT-REAL for the block without the verbalised graph. Conditions are defined in Section 3.2 and Table 8; the second and third SWAP rows are replicates under different derangements. Both stages of our pipeline are trained on all training questions unless a caption says cross-fitted. No effect below 2.0 accuracy points licenses a claim, whatever its p-value (Appendix D).

I.1 EXPLAGRAPHS

Table 14: **G-Retriever (graph transformer)**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	85.56	–	–
MEAN	85.38	-0.18	1
SWAP	84.30	-1.26	0.118
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	85.20	-0.36	0.727
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	84.48	-1.08	0.146
NODE (real rows)	84.66	-0.90	0.227
NODE (Gaussian)	85.20	-0.36	0.727
EDGE (rewired)	85.02	-0.54	0.581
REL (resampled)	85.74	$+0.18$	1
zero token	3.61	-81.95^{***}	1.09×10^{-131}
norm $\rightarrow$ word size	3.61	-81.95^{***}	3.17×10^{-130}
$\times 0.5$	77.44	-8.12^{***}	6.9×10^{-6}
$\times 2$	81.95	-3.61^{***}	0.000535
NOTXT-REAL	66.43	–	–
NOTXT-MEAN	65.52	-0.90	0.267
NOTXT-SWAP	66.43	$+0.00$	1
NOTXT-NODE	66.25	-0.18	1
NOTXT-zero	12.09	-54.33^{***}	2.46×10^{-66}
NOTXT-norm $\rightarrow$ word size	10.83	-55.60^{***}	7.08×10^{-70}

Table 15: **G-Retriever, initialisation 2**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	86.28	–	–
MEAN	84.84	-1.44	0.134
SWAP	85.02	-1.26	0.265
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	84.84	-1.44	0.134
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	83.57	-2.71^{***}	0.000275
NODE (real rows)	86.10	-0.18	1
NODE (Gaussian)	86.64	$+0.36$	0.754
EDGE (rewired)	84.84	-1.44	0.152
REL (resampled)	85.74	-0.54	0.453
zero token	3.25	-83.03^{***}	4.53×10^{-135}
norm $\rightarrow$ word size	3.43	-82.85^{***}	1.55×10^{-136}
$\times 0.5$	66.06	-20.22^{***}	1.26×10^{-18}
$\times 2$	82.13	-4.15^{*}	0.0451
NOTXT-REAL	71.84	–	–
NOTXT-MEAN	71.48	-0.36	0.912
NOTXT-SWAP	70.76	-1.08	0.61
NOTXT-NODE	73.10	$+1.26$	0.464
NOTXT-zero	11.37	-60.47^{***}	1.4×10^{-86}
NOTXT-norm $\rightarrow$ word size	11.01	-60.83^{***}	5.72×10^{-89}

Table 16: **G-Retriever, initialisation 3**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	87.91	–	–
MEAN	87.18	-0.72	0.219
SWAP	87.73	-0.18	1
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	87.73	-0.18	1
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	87.91	$+0.00$	1
NODE (real rows)	88.09	$+0.18$	1
NODE (Gaussian)	87.91	$+0.00$	1
EDGE (rewired)	87.18	-0.72	0.424
REL (resampled)	87.73	-0.18	1
zero token	3.25	-84.66^{***}	1.31×10^{-141}
norm $\rightarrow$ word size	4.51	-83.39^{***}	1.14×10^{-135}
$\times 0.5$	67.87	-20.04^{***}	3×10^{-20}
$\times 2$	85.56	$-2.35^{\dagger}$	0.066
NOTXT-REAL	70.22	–	–
NOTXT-MEAN	69.68	-0.54	0.648
NOTXT-SWAP	69.86	-0.36	0.851
NOTXT-NODE	71.30	$+1.08$	0.286
NOTXT-zero	11.37	-58.84^{***}	2.37×10^{-75}
NOTXT-norm $\rightarrow$ word size	11.73	-58.48^{***}	8.31×10^{-75}

Table 17: **G-Retriever, initialisation 4**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	89.71	–	–
MEAN	89.71	$+0.00$	1
SWAP	90.07	$+0.36$	0.625
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	89.71	$+0.00$	1
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	90.07	$+0.36$	0.5
NODE (real rows)	89.71	$+0.00$	1
NODE (Gaussian)	89.71	$+0.00$	1
EDGE (rewired)	90.07	$+0.36$	0.688
REL (resampled)	90.07	$+0.36$	0.625
zero token	2.17	-87.55^{***}	2×10^{-146}
norm $\rightarrow$ word size	3.25	-86.46^{***}	1.28×10^{-144}
$\times 0.5$	15.88	-73.83^{***}	4.29×10^{-113}
$\times 2$	82.31	-7.40^{***}	1.53×10^{-6}
NOTXT-REAL	83.39	–	–
NOTXT-MEAN	84.12	$+0.72$	0.672
NOTXT-SWAP	82.49	-0.90	0.603
NOTXT-NODE	81.95	-1.44	0.358
NOTXT-zero	12.45	-70.94^{***}	3×10^{-103}
NOTXT-norm $\rightarrow$ word size	13.00	-70.40^{***}	2.57×10^{-104}

Table 18: **G-Retriever (GAT)**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	85.20	–	–
MEAN	84.84	-0.36	0.754
SWAP	85.20	$+0.00$	1
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	85.02	-0.18	1
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	85.20	$+0.00$	1
NODE (real rows)	85.02	-0.18	1
NODE (Gaussian)	84.66	-0.54	0.508
EDGE (rewired)	85.38	$+0.18$	1
REL (resampled)	84.84	-0.36	0.5
zero token	3.61	-81.59^{***}	4.3×10^{-131}
norm $\rightarrow$ word size	3.07	-82.13^{***}	1.42×10^{-133}
$\times 0.5$	77.62	-7.58^{***}	3.89×10^{-5}
$\times 2$	83.21	-1.99	0.161
NOTXT-REAL	76.90	–	–
NOTXT-MEAN	76.90	$+0.00$	1
NOTXT-SWAP	76.35	-0.54	0.701
NOTXT-NODE	77.08	$+0.18$	1
NOTXT-zero	12.09	-64.80^{***}	2.39×10^{-97}
NOTXT-norm $\rightarrow$ word size	10.47	-66.43^{***}	5.65×10^{-100}

Table 19: **G-Retriever (GCN)**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	78.88	–	–
MEAN	79.78	$+0.90$	0.405
SWAP	79.24	$+0.36$	0.86
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	80.14	$+1.26$	0.23
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	77.44	-1.44	0.215
NODE (real rows)	79.96	$+1.08$	0.345
NODE (Gaussian)	78.88	$+0.00$	1
EDGE (rewired)	79.60	$+0.72$	0.503
REL (resampled)	79.42	$+0.54$	0.25
zero token	3.61	-75.27^{***}	1.16×10^{-120}
norm $\rightarrow$ word size	2.89	-75.99^{***}	7.49×10^{-122}
$\times 0.5$	40.61	-38.27^{***}	1.43×10^{-35}
$\times 2$	70.58	-8.30^{***}	0.00086
NOTXT-REAL	69.86	–	–
NOTXT-MEAN	69.49	-0.36	0.845
NOTXT-SWAP	69.31	-0.54	0.85
NOTXT-NODE	67.87	$-1.99^{\dagger}$	0.0895
NOTXT-zero	12.09	-57.76^{***}	1.22×10^{-80}
NOTXT-norm $\rightarrow$ word size	10.65	-59.21^{***}	1.86×10^{-85}

Table 20: **CONST**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	85.56	–	–
MEAN	85.02	-0.54	0.375
SWAP	85.56	$+0.00$	1
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	85.02	-0.54	0.375
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	84.84	-0.72	0.125
NODE (real rows)	85.56	$+0.00$	1
NODE (Gaussian)	85.56	$+0.00$	1
EDGE (rewired)	85.56	$+0.00$	1
REL (resampled)	85.56	$+0.00$	1
zero token	2.17	-83.39^{***}	1.14×10^{-135}
norm $\rightarrow$ word size	5.42	-80.14^{***}	6.84×10^{-126}
$\times 0.5$	61.73	-23.83^{***}	2.23×10^{-23}
$\times 2$	83.39	$-2.17^{\dagger}$	0.0884
NOTXT-REAL	51.99	–	–
NOTXT-MEAN	51.62	-0.36	0.5
NOTXT-SWAP	51.99	$+0.00$	1
NOTXT-NODE	51.99	$+0.00$	1
NOTXT-zero	12.27	-39.71^{***}	3.29×10^{-38}
NOTXT-norm $\rightarrow$ word size	11.19	-40.79^{***}	1.71×10^{-40}

Table 21: **CONST at word norm**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	85.56	–	–
MEAN	85.02	-0.54	0.25
SWAP	85.56	$+0.00$	1
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	85.02	-0.54	0.25
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	85.56	$+0.00$	1
NODE (real rows)	85.56	$+0.00$	1
NODE (Gaussian)	85.56	$+0.00$	1
EDGE (rewired)	85.56	$+0.00$	1
REL (resampled)	85.56	$+0.00$	1
zero token	2.17	-83.39^{***}	1.14×10^{-135}
norm $\rightarrow$ word size	85.56	$+0.00$	1
$\times 0.5$	4.87	-80.69^{***}	8.83×10^{-127}
$\times 2$	7.04	-78.52^{***}	3.17×10^{-123}
NOTXT-REAL	59.75	–	–
NOTXT-MEAN	57.22	-2.53	0.261
NOTXT-SWAP	59.75	$+0.00$	1
NOTXT-NODE	59.75	$+0.00$	1
NOTXT-zero	12.27	-47.47^{***}	1.57×10^{-51}
NOTXT-norm $\rightarrow$ word size	59.75	$+0.00$	1

Table 22: **MLPQ**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	84.84	–	–
MEAN	85.02	+0.18	1
SWAP	85.20	+0.36	0.688
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	85.02	+0.18	1
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	85.20	+0.36	0.5
NODE (real rows)	84.84	+0.00	1
NODE (Gaussian)	84.84	+0.00	1
EDGE (rewired)	84.84	+0.00	1
REL (resampled)	84.84	+0.00	1
zero token	3.61	-81.23^{***}	1.7×10^{-130}
norm $\rightarrow$ word size	3.97	-80.87^{***}	6.71×10^{-130}
$\times 0.5$	74.19	-10.65^{***}	1.26×10^{-7}
$\times 2$	84.12	−0.72	0.597
NOTXT-REAL	77.62	–	–
NOTXT-MEAN	77.62	+0.00	1
NOTXT-SWAP	77.80	+0.18	1
NOTXT-NODE	77.62	+0.00	1
NOTXT-zero	12.09	-65.52^{***}	3.19×10^{-91}
NOTXT-norm $\rightarrow$ word size	9.75	-67.87^{***}	2.61×10^{-102}

Table 23: **G-Retriever with GELU (no Stage 1)**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	73.29	–	–
SWAP	72.20	−1.08	0.631

Table 24: **Stage 1 + Sigmoid projector**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	84.48	–	–
MEAN	84.48	+0.00	1
SWAP	84.66	+0.18	1

Table 25: **Stage 1 + GELU, projector only**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	57.76	–	–
MEAN	58.84	$+1.08$	0.735
SWAP	51.08	-6.68^{*}	0.0167
NODE (real rows)	56.50	-1.26	0.685
NODE (Gaussian)	48.56	-9.21^{***}	0.000616
EDGE (rewired)	51.62	-6.14^{*}	0.02
REL (resampled)	55.96	-1.81	0.529
norm $\rightarrow$ word size	3.97	-53.79^{***}	1.32×10^{-79}
$\times 0.5$	33.03	-24.73^{***}	1.18×10^{-17}
$\times 2$	57.58	-0.18	1

Table 26: **Ours, joint (the method)**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	85.74	–	–
MEAN	86.10	+0.36	0.832
SWAP	85.20	−0.54	0.749
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	86.10	+0.36	0.832
NODE (real rows)	85.20	−0.54	0.648
NODE (Gaussian)	86.10	+0.36	0.824
EDGE (rewired)	86.82	+1.08	0.21
GSWAP (other graph)	84.30	−1.44	0.243
SWAP (permutation 2)	86.82	+1.08	0.362
SWAP (permutation 3)	85.02	−0.72	0.585
NOTXT-REAL	69.68	–	–
NOTXT-MEAN	69.86	+0.18	1
NOTXT-SWAP	69.86	+0.18	1
NOTXT-GSWAP	69.13	−0.54	0.771

Table 27: **Ours, joint, cross-fitted**, 554 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	76.35	–	–
MEAN	72.74	−3.61*	0.0265
SWAP	77.44	+1.08	0.545
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	72.92	−3.43*	0.037
NODE (real rows)	77.08	+0.72	0.636
NODE (Gaussian)	75.45	−0.90	0.511
EDGE (rewired)	76.53	+0.18	1
NOTXT-REAL	64.44	–	–
NOTXT-MEAN	61.37	−3.07	0.292
NOTXT-SWAP	64.62	+0.18	1

Table 28: **Ours, no entity features**, 554 test questions.

Condition	Accuracy	Difference	p
REAL	84.48	–	–
MEAN	87.55	+3.07**	0.00232
SWAP	84.48	+0.00	1
NODE (real rows)	84.66	+0.18	1
EDGE (rewired)	84.66	+0.18	1
GSWAP (other graph)	84.12	−0.36	0.754
NOTXT-REAL	70.22	–	–
NOTXT-MEAN	70.94	+0.72	0.834

I.2 SceneGraphs

Table 29: **G-Retriever (graph transformer)**, 2000 test questions.

Condition	Accuracy	Difference	p
REAL	81.35	–	–
MEAN	81.00	-0.35	0.614
SWAP	80.15	$-1.20^{\dagger}$	0.0566
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	80.30	-1.05	0.119
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	82.05	$+0.70$	0.338
NODE (real rows)	81.55	$+0.20$	0.699
NODE (Gaussian)	81.80	$+0.45$	0.272
EDGE (rewired)	81.55	$+0.20$	0.793
REL (resampled)	81.20	-0.15	0.876
zero token	11.30	-70.05^{***}	$< 10^{-300}$
norm $\rightarrow$ word size	11.35	-70.00^{***}	$< 10^{-300}$
$\times 0.5$	76.85	-4.50^{***}	1.53×10^{-7}
$\times 2$	80.55	-0.80	0.221
NOTXT-REAL	30.20	–	–
NOTXT-MEAN	30.65	$+0.45$	0.686
NOTXT-SWAP	28.80	-1.40	0.187
NOTXT-NODE	29.75	-0.45	0.684
NOTXT-zero	12.50	-17.70^{***}	3.5×10^{-45}
NOTXT-norm $\rightarrow$ word size	11.60	-18.60^{***}	2.78×10^{-52}

Table 30: **G-Retriever (GAT)**, 2000 test questions.

Condition	Accuracy	Difference	p
REAL	80.20	–	–
MEAN	80.80	$+0.60$	0.213
SWAP	80.40	$+0.20$	0.781
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	80.60	$+0.40$	0.434
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	80.15	-0.05	1
NODE (real rows)	80.00	-0.20	0.757
NODE (Gaussian)	80.50	$+0.30$	0.497
EDGE (rewired)	80.30	$+0.10$	0.856
REL (resampled)	80.35	$+0.15$	0.25
zero token	11.30	-68.90^{***}	$< 10^{-300}$
norm $\rightarrow$ word size	9.50	-70.70^{***}	$< 10^{-300}$
$\times 0.5$	78.90	$-1.30^{\dagger}$	0.0724
$\times 2$	78.55	-1.65^{*}	0.0266
NOTXT-REAL	28.65	–	–
NOTXT-MEAN	28.90	$+0.25$	0.817
NOTXT-SWAP	28.20	-0.45	0.638
NOTXT-NODE	29.80	$+1.15$	0.224
NOTXT-zero	12.50	-16.15^{***}	5.22×10^{-36}
NOTXT-norm $\rightarrow$ word size	10.95	-17.70^{***}	3.5×10^{-45}

Table 31: **G-Retriever (GCN)**, 2000 test questions.

Condition	Accuracy	Difference	p
REAL	80.55	–	–
MEAN	80.45	-0.10	0.921
SWAP	79.50	-1.05	0.137
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	80.35	-0.20	0.752
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	79.00	-1.55^{*}	0.0178
NODE (real rows)	79.30	$-1.25^{\dagger}$	0.0725
NODE (Gaussian)	79.60	-0.95	0.181
EDGE (rewired)	79.30	$-1.25^{\dagger}$	0.0661
REL (resampled)	80.55	$+0.00$	1
zero token	11.30	-69.25^{***}	$< 10^{-300}$
norm $\rightarrow$ word size	11.80	-68.75^{***}	$< 10^{-300}$
$\times 0.5$	78.25	-2.30^{**}	0.00388
$\times 2$	78.90	-1.65^{*}	0.019
NOTXT-REAL	29.80	–	–
NOTXT-MEAN	29.55	-0.25	0.857
NOTXT-SWAP	28.85	-0.95	0.42
NOTXT-NODE	30.90	$+1.10$	0.334
NOTXT-zero	12.50	-17.30^{***}	2.73×10^{-46}
NOTXT-norm $\rightarrow$ word size	12.40	-17.40^{***}	5.23×10^{-47}

Table 32: **CONST**, 2000 test questions.

Condition	Accuracy	Difference	p
REAL	81.40	–	–
MEAN	81.25	-0.15	0.25
SWAP	81.40	$+0.00$	1
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	81.25	-0.15	0.25
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	80.95	-0.45^{**}	0.00391
NODE (real rows)	81.40	$+0.00$	1
NODE (Gaussian)	81.40	$+0.00$	1
EDGE (rewired)	81.40	$+0.00$	1
REL (resampled)	81.40	$+0.00$	1
zero token	12.20	-69.20^{***}	$< 10^{-300}$
norm $\rightarrow$ word size	14.80	-66.60^{***}	$< 10^{-300}$
$\times 0.5$	74.50	-6.90^{***}	2.29×10^{-16}
$\times 2$	79.45	-1.95^{**}	0.00483
NOTXT-REAL	32.55	–	–
NOTXT-MEAN	32.40	-0.15	0.755
NOTXT-SWAP	32.55	$+0.00$	1
NOTXT-NODE	32.55	$+0.00$	1
NOTXT-zero	13.80	-18.75^{***}	1.23×10^{-53}
NOTXT-norm $\rightarrow$ word size	12.35	-20.20^{***}	2.46×10^{-63}

Table 33: **Stage 1 + GELU, projector only**, 2000 test questions.

Condition	Accuracy	Difference	p
REAL	69.15	–	–
MEAN	48.95	-20.20^{***}	6.8×10^{-48}
SWAP	34.10	-35.05^{***}	1.11×10^{-127}
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	49.05	-20.10^{***}	2.05×10^{-47}
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	38.85	-30.30^{***}	4.36×10^{-116}
NODE (real rows)	60.25	-8.90^{***}	7.43×10^{-15}
NODE (Gaussian)	60.05	-9.10^{***}	2.34×10^{-16}
EDGE (rewired)	66.70	-2.45^{*}	0.0159
REL (resampled)	68.30	-0.85	0.244
zero token	11.30	-57.85^{***}	6.61×10^{-293}
norm $\rightarrow$ word size	15.50	-53.65^{***}	1.13×10^{-262}
$\times 0.5$	54.70	-14.45^{***}	1.12×10^{-38}
$\times 2$	64.65	-4.50^{***}	2.06×10^{-7}
NOTXT-REAL	46.60	–	–
NOTXT-MEAN	22.95	-23.65^{***}	7.41×10^{-69}
NOTXT-SWAP	22.45	-24.15^{***}	1.66×10^{-78}
NOTXT-NODE	38.90	-7.70^{***}	1.31×10^{-13}
NOTXT-zero	12.50	-34.10^{***}	1.03×10^{-146}
NOTXT-norm $\rightarrow$ word size	12.80	-33.80^{***}	1.32×10^{-143}

Table 34: **Ours, joint, standard recipe**, 2,000 `SceneGraphs` test questions. The reference is REAL above the midrule and NOTXT-REAL below it.

Condition	Accuracy	Difference	p
REAL	72.50	–	–
MEAN	71.10	-1.40	.182
SWAP	69.10	-3.40^{***}	6.31×10^{-4}
SWAP (permutation 2)	67.85	-4.65^{***}	2.32×10^{-6}
SWAP (permutation 3)	67.85	-4.65^{***}	1.21×10^{-5}
$\boldsymbol{\mu} + 0(\mathbf{t}_i - \boldsymbol{\mu})$	70.50	-2.00^{*}	.0481
NODE (real rows)	67.40	-5.10^{***}	1.33×10^{-10}
NODE (Gaussian)	67.40	-5.10^{***}	1.57×10^{-11}
EDGE (rewired)	71.45	-1.05	.186
GSWAP (other graph)	67.20	-5.30^{***}	3.55×10^{-9}
NOTXT-REAL	32.60	–	–
NOTXT-MEAN	26.00	-6.60^{***}	3.57×10^{-8}
NOTXT-SWAP	21.75	-10.85^{***}	8.93×10^{-19}
NOTXT-GSWAP	29.50	-3.10^{**}	.00600

Table 35: **G-Retriever, half split**, 2000 test questions.

Condition	Accuracy	Difference	p
REAL	77.40	–	–
MEAN	77.30	-0.10	0.918
SWAP	77.20	-0.20	0.801
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	77.35	-0.05	1
NODE (real rows)	77.50	$+0.10$	0.856
NODE (Gaussian)	77.35	-0.05	1
EDGE (rewired)	77.20	-0.20	0.775
NOTXT-REAL	25.50	–	–
NOTXT-MEAN	24.90	-0.60	0.5
NOTXT-SWAP	24.40	-1.10	0.199

Table 36: **Ours, joint, cross-fitted, initialisation 1**, 2000 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	76.90	–	–
MEAN	75.40	$-1.50^{\dagger}$	0.0829
SWAP	72.45	-4.45^{***}	2.02×10^{-7}
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	74.95	-1.95^{*}	0.0242
NODE (real rows)	75.85	-1.05	0.158
NODE (Gaussian)	76.10	-0.80	0.323
EDGE (rewired)	74.95	-1.95^{**}	0.00202
NOTXT-REAL	26.20	–	–
NOTXT-MEAN	25.45	-0.75	0.474
NOTXT-SWAP	23.60	-2.60^{*}	0.0123

Table 37: **Ours, joint, cross-fitted, initialisation 2**, 2000 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	77.55	–	–
MEAN	76.45	-1.10	0.191
SWAP	75.10	-2.45^{**}	0.00146
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	76.50	-1.05	0.214
NODE (real rows)	78.55	$+1.00$	0.116
NODE (Gaussian)	78.15	$+0.60$	0.366
EDGE (rewired)	77.85	$+0.30$	0.734
NOTXT-REAL	31.25	–	–
NOTXT-MEAN	33.00	$+1.75$	0.13
NOTXT-SWAP	29.55	-1.70	0.136

Table 38: **Ours, no entity features**, 2000 test questions. Best epoch 3 of 5, with validation loss 0.326, 0.287, 0.283, 0.2826, 0.302 over epochs 0 to 4.

Condition	Accuracy	Difference	p
REAL	79.65	–	–
MEAN	73.90	-5.75^{***}	2.1×10^{-11}
SWAP	70.05	-9.60^{***}	1.2×10^{-23}
NODE (real rows)	79.75	$+0.10$	0.73
EDGE (rewired)	80.05	$+0.40$	0.56
GSWAP (other graph)	80.00	$+0.35$	0.54
NOTXT-REAL	30.10	–	–
NOTXT-MEAN	17.25	-12.85^{***}	2.0×10^{-35}

SWAP on our standard-recipe `SceneGraphs` model is not an artefact of one pairing: two further derangements give -4.65 ($p = 2.3 \times 10^{-6}$) and -4.65 ($p = 1.2 \times 10^{-5}$) on the same 2,000 questions, against the -3.40 reported in Table 4, which is the weakest of the three.

I.3 WEBQSP

Table 39: **G-Retriever (graph transformer)**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	70.33	–	–
MEAN	68.49	-1.84^{*}	0.0285
SWAP	69.41	-0.92	0.311
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	70.70	$+0.37$	0.687
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	70.95	$+0.61$	0.495
NODE (real rows)	70.39	$+0.06$	1
NODE (Gaussian)	70.21	-0.12	0.933
EDGE (rewired)	71.01	$+0.68$	0.46
REL (resampled)	70.82	$+0.49$	0.575
GSWAP (other graph)	69.78	-0.55	0.569
SWAP (permutation 2)	69.96	-0.37	0.711
SWAP (permutation 3)	70.15	-0.18	0.874
zero token	10.69	-59.64^{***}	1.59×10^{-266}
norm $\rightarrow$ word size	4.85	-65.48^{***}	$< 10^{-300}$
$\times 0.5$	63.02	-7.31^{***}	1.9×10^{-12}
$\times 2$	66.09	-4.24^{***}	7.26×10^{-6}
NOTXT-REAL	41.34	–	–
NOTXT-MEAN	42.38	$+1.04$	0.187
NOTXT-SWAP	41.15	-0.18	0.886
NOTXT-NODE	41.28	-0.06	1
NOTXT-GSWAP	40.48	-0.86	0.356
NOTXT-zero	4.05	-37.29^{***}	1.79×10^{-157}
NOTXT-norm $\rightarrow$ word size	2.76	-38.57^{***}	1.56×10^{-170}

Table 40: **G-Retriever + LoRA**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	74.32	–	–
MEAN	73.89	-0.43	0.634
SWAP	73.59	-0.74	0.382
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	73.89	-0.43	0.636
NODE (real rows)	73.34	-0.98	0.224
NODE (Gaussian)	73.59	-0.74	0.359
EDGE (rewired)	74.20	-0.12	0.935
NOTXT-REAL	10.38	–	–
NOTXT-MEAN	11.49	$+1.11$	0.215
NOTXT-SWAP	11.43	$+1.04$	0.221

Table 41: **G-Retriever with Qwen2.5-3B**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	55.47	–	–
MEAN	54.98	-0.49	0.48
SWAP	54.79	-0.68	0.428
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	54.98	-0.49	0.48
NODE (real rows)	54.85	-0.61	0.378
NODE (Gaussian)	54.67	-0.80	0.232
EDGE (rewired)	55.41	-0.06	1
NOTXT-REAL	34.40	–	–
NOTXT-MEAN	30.65	-3.75^{***}	2.45×10^{-5}
NOTXT-SWAP	32.31	-2.09^{*}	0.0286

Table 42: **Stage 1 + GELU, projector only**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	64.62	–	–
MEAN	49.88	-14.74^{***}	1.26×10^{-30}
SWAP	54.73	-9.89^{***}	4.62×10^{-15}
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	49.88	-14.74^{***}	1.26×10^{-30}
$\boldsymbol{\mu} + 2(\mathbf{t}_i - \boldsymbol{\mu})$	37.65	-26.97^{***}	8.63×10^{-82}
NODE (real rows)	62.22	-2.40^{*}	0.0289
NODE (Gaussian)	63.08	-1.54	0.173
EDGE (rewired)	64.68	$+0.06$	1
REL (resampled)	63.82	-0.80	0.424
zero token	10.69	-53.93^{***}	1.84×10^{-224}
norm $\rightarrow$ word size	14.37	-50.25^{***}	3.14×10^{-203}
$\times 0.5$	54.05	-10.57^{***}	1.04×10^{-18}
$\times 2$	54.30	-10.32^{***}	4.88×10^{-20}
NOTXT-REAL	36.30	–	–
NOTXT-MEAN	16.15	-20.15^{***}	1.76×10^{-53}
NOTXT-SWAP	28.01	-8.29^{***}	8.71×10^{-12}
NOTXT-NODE	34.28	$-2.03^{\dagger}$	0.0911
NOTXT-zero	4.05	-32.25^{***}	1.22×10^{-136}
NOTXT-norm $\rightarrow$ word size	4.12	-32.19^{***}	3.3×10^{-139}

Table 43: **Ours, joint (the method)**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	69.90	–	–
MEAN	66.77	-3.13^{***}	0.000865
SWAP	67.75	-2.15^{*}	0.0387
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	67.38	-2.52^{**}	0.00807
NODE (real rows)	68.67	-1.23	0.216
NODE (Gaussian)	68.55	-1.35	0.157
EDGE (rewired)	69.10	-0.80	0.415
GSWAP (other graph)	67.94	$-1.97^{\dagger}$	0.0515
SWAP (permutation 2)	67.75	-2.15^{*}	0.0344
SWAP (permutation 3)	66.46	-3.44^{***}	0.000785
NOTXT-REAL	28.75	–	–
NOTXT-MEAN	26.66	-2.09^{*}	0.0452
NOTXT-SWAP	26.35	-2.40^{*}	0.0283
NOTXT-GSWAP	26.47	-2.27^{*}	0.0372

Table 44: **Ours, joint + LoRA**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	74.57	–	–
MEAN	73.46	-1.11	0.195
SWAP	72.24	-2.33^{**}	0.00839
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	72.60	-1.97^{*}	0.0192
NODE (real rows)	73.77	-0.80	0.332
NODE (Gaussian)	73.40	-1.17	0.161
EDGE (rewired)	74.82	$+0.25$	0.811
NOTXT-REAL	47.91	–	–
NOTXT-MEAN	49.20	$+1.29$	0.239
NOTXT-SWAP	44.72	-3.19^{**}	0.00518

Table 45: **CONST**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	71.25	–	–
MEAN	71.44	+0.18	0.69
SWAP	71.25	+0.00	1
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	71.44	+0.18	0.69
NODE (real rows)	71.25	+0.00	1
NODE (Gaussian)	71.25	+0.00	1
EDGE (rewired)	71.25	+0.00	1
NOTXT-REAL	42.57	–	–
NOTXT-MEAN	42.14	−0.43	0.189
NOTXT-SWAP	42.57	+0.00	1

Table 46: **MLPQ**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	70.70	–	–
MEAN	70.21	−0.49	0.115
SWAP	68.98	-1.72^{*}	0.015
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	70.21	−0.49	0.115
NODE (real rows)	70.70	+0.00	1
NODE (Gaussian)	70.70	+0.00	1
EDGE (rewired)	70.70	+0.00	1
NOTXT-REAL	43.37	–	–
NOTXT-MEAN	43.61	+0.25	0.608
NOTXT-SWAP	44.04	$+0.68^{\dagger}$	0.0801

Table 47: **Ours, joint, Qwen2.5-3B**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	65.79	–	–
MEAN	63.70	-2.09^{**}	0.00187
SWAP	63.88	-1.90^{**}	0.00573
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	63.70	-2.09^{**}	0.00187
NODE (real rows)	64.99	−0.80	0.25
NODE (Gaussian)	65.17	−0.61	0.373
EDGE (rewired)	64.43	-1.35^{*}	0.0315
NOTXT-REAL	34.34	–	–
NOTXT-MEAN	34.64	+0.31	0.768
NOTXT-SWAP	35.44	+1.11	0.217

Table 48: **G-Retriever, Qwen2.5-3B, matched budget**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	57.13	–	–
MEAN	56.57	-0.55	0.478
SWAP	55.71	-1.41	0.104
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	56.70	-0.43	0.595
NODE (real rows)	56.82	-0.31	0.704
NODE (Gaussian)	56.57	-0.55	0.452
EDGE (rewired)	57.19	$+0.06$	1
NOTXT-REAL	29.30	–	–
NOTXT-MEAN	30.16	$+0.86$	0.403
NOTXT-SWAP	27.27	$-2.03^{\dagger}$	0.0508

Table 49: **G-Retriever, half split, initialisation 1**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	70.58	–	–
MEAN	70.02	−0.55	0.538
SWAP	70.15	−0.43	0.654
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	69.90	−0.68	0.46
NODE (real rows)	70.33	−0.25	0.784
NODE (Gaussian)	70.58	+0.00	1
EDGE (rewired)	70.64	+0.06	1
NOTXT-REAL	42.14	–	–
NOTXT-MEAN	43.06	+0.92	0.115
NOTXT-SWAP	41.77	−0.37	0.637

Table 50: **G-Retriever, half split, initialisation 2**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	67.75	–	–
MEAN	67.69	−0.06	1
SWAP	67.26	−0.49	0.614
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	67.57	−0.18	0.89
NODE (real rows)	67.26	−0.49	0.598
NODE (Gaussian)	67.63	−0.12	0.942
EDGE (rewired)	67.20	−0.55	0.573
NOTXT-REAL	37.96	–	–
NOTXT-MEAN	38.39	+0.43	0.562
NOTXT-SWAP	37.29	−0.68	0.4

Table 51: **G-Retriever, half split, initialisation 3**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	70.02	–	–
MEAN	70.02	+0.00	1
SWAP	69.78	−0.25	0.836
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	68.86	−1.17	0.22
NODE (real rows)	69.16	−0.86	0.356
NODE (Gaussian)	70.52	+0.49	0.617
EDGE (rewired)	69.66	−0.37	0.718
NOTXT-REAL	34.09	–	–
NOTXT-MEAN	34.89	+0.80	0.275
NOTXT-SWAP	35.01	+0.92	0.248

Table 52: **Projector only, cross-fitted**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	63.14	–	–
MEAN	59.71	-3.44^{**}	0.00341
SWAP	57.80	-5.34^{***}	6.9×10^{-6}
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	60.01	-3.13^{**}	0.0045
NODE (real rows)	61.49	-1.66	0.127
NODE (Gaussian)	62.16	-0.98	0.385
EDGE (rewired)	64.19	$+1.04$	0.32
NOTXT-REAL	36.61	–	–
NOTXT-MEAN	41.58	$+4.98^{***}$	1.71×10^{-5}
NOTXT-SWAP	33.35	-3.26^{**}	0.00517

Table 53: **Late-prompt variant, cross-fitted**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	63.64	–	–
MEAN	60.69	-2.95^{**}	0.0075
SWAP	58.42	-5.22^{***}	1.41×10^{-6}
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	59.28	-4.36^{***}	7.12×10^{-5}
NODE (real rows)	60.87	-2.76^{**}	0.00953
NODE (Gaussian)	62.59	-1.04	0.324
EDGE (rewired)	62.59	-1.04	0.327
NOTXT-REAL	38.33	–	–
NOTXT-MEAN	40.36	$+2.03^{\dagger}$	0.0812
NOTXT-SWAP	34.77	-3.56^{**}	0.0015

Table 54: **Projector dropout 0.5, cross-fitted**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	64.56	–	–
MEAN	60.32	-4.24^{***}	5.2×10^{-5}
SWAP	60.87	-3.69^{***}	0.000608
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	61.86	-2.70^{**}	0.00801
NODE (real rows)	63.27	-1.29	0.216
NODE (Gaussian)	63.21	-1.35	0.196
EDGE (rewired)	63.45	-1.11	0.262
NOTXT-REAL	41.83	–	–
NOTXT-MEAN	43.92	$+2.09^{*}$	0.0413
NOTXT-SWAP	38.88	-2.95^{**}	0.00638

Table 55: **Text-dropout variant, cross-fitted**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	67.20	–	–
MEAN	68.55	$+1.35$	0.143
SWAP	66.89	-0.31	0.794
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	68.92	$+1.72^{\dagger}$	0.0511
NODE (real rows)	69.23	$+2.03^{*}$	0.0281
NODE (Gaussian)	68.49	$+1.29$	0.172
EDGE (rewired)	69.10	$+1.90^{*}$	0.035
NOTXT-REAL	26.97	–	–
NOTXT-MEAN	31.27	$+4.30^{***}$	0.00043
NOTXT-SWAP	25.12	-1.84	0.132

Table 56: **Ours, joint, cross-fitted, initialisation 1**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	69.53	–	–
MEAN	68.00	$-1.54^{\dagger}$	0.0967
SWAP	66.34	-3.19^{**}	0.00151
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	67.87	$-1.66^{\dagger}$	0.0732
NODE (real rows)	66.28	-3.26^{***}	0.000443
NODE (Gaussian)	68.24	-1.29	0.174
EDGE (rewired)	68.98	-0.55	0.567
GSWAP (other graph)	66.71	-2.83^{**}	0.00499
SWAP (permutation 2)	67.63	$-1.90^{\dagger}$	0.0581
SWAP (permutation 3)	66.46	-3.07^{**}	0.0018
NOTXT-REAL	19.41	–	–
NOTXT-MEAN	21.99	$+2.58^{*}$	0.0252
NOTXT-SWAP	17.87	-1.54	0.179
NOTXT-GSWAP	19.04	-0.37	0.776

Table 57: **Ours, joint, cross-fitted, initialisation 2**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	66.89	–	–
MEAN	65.11	$-1.78^{\dagger}$	0.0888
SWAP	65.85	-1.04	0.322
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	63.70	-3.19^{**}	0.00139
NODE (real rows)	64.68	-2.21^{*}	0.0298
NODE (Gaussian)	65.29	-1.60	0.101
EDGE (rewired)	65.36	-1.54	0.12
NOTXT-REAL	22.91	–	–
NOTXT-MEAN	9.34	-13.57^{***}	4.3×10^{-35}
NOTXT-SWAP	21.13	-1.78	0.115

Table 58: **Ours, joint, cross-fitted, initialisation 3**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	68.43	–	–
MEAN	68.18	-0.25	0.848
SWAP	68.30	-0.12	0.948
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	68.73	$+0.31$	0.797
NODE (real rows)	68.37	-0.06	1
NODE (Gaussian)	68.80	$+0.37$	0.737
EDGE (rewired)	68.92	$+0.49$	0.598
NOTXT-REAL	28.01	–	–
NOTXT-MEAN	24.69	-3.32^{***}	0.000237
NOTXT-SWAP	25.86	-2.15^{*}	0.0163

Table 59: **Ours, joint + LoRA, cross-fitted**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	70.95	–	–
MEAN	69.78	-1.17	0.166
SWAP	70.02	-0.92	0.279
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	69.53	$-1.41^{\dagger}$	0.0845
NODE (real rows)	69.47	$-1.47^{\dagger}$	0.0669
NODE (Gaussian)	69.59	$-1.35^{\dagger}$	0.0945
EDGE (rewired)	69.29	-1.66^{*}	0.0328
NOTXT-REAL	41.09	–	–
NOTXT-MEAN	41.34	$+0.25$	0.845
NOTXT-SWAP	39.62	-1.47	0.161

Table 60: **G-Retriever + LoRA, cross-fitted**, 1628 test questions, cross-fitted recipe.

Condition	Accuracy	Difference	p
REAL	72.17	–	–
MEAN	71.93	-0.25	0.805
SWAP	71.99	-0.18	0.859
$\boldsymbol{\mu} + 0 \cdot (\mathbf{t}_i - \boldsymbol{\mu})$	70.82	$-1.35^{\dagger}$	0.0882
NODE (real rows)	71.87	-0.31	0.738
NODE (Gaussian)	71.56	-0.61	0.456
EDGE (rewired)	71.19	-0.98	0.221
NOTXT-REAL	20.64	–	–
NOTXT-MEAN	21.99	$+1.35^{*}$	0.0448
NOTXT-SWAP	21.74	$+1.11$	0.181

Table 61: **Ours, no entity features**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	72.73	–	–
MEAN	66.58	-6.14^{***}	1.28×10^{-9}
SWAP	67.08	-5.65^{***}	5.88×10^{-9}
NODE (real rows)	72.11	-0.61	0.415
EDGE (rewired)	72.30	-0.43	0.664
GSWAP (other graph)	70.76	-1.97^{*}	0.0347
NOTXT-REAL	28.38	–	–
NOTXT-MEAN	20.88	-7.49^{***}	4.94×10^{-11}

Table 62: **Ours, no question conditioning**, 1628 test questions.

Condition	Accuracy	Difference	p
REAL	69.84	–	–
MEAN	69.78	-0.06	1
SWAP	66.15	-3.69^{***}	0.000365
NODE (real rows)	67.75	-2.09^{*}	0.0367
EDGE (rewired)	69.35	-0.49	0.636
GSWAP (other graph)	67.51	-2.33^{*}	0.0247
NOTXT-REAL	30.90	–	–
NOTXT-MEAN	31.33	$+0.43$	0.742

J THE TOKEN'S FORM: MAGNITUDE AND RESIDUAL CONDITIONS

These conditions probe the token's geometry rather than which factor it carries, which is why they are not part of Table 8. Table 63 reports them for the three graph-transformer models and the two graph-free controls on ExplaGraphs; the same conditions for every other model are in Appendix I.

Table 63: Conditions that keep the token's direction and change its length, as differences from REAL. † $p < .10$, * $p < .05$, ** $p < .01$, *** $p < .001$, exact McNemar, unmarked $p \geq .10$. Blank cells were not run. ExplaGraphs 554 questions, SceneGraphs the first 2,000, WebQSP 1,628.

Dataset	Model	REAL	zero token	norm → word	×0.5	×2
ExplaGraphs	G-Retriever (GT)	85.56	-81.95^{***}	-81.95^{***}	-8.12^{***}	-3.61^{***}
	CONST	85.56	-83.39^{***}	-80.14^{***}	-23.83^{***}	$-2.17^{\dagger}$
	CONST at word norm	85.56	-83.39^{***}	0.00	-80.69^{***}	-78.52^{***}
SceneGraphs	G-Retriever (GT)	81.35	-70.05^{***}	-70.00^{***}	-4.50^{***}	-0.80
	CONST	81.40	-69.20^{***}	-66.60^{***}	-6.90^{***}	-1.95^{**}
WebQSP	G-Retriever (GT)	70.33	-59.64^{***}	-65.48^{***}	-7.31^{***}	-4.24^{***}

A trained model depends on the length its token had during training, in both directions, and a large token is not necessary: the constant held at word norm throughout training reaches the same accuracy as the others and then breaks when its length is changed afterwards.

K COMPLETE GRAPH-FREE CONTROLS

Table 64 gives the controls in full. All four models share G-Retriever's prompt, answer loss, optimiser and training schedule. CONST and MLPQ also share its projector, so they differ from G-Retriever only in the token's source; PT differs in both the source and the parameterisation, since its vectors are optimised directly in the embedding space, and its capacity differs by more than two orders of magnitude. The CONST–PT gap therefore remains confounded between parameterisation and capacity, which is why the main text does not use it as a graph-versus-no-graph comparison.

Table 64: Graph-free controls. Score is accuracy on ExplaGraphs and SceneGraphs and Hit@1 on WebQSP; parameter counts are from the WebQSP checkpoints with the LLM frozen and its embedding table excluded. Dashes are configurations we did not train.

Model	Trainable parameters	ExplaGraphs	SceneGraphs	WebQSP
G-Retriever	31,492,099	85.56	81.35	70.33
CONST	10,492,928	85.56	81.40	71.25
MLPQ	14,690,304	84.84	–	70.70
PT	40,960	60.60	70.22	52.21
ours, Stage 2	20,990,979	85.74	72.50	69.90

Accuracy does not track capacity: PT to CONST multiplies the trainable parameters by 256 and adds 19 Hit@1 on WebQSP, while CONST to our Stage 2 multiplies them by two and removes 1.35.

L GOLD-ANSWER LIKELIHOOD

Table 65 reports the measurement defined in Equation 4. It is a second reading of conditions that were already run, not a new experiment.

Three readings. In the model whose token can only carry the graph, swapping the graph and swapping the token cost the same likelihood ($+0.0845$ against $+0.0848$), which is the internal consistency check for the instrument. On SceneGraphs the graph swap is seven times the token swap,

Table 65: Paired gold-answer NLL differences, condition minus REAL, with 95% bootstrap intervals; positive means the intervention made the gold answer less likely. Every cell whose p-value we measured is significant under a Wilcoxon signed-rank test (largest $p = 10^{-4}$), so the criterion we apply is the effect size: the same weights and condition rerun give $+0.0003$ $[-0.0003, +0.0008]$, and we require 0.03, a hundred times that floor, before reading a likelihood effect. The Qwen2.5-3B rows are the matched-budget pair of Table 4.

Dataset	Model	REAL	MEAN	SWAP	GSWAP
WebQSP	G-Retriever	0.7558	$+0.0131$ $[+.0097, +.0166]$	$+0.0183$ $[+.0137, +.0229]$	$+0.0182$ $[+.0136, +.0228]$
	ours	0.8540	-0.0159 $[-.0288, -.0041]$	$+0.0895$ $[+.0727, +.1074]$	$+0.0332$ $[+.0185, +.0461]$
	ours, no entity features	0.8213	$+0.0378$ $[+.0257, +.0506]$	$+0.1111$ $[+.0954, +.1261]$	$+0.0262$ $[+.0175, +.0348]$
	ours, no question	0.8517	$+0.0070$ $[-.0071, +.0203]$	$+0.0848$ $[+.0671, +.1020]$	$+0.0845$ $[+.0668, +.1015]$
	G-Retriever, Qwen2.5-3B	1.0367	$+0.0109$ $[+.0038, +.0185]$	$+0.0734$ $[+.0623, +.0845]$	$+0.0737$ $[+.0626, +.0850]$
	ours, Qwen2.5-3B	1.0553	$+0.0123$ $[+.0066, +.0177]$	$+0.0168$ $[+.0106, +.0226]$	$+0.0144$ $[+.0076, +.0211]$
SceneGraphs	G-Retriever	0.2477	$+0.0053$ $[+.0006, +.0100]$	$+0.0169$ $[+.0102, +.0240]$	$+0.0167$ $[+.0101, +.0238]$
	ours	0.4122	-0.0287 $[-.0556, -.0027]$	$+0.0229$ $[-.0041, +.0498]$	$+0.1665$ $[+.1353, +.1999]$
ExplaGraphs	G-Retriever	0.2068	$+0.0006$ $[-.0048, +.0059]$	$+0.0014$ $[-.0063, +.0088]$	$+0.0022$ $[-.0051, +.0095]$
	ours	0.1696	-0.0047 $[-.0160, +.0052]$	$+0.0010$ $[-.0120, +.0130]$	$+0.0032$ $[-.0091, +.0152]$

the largest effect measured anywhere in this study and the distribution-level counterpart of the -5.30 accuracy drop, while the mean token *lowers* the likelihood: an empty token is better than a wrong graph. On ExplaGraphs nothing moves for either model, which settles that dataset at the distribution level and not merely below the accuracy resolution.

M COMPLETE TOKEN-CONTENT AND USE RESULTS

Table 66 extends the Llama-2 results in Table 5 with the complete Qwen2.5-3B replication and diagnostic variants. The probe is trained on half of the test tokens and evaluated on the other half by retrieving the corresponding question, answer, or verbalised graph. The candidate sets contain 814 targets on WebQSP, 1,000 on SceneGraphs, and 277 on ExplaGraphs. A constant token therefore has median rank 407, 500, and 139, respectively. Repeated SceneGraphs answers place the answer control between 476 and 500, depending on the held-out split.

The control rows validate the probe: MLPQ on WebQSP and CONST on SceneGraphs remain at the corresponding median control ranks. The cross-fitted WebQSP model was not probed and is therefore omitted.

The Qwen replication reinforces the distinction between recoverability and use. On SceneGraphs, our model produces the largest graph-replacement effect in the study (-6.50), although its graph rank is close to the constant control (450 versus 500). On ExplaGraphs, the graph is substantially more recoverable (62 versus 139), yet replacing it has no resolved effect. A linear probe can therefore fail to recover information that affects the LLM or recover information that the LLM ignores.

Qwen G-Retriever on ExplaGraphs provides a further exception: its tokens remain geometrically collapsed, yet GSWAP causes a resolved -4.51-point effect. Collapse indicates that little input-specific variation remains, but does not establish complete insensitivity to that variation. For collapsed baselines, nearly identical SWAP and GSWAP likelihood effects indicate generic sensitivity to token perturbation rather than identified graph-specific use.

Three caveats bound the decodability results. The sentence encoder truncates long graph verbalisations, a single sentence vector compresses a subgraph lossily, and the fitted map is linear. Failure to decode is therefore not proof that information is absent, whereas successful decoding is evidence that it is recoverable by this probe.

The Qwen ExplaGraphs no-entity and no-question variants use seed 1 because their seed-0 runs diverged and scored below the majority baseline.

Table 66: **Complete token-content and use results across LLMs.** Q, A, and G are median retrieval ranks; lower is better. ΔScore is the with-text GSWAP effect, and NLL reports SWAP/GSWAP. Bold score effects satisfy $|\Delta| > 2$ and $p < .05$; bold NLL values satisfy the likelihood criterion. Diagnoses also use Tables 4, 70, and 71. The binary `ExplaGraphs` answer is omitted.

Dataset	Model	Q rank	A rank	G rank	ΔScore GSWAP	ΔNLL S/GS	Diagnosis
Llama-2-7B							
`WebQSP`	G-Retriever	293	317	369	-0.55	$+0.0183/+0.0182$	SOFT PROMPT
	MLPQ	407	407	407	–	–	–
	Ours	65	60	315	$-1.97^{\dagger}$	$\mathbf{+0.0895}/\mathbf{+0.0332}$	MASKED
	no entity features	34	51	303	-1.97^{*}	$\mathbf{+0.1111}/+0.0262$	PRIOR
	no question	120	134	329	$\mathbf{-2.33}^{*}$	$\mathbf{+0.0848}/\mathbf{+0.0845}$	USED
`SceneGraphs`	G-Retriever	271	436	248	$-1.10^{\dagger}$	$+0.0169/+0.0167$	SOFT PROMPT
	CONST	500	476	500	–	–	–
	Ours	47	187	427	$\mathbf{-5.30}^{***}$	$+0.0229/\mathbf{+0.1665}$	USED
	no entity features	8	59	369	$+0.35$	$\mathbf{+0.1553}/+0.0045$	PRIOR
	no question	116	395	185	-0.25	$+0.0214/+0.0213$	MASKED
`ExplaGraphs`	G-Retriever	138	–	139	-1.08	$+0.0014/+0.0022$	SOFT PROMPT
	Ours	54	–	41	-1.44	$+0.0010/+0.0032$	DECODABLE
	no entity features	67	–	67	-0.36	$+0.0026/+0.0056$	DECODABLE
	no question	84	–	70	$+1.26$	$-0.0063/-0.0073$	DECODABLE
Qwen2.5-3B, matched training budget							
`WebQSP`	G-Retriever	348	357	388	$-1.60^{\dagger}$	$\mathbf{+0.0734}/\mathbf{+0.0737}$	SOFT PROMPT
	Ours	153	156	349	$\mathbf{-2.09}^{**}$	$+0.0168/+0.0144$	USED
	no entity features	109	110	344	-0.86	$\mathbf{+0.0321}/+0.0166$	PRIOR
	no question	188	198	352	$\mathbf{-2.33}^{**}$	$\mathbf{+0.0336}/\mathbf{+0.0333}$	USED
`SceneGraphs`	G-Retriever	326	446	340	-1.15^{*}	$+0.0127/+0.0125$	SOFT PROMPT
	Ours	112	353	450	$\mathbf{-6.50}^{***}$	$\mathbf{+0.0392}/\mathbf{+0.0410}$	USED
	no entity features	29	121	402	-0.60	$\mathbf{+0.0373}/+0.0019$	PRIOR
	no question	155	408	256	-1.90^{**}	$+0.0174/+0.0172$	DECODABLE
`ExplaGraphs`	G-Retriever	139	–	138	$\mathbf{-4.51}^{*}$	$+0.0146/+0.0138$	USED
	Ours	85	–	62	$+0.36$	$+0.0008/+0.0042$	DECODABLE
	no entity features	92	–	87	-1.99	$+0.0188/+0.0177$	DECODABLE
	no question	109	–	99	-1.44	$+0.0017/+0.0008$	DECODABLE

N VALIDATION OF THE DESIGN CHOICES

Question conditioning in Stage 1. Table 67 reports Stage 1 measured on its own, without the LLM. Top-1 is the fraction of validation samples whose own target is the most similar of all validation targets; the graph gap is the mean cosine between representations of different questions about the same graph minus the mean cosine across graphs, and is defined only on `SceneGraphs`, the one benchmark that asks several questions about one graph. Removing the question conditioning drops top-1 from 0.433 to 0.084, because the answer depends on what is asked and the encoder needs the question to select the relevant part of the scene. The graph gap shows that what it learns is not merely a copy of the question.

Table 67: Stage 1 measured without the LLM. Top-1 is computed against all validation targets of the benchmark, 19,997 on `SceneGraphs`.

Run	Target	Validation top-1	Graph gap
`SceneGraphs`, with question	full answer sentence	0.433	$0.035 \rightarrow 0.204$
`SceneGraphs`, without question	full answer sentence	0.084	Same embeddings, N/A
`SceneGraphs`, scene-disjoint halves	full answer sentence	0.376	$0.035 \rightarrow 0.264$
`WebQSP`, half split	gold answers	0.469	N/A
`ExplaGraphs`, half split	verbalised graph	0.501	N/A

Stage-1 objective. With $\mathbf{h}_i$ the pooled encoder output, $\mathbf{s}_i = \mathrm{LM}(c_i)$ the frozen sentence embedding of the target c_i, batches of size B, temperature κ and a memory bank $\mathcal{M}$ of M other samples' targets, Stage 1 minimises

$$\mathcal{L}_{\mathrm{S1}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\big(\cos(\mathbf{h}_i, \mathbf{s}_i)/\kappa\big)}{\exp\big(\cos(\mathbf{h}_i, \mathbf{s}_i)/\kappa\big) + \sum_{j\in\mathcal{M}} \exp\big(\cos(\mathbf{h}_i, \mathbf{s}_j)/\kappa\big)}, \tag{6}$$

Stage-1 targets. The target is the gold answer on `WebQSP` and the full answer sentence on `SceneGraphs`, which contains 47,487 distinct targets. `ExplaGraphs` has only two answer labels, so an answer target cannot identify a sample and we use the verbalised graph instead.

Freezing the encoder in Stage 2. With a trainable encoder the answer loss re-collapses the token even with a GELU projector, to cosine 0.9139 (Table 3). We verified that the frozen encoder reproduces the intended graph token to within 1.2×10^{-7}.

Parameterisation of the prompt token. The prompt token is a learned constant passed through a Sigmoid projector, which matches G-Retriever on all three datasets (Table 2). Raw prompt tuning scores 11–25 points lower, although this comparison does not isolate parameterisation from capacity.
Alternatives that did not work. Table 68 lists three, all trained with the cross-fitted recipe on `WebQSP` and therefore comparable with the cross-fitted joint model rather than with the main-text numbers. Each answers a question about the design.

Must the two tokens be trained together? If the split alone were what matters, adding the prompt token to a converged projector should work as well as training both from the start. It does not: accuracy stays at 63.64 and the best epoch is the first, because by then the model has settled into using the graph token for both roles and the new token has nothing to take over.

Is a second token needed at all, or only some way of stopping the graph token from being relied on as a constant? Dropout on the projector is the cheaper alternative, and it leaves one token doing both jobs: accuracy reaches 64.56 and the token drifts back towards collapse, at cosine 0.7563 against 0.1778 for the joint model.

Could the redundancy be removed from the data instead of the architecture? Hiding 40% of the verbalised edges during training makes the text an unreliable channel, which should force the token to carry the graph. Instead the model learns to compensate for missing text, and at test time, when the text is complete again, substitute tokens are as good as the real one or better, with other graphs' node features worth $+2.03$ points ($p = 0.03$).

Table 68: Alternatives to the joint Stage 2. WebQSP, cross-fitted recipe, 1,628 test questions. MEAN, SWAP and NODE are differences from each model's own REAL, with the exact McNemar test: † $p < .10$, * $p < .05$, ** $p < .01$, *** $p < .001$, unmarked $p \geq 0.1$.

Variant	Hit@1	Token cosine	MEAN	SWAP	NODE
Joint Stage 2, as used in the paper	69.53	0.1778	$-1.54^{\dagger}$	-3.19^{**}	-3.26^{***}
Projector only	63.14	0.5700	-3.44^{***}	-5.34^{***}	-1.66
Prompt token added after the projector	63.64	0.5700	-2.95^{***}	-5.22^{***}	-2.76^{***}
Dropout 0.5 on the projector	64.56	0.7563	-4.24^{***}	-3.69^{***}	-1.29
40% of verbalised edges hidden in training	67.20	0.1847	$+1.35$	-0.31	$+2.03^{**}$

O PIPELINE AND ROBUSTNESS ABLATIONS

Table 69 extends Table 4 with the complete set of retrained pipeline variants and the principal fixed-weight interventions. The no-entity and no-question rows remove an input during both training and evaluation; they are therefore pipeline ablations rather than interventions on a fixed model. Additional relation and geometric interventions are reported in Appendices I and J.

Removing entity features has a benchmark-dependent effect on task performance: it hurts Qwen on WebQSP and both LLMs on ExplaGraphs, but improves both LLMs on SceneGraphs. Its effect on token use is more consistent. Across both LLMs and all three benchmarks, the no-entity model remains sensitive to MEAN or SWAP, while no GSWAP effect clears the answer-level criterion. The token can therefore remain strongly used after its resolved graph-specific contribution disappears.

Removing question conditioning produces resolved graph use on WebQSP with both LLMs. On Qwen SceneGraphs, the GSWAP effect is statistically significant but does not clear the two-point criterion; neither LLM shows resolved graph use in the corresponding ExplaGraphs variant.

For the standard Qwen SceneGraphs pipeline, the earliest checkpoint was selected and validation loss subsequently increased $(0.2668, 0.2741, 0.2737, 0.2981, 0.3396)$. The same immediate overfitting occurs with Llama-2.

O.1 EVALUATION WITHOUT THE VERBALISED GRAPH

Tables 70 and 71 evaluate the trained models after deleting the verbalised graph from the prompt. The graph token is then the only possible graph channel. NOTXT is the resulting absolute score, and intervention columns report condition minus NOTXT. These evaluations are fixed-weight interventions: the models are not retrained without text.

A NOTXT-MEAN effect establishes dependence on sample-specific token content, but does not identify that content as graph-specific. NOTXT-GSWAP provides the graph-specific test because it changes the graph while preserving the remaining inputs. We therefore do not infer that graph information is masked by text from MEAN alone.

With Llama-2, removing the verbalised graph exposes resolved graph-specific effects for the standard pipeline on WebQSP and SceneGraphs, and for the question-free SceneGraphs variant. No corresponding effect is found on ExplaGraphs.

With Qwen2.5-3B, several models depend on sample-specific token content after the text is removed, most strongly on SceneGraphs and in the ExplaGraphs ablations. Most of these rows currently contain only MEAN; they therefore establish token dependence but do not identify the graph as the information being used. The standard Qwen ExplaGraphs pipeline instead shows no resolved text-free token effect, despite its improved task score and decodable graph information.

Table 69: **Pipeline variants under fixed-weight interventions.** *Ours* denotes the complete two-token pipeline; indented rows are retrained diagnostic variants. *Projector only* is the single-token model with the aligned frozen encoder but no separate prompt token. Intervention columns report condition minus REAL, so negative values indicate worse performance. Lower token cosine indicates greater input dependence. Dashes denote conditions not evaluated.

Dataset	Model	REAL	Token cos.	ΔMEAN	ΔSWAP	ΔNODE	ΔEDGE	ΔGSWAP
Llama-2-7B								
WebQSP	projector only	64.62	0.543	-14.74^{***}	-9.89^{***}	-2.40^{*}	$+0.06$	–
	Ours	69.90	0.142	-3.13^{***}	-2.15^{*}	-1.23	-0.80	$-1.97^{\dagger}$
	no entity features	72.73	0.125	-6.14^{***}	-5.65^{***}	-0.61	-0.43	-1.97^{*}
	no question	69.84	0.169	-0.06	-3.69^{***}	-2.09^{*}	-0.49	-2.33^{*}
	cross-fitted	69.53	0.178	$-1.54^{\dagger}$	-3.19^{**}	-3.26^{***}	-0.55	-2.83^{**}
	G-Retriever + LoRA	74.32	0.997	-0.43	-0.74	-0.98	-0.12	–
	Ours + LoRA	74.57	0.170	-1.11	-2.33^{**}	-0.80	$+0.25$	–
SceneGraphs	projector only	69.15	0.519	-20.20^{***}	-35.05^{***}	-8.90^{***}	-2.45^{*}	–
	Ours	72.50	0.133	-1.40	-3.40^{***}	-5.10^{***}	-1.05	-5.30^{***}
	no entity features	79.65	0.125	-5.75^{***}	-9.60^{***}	$+0.10$	$+0.40$	$+0.35$
	no question	76.35	0.225	-1.15	-0.25	-0.75	$+0.35$	-0.25
ExplaGraphs	projector only	57.76	0.323	$+1.08$	-6.68^{*}	-1.26	-6.14^{*}	–
	Ours	85.74	0.099	$+0.36$	-0.54	-0.54	$+1.08$	-1.44
	no entity features	84.48	0.113	$+3.07^{**}$	$+0.00$	$+0.18$	$+0.18$	-0.36
	no question	87.55	0.108	$+1.08$	$+0.90$	$+0.18$	-0.18	$+1.26$
Qwen2.5-3B, matched training budget								
WebQSP	G-Retriever	57.13	0.997	-0.55	-1.41	-0.31	$+0.06$	$-1.60^{\dagger}$
	Ours	65.79	0.200	-2.09^{**}	-1.90^{**}	-0.80	-1.35^{*}	-2.09^{**}
	no entity features	58.54	0.180	-2.03^{**}	-1.84^{*}	-0.06	$-1.23^{\dagger}$	-0.86
	no question	60.14	0.194	-0.86	-2.33^{**}	-2.70^{***}	-0.55	-2.33^{**}
SceneGraphs	G-Retriever	76.05	0.997	-0.30	$-1.05^{\dagger}$	-0.35	$+0.30$	-1.15^{*}
	Ours	72.65	0.139	-7.00^{***}	-5.75^{***}	-4.40^{***}	-1.65^{*}	-6.50^{***}
	no entity features	75.95	0.199	-2.35^{**}	-4.60^{***}	-0.20	-0.55	-0.60
	no question	75.40	0.557	-0.75	-1.40^{*}	-0.55	$+0.50$	-1.90^{**}
ExplaGraphs	G-Retriever	74.37	0.998	-1.62	-2.17	-0.72	-2.71	-4.51^{*}
	Ours	83.94	0.103	$+0.90$	$+0.00$	$+0.00$	$+0.36$	$+0.36$
	no entity features	80.14	0.116	$+2.17$	-3.07	-0.18	-0.72	-1.99
	no question	74.37	0.118	-1.44	-0.54	$+0.54$	$+0.00$	-1.44

Table 70: **Llama-2 evaluation without the verbalised graph.** Intervention columns report differences from NOTXT; the final column gives the corresponding score with the verbalised graph present. Bold GSWAP effects satisfy $|\Delta| > 2$ and $p < .05$. Dashes denote conditions not evaluated.

Dataset	Model	NOTXT	ΔMEAN	ΔSWAP	ΔGSWAP	With text
WebQSP	G-Retriever	41.34	$+1.04$	-0.18	-0.86	70.33
	Ours	28.75	-2.09^{*}	-2.40^{*}	$\mathbf{-2.27}^{*}$	69.90
	no entity features	28.38	-7.49^{***}	–	–	72.73
	no question	30.90	$+0.43$	–	–	69.84
SceneGraphs	G-Retriever	30.20	$+0.45$	-1.40	–	81.35
	Ours	32.60	-6.60^{***}	-10.85^{***}	$\mathbf{-3.10}^{**}$	72.50
	no entity features	30.10	-12.85^{***}	–	–	79.65
	no question	26.15	-8.25^{***}	-3.25^{**}	$\mathbf{-3.30}^{**}$	76.35
ExplaGraphs	G-Retriever	66.43	-0.90	0.00	–	85.56
	Ours	69.68	$+0.18$	$+0.18$	-0.54	85.74
	no entity features	70.22	$+0.72$	–	–	84.48
	no question	78.88	$+1.08$	–	–	87.55

P CROSS-FITTING: IS THE TOKEN'S USEFULNESS STAGE-1 MEMORISATION?

Stage 1 partly memorises its training answers, with top-1 about 0.9 on training samples against about 0.5 on validation samples. A reader may therefore ask whether the LLM's reliance on the token in Section 4.4 is reliance on answers the encoder memorised. We test this by training Stage 1 on one half of the training questions and Stage 2 on the other, splitting `SceneGraphs` by scene since one scene carries many questions. No test data is involved in either regime: Stage 1 uses training labels only, and on `SceneGraphs` the training and test sets share no scenes.

Table 72 shows that the reliance is partly memorisation. When the two stages see disjoint halves the interventions still hurt, but less, while accuracy rises towards the baseline trained on the same half. The two regimes are two operating points of the trade-off described in Section 4.4.

Rewiring the edges costs the cross-fitted joint model 1.95 points on `SceneGraphs` in one run and nothing in the other, so structure does not reach the LLM reliably in this regime either.

Table 71: **Qwen2.5-3B evaluation without the verbalised graph.** Intervention columns report differences from NOTXT; the final column gives the corresponding score with the verbalised graph present. Bold GSWAP effects satisfy $|\Delta| > 2$ and $p < .05$. Dashes denote conditions not evaluated.

Dataset	Model	NOTXT	ΔMEAN	ΔSWAP	ΔGSWAP	With text
WebQSP	G-Retriever	29.30	$+0.86$	$-2.03^{\dagger}$	$-1.84^{\dagger}$	57.13
	Ours	34.34	$+0.31$	$+1.11$	$+1.11$	65.79
	no entity features	30.10	$+1.97^{*}$	–	–	58.54
	no question	32.74	$+0.61$	–	–	60.14
SceneGraphs	G-Retriever	39.50	$+1.95^{*}$	–	–	76.05
	Ours	43.90	-7.10^{***}	–	–	72.65
	no entity features	37.75	-2.90^{*}	–	–	75.95
	no question	34.70	-3.15^{***}	–	–	75.40
ExplaGraphs	G-Retriever	78.88	0.00	–	–	74.37
	Ours	82.13	-1.62	$+0.18$	-0.18	83.94
	no entity features	75.99	-4.51^{*}	–	–	80.14
	no question	76.17	-7.40^{***}	–	–	74.37

Table 72: Cross-fitted runs, in which Stage 1 and Stage 2 see disjoint halves of the training questions. Accuracy is compared with the G-Retriever baseline trained on the same half; the intervention columns are differences from the model's own REAL, with the exact McNemar test. WebQSP 1,628 test questions, SceneGraphs the first 2,000, ExplaGraphs 554.

Dataset	Stage 2	Accuracy	Baseline	MEAN	SWAP	NODE	EDGE
WebQSP	projector only	63.14	70.52	-3.44^{***}	-5.34^{***}	-1.66	$+1.04$
WebQSP	projector + prompt	69.53	70.58	$-1.54^{\dagger}$	-3.19^{**}	-3.26^{***}	-0.55
SceneGraphs	projector + prompt, run 1	76.90	78.57	$-1.50^{\dagger}$	-4.45^{***}	-1.05	-1.95^{***}
SceneGraphs	projector + prompt, run 2	77.55	78.57	-1.10	-2.45^{**}	$+1.00$	$+0.30$
ExplaGraphs	projector + prompt	76.35	80.87	-3.61^{*}	$+1.08$	$+0.72$	$+0.18$

Q INITIALISATIONS AND EVALUATION NOISE

Table 73 reports three initialisations of G-Retriever and of the joint model on `WebQSP`, trained with the cross-fitted recipe on the same half. The spread across initialisations, 1.3 to 1.5 points, is larger than the difference between the two models, which is what licenses the word parity in Section 4.4. Pooling the discordant questions over the three runs, $n = 4{,}884$, leaves our token used and G-Retriever's not.

Table 73: Three initialisations on `WebQSP`, cross-fitted recipe, 1,628 test questions each. The intervention columns are differences from REAL pooled over the three runs, with the exact McNemar test.

Model	Run 1	Run 2	Run 3	Mean $\pm$ sd	MEAN	SWAP	NODE
G-Retriever	70.58	67.75	70.02	69.45 ± 1.50	≤ 0.53	≤ 0.53	≤ 0.53
Ours, joint	69.53	66.89	68.43	68.28 ± 1.33	-1.19^{**}	-1.45^{**}	-1.84^{***}

Evaluation noise between a training-script score and a separate scoring run is reported in Appendix D.

R THE VERBALISED GRAPH IS NOT ENOUGH

In this section, we bring some qualitative evidence that LLMs generally struggle to deal with verbalized graphs only. `ExplaGraphs` is the clean test, because its graph is not retrieved: it is the human-annotated explanation graph of the stance, given in full, so an error cannot be blamed on missing information. We give a frozen Llama-2-7b-chat[1] the complete graph as a list of triples inside G-Retriever's prompt, zero-shot, and score it with the benchmark's evaluator on all 554 test questions.

The majority class is 58.8%. The arguments alone give 61.0%, and adding the complete explanation graph gives 64.3%, with 198 of 554 answers still wrong. The graph fixes 68 questions and breaks 50, so verbalising it misleads the model about as often as it helps. Part of this is a bias towards "support", which accounts for 451 of 554 answers and for 161 of the 198 errors, but not all of it: Figure 7 shows an error in which the graph states the opposition explicitly.

`ExplaGraphs` #2066. gold: *counter* — arguments only: *counter* — arguments + full graph: ***support***

Argument 1: Legalized organ trade would exploit the poor.
Argument 2: Legalized organ trade would aid the poor, by giving them a large sum of money to improve their lives.
Graph: legalized organ trade $\xrightarrow{\text{causes}}$ money for organs $\xrightarrow{\text{capable of}}$ improving finances $\xrightarrow{\text{capable of}}$ improve their lives $\xrightarrow{\text{has context}}$ aid the poor; aid the poor $\xrightarrow{\textbf{is not a}}$ exploit the poor

Figure 7: **The verbalised graph can turn a right answer into a wrong one.** Frozen Llama-2-7b-chat, zero-shot, G-Retriever's `ExplaGraphs` prompt with the complete human-annotated graph as triples. The model is right from the arguments alone and wrong once the graph, which states the opposition explicitly in the relation set in bold, is added. This happens on 50 of the 554 test questions.

S EXTENDED RELATED WORK

Graph soft prompting. A growing line of work connects graph encoders to frozen or adapted LLMs by projecting graph representations into continuous tokens. GraphToken (Fatemi et al., 2024) learns

[1]The chat model is used here because the base Llama-2-7b does not answer zero-shot with this prompt; every other experiment in the paper uses the base model, as G-Retriever does.

an encoding function that emits soft tokens for a frozen LLM on synthetic graph-reasoning tasks, while GraphLLM (Chai et al., 2026) targets graph reasoning with a tuned decoder and GNP (Tian et al., 2024) adds a cross-modality pooling module for multiple-choice knowledge-graph questions. Among these, GNP and G-Retriever (He et al., 2024) are the only ones that retrieve a question-specific subgraph before encoding; the others operate on the full graph or on synthetic inputs. G-Retriever additionally verbalises the retrieved subgraph in the prompt, creating a redundant configuration that is the focus of our audit. GraphGPT (Tang et al., 2024) combines a text-grounded structural encoder with a lightweight graph–text alignment projector and dual-stage instruction tuning. These systems share the interface we study: a pooled graph representation, a trained projector, and one or a few tokens prepended to the prompt.

Multi-token and pooling-based interfaces. Very few works have explored multi-token interfaces for graph-to-LLM systems. The single-token bottleneck created by mean pooling over a subgraph has only recently been identified as a structural limitation. Grover et al. (2026) argue that aggressive mean pooling compresses entire graph substructures into a single token, creating a severe information bottleneck, and propose multi-token pooling operators to increase the bandwidth of the graph-to-LLM interface. Their work also reports that pooling methods are unstable under soft prompt tuning with a frozen LLM and require LoRA fine-tuning to stabilise. This is complementary to our diagnosis: we study a single-token interface and show that its collapse is associated with both the projector parameterisation and end-to-end answer training.

Graph–language alignment. Alignment-based methods add an auxiliary objective intended to bring graph representations closer to language representations. Align-GRAG (Xu et al., 2025) is closest to our Stage 1: it contrastively aligns a pooled graph representation with a textual target using KL divergence and contrastive losses. G-Retriever (He et al., 2024) can also be viewed as an implicit alignment method, since its projector is trained to map graph representations into the LLM's embedding space under the answer loss. Our use of contrastive pretraining is similar to Align-GRAG, but our question is different. Similarity between graph and text spaces does not establish that different graphs produce different tokens after projection, nor that the LLM uses their graph-specific content. We therefore complement representation alignment with measurements of token variation and fixed-weight interventions on the trained model.

Why graph use is not guaranteed. Two recent works already point out our diagnosis from different perspectives. Zhang et al. (2026) prove that the autoregressive language objective of graph-tokenizing LLMs decomposes into a textual dependency term and a graph–text alignment term, and that current methods optimize only the former explicitly. Graph–text alignment is therefore implicit: the training objective does not reward the model for using graph information, and a text-dominant solution can be optimal under the loss. Petkar et al. (2026) show empirically that existing graph–language benchmarks can be matched by graph-only or language-only baselines, and introduce CLEGR to require compositional reasoning over structure and text; even there, raw soft-prompted LLMs remain competitive with GNN-backed models. Together, these results establish that neither the training objective nor the benchmark forces a model to use graph information. Our work is complementary in level rather than in kind: we study the interface itself and show that a graph token can collapse to a near-constant vector even when the encoder representations are distinct, and that preventing collapse does not guarantee that the LLM uses the token's graph-specific content. This is what motivates the diagnostic protocol we introduce.

Retrieval for graph-grounded generation. G-Retriever (He et al., 2024) formulates subgraph retrieval as a prize-collecting Steiner tree problem (Bienstock et al., 1993) and is the reference implementation we study. GRAG (Hu et al., 2025) retrieves subgraphs for generation with a different scoring rule, emphasising the importance of subgraph structure and topology. GraSP (Wang et al., 2026) encodes subgraphs into soft prompts using GNNs for multi-hop KBQA and combines this with a two-stage LLM reasoning procedure. GNN-RAG (Mavromatis & Karypis, 2024) uses a GNN to retrieve reasoning paths that are then verbalised, rather than projecting them as soft tokens. Retrieval quality is orthogonal to our question: our interventions keep the retrieved subgraph fixed and change only the token computed from it.

Parameter-efficient tuning and continuous prompting. Fine-tuning an LLM can be done in several ways. Full-weight fine-tuning updates all parameters. Parameter-efficient methods update a small subset or add new parameters, and include LoRA (Hu et al., 2021), which learns low-rank updates to existing weights, and continuous prompting methods. Continuous prompting conditions a

frozen language model with learned continuous vectors. This family includes prompt tuning (Lester et al., 2021), which learns input-independent embeddings, and prefix-tuning (Li & Liang, 2021), which learns continuous prefixes made available across the model's layers. A graph token is a form of continuous prompting, but with a different intended role: it is computed from each input graph and is therefore expected to preserve input-specific information, unlike standard prompt tuning where the prompt is shared across all inputs. Our results connect these settings by showing that Graph Soft prompting, as it is, collapses into a nearly input-independent vector and consequently behaves like a conventional soft prompt. LoRA is used in our work only as a robustness check on the frozen-LLM setting, and the effect is consistent.