FROZEN TEXT EMBEDDINGS WITH A GNN IS "ALL YOU NEED" FOR TEXT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work in text classification constructs token-level graphs from the embeddings of a frozen language model and trains a graph neural network (GNN) on these graphs to perform downstream classification. We show that existing approaches lack consistent performance across standard benchmarks, with no single graph construction strategy dominating across datasets. To address this limitation, we introduce HILT-GCN, a simple hierarchical token-level graph architecture that models multi-layer interactions via layer-specific virtual hubs at linear complexity $O(LT)$. HILT-GCN achieves the strongest and most stable overall performance among frozen token-level approaches across five single-label benchmarks. Compared with strong fully fine-tuned baselines based on both small and large text encoders, HILT-GCN achieves competitive results on four out of five datasets, falling behind only on Ohsumed. By using frozen embeddings, HILT-GCN supports any new text backbone without encoder fine-tuning, achieving high accuracy while substantially reducing the parameter and training overhead of large-scale models.

1 INTRODUCTION

Motivation. Standard discriminative text classification models map an input sequence $\mathbf{w}$ to a document-level embedding $\mathbf{h} \in \mathbb{R}^d$ using a pretrained language model, e. g., BERT (Devlin et al., 2019) or Llama (Touvron et al., 2023). Typically, token representations are collapsed via non-parametric pooling or by extracting a `[CLS]` representation, followed by a linear classification head (Galke et al., 2025). While freezing the language model minimizes computational overhead, collapsing token representations into a single static vector introduces an information bottleneck that restricts the model's ability to retain task-specific token interactions. To address this limitation, approaches have been developed that construct *a token-level graph on the frozen embeddings* and fine-tune a Graph Neural Network (GNN) for text classification, e. g., Donabauer & Kruschwitz (2025); Mantri et al. (2026); Pimentel et al. (2026); Ulanovski et al. (2026).

Token-level Graph Learning for Text Classification. We abstract the existing token-level approaches for text classification to a general pipeline shown in Figure 1. Any existing approach consists of five stages, namely a frozen text embedder producing tokens, a graph constructor building a token-level graph, a GNN module trained on the graph, a readout pooling the node embeddings, and finally a classifier. The difference between the approaches lies in how these components are specified. As *Graph Constructor*, TokenGraph uses a window-based approach to construct a graph over the tokens of the last layer (Donabauer & Kruschwitz, 2025). GLOT also considers only the last layer but computes a token-level graph based on cosine similarity (Mantri et al., 2026), while Pimentel et al. (2026) exploit the last-layer's attention weights, which we refer to as AttnGraph. The approach of Ulanovski et al. (2026) considers all layers, creates a node per layer via its token averages, and constructs a Cayley graph over the nodes, which we refer to as CayleyGCN. In terms of the *GNN Module*, the approaches either use a standard Graph Convolution Network (GCN) (Kipf & Welling, 2017), Graph Attention Network (GAT) (Velickovic et al., 2018), or GATv2 (Brody et al., 2022). Finally, the *Readout* is either a mean over all graph nodes or specific nodes.

Contribution. In the literature, existing token-level approaches lack consistent performance across benchmarks, with none consistently outperforming the others across all datasets. We propose HILT-GCN, a simple token-level graph learner that models intra-layer token interactions via layer-specific

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.

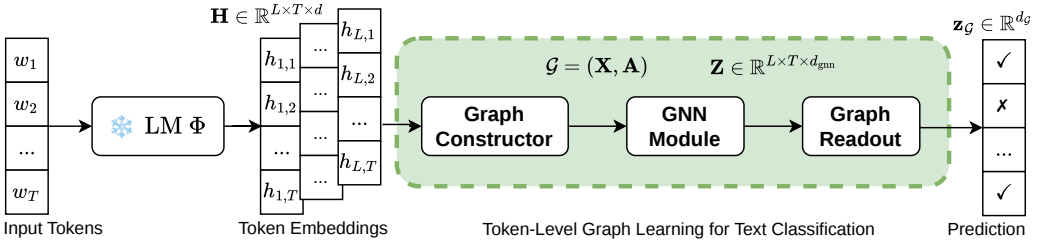


Figure 1: High-level overview of the token-level graph learning pipeline for text classification. A tokenized text sequence is passed through a frozen language model, producing layer-wise token features. A graph is constructed from these features and used for training in a GNN Module. A readout function produces a graph-level representation used for classification prediction.

virtual hubs at linear complexity $O(LT)$, consolidating representations across all layers through unified graph readout. We employ a GCN with residuals and jumping knowledge inspired by Chen et al. (2020).

Experiments on five single-label datasets show that **HILT-GCN achieves the strongest overall performance among token-level graphs on frozen embeddings**, outperforming prior graph approaches in most settings. We also compare HILT-GCN using frozen embeddings to strong fine-tuned baselines using small models like BERT (Devlin et al., 2019) and large embedders like Qwen 3 (Zhang et al., 2025). On four datasets, HILT-GCN achieves results close to fine-tuned embedders. It is only outperformed on Ohsumed, where our fine-tuned embedding-based baseline is 4 points better and establishes a new state-of-the-art (see comparison in Appendix D). This makes token-level graph learning with HILT-GCN a viable alternative to fine-tuning encoder-only models or parameter-efficient adaptation of large models. In summary, our key contributions are:

- **Architecture:** We introduce HILT-GCN, a linear-complexity token-level graph learner with multi-layer virtual hubs and Jumping Knowledge readout.
- **Downstream Efficacy (RQ 1):** We demonstrate that HILT-GCN achieves the strongest overall performance among frozen graph methods across five benchmarks, coming close to fully fine-tuned baselines on 4 out of 5 datasets.
- **Embedding Robustness (RQ 2):** We validate that HILT-GCN scales consistently across six distinct transformer backbones ranging from 110M to 8B parameters.
- **Computational Efficiency (RQ 3):** We show that HILT-GCN reduces trainable parameters by over 92% and speeds up 8B training throughput by up to 4× over LoRA.

2 Preliminaries & Problem Formulation

Problem Setting. Given a sequence of T tokens $\mathbf{w} = (w_1, \ldots, w_T)$ over vocabulary Σ, the task is to predict class probabilities $\hat{\mathbf{y}} \in [0,1]^C$ over C target classes. We target single-label classification ($\hat{\mathbf{y}}$ normalized via $\mathrm{Softmax}$).

Frozen Feature Extraction. We pass $\mathbf{w}$ through a frozen, pretrained transformer backbone Φ with L layers and hidden dimension d, extracting representations across all layers into a feature tensor:

$$\mathbf{H} = \Phi(\mathbf{w}) \in \mathbb{R}^{L\times T\times d},$$

where $\mathbf{h}_{\ell,t} \in \mathbb{R}^d$ is the representation of token t at layer ℓ.

Graph Neural Networks. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with initial node feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. As high-dimensional feature vectors are computationally expensive, typically an initial downward projection is applied to these features with $d_{\text{gnn}} \ll d$. A Graph Neural Network (GNN) updates node representations via message passing:

$$\mathbf{Z} = \mathrm{GNN}(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{|\mathcal{V}|\times d_{\text{gnn}}}.$$

Readout. For graph-level classification, the node representations are aggregated into a summary vector $\mathbf{z}_{\mathcal{G}} = \mathrm{Readout}(\mathbf{Z})$, and predictions are obtained via a classification head $\hat{\mathbf{y}} = \sigma(\mathrm{Linear}(\mathbf{z}_{\mathcal{G}}))$, where σ is the activation function and Linear is a linear projection.

3 Proposed Method: HILT-GCN

Existing token-level graph approaches deal with a trade-off regarding expressive inter-token modeling and scalability. Ulanovski et al. (2026) demonstrated in an ablation that extending graph representations to the full token level across all layers ($|\mathcal{V}| = LT$) yields significant performance gains (ranging from 4% up to 20% in relative gains depending on the benchmark). At the same time, many approaches already rely on all-to-all token interactions when building their graphs, such as (Mantri et al., 2026; Pimentel et al., 2026), increasing the algorithmic complexity from $O(T^2)$ to $O((LT)^2)$. To achieve both inter-layer token interactions and scalability, we introduce HILT-GCN. Below, we describe HILT-GCN following the five steps of the general pipeline illustrated in Figure 1.

Stage 1: Frozen Embedder. Our approach is model-agnostic and operates directly on embeddings from arbitrary pretrained transformer backbones Φ. It works uniformly with token embeddings across both bidirectional encoder-only backbones, e. g., BERT (Devlin et al., 2019), and causal decoder-only architectures, e. g., Llama (Touvron et al., 2023). These models remain frozen and serve strictly as a feature extractor. The target sample text is tokenized using the respective tokenizer for the chosen language model. The resulting token sequence $\mathbf{w}$ is used in a standard forward pass to extract the multi-layer hidden state tensor $\mathbf{H}$.

Stage 2: Graph Constructor. Given the multi-layer token tensor $\mathbf{H}$, we allocate one node for each individual token representation across all layers, yielding an initial node set of size of LT. For each token representation, we instantiate a token node $v_{\ell,t}$ such that $\mathbf{x}_{v_{\ell,t}} = \mathbf{h}_{\ell,t} \in \mathbb{R}^d$. We label the set of initial token nodes $\mathcal{V}_{\text{tok}}$. To facilitate hierarchical routing, we introduce L layer-specific virtual hub nodes v_ℓ^{hub} connected to all token nodes within their respective layer via undirected edges. Finally, we introduce a global virtual node v^{root} connected to all L layer hubs. These virtual nodes are initialized with zero vectors and are used strictly for message passing. The resulting graph contains $|\mathcal{V}| = LT + L + 1$ nodes and $|\mathcal{E}| = LT + L$ undirected edges, maintaining linear complexity $O(LT)$ with respect to sequence length and layer depth. An example is shown in Figure 2.

Stage 3: GNN Module. To propagate structural information across the graph, we employ a K-layer GCN with hidden dimension $d_{\text{gnn}} = 256$, dropout $p = 0.1$, and a default $K = 2$. We first project the multi-layer token representations $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ into the latent GNN space via LayerNorm (LN) (Ba et al., 2016), followed by a linear transformation, i. e., $\mathbf{Z}^{(0)} = \mathrm{Linear}(\mathrm{LN}(\mathbf{X})) \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{gnn}}}$. Following successful practices for training deep GNNs (Li et al., 2019; Chen et al., 2020), each layer applies a Pre-LayerNorm residual connection (He et al., 2016; Xiong et al., 2020) with GELU nonlinearities (Hendrycks & Gimpel, 2016):

$$\mathbf{Z}^{(k)} = \mathbf{Z}^{(k-1)} + \mathrm{Dropout}\left(\mathrm{GELU}\left(\mathrm{GCN}(\mathrm{LN}(\mathbf{Z}^{(k-1)}), \mathbf{A})\right)\right), \quad k \in \{1, \ldots, K\}.$$

To capture multi-hop structural hierarchies and mitigate over-smoothing, we combine all intermediate states via a concatenation of Jumping Knowledge (JK) (Xu et al., 2018), followed by LayerNorm:

$$\mathbf{Z}_{\text{JK}} = \mathrm{LN}\left(\left[\mathbf{Z}^{(0)} \,\|\, \mathbf{Z}^{(1)} \,\|\, \ldots \,\|\, \mathbf{Z}^{(K)}\right]\right) \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{JK}}},$$

where $\|$ denotes concatenation along the feature dimension, with $d_{\text{JK}} = (K+1)d_{\text{gnn}}$. By including $\mathbf{Z}^{(0)}$ directly in $\mathbf{Z}_{\text{JK}}$, the readout preserves unperturbed backbone semantics alongside k-hop topological context (e. g., localized token transformations at $k = 1$ and intra-layer hub-mediated context at $k = 2$), allowing the model to balance local token representations with broader structural context.

Stage 4: Graph Readout. To produce the graph summary vector $\mathbf{z}_{\mathcal{G}}$, we aggregate the updated representations across the LT token nodes $\mathcal{V}_{\text{tok}}$, filtering out the virtual routing nodes. We apply symmetric mean and max pooling over their representations $\mathbf{z}_v$ denoting the row of $\mathbf{Z}_{\text{JK}}$ for node v:

$$\mathbf{z}_{\mathcal{G}} = \left[\underset{v \in \mathcal{V}_{\text{tok}}}{\mathrm{mean}}(\mathbf{z}_v) \;\Big\|\; \underset{v \in \mathcal{V}_{\text{tok}}}{\max}(\mathbf{z}_v)\right] \in \mathbb{R}^{d_{\mathcal{G}}},$$

where operations are evaluated element-wise along the feature dimension, with $d_{\mathcal{G}} = 2(d_{\text{JK}})$.

Stage 5: Classifier. The graph summary vector $\mathbf{z}_{\mathcal{G}}$ is mapped to target class predictions through a two-layer Multi-Layer Perceptron (MLP) with LayerNorm, GELU activation, and dropout, where σ is Softmax for single-label classification.

$$\hat{\mathbf{y}} = \sigma\Big(\text{Linear}_2\left(\text{Dropout}(\text{GELU}(\text{LN}(\text{Linear}_1(\mathbf{z}_{\mathcal{G}}))))\right)\Big) \in [0,1]^C.$$

4 Experimental Setup

Datasets & Metrics. We report results for five single-label benchmarks: 20NG Lang (1995), R8 and R52 Cardoso-Cachopo (2007), Ohsumed Hersh et al. (1994), and MR Pang & Lee (2005). These datasets are classical benchmarks in the text classification community (Galke et al., 2025). A descriptive overview of dataset characteristics can be found in our appendix at Table 7.

For evaluation we report micro-averaged F_1 (Micro-F1) and macro-averaged F_1 (Macro-F1). For single-label datasets, Micro-F1 is mathematically equivalent to accuracy, which serves as the typical evaluation metric for such datasets (Galke et al., 2025). While Micro-F1 serves as our primary metric, we report both to account for class imbalance, as underrepresented classes heavily influence Macro-F1 while having minimal impact on Micro-F1.

Embedder. The choice of transformer baselines significantly impacts evaluation. Larger models typically produce better embeddings but also increase computational footprints. We experimented with embeddings generated across 6 different transformer backbones. These include moderate-scale backbones with around (110M–355M parameters), i. e., `bert-base-uncased` and `bert-large-uncased` (BERT) (Devlin et al., 2019) as well as `roberta-base` and `roberta-large` (Liu et al., 2019). We consider large-scale backbones (7B+ parameters), including `meta-llama/Meta-Llama-3-8B` (Llama3) (Llama, 2024), and `Qwen3-Embedding-8B` (Qwen3) (Zhang et al., 2025). Unless stated otherwise, backbones are completely frozen and function strictly as deterministic feature extractors. For clarity and readability, our main result tables include numbers primarily for `bert-base-uncased`, a standard transformer baseline used in text classification Galke et al. (2025), as well as `Qwen3-Embedding-8B`, which performs strongly on the MTEB benchmark (Muennighoff et al., 2023) and provides high-quality text embeddings.

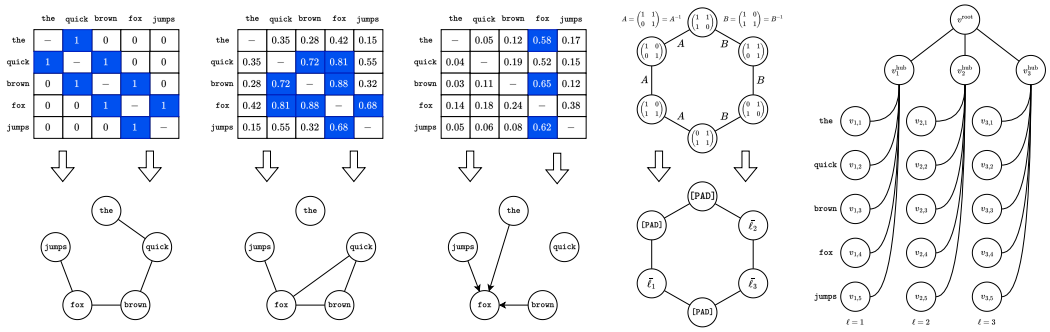
Baseline Models. We compare our approach with two simple linear architectures and four established token-level graph architectures. The non-graph architectures take the mean-pooled last-layer sequence representation in the case of encoder-based models and the last-token representation of the last-layer representation in the case of the decoder-based transformers. The linear baseline applies a Linear projection mapped directly to the class output space, while the MLP baseline uses a two-layer MLP with the same hidden size as the transformer. To show the efficacy of token-level graph approaches, we compare the non-graph architectures with re-implementations of TokenGraph (Donabauer & Kruschwitz, 2025), GLOT (Mantri et al., 2026), AttnGraph (Pimentel et al., 2026), and CayleyGCN (Ulanovski et al., 2026). An example of the graphs used by each method can be seen in Figure 2. Full architectural and hyperparameter details are summarized in Appendix A.

(a) TokenGraph (Donabauer & Kruschwitz, 2025) constructs an undirected, unweighted graph over the final-layer tokens by connecting nodes within a fixed sliding window, i. e., $|i-j| < W, W = 2$. Representations are updated via a 2-layer GAT and pooled via global mean pooling.

(b) GLOT (Mantri et al., 2026) derives fully dynamic semantic graphs by computing pairwise cosine similarities between final-layer token embeddings. Edges are instantiated if similarities exceed a threshold of $\tau = 0.6$, retaining the scalar similarity as an edge attribute. Representations are propagated through a 2-layer GAT, aggregated via layer-wise JK concatenation, and summarized using an attentional score-pooling mechanism.

(c) AttnGraph (Pimentel et al., 2026) constructs graphs by aggregating attention weights across transformer heads and layers, sparsified via a quantile threshold $q = 0.9$. Message passing is conducted with a 2-layer GATv2 with multi-dimensional edge features, followed by global mean pooling.

(d) CayleyGCN (Ulanovski et al., 2026) models inter-layer representations (across averaged layer token-representations $\bar{\mathbf{h}}_\ell$) by randomly mapping nodes onto an expander graph defined by the Cay-



(a) TokenGraph (b) GLOT (c) AttnGraph (d) CayleyGCN (e) HILT-GCN (Ours)

Figure 2: Comparison of graph characteristics for our GNN baselines with the example input sequence: `the quick brown fox jumps`, assuming 3 transformer layers. (**a**) TokenGraph with a sliding-window-based adjacency matrix, (**b**) GLOT with a pairwise token-feature cosine similarity matrix, (**c**) AttnGraph with a pairwise aggregated attention weight matrix, and (**d**) CayleyGCN with a CayleyGraph over $SL(2, \mathbb{Z}_n)$ and randomly mapped layer feature aggregations $\bar{\mathbf{h}}_\ell$. (**e**) visualizes our hierarchical approach in HILT-GCN.

ley graph over $\mathrm{SL}(2, \mathbb{Z}_n)$. Surplus nodes required to match group order are instantiated as zero-initialized virtual nodes. Propagation is performed via a 2-layer GCN with input projection and LayerNorm; virtual nodes are filtered out prior to global pooling.

Implementation Details. Experiments are repeated for five seeds $\{1, 2, 3, 4, 5\}$ with the results being averaged and the standard deviation reported in subscript. Early stopping is configured with a base patience of 20 epochs when using small-scale embedders and 10 epochs when using large-scale embedders. For our fine-tuned baselines, these patience rates are 10 and 5, respectively. The early stopping criterion is the $\mathrm{Softmax}$ cross-entropy loss of the logits on the validation set. We use a batch size of 32 for small-scale embedders and a batch size of 8, with 4 gradient accumulation steps, for large-scale embedders. To ensure a fair comparison across models with varying context windows and baseline computational limits, we truncate all input sequences to 512 tokens.

Hyperparameter Optimization. Hyperparameter Optimization is done only for the learning rate of the GNN/classification head. We run a grid search across the learning rate magnitudes of $\{1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}\}$, using the best accuracy score on the validation set as the selection criteria. For running our baselines, we decouple the learning rates of the transformer and the classification head, motivated by Howard & Ruder (2018). In this instance, the grid search is extended with transformer learning rates $\{3e^{-5}, 3.5e^{-5}, 5e^{-5}, 1e^{-5}\}$ for fully fine-tuned (FFT) small-scale embedders and learning rates $\{1e^{-4}, 2e^{-4}\}$ for parameter-efficient fine-tuned (PEFT) large-scale embedders, using LoRA Hu et al. (2022). Following Dettmers et al. (2023), LoRA is applied by injecting adapters into all linear modules. We use rank $r = 16$, scaling factor $\alpha = 16$, under Rank-Stabilized LoRA (rsLoRA) (Kalajdzievski, 2023); LoRA dropout is set to 0.05. Implementation details and detailed experimental setup can be found in Appendix A

5 Results

Downstream Efficacy (RQ 1). The results of our experiments concerning the efficacy of using frozen embeddings for text classification are shown in Table 1. A more extensive comparison table is available in Appendix C. We organize the results by embedding backbone and baseline. For comparison, we also report performance for fine-tuned small and large language model baselines. These tuned variants serve as empirical upper bounds to evaluate how closely frozen representations can approach performance without modifying the underlying weights.

A few general trends become evident from our results. First, our simple frozen baselines tend to underperform severely in comparison to GNN baselines when using typical embedding models such as BERT. This observation might lead to the conclusion that frozen embeddings are not up to the task compared to fine-tuned counterparts. For Qwen 3, however, this is not true, as simple

Table 1: **Classification performance across single-label benchmark datasets.** Test-set Micro-F1 and Macro-F1 (mean ± std over 5 runs). **Bold** and underline denote best and second-best performance among frozen methods per embedding backbone.

Method	20NG		R8		R52		Ohsumed		MR	
	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1
`bert-base-uncased`										
†*Fine-Tuned Reference Ceiling (FFT)*										
Linear (FFT)	85.45±0.30	84.59±0.36	97.67±0.10	94.71±0.42	96.32±0.38	82.43±4.49	70.05±1.21	59.45±1.90	85.36±0.15	85.34±0.14
MLP (FFT)	85.76±0.83	85.04±0.79	97.65±0.68	94.47±0.76	96.39±0.19	85.25±1.67	70.66±0.83	60.39±2.32	85.79±0.60	85.75±0.61
Simple Baselines (Frozen)										
Linear	73.58±0.19	72.76±0.17	95.48±0.39	87.05±1.08	90.97±0.36	60.23±0.74	56.05±0.33	43.44±0.44	80.59±0.51	80.58±0.51
MLP	74.72±0.16	73.87±0.16	96.04±0.16	90.02±0.44	91.99±0.12	61.46±1.09	57.21±0.16	43.88±0.13	81.32±0.14	81.31±0.14
GNN Baselines (Frozen)										
TokenGraph	81.21±0.28	80.51±0.22	97.39±0.31	93.90±0.84	95.24±0.54	77.15±1.95	63.13±0.73	47.62±1.93	83.47±0.40	83.45±0.39
GLOT	80.53±0.58	79.74±0.39	97.53±0.41	94.13±0.98	95.18±0.31	**79.61**±2.23	63.92±0.77	52.29±2.42	83.20±0.32	83.16±0.32
AttnGraph	82.97±0.52	82.15±0.59	97.41±0.34	93.39±1.78	95.23±0.39	76.93±1.46	**67.15**±0.39	**55.01**±1.45	83.26±0.51	83.20±0.54
CayleyGCN	72.94±0.29	71.94±0.24	95.96±0.33	89.69±2.90	91.58±0.54	57.92±3.00	52.92±0.52	34.46±1.38	80.65±1.03	80.62±1.04
Our Approach (Frozen)										
HILT-GCN	**84.80**±0.59	**84.07**±0.60	**97.77**±0.30	**94.60**±1.11	**95.78**±0.21	79.50±1.55	66.92±1.26	52.29±2.00	**83.67**±1.00	**83.62**±0.99
Δ over Best Frozen Baseline	**+1.83**	**+1.92**	**+0.24**	**+0.47**	**+0.54**	−0.11	−0.23	−2.72	**+0.20**	**+0.17**
`Qwen3-Embedding-8B`										
†*Fine-Tuned Reference Ceiling (PEFT)*										
Linear (PEFT)	90.00±0.28	89.30±0.26	98.53±0.20	96.10±0.43	96.99±0.30	85.99±0.96	80.80±0.47	75.36±0.69	92.34±0.19	92.34±0.19
MLP (PEFT)	90.45±0.28	89.74±0.32	98.54±0.16	95.85±0.52	97.32±0.17	86.67±0.68	82.06±0.39	76.47±0.49	92.06±0.63	92.06±0.63
Simple Baselines (Frozen)										
Linear	86.71±0.11	85.76±0.13	98.16±0.08	94.55±0.16	95.88±0.13	79.32±1.07	73.45±0.41	65.77±0.38	90.57±0.14	90.57±0.15
MLP	87.38±0.11	86.41±0.11	98.29±0.04	94.96±0.33	95.87±0.10	78.56±1.19	75.04±0.31	67.20±0.68	90.23±0.18	90.22±0.18
GNN Baselines (Frozen)										
TokenGraph	87.98±0.24	87.20±0.21	98.02±0.35	94.70±1.19	95.86±0.36	76.18±3.19	74.36±0.68	65.97±1.35	88.85±0.33	88.84±0.33
GLOT	86.26±0.41	85.28±0.50	97.93±0.20	94.93±0.47	96.02±0.23	**84.05**±1.54	76.71±0.94	69.62±1.86	89.84±0.39	89.83±0.40
AttnGraph	86.91±0.24	85.95±0.22	97.76±0.16	94.07±0.48	95.88±0.44	77.46±4.10	75.40±1.48	68.02±1.65	89.30±0.45	89.29±0.46
CayleyGCN	87.76±0.21	86.99±0.23	97.89±0.10	94.12±0.31	95.37±0.17	67.57±2.02	73.15±0.23	64.25±0.32	88.47±0.71	88.45±0.73
Our Approach (Frozen)										
HILT-GCN	**90.05**±0.18	**89.38**±0.21	**98.39**±0.08	**95.35**±0.56	**96.75**±0.54	81.45±3.62	**78.08**±0.91	**69.82**±1.20	**91.55**±0.46	**91.54**±0.46
Δ over Best Frozen Baseline	**+2.07**	**+2.18**	**+0.10**	**+0.39**	**+0.73**	−2.60	**+1.37**	**+0.20**	**+0.98**	**+0.97**

†Embedders fine-tuned directly on task data (FFT for BERT; LoRA/PEFT for Qwen) to provide an empirical upper bound.

baselines generally perform on par, sometimes even better, than GNN baselines. Second, among the GNN baselines, there is no method that generally performs best in comparison to the other baselines, e. g., on BERT, AttnGraph performs better on 20NG, while GLOT beats it on R8, with TokenGraph beating it again on MR. Our model consistently performs on par with or better than GNN baselines regardless of the embedding model, showing clear benefits. CayleyGCN seems to struggle on BERT embeddings, as the numbers are much worse than the Qwen3 counterpart. We reason that by aggregating layer nodes, CayleyGCN ends up with a much smaller graph, thus limiting the effectiveness of message-passing. This effect is worsened by older embedding models, like BERT, having only 12 transformer layers, while modern ones like Qwen3 have around 32. There are some distinct outliers that cannot be explained by variance, like Macro-F1 on Ohsumed with BERT embeddings and Macro-F1 on R52 with Qwen3 embeddings, where the gap between the best compared model is upwards of 2.5 percentage points. We argue that this shift happens because softmax cross-entropy loss is biased towards Micro-F1 optimization, as it does not account for label frequency. Still, our approach generally improves the Macro-F1 metric alongside Micro-F1, benefiting both metrics. We argue that accessing features across multiple transformer layers helps the model align frozen representations for the downstream task, showing clear increases in evaluation performance when compared to existing GNN-based approaches.

Although our fine-tuned models serve only as a reference point, the metrics produced by our HILT-GCN with frozen embeddings still come close to fine-tuned performance, especially with Qwen3 embeddings for the 20NG, R8, R52 (Micro-F1 only), and MR datasets. Only for Ohsumed, we observe a larger gap between the two paradigms. This insight seems to imply that frozen embeddings can be aligned to downstream tasks much more effectively than previously anticipated.

Robustness Across Embedding Models (RQ 2). With only two embedding models, it is unclear whether our frozen embedding classification approach transfers well to other models. To answer that question and ensure that relative performance gains are not entirely dictated by the embedding quality of the underlying transformer, we run the same experimental setup as in RQ 1, but swapping the embedders. We choose `bert-large-uncased` as a stronger variant of the base model, as well as `roberta-base` and `roberta-large` as alternative encoder-based embedding models. Unlike Qwen 3-8B, which was trained on classification tasks, the generative Llama 3-8B model was not. Therefore, for decoder-based models, we additionally evaluate Llama 3-8B to investigate

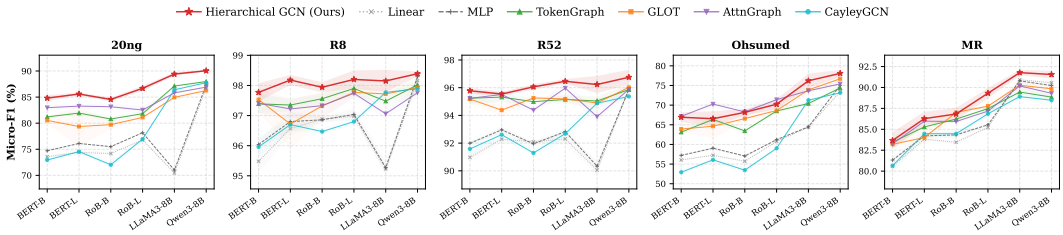


Figure 3: Comparison of Micro-F1 across benchmark datasets using different embedding models.

whether graph-based approaches can effectively leverage embeddings from generative models that were not optimized for classification.

As can be seen in Figure 3, the architectures remain relatively stable for different embedders, with larger embeddings producing better Micro-F1 scores. We see a continuation of the general trends observed for RQ 1 above. Another observation is that certain graph baselines, like GLOT, seem to struggle with certain embedding models, such as `bert-large-uncased`, where performance varies noticeably compared to the base model, hinting at instabilities during training. We assume that these instabilities stem from the threshold τ, where the default value of 0.6 might struggle with models where the cosine similarity between tokens is too close, introducing more edges than anticipated and oversmoothing the message-passing refinement of the node features. For smaller, encoder-based embeddings, it becomes evident that naive linear and MLP probing baselines generally cannot compete with graph-based approaches. This is regardless of the specific model, with the MR dataset having the smallest gap between the two paradigms. The Linear and MLP baselines can successfully exploit the Qwen3 embeddings; however, their performance drops when using Llama3 and smaller embedding models, with the exception of the MR dataset. We suspect that this is due to the binary nature of the MR dataset, possibly making it easier for models to align.

These insights imply that graph-based approaches are generally more stable for frozen feature alignment than naive linear baselines. Furthermore, we observe that Qwen3 is a strong embedder, as expected, as it performs best among all datasets except MR, where it falls behind Llama3 a bit.

Memory and Runtime Efficiency (RQ 3). To evaluate computational efficiency, we benchmark trainable parameter counts, peak training VRAM, and throughput on an NVIDIA H100 (80GB). While prior token-graph methods (Donabauer & Kruschwitz, 2025; Mantri et al., 2026) precompute and cache static graphs offline to benchmark only the GNN header without embedding model overhead, real-world inference is inductive by design. We therefore evaluate both an end-to-end inductive pipeline (w/o Caching) representing inductive computational overhead, and an embedding-cached setting (Cached) that stores the feature tensor $\mathbf{H}$ across epochs. While caching pre-constructed graphs would yield even higher throughput, caching $\mathbf{H}$ retains graph construction flexibility across configurations and establishes a more realistic, generalizable bound on efficiency. Experiments are conducted on 20NG, whose long sequence lengths impose the heaviest computational stress.

Table 2: **Efficiency comparison on 20NG.** Parameter counts (M), peak training VRAM (GB), and throughput (samples/sec) across full fine-tuning (FFT), LoRA, and HILT-GCN.

`bert-base-uncased`				
Configuration	**Trainable Params (M)**	**Peak VRAM (GB)**	**Throughput (samples/s)**	
			Train ↑	**Infer ↑**
Linear (FFT)	109.50	8.45	445.5	1612.2
HILT-GCN (w/o Caching)	0.83	12.84	565.3	895.1
HILT-GCN (Cached)	0.83	15.27	718.7	1179.6

`Qwen3-Embedding-8B`				
Configuration	**Trainable Params (M)**	**Peak VRAM (GB)**	**Throughput (samples/s)**	
			Train ↑	**Infer ↑**
Linear (PEFT / LoRA)	22.91	55.01	11.0	27.0
HILT-GCN (w/o Caching)	1.69	63.43	23.0	25.0
HILT-GCN (Cached)	1.69	49.37	44.3	90.2

As shown in Table 2, HILT-GCN reduces trainable parameters by 99.2% on BERT and 92.6% on Qwen3-8B, confining gradient tracking and optimizer states strictly to the graph header. This parameter efficiency translates directly into substantial training throughput gains on larger embedding models: on Qwen3-8B, uncached training throughput doubles LoRA (23.0 vs. 11.0 samples/s) and quadruples when cached (44.3 samples/s). Regarding memory allocation, HILT-GCN incurs higher peak VRAM on BERT because linear baselines immediately collapse sequences via single-vector pooling, whereas graph construction requires retaining the full token representation tensor $\mathbf{H}$ and assembling graph structures. At 8B scale, however, this activation overhead is amortized by the

Table 3: **Ablation on graph construction strategies.** All variants utilize our GNN backbone and differ solely in the graph construction mechanism. Results report Test-set Micro-F1 and Macro-F1 (mean ± std over 5 runs). **Bold** and underline denote best and second-best performance within each embedding backbone.

Graph Construction	20NG		R8		R52		Ohsumed		MR	
	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1
`bert-base-uncased`										
HILT-GCN (Ours)	**84.34**$_{\pm 0.51}$	**83.59**$_{\pm 0.57}$	**98.01**$_{\pm 0.22}$	94.67$_{\pm 0.65}$	**95.61**$_{\pm 0.65}$	**79.67**$_{\pm 2.67}$	66.82$_{\pm 1.20}$	51.60$_{\pm 2.13}$	**83.81**$_{\pm 0.79}$	**83.77**$_{\pm 0.79}$
TokenGraph	83.78$_{\pm 0.68}$	82.99$_{\pm 0.73}$	97.82$_{\pm 0.25}$	94.55$_{\pm 1.01}$	95.47$_{\pm 0.40}$	76.85$_{\pm 1.39}$	**67.87**$_{\pm 1.12}$	**56.10**$_{\pm 2.37}$	82.48$_{\pm 1.40}$	82.38$_{\pm 1.39}$
GLOT	83.45$_{\pm 0.50}$	82.66$_{\pm 0.61}$	97.59$_{\pm 0.33}$	94.17$_{\pm 0.79}$	95.16$_{\pm 0.18}$	76.72$_{\pm 2.44}$	67.79$_{\pm 0.93}$	55.34$_{\pm 1.36}$	81.65$_{\pm 0.32}$	81.64$_{\pm 0.33}$
AttnGraph	83.27$_{\pm 0.62}$	82.56$_{\pm 0.46}$	97.78$_{\pm 0.14}$	**94.72**$_{\pm 0.31}$	94.94$_{\pm 0.35}$	75.78$_{\pm 1.31}$	65.75$_{\pm 2.70}$	50.60$_{\pm 5.31}$	82.56$_{\pm 0.61}$	82.52$_{\pm 0.64}$
CayleyGCN	73.73$_{\pm 0.29}$	72.84$_{\pm 0.43}$	95.94$_{\pm 0.25}$	89.22$_{\pm 1.46}$	91.71$_{\pm 0.57}$	58.68$_{\pm 1.71}$	54.62$_{\pm 0.49}$	38.63$_{\pm 1.49}$	80.35$_{\pm 0.43}$	80.31$_{\pm 0.46}$
`Qwen3-Embedding-8B`										
HILT-GCN (Ours)	**90.01**$_{\pm 0.25}$	**89.40**$_{\pm 0.24}$	**98.36**$_{\pm 0.16}$	**95.33**$_{\pm 0.35}$	**96.65**$_{\pm 0.32}$	**85.34**$_{\pm 0.74}$	**78.35**$_{\pm 0.95}$	**70.01**$_{\pm 1.34}$	**91.35**$_{\pm 0.17}$	**91.35**$_{\pm 0.17}$
TokenGraph	89.17$_{\pm 0.37}$	88.51$_{\pm 0.36}$	97.83$_{\pm 0.31}$	93.92$_{\pm 1.07}$	96.33$_{\pm 0.39}$	82.85$_{\pm 2.02}$	76.99$_{\pm 0.89}$	69.53$_{\pm 1.18}$	90.80$_{\pm 0.20}$	90.80$_{\pm 0.20}$
GLOT	89.42$_{\pm 0.23}$	88.68$_{\pm 0.25}$	97.97$_{\pm 0.17}$	94.69$_{\pm 0.71}$	96.15$_{\pm 0.18}$	82.46$_{\pm 0.93}$	75.91$_{\pm 1.22}$	68.05$_{\pm 1.85}$	90.60$_{\pm 0.35}$	90.60$_{\pm 0.35}$
AttnGraph	88.94$_{\pm 0.23}$	88.14$_{\pm 0.24}$	97.76$_{\pm 0.25}$	94.01$_{\pm 0.44}$	95.93$_{\pm 0.27}$	79.69$_{\pm 4.04}$	75.06$_{\pm 0.28}$	66.59$_{\pm 0.45}$	89.63$_{\pm 0.91}$	89.62$_{\pm 0.91}$
CayleyGCN	88.00$_{\pm 0.19}$	87.18$_{\pm 0.20}$	97.65$_{\pm 0.12}$	93.53$_{\pm 0.34}$	95.47$_{\pm 0.27}$	76.63$_{\pm 3.71}$	73.56$_{\pm 0.53}$	65.48$_{\pm 0.91}$	89.41$_{\pm 0.07}$	89.41$_{\pm 0.07}$

embedding model's static weight memory. Inference throughput reflects a similar dynamic: while message passing introduces noticeable relative overhead on smaller embedding models, it reaches near-parity on Qwen3-8B (25.0 vs. 27.0 samples/s) where dense forward passes dominate runtime. Furthermore, because the embedding model remains entirely frozen, this decoupled design allows downstream resource optimizations such as intermediate layer pruning or representation dimension trimming without retraining the underlying model.

6 Ablations

We conduct systematic ablation studies to isolate the factors driving HILT-GCN's performance: (1) graph topology, (2) GNN architectural components and depth, and (3) global readout pooling. Consistent with our benchmark evaluations (Section 5), our base configuration is denoted as "Full Model (Tested)".

Ablation 1: Isolating graph construction performance. To confirm that HILT-GCN's gains are primarily from our graph topology rather than the GNN module, we substitute our graph constructor with four baselines while keeping the GNN module fixed (Table 3). Under `bert-base-uncased`, HILT-GCN achieves top performance across four of the five benchmarks, while TokenGraph remains competitive on Ohsumed. With larger `Qwen3-Embedding-8B` representations, HILT-GCN uniformly outperforms all baselines across every dataset (90.01% on 20NG and 78.35% on Ohsumed). Moreover, competing topologies also improve when paired with our GNN backbone (e.g., TokenGraph gains +1.19 on 20NG), demonstrating that while our GNN module is effective, our layer-token topology remains the primary driver of performance.

Table 4: **Ablation study on GNN architectural components, depth, and readout pooling** using frozen `Qwen3-Embedding-8B` representations. Performance is reported as Test Micro-F1 (%). "Full Model (Tested)" marks the primary baseline configuration evaluated in Section 5.

Configuration	GNN Components				Benchmark Datasets (Micro-F1)					Avg.
	K	Res	JK	Readout	20NG	R8	R52	Ohsumed	MR	
Full Model (Tested)	2	✓	✓	Mean || Max	90.05	98.39	96.75	78.08	91.55	90.96
Feature Preservation Mechanisms										
w/o Residuals	2	–	✓	Mean || Max	90.25	98.41	96.53	79.57	91.34	91.22
w/o Jumping Knowledge	2	✓	–	Mean || Max	89.93	98.43	96.82	78.91	91.34	91.09
w/o Res & JK	2	–	–	Mean || Max	89.81	98.64	97.06	80.26	91.25	91.40
Layer Depth Scaling										
Depth $K = 3$	3	✓	✓	Mean || Max	90.12	98.53	96.91	79.43	91.19	91.24
Depth $K = 4$	4	✓	✓	Mean || Max	89.93	98.24	96.52	79.85	91.17	91.14
Readout Pooling										
Global Mean Only	2	✓	✓	Mean	86.89	97.48	95.27	71.34	88.81	87.96
Global Max Only	2	✓	✓	Max	89.66	98.57	96.90	78.60	91.23	90.99

Ablation 2: GNN module configuration and depth. In Table 4 (Top/Middle), we evaluate the necessity of residual connections (Res) and JK, alongside scaling layer depth $K \in \{2, 3, 4\}$. Interestingly, omitting either Res or JK maintains competitive accuracy, while omitting both (*w/o Res & JK*) slightly improves overall performance (91.40% vs. 90.96%), with a notable boost on Ohsumed (80.26%). Scaling depth to $K = 3$ and $K = 4$ introduces negligible variation (91.24% and 91.14%). We suspect that our layer nodes inherently facilitate cross-layer feature retention, making explicit skip connections introduce optimization overhead. This implies that HILT-GCN should be able to leverage simpler architectures without such performance degradation.

Ablation 3: Readout mechanism. Table 4 (Bottom) examines the pooling components in isolation. Relying solely on Global Mean pooling leads to a steep performance drop across all benchmarks (87.96% vs. 90.96%, falling to 71.34% on Ohsumed), indicating that uniform node averaging dilutes class-discriminative token features. Conversely, Global Max pooling alone matches the full Mean $||$ Max formulation (90.99% vs. 90.96%). Because our layer tokens already aggregate broad document-level context during message passing, max pooling alone might already be sufficient for most benchmarks.

7 Related Work

Early approaches using GNNs for text classification are TextGCN (Yao et al., 2019) and BERT-GCN (Lin et al., 2021). They induced a massive, heterogeneous corpus graph containing both document and word nodes, with edges determined by global corpus statistics like TF-IDF and pointwise mutual information. These approaches were *transductive*, i. e., the text graph was induced from the entire corpus including the train and test set. However, text classification is inherently inductive (Galke et al., 2025) as newly arrived, unseen documents are to be classified. This is impractical for transductivegraphs as it requires injecting new nodes into the graph, recalculating global edge weights, and rerunning the message-passing algorithm over the entire corpus.

To overcome these issues, subsequent research shifted towards *inductive* graph construction. For example, TextING (Zhang et al., 2020) builds individual, localized word-occurrence graphs for each document independently, decoupling the test set from the train set. InducT-GCN (Wang et al., 2022) follows this inductive philosophy but still builds a global vocabulary co-occurrence graph without explicit document nodes, allowing new documents to be classified using this global structure. A highly relevant subset of inductive graph approaches focuses on transforming dense transformer token embeddings into sparse token-level graphs. The primary challenge lies in defining the edges between these token nodes. There exist integrated approaches such as GraphFormers (Yang et al., 2021) that incorporate GNN structures directly into the transformer architecture. Similarly, non-GNN architectures like AM-BERT (Yu et al., 2024) modify attention mechanisms within the backbone to capture multi-label dependencies. However, much recent work has shifted away from such highly coupled architectures, separating the embedder backbone from the GNN to allow greater scalability and modularity, e. g., Donabauer & Kruschwitz (2025); Mantri et al. (2026); Pimentel et al. (2026); Ulanovski et al. (2026), which provides the foundation for our token-graph experiments.

8 Conclusion, Limitations & Future Work

Conclusion. We provided an extensive comparison of token-level graph learning approaches for text classification and introduced our simple but effective HILT-GCN architecture. Our method shows strong text classification performance despite relying solely on frozen transformer embeddings. The approach enables an efficient training procedure without transformer backbone propagation and provides a lightweight classifier during inference. While our approach does not entirely close the gap between frozen and FFT/PEFT models, we substantially narrow the margin.

Limitations. Typical transformers like `bert-base-uncased` have a context window of 512 tokens, whereas modern baselines like `Qwen3-Embedding-8B` support $8,192$ tokens (with future models scaling further). Our experiments used BERT-level truncated inputs to control for context length and avoid scalability bottlenecks from all-to-all token interactions in GLOT and AttnGraph. Consequently, HILT-GCN may achieve even higher performance on large models if full inputs were used. Finally, for generative models like Qwen3, pretraining data contamination cannot be ruled out as a potential factor boosting benchmark scores.

9 AI USE STATEMENT

For tasks with required disclosure, we used Google Gemini to get feedback on our experimental setup and identify potential weaknesses; we did not use generative AI to design the methodology or for any other required tasks. For recommended disclosure tasks, we used Gemini, Claude, and ChatGPT to help write and edit code, and Gemini and Grammarly to improve readability and grammar. All AI-assisted code, feedback, and text edits were reviewed and checked by the authors. The authors take full responsibility for the final paper and code.

10 ETHICS STATEMENT

This work is methodological and evaluates HILT-GCN on established text classification benchmarks. All datasets are used solely for research and benchmarking purposes and in accordance with their original access conditions. The proposed method does not involve human-subject experiments, user profiling, or deployment in a decision-making system. As with other text classification models, potential risks include propagating biases present in pretrained encoders or benchmark datasets, and predictions should therefore be validated before use in high-stakes applications.

11 REPRODUCIBILITY STATEMENT

Our code is available on Open Review and will be publicly available at `<anonymized>`. Section 3 describes the conceptual architecture of our approach, while Section 4 describes the experimental setup. A thorough description of baseline parameters and hyperparameter configuration is available in Appendix A. Datasets and preprocessing steps are described within Appendix B.

12 ACKNOWLEDGMENTS

Anonymized for double-blind review.

REFERENCES

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Ana Cardoso-Cachopo. *Improving Methods for Single-label Text Categorization*. PhD thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisbon, Portugal, 2007.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pp. 1725–1735. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20v.html.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and

Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Gregor Donabauer and Udo Kruschwitz. Token-level graphs for short text classification. In Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonellotto (eds.), *Advances in Information Retrieval - 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6-10, 2025, Proceedings, Part III*, Lecture Notes in Computer Science, pp. 427–436. Springer, 2025. doi: 10.1007/978-3-031-88714-7_42. URL https://doi.org/10.1007/978-3-031-88714-7_42.

Lukas Galke and Ansgar Scherp. Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 4038–4051. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.279. URL https://doi.org/10.18653/v1/2022.acl-long.279.

Lukas Galke, Ansgar Scherp, Andor Diera, Fabian Karl, Bao Xin Lin, Bhakti Khera, Tim Meuser, and Tushar Singhal. Are we really making much progress in text classification? a comparative review, 2025. URL https://arxiv.org/abs/2204.03954.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pp. 630–645. Springer, 2016. doi: 10.1007/978-3-319-46493-0_38. URL https://doi.org/10.1007/978-3-319-46493-0_38.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

William R. Hersh, Chris Buckley, T. J. Leone, and David H. Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In W. Bruce Croft and C. J. van Rijsbergen (eds.), *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pp. 192–201. ACM/Springer, 1994. doi: 10.1007/978-1-4471-2099-5_20. URL https://doi.org/10.1007/978-1-4471-2099-5_20.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 328–339. Association for Computational Linguistics, 2018. doi: 10.18653/V1/P18-1031. URL https://aclanthology.org/P18-1031/.

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. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *CoRR*, abs/2312.03732, 2023. doi: 10.48550/ARXIV.2312.03732. URL https://doi.org/10.48550/arXiv.2312.03732.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham M. Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/c32319f4868da7613d78af9993100e42-Abstract-Conference.html.

Ken Lang. Newsweeder: Learning to filter netnews. In Armand Prieditis and Stuart Russell (eds.), *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pp. 331–339. Morgan Kaufmann, 1995. doi: 10.1016/B978-1-55860-377-6.50048-7. URL https://doi.org/10.1016/b978-1-55860-377-6.50048-7.

Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 9266–9275. IEEE, 2019. doi: 10.1109/ICCV.2019.00936. URL https://doi.org/10.1109/ICCV.2019.00936.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. Bertgcn: Transductive text classification by combining GNN and BERT. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, Findings of ACL, pp. 1456–1462. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.FINDINGS-ACL.126. URL https://doi.org/10.18653/v1/2021.findings-acl.126.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Llama. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Krishna Sri Ipsit Mantri, Carola-Bibiane Schönlieb, Zorah Lähner, and Moshe Eliasof. Towards improved sentence representations using token graphs. *CoRR*, abs/2603.03389, 2026. doi: 10.48550/ARXIV.2603.03389. URL https://doi.org/10.48550/arXiv.2603.03389.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pp. 2006–2029. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EACL-MAIN.148. URL https://doi.org/10.18653/v1/2023.eacl-main.148.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer (eds.), *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pp. 115–124. The Association for Computer Linguistics, 2005. doi: 10.3115/1219840.1219855. URL https://aclanthology.org/P05-1015/.

João Pimentel, Joana Amorim, and Frank Rudzicz. A straightforward approach to construct 'lightweight' token graphs from transformers. *Computational Linguistics*, pp. 1–52, 03 2026. ISSN 0891-2017. doi: 10.1162/COLI.a.618. URL https://doi.org/10.1162/COLI.a.618.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4593–4601. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1452. URL https://doi.org/10.18653/v1/p19-1452.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Tom Ulanovski, Eyal Blyachman, and Maya Bechler-Speicher. Improving llm final representations with inter-layer geometry. *arXiv preprint arXiv:2603.22665*, 2026.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Kunze Wang, Soyeon Caren Han, and Josiah Poon. Induct-gcn: Inductive graph convolutional networks for text classification. In *26th International Conference on Pattern Recognition, ICPR 2022, Montreal, QC, Canada, August 21-25, 2022*, pp. 1243–1249. IEEE, 2022. doi: 10.1109/ICPR56361.2022.9956075. URL https://doi.org/10.1109/ICPR56361.2022.9956075.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020. URL http://proceedings.mlr.press/v119/xiong20b.html.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5449–5458. PMLR, 2018. URL http://proceedings.mlr.press/v80/xu18c.html.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 28798–28810, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f18a6d1cde4b205199de8729a6637b42-Abstract.html.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 7370–7377. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33017370. URL https://doi.org/10.1609/aaai.v33i01.33017370. Dataset files retrieved from: https://github.com/yao8839836/text_gcn.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. aaai 2019, 2024. URL https://github.com/yao8839836/text_gcn.

Zhiyi Yu, Hong Li, and Jialin Feng. Enhancing text classification with attention matrices based on BERT. *Expert Syst. J. Knowl. Eng.*, 41(3), 2024. doi: 10.1111/EXSY.13512. URL https://doi.org/10.1111/exsy.13512.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176, 2025. doi: 10.48550/ARXIV.2506.05176. URL https://doi.org/10.48550/arXiv.2506.05176.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. Every document owns its structure: Inductive text classification via graph neural networks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 334–339. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.31. URL https://doi.org/10.18653/v1/2020.acl-main.31.

APPENDIX

A EXTENDED IMPLEMENTATION DETAILS

A.1 ARCHITECTURAL CONFIGURATIONS

To isolate topological inductive bias from parameter scale, all GNN encoders are aligned to a 2-layer message-passing pipeline with hidden dimension $d_{\text{gnn}} = 256$. Table 5 details the topological construction, edge attributes, routing mechanisms, and pooling operators across all evaluated graph approaches.

Table 5: Architectural and structural specifications across token-level graph methods.

Hyperparameter / Component	TokenGraph	GLOT	AttnGraph	CayleyGCN	HILT-GCN (Ours)
Graph Construction & Topology					
Node Universe ($\mathcal{V}$)	Final-layer tokens	Final-layer tokens	Final-layer tokens	Layer token averages	Multi-layer tokens + Virtual hubs
Total Node Cardinality ($\lvert\mathcal{V}\rvert$)	T	T	$\leq T$ (filtered)	$L + \lvert\mathcal{V}_{\text{virtual}}\rvert$	$LT + L + 1$
Graph Topology	Sliding window	Dynamic cosine similarity	Sparse multi-head attention	$\mathrm{SL}(2, \mathbb{Z}_n)$ Cayley expander	Hierarchical tree
Sparsification Criterion	$\lvert i - j\rvert \leq 2$	Threshold $\tau = 0.60$	Quantile $q = 0.90$	Group-theoretic generator	Hub-star projection
Edge Weights	Unweighted	Cosine scalar	Quantized attention scalar	Unweighted	Unweighted
Auxiliary / Virtual Nodes	None	None	None	Group order zero-nodes	L layer hubs + 1 global root
GNN Message-Passing Pipeline					
Message-Passing Operator	GAT	GAT	GATv2	GCN	GCN
Attention Heads (K)	1	1	4	–	–
Message-Passing Layers	2	2	2	2	2
Hidden Dimension (d_{gnn})	256	256	256	256	256
Dropout Rate (p)	0.50	0.00	0.30	0.10	0.10
Intermediate Layer Aggregation	Final layer	Concat Jumping Knowledge	Final layer	Layer-wise projection	Pre-LN Residuals + Concat JK
Graph Readout & Prediction Head					
Readout Masking	Full graph	Full graph	Full graph	Non-virtual nodes	Token nodes only ($\mathcal{V}_{\text{tok}}$)
Readout Operator	Global token-mean	Attentional score pooling	Global token-mean	Global mean	Symmetric $\mathrm{mean} \,\|\, \max$
Classification Head	Linear	Linear	Linear	Linear	2-layer MLP (LayerNorm, GELU)

A.2 HYPERPARAMETER SEARCH SPACE & OPTIMIZATION

Optimization Objective & Schedule. All models optimize standard categorical cross-entropy over softmax class logits:

$$\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{C} y_c \log \hat{y}_c = -\sum_{c=1}^{C} y_c \log \left(\frac{\exp(z_c)}{\sum_{j=1}^{C} \exp(z_j)} \right).$$

Optimization follows a constant learning rate schedule with a linear warmup over $\max(1 \text{ epoch}, 500 \text{ steps})$. Early stopping monitors the validation $\mathrm{Softmax}$ cross-entropy loss across all regimes.

Fine-Tuning Regimes. To evaluate parameter adaptation against frozen representations, we decouple the learning rates of the transformer backbone and the classification head (Howard & Ruder, 2018):

- **Full Fine-Tuning (FFT; Moderate-Scale Embedders):** For backbones up to 355M parameters, all weights are optimized end-to-end. We search backbone learning rates over $\{1 \times 10^{-5}, 3 \times 10^{-5}, 3.5 \times 10^{-5}, 5 \times 10^{-5}\}$, with early stopping patience set to 10 epochs.
- **PEFT via rsLoRA (Large-Scale 8B Embedders):** For 8B backbones, we apply Rank-Stabilized LoRA (rsLoRA) (Kalajdzievski, 2023) across all linear layers. We fix rank $r = 16$, scaling factor $\alpha = 16$, adapter dropout to 0.05, and search adapter learning rates within $\{1 \times 10^{-4}, 2 \times 10^{-4}\}$ with an early stopping patience of 5 epochs.

Across all three regimes (Frozen, FFT, and PEFT), the classification head learning rate is tuned across the complete four-magnitude grid $\{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$. Table 6 summarizes the exact parameter configuration.

A.3 HARDWARE INFRASTRUCTURE & COMPUTATIONAL REPRODUCIBILITY

All primary benchmarks were executed on single-GPU workstations equipped with either a single NVIDIA A100 (40GB) or a single NVIDIA H100 (80GB) GPU. To ensure strict, unconfounded

Table 6: Hyperparameter specifications and optimization search grids across experimental regimes.

Experimental Scope	Hyperparameter	Value / Grid Search Range
General Optimization (All Setups)		
	Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$)
	Weight Decay	0.01
	Effective Batch Size	32
	Gradient Accumulation Steps	1 (Moderate-Scale Embedders), 4 (Large-Scale Embedders)
	Per-Device Batch Size	32 (Moderate-Scale Embedders), 8 (Large-Scale Embedders)
	Sequence Length Truncation	512 tokens
	Warmup Schedule	Linear, $\max(1 \text{ epoch}, 500 \text{ steps})$
	Early Stopping Criterion	Validation $\mathrm{Softmax}$ Cross-Entropy Loss
	Random Seeds	$\{1, 2, 3, 4, 5\}$
Non-Structural Baselines (Linear Probe & 2-Layer MLP)		
	Representation (Encoder Backbones)	Global token-mean pooling (BERT, RoBERTa)
	Representation (Decoder Backbones)	Final token / `[EOS]` embedding (LLaMA-3, Qwen3)
Frozen Structural Probing (Primary Evaluation)		
	Backbone Status	Frozen
	Head Learning Rate (η_{head})	$\{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$
	Early Stopping Patience	20 epochs (Moderate-Scale), 10 epochs (Large-Scale)
Full Fine-Tuning (FFT; Moderate-Scale Embedders)		
	Backbone Learning Rate (η_{bb})	$\{1 \times 10^{-5}, 3 \times 10^{-5}, 3.5 \times 10^{-5}, 5 \times 10^{-5}\}$
	Head Learning Rate (η_{head})	$\{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$
	Early Stopping Patience	10 epochs
PEFT via rsLoRA (Large-Scale 8B Embedders)		
	Target Modules	All linear layers
	LoRA Rank (r) / Scaling Factor (α)	$r = 16, \alpha = 16$
	LoRA Dropout	0.05
	Adapter Learning Rate (η_{lora})	$\{1 \times 10^{-4}, 2 \times 10^{-4}\}$
	Head Learning Rate (η_{head})	$\{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$
	Early Stopping Patience	5 epochs

hardware comparability across runs, all comparative ablation experiments were executed exclusively on a single NVIDIA H100 GPU under an identical software stack.

To guarantee reproducibility, our code repository provides a declarative `pixi.toml` project configuration alongside an exact `pixi.lock` lockfile, using CUDA 13.0, and PyTorch 2.11.

B Dataset Details & Preprocessing

To evaluate the efficacy of frozen embeddings, we choose 5 established single-label text classification benchmarks. They vary across document length, label granularity, and domain vocabulary, spanning news categorization, biomedical literature, and sentiment analysis:

- **20NG** (Lang, 1995): The 20 Newsgroups (20NG) benchmark consisting of Usenet message postings partitioned across 20 distinct discussion topics, characterized by conversational text and high sequence length variance.
- **R8 & R52** (Cardoso-Cachopo, 2007): Two standard subsets of the Reuters-21578 news corpus categorized into 8 and 52 mutually exclusive economic and financial topic classes, respectively.
- **Ohsumed** (Hersh et al., 1994): A domain-specific biomedical corpus comprising clinical paper abstracts from the MEDLINE database, indexed across 23 cardiovascular disease categories.
- **MR** (Pang & Lee, 2005): Movie Review (MR) is a sentiment analysis benchmark of reviews with binary labels (positive/negative), characterized by very short sequence lengths ($\approx$ 27 tokens on average).

Data Provenance & Preprocessing Pipeline. Raw corpus files are obtained directly from the canonical TextGCN distribution repository (Yao et al., 2024). Once downloaded and placed into the corresponding project data directories, our preprocessing scripts (located in `preprocessing/`) are executed to parse and partition each corpus. For benchmarks lacking an official validation set, the pipeline deterministically extracts a stratified 10% split from the training data (marked with an asterisk in Table 7).

Table 7: Dataset characteristics for evaluated single-label text classification benchmarks. Average token counts are computed using the `bert-base-uncased` tokenizer (reported as mean ± std). Provenance format: [Original Publication] / [Distribution Source].

Dataset	# Classes	# Total Docs	Train	Val	Test	Avg. Tokens	Provenance
20NG	20	18,846	10,182	1,132*	7,532	552.82 ± 2047.39	Lang (1995) / Yao et al. (2024)
R8	8	7,674	4,936	549*	2,189	120.91 ± 127.63	Cardoso-Cachopo (2007) / Yao et al. (2024)
R52	52	9,100	5,878	654*	2,568	127.90 ± 132.86	Cardoso-Cachopo (2007) / Yao et al. (2024)
Ohsumed	23	7,400	3,021	336*	4,043	287.11 ± 122.66	Hersh et al. (1994) / Yao et al. (2024)
MR	2	10,622	6,397	711*	3,554	27.39 ± 11.33	Pang & Lee (2005) / Yao et al. (2024)

* Indicates a stratified 10% validation split partitioned from the original training set.

The pipeline standardizes every dataset into three distinct split files: `train.jsonl`, `val.jsonl`, and `test.jsonl`. Each line represents an individual document record formatted as a JSON object with two fields:

- `"text"`: A string containing the document text.
- `"labels"`: A list of strings containing the ground-truth cleartext class labels.

For the single-label benchmarks investigated in this work, the `"labels"` list contains exactly one label string per entry. This schema exists to ensure compatibility with future multi-label experiments.

C Extended Results

We provide two detailed tables summarizing the complete runs used for the aggregated table and figure for RQ1 and RQ2. These tables were too large to reasonably contain in the main paper. Table 8 contains all frozen baselines, while Table 9 contains additional fine-tuned runs for our additional embedding models. General insights from Section 5 remain consistent. Llama3 being generally strong on the MR dataset becomes more visible. When fine-tuning Llama3, the model is actually able to catch up in performance, often having the second-strongest numbers, narrowing the gap between Qwen3 noticeably while overall still being weaker.

D Comparison of our HILT-GCN Method with best Baseline and Best Results on Text Classification from the Literature

Table 10 provides a comparison of our proposed HILT-GCN with the best frozen embeddings (Qwen 3) and a fine-tuned GCN header to our fully fine-tuned baseline using Qwen 3 embeddings and the best and second best numbers (of any architecture) from the survey by Galke et al. (2025) that have been reported in the literature.

The results show the competitiveness of our approach, reaching performance close to fully fine-tuned models. Note that the results for "HILT-GCN (our) - *frozen* Qwen 3 embedding with GNN" as well as the "Fully fine-tuned Qwen 3 embedding baseline (also our)" are each a single model. While the results for the best and second fine-tuned model from the literature, according to Galke et al. (2025), are various papers ranging from a specialized GNN architecture (20NG), an augmented RoBERTa model (R8 and R52), a model consisting of an ensemble of seven Llama 2 (Ohsumed), and a GPT-3.5 with reasoning, voting, and an NLI-fine-tuned RoBERTa model for in-context shot selection (MR).

Note that in single-label classification, one has, per instance, one single true label from a set of classes. As such, Micro-F1 measures the same as accuracy Galke & Scherp (2022), as the latter is often reported in the literature.

Table 8: Evaluation results on five benchmark datasets with *frozen embedders*. Test set performance is reported using the learning rate that achieved the highest Micro-F1 on the validation set. Standard deviation across 5 runs is reported in subscript. The best result per column is in **bold**, and the second best is underlined.

Architecture	Backbone	20NG		R8		R52		Ohsumed		MR	
		Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1
Simple Baselines											
Linear	bert-base-uncased	73.58$_{\pm 0.19}$	72.76$_{\pm 0.17}$	95.48$_{\pm 0.39}$	87.05$_{\pm 1.08}$	90.97$_{\pm 0.36}$	60.23$_{\pm 0.74}$	56.05$_{\pm 0.33}$	43.44$_{\pm 0.44}$	80.59$_{\pm 0.51}$	80.58$_{\pm 0.51}$
	bert-large-uncased	74.40$_{\pm 0.61}$	73.56$_{\pm 0.61}$	96.57$_{\pm 0.24}$	91.14$_{\pm 0.54}$	92.28$_{\pm 0.22}$	62.12$_{\pm 0.79}$	57.24$_{\pm 0.15}$	45.66$_{\pm 0.21}$	83.82$_{\pm 0.32}$	83.82$_{\pm 0.32}$
	roberta-base	74.18$_{\pm 0.11}$	73.22$_{\pm 0.14}$	96.88$_{\pm 0.10}$	92.14$_{\pm 0.32}$	92.10$_{\pm 0.11}$	60.05$_{\pm 0.85}$	55.71$_{\pm 0.20}$	43.38$_{\pm 0.13}$	83.45$_{\pm 0.21}$	83.45$_{\pm 0.21}$
	roberta-large	76.80$_{\pm 0.20}$	75.97$_{\pm 0.22}$	96.98$_{\pm 0.09}$	90.94$_{\pm 0.38}$	92.30$_{\pm 0.42}$	60.48$_{\pm 1.01}$	60.77$_{\pm 0.16}$	48.98$_{\pm 0.21}$	85.22$_{\pm 0.12}$	85.21$_{\pm 0.12}$
	Meta-Llama-3-8B	70.45$_{\pm 0.25}$	69.64$_{\pm 0.27}$	95.23$_{\pm 0.15}$	86.94$_{\pm 0.42}$	90.05$_{\pm 0.30}$	55.32$_{\pm 1.40}$	64.37$_{\pm 0.14}$	55.33$_{\pm 0.34}$	90.92$_{\pm 0.10}$	90.91$_{\pm 0.10}$
	Qwen3-Embedding-8B	86.71$_{\pm 0.11}$	85.76$_{\pm 0.13}$	98.16$_{\pm 0.08}$	94.55$_{\pm 0.16}$	95.88$_{\pm 0.13}$	79.32$_{\pm 1.07}$	73.45$_{\pm 0.41}$	65.77$_{\pm 0.38}$	90.57$_{\pm 0.14}$	90.57$_{\pm 0.15}$
MLP	bert-base-uncased	74.72$_{\pm 0.16}$	73.87$_{\pm 0.16}$	96.04$_{\pm 0.16}$	90.02$_{\pm 0.44}$	91.99$_{\pm 0.12}$	61.46$_{\pm 1.09}$	57.21$_{\pm 0.16}$	43.88$_{\pm 0.13}$	81.32$_{\pm 0.14}$	81.31$_{\pm 0.14}$
	bert-large-uncased	76.11$_{\pm 0.12}$	75.23$_{\pm 0.14}$	96.80$_{\pm 0.34}$	91.30$_{\pm 0.97}$	92.96$_{\pm 0.24}$	63.75$_{\pm 1.25}$	59.04$_{\pm 0.37}$	46.36$_{\pm 0.36}$	84.21$_{\pm 0.23}$	84.21$_{\pm 0.23}$
	roberta-base	75.50$_{\pm 0.14}$	74.55$_{\pm 0.14}$	96.86$_{\pm 0.10}$	91.76$_{\pm 0.87}$	91.95$_{\pm 0.15}$	56.88$_{\pm 2.31}$	57.05$_{\pm 0.24}$	43.42$_{\pm 0.68}$	84.33$_{\pm 0.41}$	84.33$_{\pm 0.41}$
	roberta-large	78.17$_{\pm 0.11}$	77.21$_{\pm 0.13}$	97.04$_{\pm 0.19}$	91.35$_{\pm 0.46}$	92.83$_{\pm 0.16}$	57.78$_{\pm 1.23}$	61.21$_{\pm 0.31}$	48.63$_{\pm 0.18}$	85.54$_{\pm 0.08}$	85.54$_{\pm 0.08}$
	Meta-Llama-3-8B	71.01$_{\pm 0.14}$	70.30$_{\pm 0.18}$	95.27$_{\pm 0.22}$	87.15$_{\pm 0.94}$	90.34$_{\pm 0.36}$	56.22$_{\pm 1.02}$	64.47$_{\pm 0.13}$	55.23$_{\pm 0.35}$	90.79$_{\pm 0.24}$	90.79$_{\pm 0.24}$
	Qwen3-Embedding-8B	87.38$_{\pm 0.11}$	86.41$_{\pm 0.11}$	98.29$_{\pm 0.04}$	94.96$_{\pm 0.33}$	95.87$_{\pm 0.10}$	78.56$_{\pm 1.19}$	75.04$_{\pm 0.31}$	67.20$_{\pm 0.68}$	90.23$_{\pm 0.18}$	90.22$_{\pm 0.18}$
GNN Graph Baselines											
TokenGraph	bert-base-uncased	81.21$_{\pm 0.28}$	80.51$_{\pm 0.22}$	97.39$_{\pm 0.31}$	93.90$_{\pm 0.84}$	95.24$_{\pm 0.54}$	77.15$_{\pm 1.95}$	63.13$_{\pm 0.73}$	47.62$_{\pm 1.93}$	83.47$_{\pm 0.40}$	83.45$_{\pm 0.39}$
	bert-large-uncased	81.93$_{\pm 0.50}$	81.08$_{\pm 0.44}$	97.35$_{\pm 0.21}$	93.62$_{\pm 0.94}$	95.32$_{\pm 0.33}$	76.38$_{\pm 3.63}$	66.36$_{\pm 0.59}$	52.80$_{\pm 1.74}$	85.28$_{\pm 0.35}$	85.28$_{\pm 0.35}$
	roberta-base	80.80$_{\pm 0.53}$	79.81$_{\pm 0.50}$	97.56$_{\pm 0.19}$	93.79$_{\pm 0.52}$	94.99$_{\pm 0.22}$	75.03$_{\pm 1.53}$	63.45$_{\pm 0.87}$	48.05$_{\pm 2.49}$	86.22$_{\pm 0.30}$	86.20$_{\pm 0.30}$
	roberta-large	81.82$_{\pm 0.53}$	80.64$_{\pm 0.73}$	97.90$_{\pm 0.28}$	94.71$_{\pm 0.96}$	95.15$_{\pm 0.31}$	70.33$_{\pm 1.91}$	68.51$_{\pm 0.46}$	54.89$_{\pm 2.15}$	87.47$_{\pm 0.05}$	87.47$_{\pm 0.05}$
	Meta-Llama-3-8B	87.12$_{\pm 0.22}$	86.24$_{\pm 0.22}$	97.48$_{\pm 0.17}$	92.84$_{\pm 0.97}$	95.04$_{\pm 0.31}$	72.15$_{\pm 3.53}$	70.41$_{\pm 0.47}$	61.69$_{\pm 2.30}$	89.45$_{\pm 0.67}$	89.44$_{\pm 0.68}$
	Qwen3-Embedding-8B	87.98$_{\pm 0.24}$	87.20$_{\pm 0.21}$	98.02$_{\pm 0.35}$	94.70$_{\pm 1.19}$	95.86$_{\pm 0.36}$	76.18$_{\pm 3.19}$	74.36$_{\pm 0.68}$	65.97$_{\pm 1.35}$	88.85$_{\pm 0.33}$	88.84$_{\pm 0.33}$
GLOT	bert-base-uncased	80.53$_{\pm 0.58}$	79.74$_{\pm 0.39}$	97.53$_{\pm 0.41}$	94.13$_{\pm 0.98}$	95.18$_{\pm 0.31}$	79.61$_{\pm 2.23}$	63.92$_{\pm 0.77}$	52.29$_{\pm 2.42}$	83.20$_{\pm 0.32}$	83.16$_{\pm 0.32}$
	bert-large-uncased	79.36$_{\pm 3.97}$	78.29$_{\pm 3.85}$	96.72$_{\pm 0.82}$	92.43$_{\pm 2.36}$	94.38$_{\pm 0.44}$	76.97$_{\pm 1.75}$	64.63$_{\pm 3.14}$	53.63$_{\pm 5.97}$	84.02$_{\pm 1.03}$	84.01$_{\pm 1.04}$
	roberta-base	79.72$_{\pm 0.82}$	79.05$_{\pm 0.85}$	97.32$_{\pm 0.46}$	93.53$_{\pm 0.92}$	95.26$_{\pm 0.12}$	83.07$_{\pm 1.05}$	66.58$_{\pm 0.94}$	53.28$_{\pm 2.83}$	87.01$_{\pm 0.36}$	87.00$_{\pm 0.35}$
	roberta-large	81.10$_{\pm 0.25}$	80.14$_{\pm 0.29}$	97.77$_{\pm 0.15}$	94.44$_{\pm 0.54}$	95.19$_{\pm 0.28}$	79.77$_{\pm 2.17}$	68.63$_{\pm 1.11}$	59.55$_{\pm 1.96}$	87.77$_{\pm 0.10}$	87.77$_{\pm 0.10}$
	Meta-Llama-3-8B	84.95$_{\pm 0.32}$	83.70$_{\pm 0.31}$	97.72$_{\pm 0.23}$	94.17$_{\pm 0.81}$	94.90$_{\pm 0.48}$	72.34$_{\pm 1.64}$	73.88$_{\pm 1.10}$	66.77$_{\pm 1.94}$	90.27$_{\pm 0.39}$	90.27$_{\pm 0.39}$
	Qwen3-Embedding-8B	86.26$_{\pm 0.41}$	85.28$_{\pm 0.50}$	97.93$_{\pm 0.20}$	94.93$_{\pm 0.47}$	96.02$_{\pm 0.23}$	**84.05**$_{\pm 1.54}$	76.71$_{\pm 0.94}$	69.62$_{\pm 1.86}$	89.84$_{\pm 0.39}$	89.83$_{\pm 0.40}$
AttnGraph	bert-base-uncased	82.97$_{\pm 0.52}$	82.15$_{\pm 0.59}$	97.41$_{\pm 0.34}$	93.39$_{\pm 1.78}$	95.23$_{\pm 0.39}$	76.93$_{\pm 1.46}$	67.15$_{\pm 0.39}$	55.01$_{\pm 1.45}$	83.26$_{\pm 0.51}$	83.20$_{\pm 0.54}$
	bert-large-uncased	82.50$_{\pm 1.25}$	81.52$_{\pm 1.35}$	97.56$_{\pm 0.23}$	94.51$_{\pm 0.41}$	95.37$_{\pm 0.44}$	77.81$_{\pm 2.26}$	68.41$_{\pm 1.56}$	57.14$_{\pm 3.90}$	86.23$_{\pm 0.27}$	86.22$_{\pm 0.27}$
	roberta-base	82.69$_{\pm 0.46}$	81.85$_{\pm 0.57}$	97.33$_{\pm 0.39}$	92.36$_{\pm 1.54}$	94.38$_{\pm 0.14}$	67.11$_{\pm 0.37}$	68.28$_{\pm 1.48}$	54.91$_{\pm 4.47}$	85.93$_{\pm 0.22}$	85.91$_{\pm 0.21}$
	roberta-large	82.52$_{\pm 0.74}$	81.65$_{\pm 0.83}$	97.68$_{\pm 0.49}$	94.01$_{\pm 1.58}$	95.32$_{\pm 0.42}$	76.79$_{\pm 3.92}$	69.50$_{\pm 1.64}$	57.23$_{\pm 3.64}$	87.08$_{\pm 0.59}$	87.05$_{\pm 0.61}$
	Meta-Llama-3-8B	86.13$_{\pm 0.35}$	85.12$_{\pm 0.39}$	97.23$_{\pm 0.21}$	92.35$_{\pm 0.81}$	94.50$_{\pm 0.20}$	73.64$_{\pm 2.00}$	73.52$_{\pm 0.48}$	65.93$_{\pm 1.38}$	90.37$_{\pm 0.47}$	90.37$_{\pm 0.47}$
	Qwen3-Embedding-8B	86.91$_{\pm 0.24}$	85.95$_{\pm 0.22}$	97.76$_{\pm 0.16}$	94.07$_{\pm 0.48}$	95.88$_{\pm 0.44}$	77.46$_{\pm 4.10}$	75.40$_{\pm 1.48}$	68.02$_{\pm 1.65}$	89.30$_{\pm 0.45}$	89.29$_{\pm 0.46}$
CayleyGCN	bert-base-uncased	72.94$_{\pm 0.29}$	71.94$_{\pm 0.24}$	95.96$_{\pm 0.33}$	89.69$_{\pm 2.90}$	91.58$_{\pm 0.54}$	57.92$_{\pm 3.00}$	52.92$_{\pm 0.52}$	34.46$_{\pm 1.38}$	80.65$_{\pm 1.03}$	80.62$_{\pm 1.04}$
	bert-large-uncased	74.55$_{\pm 0.42}$	73.69$_{\pm 0.40}$	96.70$_{\pm 0.19}$	91.46$_{\pm 0.33}$	92.61$_{\pm 0.28}$	62.89$_{\pm 1.41}$	56.08$_{\pm 0.68}$	37.36$_{\pm 1.95}$	84.47$_{\pm 0.40}$	84.43$_{\pm 0.41}$
	roberta-base	72.03$_{\pm 0.59}$	71.00$_{\pm 0.64}$	96.47$_{\pm 0.10}$	88.55$_{\pm 1.02}$	91.29$_{\pm 0.25}$	53.13$_{\pm 1.76}$	53.45$_{\pm 0.50}$	34.00$_{\pm 1.51}$	84.47$_{\pm 0.37}$	84.44$_{\pm 0.39}$
	roberta-large	76.95$_{\pm 0.19}$	76.03$_{\pm 0.19}$	96.80$_{\pm 0.13}$	90.67$_{\pm 0.84}$	92.68$_{\pm 0.45}$	56.28$_{\pm 2.22}$	59.01$_{\pm 0.47}$	41.61$_{\pm 1.33}$	86.85$_{\pm 0.44}$	86.84$_{\pm 0.46}$
	Meta-Llama-3-8B	86.40$_{\pm 0.28}$	85.57$_{\pm 0.25}$	97.77$_{\pm 0.07}$	93.63$_{\pm 0.33}$	94.88$_{\pm 0.37}$	66.47$_{\pm 3.07}$	71.26$_{\pm 0.37}$	62.71$_{\pm 0.74}$	88.91$_{\pm 0.34}$	88.90$_{\pm 0.36}$
	Qwen3-Embedding-8B	87.76$_{\pm 0.21}$	86.99$_{\pm 0.23}$	97.89$_{\pm 0.10}$	94.12$_{\pm 0.31}$	95.37$_{\pm 0.17}$	67.57$_{\pm 2.02}$	73.15$_{\pm 0.23}$	64.25$_{\pm 0.32}$	88.47$_{\pm 0.71}$	88.45$_{\pm 0.73}$
Our Approach											
HILT-GCN (Ours)	bert-base-uncased	84.80$_{\pm 0.59}$	84.07$_{\pm 0.60}$	97.77$_{\pm 0.30}$	94.60$_{\pm 1.11}$	95.78$_{\pm 0.21}$	79.50$_{\pm 1.55}$	66.92$_{\pm 1.26}$	52.29$_{\pm 2.00}$	83.67$_{\pm 1.00}$	83.62$_{\pm 0.99}$
	bert-large-uncased	85.59$_{\pm 0.28}$	84.90$_{\pm 0.29}$	98.18$_{\pm 0.17}$	95.04$_{\pm 0.73}$	95.54$_{\pm 0.15}$	71.25$_{\pm 3.45}$	66.54$_{\pm 1.90}$	51.01$_{\pm 4.81}$	86.27$_{\pm 0.84}$	86.26$_{\pm 0.84}$
	roberta-base	84.55$_{\pm 0.36}$	83.93$_{\pm 0.37}$	97.94$_{\pm 0.17}$	94.53$_{\pm 0.56}$	96.06$_{\pm 0.20}$	79.06$_{\pm 2.03}$	68.16$_{\pm 0.78}$	55.99$_{\pm 2.33}$	86.83$_{\pm 0.72}$	86.82$_{\pm 0.73}$
	roberta-large	86.68$_{\pm 0.51}$	86.01$_{\pm 0.47}$	98.20$_{\pm 0.30}$	95.13$_{\pm 0.66}$	96.46$_{\pm 0.17}$	80.62$_{\pm 2.22}$	70.23$_{\pm 1.25}$	57.73$_{\pm 3.93}$	89.31$_{\pm 0.64}$	89.29$_{\pm 0.65}$
	Meta-Llama-3-8B	89.41$_{\pm 0.31}$	88.77$_{\pm 0.32}$	98.15$_{\pm 0.37}$	94.76$_{\pm 1.56}$	96.23$_{\pm 0.62}$	81.17$_{\pm 5.24}$	76.20$_{\pm 1.82}$	67.14$_{\pm 2.98}$	**91.77**$_{\pm 0.34}$	**91.76**$_{\pm 0.34}$
	Qwen3-Embedding-8B	**90.05**$_{\pm 0.18}$	**89.38**$_{\pm 0.21}$	**98.39**$_{\pm 0.08}$	**95.35**$_{\pm 0.56}$	**96.75**$_{\pm 0.54}$	81.45$_{\pm 3.62}$	**78.08**$_{\pm 0.91}$	**69.82**$_{\pm 1.20}$	91.55$_{\pm 0.46}$	91.54$_{\pm 0.46}$
Best over baseline		**+2.07**	**+2.18**	**+0.10**	**+0.39**	**+0.73**	**-2.60**	**+1.37**	**+0.20**	**+0.85**	**+0.85**

Table 9: Evaluation results of *fine-tuned SLM*s and *LoRA-finetuned LLMs* on five benchmark datasets. Test set performance is reported using the learning rate that achieved the highest Micro-F1 on the validation set. Standard deviation across 5 runs is reported in subscript. The best result per column is in **bold**, and the second best is underlined.

Architecture	Backbone	20NG		R8		R52		Ohsumed		MR	
		Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1
Linear	bert-base-uncased	85.45$_{\pm 0.30}$	84.59$_{\pm 0.36}$	97.67$_{\pm 0.10}$	94.71$_{\pm 0.42}$	96.32$_{\pm 0.38}$	82.43$_{\pm 4.49}$	70.05$_{\pm 1.21}$	59.45$_{\pm 1.90}$	85.36$_{\pm 0.15}$	85.34$_{\pm 0.14}$
	bert-large-uncased	85.12$_{\pm 0.58}$	84.31$_{\pm 0.55}$	97.91$_{\pm 0.31}$	94.60$_{\pm 0.84}$	96.67$_{\pm 0.30}$	86.50$_{\pm 1.64}$	72.84$_{\pm 1.05}$	62.91$_{\pm 1.48}$	87.87$_{\pm 0.57}$	87.86$_{\pm 0.57}$
	roberta-base	84.51$_{\pm 1.26}$	83.69$_{\pm 1.21}$	97.90$_{\pm 0.37}$	94.34$_{\pm 0.67}$	96.44$_{\pm 0.34}$	84.44$_{\pm 3.98}$	72.28$_{\pm 1.83}$	62.82$_{\pm 2.54}$	88.30$_{\pm 0.35}$	88.29$_{\pm 0.36}$
	roberta-large	86.61$_{\pm 0.38}$	85.77$_{\pm 0.31}$	98.16$_{\pm 0.26}$	95.06$_{\pm 0.71}$	96.75$_{\pm 0.29}$	84.69$_{\pm 2.84}$	74.74$_{\pm 0.62}$	66.31$_{\pm 0.55}$	91.11$_{\pm 0.57}$	91.11$_{\pm 0.57}$
	Meta-Llama-3-8B	89.80$_{\pm 0.58}$	89.08$_{\pm 0.61}$	98.17$_{\pm 0.09}$	95.28$_{\pm 1.01}$	96.30$_{\pm 0.46}$	84.97$_{\pm 1.48}$	78.96$_{\pm 0.69}$	73.27$_{\pm 0.55}$	92.61$_{\pm 0.57}$	92.60$_{\pm 0.57}$
	Qwen3-Embedding-8B	90.00$_{\pm 0.28}$	89.30$_{\pm 0.26}$	98.53$_{\pm 0.20}$	**96.10**$_{\pm 0.43}$	96.99$_{\pm 0.30}$	85.99$_{\pm 0.96}$	80.80$_{\pm 0.47}$	75.36$_{\pm 0.69}$	92.34$_{\pm 0.19}$	92.34$_{\pm 0.19}$
MLP	bert-base-uncased	85.76$_{\pm 0.83}$	85.04$_{\pm 0.79}$	97.65$_{\pm 0.68}$	94.47$_{\pm 0.76}$	96.39$_{\pm 0.19}$	85.25$_{\pm 1.67}$	70.66$_{\pm 0.83}$	60.39$_{\pm 2.32}$	85.79$_{\pm 0.60}$	85.75$_{\pm 0.61}$
	bert-large-uncased	86.02$_{\pm 0.51}$	85.24$_{\pm 0.45}$	97.72$_{\pm 0.29}$	94.46$_{\pm 0.50}$	96.28$_{\pm 0.57}$	84.11$_{\pm 4.32}$	72.19$_{\pm 0.65}$	62.80$_{\pm 1.36}$	87.68$_{\pm 0.40}$	87.68$_{\pm 0.40}$
	roberta-base	85.35$_{\pm 0.30}$	84.53$_{\pm 0.34}$	97.96$_{\pm 0.28}$	94.39$_{\pm 0.81}$	96.56$_{\pm 0.36}$	86.25$_{\pm 1.53}$	72.65$_{\pm 0.38}$	62.67$_{\pm 0.91}$	88.91$_{\pm 0.77}$	88.90$_{\pm 0.77}$
	roberta-large	86.74$_{\pm 0.68}$	85.97$_{\pm 0.67}$	98.24$_{\pm 0.36}$	95.39$_{\pm 0.86}$	96.88$_{\pm 0.39}$	84.11$_{\pm 3.93}$	72.85$_{\pm 0.90}$	63.92$_{\pm 1.15}$	91.06$_{\pm 0.60}$	91.05$_{\pm 0.61}$
	Meta-Llama-3-8B	90.08$_{\pm 0.79}$	89.34$_{\pm 0.79}$	97.97$_{\pm 0.18}$	94.34$_{\pm 0.71}$	97.05$_{\pm 0.29}$	**87.54**$_{\pm 1.87}$	81.12$_{\pm 0.29}$	76.30$_{\pm 0.50}$	**92.62**$_{\pm 0.65}$	**92.62**$_{\pm 0.65}$
	Qwen3-Embedding-8B	**90.45**$_{\pm 0.28}$	**89.74**$_{\pm 0.32}$	**98.54**$_{\pm 0.16}$	95.85$_{\pm 0.52}$	**97.32**$_{\pm 0.17}$	86.67$_{\pm 0.68}$	**82.06**$_{\pm 0.39}$	**76.47**$_{\pm 0.49}$	92.06$_{\pm 0.63}$	92.06$_{\pm 0.63}$

Table 10: Comparison of Micro-F1 scores HILT-GCN (frozen embeddings with GCN header) to a fully-finetuned embedder (with linear/MLP header) and the best model of any architecture (from the survey by Galke et al. (2025)).

Model	20NG	R8	R52	Ohsumed	MR
HILT-GCN (our) - *frozen* Qwen 3 embedding with GNN	90.05$_{\pm 0.18}$	98.39$_{\pm 0.08}$	96.75$_{\pm 0.54}$	78.08$_{\pm 0.91}$	91.55$_{\pm 0.46}$
Fully fine-tuned Qwen 3 embedding baseline (our)	90.45$_{\pm 0.28}$	98.54$_{\pm 0.16}$	97.32$_{\pm 0.17}$	**82.06**$_{\pm 0.39}$	92.06$_{\pm 0.63}$
Best fine-tuned model from the literature (Galke et al., 2025)	**92.68**$_{\pm 0.30}$	**98.97**	**98.12**	77.41	**92.39**
Second-best fine-tuned model from the literature	90.32	98.43	97.17	75.94$_{\pm 0.33}$	90.17

E Opportunities for Future Research

Text classification is only one specific use case of embedding-based downstream tasks. Other tasks that also run directly on the embedding representation (e. g., retrieval, clustering, summarization, ...) could also benefit from this general setup. Modern transformers are mostly decoder-based, and thus often support instruction capabilities. For the sake of comparison and simplicity, we yet did not use these features. Implementing them within the pipeline might improve evaluation performance, but would require more tokens in the context. While the GNN classifier itself is relatively lightweight, the number of transformer layers and the feature size of the transformer can introduce quite a computational overhead. We included all layers to bypass the dilemma of selecting the best layer. Not all layers may contribute equally to the classification result. Future work may include adding layer selection (Tenney et al., 2019) and using hierarchical embeddings like Matroyhska (Kusupati et al., 2022), which `Qwen3-Embedding-8B` supports.