A Datasets

A.1 Dataset format

For each dataset, all unprocessed raw files are represented in .json format. After preprocessing, we store the graph-type data compatible with PyTorch Geometric (PyG) [9] in the .pt format using PyTorch. Specifically, we have retained the raw text on nodes, the labels on nodes, the raw text on edges, and the adjacency matrix. We uniformly store the text embeddings of node and edge text in .npy files and load them during data processing.

A.2 Datasets license

The datasets are subject to the MIT license. For precise license information, please refer to the corresponding GitHub repository.

B Experiment

B.1 Implementation Details

GNNs are mainly derived from the implementation in the PyG library [9]. For the node classification task, numerical node labels corresponding to the nodes within the graph are necessary. This involves converting the categorical node categories found in the original data into numerical node labels within the graph. For the link prediction, we randomly sample node pairs that do not exist in the graph as negative samples, along with some edges present as positive samples. For LLM-based predictor methods, we focus on node classification and link prediction tasks. For node classification, inspired by the recent LLM-based classification algorithm [26], we use GPT-4 and GPT-3.5-TURBO models to predict the classification of text nodes by providing the probability for each class. We randomly select 1,000 text nodes along with all classification labels for this task. For the link prediction task, we also apply the GPT-4 and GPT-3.5-TURBO models to determine whether two text edges are related, providing an answer with the corresponding probability. For this task, we randomly select 1,000 pairs of positive text edge indices from the graph and an equal number of negative edges.

B.2 Effectiveness Analysis for Link Prediction

In this subsection, we further analyze the link prediction from the various models applied in the study. Table 6 and 7 represent the effect of link prediction on different datasets from various distinct. We can further draw several observations from Table 6 and 7. First, For PLM-based and GNN-based methods, the state-of-the-art methods for Goodreads-Comics and Goodreads-History datasets are GeneralConv and GINE, respectively. Under the condition of using the same embeddings, they outperform the worst method by approximately 6% and 7% in terms of AUC and F1 across these two datasets. For the Reddit dataset, the state-of-the-art method is GeneralConv. It outperforms the worst method by approximately 3% and 5% in terms of AUC and F1, respectively. Second, for the LLM as a predictor method, we find that they do not perform well in predicting links. The best method among them has an AUC and F1 gap of approximately 10% - 30% compared to the best PLM-based and GNN-based methods for all datasets. Third, Using edge text provides at least approximately a 3% improvement in AUC and at least approximately an 8% improvement in F1 compared to not using edge text for all datasets.

B.3 Effectiveness Analysis for Node Classification

In this subsection, we further analyze the node classification results from various models. Table 8 and 9 display the impact on different datasets. We can derive some insights. First, for PLM-based and GNN-based methods, the state-of-the-art models for Goodreads-Comics and Goodreads-History are GeneralConv and GINE, respectively, outperforming the worst method by approximately 8% and 15% in AUC-micro and F1-micro for Goodreads-Comics, and by 6% and 9% for Goodreads-History. GraphTransformer outperforms the worst method by approximately 2% and 1% in ACC and F1 for Citation. Second, LLM as Predictor methods perform poorly in node classification, with the best method showing an AUC-micro gap of about 20% compared to the best PLM-based and GNN-based methods. Their low F1-micro score could be due to the large number of predicted categories. Third,

Table 6: Link prediction AUC and F1 among PLM-based, GNN-based methods. The best method for each PLM embedding on each dataset is shown in bold.

Methods	Goodreads-Comics								Goodreads-History					
	GPT-3.5-TURBO		BERT-Large		BERT		None		BERT-Large		BERT		None	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
MLP	0.8902	0.8136	0.8900	0.8130	0.8900	0.8128	0.8928	0.8167	0.8922	0.8897	0.8923	0.8897	0.8913	0.8149
GraphSAGE	0.9406	0.8689	0.9511	0.8854	0.9537	0.8860	0.9403	0.8732	0.9587	0.8702	0.9591	0.8698	0.9053	0.8320
GeneralConv	0.9478	0.8843	**0.9535**	**0.8930**	**0.9544**	**0.8942**	**0.9458**	**0.8825**	0.9624	**0.8900**	0.9629	0.8897	0.9117	0.8426
GINE	**0.9489**	**0.8870**	0.9480	0.8857	0.9471	0.8833	0.9446	0.8819	**0.9631**	0.8669	**0.9634**	**0.8937**	**0.9132**	**0.8448**
EdgeConv	0.9448	0.8819	0.9495	0.8867	0.9477	0.8853	0.9444	0.8810	0.9457	0.8695	0.9456	0.8650	0.9036	0.8345
GraphTransformer	0.9380	0.8687	0.9433	0.8747	0.9466	0.8781	0.9362	0.8661	0.9589	0.8698	0.9590	0.8690	0.8985	0.8256

Methods	Reddit							
	GPT-3.5-TURBO		BERT-Large		BERT		None	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
MLP	0.9909	0.9651	0.9866	0.9576	0.8900	0.8128	0.8928	0.8167
GraphSAGE	0.9908	0.9810	0.9897	0.9800	0.9537	0.8860	0.9403	0.8732
GeneralConv	**0.9964**	0.9809	0.9956	**0.9815**	**0.9544**	**0.8942**	**0.9458**	**0.8825**
GINE	0.9962	0.9809	**0.9958**	0.9801	0.9471	0.8833	0.9446	0.8819
EdgeConv	0.9926	**0.9818**	0.9926	0.9803	0.9477	0.8853	0.9444	0.8810
GraphTransformer	0.9944	0.9810	0.9940	0.9803	0.9466	0.8781	0.9362	0.8661

Table 7: Link prediction results for LLM as Predictor methods. The best method on each dataset is shown in bold.

Methods	Goodreads-Comics		Goodreads-History		Reddit	
	AUC	F1	AUC	F1	AUC	F1
GPT-3.5-TURBO	0.4565	**0.3588**	0.6031	0.5234	0.4980	0.3440
GPT-4	**0.5446**	0.2461	**0.8661**	**0.8685**	**0.6632**	**0.6478**

Table 8: Node Classification ACC, Micro-AUC, Micro-F1 and F1 among PLM-based, GNN-based methods. AUC* and F1* represent Micro-AUC and Micro-F1 respectively. The best method for each PLM embedding on each dataset is shown in bold.

Methods	Goodreads-Comics								Goodreads-History					
	GPT-3.5-TURBO		BERT-Large		BERT		None		BERT-Large		BERT		None	
	AUC*	F1*	AUC*	F1*	AUC*	F1*	AUC*	F1*	AUC*	F1*	AUC*	F1*	AUC*	F1*
MLP	0.8361	0.5117	0.8360	0.5211	0.8370	0.5214	0.8373	0.5214	0.7831	0.8099	0.7825	0.8097	0.7824	0.8096
GraphSAGE	0.9068	0.7379	0.8965	0.7118	0.8965	0.7088	0.8689	0.6401	0.8543	0.8975	0.8538	0.8970	0.8044	0.8088
GeneralConv	**0.9107**	**0.7455**	**0.8982**	0.7134	**0.8991**	0.7116	**0.8739**	0.6541	0.8543	0.8986	0.8538	0.8981	0.8119	0.8126
GINE	0.9006	0.7187	0.8943	0.7084	0.8932	0.7140	0.8627	0.6457	0.8541	**0.9015**	0.8549	**0.9022**	**0.8133**	**0.8226**
EdgeConv	0.9015	0.7127	0.8923	0.7066	0.8931	0.7089	0.8648	0.6260	0.8520	0.8974	0.8515	0.8960	0.8059	0.8116
GraphTransformer	0.9027	0.7285	0.8940	**0.7175**	0.8966	**0.7151**	0.8704	**0.6554**	**0.8555**	0.9009	**0.8647**	0.8995	0.8101	0.8089

Methods	Reddit							
	GPT-3.5-TURBO		BERT-Large		BERT		None	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
MLP	0.9839	0.9817	0.9793	0.9774	0.9803	0.9784	0.9795	0.9779
GraphSAGE	0.9974	0.9962	**0.9975**	0.9964	0.9973	0.9963	**0.9974**	**0.9965**
GeneralConv	**0.9975**	**0.9966**	0.9974	0.9963	0.9973	0.9964	0.9973	0.9964
GINE	0.9973	0.9962	0.9973	0.9963	**0.9974**	0.9965	**0.9974**	0.9962
EdgeConv	0.9973	0.9960	0.9973	0.9960	0.9973	0.9960	0.9973	0.9959
GraphTransformer	0.9973	0.9963	0.9974	**0.9965**	**0.9974**	**0.9966**	0.9973	0.9964

Table 9: Node Classification ACC, Micro-AUC, Micro-F1 and F1 for LLM as Predictor methods. AUC* and F1* represent Micro-AUC and Micro-F1 respectively. The best method on each dataset is shown in bold.

Methods	Goodreads-Comics		Goodreads-History		Reddit	
	AUC*	F1*	AUC*	F1*	ACC	F1
GPT-3.5-TURBO	0.4900	0.0400	0.6827	0.4147	0.8625	0.9262
GPT-4	**0.5600**	**0.0600**	**0.8202**	**0.7394**	**0.9767**	**0.9882**

incorporating edge text results in at least a 3% improvement in AUC-micro and a 6% improvement in F1-micro across almost all datasets, compared to not using edge text.

C Discussion

Notably, employing APIs like GPT4 for extensive graph tasks may result in considerable expenses under current billing models. Additionally, deploying open-source large models such as LLaMa for tasks like parameter updates or inference in local environments demands substantial computational resources and storage capacity. Consequently, enhancing the efficiency of LLMs for graph-related tasks remains a critical concern. Moreover, the constraints imposed by context windows in LLMs also impact their effectiveness in encoding node and edge text within TEGs.

D Limitation

Comprehensive evaluation of tasks often demands significant computational resources, which can be a burden for researchers and smaller organizations.