Large Language Model Guided Graph Clustering

Puja Trivedi
University of Michigan
pujat@umich.edu

Nurendra Choudhary
Amazon Inc.
nurendc@amazon.com

Edward W. Huang
Amazon Inc.
ewhuang@amazon.com

Vassilis N. Ioannidis
Amazon Inc.
ivasilei@amazon.com

Karthik Subbian
Amazon Inc.
ksubbian@amazon.com

Danai Koutra
University of Michigan
dkoutra@umich.edu

Abstract

Graph clustering on text-attributed graphs (TAGS), i.e., graphs that include natural language text as additional node information, is typically performed using graph neural networks (GNNs), which forego the text in lieu of embeddings. While GNN methods ensure scalability and effectively leverage graph topology, text attributes contain rich information that can be leveraged using large language models (LLMs). However, many real-world applications have limited hardware resources or LLM API call budgets that prevent their naive use. To reconcile these constraints when performing clustering on TAGs, we propose an active learning framework that performs **g**raph **c**lustering using **L**LM **r**efinment (GCLR) by selectively prompting an imperfect LLM oracle for feedback and, subsequently, finetuning the GNN-based clustering solution to incorporate the feedback. GCLR uses different prompting strategies to improve the LLM's reliability as an oracle and uses noise-controlling fine-tuning to handle this imperfect, but useful feedback. Extensive experiments demonstrate that GCLR can significantly improve clustering performance over state-of-the-art GNN methods.

1 Introduction

Graph clustering seeks to perform an unsupervised assignment of nodes to different clusters such that the resulting assignments capture salient topology and uncover useful concepts [1–3]. Most modern, performative clustering methods utilize graph neural network (GNN) encoders due to their expressivity [4], scalability, and ability to effectively handle vector-valued node attributes [5, 6]. Recently, however, there has been growing interest in *text-attributed graphs* (TAGs) [7, 8], where natural language text is available as an additional node attribute. GNNs are not able to directly handle text and instead utilize semantic embeddings, potentially limiting performance. To this end, a variety of (pre/co/joint) training-based [9–13] and graph specific prompting-based strategies [14–18] have been recently proposed for using large language models (LLMs) [19, 20] in conjunction with GNNs on *supervised* tasks, e.g., link prediction, node classification, and graph classification, to directly handle this text. While clustering on TAGs could also benefit from joint LLM+GNN methods, it not only remains unclear how to adapt existing supervised approaches for unsupervised graph clustering, but also is prohibitively expensive in many real-world applications due to relatively significant hardware requirements, incurred through training or hosting LLMs, or API expenditure, incurred by prompting over large sets of nodes.

Our Work. To this end, we propose GCLR (**G**raph **C**lustering with **L**LM **R**efinement), a flexible active learning framework specifically designed for clustering on TAGS. It uses carefully designed prompting strategies to elicit more reliable and useful feedback for clustering from the LLM and uses simple strategies when fine-tuning to improve tolerance to noisy labels, overall outperforming GNN-based clustering methods. (See Fig. 1 for an overview.)

P. Trivedi et al., Large Language Model Guided Graph Clustering (Extended Abstract). Presented at the Third Learning on Graphs Conference (LoG 2024), Virtual Event, November 26–29, 2024.

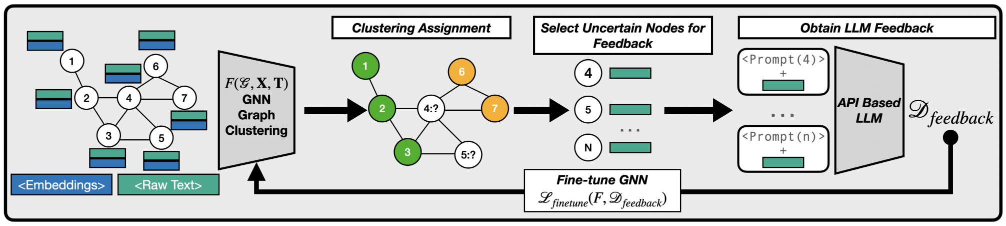


Figure 1: *Overview of GCLR, active learning framework for refining graph clustering solutions.*

Related Work. Please see App. B for a detailed discussion of related work. In brief, we note that recent approaches that seek to combine graphs/ GNNs and natural-language/LLMs can be categorized as being “predictors" (the LLM provides predictions), “enhancers" (sentence transformers or other LLMs are used to provide input node features), or “aligners" (GNNs and LLMs jointly trained to perform the task) [21]. Various mechanisms, including prompting [22], fine-tuning [23], variational expectation maximization [24], joint optimization [25], and distillation [26], have been proposed to fulfill these roles. Instead, GCLR uses the LLM as a refiner and enhancer, as the LLM is only prompted to provide feedback for updating the underlying GNN-based graph clustering solution and sentence transformers are used to provide input node embeddings.

2 GCLR: Graph Clustering with LLM Refinement

Let $\mathcal{G} = (\mathbf{A} \in \mathbb{R}^{N\times N}, \mathbf{X} \in \mathbb{R}^{N\times d}, \mathcal{T}, [\mathcal{Y}])$ represent an N node graph with corresponding adjacency matrix, node attributes, node-level text information, and optional ground-truth cluster assignment, where K is the desired number of clusters, and d the dimension of the hidden representation. Deep attributed graph clustering seeks to learn a function, $\mathbf{F} : (\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{Z}^{N\times d}$, that outputs node representations, and (an optionally parameterized) clustering algorithm, $\mathbf{C} : (\mathbf{Z}, K) \rightarrow [0, K]^N$, which provides assignments, $\mathbf{K}^{N\times K}$. We assume that clustering assignments are imperfect, namely that there are mis-assigned samples, and the LLM's world-knowledge can improve $\mathbf{K}$. We further assume that the LLM is only accessible through a limited budget, $\mathbf{B}$, of API calls. This problem formulation is naturally amenable to an active learning (AL) set-up, which consists of three key components: a *query function*, $\mathbf{Q}$, which determines which samples from the unlabeled data pool should be selected for obtaining feedback, *an oracle*, which provides feedback to create a labeled dataset, $\mathcal{D}_{\text{feedback}}$, and a *training protocol*, which defines a loss, $\mathcal{L}_{\text{feedback}}$, and update procedure for how the model will incorporate said feedback. Belwo, we introduce GCLR as an instantiation.

2.1 Eliciting Feedback from LLM for Graph Clustering

While feedback in AL typically corresponds to an oracle selecting a label from a predefined set of classes, it is less clear what form feedback should take when performing clustering and how to prompt the LLM to obtain it. To this end, we discuss the pros/cons of 3 different strategies.

Triplet-Based Prompting. ClusterLLM [27], a recently proposed SOTA LLM guided *text* clustering method, prompts by first selecting uncertain samples (e.g., queries), $\mathcal{Q}_i$, and two random samples from each query's two nearest clusters, and then ask the LLM to predict which of the two samples is “more similar" to $\mathcal{Q}_i$; the more similar sample is considered a “positive" sample and the other is a “negative" sample. Here, $\mathcal{D}_{\text{feedback}}$ corresponds to the set of triplets (query, positive, negative) determined by the LLM and $\mathcal{L}_{\text{feedback}}$ is InfoNCE. While such an approach can conceptually be applied to graph clustering, there are some limitations. Namely, insofar as clustering requires learning a similarity function that can be used to partition samples into meaningful groups, it is important that the oracle is aware of this function so the resulting feedback is aligned to the existing partitioning. In text clustering, since both the encoder and the oracle are text based models, they share a prior for this similarity function. In contrast, when performing graph clustering, the GNN incorporates topological information unavailable to the LLM and may utilize a different function.

In-Context Similarity Learning. Given their in-context learning capabilities [28, 29], we consider a prompt that allows the LLM to directly infer the similarity function by providing several examples of the node's raw text and their corresponding cluster IDs, and the text of the unlabeled query. Here, the LLM can be seen as performing a prediction task amongst pseudo-labels (cluster ids), where $\mathcal{D}_{\text{feedback}} = \{([0, \ldots K]|i \in \mathcal{Q}\}$. Note the choice of $\mathcal{L}_{\text{feedback}}$ is flexible and discuss it later. Notably, by ensuring that the prompt contains samples from all clusters, the LLM can (i) better infer what concepts underlie clusters and (ii) predict an assignment for a query that does not belong to the top-2 clusters. This allows us to circumvent the previous issue where the upper-bound on refined performance was restricted by the number of samples where the preferred assignment was contained in the top-2 clusters. Note, that directly inferring the similarity function from in-context examples becomes more difficult, even potentially unfeasible, as the number of clusters grows.

Concept-based Prompting. Drawing inspiration from topic modeling [30, 31], we design an additional "concept-based" prompting strategy where we first prompt the LLM to infer the concepts that were used to group samples and then create a prediction task where the LLM is prompted to select amongst the generated concepts. Specifically, we provide the LLM with samples from each cluster to generate a title and short description explaining the grouping, which are then used as options for the LLM to match a query to the most similar cluster. Notably, by providing the titles/descriptions of all clusters, we can avoid the upper-bound encountered by triplets, as well as better scale to more clusters and use on-average, shorter prompts than in-context prompting.

Experimental Setup. We verify the effectiveness of the proposed feedback elicitation strategies on several public graph datasets, where the provided node labels serve as ground-truth cluster labels. `mixtral-8x-7b` is used as the oracle, and four different graph clustering backbones are used to obtain the initial clustering solutions. We sort the samples according to the entropy of the distance to the two nearest clusters (a proxy for sample difficulty) and prompt the LLM for each sample as per the discussed strategies. See App. F for more details.

Table 1: Reliability of LLM as an Annotator. The accuracy of the GNN-based clustering solution and three prompting strategies are reported at the 10\50\100-th most difficult percentile of the dataset. The best performance overall is **bolded**, while any prompting-based method is colored if it exceeds the accuracy of the GNN, and the 2nd best prompting-based method is underlined.

Dataset	Method	GNN	Concepts	Incontext	Triplets
		Graph Only	LLM Only		
citeseer	diffpool	32.1\36.2**49.7**	**36.2** **41.1** \49.1	34.6 \36.2 \46.7	29.2\34.1\44.0
	dinknet	40.6**54.7****70.3**	30.8\32.9\47	**48.7**\48.3\59.6	43.1\50.6\62.1
	dmon	36.5\38.2**44.1**	**40.9****39.9**\43.9	36.2\37.7\42.9	36.8\38\41.7
	mincut	35.8**52.2****66.5**	38.4\46.1\58.5	**42.1**\50.5\60.5	34.3\46.5\57.1
cora	diffpool	32.6**40****54.7**	**35.6**\36.0\37.7	34.4\36.6\50.2	33.7\36.9\48.8
	dinknet	**37.4****50.7****65.8**	32.2\36.8\39	24.8\36.0\52.7	35.2\47\58.2
	dmon	42.6**52.4****60.9**	36.3\41.4\40.7	**46.3**\51.3\56.9	40\47.9\54
	mincut	40**53.6****68.4**	42.2\46.5\55.7	**43.7**\50.5\63.3	37.8\49.8\60.9
wikics	diffpool	25.5\32.2\48.3	**36.0****40.4****52.7**	33.9\37.1\47.9	25.9\30.8\44.2
	dinknet	37.7\51.2**66.5**	**51.2****56.5**\64.8	35.8\36.9\51.1	35.0\44.5\54.8
	dmon	28.1\31.2\36.9	**55.2****55.2****57.2**	39.9\41.3\41.3	28.7\31.2\35.8
	mincut	36.5\24.4\26.9	31.9**29.6**\29.8	**37.5**\27.9**31.0**	32.4\24.1\25.2

Results. We observe, in Table 1, that across datasets and clustering methods, that the "concepts" strategy is the best or second best performing prompting strategy most often. While In-Context prompting achieves comparable performance on some datasets, we note that it is significantly more expensive. Indeed, every In-Context prompt contains multiple exemplars per cluster, while "concepts" only processes these exemplars once to obtain the generated titles and descriptions, which are then directly used in the prompt. "Triplets" is the cheapest strategy in terms of token length, but lags behind on performance, failing to achieve the best performance on any dataset. Lastly, we note that the GNN outperforms the LLM on full dataset (100th percentile) accuracy on 9/12 settings, indicating that, in addition to being prohibitively expensive, prompting the LLM for every node would not be as effective as the initial GNN solution.

2.2 Refining GNN-Based Clustering with Feedback

Given the LLM's feedback, we must incorporate it into the GNN to scalably improve the overall clustering solution. We note that while reconstructive [32, 33] and adversarial frameworks [34] were initially popular for graph clustering, we focus on more recent contrastive [35–38] and pooling-based methods [39–41] as they are more scalable and performative. Further note that both "in-context" and "concept" prompting induce a dataset that consists of queried nodes and predicted cluster ids. Thus, we can consider refinement as a supervised task with LLM-provided pseudo-labels. When working with pooling-based methods, $\mathbf{F}$ directly predicts the cluster assignment; with contrastive methods, a classifier can be initialized using cluster centers. Then, $\mathcal{L}_{feedback}(\mathcal{D}_{feedback}, \mathbf{F})$ can be defined using the cross-entropy loss. While other losses, such as triplet [42], InfoNCE [43], SupCon [44], are certainly possible, we empirically find that cross-entropy is effective.

Strategies for Handling Noisy Labels. Given that the prompting strategies induce a classification task, we use the model's predicted confidence in order to eliminate potentially noisy labels. Namely, we compute the LLM's confidence in its predictions by obtaining log-probability of the top-2 tokens corresponding to cluster predictions. Alternative prompting strategies and specialized losses have been proposed for better calibration [45–47] but we do not consider them due to their additional expense. Empirically, we find that token-level log probability is sufficient.

Table 2: LLM Labels Provide Complementary Information For Active Learning. Here, we compare the performance of different feedback mechanisms and finetuning losses. We observe that (i) while both LLM (9/12 Acc.) and GNN (10/12 Acc) feedback generally improves performance over the starting solution, that LLM feedback with the cross entropy loss achieves the best accuracy overall (8/12), though performance on intrinsic metrics is more mixed; (ii) on Cora, where GNN feedback was more reliable than LLM feedback, we see that using the GNN pseudo labels is more effective; (iii) on WikiCS, where LLM feedback is much more reliable, we see dominant performance by LLM feedback with cross entropy loss; and (iv) we see that the cross entropy loss (9/12 Acc., 7/12 Modularity, 7/12 NMI) is more effective than the triplet for finetuning.

		(starting performance) \ GNN Feedback + Cross Ent. Loss \ LLM Feedback + Triplet Loss \ LLM Feedback + Cross Ent. Loss					
Dataset	**Method**	**Acc.** (↑)	**NMI** (↑)	**F1** (↑)	**ARI** (↑)	**COND** (↓)	**MOD** (↑)
citeseer	diffpool	(47.09) \54.69 \48.05 **58.96**	(25.59) \25.94 \21.50 **26.84**	(23.08) **23.57** \14.65 \19.70	(43.09) **43.33** \33.22 \41.41	(0.23) **0.23** \0.25 \0.24	(0.56) **0.56** \0.45 \0.50
	dinknet	(66.46) \66.43 **67.36** **67.40**	(43.08) **43.30** \19.16 \36.97	(42.43) **41.30** \16.37 \27.16	(60.39) **60.58** \42.49 \47.91	(0.07) **0.07** \0.29 \0.09	(0.70) **0.70** \0.51 \0.62
	dmon	(47.89) \49.85 \48.75 **49.87**	(28.49) **28.77** \27.11 \27.12	(24.29) **24.61** \18.86 \14.46	(43.65) **43.71** \34.14 \29.87	(0.19) \0.19 \0.25 **0.15**	(0.60) **0.60** \0.45 \0.47
	mincut	(64.18) \66.70 **69.82** **67.51**	(44.41) **46.21** \40.48 \39.60	(41.95) **43.25** \38.54 \35.81	(61.72) **62.11** \59.54 \59.81	(0.08) **0.09** \0.13 \0.17	(0.73) **0.73** \0.67 \0.64
cora	diffpool	(59.97) **63.6** \43.38 \51.35	(43.46) **42.70** \20.97 \22.21	(36.58) **35.65** \7.83 \6.49	(56.76) **55.64** \29.3 \29.05	(0.24) **0.25** \0.38 \0.32	(0.60) **0.60** \0.33 \0.34
	dinknet	(68.26) \66.84 **67.32** \65.16	(51.98) **50.87** \25.01 \23.42	(44.21) **40.50** \15.16 \9.25	(62.09) **59.20** \41.86 \27.40	(0.12) \0.11 \0.30 **0.08**	(0.70) **0.67** \0.49 \0.29
	dmon	(57.56) **60.27** \59.06 \56.70	(41.60) **42.24** \30.18 \30.06	(33.76) **34.64** \20.66 \13.67	(50.94) **51.40** \39.44 \29.40	(0.27) \0.26 \0.38 **0.12**	(0.56) **0.58** \0.42 \0.33
	mincut	(64.17) **66.63** \59.91 \61.62	(48.92) **48.92** \39.74 \41.61	(40.35) **40.35** \29.43 \30.54	(58.33) **58.33** \47.28 \54.01	(0.14) **0.14** \0.21 \0.28	(0.70) **0.70** \0.56 \0.54
wikics	diffpool	(43.15) \49.69 **55.44** **58.03**	(26.27) \26.36 **37.20** **35.03**	(18.87) \19.50 **31.10** **26.28**	(39.88) \39.70 **41.12** **46.48**	(0.34) \0.35 **0.30** \0.34	(0.48) **0.47** \0.36 \0.44
	dinknet	(66.80) \73.65 \67.48 **74.00**	(49.00) **51.84** \47.49 \51.25	(47.80) **53.04** \46.18 \51.57	(56.23) **63.06** \56.97 **63.06**	(0.23) **0.21** \0.28 \0.23	(0.55) \0.55 \0.52 **0.54**
	dmon	(38.60) \39.68 **43.28** **51.87**	(27.47) \27.49 **29.33** **32.51**	(20.55) \20.65 **27.48** **31.04**	(34.02) \34.18 **34.48** **36.49**	(0.48) \0.47 **0.42** **0.26**	(0.33) **0.33** **0.33** \0.31
	mincut	(24.70) \32.84 **38.52** **46.36**	(6.14) \8.32 **17.99** **16.52**	(-0.37) \-0.32 **18.02** **4.8**	(7.91) \8.45 **24.71** **24.36**	(0.04) **0.04** \0.45 \0.47	(0.03) \0.05 **0.30** **0.27**

To further stabilize and improve training, we augment $\mathcal{D}_{feedback}$ with samples well-clustered by the GNN, where probits of the predicted clusters are used to identify confident assignments. The loss is computed separately for the LLM-labeled and GNN-labeled samples, and aggregated as $\alpha\mathcal{L}_{finetune,LLM} + \beta\mathcal{L}_{finetune,GNN}$, where α and β are constrained to be a convex combination. By varying α and β, we can express different levels of certainty in the feedback. In practice, we find setting α and β to 0.5 leads to strong performance. Since the optimal weighting is not known apriori, creating a simple deep ensemble [48] by varying α, β to train multiple independent models can further improve performance. Though this incurs additional training expenditure, it is not substantial with respect to training the initial model.

3 Experiments

Our experimental set-up is as follows. *Baselines.* We consider the following graph clustering baselines: MinCutPool [40], DMoN [39], DiffPool [41], and DinkNet [35]. *Metrics.* As we use public datasets with available ground-truth clustering, we report accuracy, Normalized Mutual Information, F1, and Adjusted Rand-Index between the predicted and labeled clusters. We intrinsically assess the clustering quality using conductance and modularity (see App. G). We use embeddings obtained from SBERT as node features for all experiments. *Datasets.* We provide the dataset statistics in Table H. *Training.* Both the initial GNN and finetuned models are trained with early-stopping and the learning rate is tuned amongst 1e-4 and 1e-3. *GCLR*. We use `mixtral-8x-7b` as the oracle LLM and seek feedback on 10% of the nodes in the dataset. (See App. E for additional results with ChatGPT.) $\mathbf{Q}$ is defined according to prediction entropy [49]. α and β are both set to 0.5, unless otherwise noted. Results are averaged over 10 seeds.

Observation 1. We begin by confirming that the LLM provides valuable information through its feedback by demonstrating, in Table 2, that subsequent finetuning not only improves performance over the starting clustering solution but also over finetuning on GNN pseudo labels, when reliable. Additionally, we find that using the cross entropy loss is more effective than the triplet loss when finetuning using the LLM feedback. This is in contrast to ClusterLLM, which focused on triplets.

Observation 2. Next, we seek to understand how filtering samples according to confidence can improve GCLR's performance. We do note that both the GNN and LLM feedback are not guaranteed to be calibrated, but nonetheless empirically find their confidences useful. In particular, in Table **??**, we set $\alpha = 0.5$ and $\beta = 0.5$, and consider 2 different filterings: one where the GNN's confidence interval is high and the other where the LLM's confidence interval is high. We find that updating the model using only high confidence LLM feedback (80th percentile) and GNN feedback at lower percentile improves the accuracy on Cora and Citeseer.

Observation 3. In settings where the LLM's feedback is less reliable than the GNN's, it is possible to harm the initial clustering solution. For example, in Table 1, on Cora, the LLM's feedback is less reliable than the GNN's, and in Table 2, we see finetuning on GNN feedback leads to better performance. However, we note that even if the LLM's feedback is unreliable it may still contain valuable information. To this end, we create a simple deep ensemble that captures different levels of certainity in either source's feedback by varying α and β when aggregating the loss. In particular, we train 5 different models, where we sample $\alpha \in [0, 0.1, \dots 0.5]$ and $\beta \in [0.5, 0.6 \dots 1]$ at evenly spaced intervals. In Table 4, we show that using this ensemble can improve performance over a single model where $\alpha = \beta = 0.5$.

References

[1] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006. 1, 21

[2] M. E. J. Newman and Gesine Reinert. Estimating the number of communities in a network. *CoRR*, abs/1605.02753, 2016.

[3] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pages 745–754. IEEE Computer Society, 2012. doi: 10.1109/ICDM.2012.138. URL https://doi.org/10.1109/ICDM.2012.138. 1, 21

[4] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019. 1

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017. 1

[6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2018. 1

[7] 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 *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[8] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. 1

[9] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022. 1

[10] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

[11] Vassilis N. Ioannidis, Xiang Song, Da Zheng, Houyu Zhang, Jun Ma, Yi Xu, Belinda Zeng, Trishul Chilimbi, and George Karypis. Efficient and effective training of language and graph neural network models. *CoRR*, abs/2206.10781, 2022.

[12] Costas Mavromatis, Vassilis N. Ioannidis, Shen Wang, Da Zheng, Soji Adeshina, Jun Ma, Han Zhao, Christos Faloutsos, and George Karypis. Train your own GNN teacher: Graph-aware distillation on textual graphs. In *Proc. European. Conf. on Machine Learning and Knowledge Discovery in Databases (ECML KDD)*, 2023.

[13] Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl Yang, Yi Xu, et al. Graph-aware language model pre-training on a large graph corpus can help multiple graph applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5270–5281, 2023. 1

[14] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. Explanations as features: Llm-based features for text-attributed graphs. *CoRR*, abs/2305.19523, 2023. 1, 10

[15] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M. Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space. *CoRR*, abs/2310.01089, 2023.

[16] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *CoRR*, abs/2310.04560, 2023. doi: 10.48550/ARXIV.2310.04560. URL https://doi.org/10.48550/arXiv.2310.04560.

[17] Jiayan Guo, Lun Du, and Hengyu Liu. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking. *CoRR*, abs/2305.15066, 2023.

[18] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. *CoRR*, abs/2310.13023, 2023. 1

[19] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. 1

[20] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073, 2022. 1

[21] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *CoRR*, abs/2312.02783, 2023. 2

[22] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. StructGPT: A general framework for large language model to reason over structured data. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.574. URL https://aclanthology.org/2023.emnlp-main.574. 2

[23] Hao Liu, Jiarui Fend, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4IT2pgc9v6. 2

[24] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=q0nmYciuuZN. 2

[25] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*, 2023. 2

[26] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv preprint arXiv:2110.07178*, 2021. 2

[27] Yuwei Zhang, Zihan Wang, and Jingbo Shang. Clusterllm: Large language models as a guide for text clustering. In *Proc. Int. Conf. on Empirical Methods in Natural Language Processing, (EMNLP)*, 2023. 2, 11

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. 2, 12

[29] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 12

[30] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering. *CoRR*, abs/2307.00524, 2023. doi: 10.48550/ARXIV.2307.00524. URL https://doi.org/10.48550/arXiv.2307.00524. 3, 12

[31] Chau Minh Pham, Alexander Miserlis Hoyle, Simeng Sun, and Mohit Iyyer. Topicgpt: A prompt-based topic modeling framework. *CoRR*, abs/2311.01449, 2023. doi: 10.48550/ARXIV.2311.01449. URL https://doi.org/10.48550/arXiv.2311.01449. 3, 12

[32] Hongyuan Zhang, Pei Li, Rui Zhang, and Xuelong Li. Embedding graph auto-encoder for graph clustering. *IEEE Trans. Neural Networks Learn. Syst.*, 2023. 3

[33] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proc. Int. Web Conf. (WebConf)*, 2020. 3

[34] Lei Gong, Sihang Zhou, Wenxuan Tu, and Xinwang Liu. Attributed graph clustering with dual redundancy reduction. In *Proc. of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI*, 2022. 3

[35] Yue Liu, Ke Liang, Jun Xia, Sihang Zhou, Xihong Yang, Xinwang Liu, and Stan Z. Li. Dink-net: Neural clustering on large graphs. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2023. 3, 4

[36] Jun Xia, Lirong Wu, Ge Wang, Jintao Chen, and Stan Z. Li. Progcl: Rethinking hard negative mining in graph contrastive learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2022.

[37] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Velickovic, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

[38] Fnu Devvrit, Aditya Sinha, Inderjit S. Dhillon, and Prateek Jain. S3GC: scalable self-supervised graph clustering. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[39] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *J. Mach. Learn. Res.*, 2023. 3, 4

[40] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2020. 4

[41] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2018. 3, 4

[42] Geoffrey Hinton and Sam T Roweis. Dimensionality reduction by learning an invariant mapping. *Neural computation*, 15(6):1373–1396, 2003. 3

[43] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A. Alemi, and George Tucker. On Variational Bounds of Mutual Information. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2019. 3

[44] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[45] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics, 2023. 3

[46] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, 2023.

[47] Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A. Heller, and Subhrajit Roy. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. 2023. 3

[48] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017. 4

[49] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pages 112–119. IEEE, 2014. doi: 10.1109/IJCNN.2014.6889457. URL `https://doi.org/10.1109/IJCNN.2014.6889457`. 4

[50] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. Simteg: A frustratingly simple approach improves textual graph learning. *CoRR*, abs/2308.02565, 2023. 10

[51] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. How powerful are graph neural networks? In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024. 10

[52] Jing Zhu, Xiang Song, Vassilis N. Ioannidis, Danai Koutra, and Christos Faloutsos. Touchup-g: Improving feature representation through graph-centric finetuning. *CoRR*, abs/2309.13885, 2023. doi: 10.48550/ARXIV.2309.13885. 10

[53] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1997. 21

[54] Michael D Plummer and László Lovász. *Matching theory*. Elsevier, 1986. 21

[55] Ka Yee Yeung and Walter L Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001. 21

[56] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *CoRR*, abs/1707.03815, 2017. 22

[57] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2016. 22

[58] Péter Mernyei and Catalina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *CoRR*, abs/2007.02901, 2020. 22

A Appendix / supplemental material

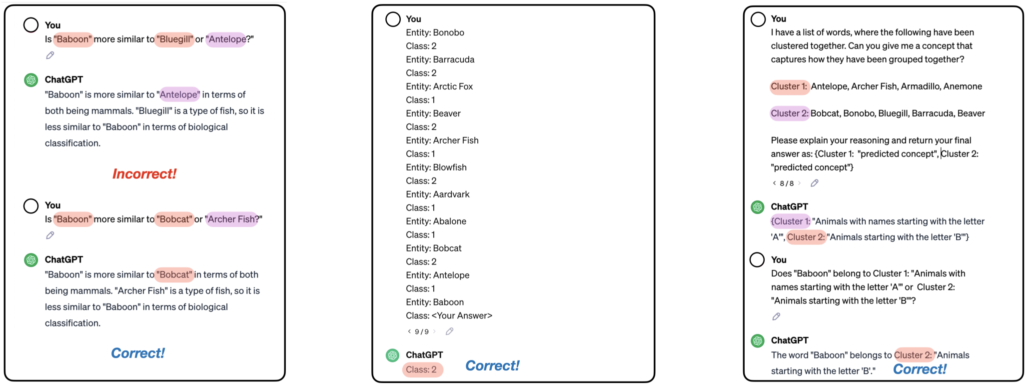
- Expanded Related Work (Sec. B)
- Expanded Prompting Discussion (Sec. C)
- Additional + Ablation Results (Sec. D)
- Additional Results with ChatGPT (Sec. E)
- Prompt Examples (Sec. F)
- Details about Metrics (Sec. G)
- Reproducibility (Sec. H)

B Expanded Related Work

Here we discuss some proposed LLM+GNN models, and their applicability to graph clustering. First, we note that existing work has primarily focused on supervised tasks (mostly node classification and to a lesser extent link prediction), and does not assume budget constraints, prompting over the entire graph or finetuning PLMs/LLMs. For example, TAPE [14], a recent prompting focused LLM-as-Encoder method, prompts the LLM at every node for a class prediction and explanation, before fine tuning a pretrained language model to obtain embeddings. Prompting for every node can be extremely expensive in the case of large graphs, and, in our setting, we do not have pre-determined class labels to simplify how feedback is obtained from the LLM, making it challenging to finetune the PLM. Similarly, SimTeG [50], a fine-tuning based LLM-as-Encoder method, uses LoRA to train the LLM directly on the downstream node classification task, before extracting embeddings for training a GNN. Such an approach requires both supervision (which is unavailable in graph clustering) and fine-tuning of language models, which can incur expensive hardware and skills requirements. LLM-GNN [51] is a concurrent LLM-as-Annotator method that selectively prompts the LLM for feedback, but only considers a node classification task. Here, provided class labels ensure that the GNN and LLM are using aligned similarity functions, making it easier to obtain useful feedback. In contrast, on graph clustering, the LLM must infer as well as align with the GNN's implicit similarity function to provide meaningful feedback. On the other hand, LLM-as-Predictor methods seek to pass structural and textual attribute information directly to the LLM to make predictions. However, in our setting, where we assume a limited budget, it may be infeasible to prompt every node to obtain a cluster assignment. Other LLM-as-Predictor methods seek to perform graph-aware finetuning of PLMs and LLMs [52], which can also be expensive. Lastly, we note that to the best of our knowledge, graph clustering has not been explored by LLM-as-Predictor methods, so it is unclear if LLMs are able to infer sufficient topological information to effectively assign clusters.

C Expanded Prompting Discussion

Here, we expand the introduction of GCLR, our framework for graph clustering with LLM refinement (Fig. 1). In particular, we discuss, in more detail, how to obtain useful feedback for graph clustering from LLMs and then present how to identify and refine the initial solution accordingly.



(a) Triplet Based Feedback **(b) In-Context Based Prompt** **(c) Concept Based Prompt**

Figure 2: Example of LLM Feedback. Using the graph in Fig. 3, we prompt `chat-gpt-3.5-turbo` with different strategies to demonstrate the importance of aligning the LLM's and GNN's implicit similarity functions. Indeed, we see that triplet-based prompting can be unreliable as it does not allow the LLM to infer the underlying similarity. For example, with the query, "Baboon" with triplets containing the land animals from from Cluster 1 (starts with B) and aquatic animals from Cluster 2, the LLM assigns Baboon to cluster 1, which is consistent with the graph solution. However, when we prompt `chat-gpt-3.5` with a triplet containing *aquatic* animals from Cluster 1 and *land* animals from Cluster 2, the LLM assigns the query to Cluster 2 as it is also a land animal. In contrast, we find that both concept-based and incontext-based prompting are able to correctly infer the GNN's similarity function.

While feedback in traditional AL typically corresponds to an oracle selecting a label from a predefined set of classes, it is less clear what form the feedback should take when performing clustering. Intuitively, feedback should help improve the similarity of the queried node with the cluster that it belongs to. However, the precise form of the feedback may vary, and it's unclear how to prompt the LLM to accurately ascertain this information.

To this end, we discuss the advantages and disadvantages of three different strategies for prompting the LLM to obtain clustering feedback. We begin by discussing a recently proposed strategy for LLM guided text clustering.

Triplet-Based Prompting. ClusterLLM [27] is a recently proposed state-of-the-art LLM guided *text* clustering method that first selects uncertain samples (e.g., queries), $\mathcal{Q}_i$, and two random samples from each query's two nearest clusters, and then prompts the LLM to predict which of the two samples is "more similar" to $\mathcal{Q}_i$; the more similar sample is considered a "positive" sample and the other is a "negative" sample. Here, $\mathcal{D}_{\text{feedback}}$ corresponds to the set of triplets (query, positive, negative) determined by the LLM and $\mathcal{L}_{\text{feedback}}$ is InfoNCE. While such an approach can conceptually be applied to graph clustering, there are some limitations.

Insofar as clustering requires learning a similarity function that can be used to partition samples into meaningful groups, it is important that the oracle is aware of this function so the resulting feedback is aligned to the existing partitioning. In text clustering, since both the encoder (BERT, E5, etc) and the larger, oracle LLM (Chat-GPT, Llama) are text based models, they share a similar prior for this similarity function. In contrast, when performing graph clustering, the GNN incorporates topological information unavailable to the LLM and may utilize a different function than the LLM. Indeed, in Fig. 3, we construct a simple synthetic example where the GNN and LLM utilize different similarity functions to identify concepts by design. We observe, in Fig. 2a, that the oracle (`chat-gpt-3.5-turbo`) provides *un*reliable feedback when the triplet prompt contains random samples that do *not* overlap with the GNN's similarity function, but is reliable when the random samples are selected to align with the LLM's implicit similarity function.

Finally, we note that the performance of triplet-based feedback is closely tied to the quality of the initial clustering solution, artificially handicapping the LLM's performance. Given that the initial clustering solution is imperfect, randomly selecting samples from the two closest clusters can create triplets that do not actually represent the corresponding clusters, leading the LLM to perform a meaningless selection. Moreover, there is a loose upper-bound of the triplet formulation as the queries' "correct" cluster must be within the top-2 closest clusters. If this is not the case, the LLM will necessarily have to respond to an ill-formed triplet and will provide incorrect feedback. Due to the rapidly increasing capabilities of LLMs, it is possible that future LLMs will achieve perfect performance on valid triplets, however, the error incurred by ill-formed triplets is irreducible.

In-Context Similarity Learning. As discussed above, it is critical that the LLM can infer the similarity function implemented by the GNN. Given the impressive in-context learning capabilities of LLMs [28, 29], we consider a prompt that allows the LLM to directly infer it by providing several examples of the node's raw text and their corresponding cluster IDs, and the text of the unlabeled query (See Fig. 2b for an example.) Here, the LLM can be seen as performing a prediction task amongst pseudo-labels defined by the initial clustering, where $\mathcal{D}_{\text{feedback}} = \{([0, \dots K] | i \in \mathcal{Q}\}$. We note that the choice of $\mathcal{L}_{\text{feedback}}$ is flexible and discuss it in detail later. Notably, by ensuring that the prompt contains samples from all clusters, the LLM can (i) more holistically infer what concepts underlie clusters and (ii) predict an assignment for a query that does not belong to the top-2 clusters. This allows us to circumvent the previous issue where the upper-bound on refined performance was restricted by the number of samples where the preferred assignment was contained in the top-2 clusters.

However, directly inferring the similarity function from in-context examples becomes more difficult as the number of clusters grows as (i) the number of exemplars must correspondingly reduce to remain within the context length and (ii) if the number of clusters is sufficiently large, it is not possible to provide exemplars from all clusters. Furthermore, the selection and ordering of exemplars can have a significant impact of the LLM's ability to correctly predict a query's assignment, leading to potential loss of performance during fine-tuning.

Concept-based Prompting. To avoid the aforementioned issues with incontext-prompting, we draw inspiration from topic modeling [30, 31] and design an additional "concept-based" prompting strategy where we first prompt the LLM to infer the concepts that were used to group samples and then create a prediction task where the LLM is prompted to select amongst the generated concepts. (See Fig. 2c for an example.) To generate concepts, we provide the LLM samples from each cluster and ask it to provide a "title" and "short description" that explains how these samples are grouped together. These generated titles and descriptions are then provided as options for the LLM to identify the most similar cluster for a particular query. Notably, by providing the titles/descriptions of all clusters, we can avoid the upper-bound encountered by triplets while simultaneously allowing the LLM to at least partially infer the GNN's similarity function.

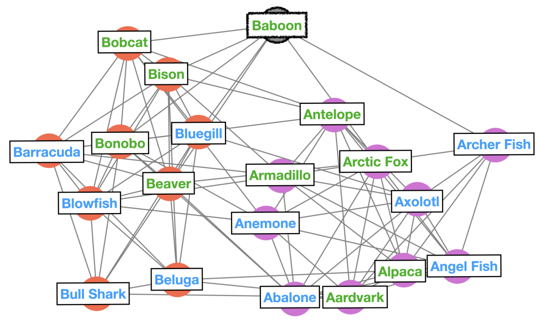


Figure 3: Unaligned Notions of Similarity. The following stochastic block model graph has clusters that correspond to whether a particular animal's name begins with **"A"** or **"B."** However, an alternative clustering according to **"land"** vs. **"aquatic"** animals is also valid and more semantically interesting. Indeed, when GPT-3.5 is asked whether a "Baboon" is more similar to a "Bluegill" or "Antelope," it replies with "Antelope" as it is also a land mammal. This emphasizes that (i) simple pairwise comparisons may not be sufficient for providing feedback and (ii) LLMs and GNN clustering algorithms may utilize disparate notions of similarity.

Moreover, by using cluster titles/descriptions instead of multiple exemplars per cluster, concept-based prompting uses much shorter prompts and better scale as the number of clusters grows in comparsion to in-context prompting. Indeed, as the number of clusters grows, In-Context prompting would require decreasing the number of exemplars per cluster to fit the context length. Moreover, this context must be passed every time feedback is obtained. In contrast, the titles/descriptions are generated once in a preprocessing step, and subsequently reused through a shorter, multiple choice-style prompt. Finally, we note that creating titles/descriptions may help denoise the exemplars as the LLM seeks to understand how they were grouped together.

D Ablation Results

Table 3: Effect of Confidence Filtering. While we do not know the reliability of either the LLM or GNN's feedback apriori, we can use their confidence to select samples where the feedback is more likely to be reliable to avoid finetuning on misleading samples. Here, we filter samples based on the ascending confidence percentile, so the 80th percentile corresponds to samples whose confidence is greater than or equal to 80% of total samples. We observe that filtering improves performance without filtering (11/24 Acc.) and over the starting (no finetuning) solution (17/24 Acc.). In particular, 80% LLM and 20% GNN filtering improves performance over no filtering (8/12 NMI, 10/12 Mod.) On WikiCS, no filtering performs the best, suggestive of the LLM's better reliability. Best performance is **bolded** and accuracy of the starting solution is in parentheses.

Dataset	Method	LLM	GNN	Acc.	NMI	F1	ARI	COND	MOD
citeseer	diffpool (47.09)	20	80	53.04	22.67	15.06	34.93	0.31	0.45
		80	20	56.71	**26.94**	**23.18**	**41.90**	**0.21**	**0.56**
		0	0	**58.96**	26.84	19.70	41.41	0.24	0.50
	dinknet (66.35)	20	80	67.61	38.14	32.03	50.99	**0.08**	0.64
		80	20	**67.43**	**40.23**	**37.88**	**56.47**	0.10	**0.67**
		0	0	67.40	36.97	27.16	47.91	0.09	0.62
	dmon (47.89)	20	80	**51.21**	26.85	18.27	31.64	**0.15**	0.50
		80	20	51.14	**30.06**	**25.30**	**41.72**	0.17	**0.59**
		0	0	49.87	27.12	14.46	29.87	**0.15**	0.47
	mincut (64.17)	20	80	61.42	31.79	26.94	47.84	0.26	0.56
		80	20	65.40	**41.32**	**38.01**	59.37	**0.13**	**0.69**
		0	0	**67.51**	39.60	35.81	**59.81**	0.17	0.64
cora	diffpool (59.97)	20	80	55.28	29.53	16.07	39.33	0.39	0.39
		80	20	**61.94**	**41.64**	**36.77**	**55.67**	**0.27**	**0.57**
		0	0	51.35	22.21	6.49	29.05	0.32	0.34
	dinknet (66.20)	20	80	67.15	36.21	24.09	42.83	0.13	0.50
		80	20	**67.87**	**48.03**	**36.82**	**52.04**	0.12	**0.66**
		0	0	65.16	23.42	9.25	27.40	**0.08**	0.29
	dmon (57.55)	20	80	58.07	36.72	24.99	40.19	0.23	0.47
		80	20	**62.06**	**41.79**	**35.56**	**50.52**	0.25	**0.57**
		0	0	56.70	30.06	13.67	29.40	**0.12**	0.33
	mincut (64.17)	20	80	61.04	38.40	28.22	48.95	0.34	0.50
		80	20	**64.55**	**47.15**	**38.89**	**57.82**	**0.19**	**0.65**
		0	0	61.62	41.61	30.54	54.01	0.28	0.54
wikics	diffpool (43.34)	20	80	51.53	27.52	17.87	**37.83**	0.41	0.39
		80	20	50.60	24.03	16.68	34.87	0.40	0.42
		0	0	**58.03**	**35.03**	26.28	46.48	**0.34**	**0.44**
	dinknet (71.25)	20	80	66.51	45.90	41.76	54.10	0.26	0.53
		80	20	66.79	48.39	41.85	55.66	**0.23**	**0.54**
		0	0	**74.00**	**51.25**	**51.57**	**63.06**	0.23	**0.54**
	dmon (37.515)	20	80	42.81	27.14	19.03	30.33	0.37	0.29
		80	20	40.92	28.11	20.24	32.39	0.46	**0.33**
		0	0	**51.87**	**32.51**	**31.04**	**36.49**	**0.26**	0.31
	mincut (24.70)	20	80	42.79	**19.16**	**7.50**	17.07	**0.27**	0.23
		80	20	43.58	14.74	3.77	19.57	0.30	0.14
		0	0	**46.36**	16.52	4.80	**24.36**	0.47	**0.27**

Table 4: Ensembling Improves Performance with Unreliable Feedback. Here, we create a deep ensemble by sampling different α and β to simulate different levels of confidence in each ensemble source. On Cora, where the LLM's feedback is known to be unreliable, we find that ensembling improves the performance of over a single model where $\alpha = 0.5$ and $\beta = 0.5$, and surpasses the performance of the starting solution as desired. Overall, this indicates that GCLR can help improve the initial clustering solution (highlighted in gray) even with unreliable feedback.

Method	Ens?	Acc.	NMI	F1	ARI	COND	MOD
diffpool	starting	59.97	43.36	36.58	56.76	0.24	0.60
	✗	51.35	22.21	6.49	29.05	0.32	0.34
	✓	**61.88**	**45.74**	**38.97**	**58.20**	**0.22**	**0.62**
dinknet	starting	68.26	51.98	44.21	62.09	0.12	**0.70**
	✗	65.16	23.42	9.25	27.40	**0.08**	0.29
	✓	**69.36**	**52.66**	**45.28**	**63.12**	0.12	**0.70**
dmon	starting	57.56	41.60	33.76	50.94	0.27	0.56
	✗	56.70	30.06	13.67	29.40	**0.12**	0.33
	✓	**60.60**	**43.25**	**37.60**	**52.41**	0.24	**0.58**
mincut	starting	64.17	48.92	40.35	58.33	**0.14**	**0.70**
	✗	61.62	41.61	30.54	54.01	0.28	0.54
	✓	**64.63**	**48.96**	**40.77**	**58.79**	**0.14**	**0.70**

Table 5: Query Function Ablation. We report performance on the following query strategies: random sampling \ entropy sampling \ least confidence \ margin sampling. We observe that while there is a slight decrease in performance when using random sampling as the query function, overall margin sampling perform similarly to entropy sampling. Least confidence sampling, in fact, improves performance on a few cases.

Dataset	Method	Acc.	NMI	F1	ARI	COND	MOD
citeseer	diffpool	49.45\59.56\60.19\59.38	26.47\27.75\28.38\23.21	8.47\13.16\12.88\8.74	33.87\31.93\31.16\29.23	0.16\0.11\0.09\0.1	0.39\0.34\0.33\0.29
	dinknet	46.14\56.99\58.16\57.9	6.62\35.41\35.8\36.2	2.81\22.96\22.28\22.96	10.65\37.79\37.39\40.21	0.02\0.22\0.22\0.21	0.07\0.43\0.43\0.44
	dmon	37.94\51.74\52.18\51.31	7.73\27.88\28.34\27.54	2.53\18.42\18.2\17.98	11.86\33.67\33.88\32.75	0.06\0.15\0.15\0.15	0.16\0.51\0.5\0.5
	mincut	64.08\63.4\63.56\61.16	34.45\34.74\35.16\39.89	30.4\31.31\31.42\30.13	55.48\54.43\55.27\49.44	0.24\0.23\0.23\0.31	0.56\0.58\0.58\0.52
cora	diffpool	68.7\66.53\66.55\66.52	43.99\45.88\45.32\45.52	36.35\42.32\42.02\41.83	55.24\54.14\53.08\53.96	0.32\0.26\0.26\0.26	0.5\0.53\0.53\0.53
	dinknet	35.33\42.7\42.4\42.67	14.46\26.75\26.51\26.92	10.65\19.01\18.61\19	11.3\30.6\30.01\30.12	0.08\0.39\0.38\0.37	0.13\0.29\0.29\0.29
	dmon	46.14\56.99\58.16\57.9	6.62\35.41\35.8\36.2	2.81\22.96\22.28\22.96	10.65\37.79\37.39\40.21	0.02\0.22\0.22\0.21	0.07\0.43\0.43\0.44
	mincut	61.16\61.52\60.5\60.61	39.89\40.94\39.78\40.05	30.13\29.34\29\30.1	49.44\50.6\50.34\50.5	0.31\0.31\0.33\0.32	0.52\0.51\0.5\0.5
wikics	diffpool	37.94\51.74\52.18\51.31	7.73\27.88\28.34\27.54	2.53\18.42\18.2\17.98	11.86\33.67\33.88\32.75	0.06\0.15\0.15\0.15	0.16\0.51\0.5\0.5
	dinknet	64.08\63.4\63.56\61.78	34.45\34.74\35.16\33.38	30.4\31.31\31.42\29	55.48\54.43\55.27\54.02	0.24\0.23\0.23\0.25	0.56\0.58\0.58\0.57
	dmon	35.33\42.7\42.4\42.67	14.46\26.75\26.51\26.92	10.65\19.01\18.61\19	11.3\30.6\30.01\30.12	0.08\0.39\0.38\0.37	0.13\0.29\0.29\0.29
	mincut	46.4\46.92\44.46\45.76	22.06\18.61\20.11\18.19	11.57\5.6\6.79\9.08	28.09\22.16\22.67\22.37	0.27\0.24\0.28\0.21	0.22\0.16\0.2\0.16

Table 6: Ablation on the Labeling Budget. We report performance when the LLM labeling budget is 20% \ 40% \ 60% \ 80% \ 100%. We find that increasing the budget does not substantially increase performance, unlike traditional active learning. We hypothesize this is partially due to regularizing training using GNN pseudo-labels and the imperfect LLM oracle. This suggests that maximizing the API budget by labeling every possible node may have diminishing returns. By selecting the most difficult samples early on, GCLR more effectively uses any sized budget.

Dataset	Method	Acc.	NMI	F1	ARI	COND	MOD
citeseer	diffpool	54.43\53.82\52.77\53.05\54.66	23.52\22.7\22.59\22.76\22.21	15.33\15.3\15.2\15.54\15.23	35.44\36.46\36.72\36.88\36.52	0.3\0.3\0.31\0.32\0.32	0.45\0.45\0.45\0.44\0.44
	dinknet	69.81\69.84\69.81\69.81\69.81	34.15\36.98\36.61\36.38\36.19	26.36\29.83\29.23\28.97\28.82	45.51\48.06\47.35\47.13\46.93	0.07\0.07\0.07\0.07\0.07	0.6\0.62\0.61\0.61\0.61
	dmon	51.81\51.63\51.62\51.3\51.25	28.17\29.15\28.99\29.19\29.1	18\18.82\18.37\18.65\18.56	31.84\32.72\32.66\32.53\32.45	0.13\0.14\0.15\0.15\0.15	0.5\0.5\0.5\0.5\0.5
	mincut	63.57\61.68\63.12\63.98\64.14	34.44\32.88\33.4\32.94\32.7	29.95\27.77\28.04\27.41\27.23	55.75\54.3\54.29\53.44\52.27	0.26\0.31\0.3\0.31\0.3	0.55\0.51\0.51\0.51\0.51
cora	diffpool	55.55\55.44\55.61\56.67\56.53	24.17\24.81\24.9\24.97\25.18	9.91\10.06\10.35\10.9\11.02	31.91\32.92\33.57\34.3\34.25	0.41\0.43\0.44\0.44\0.44	0.34\0.33\0.32\0.33\0.34
	dinknet	59.97\60.01\60.01\59.97\60.04	24.84\25.72\25.36\25.53\25.23	10.19\10.89\10.43\10.43\10	30.36\30.74\30.58\30.79\30.47	0.09\0.09\0.09\0.09\0.09	0.32\0.33\0.32\0.32\0.31
	dmon	58.14\58.89\59\58.91\58.91	35.71\36.31\36.95\37.39\37.5	22.31\21.97\21.82\21.85\22.25	38.78\37.71\37.93\39.24\39.44	0.21\0.22\0.21\0.2\0.19	0.43\0.43\0.44\0.44\0.45
	mincut	60.43\62.17\59.4\60.54\60.76	38.36\37.51\38.47\38.18\38.84	28.27\27.65\28.51\28.37\29.36	48.36\47.79\48.61\47.87\47.84	0.36\0.37\0.38\0.37\0.38	0.47\0.46\0.45\0.46\0.46
subtagwikics	diffpool	49.75\51.71\52.18\51.31\52.5	30.03\30.54\30.14\31.23\30.73	20.62\22.71\19.96\19.74\20.97	40.33\40.68\37.45\38.23\39.88	0.43\0.39\0.39\0.39\0.41	0.39\0.42\0.42\0.38\0.4
	dinknet	66.56\66.54\66.54\66.51\66.52	46.13\45.77\45.73\45.68\45.62	42.49\42.43\42.31\42.14\42.05	54.59\54.27\54.16\54.14\54.21	0.26\0.25\0.26\0.26\0.26	0.53\0.53\0.53\0.53\0.53
	dmon	42.99\42.83\42.9\43.05\43.13	27.31\27.7\27.83\28.09\28.32	19.24\19.35\19.57\19.73\19.97	30.42\30.68\30.76\31.06\31.12	0.37\0.37\0.37\0.36\0.36	0.29\0.29\0.3\0.3\0.3
	mincut	44.91\44.01\44.53\43.05\44.98	18.36\17.12\20.1\19.83\18.77	5.11\5.97\9.44\9.48\6.2	21.5\18.35\25.49\18.43\19.59	0.28\0.22\0.3\0.3\0.26	0.16\0.14\0.15\0.17\0.19

E Additional Results with ChatGPT

In this section, we evaluate GCLR using feedback obtained from ChatGPT-3.5-Turbo, instead of Mixtral-8b, to demonstrate its robustness to choice of LLM. We note that obtaining feedback from ChatGPT is fairly expensive for us, so we only obtain feedback on 200 nodes. We select the 200 most difficult nodes for feedback, where difficult is defined according to the entropy of the distance to a sample's two nearest clusters. Here, a sample that is equidistant and relatively from the cluster centers would is more difficult and is selected over a sample that is close to a single center (well-clustered). Given the strong performance of GCLR with even this limited number of samples from a very powerful LLM, suggests that performance would be further improved with a larger budget. We note that due to the limited number of feedback samples, we perform a single round of fine-tuning to prevent over-fitting to feedback samples, instead of dividing the feedback over multiple rounds. Finally, please note that we had to retrain the base GNNs for these experiments, so the starting accuracy of the original GNNs may be slightly different that those reported in the main paper. All results are reported using the "concepts" feedback strategy unless otherwise noted. We strongly emphasize, however, that we are interested in observing the improvement of GCLR relative to the starting model, and we clearly observe its benefits in the following tables.

Table 7: Feedback Elicitation. We evaluate three different strategies for obtaining LLM guidance by measuring their accuracy in predicting the correct cluster assignment (wrt to known ground-truth label) on the 200 *hardest* samples as per the initial GNN clustering. The GNN's accuracy on **ALL** samples is reported in parenthesis. We observe that the Concepts strategy achieves the best performance on 10/12 datasets.

Dataset	Clustering Method	Concepts	InContext	Triplets
citeseer	diffpool (0.496)	**0.295**	0.24	0.26
	dinknet (0.703)	**0.385**	**0.385**	0.38
	dmon (0.441)	**0.415**	0.34	0.35
	mincut (0.665)	**0.415**	0.33	0.37
cora	diffpool (0.547)	0.14	**0.29**	0.29
	dinknet (0.658)	**0.355**	0.235	0.295
	dmon (0.609)	**0.355**	0.15	0.31
	mincut (0.684)	**0.54**	0.25	0.235
WikiCS	diffpool (0.483)	**0.365**	0.290	0.235
	dinknet (0.665)	0.24	0.240	**0.330**
	dmon (0.370)	**0.335**	0.235	0.27
	mincut (0.269)	**0.08**	0.01	0.015

Table 8: ChatGPT Provides Complementary Information When Finetuning In order to demonstrate ChatGPT provided labels capture complementary, *beneficial* information to the GNN, here, we compare performance of models that were *only* fine-tuned with GNN pseudo-labels and those that were fine-tuned with *GNN and LLM pseudo-labels*. Notably, we do *not* filter the LLM's nor the GNN labels for high confidence; allowing the mistakes from either source. The better result is underlined between (GNN Only / LLM+GNN). We observe that incorporating the raw LLM feedback improves the clustering solution noticeably on the extrinsic metrics (7/12 Acc), (10/12 NMI), (9/12 F1) but has mixed, but competitive performance on extrinsic metrics.

Method	*Dataset*	*Acc*	*NMI*	*ARI*	*F1*	*Cond*	*Mod*
DiffPool	Citeseer	54.110 / 55.740	33.710 / 36.240	27.430 / 30.920	45.290 / 49.100	0.146 / 0.164	0.633 / 0.630
DinkNet	Citeseer	69.520 / 69.718	45.200 / 45.733	44.370 / 45.343	65.330 / 65.570	0.068 / 0.065	0.701 / 0.706
Dmon	Citeseer	46.400 / 49.030	29.585 / 30.550	24.295 / 26.670	43.395 / 44.230	0.210 / 0.199	0.582 / 0.573
MinCut	Citeseer	67.360 / 67.950	46.520 / 46.960	44.820 / 46.000	65.160 / 65.420	0.081 / 0.078	0.726 / 0.720
DiffPool	Cora	60.160 / 59.270	45.790 / 39.550	40.320 / 29.880	52.350 / 51.340	0.200 / 0.211	0.610 / 0.511
DinkNet	Cora	60.860 / 64.700	47.930 / 50.420	33.520 / 36.440	50.530 / 55.940	0.124 / 0.110	0.620 / 0.642
Dmon	Cora	62.080 / 61.410	42.345 / 42.615	35.055 / 33.995	54.220 / 53.885	0.241 / 0.253	0.581 / 0.574
MinCut	Cora	68.650 / 71.530	52.270 / 53.830	47.050 / 49.950	63.740 / 64.960	0.146 / 0.152	0.705 / 0.691
DinkNet	WikiCS	63.510 / 62.770	49.680 / 49.230	44.050 / 43.730	59.130 / 58.520	0.243 / 0.245	0.536 / 0.540
DiffPool	WikiCS	52.390 / 52.070	37.500 / 39.500	27.520 / 28.820	48.230 / 46.820	0.304 / 0.294	0.504 / 0.513
Dmon	WikiCS	38.420 / 38.390	30.420 / 30.910	23.140 / 23.400	33.380 / 33.390	0.444 / 0.438	0.358 / 0.367
MinCut	WikiCS	30.430 / 34.370	17.040 / 21.750	0.900 / 4.490	12.810 / 16.160	0.054 / 0.073	0.118 / 0.134

Table 9: GCLR with ChatGPT Improves the Performance of Graph Clustering Solutions. Here, we consider GCLR's performance across different confidence filtering levels (for both the GNN and LLM), and compare its performance when using the triplet loss (instead of cross entropy). In particular, we consider two different confidence percentiles, 20% and 80%, denoted low and high below respectively. Aside from DinkNet, which uses a contrastive loss during training, we find that GCLR with cross-entropy and confidence filtering improves the performance over the starting GNN solution. The performance of starting GNN solution is denoted in parenthesis. Second Best, First. (Cross Entropy /Triplets).

Dataset	*Method*	*LLM Conf.*	*GNN Conf.*	*Acc*	*NMI*	*ARI*	*F1*	*Cond*	*Mod*
Citeseer	DiffPool (49.6)	low	low	53.360 / 49.560	34.460 / 28.730	30.040 / 26.320	46.430 / 44.770	0.175 / 0.199	0.619 / 0.606
Citeseer	DiffPool	low	high	52.480 / 47.390	32.050 / 25.990	27.670 / 23.830	45.110 / 41.160	0.160 / 0.232	0.618 / 0.565
Citeseer	DiffPool	high	low	51.570 / 49.690	36.200 / 28.920	29.870 / 26.460	43.270 / 44.830	0.153 / 0.200	0.639 / 0.606
Citeseer	DinkNet (70.3)	low	low	69.400 / 71.030	45.440 / 46.370	44.600 / 49.500	65.130 / 66.640	0.072 / 0.066	0.696 / 0.721
Citeseer	DinkNet	low	high	64.780 / 70.090	42.740 / 45.600	38.540 / 47.720	59.130 / 65.390	0.066 / 0.064	0.655 / 0.717
Citeseer	DinkNet	high	low	69.240 / 71.030	44.650 / 46.320	44.130 / 49.430	64.680 / 66.610	0.072 / 0.067	0.696 / 0.720
Citeseer	Dmon (44.1)	low	low	48.780 / 45.135	30.090 / 29.680	26.050 / 27.645	45.570 / 36.660	0.219 / 0.191	0.574 / 0.547
Citeseer	Dmon	low	high	50.410 / 46.030	32.230 / 30.945	27.880 / 29.435	44.750 / 37.540	0.193 / 0.167	0.577 / 0.571
Citeseer	Dmon	high	low	48.590 / 45.200	30.620 / 29.690	26.110 / 27.675	45.650 / 36.735	0.208 / 0.191	0.583 / 0.546
Citeseer	MinCut (66.50)	low	low	68.490 / 70.030	47.370 / 47.650	46.950 / 48.360	65.620 / 66.170	0.075 / 0.061	0.719 / 0.740
Citeseer	MinCut	low	high	68.680 / 71.940	47.570 / 48.860	47.260 / 50.600	65.570 / 67.350	0.072 / 0.065	0.717 / 0.729
Citeseer	MinCut	high	low	68.080 / 70.030	46.950 / 47.690	46.260 / 48.400	65.540 / 66.180	0.072 / 0.061	0.729 / 0.740
Cora	DiffPool (54.7)	low	low	59.710 / 53.210	41.250 / 39.010	33.440 / 32.120	51.400 / 49.130	0.213 / 0.268	0.542 / 0.571
Cora	DiffPool	low	high	59.310 / 54.470	39.610 / 40.450	30.640 / 33.220	49.860 / 49.720	0.223 / 0.245	0.506 / 0.587
Cora	DiffPool	high	low	61.630 / 53.360	42.590 / 39.110	40.430 / 32.340	53.510 / 49.220	0.206 / 0.270	0.571 / 0.568
Cora	DinkNet (65.8)	low	low	63.770 / 65.030	47.270 / 49.510	36.480 / 42.510	53.760 / 54.900	0.127 / 0.118	0.639 / 0.680
Cora	DinkNet	low	high	63.070 / 64.550	45.450 / 50.640	35.810 / 42.200	53.600 / 54.490	0.151 / 0.110	0.639 / 0.687
Cora	DinkNet	high	low	61.630 / 65.140	43.900 / 49.620	34.460 / 42.620	48.200 / 55.050	0.144 / 0.119	0.629 / 0.680
Cora	Dmon (60.9)	low	low	61.340 / 55.650	42.910 / 36.625	33.880 / 26.955	54.080 / 47.400	0.244 / 0.296	0.585 / 0.514
Cora	Dmon	low	high	44.500 / 56.280	32.140 / 38.110	15.390 / 27.680	33.090 / 48.590	0.201 / 0.282	0.425 / 0.526
Cora	Dmon	high	low	62.520 / 55.595	42.380 / 36.620	35.390 / 26.895	54.860 / 47.365	0.233 / 0.295	0.592 / 0.514
Cora	MinCut (68.4)	low	low	71.680 / 71.230	53.920 / 54.960	50.750 / 48.640	65.570 / 62.940	0.150 / 0.128	0.695 / 0.704
Cora	MinCut	low	high	72.050 / 71.900	54.070 / 54.510	51.270 / 49.910	66.740 / 63.380	0.154 / 0.128	0.692 / 0.702
Cora	MinCut	high	low	71.530 / 71.310	53.750 / 55.100	50.890 / 48.790	65.140 / 63.020	0.153 / 0.127	0.691 / 0.704
WikiCS	DiffPool (48.3)	low	low	54.330 / 52.100	40.280 / 31.260	31.920 / 27.480	48.220 / 45.690	0.291 / 0.294	0.515 / 0.503
WikiCS	DiffPool	low	high	54.240 / 51.960	40.350 / 29.060	32.290 / 26.800	47.040 / 45.330	0.279 / 0.301	0.520 / 0.490
WikiCS	DiffPool	high	low	52.680 / 51.920	37.510 / 31.090	29.500 / 27.230	47.030 / 45.590	0.294 / 0.293	0.506 / 0.503
WikiCS	DinkNet (66.5)	low	low	62.470 / 65.800	48.900 / 48.780	43.380 / 44.380	58.020 / 59.610	0.243 / 0.233	0.546 / 0.548
WikiCS	DinkNet	low	high	62.760 / 64.890	48.920 / 47.640	43.400 / 42.900	58.120 / 58.090	0.243 / 0.244	0.545 / 0.552
WikiCS	DinkNet	high	low	61.870 / 65.800	48.750 / 48.780	43.170 / 44.380	57.220 / 59.610	0.245 / 0.233	0.544 / 0.548
WikiCS	Dmon (37.0)	low	low	38.870 / 36.970	31.080 / 29.350	23.600 / 23.550	33.700 / 32.560	0.430 / 0.463	0.373 / 0.347
WikiCS	Dmon	low	high	44.690 / 37.260	33.850 / 27.830	26.210 / 23.130	38.310 / 32.240	0.381 / 0.462	0.401 / 0.342
WikiCS	Dmon	high	low	38.350 / 36.940	30.910 / 29.350	23.530 / 23.530	33.220 / 33.140	0.436 / 0.463	0.368 / 0.348
WikiCS	MinCut (26.90)	low	low	37.230 / 26.860	23.030 / 12.770	8.150 / -0.770	17.410 / 10.480	0.072 / 0.055	0.143 / 0.064
WikiCS	MinCut	low	high	40.750 / 28.660	22.270 / 10.920	13.870 / 3.400	19.520 / 11.750	0.083 / 0.337	0.182 / 0.139
WikiCS	MinCut	high	low	36.430 / 26.620	21.990 / 13.200	6.820 / -0.770	16.090 / 10.320	0.058 / 0.058	0.139 / 0.069

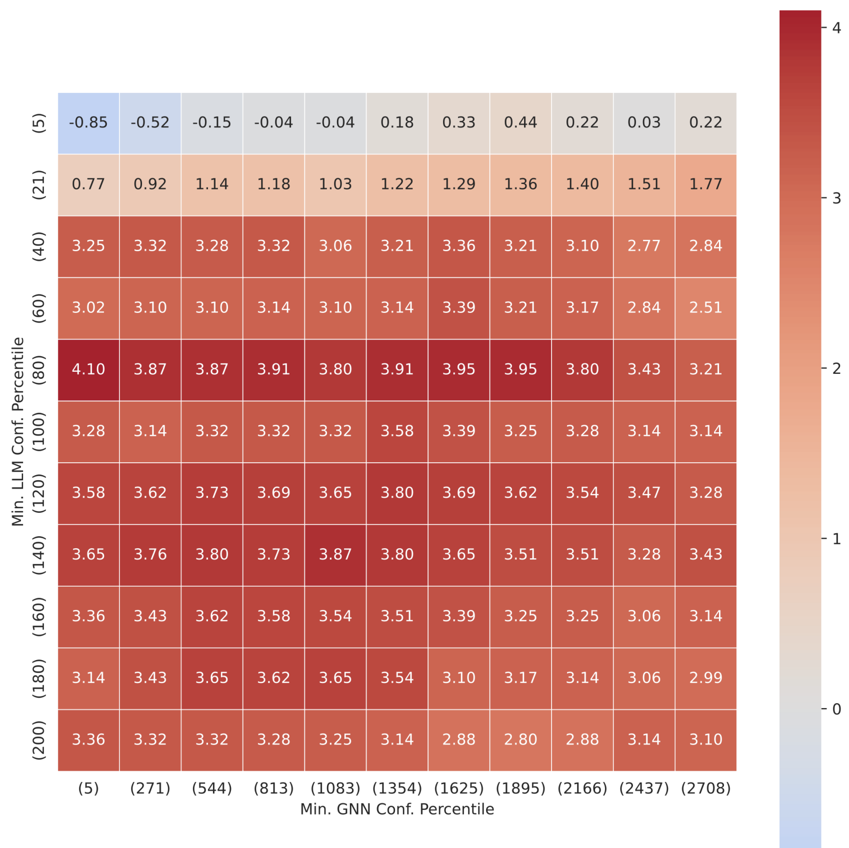


Figure 4: Ablation on Sensitivity to Confidence with ChatGPT Feedback. Here, we consider the sensitivity of GCLR to the confidence filtering percentiles. Namely, we take only the top [0,10,—, 100]th percentile of the feedback data and report the change in accuracy to the starting solution using the CORA dataset. The number of samples at a particular percentile are indicated in parentheses, α and β are set to 0.5. We see that the best performance is obtained at a moderate confidence percentile for both the GNN and LLM.

F Prompt Examples

Table 10: Prompt Example: Triplets, CORA

PROMPT: Task: I'm clustering papers in a citation network according to research area and need help determining where a particular query sample belongs given its abstract and title. I will give you the abstracts/titles of two samples belonging to nearby clusters and you should select the abstract/title that is more similar to the query in terms of research topic. Please explain your reasoning and return your answer in a JSON format: {selection: [1,2,-1(neither or unsure)], reasoning: [your reasoning]}.

[SAMPLE 1]
`<Sample from 1st (2nd) Closest Cluster>]`

[SAMPLE 2]
`<Sample from 2nd (1st) Closest Cluster>]`

[QUERY]
`<Sample of Query Sample>]`

[ANSWER]

Table 11: Prompt Example: Incontext, CORA

PROMPT:
[Example]
`<Sample>`
{Category: `<GNN's Predicted Cluster>`}

...
[Example]
`<Sample>`
{Category: `<GNN's Predicted Cluster>`}

[Task]
Given the above examples, please identify the correct category for the following query sample. Please explain your reasoning and return your answer in a JSON format: category: [your prediction], reasoning: [your reasoning]. If you're unsure of an answer, select category -1.

[QUERY]
`<QUERY>`

[ANSWER]

Table 12: Prompt Example: Concepts, CORA

CONCEPTS GENERATION PROMPT: Task: I'm clustering papers in a citation network according to research area and need help coming up with cluster names. The following `num-exemplars` papers that have been clustered together and I'm going to give you their abstract/titles. Can you propose a <7 word research topic and 2-3 sentence description for this cluster? Try not to make it too specific or too broad, and explain your reasoning. Return your answer in a JSON format: {topic: [your topic], description: [your description], reasoning: [your reasoning]}.

SAMPLES FROM CLUSTER:
`Sample 1`
`Sample 2`
`...`
`Sample Num-Exemplars`

Answer:

CONCEPT PREDICTION PROMPT:
[Task]
I'm currently working on clustering papers within a citation network based on their abstracts/titles. I'm seeking assistance in determining the cluster association for a specific uncertain sample. You'll be provided with the abstract/title of this sample, along with the titles and short descriptions of `num-clusters` potential clusters. Your task involves carefully reading each cluster title and description, taking a thoughtful approach, and selecting the cluster that best aligns with the confusing sample. Please provide your answer in JSON format, including the predicted cluster number, title of the predicted cluster, and your detailed reasoning. Your response should look like this: {cluster: [your predicted cluster number], cluster title: [title of predicted cluster], reasoning: [your reasoning for choosing this cluster]}. Take your time and ensure clarity in your explanation.

[CLUSTER TITLES]
1. <GENERATED TITLE>
Description: <GENERATED TITLE DESCRIPTION>

2. <GENERATED TITLE>
Description: <GENERATED TITLE DESCRIPTION>

...
NUM-CLUSTERS. <GENERATED TITLE>
Description: <GENERATED TITLE DESCRIPTION>

[UNCERTAIN SAMPLE]
QUERY

[ANSWER]

G Metrics

We consider the following extrinsic and graph topology-based metrics in our evaluation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{X}, [\mathcal{Y}])$ represent a graph with its respective node-set, edge-set, raw node based text information, embedded node attribute information (e.g., some embedding of a node's text), and optional ground-truth cluster assignment. Further, let N be the number of the nodes, M be the number of edges, C be the desired (or ground-truth) number of clusters, d the dimension of the hidden representation, $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the corresponding adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N \times d}$ be a matrix representation of $\mathcal{X}$, $\mathbf{Y} \in [0,1]^C$, $\mathbf{d}_v$ be the degree vector of a particular node v, and c_v be the *predicted* cluster of a given node v.

- **Modularity [1].** Modularity measures the deviation with respect to nodes belonging to the same cluster against the expectation of the nodes being connected given a null model where nodes are connected randomly. Graphs with high modularity will have clusters where the majority of the edges are contained with some cluster and few edges that cross the clusters. Modularity falls within $[-\frac{1}{2}, 1]$, where a positive score indicates that the clustering structure that is above random, and is defined as follows:

$$Q = \frac{1}{2m} \sum_{ij} \left[\mathbf{A}_{[ij]} - \frac{d_i d_j}{2m} \right] \mathbb{1}[c_i = c_j].$$

- **Conductance [3, 53].** Also known as the Cheeger coefficient, this metric measures how quickly a random walk on a graph will reach its stationary distribution. Given a particular cluster, $\hat{c}$, the number of edges belonging to that cluster (intra-cluster edges) can be computed as $r_{\hat{c}} = \sum_{u,v \in \mathbf{A}} \mathbb{1}[c_u = \hat{c}, c_v = \hat{c}]$, and the number of edges are not fully contained in $\hat{c}$ (inter-cluster edges) can be computed as $s_{\hat{c}} = \sum_{u,v \in \mathbf{A}} \mathbb{1}[c_u = \hat{c}, c_v \neq \hat{c}]$. Then, conductance is defined as the average ratio of intra- and inter- cluster edges, where tight clusters are expected to have relatively fewer inter cluster edges.

$$\phi = \frac{1}{C} \sum_{\hat{c}}^{C} \frac{s_{\hat{c}}}{r_{\hat{c}} + s_{\hat{c}}}$$

- **Accuracy.**

$$ACC = \sum_{i=1}^{n} \frac{\phi(y_i, map(\hat{y_i}))}{n} \tag{1}$$

  $\hat{y_i}$ represents the predicted cluster ID, while y_i indicates the ground truth cluster ID label. $map(.)$ denotes the Kuhn-Munkres algorithm [54] which aligns the predicted cluster-ID with the class-ID, and indicator function $\phi(.)$ is formulated as:

$$\phi(y_i, map(\hat{y_i})) = \begin{cases} 1 & \text{if } y_i = map(\hat{y_i}) \\ 0 & \text{else} \end{cases} \tag{2}$$

- **Normalized Mutural Information.**

$$NMI = -\frac{2 \sum_{\hat{y}} \sum_{y} p(\hat{y}, y) \log \frac{p(\hat{y},y)}{p(\hat{y})p(y)}}{\sum_i p(\hat{y_i}) \log (p(\hat{y_i})) + \sum_j p(y_j) \log (p(y_j))} \tag{3}$$

  where $p(y), p(\hat{y})$, and $p(\hat{y}, y)$ represent the distribution of predicted results, distribution of the ground truth, and joint distribution of them, respectively.

- **Adjusted Random Index.**

$$ARI = \frac{RI - expectedRI}{max(RI) - expectedRI} \tag{4}$$

  where RI and $expectedRI$ signifies the Rand Index and expected Rand Index [55], respectively. An ARI of 0 suggests disagreement between real and modeled clustering in pairing, whereas an ARI of 1 indicates concordance between real and modeled clustering, representing identical clusters.

- **F1-Score.**

$$F1 = \frac{2.Precision.Recall}{Precision + Recall} \tag{5}$$

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN} \tag{6}$$

where TP, FP, and FN indicate the number of true positive, false positive, and false negative samples, respectively.

H Reproducibility Statement

All code will be released upon acceptance. We dropped the computation linguistic and web-technology categories from WikiCS to create a more even and separate labeling for evaluation. We use the mixtral-8x-7b model, and a G.5 (8 gpu) instance on AWS. We repeat results over 3 seeds for obtaining feedback. We used 10 seeds for finetuning. We provide an anonymous code repo at `https://anonymous.4open.science/r/GCLR_ARR-EDD8/`.

Table 13: Dataset Statistics.

Dataset	Num Nodes	Num Edges	Num Clusters
Cora [56]	2,708	5,429	7
Citeseer [57]	3,327	4,732	6
WikiCS[1] [58]	10,601	204120	8

I Example of Generated Titles

Table 14: Generated Concepts. Below, are examples of concepts generated by `chatgpt-3.5-turbo` on Cora with MinCut as the GNN clustering algorithm. While some concepts are imperfect, e.g., rule learning or theory, other topics are well captured. Applying self-refinement strategies could improve these generated concepts, at additional budget expenditure.

True	Generated
Reinforcement Learning	Reinforcement Learning and Dynamic Programming
Genetic Algorithms	Evolutionary Algorithms in Problem Solving
Rule Learning	Error Bounds in Learning Algorithms
Theory	Feature Selection in Machine Learning'
Probabilistic Methods	Bayesian Statistical Methods
Case Based	Improving Case-Based Reasoning Adaptation
Neural Networks	Neural Network Self-Organization