Can Inherent Graph Structure Enhance In-Context Learning?

Anonymous Author(s)

Abstract

In-context learning (ICL) is one of the key capabilities of large language models (LLMs), allowing them to generate accurate outputs given a set of input-output examples and a test query. In fact, the effectiveness of ICL heavily depends on the quality and modeling of selected examples. Existing methods typically focus on simple heuristic strategies or learnable models to select and sort the examples for improving ICL. However, they overlook the **inherent structural relationships** among examples and the reasoning information embedded within these relationships. Moreover, as the number of examples increases, the computational overhead of LLMs grows quadratically, reducing the efficiency of ICL. To address these challenges, we propose ContextG, a framework that mines and encodes the inherent structure among the examples and test query to boost the ICL. By explicitly introducing structure, our method enables the LLM with a deeper understanding of the relationships among examples, potentially enhancing its reasoning ability. Specifically, we first select examples based on similarity and construct graph structures consisting of examples and the test query. Then, we filter the examples again based on their structural importance. Finally, we hierarchically encode the structure and align with textual data to enrich the contextual information. Extensive experiments show the effectiveness and efficiency of our model. On commonly used benchmark datasets, our method outperforms the SOTA by an average of 2.74%. In task transfer settings, we also achieve the highest average performance. As the first work to introduce graph structures into ICL, ContextG provides deeper insights into the in-context modeling of LLMs.

CCS Concepts

• **Computing methodologies** → **Information extraction**.

Keywords

In-context learning, structure learning, graph learning

ACM Reference Format:

Anonymous Author(s). 2018. Can Inherent Graph Structure Enhance In-Context Learning?. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In-context learning (ICL) has emerged as a powerful capability of large language models (LLMs) [3]. By providing LLMs with a few input-output pairs, termed demonstrations, ICL allows models to infer the desired task and generalize effectively to unseen inputs [8]. Due to the large parameter size of LLMs, training them is often infeasible or costly. As a result, ICL, which allows LLMs to adapt to different data and tasks without parameter updates, has gained significant attention as an efficient alternative [33].

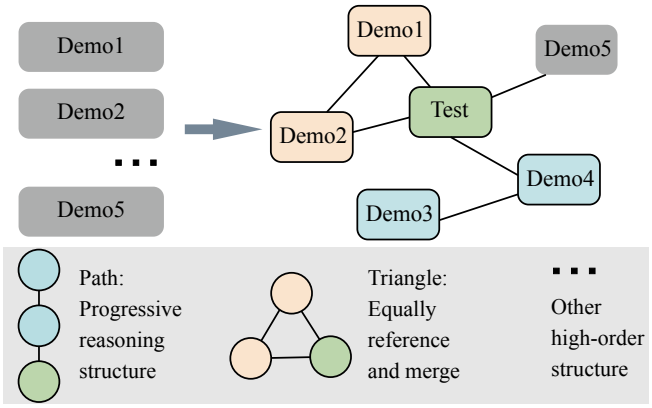

Figure 1: Illustration of the meaning of mining the graph structure representing the relationships between examples and the test query.

Numerous experimental [59] and theoretical [6, 54] studies have shown that the quality of demonstrations significantly impacts the performance of LLMs on the target tasks. Moreover, due to the limited length of the input context window in LLMs, and the quadratic growth in both computation time and memory usage caused by the self-attention mechanism as the input size increases [24], handling large numbers of examples becomes increasingly challenging. This presents a key limitation in real-world scenarios, where both the *quality* and *efficiency* of ICL must be carefully managed.

To address the quality issue, existing methods primarily focus on selection and ordering of demonstrations, employing various heuristic [43] or learnable approaches [55] to obtain higher-quality example sets. For the efficiency issue, current solutions mainly rely on compressing the length of example text [12] or distilling the text into vectors [28].

However, these methods overlook the **inherent graph structure** among the examples and the test query. Specifically, while many approaches provide high-quality examples, they fail to capture the high-order relationships among them and the reasoning potential within these relationships. It is well known that graph structures are often used as carriers for storing knowledge and reasoning logic [32, 42]. As shown in Fig. 1, incorporating structure among examples (demos) provides LLMs with more than just individual examples, enabling progressive reasoning in path structures, joint reference in triangle structures, *etc.* Furthermore, when compressing examples into vectors, existing methods focus on preserving the semantic information of the samples themselves, while

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, Woodstock, NY

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

neglecting the high-order contextual correlation that can be incorporated by encoding the relationships among examples.

Therefore, we propose to capture such informative inherent graph structure, which allows us to simultaneously capture the relationships among the test query and the examples, as well as provide an additional reasoning structure for the LLMs. Moreover, since the nodes in the graph structure have no order, we can jointly encode the structure and text of the examples, efficiently and effectively eliminating the impact of order.

Note that the method for constructing the graph structure should both reflect the textual relationships between examples and enable effective example selection. Thus, how to construct informative graph structure among both examples and the test query is the first challenge. Moreover, the constructed graph structure cannot be directly utilized or understood by the LLM. Thus, the second challenge is how to jointly encode the graph structure and the examples in a way that not only preserves the relational information among examples but also aligns with the textual information.

In this work, we introduce ContextG to efficiently and effectively enhance the performance of ICL. Specifically, we first retrieve a set of candidate examples and construct learnable graph structure among them and the test query. Based on this structure, we calculate the importance of each node and select the most crucial examples as representatives. This addresses the first challenge of mining structural relationships and selecting the examples. On the other hand, the graph structure requires comprehensive encoding, as well as the different modalities of the graph and the text requires fusion. We propose hierarchical structural information modeling at three levels—node positional, path, and subgraph. Moreover, we align this with the textual content to form the final embedding vectors. This resolves the second challenge of enabling LLMs to grasp the reasoning logic within the structure.

Empirically, we show that our approach outperforms SOTA baselines on multiple datasets by an average of 2.74%. Additionally, task transfer experiments demonstrate that the learnable components can transfer across different task contexts, providing evidence that our method efficiently and effectively enhances ICL performance.

The contributions of this paper are summarized as follows:

- We propose extracting inherent graph structures to introduce structural information among in-context examples and query that can be utilized for reasoning.
- We introduce an in-context learning framework, ContextG, that integrates graph structure and text, injecting relational information from in-context examples into the LLMs.
- Extensive experiments on both single task and task transfer settings demonstrate the effectiveness and efficiency of ContextG, surpassing SOTA ICL methods.

2 Related Work

2.1 In context learning

As defined by [9], ICL is a paradigm that allows language models to learn tasks given only a few examples in the form of demonstration. To enhance the effectiveness of ICL, existing works have explored ways of selecting [22, 27, 37, 52, 62], ordering [29, 31], and compression [10, 24, 39] related in-context examples to produce exemplars with rich semantic information for target input. To retain the stability of the ICL method, researchers have made notable efforts in in-context example selection. As for unsupervised methods, KATE [27] captures the semantic similarities between text input and in-context examples, and vote-k [46] combines graphs and confidence scores to sample representative demonstrations. In terms of supervised methods such as EPR [44], it proposes to train a dense retriever using contrastive learning with prediction signals from LM inferencer. Following EPR, CEIL [55] further formulates in-context example selection as a subset selection problem to consider the diversity within the selected examples. On the other hand, even assisted with different selection strategies, ICL methods are restrained by the context length in LLMs[24], failing to efficiently gather as much information as possible. To boost the efficiency, significant efforts like prompt tuning [20], condensing demonstrations [24], regrouping and selection[52], propose methods to capture as rich information as possible from in-context examples while keeping efficient. However,from the perspective of ICL effectiveness, existing methods overlook the inherent relationships between examples. In this paper, we propose to mine the intrinsic graph structure between examples. For efficiency, our method compresses both the graph structure and text into input vectors.

2.2 Graph Enhanced LLMs

Graphs are an essential data structure utilized to represent complex relationships. GNNs are designed to exploit graphs and capable of capturing the underlying structural information and dependencies [11, 18, 50]. With the rapid development of LLMs, researchers are increasingly seeking ways to enhance their capabilities. As a result, there has been a surge in studies at the intersection of graphs and LLMs [13, 15, 23, 36, 51, 63]. This cross-disciplinary exploration spans various fields and aspects, ranging from the design of more generalizable graph models to better leverage graph data [19, 21, 26, 47, 57, 60], to the study of inter-modal relationships in multimodal research [14, 38, 58]. Moreover, existing research demonstrates that graphs can indeed enhance LLMs. Current work primarily focuses on two key areas: one is improving the reasoning capabilities of LLMs [1, 19], and the other is enhancing their collaboration abilities [4, 25]. This paper leverages graph structures as a medium to model the relationships between in-context examples, taking advantage of the property that nodes in a graph are not constrained by order, thereby allowing the LLM to be unaffected by the sequence of examples.

In contrast, our work redefines the role of graph structure within LLMs, using them as a means to construct and represent relationships between examples in the ICL scenario, thereby enhancing the general framework of ICL. Unlike existing methods that rely on graph structures as input or do not utilize graph structures at all, we employ the graph structure as a container to organize the relationships between examples. This approach bridges the gap between *unstructured, text-based reasoning* and *structured problem-solving paradigms.*

3 Method

In this section, we introduce our method, ContextG, that capable of efficiently and effectively choose and encode in-context examples to enhance ICL.

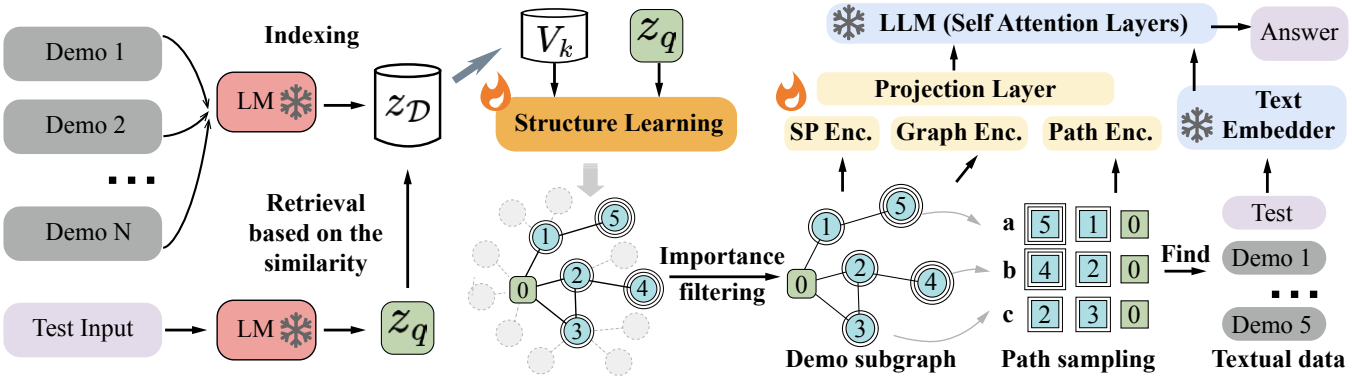

Figure 2: Overall architecture of ContextG. 'SP Enc.' stands for structural positional encoding.

3.1 Preliminaries

Here, we briefly introduce the problem setting of ICL which refers to the ability of LLM that infers tasks from context [3]. Given a demonstration set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i and y_i denote the input and output tokens respectively, and text input x_q, the LLM generates the output y as

$$y \sim \mathcal{P}_{LLM}(y|\,\text{concat}(\mathbf{E}_D, \mathbf{E}_{x_q})), \tag{1}$$

where $\sim$ stands for decoding strategies that convert output of LLM into task labels, $\mathbf{E}_{(\cdot)}$ is the embedding from LLM. The vanilla ICL is randomly select the demonstration set $\mathcal{D}$. However, the performance can be enhanced by enhancing the quality of the retrieved in-context examples $e_i = (x_i, y_i)$ from $\mathcal{D}$. Note that, this paper focuses on the selection, ordering and encoding of in-context examples, rather than on the design of templates or tuning of LLM.

Overview. The pipeline of ContextG is shown in Fig. 2. Given a test example, the model first selects the k most similar examples from the demonstration set. Then, an adaptive structure learning method is employed to construct a graph centered around the test query, capturing the relational structure among the selected examples. Finally, a structured context encoder is utilized to aggregate and encode the graph-structured examples, which are then fed into the LLM for inference.

3.2 Graph Construction

The first stage of ContextG involves extracting the most relevant in-context examples from unstructured textual data based on their relevance to the test query and further uncovering their inherent structural relationships. Inspired by Retrieval-Augmented Generation (RAG), this stage consists of three key steps: first, performing vector-based *indexing* for both the demonstration repository and the test query; second, *retrieving* the most relevant examples from the indexed repository based on the test query; and finally, *constructing a subgraph* centered around the test query, capturing the structural relationships among the retrieved examples.

Demonstrations indexing. As widely recognized [59], the better the ICL examples are chosen based on test query, the better the LLM's performance on a given task. However, in many scenarios, the demonstration set can be extremely large, making direct selection computationally expensive. To address this, we propose vectorizing both the demonstration set and the test query into embeddings, enabling efficient retrieval of the most relevant examples through similarity-based search.

Specifically, given an example $e_i = (x_i, y_i)$ in demonstration set $\mathcal{D}$, we utilize a pre-trained LM, such as SentenceBert [41], to generate the embedding z_i:

$$t_i = \text{Template}(e_i), \quad z_i = \text{LM}(t_i) \in \mathbb{R}^d, \tag{2}$$

where Template is manually designed to integrate the information of x_i and y_i into a coherent textual prompt, and d is the dimension of the embedding. Similarly, test query x_q is also encoded into z_q using the same LM:

$$z_q = \text{LM}(x_q) \in \mathbb{R}^d. \tag{3}$$

Examples retrieval. Carefully choosing in-context examples that are both similar and diverse plays a crucial role in enabling the LLM to grasp the full extent and complexity of the given task. Balancing *similarity to the test query* while maintaining *importance among the selected examples* is a fundamental challenge in in-context learning. To address this, we adopt a two-step approach: first, during the retrieval process, we obtain a broad set of similar training examples, forming an example pool $\mathcal{P}$ as candidate selections. Then, in the subsequent subgraph construction step, we further evaluate and refine the importance of the examples, ensuring that the final in-context example set maintains both relevance and importance. In this retrieval step, our primary focus is on constructing an example pool $\mathcal{P}$ with high similarity to the test query. After the indexing process, where both the examples and the test query are stored as vector representations, we employ k-nearest neighbors (k-NN) retrieval to efficiently identify the most relevant examples. Specifically, the retrieval is formalized as

$$V_k = \text{argtopk}_{i \in \mathcal{D}} \cos(z_q, z_i), \tag{4}$$

where cos is the cosine similarity that we use to calculate the similarity between the embedding of test query x_q and the examples in demonstration set $\mathcal{D}$. The argtopk is the function that collect the top-k elements based on the similarity. Since the next step involves further filtering of the retrieved set V_k, the hyperparameter k should be set larger than m, the final size of the in-context example set S_m, to allow for the following refined selection process.

Subgraph construction. The core step of our method is the subgraph construction, which involves uncovering the intrinsic relationships among the retrieved examples to form a structured graph. This graph structure is then leveraged to compress and represent the examples more effectively, ensuring a more structured and informative input for the LLM.

The subgraph construction step consists of two parts: First, we learn the graph structure by applying structural learning to the example set obtained from Eq. 4 and the target test query. Second, we compute PageRank [35] to retain examples with higher importance.

Specifically, we use an attention-based approach to learn the graph structure. Given the embedding of examples $z_i, i = 1, ...k$ in the set V_k and test query z_q, the graph structure $\mathbf{A}$ can be constructed as

$$E_{i,j} = f_\phi(W \odot z_i, W \odot z_j), \tag{5}$$

$$A_{i,j}(z) = \begin{cases} \sigma(E_{i,j}), & j \in \text{top-k}(E_{i,:}) \\ 0, & j \notin \text{top-k}(E_{i,:}) \end{cases}, \tag{6}$$

where z_i and z_j denote two embedding from the set $V_k \cup z_q$. $\odot$ is the Hadamard operation, W is a learnable parameter vector, f_ϕ denotes the similarity metric such as cosine similarity, σ stands for non-linear activation function like relu.

Pagerank filtering. Furthermore, we filter examples with higher importance. Since the graph structure $\mathbf{A}$ has already been constructed based on the embeddings of examples, structural insights can be directly incorporated when measuring importance of samples. Specifically, structural importance can be understood as selecting nodes that contribute more informative and diverse content. We propose using PageRank to capture each node's importance or information propagation ability within the graph. A node with stronger propagation ability indicates that it can provide more valuable and diverse information. Thus, we compute PageRank scores for every examples in the set V_k and select top M as the most representative diverse examples. Formally, the PageRank is calculated as

$$R = (\mathbf{I}_d - \beta \mathbf{A})^{-1} \left(\frac{1-\beta}{d}\right) \mathbf{1}, \tag{7}$$

$$S_m = \text{argtopm}_{i \in V_k} R, \tag{8}$$

where R is the size of $k \times 1$, $\mathbf{I}_d$ is the identity matrix, $\beta \in (0, 1)$ is the teleport (or restart) probability here, $\mathbf{1}$ is an d-dimensional vector in which all elements are equal to 1. And S_m is the final example set that contains examples both similar to test query and contains most information in V_k. Moreover, since the nodes S_m have been selected, we also modify the extracted subgraph accordingly:

$$S^* = S_m \cup z_q, \tag{9}$$

$$E^* = \begin{cases} 1, & (v_i, v_j \in S^*) \wedge (A_{ij} > 0) \\ 0, & otherwise \end{cases}, \tag{10}$$

$$\mathbf{A}^* = (S^*, E^*). \tag{11}$$

Notably, both the textual data of these m selected examples $e_i, i = 1, ..., m$ and the graph structure formed with the test query will be used in the next step for structural encoding.

3.3 Structure Encoding

In this section, we describe how to jointly encode the learned graph structure and textual information of selected examples. Our goal is to address two key challenges in in-context learning: the long-standing issue of *example ordering* and the *computational overhead* caused by excessively long example sets in LLMs.

To address the example ordering issue, we propose to employ *graph structural encoding strategy*, and prepend the *hierarchical embedding* of node-, path- and subgraph-level.

Structural positional encoding. To eliminate the impact of example ordering, we compute a structural positional encoding for each node based on the learned graph structure. Specifically, we encode the node $v_i \in S^*$ as

$$p_j = ||_{i=1}^{l} \, diag((\mathbf{D}^{-1}\mathbf{A}^*)^i), \tag{12}$$

$$h^n = ||_{j=1}^{m} \, p_j, \tag{13}$$

where $\mathbf{D}$ is the degree matrix of $\mathbf{A}^*$, $diag$ and l are the diagonal elements and the step of the random walk matrix respectively. And $p_i \in \mathbb{R}^l$ is the structural encoding of node v_i, $||$ denote the concatenate operation.

Path encoding. We have obtained the node representations z using the LM in Eq. 2. To bridge the gap between node-level structural positional feature and subgraph features, we propose using path embedding to further capture the relationships within the internal graph structure of the examples. Specifically, we start from the test node v_q and perform BFS and sampling to generate c paths $\mathbf{P}$ of length r. Then, we aggregate the node embedding along each path using a GRU [5] to obtain the path embedding, formulated as

$$h_i^p = \text{GRU}(\{z_j \mid j \in \mathbf{P}_i\}), i = 1, ..., c, \tag{14}$$

$$h^p = ||_{i=1}^{c} \, h_i^p. \tag{15}$$

This part allows ContextG to capture local structural information around node v_q while also preserving connectivity patterns within its nearby neighborhood.

Subgraph encoding. Subgraph encoding is the final step in structural representation and the most crucial for preserving the intrinsic relationships between examples. Specifically, we first use a graph encoder to aggregate the encoded node embeddings $z_i, v_i \in S^*$ and their relationships. In this work, we adopt a standard Graph Attention Network (GAT) [50]. Formally, the graph representation is computed as:

$$h^g = \text{pool}\left(\text{GNN}\left(z_i\right)\right), \quad v_i \in S^*, \tag{16}$$

where pool is the mean pooling operation.

Embedding merging. Although aggregating structure-aware representations h^n, h^p, h^g provides global relational information for the inference LLM, the textual information itself may be compressed or lost during aggregation. To mitigate this, we also integrate the raw textual information e_i of all selected nodes into the LLM. To align the structural representations and the vector space of the LLM, we use a projection layer:

$$h_{\text{all}} = \text{MLP}(h^n || h^p || h^g) \in \mathbb{R}^{d_l}, \tag{17}$$

$$h_{\mathcal{D}} = h_{\text{all}} || \, \text{TextEmbedder}(S^*), \tag{18}$$

where d_l is the hidden size of the LLM, TextEmbedder is the first layer of the pre-trained and frozen LLM. And $h_{\mathcal{D}}$ is the final input from the demonstration set $\mathcal{D}$ into the LLM. A key point is that the text order in Eq. 18 must match the structural positional encoding order in Eq. 13. This ensures that the structural positional encoding of the nodes can effectively eliminate the order-related bias introduced by the node text.

3.4 Optimization and Inference

This section involves generating the answer or label of tasks and training the learnable parameters. We use loss function to train all the parameters mentioned earlier, excluding the parameters of the LLM used in the subsequent steps.

We use the prediction task loss to ensure that the entire model can successfully complete the final task, specifically by improving the LLM's ability to make accurate decisions for downstream tasks through the provided examples. In each training iteration, we select one training sample x_q as the final task and extract k examples V_k from other training samples according to our scheme in above sections, followed by subsequent filtering and encoding operations. Ultimately, the embeddings h formed by these examples are used to instruct the LLM to generate the task output y. Formally, the prediction task loss $\mathcal{L}_{\text{pred}}$ is defined as:

$$\mathcal{L}_{\text{pred}} = \log P_{\text{LLM}}\left(y \mid \left(\mathbf{h}_{\mathcal{D}} || \mathbf{x}_q\right)\right). \tag{19}$$

Table 1: Comparison with other classical ICL paradigms. 'Prompt-T' stands for prompt tuning. 'Hierarchical' means hierarchical encoding for examples.

	Selection	Ordering	Structure	Vectorized	Hierarchical
Prompt-T				✓	
EPR	✓				
CEIL	✓	✓			
MEND		No need		✓	
ContextG	✓	No need	✓	✓	✓

Paradigm comparison. Tab. 1 compares existing ICL paradigms with our model ContextG, which integrates multiple heuristic criteria to enhance ICL. For *demonstration selection*, MEND [24] and Prompt-Tuning [20] focus on other aspects and can be combined with different selection methods. EPR [44] emphasizes relevance but neglects diversity. CEIL [55] improves on the DPPs to balance similarity and diversity, but it still evaluates relevance based on pairwise similarity, ignoring structural relationships. In contrast, ContextG employs graph-based metrics like PageRank to score example importance, leading to higher-quality selections. For *example ordering*, Prompt-Tuning and EPR do not explicitly mitigate ordering biases. CEIL attempts to address this issue but still treats examples as a linear sequence, limiting its ability to eliminate order effects. MEND utilized distilled vectors to reduce ordering sensitivity, but may still be indirectly affected. ContextG overcomes this issue through graph and permutation invariant GNNs. For *vectorized representations*, Prompt-Tuning employs soft prompt embeddings, while MEND and ContextG utilize vectorized inputs to enhance efficiency and reduce the computational cost of LLM inference. The key innovation of ContextG lies in its graph-based framework, which integrates *structural relationships* and *hierarchical encoding*. By leveraging graph structures, ContextG improves selection quality, ordering robustness, and computational efficiency, potentially offering new insights for ICL.

4 Experiments

In this section, we answer the following four questions through experiments to validate the effectiveness of ContextG[1]:

- **RQ1.** Does our in-context information modeling method truly enhance the LLM's ability to utilize and comprehend contextual information?
- **RQ2.** Can ContextG maintain performance improvements even when applied to tasks with distribution shifts or transfer scenarios?
- **RQ3.** Does each component of ContextG contribute to enhancing ICL performance?
- **RQ4.** Is ContextG sufficiently robust to be effectively applied across various downstream LLM architectures?
- **RQ5.** Is ContextG computationally efficient and cost-effective for practical deployment?

4.1 Effectiveness

To answer the **RQ1**, we compare ContextG with other baselines on different datasets and inference LLMs.

Settings. We evaluate ContextG on four diverse datasets across different tasks, summarized in Tab. 2. SST5 [45] assesses sentiment analysis, HellaSwag [61] evaluates commonsense reasoning, MNLI [53] covers natural language inference, and MRPC [7] focuses on paraphrase detection. We compare ContextG against various baselines that improve demonstration selection, ordering and distillation, including both learning-free and learning-based approaches. Detailed descriptions of these datasets and baselines are provided in the App. A.

For LLM inference, we employ GPT-2 Large (774M) [40] and GPT-2 XL (1.5B) [40]. We evaluate on the validation splits of the datasets, as some test splits do not have ground-truth labels. The number of in-context examples is set to 16 for both training and evaluation across all datasets and models, except in zero-shot settings, where no demonstrations are provided. This constraint ensures that the input remains within the context length limitations of the LLMs (e.g., 1024 tokens for both GPT-2 Large and GPT-2 XL). For PromptTuning, Vanilla ICL, and MEND, which lack an inherent demonstration selection mechanism, we employ a random selection strategy. This method selects in-context examples from the demonstration pool without repetition and explicitly removes the test example if it is selected during the process. We frame all the classification problem as a multiple-choice task [3], where the model processes the context concatenated with each candidate answer and selects the option with the highest likelihood. Performance is measured using accuracy (Acc.), computed by comparing model predictions against ground truth labels.

Results. Overall, ContextG outperforms all baselines across datasets and tasks and achieves the highest accuracy. On average, ContextG improves performance by 34% over Vanilla ICL and 7.6%

[1]Our codes are available at https://anonymous.4open.science/r/GinContext/.

Table 2: Performance comparison against baselines on four datasets using two inference models. The results are reported in accuracy, with the best-performing method highlighted in bold for each dataset.

gpt2-large	SST-5	HellaSwag	MRPC	MNLI	gpt2-xl	SST-5	HellaSwag	MRPC	MNLI
Zero-shot	24.57	43.28	68.14	35.43	Zero-shot	25.84	48.89	60.54	36.30
PromptTuning	30.41	43.09	63.73	35.89	PromptTuning	27.24	49.14	61.03	38.64
Vanilla ICL	24.89	43.05	67.65	42.12	Vanilla ICL	29.50	48.71	68.14	37.29
MEND	44.80	42.88	71.54	46.51	MEND	47.68	48.49	66.90	43.99
CEIL	43.72	41.05	62.25	45.38	CEIL	46.58	48.78	68.53	64.85
MEND+CEIL	45.21	43.76	70.43	52.73	MEND+CEIL	46.99	47.81	71.27	58.95
ContextG	**47.84**	**46.30**	**72.17**	**62.90**	ContextG	**48.69**	**51.32**	**73.04**	**65.81**

Table 3: Performance when transferred across tasks. The columns represent task transfers. Results are averaged across tasks, with standard deviations shown where applicable. The best-performing models are highlighted in bold.

	Models	non-Class-> Class	non-QA->QA	HR->LR	non-Para->Para	average
gpt2-large	Zero-shot	34.36	44.58	34.77	34.12	36.96
	PromptTuning	38.78	38.71	40.68	34.23	38.10
	Vanilla ICL	41.30±2.15	45.81±1.34	41.26±2.26	38.93±1.15	41.83
	MEND	43.38±1.62	44.29±0.86	40.92±1.80	42.54±0.44	42.78
	CEIL	**44.62±1.99**	44.41±1.12	42.01±2.01	41.00±1.01	43.01
	MEND+CEIL	43.99±1.81	44.19±1.23	41.52±1.17	40.89±0.89	42.65
	ContextG	44.55±1.75	**46.31±1.02**	**43.19±1.64**	**42.79±1.14**	**44.21**
gpt2-xl	Zero-shot	32.08	46.09	33.95	33.61	36.43
	PromptTuning	38.78	41.45	40.83	35.52	39.15
	Vanilla ICL	40.63±2.53	48.32±0.88	42.27±2.08	37.53±1.04	42.19
	MEND	43.37±1.50	45.95±0.66	42.16±1.81	42.53±1.20	43.50
	CEIL	44.97±1.77	46.88±1.02	42.59±1.08	41.11±1.12	43.89
	MEND+CEIL	44.93±1.09	45.18±0.99	41.16±1.10	41.46±1.31	43.18
	ContextG	**45.92±1.06**	**49.21±1.24**	**44.27±1.16**	**43.82±1.55**	**45.81**

over the strongest baseline MEND+CEIL, demonstrating its ability to leverage in-context examples effectively.

On MNLI, a challenging natural language inference task, ContextG outperforms MEND+CEIL by 15% and Vanilla ICL by 46%. This improvement is substantially larger than the typical 5% gain over MEND+CEIL on other tasks. One reason may be MNLI's large dataset, which provides a richer example pool, making demonstration selection more important. By utilizing graph structures and selecting high-connectivity nodes, ContextG improves contextual understanding. Additionally, MNLI requires cross-domain reasoning, where structural relationships between examples is critical. ContextG captures these relationships effectively through hierarchical encoding approach.

The performance gap between GPT-2 Large and GPT-2 XL as inference LLM further highlights ContextG 's robustness. While GPT-2 XL (1.5B) generally outperforms GPT-2 Large (774M) across all methods, ContextG maintains consistent relative improvements over baselines. For example, on HellaSwag, ContextG surpasses MEND+CEIL by 5.8% on GPT-2 Large and 5.4% on GPT-2 XL, demonstrating its scalability across different model sizes.

4.2 Transfer Between Tasks

To answer the **RQ2**, we evaluate ContextG 's generalization ability by transferring it across different types of tasks.

Settings. Following Sec. 4.1, we employ GPT-2 Large as the inference LLM and utilize 16 demonstrations. For classification tasks, we evaluate performance using accuracy, while for non-classification tasks, we adopt the Macro-F1 score, consistent with MEND. The Macro-F1 score averages the F1 scores across all classes, balancing precision and recall. We evaluate our model on the metaICL [34] dataset. Detailed descriptions of this dataset are provided in the App. A. The trainable baseline learning model is fine-tuned on the meta-train partition and evaluated on the meta-test partition. The results are presented in the Tab. 3, showing average scores along with standard deviations. The standard deviation values indicate the variability introduced by different sets of demonstrations. Note that for PromptTuning and Zero-shot, no demonstration retrieval is required, and therefore, their standard deviation is zero.

Results. The results in Tab. 3 demonstrate the generalization ability of ContextG when transferred across different tasks. Overall, ContextG consistently outperforms the baselines across all task types for both GPT-2 Large and GPT-2 XL.

We observe that the performance of MEND+CEIL is lower than that of MEND and CEIL individually. This is likely because each model excels in different tasks: MEND performs best on non-Para to Para tasks (13.3% better than Vanilla ICL), while CEIL excels on non-Class to Class tasks (10.68% better than Vanilla ICL). The

combination results in a trade-off, as their structures are not inherently compatible. In contrast, ContextG integrates demonstration selection and structural encoding within a unified graph-based framework, ensuring seamless compatibility and superior performance across tasks.

The experimental results reveal negative transfer issue, particularly for non-QA to QA and HR to LR tasks. On these tasks, the performance of our learning-based baseline models is either similar to or even worse than Vanilla ICL. However, ContextG consistently outperforms Vanilla ICL, demonstrating strong generalization ability with gains of 1.84% and 4.73% on each task. This advantage likely arises from the fact that existing baselines focus primarily on the relevance between demonstrations and the test query, which varies across tasks. In contrast, our model not only considers this relevance but also the interrelationships between demonstrations, which may remain more invariant and facilitate better task transfer.

Table 4: Ablation study on ContextG. '-' means remove certain component. 'Enc.' stands for encoding, 'Subg.' and 'SP' stands for subgraph and structural positional encoding.

gpt2-large	SST-5	HellaSwag	MRPC	MNLI
model	47.84	46.30	72.17	62.90
- Graph	39.76	43.62	67.88	43.05
- PR filter	47.61	46.08	71.22	60.13
- All encoding	45.93	44.19	69.03	49.18
- Subg encoding	43.15	44.78	70.35	52.50
- Path encoding	43.19	44.99	71.29	59.88
- SP encoding	44.17	45.11	71.71	60.55

4.3 Ablation Study

To answer the **RQ3**, we break down ContextG into its individual components to evaluate the contribution of each module.

Settings. We conduct experiments using the diverse task datasets from Sec. 4.1, keeping the settings and evaluation metrics unchanged. For inference, we use the GPT-2 Large. Our model consists of two major parts: The first part is further divided into two levels. At the first level, we completely remove the graph structure (- Graph), reducing our method to a simple similarity-based example retrieval approach. At the second level, we remove the PageRank-based node importance filtering (- PR filter). To ensure a fair comparison, we adjust the number of retrieved examples to match the number of examples m. The second part is also divided into two levels. At the first level, we remove all encoding mechanisms (- All Enc.), meaning the extracted graph structure is only used for node importance filtering. At the second level, we further conduct three separate ablations by individually removing the modeling of subgraph (- Subg Enc.), path (- Path Enc.), and structural positional information (- SP Enc.), respectively. Moreover, we also conduct experiments that vary in number of demonstrations on SST-5. This experiment demonstrates the model's robustness in handling varying In-context window requirements.

Results. The ablation study results in Tab. 4 demonstrate the contribution of each component in ContextG to the overall performance. Removing the graph structure leads to a significant drop, indicating that ContextG enhances ICL by leveraging intrinsic graph structure. In particularly, MNLI experiences the largest drop (31.5%), likely due to its reliance on structural relationships for complex cross-domain reasoning.

For graph extraction and processing, removing the PageRank-based node importance filtering causes a minor performance drop on SST-5 (0.48%) and MRPC (1.32%) but a larger decline on HellaSwag (4.75%) and MNLI (4.40%). The PageRank-based filter enhances the structural significance of selected demonstrations by assessing their node propagation ability. This structural information is particularly crucial for MNLI and HellaSwag, as both tasks emphasize reasoning and relationships between demonstrations. In contrast, SST-5 and MRPC are simpler, focusing on individual examples. These results suggest that we can selectively apply ContextG components depending on the complexity of the downstream task and its structural dependencies.

For the structural encoding part, removing all structural encodings causes an 8.67% performance drop but still outperforms removing the graph structure, showing the value of structural information in selecting demonstrations. Among individual encoding components, subgraph encoding has the most impact. This funding highlights the importance of global structure among demonstrations ignored by existing ICL methods. Specially, for SST-5, all three encoding approaches are crucial, as removing any one causes an 8–9% drop, which is greater than the 3.99% drop from removing all structural encoding. It is likely due to the SST-5's strong dependence on example ordering and structural hierarchy. This result suggests that the encoding components of ContextG complement and interact with each other, the removal of any one may impact the others.

Fig. 3 illustrates the impact of the number of demonstrations on model performance. ContextG shows a sharp boost at low counts ($1 \rightarrow 2$), while CEIL improves significantly at higher counts ($8 \rightarrow 16$). This may be because ContextG provides LLMs with both structural and textual encoding, and its PageRank-based selection prioritizes nodes with the highest information propagation ability. This finding suggests that ContextG is well-suited for few-shot learning scenarios. Additionally, ContextG's steady improvement, rather than fluctuating performance, suggests its superior ability to fully utilize the provided demonstrations.

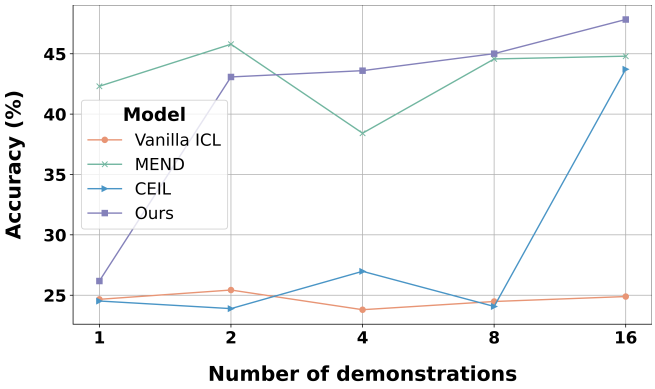

Figure 3: Varying the number of demonstrations m.

Table 5: Performance when using different LLMs for training and inference. We use GPT-2 Large for training and others for inference. 'GPT2l' is GPT2 Large (774M), 'GPT-N' is GPT-Neo (2.7B), 'LLaMA' is LLaMA-7B, 'Qwen' is Qwen2.5 (7B).

SST-5	GPT2l	GPT-N	LLaMA	Qwen
MEND	71.54	71.99	73.11	74.26
CEIL	62.25	67.89	66.42	70.34
MEND+CEIL	70.43	70.88	71.17	73.01
ContextG	72.17	72.25	73.95	75.64

4.4 Using Different LLMs

To answer the **RQ4**, we use different LLMs for model training and inference to demonstrate the robustness of our approach.

Settings. For this experiment, all models were initially fine-tuned on GPT-2 Large [40], with the first column in Tab. 5 serving as a reference where the LLM architecture remains consistent. To assess the transferability of ContextG, we validate its performance on LLMs from the same family but with different parameter sizes GPT-Neo [2] and models from different families LLaMA [49], Qwen2.5 [48]. We report Accuracy metrics on the MRPC [7] dataset. The baseline models are outlined in Tab. A.2. It should be noted that Zero-shot and Vanilla ICL are training-free approaches and thus do not encounter discrepancies between the LLM used for fine-tuning and inference. Therefore, these two approaches are not suitable for the task transfer experiment, and we use the other baselines, excluding these two.

Results. As shown in Tab. 5, our results surprisingly demonstrate that all methods outperform the original LLM inference backbone, GPT-2 Large, on larger-scale LLM inferencers. This suggests that ICL methods exhibit strong generalizability across different downstream inferencers, with potential performance improvements driven by the use of more powerful LLM inferencers. Notably, ContextG consistently delivers the best performance across all downstream LLM backbones, highlighting the robustness and stability of our approach. Through comparison, we find that the stronger the LLM, the better the effectiveness of our method, which also proves that the information extracted by our method can be understood by a more advanced inference LLM.

4.5 Efficiency

To answer the **RQ5**, we compare the training and inference time with the baselines to demonstrate the efficiency of our model.

Settings. The computational efficiency of ICL methods during inference is a crucial consideration for their deployment in real-world applications. Consequently, inference efficiency remains a central focus of our study. We distinguish two primary steps in the inference process of ContextG: (1) the generation of compact vector embeddings from in-context examples, and (2) the subsequent inference performed by the LLM using these distilled vectors in conjunction with the test query. Prompt Tuning, Zero-shot, and Vanilla ICL directly incorporate selected examples as prompts into the LLM while MEND and ContextG input distilled vector embeddings into the downstream LLM. Notably, Zero-shot serves as a baseline for inference time since it does not require any demonstration selection or processing. To evaluate efficiency, we measure the processing time from demonstration selection to prediction output. All experiments are conducted on a single NVIDIA A800 GPU with a batch size of 1 to ensure fair comparisons. For memory consumption, we report the peak GPU usage. Each inference is repeated 10 times, and we report the average result. Other settings remain consistent with Sec. 4.1.

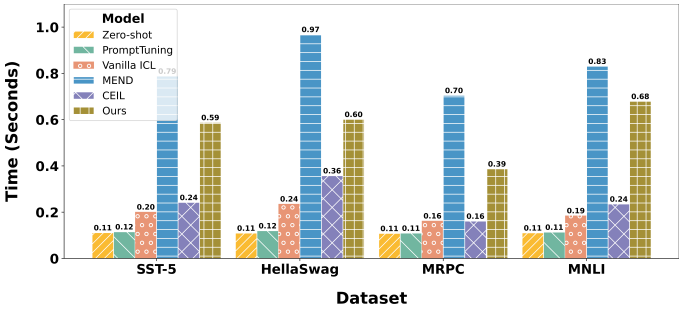

Figure 4: Inference time (seconds) comparison with baselins.

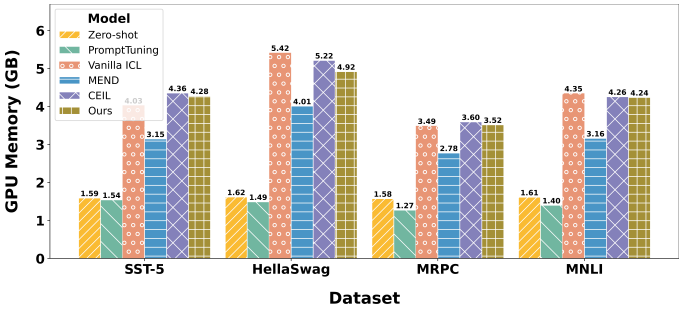

Figure 5: GPU Peak Memory (GB) comparison with baselins.

Results. As illustrated in Fig. 4 and Fig. 5, ContextG exhibits lower peak GPU memory usage than CEIL across all datasets, with notable improvements over Vanilla ICL on complex datasets like HellaSwag (9.22%) and MNLI (2.53%), making it suitable for resource-constrained environments. While both MEND and ContextG require longer inference times due to additional vectorization step, ContextG is approximately 30% faster than MEND. Unlike MEND, which only performs demonstration distillation, ContextG combines example selection and encoding, highlighting its efficiency advantage.

5 Conclusion

In this paper, we propose a novel approach, ContextG, for enhancing in-context learning by leveraging the inherent structure between examples and test queries. Our framework introduces graph structures to capture the relationships between examples, improving both the effectiveness and efficiency of the ICL process. Through extensive experiments, we demonstrate that our model outperforms SOTA baselines and achieves the competitive performance in task transfer settings. By explicitly modeling example relationships, we provide deeper insights into the potential of graph structures in boosting reasoning capabilities of ICL.

References

[1] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17682–17690.

[2] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. https://api.semanticscholar.org/CorpusID:245758737

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Zheng Chen, Ziyan Jiang, Fan Yang, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Aram Galstyan. 2023. Graph Meets LLM: A Novel Approach to Collaborative Filtering for Robust Conversational Understanding. *arXiv preprint arXiv:2305.14449* (2023).

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs.CL] https://arxiv.org/abs/1406.1078

[6] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559* (2022).

[7] Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. COLING, Geneva, Switzerland, 350–356. https://aclanthology.org/C04-1051/

[8] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[9] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-context Learning. arXiv:2301.00234 [cs.CL] https://arxiv.org/abs/2301.00234

[10] Jun Gao, Ziqiang Cao, and Wenjie Li. 2024. Unifying demonstration selection and compression for in-context learning. *arXiv preprint arXiv:2405.17062* (2024).

[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*, Vol. 30. 1025–1035.

[12] Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. 2022. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713* (2022).

[13] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630* (2024).

[14] Yufei He, Yuan Sui, Xiaoxin He, Yue Liu, Yifei Sun, and Bryan Hooi. [n. d.]. UniGraph2: Learning a Unified Embedding Space to Bind Multimodal Graphs. In *THE WEB CONFERENCE 2025*.

[15] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[16] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing Format Boundaries With a Single QA System. In *Findings of EMNLP*.

[17] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]

[19] Bin Lei, Chunhua Liao, Caiwen Ding, et al. 2023. Boosting logical reasoning in large language models through a new framework: The graph of thought. *arXiv preprint arXiv:2308.08614* (2023).

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3045–3059. doi:10.18653/v1/2021.emnlp-main.243

[21] Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. 2024. Graphadapter: Tuning vision-language models with dual knowledge graph. *Advances in Neural Information Processing Systems* 36 (2024).

[22] Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified Demonstration Retriever for In-Context Learning. arXiv:2305.04320 [cs.CL]

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

[24] Yichuan Li, Xiyao Ma, Sixing Lu, Kyumin Lee, Xiaohu Liu, and Chenlei Guo. 2024. MEND: Meta dEmonstratioN Distillation for Efficient and Effective In-Context Learning. arXiv:2403.06914 [cs.CL] https://arxiv.org/abs/2403.06914

[25] Yihao Li, Ru Zhang, and Jianyi Liu. 2024. An enhanced prompt-based LLM reasoning scheme via knowledge graph-integrated collaboration. In *International Conference on Artificial Neural Networks*. Springer, 251–265.

[26] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2023. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149* (2023).

[27] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[28] Sheng Liu, Haotian Ye, Lei Xing, and James Zou. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668* (2023).

[29] Yinpeng Liu, Jiawei Liu, Xiang Shi, Qikai Cheng, Yong Huang, and Wei Lu. 2024. Let's Learn Step by Step: Enhancing In-Context Learning Ability with Curriculum Learning. *arXiv preprint arXiv:2402.10738* (2024).

[30] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG] https://arxiv.org/abs/1711.05101

[31] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. arXiv:2104.08786 [cs.CL] https://arxiv.org/abs/2104.08786

[32] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061* (2023).

[33] Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624* (2024).

[34] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to Learn In Context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 2791–2809. doi:10.18653/v1/2022.naacl-main.201

[35] Lawrence Page. 1998. The pagerank citation ranking: Bringing order to the web. Technical report. *Stanford Digital Library Technologies Project, 1998* (1998).

[36] Shirui Pan, Yizhen Zheng, and Yixin Liu. 2024. Integrating graphs with large language models: Methods and prospects. *IEEE Intelligent Systems* 39, 1 (2024), 64–68.

[37] Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting Demonstration Selection Strategies in In-Context Learning. arXiv:2401.12087 [cs.CL] https://arxiv.org/abs/2401.12087

[38] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862* (2024).

[39] Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. 2023. Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning*. PMLR, 27854–27875.

[40] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[41] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).

[42] Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. 2023. Neural graph reasoning: Complex logical query answering meets graph databases. *arXiv preprint arXiv:2303.14617* (2023).

[43] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[44] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning To Retrieve Prompts for In-Context Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 2655–2671. doi:10.18653/v1/2022.naacl-main.191

[45] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (Eds.). Association for Computational Linguistics, Seattle, Washington, USA, 1631–1642. https://aclanthology.org/D13-1170/

[46] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Selective Annotation Makes Language Models Better Few-Shot Learners. *ArXiv*

(2022).
[47] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 491–500.
[48] Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. https://qwenlm.github.io/blog/qwen2.5/
[49] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL] https://arxiv.org/abs/2302.13971
[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[51] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems* 36 (2024).
[52] Song Wang, Zihan Chen, Chengshuai Shi, Cong Shen, and Jundong Li. [n. d.]. Mixture of Demonstrations for In-Context Learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
[53] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 1112–1122. doi:10.18653/v1/N18-1101
[54] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080* (2021).
[55] Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional Exemplars for In-context Learning. arXiv:2302.05698 [cs.CL] https://arxiv.org/abs/2302.05698
[56] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A Few-shot Learning Challenge for Cross-task Generalization in NLP. In *EMNLP*.
[57] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, Yongfeng Zhang, et al. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134* 4, 5 (2023), 7.
[58] Minji Yoon, Jing Yu Koh, Bryan Hooi, and Ruslan Salakhutdinov. 2023. Multimodal graph learning for generative tasks. *arXiv preprint arXiv:2310.07478* (2023).
[59] Guoxin Yu, Lemao Liu, Mo Yu, Yue Yu, and Xiang Ao. 2024. Rethinking the Evaluation of In-Context Learning for LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 14068–14082. doi:10.18653/v1/2024.emnlp-main.779
[60] Junchi Yu, Ran He, and Rex Ying. 2023. Thought propagation: An analogical approach to complex reasoning with large language models. *arXiv preprint arXiv:2310.03965* (2023).
[61] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 4791–4800. doi:10.18653/v1/P19-1472
[62] Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active Example Selection for In-Context Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 9134–9148. doi:10.18653/v1/2022.emnlp-main.622
[63] Ziwei Zhang, Haoyang Li, Zeyang Zhang, Yijian Qin, Xin Wang, and Wenwu Zhu. 2023. Graph meets llms: Towards large graph models. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

A Experiment Details

A.1 Datasets

The datasets we use in the Sec. 4.1 are shown in Tab. 6

SST-5 [45] is a sentiment analysis dataset containing movie reviews annotated with five sentiment labels "very positive", "positive' 'neutral", "negative", and "very negative". It includes phrase-level annotations, enabling fine-grained analysis of sentiment compositionality.

MNLI [53] is a natural language inference dataset with sentence pairs labeled as entailment, contradiction, or neutral. It spans multiple genres, making it a robust benchmark for cross-domain reasoning.

MRPC [7] is a paraphrase identification dataset consisting of sentence pairs labeled as semantically equivalent or not. It is widely used for evaluating text similarity and paraphrase detection models.

Hellaswag [61] is a commonsense reasoning dataset featuring multiple-choice questions about plausible sentence continuations. Each question provides a context and four candidate answers: one correct continuation and three adversarially generated incorrect options designed to deceive machines.

MetaICL [34] is a benchmark specifically designed for in-context learning tasks. It is derived from the CrossFit [56] dataset and UnifiedQA [16] dataset. MetaICL comprises 142 diverse NLP tasks, including text classification, question answering, and natural language inference, divided into non-overlapping meta-train and meta-test sets. The meta-train set includes 61 tasks with extensive examples, while the meta-test set contains 52 target tasks across seven splits (see Tab. 7 for details). This design enables rigorous evaluation of few-shot generalization across diverse and challenging task transitions.

A.2 Baselines

We evaluate ContextG on four diverse datasets across different tasks, summarized in Tab. 2. SST5 [45] assesses sentiment analysis, Hellaswag [61] evaluates commonsense reasoning, MNLI [53] covers natural language inference, and MRPC [7] focuses on paraphrase detection.

- **PromptTuning**: A technique that fine-tunes task-specific prompts to adapt pre-trained language models to downstream tasks, enhancing performance without modifying the model parameters [20].
- **Zero-shot**: A method where the model performs inference without any in-context examples, relying solely on its pre-trained knowledge to make predictions.
- **Vanilla ICL**: The standard in-context learning approach, where the model is provided with a few task-specific examples directly in the input context to guide its predictions.
- **MEND**: A demonstration distillation method that condenses lengthy demonstrations into compact vectors, improving in-context learning efficiency through a two-stage training process involving knowledge distillation [24].
- **CEIL**: A learning-based retriever for demonstration selection that uses Determinant Point Processes (DPPs) to optimize the diversity and relevance of the examples [55].

Dataset	Task	#Train	#Validation	Prompt	Options
SST-5	Sentiment Analysis	8,534	1,101	{input} It is {output}	["very negative", "negative", "neutral", "positive", "very positive"]
MRPC	Paraphrase Detection	3,668	408	{input1} Can we say "{input2}"? {output}	["No", "Yes"]
MNLI	Natural Language Inference	392,568	19,647	{input1} Can we say "{input2}"? {output}	["Yes", "Maybe", "No"]
HellaSwag	Commonsense Reasoning	52,611	20,006	{input}, what happens next? {output}	Null

Table 6: Summary of datasets used in our experiments with corresponding prompts.

meta-train			meta-test		
Setting	# task	Avg. Len.	**Setting**	# task	Avg. Len.
Class	43	44.54	Class	20	56.21
non-Class	37	91.45			
QA	37	91.58	QA	22	57.84
non-QA	33	72.50			
non-NLI	55	54.51	NLI	8	61.61
HR	61	82.44	LR	26	35.31
non-Para	59	55.97	Para	4	54.06

Table 7: Seven partitions of MetaICL.

- **CEIL+MEND**:A hybrid model that integrates the demonstration selection capabilities of CEIL with the distillation efficiency of MEND. This is achieved by first using CEIL to select relevant demonstrations from the target datasets, and then feeding these demonstrations into MEND, replacing its inherent random selection method.

All learning-based baselines are trained on the respective training splits of each dataset. For **MEND** and **PromptTuning**, we utilize the AdamW [30] optimizer with a learning rate of 5e-5. Training is conducted for 10,000 steps with a batch size of 8 on a single NVIDIA A800 GPU. For **CEIL**, we use the Adam optimizer [17] with batch size of 64 and learning rate of 1e-5, training for 30 epochs on an NVIDIA A800 GPU with early stopping patience set to 5.

B Limitation and Future Work

Our method retains the same template for demonstrations as other approaches. Future work could leverage the extracted graph structure to design different templates, further enhancing ICL performance.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009