PiVe: Prompting with Iterative Verification Improving Graph-based Generative Capability of LLMs

Jiuzhou Han[♮] **Nigel Collier**[♯] **Wray Buntine**[♭] **Ehsan Shareghi**[♮]
[♮] Department of Data Science & AI, Monash University
[♭] College of Engineering and Computer Science, VinUniversity
[♯] Language Technology Lab, University of Cambridge
jiuzhou.han@monash.edu

Abstract

Large language models (LLMs) have shown great abilities of solving various natural language tasks in different domains. Due to the training objective of LLMs and their pre-training data, LLMs are not very well equipped for tasks involving structured data generation. We propose a framework, Prompting with Iterative Verification (PiVe), to improve graph-based generative capability of LLMs. We show how a small language model could be trained to act as a verifier module for the output of an LLM (i.e., ChatGPT, GPT-4), and to iteratively improve its performance via fine-grained corrective instructions. We also show how the verifier module could apply iterative corrections offline for a more cost-effective solution to the text-to-graph generation task. Experiments on three graph-based datasets show consistent improvement gained via PiVe. Additionally, we create GenWiki-HIQ and highlight that the verifier module can be used as a data augmentation tool to help improve the quality of automatically generated parallel text-graph datasets.[1]

1 Introduction

Large language models (LLMs) like ChatGPT and GPT-4 (OpenAI, 2023) have been quite successful in solving different generative and reasoning tasks. The combination of their abilities in leveraging in-context learning as well as instruction following have unlocked new state-of-the-art results across the natural language processing (NLP) field. The existing LLMs are mostly pre-trained on a huge volume of unstructured data from the internet including books, articles, webtexts, repositories, Wikipedia, etc. Training on unstructured data naturally leads to relatively poor performance when dealing with tasks that demand organizing text into structured machine-readable format.

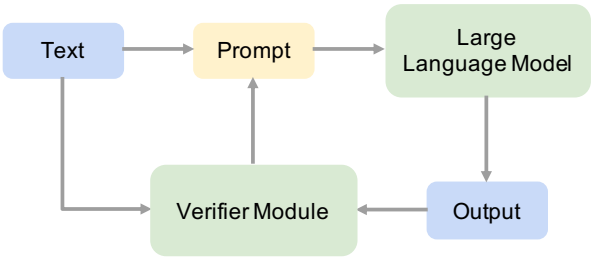

Figure 1: Framework of PiVe.

A semantic graph, as a form of graph-structured data, stores information in a machine-accessible way (van Harmelen et al., 2008). Generating a semantic graph from text is known as text-to-graph (T2G) generation and is previously attempted mostly by fine-tuning small language models (Xu et al., 2022; Guo et al., 2020). However, generating graph-structured data remains a challenge for LLMs even in the presence of reasonable number of few-shot examples. In fact, regardless of the number of few-shot examples or prompting style the outputs from LLMs (e.g., GPT-3.5) still contain errors and require correction (§4.4).

In this paper, we focus on how to improve the graph-based generative capability of LLMs. To this end, we propose the Prompting through Iterative Verification (PiVe) framework shown in Figure 1. Specially, PiVe involves leveraging an external verifier module (i.e., a much smaller LM) and incorporating the feedback from verifier module into the prompt. PiVe iteratively utilises the verifier module and refines the prompts, via corrective instructions, before sending them back into the LLM, leading to substantially improved quality of the generated semantic graphs.

In particular, to train the verifier modules, we start from a seed dataset of text and graph (`T`,`G`) pairs, and construct an arbitrarily large graph-perturbation dataset via a simple procedure which takes any graph `G` from the seed set and perturbs it arbitrarily on its entities (`E`), relations (`R`), or

[1] Our code and data are available at https://github.com/Jiuzhouh/PiVe.

Findings of the Association for Computational Linguistics: ACL 2024, pages 6702–6718
August 11-16, 2024

triples (Tr). The text and perturbed graph ($\bar{G}$), along with a corrective description to invert the applied perturbation (IP) form a verification dataset of (T,$\bar{G}$,IP) triples which serve as the training data for self-supervised learning of our verifier module. The verification dataset could be as large as desired (i.e., for any seed dataset D, containing graphs of |E| entities, |R| relations, |Tr| triples, it could produce $\mathcal{O}(|D| \times |E| \times |R| \times |Tr|)$ perturbations only by *deleting*.[2] We then devise fine-tuning and instruction-tuning to train domain-specific and unified verifiers, respectively.

During the T2G generation via the LLM (e.g., in the zero-shot setting "*Transform the text into a semantic graph: Text: ... Graph:*"), the verifier takes the text T, the output graph from the LLM, and sends a corrective signal to the LLM (e.g., "*Transform the text into a semantic graph and add the given triples to the generated semantic graph: Text: ... Triples: ... Graph:*"). This process continues till the verifier module verifies the output as *correct* and terminates. We refer to this as *Iterative Prompting*. Additionally, there is another (more cost effective) mode to the verifier module, which starts by calling the LLM once at the start to get an initial graph, and then the rest of the corrective steps are all applied step-by-step and iteratively through the verifier offline. We refer to this as *Iterative Offline Correction*.

Our extensive experiment results on three graph-based datasets demonstrate the effectiveness of the proposed PiVe framework in *consistently* improving the quality of the LLM output via providing iterative corrective guidance by an average of 26% across 3 datasets. We also create GenWiki-HIQ, a high-quality text-graph dataset and show how verifier module could be leveraged as a data augmentation technique to improve the quality of automatically constructed text-graph datasets.

2 Basic Definitions

A semantic graph is a network that represents semantic relations between entities. Each semantic graph has its corresponding verbalisation, and can have different textual representations. A set of triples (i.e., [subject, predicate, object]) represents a semantic graph. Given a text, the task of text-to-graph generation is to query an LLM to generate a semantic graph of the text. The semantic graph should cover the information in the text as much as possible.

To prompt the LLM, we use few-shot by showing an example of T2G in the prompt to specify the expected format of the semantic graph (i.e., set of triples). We report experiments under various number of shots (§4.6). The basic form of instruction we use in the prompt is "*Transform the text into a semantic graph.*" followed by a text and a semantic graph pair as a demonstration. We also show results under different prompting strategies (§4.7). Different demonstrations are used for different datasets to adapt to the style of different datasets. We show the used demonstrations in Appendix G.

3 The PiVe Framework

We first explain our training protocol for the verifier module (§3.1), and then present our framework of iterative verification prompting (§3.2).

3.1 Verifier Module

The quality of the generated semantic graph from LLM prompting could be quite poor. For instance the LLM often misses triples in the generated graph. In other words, some semantic relations between entities in the text are difficult to be captured for LLMs when they are generating a semantic graph. To detect the missing or incorrect parts of the generated semantic graph, we design a verifier module.

The verifier module is trained on a small pre-trained LM (§4.2). A typical graph-based dataset contains parallel text and semantic graph (T,G) pairs. For different graph-based datasets, we use their corresponding training data for the seed dataset to create data for the verifier module. In particular, for each text-graph pair in the original dataset, we create one correct instance and one perturbed instance. We concatenate the text with graph using a separator token <S> and the target is to generate a specific output, denoted as IP, during training. For correct instances, the IP is simply the word "Correct". For perturbed instances, we have two methods to create them:

- Random method: if the graph contains more than one triple, we randomly omit one triple from it and concatenate the text with perturbed graph using a separated token <S>. The target is to generate the missing part (e.g., triple Tr).
- Heuristic method: Based on the observation that LLMs tend to miss the triples whose subject and object are not in the text, in addition

[2] We also try other perturbation methods and show the results in Appendix C.

to randomly omitting one triple from the graph we also omit the triple from the graph if subject and object of it are not in the text.

The output to generate for perturbed examples is the missing triple, `Tr`. By utilising these two methods, we can create an arbitrarily large verification dataset to train a verifier module which will be used at inference time during prompting the LLM.

3.2 Iterative Prompting

During the LLM prompting, the generated semantic graphs from LLMs is fed into the verifier module and the outputs from the verifier module is collected. If verifier generated "Correct" in its output, it means we do not need to make changes to the generated graph. Otherwise, the generated output from the verifier is added to the original prompt to create a new prompt. The new prompt is then used to query the LLM again. We repeat the whole process iteratively. The iteration process will stop when no missing triple is predicted or a maximum number of iterations is reached.

New Prompt Design As with the prompt used in the first iteration, we still provide an example in the new prompt for the subsequent iterations. The new prompt is "*Transform the text into a semantic graph and also add the given triples to the generated semantic graph.*" In addition to the text, we also include some triples predicted by the verifier module which LLMs are likely to miss. This explicitly instructs the LLM to generate semantic graph and include the given triples. The given triple set contains the predicted missing triples from each iteration, which prevents the LLM from making the same mistakes as in previous iterations. See Appendix G for the used demonstrations.

3.3 Iterative Offline Correction

Similar to Iterative Prompting, the Offline Correction starts from the online LLM, but then continues with the step-by-step verification and correction steps offline. This approach is more cost effective as it relies only on one API call per instance (as opposed to several API calls of iterative prompting), however it is potentially weaker as it relies on the capability of the small verifier LM to both verify and apply the needed corrections. The offline correction stop under the same stopping criterion to Iterative Prompting.

4 Experiments

We describe the datasets and pre-processing method (§4.1), introduce the models and implementation details (§4.2) and the evaluation metrics (§4.3). In Subsection 4.4, we describe the main result from PiVe, and compare the two modes of verifier: Iterative Prompting vs. Iterative Offline Correction (§4.5). We then conduct various configurations of shots (§4.6), and prompting (§4.7). In Subsection 5, we show how PiVe could be used for data augmentation of automatically generated graph-text datasets (e.g., GenWiki).

4.1 Datasets and Preprocessing

We evaluate PiVe on three graph-based datasets, KELM (Agarwal et al., 2021), WebNLG+2020 (Gardent et al., 2017), GenWiki (Jin et al., 2020).

KELM is a large-scale synthetic corpus that consists of the English Wikidata KG and the corresponding natural text. It has ∼15M sentences synthetically generated using a fine-tuned T5 model. Each graph in KELM is a linearised KG containing a list of triples of the form `[subject, relation, object]`. If a triple has a sub-property, then it is quadruplet instead. We use a subset (∼60K) of KELM which is named as KELM-sub. The creation of KELM-sub follows two criteria. We found that most graphs in KELM contain no more than six triples and only around 2,500 graphs contain more than six triples. Therefore, 1) we only consider the graphs with no more than six triples, and 2) we do not consider the graphs containing any triple with a sub-property. Based on these two criteria, for each size of graph (from one triple to six triples), we sampled equal number of (`T`,`G`) pairs. In total, the created KELM-sub contains 60,000/1,800/1,800 samples as train/validation/test set.

WebNLG+2020 contains a set of triples extracted from DBpedia (Auer et al., 2007) in 16 distinct DBpedia categories and text description generated using diverse lexicalisation patterns. It contains ∼38K graphs and each graph has at most three different descriptions.

GenWiki is a large-scale, general-domain dataset collected from general Wikipedia which contains 1.3 million non-parallel text and graphs with shared content. It has two versions: $\text{GenWiki}_{\text{FULL}}$ (∼1.3M), and a fine version, $\text{GenWiki}_{\text{FINE}}$ (∼750K), which adds constraints on the text and

Dataset	Train	Dev
KELM-sub	110,000	3,300
WebNLG & GenWiki	70,630	2,500

Table 1: Statistics of the seed datasets for training the verifier modules on three datasets.

graphs to force them to contain highly overlapped entity sets. Both datasets are collected in a scalable and automatic way. GenWiki also has a human-annotated test set of 1,000 parallel text-graph pairs with high quality. Since both GenWiki$_{\text{FULL}}$ and GenWiki$_{\text{FINE}}$ are non-parallel text and graphs datasets, we cannot use it to train the verifier module. However, the relation types in GenWiki and WebNLG have some overlaps, so we use the verifier module trained on WebNLG+2020 and test it on GenWiki test set.

We use the method described in Section 3.1 to create the data for training the verifier module. Table 1 shows the statistics of the created training data on these three seed datasets.

4.2 The LLM and Verifier Modules

ChatGPT (`gpt-3.5-turbo`) is used as our default LLM to perform the T2G task.[3] We also experiment with GPT-4 in Subsection 4.7. For verifier module, we use T5-Large (Raffel et al., 2020), and Flan-T5-XXL (Chung et al., 2022) as the backbone models for dataset-specific verifier module, and unified verifier module, respectively. T5 models follow the encoder-decoder architecture and treat all NLP tasks as unified text-to-text transduction tasks. Flan-T5 is instruction-fine-tuned version of T5 which was trained on 1,836 NLP tasks initialized from fine-tuned T5 checkpoint. For T5-large, we fine-tune all parameters for separate verifier modules per each dataset. While for Flan-T5-XXL, we use LoRA (Hu et al., 2022) as a parameter-efficient fine-tuning method, to train a unified verifier module which can follow the instruction. When using the unified verifier, we specify the dataset name in the instructions as datasets have different naming convention for relations.

The verifier is implemented using Pytorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020). For the training, we use Adam optimizer (Kingma and Ba, 2015). Details about hyperparameter setting is provided in Appendix A. For the implementation of parameter efficient training method used in Flan-T5-XXL, we use PEFT (Mangrulkar et al., 2022) and 8-bit quantization technique (Dettmers et al., 2022). All training was done using a single A40 GPU with 48GB of RAM.

4.3 Evaluation Metrics

To evaluate the quality of the generated graphs given the ground-truth graphs, we use four automatic evaluation metrics:

Triple Match F1 (T-F1) calculates F1 score based on the precision and recall between the triples in the generated graph and the ground-truth. We calculate the F1 scores for all test samples and compute the average F1 score as the final triple Match F1 score.

Graph Match F1 (G-F1) focuses on the entirety of the graph and evaluates how many graphs are exactly produced the same. For all test samples, we calculate the F1 score based on the precision and recall between all predicted graphs and all ground-truth graphs. This F1 score is the final Graph Match F1 score. Since graphs are represented in a linearised way, we could not simply use the string match method to check whether two graphs are the same. Instead, we first build directed graphs from linearised graphs using NetworkX (Laboratory et al., 2008), then we consider the two graphs to be the same when all node and edge attributes match.

G-BERTScore (G-BS) is a semantic-level metric proposed by (Saha et al., 2021), which extends the BERTScore (Zhang et al., 2020) for graph-matching. It takes graphs as a set of edges and solve a matching problem which finds the best alignment between the edges in predicted graph and those in ground-truth graph. Each edge is considered as a sentence and BERTScore is used to calculate the score between a pair of predicted and ground-truth edges. Based on the best alignment and the overall matching score, the computed F1 score is used as the final G-BERTScore.

Graph Edit Distance (GED) (Abu-Aisheh et al., 2015) computes the distance between the predicted graph and the ground-truth graph. It measures how many edit operations (addition, deletion, and replacement of nodes and edges) are required for transforming the predicted graph to a graph isomorphic to the ground-truth graph. Lower GED between two graphs indicates the two graphs are more similar. In practice, the cost of each operation

[3]We also compare the graph-based generative capability between ChatGPT and GPT-3 in Appendix D and report the fine-tuned results on small language model in Appendix E.

		Single Verifier Module				Unified Verifier Module			
		T-F1↑	**G-F1↑**	**G-BS↑**	**GED↓**	**T-F1↑**	**G-F1↑**	**G-BS↑**	**GED↓**
KELM-sub	Base	13.50	4.89	83.92	13.20	13.50	4.89	83.92	13.20
	Iteration 1	17.92	5.78	85.91	12.37	19.64	6.00	86.39	12.08
	Iteration 2	19.46	6.44	86.57	12.08	22.11	6.44	87.31	11.68
	Iteration 3	20.17	6.61	86.83	11.95	23.11	7.50	87.70	11.35
WebNLG	Base	17.29	13.43	89.59	11.46	17.29	13.43	89.59	11.46
	Iteration 1	18.32	14.00	89.74	11.23	18.22	13.83	89.67	11.23
	Iteration 2	18.57	14.00	89.82	11.22	18.55	13.88	89.74	11.20
GenWiki	Base	20.13	6.60	88.48	10.99	20.13	6.60	88.48	10.99
	Iteration 1	20.54	6.80	88.70	10.87	20.88	6.70	88.66	10.90
	Iteration 2	21.09	6.80	88.78	10.83	20.99	6.70	88.91	10.88

Table 2: Results of using PiVe on three datasets across all metrics. Single verifier module represents single dataset-specific verifier module trained on T5-Large and Unified verifier module is trained on Flan-T5-XXL using instruction-tuning.

is set to be 1. For each sample, GED is normalized by a normalizing constant which is the upper bound of GED to make sure it is between 0 and 1. For demonstration, we multiply GED by 100 to show more decimals.

4.4 Results

We report the evaluation results of using PiVe with ChatGPT on the test set of three datasets in Table 2. All results presented under Base mean the direct output of the LLM without any verification. By utilising PiVe, on each dataset, we can see the consistent improvement of the quality of the generated graphs. For instance, in GenWiki which uses the same verifier module that was trained on the training data of WebNLG, the improvement of the scores over all metrics indicates the effectiveness of PiVe. Since the graphs are generated by the LLM through one-shot learning, G-F1 as the most strict metric, it is hard to get high G-F1 score (basically aiming for exact match without any minor deviation in wording, spelling, entities, or relations).

On WebNLG and GenWiki datasets, single verifier module performs slightly better than unified verifier module. While on KELM-sub dataset, unified module performs far better. We speculate this is due to the size of training data for KELM-sub verifier module being larger than that for WebNLG and GenWiki (as shown in Table 1). Since unified verifier module combines the training data of different datasets, more training data leads to better performance for instruction-tuning. We conducted human evaluation which we include in Appendix F due to page limit.

4.5 Iterative Prompting vs. Iterative Offline Correction

Instead of iteratively prompting the LLM, another way to utilise the results from verifier module is to append the predicted missing triples to the previously generated graph. The results of the comparison between iterative prompting and iterative offline correction using single verifier module and unified verifier module on KELM dataset is shown in Table 3. Iterative offline correction performs worse than iteratively prompting. This might be because iteratively prompting has the chance of doing self-correction. In each iteration, when we prompt the LLM, the generated graphs can probably correct the mistakes that were made in previous iteration. For example, in Figure 4, in Base the predicted relation regarding birth date is "birth year", while the reference is "date of birth". As the PiVe iteration continues, in Iteration 2, the relation "birth year" is regenerated as "date of birth" even though we didn't mention this in the prompt. Due to the page limit, we report the comparison results on WebNLG and GenWiki datasets in Appendix B. Similarly, iterative prompting can achieve better results than iterative offline correction over all using different verifier modules.

4.6 Impact of More Shots

While in our main experiments, for cost reason, we used only one-shot demonstrations for the LLM prompting (i.e., GPT-3.5), we show that PiVe is effective in improving the results regardless of the underlying number of shots. Here we report the results of k-shot (k=6, 8, 10) with the iterative

		Iterative Prompting					Iterative Offline Correction				
		Time	T-F1↑	G-F1↑	G-BS↑	GED↓	Time	T-F1↑	G-F1↑	G-BS↑	GED↓
Single Verifier	Base	36.0	13.50	4.89	83.92	13.20	36.0	13.50	4.89	83.92	13.20
	Iteration 1	+13.5	17.92	5.78	85.91	12.37	+4.5	17.76	5.83	86.42	12.37
	Iteration 2	+4.7	19.46	6.44	86.57	12.08	+1.1	18.51	6.11	86.91	12.19
	Iteration 3	+2.1	20.17	6.61	86.83	11.95	+0.2	18.55	6.17	86.94	12.18
Unified Verifier	Base	36.0	13.50	4.89	83.92	13.20	36.0	13.50	4.89	83.92	13.20
	Iteration 1	+118.9	19.64	6.00	86.39	12.08	+105.0	16.99	5.67	87.08	12.95
	Iteration 2	+46.8	22.11	6.44	87.31	11.68	+40.6	17.76	5.67	87.48	12.96
	Iteration 3	+21.1	23.11	7.50	87.70	11.35	+10.4	17.85	5.67	87.52	12.96

Table 3: Comparison between Iterative Prompting and Iterative Offline Correction on KELM-sub dataset across all metrics using Single Verifier and Unified Verifier. Time denotes the total inference time in minutes.

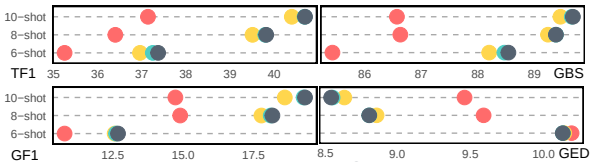

Figure 2: Results of various number of shots (k=6, 8, 10) on KELM-sub with Iterative Offline Correction. The colors represent Base, and corrective iterations 1, 2, 3.

	T-F1↑	G-F1↑	G-BS↑	GED↓
Base	13.50	4.89	83.92	13.20
Iteration 1	14.54	3.00	84.85	13.90
Iteration 2	14.33	2.44	84.57	14.43
Iteration 3	13.79	2.28	84.20	15.18

Table 4: Self-Refine (Madaan et al., 2023) results on KELM-sub using Single Verifier.

offline correction (i.e., only using the LLM once to get the initial graph, while the correction steps are all applied step-by-step and offline). Figure 2 demonstrates the results on KELM-sub using unified verifier with iterative offline correction. The results show, as expected, that PiVe still provides consistent improvement even with the increase in the number of shots. Additionally, as the shots grow the improvement from PiVe also increases.

4.7 Baselines on LLMs

To probe other prompting techniques as baselines of generating graphs from the LLM, we compare three diverse prompts. The first one we use is the default prompt used across our main experiments. This prompt is fairly direct and simple. Prompt 1: `Transform the text into a semantic graph.` In the second prompt, we aim to instruct the LLM to generate larger graph with more triples. This is to increase the chance of LLM recovering more triples during the generation. Prompt 2: `Transform the text into a semantic graph consisting of a set of triples. Generate as many triples as possible.` For the third prompt, inspired by Chain-of-thought (Wei et al., 2022; Kojima et al., 2022) approach, we also ask the LLM to generate the semantic graph in two steps. Prompt 3: `Transform the text into a semantic graph consisting of a set of triples. First produce all relations possible, then produce the graph.`

We conduct experiments on ChatGPT (`gpt-3.5-turbo`) and GPT-4 (`gpt-4`) in 6-shot learning on KELM-sub, using unified verifier with iterative offline correction. The results are shown in Figure 3 (for detailed numbers see Table 14 and Table 15 in Appendix). In general, as expected, GPT-4 performs far better than ChatGPT on the T2G task, but the effect of different prompts varies across these two models. Specifically, on ChatGPT, Prompt 2 achieves the best results while on GPT-4, Prompt 1 is outperforming the rest on most metrics. PiVe can consistently improve the results across all different settings, with the biggest jump in performance emerging in the first iteration, with slight improvements also observed between the second and third iterations of correction.

We also compare PiVe with the recent Self-Refine (Madaan et al., 2023) method, which leverages LLM itself to provide feedbacks for self-refinement. Table 4 shows the self-refine results on KELM-sub dataset. The results show that self-refine could not provide effective feedback, thus leading to the performance drop as the iteration goes. The performance gap is obvious comparing with our PiVe result. Since LLMs are not trained as rigorously on structured data compared to text, expecting them to provide meaningful feedbacks on their outputs is expected to fail.

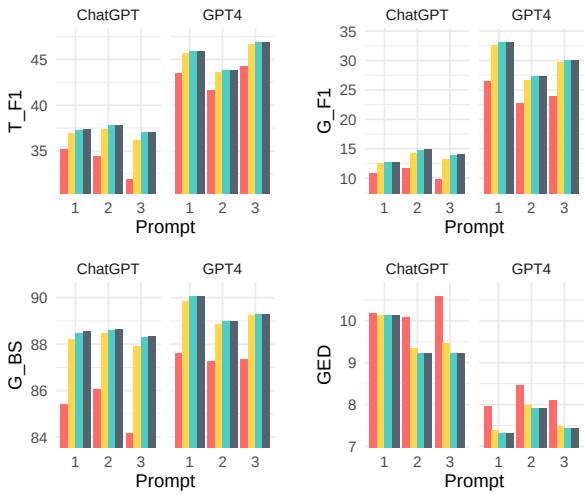

Figure 3: Results of using 3 diverse prompts with 6-shot on KELM-sub with Iterative Offline Correction on ChatGPT and GPT-4. The colors represent Base, and corrective iterations 1, 2, 3.

4.8 Computational Cost and Trade-off

Training and Inference The training and inference of both single verifiers and unified verifier are on a single A40 GPU. Each single verifier takes around 6 hours and the unified verifier takes around 40 hours to train. The computation cost for training of verifiers is a feasible one-off cost. Once the training is finished, the inference of the verification of each instance takes 0.15s for single verifier, and 3.5s for unified verifier. See the Time column in Table 3. Different verifiers present performance-speed trade-offs and are significantly effective in augmenting the LLMs.

Stopping Criterion In theory, the verification module could run till no missing triple is predicted or a maximum number of iterations is reached. However, running more iterations increases the associated cost (i.e., OpenAI API charges). We set a maximum of 3 iterations.

4.9 PiVe Examples

In Figure 4, we demonstrate an example from KELM-sub test set using unified verifier. In Base, based on the prediction from the LLM, the verifier module predicts the missing triple `["Francisco Uranga", "occupation", "swimmer"]`. By suggesting this missing triple in the next iteration of prompt, the prediction from LLM includes it. Then, the verifier predicts the missing triple `["Francisco Uranga", "sex or gender", "male"]`. In Iteration 2, both of these two missing triples are included in the prediction from LLM, and at this time, the verifier predicts "Correct". The prediction from Iteration 2 contains all information correctly in the reference. See another example in Appendix H.

5 GenWiki-HIQ

Training a good G2T or T2G model requires a large amount of high-quality parallel text and graph pairs or pre-training protocols to accommodate for lack of data (Han and Shareghi, 2022). However, creating the parallel data by human is a labour-intensive and time-consuming work. Jin et al. (2020) proposed GenWiki, an automatically constructed large dataset containing non-parallel text and graphs with shared content. Although the text and graphs contain shared content, it can still only be used for unsupervised training due to the low entity and relation overlap between text and graph. Our verifier module can naturally serve as a data augmentation tool to improve the overlap between the text and graph of automatically constructed datasets.

Iterative Augmentation. Based on GenWiki$_{\text{FINE}}$ (∼750K), first we filter out the text that has little overlap with the graph. After filtering, we got around 110K text-graph pairs called GenWiki$_{\text{FINE-f}}$. Then following the process described in Section 3.1, we use the WebNLG verifier module and the iterative offline correction to improve the coverage and quality of GenWiki$_{\text{FINE-f}}$ and formed GenWiki-HIQ. The maximum number of iterations is four.

To evaluate the effectiveness of the verifier module as data augmentation tool, as well the quality of the generated graph, first we use Flan-T5-XL model to generate a description of each graph in zero-shot setting by using the prompt `"Transform the semantic graph into a description."` for each iteration. Then we leverage automatic quality evaluation metrics to calculate the score between the generated description and the corresponding text. Ideally, the higher the similarity between the graph and the corresponding text, the higher the score of the generated description and corresponding text. We use four commonly used quality evaluation metrics which are BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), TER (Snover et al., 2006), BERTScore.

Result. We used the dataset-specific verifier module to do the data augmentation. We conducted four iterations and the evaluation results are shown in Ta-

Text: Francisco Uranga was born in 1905 and represented Argentina at the 1928 Summer Olympics. He competed in the men's 50 metre freestyle.

Base

LLM Prediction: [["Francisco Uranga", "birth year", "1905"], ["Francisco Uranga", "represented", "Argentina"], ["Argentina", "participated in", "1928 Summer Olympics"], ["Francisco Uranga", "competed in", "men's 50 metre freestyle"]]

Verification Module Output: ["Francisco Uranga", "occupation", "swimmer"]

Iteration 1

LLM Prediction: [["Francisco Uranga", "occupation", "swimmer"], ["Francisco Uranga", "birth year", "1905"], ["Francisco Uranga", "represented", "Argentina"], ["Argentina", "participated in", "1928 Summer Olympics"], ["Francisco Uranga", "competed in", "men's 50 metre freestyle"]]

Verification Module Output: ["Francisco Uranga", "sex or gender", "male"]

Iteration 2

LLM Prediction: [["Francisco Uranga", "occupation", "swimmer"], ["Francisco Uranga", "sex or gender", "male"], ["Francisco Uranga", "date of birth", "1905"], ["Francisco Uranga", "represented", "Argentina"], ["Argentina", "participated in", "1928 Summer Olympics"], ["Francisco Uranga", "competed in", "men's 50 metre freestyle"]]

Verification Module Output: Correct

Reference: [["Francisco Uranga", "occupation", "swimmer"], ["Francisco Uranga", "date of birth", "01 January 1905"], ["Francisco Uranga", "country of citizenship", "Argentina"], ["Francisco Uranga", "sex or gender", "male"], ["Francisco Uranga", "participant of", "1928 Summer Olympics"], ["Francisco Uranga", "sport", "Swimming (sport)"]]

Figure 4: An example from KELM-sub test set using unified verifier module.

	BLEU↑	METEOR↑	TER↓	BERTScore↑
Base	4.75	18.02	80.72	89.47
Iteration 1	11.86	26.25	73.79	91.49
Iteration 2	14.90	29.69	72.29	92.00
Iteration 3	15.93	31.05	72.01	92.14

Table 5: Results of iterative augmentation on filtered 110K text-graph pairs from GenWiki$_{\text{FINE}}$ across four metrics after each iteration. We take the text-graph pairs from Iteration 3 as the created GenWiki-HIQ dataset.

ble 5. The results in Base represent the scores over non-parallel graph-text pairs from GenWiki$_{\text{FINE}}$, which have low overlap between graph and text. By using verifier module iteratively, we add more missing triples to the original graph, thus leading the higher quality scores. As the iteration progresses, fewer missing triples are added and we take the augmented graph-text pairs from the last iteration as the final created GenWiki-HIQ dataset. We also conducted G2T experiments in Appendix 5.1 to further demonstrate the quality of GenWiki-HIQ. The G2T model trained on GenWiki-HIQ performs far better than the G2T model trained on GenWiki$_{\text{FINE-f}}$ on the human annotated GenWiki test set. This indicates that GenWiki-HIQ contains parallel text-graph pairs with high overlap.

Qualitative Example. In Figure 5, we demonstrate an example from the created GenWiki-HIQ dataset and the original graph in GenWiki$_{\text{FINE}}$. After the data augmentation process, the graph in GenWiki-HIQ contains more information in text.

Text: Timma is a village development committee in Bhojpur District in the Kosi Zone of eastern Nepal. At the time of the 1991 Nepal census it had a population of 3336 persons living in 621 individual households.

GenWiki-FINE:
[[Timma, populationTotal, 3336],
[Timma, pushpinMap, Nepal],
[Timma, country, Nepal]]

GenWiki-HIQ:
[[Timma, populationTotal, 3336],
[Timma, pushpinMap, Nepal],
[Timma, country, Nepal],
[Timma, is Part Of, Bhojpur District Kosi Zone],
[Timma, county Development Committee, Bhojpur District],
[Timma, number Of Households, 621]]

Figure 5: An example from GenWiki-HIQ compared to the original graph in GenWiki$_{\text{FINE}}$.

5.1 G2T Results on GenWiki-HIQ

To further verify the quality of GenWiki-HIQ dataset, we use T5-large as the backbone model to train a G2T model, which generates the corresponding text based on the graph. Then we test it on the original GenWiki test set containing a 1,000 high-quality human annotated parallel text-graph pairs. As comparison, we also train another G2T

	BLEU↑	METEOR↑	TER↓	BERTScore↑
$\text{GenWiki}_{\text{FINE-f}}$	35.71	36.67	65.19	93.74
GenWiki-HIQ	48.17	42.03	41.94	95.44

Table 6: Results of G2T generation on original GenWiki test set training on different datasets. $\text{GenWiki}_{\text{FINE-f}}$ contains the filtered 110K text-graph pairs from original $\text{GenWiki}_{\text{FINE}}$ as described in Section 5. GenWiki-HIQ is the augmented dataset based on $\text{GenWiki}_{\text{FINE-f}}$.

model on $\text{GenWiki}_{\text{FINE-f}}$ which is the seed dataset of GenWiki-HIQ.

The result is demonstrated in Table 6. On original Genwiki test set, the model trained on GenWiki-HIQ performs far better than the model trained on $\text{GenWiki}_{\text{FINE-f}}$ across all metrics. This indicates that GenWiki-HIQ contains parallel text-graph pairs with high overlap.

6 Background and Related Work

6.1 In-Context Learning

With the scaling of model size and training corpus size (Brown et al., 2020; Chowdhery et al., 2022), LLMs demonstrate new abilities of learning from a few demonstrations which contain some training examples (Dong et al., 2023). As a new paradigm, In-Context Learning does not require parameter updates and directly performs predictions on the pre-trained language models. The provided demonstration examples in the prompt follow the same format, which are usually written in natural language templates. By concatenating a query question with the demonstrations in the prompt, LLMs can learn from the given examples and make a prediction of the query question. Previous research (Liu et al., 2022; Lu et al., 2022) has shown that the number and order of the demonstrations can influence the In-Context Learning performance. These are further points of future investigation, which could potentially improve the initial graph produced by the LLM, which could further be corrected with the PiVe framework.

6.2 Instruction-Tuning

Instruction-Tuning (Mishra et al., 2022; Wang et al., 2022; Longpre et al., 2023) is a framework of doing multi-task learning, which enables the use of human-readable instructions to guide the prediction of LLMs. This novel training paradigm can improve the performance of the downstream tasks and also shows great generalisation ability on unseen tasks (Chung et al., 2022; Sanh et al., 2022). Wang et al. (2023) proposed a unified information extraction framework based on multi-task instruction-tuning. Zhou et al. (2023) utilised instruction-tuning to perform controlled text generation following certain constraints. In our work, we use instruction-tuning to train a unified verifier module, which can follow the instruction to perform predictions on different datasets.

6.3 Verifiers

Leveraging small models could further improve the performance of LLMs. Cobbe et al. (2021) proposed to solve math word problem by utilising verifier. The verifier is used to judge the correctness of model-generated solutions. During test time, based on multiple candidate solutions generated, verifier calculates the correctness probability and the final answer will be selected by the verifier from the ranked list. Welleck et al. (2023) proposed self-correction, an approach that trains a small model to iteratively apply self-correction. The idea of self-correction looks similar to our PiVe. While Welleck et al. (2023) focuses on the design of a self-correcting language model, PiVe presents a very simple verifier module design and a simple data perturbation strategy to train such model. The ideas presented in our work are developed concurrently and independently.

7 Conclusion

We proposed PiVe, an iterative verification framework, to improve the graph-based generative capability of LLMs. We illustrated how a simple perturbation technique could be used to build data for training a verifier module which both verifies and corrects outputs from an LLM. We used different training strategies to build both dataset-specific verifiers with fine-tuning, and a unified verifier with instruction-tuning. Our verifier module could act both as an iterative prompting guide to improve outputs of an LLM, as well as an iterative offline correction system that starts from an LLM outputs but continuously improves it offline. The experimental results on three graph-based datasets demonstrates the effectiveness of PiVe. Furthermore, PiVe can also be used as a data augmentation technique to help improve the quality of automatically generated parallel text-graph datasets. By using verifier module, we created GenWiki-HIQ, a dataset containing 110K parallel text and graphs with high overlap for future research in text-graph domain.

Limitations

Although the proposed framework is a straightforward and effective method of improving the generative capabilities of black box LLMs in graph generation, it still has some limitations. Firstly, PiVe is only designed for few-shot prompting setting on LLMs, using an external verifier module to enhance their generative capabilities. The improvement is less significant when utilising PiVe on LMs that have been fine-tuned on the task data. Secondly, PiVe is not designed for free-form text generation tasks. Due to the unique aspect of graph, which has a specific structure, it allows for a much more fine-grained detection of errors and enables a richer corrective feedback. Translation between text and other similar modalities of data (e.g., table, SQL) can also effectively leverage our verification mechanism. Thirdly, in this work, we only focus on the triple missing mistake made by LLMs, so that the verifier module is not sensitive to the order of the head entity and tail entity. This means when the order of the head entity and tail entity in a triple of a generated graph from LLMs is incorrect, verifier module is not able to detect this type of mistake. It would be more effective if other error-detection heuristic methods are developed for creating the training dataset of the verifier.

Ethics Statement

Our work is built on top of existing pre-trained language models. Our goal was not to attend to alleviate the well-documented issues (e.g., privacy, undesired biases, etc) that such models embody. For this reason, we share the similar potential risks and concerns posed by these models.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An exact graph edit distance algorithm for solving pattern recognition problems. In *ICPRAM 2015 - Proceedings of the International Conference on Pattern Recognition Applications and Methods, Volume 1, Lisbon, Portugal, 10-12 January, 2015*, pages 271–278. SciTePress.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3554–3565. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

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. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang,

Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit optimizers via block-wise quantization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey for in-context learning. *CoRR*, abs/2301.00234.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *CoRR*, abs/2006.04702.

Jiuzhou Han and Ehsan Shareghi. 2022. Self-supervised graph masking pre-training for graph-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4845–4853, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2398–2409. International Committee on Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

Los Alamos National Laboratory, United States. Department of Energy. Office of Scientific, and Technical Information. 2008. *Exploring Network Structure, Dynamics, and Function Using Networkx*. United States. Department of Energy.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, pages 100–114. Association for Computational Linguistics.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *CoRR*, abs/2301.13688.

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. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8086–8098. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3470–3487. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7716–7740. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Matthew G. Snover, Bonnie J. Dorr, Richard M. Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers, AMTA 2006, Cambridge, Massachusetts, USA, August 8-12, 2006*, pages 223–231. Association for Machine Translation in the Americas.

Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors. 2008. *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *CoRR*, abs/2304.08085.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5085–5109. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Yi Xu, Luoyi Fu, Zhouhan Lin, Jiexing Qi, and Xinbing Wang. 2022. INFINITY: A simple yet effective unsupervised framework for graph-text mutual conversion. *CoRR*, abs/2209.10754.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International*

Hyperparameter	Assignment
Model	T5-Large
Epoch	5
Batch Size	16
Optimizer	Adam
Learning Rate	2×10^{-5}
Warm-up Step	500
Beam Size	5

Table 7: Hyperparameters of single verification module.

Hyperparameter	Assignment
Model	FLAN-T5-XXL
Epoch	2
Batch Size	48
Optimizer	Adam
Learning Rate	3×10^{-5}
Warm-up Step	100
Beam Size	4

Table 8: Hyperparameters of unified verification module.

Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. *CoRR*, abs/2304.14293.

Appendix

A Hyperparameter Setting

B Additional Experiment Result

Table 9 and Table 10 show the results of the comparison between iteratively prompt and iterative offline correction on WebNLG and GenWiki datasets.

C Effect of Perturbation Method

As described in Section 3.1, we perturb the graph by omitting one triple when building the verifier module of PiVe. In addition, we also investigated other perturbation methods to train a verifier module, such as perturbing the head entity, relation and tail entity. To be specific, for head entity perturbation, if the graph contains more than one triple, we randomly choose one triple and replace the head entity with a different head entity from other triples of the same graph. Likewise, we replace the relation and tail entity for relation perturbation and tail entity perturbation, respectively. The target is to predict the original triple. Then we train different verifier modules using these three perturbation methods on KELM-sub. The results of doing different perturbations using Iterative Offline Correction is shown in Table 11.

Comparing with the result of omitting triple perturbation method shown in Table 3 using Single Verifier with Iterative Offline Correction, these three perturbation methods have varying effects. While the relational perturbation works in terms of T-F1, with more iterations, the G-BS score generally goes down for all these perturbations. This indicates the verifier module could potentially inject wrong corrections if not trained with the proper perturbation mechanism. We speculate the reason is because LLMs are less likely to make mistakes at entity level, so these perturbation methods are not useful for training a verifier module. This also indicates when building a verifier module, choosing reasonable perturbation methods is significant and necessary.

D ChatGPT vs GPT-3

To further highlight the generalisation ability of PiVe, in addition to ChatGPT, we also experiment with GPT-3 (`text-davinci-003`) as the backbone LLM to perform the T2G task. We perform experiments on KELM-sub dataset using iterative prompting and iterative offline correction with different verifiers. The results are shown in Table 13. Compared with the results of using ChatGPT (shown in Table 3), GPT-3 has a better graph-based generative capability. Nonetheless, PiVe can still consistently further improve its results over all settings. Using iterative prompting with the unified verifier can achieve the best result on KELM-sub.

E Comparison with Fine-tuned Baselines

While our work focuses on the fundamental question of "How can we improve the generative capabilities of black box LLMs in graph generation?", for completeness we also provide results of fine-tuned T5 (Raffel et al., 2020) in Table 12. As expected, fine-tuning on large amount of data surpasses few-shot prompting. This underscores the struggle LLMs face in transduction problems, and the need for additional mechanisms (like PiVe) to help LLM improvement.

		Iterative Prompting				Iterative Offline Correction			
		T-F1↑	G-F1↑	G-BS↑	GED↓	T-F1↑	G-F1↑	G-BS↑	GED↓
Single Verifier	Base	17.29	13.43	89.59	11.46	17.29	13.43	89.59	11.46
	Iteration 1	18.32	14.00	89.74	11.23	18.03	13.55	89.16	11.52
	Iteration 2	18.57	14.00	89.82	11.22	18.10	13.55	89.19	11.51
Unified Verifier	Base	17.29	13.43	89.59	11.46	17.29	13.43	89.59	11.46
	Iteration 1	18.22	13.83	89.67	11.23	18.02	13.49	89.21	11.61
	Iteration 2	18.55	13.88	89.74	11.20	18.06	13.49	89.07	11.65

Table 9: Comparison between Iterative Prompting and Iterative Offline Correction on WebNLG dataset across all metrics using Single Verifier and Unified Verifier.

		Iterative Prompting				Iterative Offline Correction			
		T-F1↑	G-F1↑	G-BS↑	GED↓	T-F1↑	G-F1↑	G-BS↑	GED↓
Single Verifier	Base	20.13	6.60	88.48	10.99	20.13	6.60	88.48	10.99
	Iteration 1	20.54	6.80	88.70	10.87	20.24	6.70	89.00	10.95
	Iteration 2	21.09	6.80	88.78	10.83	20.32	6.80	89.07	10.93
Unified Verifier	Base	20.13	6.60	88.48	10.99	20.13	6.60	88.48	10.99
	Iteration 1	20.88	6.70	88.66	10.90	20.37	6.60	89.08	10.96
	Iteration 2	20.99	6.70	88.91	10.88	20.42	6.60	89.11	10.94

Table 10: Comparison between Iterative Prompting and Iterative Offline Correction on GenWiki dataset across all metrics using Single Verifier and Unified Verifier.

		T-F1↑	G-F1↑	G-BS↑	GED↓
Head	Base	13.50	4.89	83.92	13.20
	Iteration 1	13.66	4.89	83.23	13.31
	Iteration 2	13.65	4.89	81.99	13.31
	Iteration 3	13.65	4.89	80.83	13.31
Relation	Base	13.50	4.89	83.92	13.20
	Iteration 1	15.09	4.94	83.68	12.97
	Iteration 2	15.29	4.94	82.90	12.95
	Iteration 3	15.33	4.94	82.09	12.96
Tail	Base	13.50	4.89	83.92	13.20
	Iteration 1	13.52	4.89	83.83	13.21
	Iteration 2	13.51	4.89	83.74	13.22
	Iteration 3	13.50	4.89	83.64	13.23

Table 11: Results of doing different perturbations to the graph on KELM-sub to train a Single Verifier with Iterative Offline Correction.

	T-F1↑	G-F1↑	G-BS↑	GED↓
KELM-sub	58.45	47.26	94.12	8.48
WebNLG	54.77	45.31	93.51	9.11
GenWiki	36.34	29.69	91.14	9.74

Table 12: Fine-tuning results of text-to-graph generation on three datasets on T5-Large model.

F Human Evaluation

We conducted a human evaluation on 105 randomly sampled instances from three datasets (KELM-sub, WebNLG, GenWiki). Specifically, for each dataset, first we took the test set outputs from the first iteration and the last iteration, then we randomly sampled 35 instances from those with different outputs. The output from the first iteration is the original ChatGPT output without using PiVe, and the output from the last iteration is the result after using PiVe. For the evaluation process we recruited three annotators (1 PhD graduate and 2 PhD students in Computer Science and NLP) to select, for a given text and two graph outputs, which graph matches the text better. Each annotator should only choose one graph per each instance and evaluate all 105 instances.

After annotation, we took majority voting over the result of each instance, then calculated the number of wins for ChatGPT with or without PiVe. The results are shown in Table 16. From the results, we can see ChatGPT with PiVe wins on 85 out of 105 samples and the total winning rate is over 80%. This indicates the PiVe can effectively improve the graph-based generative capability of LLMs.

For the cases that PiVe failed to improve, we did error analysis and found that there were mainly two types of mistakes that PiVe made: redundancy and inaccuracy. In Figure 6, we demonstrate two examples containing these two types of mistakes shown in red text. In the first example, the triple `["Train song Mermaid", "instrument","Singing"]` predicted by PiVe is redundant. In the second example, the relation

		Iterative Prompting				Iterative Offline Correction			
		T-F1↑	G-F1↑	G-BS↑	GED↓	T-F1↑	G-F1↑	G-BS↑	GED↓
Single Verifier	Base	15.11	7.72	83.63	12.91	15.11	7.72	83.63	12.91
	Iteration 1	19.55	8.78	85.90	11.98	18.97	8.72	86.47	12.10
	Iteration 2	21.57	9.33	86.59	11.56	19.65	9.00	86.76	11.96
	Iteration 3	22.49	9.89	87.10	11.37	19.69	9.00	86.77	11.95
Unified Verifier	Base	15.11	7.72	83.63	12.91	15.11	7.72	83.63	12.91
	Iteration 1	21.40	8.78	86.43	11.69	20.46	8.78	87.18	11.90
	Iteration 2	24.37	9.50	87.50	11.12	21.54	9.06	87.56	11.71
	Iteration 3	26.06	10.22	87.99	10.83	21.57	9.06	87.57	11.71

Table 13: Results of using GPT-3-davinci as the backbone LLM on KELM-sub dataset over different settings.

		T-F1↑	G-F1↑	G-BS↑	GED↓
Prompt 1	Base	35.25	10.78	85.43	10.19
	Iteration 1	36.96	12.56	88.20	10.14
	Iteration 2	37.25	12.61	88.47	10.13
	Iteration 3	37.37	12.68	88.54	10.13
Prompt 2	Base	34.46	11.56	86.07	10.08
	Iteration 1	37.38	14.22	88.48	9.35
	Iteration 2	37.81	14.72	88.61	9.24
	Iteration 3	37.85	14.89	88.62	9.23
Prompt 3	Base	31.89	9.78	84.16	10.58
	Iteration 1	36.15	13.22	87.89	9.46
	Iteration 2	37.03	13.94	88.29	9.23
	Iteration 3	37.11	13.95	88.34	9.23

Table 14: Results of using diverse prompts with 6-shot learning on KELM-sub with Iterative Offline Correction on ChatGPT.

		T-F1↑	G-F1↑	G-BS↑	GED↓
Prompt 1	Base	43.46	26.50	87.60	7.97
	Iteration 1	45.68	32.50	89.86	7.39
	Iteration 2	45.87	33.06	90.04	7.32
	Iteration 3	45.87	33.06	90.05	7.31
Prompt 2	Base	41.67	22.67	87.28	8.47
	Iteration 1	43.64	26.61	88.87	7.98
	Iteration 2	43.79	27.22	88.99	7.91
	Iteration 3	43.79	27.28	89.00	7.91
Prompt 3	Base	44.30	23.89	87.36	8.11
	Iteration 1	46.65	29.61	89.22	7.49
	Iteration 2	46.84	30.06	89.28	7.45
	Iteration 3	46.85	30.08	89.30	7.45

Table 15: Results of using diverse prompts with 6-shot learning on KELM-sub with Iterative Offline Correction on GPT-4.

Dataset	# with PiVe wins	# w/o PiVe wins
KELM-sub	31	4
WebNLG	28	7
GenWiki	26	9
Total	85	20

Table 16: Human evaluation results on 105 samples from three datasets using ChatGPT with or w/o PiVe.

"date Of Retirement" in the triple ["Alan Shepard","date Of Retirement","1963"] is inaccurate. We speculate these behaviours were caused due to the presence of many similar texts with similar graphs in the training data. During training, PiVe learned the potential connections between these similar graphs, thus leading to redundant and inaccurate triples at prediction.

G Demonstrations in Prompt

Figure 7 shows the demonstrations used for KELM-sub and Figure 8 shows the demonstrations used for WebNLG and GenWiki. In Iteration 1, we use the demonstration that does not contain the missing triples. For subsequent iterations, we include the missing triples in the demonstration.

H PiVe Examples

Figure 9 illustrates another example of PiVe from WebNLG test set using single verification module. In Base, the verification module predict the missing triple ["Agremiação Sportiva Arapiraquense", "ground", "Estádio Municipal Coaracy da Mata Fonseca"], even though there is a similar triple but containing mistakes in the prediction from the LLM. In Iteration 1, the LLM corrects the mistakes in the previous iteration, and also includes the predicted missing triple. Based on the prediction from the LLM, the verification module predict the missing triple ["Campeonato Brasileiro Série C", "country", "Brazil"]. In Iteration 2, the verification module predict "Correct" to the final prediction from the LLM. After three iterations using PiVe, the predicted graph contains all information in the reference.

Text: While pop rock can trace its stylistic roots back to rock music, Reggae music evolved out of different musical genre, known as ska. Interestingly, the Train song, Mermaid, belongs to the genre of pop rock, but is also considered to be of the reggae genre as well.

Reference: [["Mermaid Train song", "genre", "Pop rock"], ["Mermaid Train song", "genre", "Reggae"], ["Pop rock", "stylistic Origin", "Rock music"], ["Reggae", "stylistic Origin", "Ska"]]

ChatGPT w/o PiVe: [["pop rock", "stylistic roots", "rock music"], ["Reggae music", "evolved from", "ska"], ["Train song Mermaid", "genre", "pop rock"], ["Train song Mermaid", "also considered", "reggae genre"]]

ChatGPT with PiVe: [["pop rock", "stylistic roots", "rock music"], ["Reggae music", "evolved out of", "ska"], ["Train song Mermaid", "genre", "pop rock"], ["Train song Mermaid", "genre", "reggae"], ["Train song Mermaid", "instrument", "Singing"]]

Text: Alan Shepard was born November 18th, 1923 in New Hampshire and later died in California. Shepard graduated from NWC with an M.A. in 1957 and was also the Chief of the Astronaut Office in 1963.

Reference: [["Alan Shepard", "birth Date", "1923-11-18"], ["Alan Shepard", "served As Chief Of The Astronaut Office In", "1963"], ["Alan Shepard", "death Place", "California"], ["Alan Shepard", "birth Place", "New Hampshire"], ["Alan Shepard", "alma Mater", "NWC M.A. 1957"]]

ChatGPT w/o PiVe: [["Alan Shepard", "birth date", "November 18th, 1923"], ["Alan Shepard", "birth place", "New Hampshire"], ["Alan Shepard", "death place", "California"], ["Alan Shepard", "education", "M.A. from NWC"], ["Alan Shepard", "occupation", "Chief of the Astronaut Office"]]

ChatGPT with PiVe: [["Alan Shepard", "birth place", "New Hampshire"], ["Alan Shepard", "birth date", "November 18th, 1923"], ["Alan Shepard", "death place", "California"], ["Alan Shepard", "education", "M.A. from NWC"], ["Alan Shepard", "occupation", "Chief of the Astronaut Office"], ["Alan Shepard", "status", "Deceased"], ["Alan Shepard", "date Of Retirement", "1963"]]

Figure 6: Two examples of PiVe making two types of mistakes: redundancy and inaccuracy.

Demonstration for Iteration 1:
Transform the text into a semantic graph.
Example:
Text: Shotgate Thickets is a nature reserve in the United Kingdom operated by the Essex Wildlife Trust.
Semantic Graph: [["Shotgate Thickets", "instance of", "Nature reserve"], ["Shotgate Thickets", "country", "United Kingdom"], ["Shotgate Thickets", "operator", "Essex Wildlife Trust"]]

Demonstration for Subsequent Iterations:
Transform the text into a semantic graph and also add the given triples to the generated semantic graph.
Example:
Text: Shotgate Thickets is a nature reserve in the United Kingdom operated by the Essex Wildlife Trust.
Triples: ["Shotgate Thickets", "instance of", "Nature reserve"], ["Shotgate Thickets", "country", "United Kingdom"]
Semantic graph: [["Shotgate Thickets", "instance of", "Nature reserve"], ["Shotgate Thickets", "country", "United Kingdom"], ["Shotgate Thickets", "operator", "Essex Wildlife Trust"]]

Figure 7: The demonstrations used in prompt for KELM-sub.

Demonstration for Iteration 1:
Transform the text into a semantic graph.
Example:
Text: Sportpark De Toekomst is located in Ouder-Amstel, Netherlands. It is owned and operated by AFC Ajax N.V. and their tenants include the Ajax Youth Academy.
Semantic graph: [["Sportpark De Toekomst", "location", "Ouder-Amstel"], ["Sportpark De Toekomst", "country", "Netherlands"], ["Sportpark De Toekomst", "owner", "AFC Ajax N.V."], ["Sportpark De Toekomst", "operator", "AFC Ajax N.V."], ["Sportpark De Toekomst", "tenant", "Ajax Youth Academy"]]

Demonstration for Subsequent Iterations:
Transform the text into a semantic graph and also add the given triples to the generated semantic graph.
Example:
Text: Sportpark De Toekomst is located in Ouder-Amstel, Netherlands. It is owned and operated by AFC Ajax N.V. and their tenants include the Ajax Youth Academy.
Triples: ["Sportpark De Toekomst", "country", "Netherlands"], ["Sportpark De Toekomst", "operator", "AFC Ajax N.V."]
Semantic graph: [["Sportpark De Toekomst", "location", "Ouder-Amstel"], ["Sportpark De Toekomst", "country", "Netherlands"], ["Sportpark De Toekomst", "owner", "AFC Ajax N.V."], ["Sportpark De Toekomst", "operator", "AFC Ajax N.V."], ["Sportpark De Toekomst", "tenant", "Ajax Youth Academy"]]

Figure 8: The demonstrations used in prompt for WebNLG and GenWiki.

Text: Agremiação Sportiva Arapiraquense with 17000 members play in the Brazil Campeonato Brasileiro Série C league. Estádio Municipal Coaracy da Mata Fonseca in located in Arapiraca and ground is Agremiação Sportiva Arapiraquense.

Base

LLM Prediction: [["Agremiação Sportiva Arapiraquense", "member count", "17000"], ["Agremiação Sportiva Arapiraquense", "league", "Campeonato Brasileiro Série C"], ["Estádio Municipal Coaracy da Mata Fonseca", "location", "Arapiraca"], ["Estádio Municipal Coaracy da Mata Fonseca", "home ground", "Agremiação Sportiva Arapiraquense"]]

Verification Module Output: ["Agremiação Sportiva Arapiraquense", "ground", "Estádio Municipal Coaracy da Mata Fonseca"]

Iteration 1

LLM Prediction: [["Agremiação Sportiva Arapiraquense", "membership", "17000"], ["Agremiação Sportiva Arapiraquense", "league", "Campeonato Brasileiro Série C"], ["Agremiação Sportiva Arapiraquense", "ground", "Estádio Municipal Coaracy da Mata Fonseca"], ["Estádio Municipal Coaracy da Mata Fonseca", "location", "Arapiraca"]]

Verification Module Output: ["Campeonato Brasileiro Série C", "country", "Brazil"]

Iteration 2

LLM Prediction: [["Agremiação Sportiva Arapiraquense", "membership", "17000"], ["Agremiação Sportiva Arapiraquense", "league", "Campeonato Brasileiro Série C"], ["Agremiação Sportiva Arapiraquense", "ground", "Estádio Municipal Coaracy da Mata Fonseca"], ["Estádio Municipal Coaracy da Mata Fonseca", "location", "Arapiraca"], ["Campeonato Brasileiro Série C", "country", "Brazil"]]

Verification Module Output: Correct

Reference: [["Agremiação Sportiva Arapiraquense", "league", "Campeonato Brasileiro Série C"], ["Agremiação Sportiva Arapiraquense", "number Of Members", "17000"], ["Agremiação Sportiva Arapiraquense", "ground", "Estádio Municipal Coaracy da Mata Fonseca"], ["Campeonato Brasileiro Série C", "country", "Brazil"], ["Estádio Municipal Coaracy da Mata Fonseca", "location", "Arapiraca"]]

Figure 9: An example from WebNLG test set using single verification module.