# Directions Towards Efficient and Automated Data Wrangling with Large Language Models

Zeyu Zhang[1,2] Paul Groth[1] Iacer Calixto[2,1] Sebastian Schelter[1]
[1]*University of Amsterdam* [2]*Amsterdam UMC*
z.zhang2@uva.nl p.groth@uva.nl i.coimbra@amsterdamumc.nl s.schelter@uva.nl

***Abstract*—Data integration and cleaning have long been a key focus of the data management community. Recent research indicates the potential of large language models (LLMs) for such tasks. However, scaling and automating data wrangling with LLMs for real-world use cases poses additional challenges. Manual prompt engineering for example, is expensive and hard to operationalise, while full fine-tuning of LLMs incurs high compute and storage costs.**

**Following up on previous work, we evaluate parameter-efficient fine-tuning (PEFT) methods for efficiently automating data wrangling with LLMs. We conduct a study of four popular PEFT methods on differently sized LLMs for ten benchmark tasks, where we find that PEFT methods achieve performance on-par with full fine-tuning, and that we can leverage small LLMs with negligible performance loss.**

**However, even though such PEFT methods are parameter-efficient, they still incur high compute costs at training time and require labeled training data. We explore a zero-shot setting to further reduce deployment costs, and propose our vision for ZEROMATCH, a novel approach to zero-shot entity matching. It is based on maintaining a large number of pretrained LLM variants from different domains and intelligently selecting an appropriate variant at inference time.**

## I. INTRODUCTION

Data wrangling problems like entity matching [1], error detection [2], [3] and data imputation [4] have long been a focus of the data management community.

**The potential of LLMs for data wrangling**. Recent research indicates the potential of large language models (LLMs) for data wrangling [5]. In particular, Narayan et al. [6] show that few-shot prompted LLMs (with carefully chosen few-shot samples) obtain state-of-the art performance on several data wrangling benchmark tasks.

**Automation and scalability challenges**. While this strong performance of few-shot prompted LLMs is exciting, there exist several challenges in applying such methods in real-world scenarios [7]. As an example, imagine a cloud provider who wants to offer LLM-based data wrangling services to its customers. This cloud provider cannot use few-shot prompting for their service, as these prompts have to be manually designed with access to the underlying data. This is $(i)$ not scalable as ML experts for these tasks are hard to hire and expensive; and $(ii)$ not operationalisable, as employees of a cloud provider cannot access their customers' data for legal reasons. Automatable alternatives such as fine-tuning the LLMs also incur scalability challenges: $(i)$ the cloud provider cannot fine-tune a single LLMs for all customers, the data and models for different customers must be strictly segmented for legal reasons [8] instead. However, creating and maintaining a full fine-tuned model copy per customer (or even worse per customer/task/dataset combination) does not scale either due to the incurred high compute and storage costs.

**Parameter- and compute-efficient data wrangling with LLMs**. In order to address some of the issues above, the ML community has developed parameter-efficient fine-tuning (PEFT) methods [9]–[12], which can be automatically trained. We recently showed in preliminary work [7] that one such method [10] achieves performance close to full fine-tuning in many cases with a fraction of the parameter updates (less than 1%) required for full fine-tuning. This preliminary work has several shortcomings: it only includes a single LLM and PEFT method, and does not discuss the computational costs of the PEFT methods, which are close to the cost of full fine-tuning, as their training still needs to backpropagate errors through the full model. We address these shortcomings in this vision paper with the following contributions.

Contribution 1: *Extended study on parameter-efficient fine-tuning of LLMs for data wrangling (Section II).* We evaluate four popular PEFT methods and three baselines on differently sized LLMs for the ten benchmark tasks from [6], and additionally measure the training and inference times. We find that the PEFT methods can achieve performance on-par with full fine-tuning, and that we can leverage small LLMs with negligible performance loss. However, even though the PEFT methods are very parameter-efficient, they still incur high compute costs at training time.

Contribution 2: *Vision for zero-shot entity matching (Section III).* We explore a zero-shot setting to further reduce deployment costs. We find experimental evidence for the transferability of learned model adaptations for entity matching in such a setting, if they are pretrained on a suitable dataset. Based on this, we propose a vision for ZEROMATCH, a novel approach to zero-shot entity matching, based on maintaining a large number of pretrained LLM variants from different domains and intelligently selecting an appropriate model at inference time.

Contribution 3: *Code and results for reproducibility.* We provide our code and experimental results at https://github.com/Jantory/cpwrangle

TABLE I: Benchmark datasets from [6] with their corresponding task and domain.

| Task | Dataset | Domain | #Samples |
|---|---|---|---|
| Entity matching | Beer | food | 450 |
| | iTunes-Amazon | music | 539 |
| | Fodors-Zagats | food | 946 |
| | Walmart-Amazon | electronics | 10,242 |
| | Amazon-Google | software | 11,460 |
| | DBLP-ACM | citation | 12,363 |
| | DBLP-Google | citation | 28,707 |
| Error detection | Hospital | healthcare | 19,000 |
| Data Imputation | Buy | electronics | 651 |
| | Restaurant | address | 864 |

## II. Experimental Study

We conduct an extensive empirical study to evaluate the prediction quality, parameter efficiency and computational efficiency of various fine-tuning techniques and LLMs for data wrangling. In-line with previous work [7], we only include automatable methods which do not require manual interventions. Hence, we do not, for example, perform manual prompt engineering to select few-shot samples.

**Experimental setup**. We employ four popular PEFT methods from HuggingFace [13] to fine-tune models with fewer parameter updates:

- *Prompt-Tuning* [12] – solely focuses on the model input by prepending soft prompts (virtual tokens).
- *P-Tuning* [11] – learns a template filled with virtual tokens to which the original input is adapted.
- *Prefix-Tuning* [10] – prepends learnable virtual tokens to both the input and the output of each transformer layer.
- *LoRA* [9] – learns a low-rank decomposition of the weight update for fine-tuning.

*Datasets and tasks*. Following [7], we use the ten benchmark datasets from [6] originating from various domains, as detailed in Table I. The tasks are entity matching on seven datasets (where the goal is to identify equivalent entities within different tables), error detection on one dataset (where the goal is to detect errors in cells of a table) and data imputation in two datasets (where the goal is to impute missing values in a column). We use the predefined train/test splits and measure the F1 score for entity matching and error detection, and accuracy for data imputation.

*Models and hyperparameters*. We use Google T5 [14], an encoder-decoder model that unifies several text-related tasks into a common task with the same learning objective. In particular, we employ three variants of T5: T5-small (60.5M parameters), T5-base (223M parameters) and T5-large (738M parameters). A single GTX 1080 ti GPU with 11GB of memory is used for training and inference.

Due to the large number of combinations between models, PEFT methods and datasets, we perform hyperparameter search only once on the challenging Amazon-Google dataset and the medium-sized T5-base model, and subsequently apply the discovered values to the other experimental configurations. We use a learning rate of 0.2 for most methods, except for LoRA, which uses a learning rate of 0.001. We set both the rank and scaling factor to eight for LoRA. We utilize 50 virtual tokens for both P-Tuning and Prefix-Tuning, whereas Prompt-Tuning uses 60 virtual tokens. We train the T5-small and T5-base models for 200 epochs with a batch size of 16, except for DBLP-ACM, where we train T5-base for 100 epochs with a batch size of 12 only, due to memory constraints. The T5-large models are trained for 100 epochs with a batch size of eight, however in the case of DBLP-ACM and DBLP-Google, we train them for 50 epochs only with a batch size of four. We cannot fully fine-tune T5-large unfortunately due to memory issues, which even occur with an additional GPU.

*Baselines*. We compare our PEFT models against three automatable baselines:

- The commercial GPT-3 model (with 175B parameters) from OpenAI with zero-shot prompting as evaluated in [6].
- The AutoML library autogluon [15] from Amazon Research, for which we model our tasks as tabular classification problems.
- Fully fine-tuned versions of T5-small and T5-base.

**Results for prediction quality**. We detail the F1 and accuracy scores of our study in Table II. The symbol '$\star$' denotes that the results are sourced from published papers, bold values highlight the best performance achieved across all PEFT models on a specific dataset, and underlined values indicate the second best result. We find that PEFT methods drastically outperform the zero-shot GPT-3 baseline, even when using the T5-small model, which has only 60.5M params (about 3,000 times less then GPT-3). The LoRA method on T5-small achieves a mean performance of 90.96, compared to only 66.71 for GPT-3. The PEFT methods also drastically outperform the AutoML baseline as well, which reaches a mean score of 76.88 only.

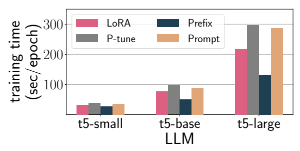
Among the PEFT methods, LoRA provides the highest quality (with the strongest performance in six out of ten datasets), while Prefix-Tuning also shows strong performance in several cases. We find that applying PEFT methods to T5-large provides higher mean performance than for the smaller T5-base and T5-small models, however the differences are minor (92.24% compared to 92.3% and 90.96%), even though T5-large has an order of magnitude more parameters than T5-small. For T5-small and T5-base, the LoRA method even outperforms full fine-tuning by a small margin, confirming results observed for other tasks [9].

**Results for training and inference time**. Next, we evaluate the computational costs for training and inference with a single a single GTX 1080 ti GPU with 11GB of memory.

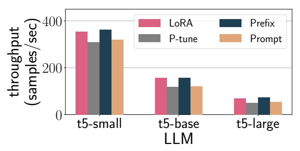
*Training time*. We measure the training time for all PEFT methods and LLMs. We find that the relative performance of the PEFT methods is similar across all datasets, and plot results for Amazon-Google as a representative example in Figure 1(a). There, P-Tuning and Prompt-Tuning are the slowest methods with up to 249 seconds per epoch on T5-large, LoRA

TABLE II: F1/accuracy scores of various PEFT methods for ten data wrangling benchmark datasets. Bold scores indicate the highest performance for a dataset, underlined scores indicate the second-best result.

| LLM | Method | Parameter Updates | Beer | iTunes-Amazon | Fodors-Zagats | Walmart-Amazon | Amazon-Google | DBLP-ACM | DBLP-Google | Hospital | Buy | Restau-rant | Mean Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPT-3 (175B) | Zero-shot ⋆ | - | 78.60 | 65.90 | 87.20 | 60.60 | 54.30 | 93.50 | 64.60 | 0.07 | 84.60 | 70.90 | 66.71 |
| - | AutoML | - | 72.00 | 76.60 | 81.63 | 39.21 | 58.12 | 97.41 | 91.13 | 98.63 | 78.46 | 75.58 | 76.88 |
| T5-small (60.5M) | Prompt | 48K | 89.66 | 91.53 | **100.00** | 75.75 | 71.26 | 98.87 | 95.26 | 75.61 | 87.69 | 33.72 | 81.94 |
| | P-tune | 212K | 78.58 | 90.20 | 95.24 | 63.80 | 67.62 | 98.75 | 94.89 | 94.07 | 83.08 | 34.88 | 80.11 |
| | Prefix | 309K | 69.57 | 88.13 | 97.78 | 0.00 | 49.14 | 92.84 | 91.79 | 33.39 | 92.31 | 61.63 | 67.66 |
| | LoRA | 296K | **93.33** | 96.30 | 97.67 | 79.19 | 72.46 | 98.76 | 95.26 | 94.84 | 92.31 | 89.53 | 90.96 |
| | Fine-tune | 60,500k | 83.87 | 96.30 | 97.67 | 79.56 | 69.41 | **98.99** | 95.01 | 98.00 | 92.31 | 88.37 | 89.95 |
| T5-base (223M) | Prompt | 67K | 72.00 | 93.10 | 90.00 | 76.50 | 70.23 | 98.44 | 95.36 | 96.67 | 86.15 | 33.72 | 81.22 |
| | P-tune | 312K | 77.78 | 88.52 | **100.00** | 81.08 | 69.92 | 98.32 | 95.52 | 95.44 | 86.15 | 58.13 | 85.09 |
| | Prefix | 914K | 85.71 | 90.20 | **100.00** | 68.50 | 65.73 | 97.21 | 94.67 | 91.23 | 92.30 | 59.30 | 84.49 |
| | LoRA | 892K | 90.32 | 96.15 | **100.00** | **86.39** | 72.49 | 98.20 | 95.63 | 95.44 | 93.84 | **91.86** | 92.03 |
| | Fine-tune | 223,000K | **93.33** | 91.23 | 97.67 | 77.69 | 68.91 | 98.53 | 93.98 | 94.97 | **95.38** | **91.86** | 90.36 |
| T5-large (783M) | Prompt | 74K | 75.86 | 78.12 | 97.67 | 86.16 | **76.43** | 98.32 | **95.93** | 91.62 | 83.08 | 37.20 | 82.04 |
| | P-tune | 369K | 0.00 | **96.42** | 78.04 | 86.23 | 73.89 | 93.77 | 80.90 | 95.20 | 86.15 | 75.58 | 76.62 |
| | Prefix | 2,435K | 86.67 | 94.33 | **100.00** | 82.09 | 72.27 | 96.16 | 94.19 | 95.20 | 92.30 | 73.26 | 88.65 |
| | LoRA | 2,362K | **93.33** | 96.30 | **100.00** | 82.64 | 74.21 | 96.80 | 95.11 | **99.45** | 93.84 | 90.70 | **92.24** |



(a) Training time per epoch on Amazon-Google.



(b) Mean inference throughput over all datasets.

Fig. 1: Training time per epoch and mean inference throughput for different PEFT methods and LLMs.

is slighly faster than those (with 217 seconds/epoch on T5-large), while Prefix-Tuning turns out to be the fastest PEFT methods across all LLMs. With 133 seconds/epoch on T5-large, it is roughly twice as fast as the slowest PEFT methods. As expected, we encounter drastic differences between LLMs: tuning T5-small is 4.8-7.9 times faster than T5-large, and 1.8 to 2.6 times faster than T5-base. This is remarkable since the prediction quality differences are minor for some PEFT methods like LoRA.

Furthermore, our results highlight a drawback of the existing PEFT methods: they are designed for parameter efficiency (and need two orders of magnitude less parameters than full fine-tuning), but not for compute efficiency. Their training time is still comparably high, even the fastest method Prefix-Tuning is only twice as fast as full fine-tuning on T5-base for example. This is due to the fact that the PEFT methods still need to run backpropagation through all model layers.

*Inference throughput*. Data wrangling is usually applied to large datasets in offline scenarios, which means we are interested in maximising throughput. Therefore, we measure the inference performance with a single GTX 1080 ti for all PEFT methods and LLMs on all datasets. We randomly choose 500 test samples per dataset, leverage the maximum possible batch size (determined by increasing the batch size until we encounter memory issues), measure the processing time with PyTorch's benchmarking package, and repeat each run five times. Note that we could not apply PyTorch's JIT optimisation due to dynamic code in the PEFT implementations.

We compute the mean throughput in terms of samples per second across all runs of each PEFT method and plot the results in Figure 1(b). We find that Prefix-Tuning provides the highest throughput (more than 360 samples/second on T5-small), closely followed by LoRA. P-Tuning and Prompt-Tuning have a lower throughput, but the difference is relatively small in general. We find drastic differences for the LLMs: the inference throughput on T5-small is 5-6 times higher than on T5-large, and more than two times higher than on T5-base. This is remarkable since the prediction quality differences were minor for methods like LoRA.

**Take-aways**. In summary, we find that the PEFT methods can achieve on-par performance with full fine-tuning and that we can leverage the small variant of T5 with negligible performance loss. However, even though the PEFT methods are very parameter efficient, they still incur high compute costs at training time.

## III. Our Vision for ZeroMatch

The high training costs of PEFT methods for data wrangling (and connected to this, the requirement of labeled training data for a given target dataset) still make their deployment expensive. To further reduce the deployment costs of LLMs for data wrangling in challenging real-world settings (Section I), we would ideally like to apply the LLMs in a zero-shot setting where no training (data) is required. We explore such a setting in the following.

### *A. Evidence for the Transferability of Learned Adaptations*

Manually engineered prompts often work well across different datasets for the same task. Narayan et al. [6] for example use the same prompt template "*Are Product A and Product*

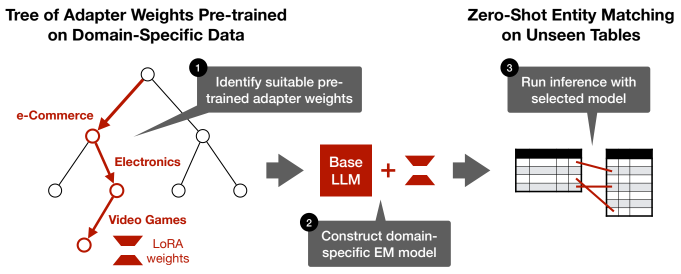

Tree of Adapter Weights Pre-trained on Domain-Specific Data
Zero-Shot Entity Matching on Unseen Tables
1 Identify suitable pre-trained adapter weights
3 Run inference with selected model
e-Commerce
Electronics
Base LLM
Video Games
LoRA weights
2 Construct domain-specific EM model

Fig. 2: Vision for ZEROMATCH, a novel approach to zero-shot entity matching, based on maintaining a large number of pre-trained LLM variants from different domains and intelligently selecting an appropriate model at inference time.

*B the same?*" together with different serialised few-shot samples for entity matching in the product-related Beer, Fodors-Zagats, DBLP-ACM and DBLP-Google datasets. Motivated by this observation, we empirically validate whether PEFT-tuned models also transfer well in a zero-shot setting.

**Experimental setup**. We choose the T5-base model and compute the prediction quality for each benchmark dataset in a zero-shot setting, using the PEFT model variants learned on the other datasets. We compare the resulting F1/accuracy scores to the score of the zero-shot performance of GPT-3 .

TABLE III: F1-scores for entity-matching in the challenging zero-shot setting, where no training data is available. We can always find an adapted version of T5-large that outperforms GPT-3 (which has two orders of magnitude more parameters).

| **Target** | **GPT-3** | **T5-base (223M)** | | | |
|---|---|---|---|---|---|
| **dataset** | **(175B)** | LoRA | pretrained on | Prompt | pretrained on |
| iTunes-Amazon | 65.90 | **94.73** | Beer | 91.52 | Walmart-Amazon |
| Beer | 78.60 | **93.33** | DBLP-Google | 87.50 | Walmart-Amazon |
| Fodors-Zagats | 87.50 | **100.00** | iTunes-Amazon | 97.67 | Walmart-Amazon |
| Walmart-Amazon | 60.60 | **62.92** | Beer | 45.51 | DBLP-Google |
| Amazon-Google | 54.30 | **62.75** | DBLP-Google | 61.85 | Walmart-Amazon |
| DBLP-ACM | 93.50 | 93.73 | DBLP-Google | **96.25** | DBLP-Google |
| DBLP-Google | 64.60 | **88.96** | DBLP-ACM | 81.34 | Walmart-Amazon |

**Results and discussion**. We observe that the transfer between tasks does not work: for example, applying adapted models from data imputation to entity matching or vice versa always results in an accuracy/F1 score of 0.0. The transfer does also not work between the two data imputation datasets, which originate from different domains.

However, we encounter surprising results with LoRA and Prompt-Tuning for the entity matching datasets, as detailed in Table III. There always exists a LoRA model which outperforms GPT-3 in this zero-shot setting (often by a drastic margin of more than 15%). In six out of seven cases, there is also a prompt-tuned model, which outperforms GPT-3 (often by more than 10%). This is a strong indication for the high performance potential of adapted T5 models for entity matching in the zero-shot setting, assuming that one can identify a suitable dataset for pretraining.

## *B. Efficient Zero-Shot Entity Matching with LLMs*

Our previous experiment shows the potential of transfering LoRA weights and soft prompts for zero-shot entity matching, if they are pretrained on a suitable dataset. Based on this insight, we formulate our vision for ZEROMATCH, illustrated in Figure 2, a novel approach to zero-shot entity matching with LLMs:

*(1) "A tree of pretrained adapter weights"*: As a preparatory step, we will pretrain different LoRA weights for a suitable base model on a variety of datasets from different domains, inspired by [16]. For that, we will leverage publicly available datasets, as well as data synthesized by a state-of-the-art LLM such as GPT-4 based on existing relational schemas [17]. We will our organise these weights in a tree structure, where each node retains adapter weights pretrained for a particular data domain, with its child nodes represent more detailed subdomains.

*(2) "Intelligent model selection at inference time"* At inference time, we will intelligently choose which LoRA weights to leverage for prediction. For that, we will consult our tree to identify the adapter weights most suitable the target dataset, e.g., by inspecting intermediate layer activations of the corresponding model. Furthermore, we will explore combining several models via ensembling.

Major challenges for this approach will be to ensure that the prediction quality is high enough and to minimise the negative impact on prediction throughput incurred by the overhead of having to choose an appropriate LLM variant.

## REFERENCES

[1] M. Stonebraker *et al.*, "Data integration: The current status and the way forward." *IEEE Data Engineering Bulletin*, 2018.
[2] Z. Abedjan *et al.*, "Detecting data errors: Where are we and what needs to be done?" *PVLDB*, 2016.
[3] S. Schelter *et al.*, "Automating large-scale data quality verification," *PVLDB*, 2018.
[4] F. Biessmann *et al.*, "Datawig: Missing value imputation for tables." *JMLR*, 2019.
[5] R. C. Fernandez *et al.*, "How large language models will disrupt data management," *PVLDB*, 2023.
[6] A. Narayan *et al.*, "Can foundation models wrangle your data?" *PVLDB*, 2022.
[7] D. Vos *et al.*, "Towards parameter-efficient automation of data wrangling tasks with prefix-tuning," *TRL@NeurIPS*, 2022.
[8] T. Addair, "Serving 100s of LLMs on 1 GPU with LoRAX," https://www.youtube.com/watch?v=i6zVvfvIFpc, 2023.
[9] E. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," *ICLR*, 2022.
[10] X. L. Li *et al.*, "Prefix-tuning: Optimizing continuous prompts for generation," *ACL*, 2021.
[11] X. Liu *et al.*, "GPT understands, too," *AI Open*, 2023.
[12] B. Lester *et al.*, "The power of scale for parameter-efficient prompt tuning," *EMNLP*, 2021.
[13] S. Mangrulkar *et al.*, "PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods," https://github.com/huggingface/peft, 2022.
[14] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, 2020.
[15] N. Erickson *et al.*, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," *arXiv preprint arXiv:2003.06505*, 2020.
[16] S. Gururangan *et al.*, "Don't stop pretraining: Adapt language models to domains and tasks," *ACL*, 2020.
[17] T. Döhmen *et al.*, "Gitschemas: A dataset for automating relational data preparation tasks," *DBML@ICDE*, 2022.