Towards Parameter-Efficient Automation of Data Wrangling Tasks with Prefix-Tuning

David Vos
TRL @ NeurIPS

INDE Lab, University of Amsterdam
David Vos, Till Döhmen, Sebastian Schelter
Contents

- Introduction of data wrangling tasks
- Data wrangling and large language models (LLMs)
- Prefix-tuning methodology
- Our contributions
- Experimental setup
- Results, conclusion and future research
Data Wrangling Tasks

Entity Matching (EM)

Error Detection (ED)

Data Imputation (DI)

The benchmark we use consists of 7 EM, 2 ED and 1 DI datasets

Given an entry, infer the missing value(s).
State of the art for data wrangling

Few-shot prompting GPT-3 achieves SoTA performance on 7/10 benchmarks [1].

1. **Format table rows using linearization**
2. **Select suitable prompt-samples (requires manual selection)**
   Randomly selecting results in an average drop of 14.7 F1 points
3. **Format a prompt**
   Minor modifications caused an average variance of 9.4 F1 points

Parameter-efficient and SoTA performance but hard to scale due to lack of automation and privacy concerns.

Is there an automatable and privacy-friendly alternative?

Maybe finetuning?
Finetuning can be performed for smaller LLMs like T5.

Format table rows using linearization

Update LLM weights with full finetuning procedure

Duplicate model weights for each new task

Automatable and more privacy-friendly with similar performance but hard to scale due to expensive model duplications.
Prompting vs. Finetuning

**Finetuning**
- **Automatable**
  Can be optimized using standard techniques
- **More privacy friendly**
  Can be optimized without looking at data
- **Less scalable**
  Requires a model copy for each new task

**Prompting**
- **Not easily automatable**
  Requires manual labour and expertise
- **Less privacy friendly**
  Requires manual inspection of training samples
- **Scalable**
  Scales to new tasks without extra parameters

Can we take the best of both prompting and finetuning?
Prefix-tuning?
Prefix-tuning learns a continuous prefix instead of engineering a discrete prompt.

LLM parameters are frozen.
Prefix-tuning

Prefix-tuning learns a continuous prefix instead of defining a discrete prompt [2].

- **Parameter-efficient**
  The prefix generation requires only 0.4% of the parameters required for finetuning.

- **Automatable and privacy friendly**
  Prefix parametrization can be trained end-to-end similar to finetuning.

An example for entity matching:

**Discrete prompt:** Are entity A and Entity B the same? entity A: <entity A>, entity B <entity B>

**Prefix-tuning:** entity A: <entity A>, entity B: <entity B>
Prefix-tuning

**Finetuning (updates all LLM parameters)**

```
Error detection
Data imputation
Entity matching
```

"<entity description A> <entity description B> Are entity A and B the same?"

```
Transformer activations
```

\[ h_i = \text{LM}_\phi(z_i, h_{<i}) \]

**Prefix-tuning (keeps LLM parameters frozen and updates the tiny prefix network)**

```
Error detection
Data imputation
Entity matching
```

"<entity description A> <entity description B> Are entity A and B the same?"

```
Transformer activations
```

\[ h_i = \begin{cases} 
\text{P}_\theta[i, :], & \text{if } i \in P_{\text{idx}}, \\
\text{LM}_\phi(z_i, h_{<i}), & \text{otherwise}.
\end{cases} \]

Figure based on Figure 1 from [2]: Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. ACL, 2021.
We take prefix-tuning to data wrangling

Parameter-efficient and automatable, perfect for big data environments

We compare prefix-tuning to finetuning T5. Prefix-tuning GPT-3 is currently impossible.
Table entries are serialized:

**Entity matching**

“Product A: \texttt{serialize(tuple-i)}. Product B: \texttt{serialize(tuple-j)}. Are product A and product B the same?”

**Error Detection**

“\texttt{serialize(tuple)} Is there an error in \textit{attribute-i: value-i}?”

**Data Imputation**

“\texttt{serialize(tuple)} \textit{attribute-i}?”
Experimental Setup

- T5-base implementation by Hugging Face
- Trained for 50 epochs or 100 if there is no clear convergence
- Experimented with different learning rates

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Domain</th>
<th>#Samples</th>
<th>Frac. Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity matching</td>
<td>Beer, iTunes-Amazon, Fodors-Zagats, Walmart-Amazon, Amazon-Google, DBLP-ACM, DBLP-Google</td>
<td>food, music, food, electronics, software, citation</td>
<td>450, 539, 946, 10,242, 11,460, 12,363, 28,707</td>
<td>15.1%, 24.5%, 11.6%, 90.6%, 10.2%, 25%, 18.6%</td>
</tr>
<tr>
<td>Error detection</td>
<td>Hospital</td>
<td>healthcare</td>
<td>19,000</td>
<td>2.7%</td>
</tr>
<tr>
<td>Data Imputation</td>
<td>Buy, Restaurant</td>
<td>electronics, address</td>
<td>651, 864</td>
<td>-</td>
</tr>
</tbody>
</table>
Experimental Setup

We compare prefix-tuning to two other methods:

1. **Zero-shot prompting GPT-3**
   Both methods are automatable, parameter-efficient and privacy friendly.

2. **Finetuning T5**
   Finetuning T5 is less scalable than prefix-tuning.
Table 3: Prefix-tuning drastically outperforms (trainingless) zero-shot prompting across all tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Metric</th>
<th>Prefix-tuning T5 (220M params)</th>
<th>Zero-shot prompting GPT-3 (175B params)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity matching</td>
<td>DBLP-Google</td>
<td>F1-score</td>
<td>0.9517</td>
<td>0.646</td>
</tr>
<tr>
<td>Entity matching</td>
<td>DBLP-ACM</td>
<td>F1-score</td>
<td>0.981</td>
<td>0.935</td>
</tr>
<tr>
<td>Entity matching</td>
<td>iTunes-Amazon</td>
<td>F1-score</td>
<td>0.9286</td>
<td>0.659</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Fodors-Zagats</td>
<td>F1-score</td>
<td>0.9767</td>
<td>0.872</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Beer</td>
<td>F1-score</td>
<td>0.8571</td>
<td>0.786</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Walmart-Amazon</td>
<td>F1-score</td>
<td>0.7961</td>
<td>0.606</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Amazon-Google</td>
<td>F1-score</td>
<td>0.6642</td>
<td>0.543</td>
</tr>
<tr>
<td>Imputation</td>
<td>Buy</td>
<td>Accuracy</td>
<td>0.9231</td>
<td>0.846</td>
</tr>
<tr>
<td>Imputation</td>
<td>Restaurant</td>
<td>Accuracy</td>
<td>0.8488</td>
<td>0.709</td>
</tr>
<tr>
<td>Error detection</td>
<td>Hospital</td>
<td>F1-score</td>
<td>0.9766</td>
<td>0.069</td>
</tr>
</tbody>
</table>
Table 2: Relative performance of prefix-tuning compared to finetuning on ten data wrangling tasks. In five out of ten cases, prefix-tuning is within 2.3% of the performance of finetuning, even though it leverages only 0.39% of the parameter updates required for finetuning the full model.

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Metric</th>
<th>Prefix-tuning</th>
<th>Finetuning</th>
<th>Rel. Perf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity matching</td>
<td>DBLP-Google</td>
<td>F1-score</td>
<td>0.9517</td>
<td>0.9552</td>
<td>99.6%</td>
</tr>
<tr>
<td>Entity matching</td>
<td>DBLP-ACM</td>
<td>F1-score</td>
<td>0.981</td>
<td>0.9876</td>
<td>99.3%</td>
</tr>
<tr>
<td>Error detection</td>
<td>Hospital</td>
<td>F1-score</td>
<td>0.9766</td>
<td>0.9912</td>
<td>98.5%</td>
</tr>
<tr>
<td>Entity matching</td>
<td>iTunes-Amazon</td>
<td>F1-score</td>
<td>0.9286</td>
<td>0.9455</td>
<td>98.2%</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Fodors-Zagats</td>
<td>F1-score</td>
<td>0.9767</td>
<td>1.000</td>
<td>97.7%</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Beer</td>
<td>F1-score</td>
<td>0.8571</td>
<td>0.8966</td>
<td>95.6%</td>
</tr>
<tr>
<td>Imputation</td>
<td>Buy</td>
<td>Accuracy</td>
<td>0.9231</td>
<td>0.9692</td>
<td>95.2%</td>
</tr>
<tr>
<td>Imputation</td>
<td>Restaurant</td>
<td>Accuracy</td>
<td>0.8488</td>
<td>0.8953</td>
<td>94.8%</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Walmart-Amazon</td>
<td>F1-score</td>
<td>0.7961</td>
<td>0.8806</td>
<td>90.4%</td>
</tr>
<tr>
<td>Entity matching</td>
<td>Amazon-Google</td>
<td>F1-score</td>
<td>0.6642</td>
<td>0.7436</td>
<td>89.3%</td>
</tr>
</tbody>
</table>
Conclusions

- Performance is within 2.3% of finetuning for five out of ten cases
- Prefix-tuning with T5 outperforms zero-shot prompting with GPT-3
- Prefix-tuning is an excellent option for large enterprise solutions

Future Research

- Scale prefix-tuning approaches to larger models (GPT-3 or alternatives like GPT-J or Bloom)
- Develop more advanced parametrization of the prefix
Any questions?

My contact details

✉ vos.dja@gmail.com
🌐 www.davidvos.dev
🐦 @dja_vos
LinkedIn davidjavos

Acknowledgements

Qualcomm
INDE lab

Ellis Unit Amsterdam
AR Amsterdam