LLM AAA: Making Large Language Models as Active Annotators

Ruoyu Zhang1,2, Yanzeng Li1, Yongliang Ma2, Ming Zhou2, Lei Zou1
1Wangxuan Institute of Computer Technology, Peking University, Beijing, China
2Langboat Technology, Beijing, China
{ry_zhang, zoulei}@pku.edu.cn, liyazeng@stu.pku.edu.cn,
{mayongliang, zhouming}@langboat.com

Abstract

Prevalent supervised learning methods in natural language processing (NLP) are notoriously data-hungry, which demand large amounts of high-quality annotated data. In practice, acquiring such data is a costly endeavor. Recently, the superior performance of large language models (LLMs) has propelled the development of dataset generation, where the training data are solely synthesized from LLMs. However, such an approach usually suffers from low-quality issues and requires orders of magnitude more labeled data to achieve satisfactory performance. To fully exploit the potential of LLMs and make use of massive unlabeled data, we propose LLM AAA, which takes LLMs as annotators and puts them into an active learning loop to determine what to annotate efficiently. To learn robustly with pseudo labels, we optimize both the annotation and training processes: (1) we draw k-NN samples from a small demonstration pool as in-context examples, and (2) we adopt the automatic reweighting technique to assign training samples with learnable weights. Compared with previous approaches, LLM AAA features both efficiency and reliability. We conduct experiments and analysis on two classic NLP tasks, namely entity recognition and relation extraction. With LLM AAA, task-specific models trained from LLM-generated labels can outperform their teacher LLMs within only hundreds of annotated examples, which is much more cost-effective than other baselines.

1 Introduction

Large language models (LLMs) have exhibited remarkable few-shot performance in a wide range of tasks, with only a few demonstrations and well-designed prompts (Brown et al., 2020; Ding et al., 2022; Liu et al., 2023). However, with rapid advancements comes vast potential risks in adopting LLMs for widespread downstream production applications. One of the main concerns is about data privacy and security. Under the prevalent “Language-Model-as-a-Service” (LMaaS, Sun et al., 2022) setting, users are required to feed their own data, potentially including sensitive or private information, to third-party LLM vendors to access the service, which increases the risk of data leakage (Lyu et al., 2020; Yu et al., 2022; Li et al., 2023). Besides, LLMs usually consume abundant tokens by continuous requests to APIs, where the marginal cost and latency become substantial in large-scale or real-time applications, hindering LLMs’ practical deployment in cost-sensitive scenarios (Goyal et al., 2020; Cao et al., 2023).

On the other hand, training task-specific models (TAMs) for NLP tasks necessitates extensive amounts of labeled data. Due to the superior generative capacity of LLMs, some researchers attempt to synthesize training data with text generation (Meng et al., 2022; Ye et al., 2022), as depicted...
in Figure 1. However, the generated text usually struggles with low-quality issues and may exhibit domain shifts with test data (Gao et al., 2023). To exploit the abundant unlabeled corpus, an alternative is to employ LLMs as annotators, which generate labels in a zero-shot or few-shot manner. While this approach seems promising, it is important to acknowledge that LLM-generated labels inevitably contain noise, especially when applied to challenging tasks and domain-specific data (Agrawal et al., 2022; Kazemi et al., 2023). Besides, larger models come with heavier expenses, and it is also crucial to reduce the annotation cost when the budget is restricted.

To enhance the reliability (i.e., accuracy) of TAMs’ performance as well as to ensure the data efficiency in annotation cost, we propose LLMAAA, an innovative framework that integrates active learning into the LLM annotation process, i.e., making LLMs as Active Annotators. By exploring different active acquisition strategies, LLMAAA enables the LLM to annotate more informative instances that benefit model performance more. To train TAMs reliably, we optimize both the annotation and training processes within LLMAAA framework. Firstly, we employ prompt engineering techniques to enhance LLMs’ performance by (1) selecting $k$-NN samples from a demonstration pool as in-context examples, and (2) building fine-level descriptions aligned with natural language for unnatural labels (e.g., category labels in the RE task). The valuable contextual information helps improve the quality of LLM annotations substantially. During training, we adopt the automatic reweighting technique (Ren et al., 2018) to assign learnable weights to the $silver^2$ training samples. This strategy allows the model to prioritize more informative and representative samples while simultaneously reducing the impact of noisy annotations.

We evaluate LLMAAA on two practical NLP tasks: named entity recognition (NER) and relation extraction (RE). Experiments show that: (1) with small-scale $gold$ data (~100 examples) serving for demonstration and validation, the trained TAMs can outperform their teacher LLMs within hundreds of $silver$ samples via LLMAAA; (2) our approach is significantly more data efficient compared to prevalent data generation methods, which usually require large-scale synthetic training data (size varying from 10k to 200k, Ye et al., 2022; Gao et al., 2023). These results confirm the potential of LLMAAA as a practical and cost-efficient solution to make LLMs as good annotators. The TAMs created through our framework offer advantages in terms of task-specific performance, data privacy, and inference costs, which release the capacity of LLMs for real-world productivity.

We summarize our contributions as follows:

- We propose LLMAAA, a framework to employ LLMs as annotators, featuring both efficiency and reliability.
- LLMAAA is capable to train TAMs that outperform teacher LLMs within hundreds of annotated samples, on classic NLP tasks like NER and RE.
- LLMAAA sheds light on the practical substitution of LLMs, with a cost-effective, privacy-ensured, yet well-performing solution.

2 Related Work

LLM and In-Context Learning Large language models (LLMs), usually pretrained on large-scale corpus to capture rich linguistic patterns and generate coherent text (Brown et al., 2020; Raffel et al., 2020; Chowdhery et al., 2022; OpenAI, 2023; Touvron et al., 2023), have shown remarkable performance in a wide range of NLP tasks (Min et al., 2021; Zhao et al., 2023). With the proposal of in-context learning (Brown et al., 2020), prompt engineering has been extensively explored to steer LLMs’ behavior for desired outcomes. These techniques design specific prompts or instructions to guide models’ outputs (Ding et al., 2022; Liu et al., 2023), either in rule-based (Shin et al., 2020) or learning-based (Lester et al., 2021) manners. Recent trend focuses on the strong reasoning capabilities of LLMs and enhances LLMs’ performance on complex task with chain-of-thought (CoT) prompting (Wei et al., 2023a). In general, prompt engineering improves the controllability and performance of LLMs in few-shot and zero-shot settings (Zhong et al., 2021), and enables LLMs to solve specific tasks, e.g., information extraction (Wei et al., 2023b; Wang et al., 2023).

Dataset Synthesis Supervised learning methods in NLP are often limited by high-quality annotated data. To address the bottleneck, researchers have explored techniques to synthesize training
data with LLMs, either by annotation or by generation. Following the first line of research, Feng et al. (2021); Chen et al. (2023) employ LLMs as unsupervised annotators to generate dialogue datasets. Recently, AnnoLLM (He et al., 2023) makes LLMs’ performance on par with crowdsource annotators by chain-of-thought prompting and self-generated explanations. As a contemporary work, Bansal and Sharma (2023) use LLMs for annotation in the domain transfer setting. Under the formulation of active learning, they propose a new metric, conditional informativeness, that works well with noisy labels. Among generation-based methods, Wang et al. (2021) first use LLM with few-shot prompts to generate training data. Schick and Schütze (2021) attempt to generate labeled text counterparts and text pairs for semantic textual similarity tasks. ZEROGEN (Ye et al., 2022) and SUNGEN (Gao et al., 2023) further extend this practice to zero-shot learning by training small models with zero-shot LLM-generated datasets. However, these approaches still suffer the low-quality and domain-shift issues of the synthetic data, and none of them consider the cost efficiency of data generation via LLMs.

**Labor Efficiency and Active Learning** Active learning is a technique proposed to minimize the annotation cost during the labeling process (Settles, 2009; Ren et al., 2021). A popular setting for active learning is the pool-based paradigm, which aims to select the most beneficial samples from an unlabeled data pool based on criteria including uncertainty (Lewis and Gale, 1994; Houlshby et al., 2011; Gal et al., 2017), diversity (Huang et al., 2010; Sener and Savarese, 2018), and hybrid objectives (Du et al., 2017; Yang et al., 2017; Ash et al., 2020; Margatina et al., 2021). The selected samples are annotated by human annotators and then added into the labeled dataset iteratively.

3 **LLM as Active Annotator**

To exploit LLMs’ superior few-shot performance and leverage abundant unlabeled data, we attempt to take LLM as annotator and train task-specific models for inference. An ideal process should be both efficient and reliable: we want to learn TAMs robustly with minimal LLM-generated labels.

Concretely, our solution is to make LLMs as **Active Annotator**. As shown in Figure 2, LLMAAA comprises three key components: (1) an LLM annotator that generates pseudo labels of given data, (2) an active acquisition mechanism for efficient data selection, and (3) an automatic reweighting technique to ensure robust learning with noisy labels. LLMAAA iterates the three stages to gradually produce stronger TAMs.

3.1 **Optimizing LLM as Better Annotator**

In-context learning (i.e. **Prompting**) enables LLM to conduct few-shot inference without fine-tuning. Given a manually-designed prompt $T(\cdot, \cdot)$, a demonstration set $S = \{(x^i, y^i)\}_{i=1}^k$ and the query example $x_q$, **Prompting** first builds a sentence $T(S, x_q)$, conditioned on which LLM then generates a text sequence

$$y_q = \arg\max_y P_{LM}(y | T(S, x_q)).$$

Finally, $y_q$ is mapped to the label space $\mathcal{Y}$.

Despite the decent abilities, previous studies show that the design of task-specific prompts has
a large impact on performance, varying between near state-of-the-art and random guess (Gao et al., 2021; Lu et al., 2022b). Finding the best prompts for given tasks and given data points is intractable. However, there are several principles turn out to be effective, compared with plain instruction.

**k-NN Example Retrieval** To select good in-context examples, Liu et al. (2022) propose a k-NN retrieval strategy, which first embeds the demonstration pool \(D_{\text{demo}}\) and query sample to vector representations, and then retrieves the nearest \(k\) neighbors of the query to form its exemplars. The rationale behind this is that semantically similar examples may help LLM answer the query better. Following their practice, we use Sentence-BERT (Reimers and Gurevych, 2019, 2020) to build the representations.

**Label Verbalizer** In classification tasks, the surface forms of labels may induce difficulties and ambiguities. Taking relation classification for instance, the label “per:parents” can indicate either “subject is the parent of object” or “object is the parent of subject”, depending on its definition. To address this problem, we utilize a label verbalizer to transform the surface forms to natural language descriptions with pre-defined templates (Sainz et al., 2021; Lu et al., 2022a), serving as fine-level guidance. The semantic templates we use are shown in Table 7.

### 3.2 Active Data Acquisition

Active learning (AL) seeks to reduce labeling efforts by strategically choosing which examples to annotate. We consider the standard pool-based setting, assuming that a large pool of unlabeled data \(D_{\text{pool}}\) is available. AL loop starts with a seed labeled set \(D_{\text{labeled}}\). At each iteration, we train a model \(M\) on \(D_{\text{labeled}}\) and then use acquisition function \(f(\cdot, M)\) to acquire a batch \(B\) consisting of \(b\) examples from \(D_{\text{pool}}\). We then query the LLM annotator to label \(B\). The labeled batch is then removed from the pool \(D_{\text{pool}}\) and added to labeled set \(D_{\text{labeled}}\), and will serve as training data for the next iteration. The process is repeated for \(t\) times.

Active acquisition strategies generally maximize either uncertainty or diversity. On one hand, uncertainty-based methods leverage model predictions to select hard examples. On the other hand, diversity-based methods exploit the heterogeneity of sampled data. We will cover some common strategies for thorough comparisons, and illustrate with classification task for simplicity\(^3\).

#### Random

We consider random selection as baseline, which samples uniformly from \(D_{\text{pool}}\). Typically pool data and test data share the same distribution, thus the sampled batch is expected to be i.i.d. with test data.

**Maximum Entropy** Entropy is one of the most widely used estimations of uncertainty (Settles, 2009). Data for which the model \(M\) has the highest entropy are sampled for annotation according to

\[
\text{argmax}_{\mathbf{x} \in D_{\text{pool}}} \left( \sum_{y \in Y} P_M(y|x) \log P_M(y|x) \right).
\]

#### Least Confidence

Culotta and McCallum (2005) propose to sort examples with the probability assigned by \(M\) to predicted class \(\hat{y}\), which samples

\[
\text{argmax}_{\mathbf{x} \in D_{\text{pool}}} \left( 1 - P_M(\hat{y}|x) \right).
\]

#### K-Means

Diversity sampling intends to select batches of data that is heterogeneous in the feature space. Following Yuan et al. (2020), we apply \(k\)-means clustering to the \(l_2\)-normalized embeddings of \(M\), and sample the nearest neighbors of the \(k\) cluster centers.

### 3.3 Robust Learning with Noisy Labels

LLM annotators inevitably produce noisy labels, especially with harder tasks and domain-specific data. To stay robust against training label bias, we adopt the automatic reweighting technique (Ren et al., 2018) to assign different weights to training examples adaptively.

We assume that a small-scale validation set \(D_{\text{val}}\) with clean labels (e.g. human annotations) is available throughout learning, with \(|D_{\text{val}}| \ll |D_{\text{pool}}|\). Concisely, automatic reweighting learns sample weights \(w\) by a meta-learning objective that minimizes validation loss w.r.t. \(w\), and uses online approximation to eliminate the nested loop of optimization. The training process of TAM is shown in Algorithm 1.

### 4 Tasks

We instantiate LLM\(\text{AAA}\) with two tasks: named entity recognition (NER) and relation extraction\(^4\).

---

\(^3\)Adaptation to other settings (e.g. sequence tagging) will be introduced in § 4.

\(^4\)We use BERT family as \(M\)'s encoder, and the embeddings refer to BERT output.
Algorithm 1: Automatic Reweighting

Input: Noisy data $D_{train}$, clean data $D_{val}$, batch size $n$, $m$, initial parameter $\theta_0$, step $S$

Output: Trained parameter $\theta_S$

for $s = 0, \ldots, S - 1$ do

$B_{train} \leftarrow \text{SampleBatch} (D_{train}, n)$
$B_{val} \leftarrow \text{SampleBatch} (D_{val}, m)$

$\{\hat{y}_{\text{train}}^{i} \}_{i=1}^{n} \leftarrow \text{Forward} (B_{train}, \theta_s)$

// build computation graph with automatic differentiation

$\epsilon \leftarrow 0$; $l_{\text{train}} \leftarrow \sum_{i=1}^{n} \epsilon \mathcal{L}(y_{\text{train}}^{i}, \hat{y}_{\text{train}}^{i})$
$\nabla \theta_s \leftarrow \text{Backward}(l_{\text{train}}, \theta_s)$
$\hat{\theta}_s \leftarrow \theta_s - \alpha \nabla \theta_s$
$\{\hat{y}_{\text{val}}^{i} \}_{i=1}^{m} \leftarrow \text{Forward}(B_{val}, \hat{\theta}_s)$

$l_{\text{val}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y_{\text{val}}^{i}, \hat{y}_{\text{val}}^{i})$
$\nabla \epsilon \leftarrow \text{Backward}(l_{\text{val}}, \epsilon)$

// truncate weights to zero, and normalize to one

$\hat{w} \leftarrow \max(-\nabla \epsilon, 0)$
$w \leftarrow \frac{\hat{w}}{\sum_{j} \hat{w}^j + 0 \sum_{j} w^j}$
$\hat{l}_{\text{train}} \leftarrow \sum_{i=1}^{n} w_i \mathcal{L}(y_{\text{train}}^{i}, \hat{y}_{\text{train}}^{i})$
$\nabla \theta_s \leftarrow \text{Backward}(\hat{l}_{\text{train}}, \theta_s)$
$\theta_{s+1} \leftarrow \text{OptimizerStep}(\theta_s, \nabla \theta_s)$

End for

5 Experiments and Analysis

5.1 Setup

Dataset We experiment with three different NLP datasets: Chinese OntoNotes 4.0 (Weischedel et al., 2011) and English CoNLL03 (Tjong Kim Sang and De Meulder, 2003) for NER, and Re-TACRED (Stoica et al., 2021) for RE. For Re-TACRED, we select a subset describing personal relationships and balance the NA relation instances to the original portion. Details of dataset statistics are described in Appendix A. We report the precision, recall, and micro F1 for both tasks.

Baselines We compare LLM-A for main experiments, and adopt BERT (Devlin et al., 2019; Cui et al., 2021) as TAM’s encoder. We also explore with other LLM annotators, GPT-3 (Brown et al., 2020) and GPT-4 (OpenAI, 2023), in § 6. We randomly sample 100 examples from the original validation sets as gold data, reusing the same set for demonstration $D_{\text{demo}}$ and validation $D_{\text{val}}$. We use the original training sets as $D_{\text{pool}}$ and randomly initialize seed labeled set $D_{\text{labeled}}$ with a size of 50 and acquire 50 samples per batch for 10 iterations, which generates 500 silver annotated samples in total. We generate 500 and 5,000 samples via ZEROGEN and FEWGEN for comparison. TAMs under all settings are trained three times with different random seeds, and we report the mean and standard deviation in the results. The training process and hyperparameters are detailed in Appendix B.

Model We use the same model architecture as Baldini Soares et al. (2019), which first encloses entity spans with special tokens [E] and \[\text{E}\], then encodes the sentence with BERT. The concatenated embedding of subject and object is fed into a linear classifier with activation for final prediction.
Table 1: Evaluation results for LLMs and other baselines across three different datasets, using ChatGPT as LLM backbone. We report the mean and standard deviation of three separate runs for each method. Since we set the temperature to 0 in PROMPTING, its results are deterministic and we only run evaluation once. We also denote the amount of data (gold/silver) that TAM used for training.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Data</th>
<th>Chinese OntoNotes 4.0</th>
<th>English CoNLL03</th>
<th>Re-TacRED-subset</th>
<th>Avg. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>P R F1</td>
<td>P R F1</td>
<td>P R F1</td>
<td>P R F1</td>
</tr>
<tr>
<td>PROMPTING</td>
<td>100 / -</td>
<td>67.72 74.02 70.73</td>
<td>79.18 83.59 81.33</td>
<td>64.21 86.68 73.77</td>
<td>75.28</td>
</tr>
<tr>
<td>SUPERVISED</td>
<td>100 / -</td>
<td>70.54 75.66 74.14 73.00 78.44</td>
<td>77.16 80.52 80.10</td>
<td>82.36 91.82 78.25 79.07</td>
<td>75.07</td>
</tr>
<tr>
<td>ZEROGEN</td>
<td>- / 5000</td>
<td>62.10 71.87 66.62 66.21 60.58</td>
<td>71.14 71.10 71.07 61.60 78.25 68.75</td>
<td>51.40 84.29 66.57 66.84</td>
<td>68.84</td>
</tr>
<tr>
<td>FEWGEN</td>
<td>100 / 500</td>
<td>71.78 71.09 66.71 71.35 72.35</td>
<td>73.06 69.87 71.13 67.21 77.84 71.97</td>
<td>69.21 71.26 69.20 71.07 74.68</td>
<td>74.68</td>
</tr>
<tr>
<td>LLMMAA-random</td>
<td>100 / 500</td>
<td>68.85 71.63 70.21</td>
<td>77.69 50.71 79.17 71.32</td>
<td>63.23 89.72 76.44 75.26</td>
<td>75.26</td>
</tr>
<tr>
<td>LLMMAA-confidence</td>
<td>100 / 500</td>
<td>72.62 75.49 74.00</td>
<td>82.91 72.93 74.10 74.40 78.20 68.70</td>
<td>71.10 92.24 71.26 75.68</td>
<td>74.68</td>
</tr>
</tbody>
</table>

Table 2: A case study of generated data with ZEROGEN on Re-TACRED. We leverage ChatGPT as the text generator, and the full prompts we use can be found in Appendix B.3.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Generated Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>per:parents</td>
<td>Mary\textsubscript{subj}’s father is Adam\textsubscript{obj}.</td>
</tr>
<tr>
<td></td>
<td>Tom\textsubscript{obj}’s mother, Mary\textsubscript{obj}, lives in New York.</td>
</tr>
<tr>
<td></td>
<td>Michelle Obama\textsubscript{subj}’s parents are Fraser C. Robinson III and Marian Shields Robinson\textsubscript{obj}.</td>
</tr>
<tr>
<td>per:children</td>
<td>Mike\textsubscript{subj}’s son is named Jack\textsubscript{obj}.</td>
</tr>
<tr>
<td></td>
<td>Lily\textsubscript{obj}’s children are Alex and Bella\textsubscript{obj}.</td>
</tr>
<tr>
<td></td>
<td>Sarah\textsubscript{subj} has a daughter named Emily\textsubscript{obj}.</td>
</tr>
</tbody>
</table>

We follow consistent principles in prompt design. Empirically, we find that in-context examples bring marginal benefit to RE, while label verbalizer is a technique specifically designed for the classification task. Therefore, we apply \(k\)-NN example retrieval to NER and label verbalizer to RE separately. We set \(k\) to 5 for all experiments, including FEWGEN. Refer to Appendix B.3 for full prompts.

5.2 Overall Results

Table 1 denotes our main experiment results. LLMMAA with least confidence as acquisition function outperforms all comparative baselines across all datasets, with 74.00%, 82.84% and 80.79% F1 scores on Chinese OntoNotes 4.0, English CoNLL03 and Re-TacRED-subset, respectively.

Comparing with PROMPTING (i.e. the LLM annotator), LLMMAA shows steady improvement (4% in average score) with TAMs of much fewer parameters and lower inference latency, indicating that LLMMAA provides a decent substitute for LLMs in real-world deployments. LLMMAA also surpasses SUPERVISED, where TAMs are trained on clean-labeled but smaller-scale data. This suggests that LLMMAA is capable of deriving rich knowledge beyond the limited demonstration/valuation set on unlabeled data, which benefits generalization.

We also notice that generation-based methods, i.e. ZEROGEN and FEWGEN, fail to establish on-par results, even with 10× more data in zero-shot setting. We argue that the text-generation abilities of LLMs are exaggerated in complex scenarios. To demystify the illusion, we devise a case study on Re-TACRED, as is shown in Table 2. ZEROGEN tends to generate simple templated sentences that deviate from the news domain, i.e. the original corpus of Re-TACRED. These results may induce low-quality and domain-shift issues that hamper TAMs’ performance. FEWGEN’s performance improves with in-context examples, however, it still lags far behind LLMMAA. In contrast, exploiting the unlabeled data effectively alleviates these problems with much higher efficiency, where only hundreds of annotated samples are sufficient for satisfactory performance.

5.3 Ablations

5.3.1 Effects of Prompt Engineering

Though ChatGPT can well follow human instructions in general, it still struggles with difficult tasks and domain-specific data. We compare the inference...
5.3.2 Accelerating with Active Learning

Figure 3 shows LLM AAA’s performance with different active acquisition strategies, shown by F1 scores. The dashed lines denote PROMPTING’s results. For each method, we report the mean and standard deviation of three runs initialized with different random seeds.

Figure 4: Results for analyzing the effects of automatic reweighting. We remove the online-approximated sample weights and train TAMs with standard loss objectives for ablation. The dashed lines denote PROMPTING’s performance. For each method, we report the mean and standard deviation of F1 scores within three different runs.

Table 3: Comparison results between plain instructions and optimized prompts in F1 scores.

<table>
<thead>
<tr>
<th></th>
<th>OntoNotes</th>
<th>CoNLL</th>
<th>Re-TacRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Instruction</td>
<td>49.62</td>
<td>55.74</td>
<td>70.94</td>
</tr>
<tr>
<td>+k-NN Examples</td>
<td>70.73</td>
<td>81.33</td>
<td>-</td>
</tr>
<tr>
<td>+Label Verbalizer</td>
<td>-</td>
<td>-</td>
<td>73.77</td>
</tr>
</tbody>
</table>

LLMs’ zero-shot performance is unsatisfactory. In addition, the label verbalizer can help align unnatural labels with natural language descriptions, which improves the performance in RE (from 70.94% to 73.77% in F1). These findings emphasize that prompt engineering is crucial for building strong annotators, and incorporating similar and aligned contexts contributes to better inference.
Table 4: Results on Chinese OntoNotes 4.0 for PROMPTING and LLMAAA with different LLMs. LLMAAA uses least confidence as the acquisition function, and annotates 500 samples for TAM training.

<table>
<thead>
<tr>
<th>Backbone Method</th>
<th>PROMPTING</th>
<th>LLMAAA-confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3</td>
<td>57.26</td>
<td>56.09</td>
</tr>
<tr>
<td>ChatGPT</td>
<td>72.66</td>
<td>75.49</td>
</tr>
<tr>
<td>GPT-4</td>
<td>73.47</td>
<td>76.42</td>
</tr>
</tbody>
</table>

Table 6.1 LLMAAA with Different Annotators

6.1 LLMAAA with Different Annotators

To guarantee the universal effectiveness of LLMAAA, we further investigate the performance with other LLM annotators, i.e. GPT-3 (Brown et al., 2020) and GPT-4 (OpenAI, 2023). Due to budgetary considerations, we opt to restrict our experiments to OntoNotes. The precision, recall and F1 score are shown in Table 4. The results indicate that LLMAAA benefits from better annotators with continuous improvements, and more importantly, TAMs trained by LLMAAA outperform the LLM annotators consistently. The student outperforms the weak teacher by a large margin (27% in F1 for GPT-3). As the teacher grows stronger, this gap narrows down. This trend meets our expectations: since student TAMs are trained with a fixed budget of data (500 samples), enhancing the capabilities of teacher LLMs will gradually approach the performance ceiling of the students. More annotation budget and more powerful TAMs can help extend this limit, while we leave the exploration for future research.

6.2 Why Can Students Outperform Teachers?

An interesting observation across our experiments is that student TAMs trained with generated labels can outperform teacher LLMs, i.e. LLMAAA > PROMPTING, even without sample reweighting, as shown by Figure 4. Such results partially align with previous findings in knowledge distillation (Wang, 2021; Song et al., 2021) and pseudo-label-based learning (Lee, 2013; Sanyal et al., 2022; Min et al., 2023), which share similar yet slightly different settings with LLMAAA.

We attempt to further explain the phenomenon in a simplified setting, where we consider a binary classification task that predicts $y$ for $x \sim D(x)$, where $D(x)$ is discrete as in language space. For simplicity, we let $y = 1$ denote the correct label and $y = 0$ otherwise. We first make the natural assumption that the teacher’s performance is above chance, i.e. the accuracy $p > 0.5$. Querying teacher for target sample $x_t$ will generate pseudo label $y_t \sim \text{Bernoulli}(p)$. If the student is a universal function approximator $S(x; \theta)$ that outputs a scalar as probability that $\hat{y} = 1$, then minimizing the...
cross-entropy loss
\[ \min \theta \mathbb{E}_{x_t \sim D(x), y_t \sim B(p)}[-y_t \log(S(x_t; \theta))
- (1 - y_t) \log(1 - S(x_t; \theta))] \]
will reach optimal with \( S(x; \theta) = p \). Usually we predict with heuristics that \( \hat{y} = 1 \) if \( S(x; \theta) > 0.5 \). With the previous assumption, we have \( \hat{y} = 1 \), which means that \( S \) always predicts correctly. This toy case nonetheless explains that an ordinary teacher can raise better students. Though teacher LLMs are deterministic for specific \( x \) when the temperature is set to 0, their predictions are yet statistically random in \( D(x) \), where the same conclusion holds.

We shall point out that the above discussion considers a much-relaxed setting, where we attempt to account for an intuitive understanding on why students outperform teachers in the hard label distillation problem. We leave the rigorous theoretical analysis for future work.

7 Conclusion

In this work, we propose LLMAAA, a framework that uses LLMs as active annotators to address the challenges of data scarcity in NLP tasks. With active learning strategies, LLMAAA allows LLMs to label more informative samples that promote TAMs’ performance efficiently. We also optimize for reliability within the framework, which uses prompt engineering techniques and automatic reweighting to improve annotation quality and to reduce the impact of noisy labels, respectively. Experiments on NER and RE tasks demonstrate the effectiveness of LLMAAA. The evaluation results highlight the efficiency and reliability of LLMAAA. Trained with just hundreds of LLM-annotated samples, TAMs are able to outperform their teacher LLMs substantially. Besides, LLMAAA is also much more efficient compared to prevalent data generation methods, which usually require orders of magnitude more synthetic training data. These findings reveal that LLMAAA offers a cost-effective, privacy-ensured, yet well-performing solution to apply LLMs in practical scenarios.

Limitations

Although LLMAAA demonstrates success in transferring and exceeding LLMs’ capabilities with cheaper TAMs, it does come with certain limitations. The main difference between the setting in LLMAAA and previous zero-shot generation-based methods, e.g., ZEROGEN and SUNGEN, is that we use an unlabeled data pool \( D_{\text{pool}} \) and oracle-annotated data \( D_{\text{demo}}/D_{\text{val}} \) to provide extra knowledge. However, we shall point out that unlabeled text is readily available in many real-world scenarios, thus it is practical to make the pool-based assumption. Additionally, in complex tasks where zero-shot inference fails (like NER in our experiments), it is costly yet necessary to incorporate demonstrations for LLMs. In LLMAAA, we strive for minimizing human efforts by restricting the oracle-annotated data to a small scale (100 samples), and exploiting the same data for demonstration and validation. Another bottleneck is the model capacities of teacher LLMs and student TAMs. On one hand, a weak teacher is unable to teach excellent students that are ready to be used for applications (e.g., GPT-3). On the other hand, TAMs are bounded depending on their architectures. When the teacher surpasses the ceiling, it will be theoretically impossible for students to outperform teachers. Despite these cases, we are optimistic that LLMAAA is effective in most situations.

We adopt the proprietary GPT family as annotators in experiments, which are provided by OpenAI in a black-box manner. Though powerful, this practice may raise several concerns, e.g., the potential exposure to test data. Nevertheless, we believe that given the comprehensive analysis in § 6.1, it does not affect the effectiveness of our method.

Ethics Statement

This work utilizes publicly available benchmark datasets, and we respect and adhere to their licenses and agreements. Our proposed method involves the use of LLMs for data annotation, as discussed in GPT3Mix (Yoo et al., 2021). This paradigm still poses several challenges, such as the potential biases or toxic content in the generated data. Therefore, it is crucial to exercise caution when employing our method to invoke LLMs for generating data and when utilizing TAMs trained on such generated data. Applying our work to downstream tasks such as NER and RE may result in issues such as mis-extraction and false information, and may fail in some cases. When employing our method, it is essential to consider using debiasing (Schick et al., 2021) or manual checking to mitigate these concerns.
Acknowledgements

We appreciate the anonymous reviewers for their valuable advice on this manuscript. We would like to thank Tianhao Wu for the insightful discussion and feedback. This work was supported by NSFC under grant 61932001 and U20A20174. The corresponding author of this paper is Lei Zou (zoulei@pku.edu.cn).

References


Parikshit Bansal and Amit Sharma. 2023. Large language models as annotators: Enhancing generalization of nlp models at minimal cost.


Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. Pumer: Pruning and merging tokens for efficient vision language models.


Mehran Kazemi, Sid Mittal, and Deepak Ramachandran. 2023. Understanding finetuning for factual knowledge extraction from language models.


Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022b. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Proceedings of the


Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding. In Advances in Neural Information Processing Systems.


Table 5: Statistics of relation types in each split of Re-TACRED-subset. We replace the label "per:origin" with "per:nationality" for clarity.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Training</th>
<th>Validation</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>per:age</td>
<td>421</td>
<td>256</td>
<td>208</td>
</tr>
<tr>
<td>per:nationality</td>
<td>295</td>
<td>222</td>
<td>115</td>
</tr>
<tr>
<td>per:parents</td>
<td>182</td>
<td>69</td>
<td>106</td>
</tr>
<tr>
<td>per:children</td>
<td>275</td>
<td>114</td>
<td>55</td>
</tr>
<tr>
<td>per:siblings</td>
<td>211</td>
<td>33</td>
<td>66</td>
</tr>
<tr>
<td>per:spouse</td>
<td>271</td>
<td>189</td>
<td>73</td>
</tr>
<tr>
<td>no_relation</td>
<td>2,482</td>
<td>1,324</td>
<td>934</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>4,137</td>
<td>2,207</td>
<td>1,557</td>
</tr>
</tbody>
</table>

The English CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003) is a NER dataset that contains four entity types: PER (person), LOC (location), ORG (organization), and MISC (miscellaneous entities), which consists of 14,041/3,250/3,453 sentences for training/validation/testing.

Re-TACRED (Stoica et al., 2021) is a revised version of TACRED (Zhang et al., 2017), a large-scale crowdsourced annotated RE dataset. It originally has 40 relation types. Including all these types will lead to much longer prompts, which may exceed the API length limit and receive responses with higher latency. Therefore, we opt to select a subset of relations that describe personal relationships for study. We keep all these relation instances in training/validation/testing sets, and balance the NA relation instances to the original portion. The statistics for each relation type is shown in Table 5.

For all three datasets, we randomly sample 100 examples from the original validation sets and reuse the same data for demonstration $D_{demo}$ and validation $D_{val}$. We use the full training sets as the initial $D_{pool}$, from which we randomly sample active learning’s seed labeled sets $D_{labeled}$ with a size of 50.

**B Implementations**

**B.1 LLM Inference APIs**

We access OpenAI APIs by Azure service. The API we use for each model is depicted in Table 6. Since ChatGPT and GPT-4 will continue to be updated, they may generate different responses as time changes, even when the temperature is 0.

<table>
<thead>
<tr>
<th>Model</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3</td>
<td>text-curie-001</td>
</tr>
<tr>
<td>ChatGPT</td>
<td>gpt-35-turbo</td>
</tr>
<tr>
<td>GPT-4</td>
<td>gpt-4</td>
</tr>
</tbody>
</table>

Table 6: Azure OpenAI service API that we use.

**B.2 Training Task-Specific Models**

For all experiments that train TAMs for inference (i.e. LLMAA, ZERGEN, FEWGEN and SUPERVISED), we repeat each with three random seeds, resulting in different parameter initialization and random data sampling. We report the mean and standard deviation in our results.

We use bert-base-cased (Devlin et al., 2019) as TAMs’ encoders with a learning rate of 5e-5 for English data (CoNLL03 and Re-TACRED), and chinese-bert-base-wwm (Cui et al., 2021) with a learning rate of 2e-5 for Chinese data (OntoNotes 4.0). The learning rate of other parameters (i.e. linear classifiers) is set to 1e-4. We optimize the models via AdamW (Loshchilov and Hutter, 2019), with $\epsilon = 1e-6$, under a linear warmup schedule for the first 6% steps. We train all TAMs with a batch size of 8 for 40 epochs and take the checkpoint with the highest validation performance for final prediction.

**B.3 Prompts**

The full prompts we use for annotation are shown in Table 7. In Re-TACRED, we provide prompts both with and without verbalized labels. To add demonstration, we insert each sample’s text into input and label to output. The target sample is added to the last input, and the last output is left blank for prediction.

We also show the prompts for generation in Table 8. We use them similarly to annotation. In the zero-shot setting, to help models generate desired outputs, we use a default example to inform LLMs about the output format.

**C Annotation Examples**

We show two annotation examples of correct/partially wrong annotations from the CoNLL 2003 NER dataset in Listing 1. The first example is exactly correct, and the second example contains hallucinations that do not exist in ground truth: "April", "March", and "Thursday".
Table 7: Annotator’s prompts for each task.

Listing 1: Annotation examples.
<table>
<thead>
<tr>
<th>Task</th>
<th>Prompting</th>
</tr>
</thead>
</table>
| CoNLL 03     | **Description**
You are an intelligent text data generator. Generate {} high-quality and diverse sentences in news domain containing entities for the following types:
[person (PER), location (LOC), organization (ORG), miscellaneous entity (MISC)]
Write one sample per line. No other output.

**Format**
Example: "Example: []"
Output: “Output: []”

**Default Example**
{"text": text, "entities": [{"name": name, "type": type}]}

| OntoNotes 4.0 | **Description**
你是一名新闻领域的文本生成助手。生成{}个流畅、通顺、多样的中文句子，并包含下面这些类型的命名实体（entity）：
[人名（PER），组织机构名（ORG），地缘政治实体（GPE），地理位置（LOC）]
每行输出一个样本，不要有任何额外的输出。

**Format**
Example: “示例： []”
Output: “输出： []”

**Default Example**
{"text": text, "entities": [{"name": name, "type": type}]}

| Re-TACRED     | **Description**
You are an intelligent text data generator. Generate {} high-quality and diverse sentences in news domain containing relational triplet for the following relation types:
- per:age : the age of SUBJ is OBJ
- per:parents : SUBJ’s parent is OBJ
- per:spouse : SUBJ’s spouse is OBJ
- per:siblings : SUBJ is the sibling of OBJ
- per:children : SUBJ’s children is OBJ
- per:nationality: SUBJ’s nationality is OBJ
- no_relation : SUBJ has no known relations to OBJ
Write one sample per line in json format. Subject and object must appear in the sentence. No other output.

**Format**
Example: "Example: []"
Output: “Output: []”

**Default Example**
{"text": text, "subject": subject, "object": object, "relation": relation}

---

Table 8: Generator’s prompts for each task.