Multistage Collaborative Knowledge Distillation from a Large Language Model for Semi-Supervised Sequence Generation

Anonymous ACL submission

Abstract

We study semi-supervised sequence generation tasks, where the few labeled examples are too scarce to finetune a model, and meanwhile, few-shot prompted large language models (LLMs) exhibit room for improvement. These tasks, such as parsing, are both expensive to annotate and under-represented in the LLM pretraining. In this paper, we present the discovery that the student model distilled from a few-shot prompted LLM can commonly generalize better than its teacher to unseen examples on such a task. Our analysis indicates that this happens because the student can learn a general pattern from the high-quality pseudolabels produced by the teacher, while helpfully not learning a pattern for the low-quality ones.

Leveraging this discovery, we propose a new method, Multistage Collaborative Knowledge Distillation from an LLM (MCKD), for these tasks. MCKD first few-shot prompts an LLM to produce pseudolabels for unlabeled data. Then at each stage of an iterative KD process, a new pair of students is trained on disjoint partitions of the pseudolabeled data, and produces new and improved pseudolabels for their unseen partitions. We show the effectiveness of MCKD on four syntactic and semantic parsing datasets. On CRAFT biomedical parsing, for example, 3-stage MCKD with 50 labeled examples outperforms an LLM teacher and vanilla KD by 7.5% and 3.7% parsing F1, respectively, and matches the performance of supervised finetuning with 500 examples.

1 Introduction

Low-resource tasks are common in real life, including within specialized domains, since data annotation often requires expert knowledge and incurs significant costs (Verspoor et al., 2012). Semi-supervised learning has been proposed as a solution when abundant unlabeled data are available (Blum and Mitchell, 1998; McClosky et al., 2006; Han et al., 2018; Kontonis et al., 2023). In a typical application, a model is trained on limited labeled data and produces pseudolabels for unlabeled data (McClosky et al., 2008; Amini et al., 2022). The pseudolabeled data are then filtered according to confidence thresholds and used to train a new model. In more extreme few-shot cases, labeled data are too scarce to finetune a model to begin with. Large language models (LLMs) offer a useful mechanism for synthesizing pseudolabels, thanks to their remarkable ability to learn *in context* from only a handful of demonstrations (Wang et al., 2021b; Yoo et al., 2021; Ding et al., 2022; Wang et al., 2023). Smaller and faster models can then be trained for these tasks using knowledge distillation (KD) from LLMs.

In this paper, we study a challenging semi-supervised sequence generation setting where labeled data are too few to finetune a model and few-shot prompted LLMs exhibit room for improvement. These will happen when the task is both expensive to annotate and under-represented in the pretraining of off-the-shelf LLMs. For example, it took 80 annotators around 2.5 years to parse 20k sentences of biomedical text in the CRAFT corpus (Verspoor et al., 2012). Meanwhile, pretrained LLMs do not always excel at tasks in specialized domains (Kung et al., 2022; Singhal et al., 2022) and tasks that involve specialized structured outputs, *e.g.*, semantic or syntactic parsing. The overarching research question of this paper is whether LLMs can still be leveraged in such scenarios to develop strong prediction models.

To this end, we examine knowledge distillation (KD) from a few-shot prompted LLM to a much smaller model. We observe that the student can commonly outperform its LLM teacher on **unseen evaluation data** from such tasks. Our analysis reveals that the student can learn a general pattern from high-quality pseudolabels from the teacher, while helpfully failing to capture low-quality ones in its learned general distribution due to their noisy

nature. This discovery is encouraging because it opens up the possibility for leveraging the KD students as teachers for further distillation.

Leveraging the discovery, we propose a novel method, Multistage Collaborative KD from an LLM (MCKD), for semi-supervised sequence generation. MCKD first collects pseudolabels for a large amount of unlabeled data from a few-shot prompted LLM, bootstraping a multistage KD process. At each KD stage, a new pair of students is trained on distinct partitions of the pseudolabeled data and asked to produce pseudolabels for the data that they have not been trained on. This process improves upon vanilla KD through two core mechanisms: (1) **collaborative** cross-partition labeling at each KD stage, which splits all pseudolabeled data into two mutually exclusive partitions and leverages the observed strong generalization capabilities of KD students to relabel them, and (2) **multistage** KD, whereby student models continue to generalize better and produce higher quality pseudolabels than their teachers over multiple iterations.

MCKD outperforms both the LLM teacher and relevant KD baselines in our evaluation, and is competitive with supervised finetuning with much more labeled data. On CRAFT constituency parsing, for example, MCKD with 50 labeled examples outperforms the LLM and vanilla KD by 7.5% and 3.7% F1, respectively, and matches supervised finetuning with 500 examples. On ATIS semantic parsing, MCKD with 50 labeled instances outperforms the LLM and vanilla KD by 7.2% and 2.3% F1, matching supervised finetuning with >750 examples.

The following is a summary of our contributions:

- We study if LLMs can be leveraged to train fast and accurate student models for semi-supervised sequence generation tasks where the LLM itself exhibits room for improvement when prompted with in-context learning examples. We find that KD students can often achieve better generalization than their LLM teachers (Section 5.1).
- We propose MCKD, a novel KD-based solution for such tasks. Data partitioning and cross-partition labeling enable MCKD to gradually improve the quality of pseudolabels over multiple stages of distillation, yielding increasingly better generations of students (Section 2).
- MCKD substantially outperforms prompted LLM, finetuned, and KD baselines on multiple low-resource tasks (Section 3-4).
- Further analysis shows that: (*a*) MCKD students correct many of the teacher's errors, and (*b*) MCKD scales well with increasing amounts of available unlabeled data (Section 5.2-5.3).

2 MCKD: Multistage Collaborative Knowledge Distillation from an LLM

2.1 Problem setup

Semi-supervised sequence-to-sequence generation tasks consist of few labeled examples $\mathcal{D}^{\text{labeled}} = \{x_i, y_i\}_{i=1}^{N_{\text{labeled}}}$ and many unlabeled examples $\mathcal{D}^{\text{unlabeled}} = \{x_i\}_{i=1}^{N_{\text{unlabeled}}}$ for training, where x_i and y_i are all sequences. We focus on a scenario in which (1) $\mathcal{D}^{\text{labeled}}$ is not large enough for training a capable prediction network via direct supervised finetuning, and (2) LLM few-shot prompting with demonstrations sampled from $\mathcal{D}^{\text{labeled}}$ exhibits room for improvement. This is a challenging but prevalent scenario that occurs when a task, such as parsing, is both expensive to annotate and under-represented in LLM pretraining.

2.2 Method

Figure 1 illustrates the MCKD algorithm. We detail the steps below.

LLM Prompting. We first sample examples from the labeled dataset $\mathcal{D}^{\text{labeled}}$ and then few-shot prompt an LLM with those to produce pseudolabels for unlabeled data $\mathcal{D}^{\text{unlabeled}}$.

Intermediate KD stages. Since we focus on tasks where few-shot prompted LLMs exhibit room for improvement, we consider the possibility that models finetuned on a sufficient amount of data, even if pseudolabeled by an LLM teacher or a model distilled from it, might be able to learn the task better. In Section 5.1, we show empirical evidence that this conjecture is indeed true. To be specific, we will present the discovery that, during KD from a few-shot prompted LLM or an intermediate model distilled from the LLM, (1) the student performance improves and approaches 100% F1 on teacher pseudolabels, and (2), meanwhile, the student performance on held-out evaluation data improves and surpasses that of the teacher.

To leverage the above discovery in MCKD, we perform data partitioning and propose **cross-partition labeling** with a collaborative pair of student models at each intermediate KD stage. Concretely, we partition unlabeled data $\mathcal{D}^{\text{unlabeled}}$ randomly and evenly into two distillation sets $\mathcal{D}_A^{\text{unlabeled}}$ and $\mathcal{D}_B^{\text{unlabeled}}$. Then, at each intermediate

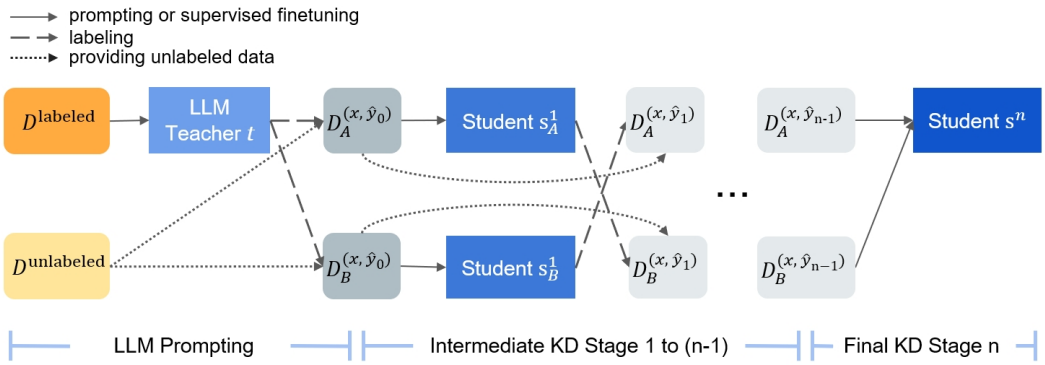


Figure 1: Overview of MCKD. **(1)** We use demonstrations from labeled data $\mathcal{D}^{\text{labeled}}$ to few-shot prompt an LLM teacher t to produce pseudolabels for unlabeled data $\mathcal{D}^{\text{unlabeled}}$. We partition $\mathcal{D}^{\text{unlabeled}}$ into $\mathcal{D}_A^{\text{unlabeled}}$ and $\mathcal{D}_B^{\text{unlabeled}}$, and let $\mathcal{D}_A^{(x,\hat{y}_0)}$ and $\mathcal{D}_B^{(x,\hat{y}_0)}$ denote the same partitions but with teacher-generated pseudolabels. **(2)** At the i-th intermediate KD stage, students s_A^i and s_B^i are trained on previously pseudolabeled data $\mathcal{D}_A^{(x,\hat{y}_{i-1})}$ and $\mathcal{D}_B^{(x,\hat{y}_{i-1})}$, respectively, and leveraged to label the other partitions $\mathcal{D}_B^{\text{unlabeled}}$ and $\mathcal{D}_A^{\text{unlabeled}}$ and produce $\mathcal{D}_B^{(x,\hat{y}_i)}$ and $\mathcal{D}_A^{(x,\hat{y}_i)}$, which will be used to train the next-stage student(s). **(3)** In the final KD stage, a single final student s^n is trained on both latest pseudolabeled partitions $\mathcal{D}_A^{(x,\hat{y}_{n-1})}$ and $\mathcal{D}_B^{(x,\hat{y}_{n-1})}$.

KD stage, we **(1)** train a collaborative pair of students using the mutually exclusive data partitions with pseudolabels generated by model(s) from the previous stage, which can be the LLM teacher or a pair of distilled previous-stage students, and **(2)** let each current-stage student produce pseudolabels for the partition that it has not been trained on, in order to gradually improve the quality of pseudolabels for $\mathcal{D}^{\text{unlabeled}}$ over stages.

Why do we need cross-partition labeling? Cross-partition labeling is key to the operation of MCKD. Since the student model almost perfectly fits its training data, i.e., pseudolabels produced by the previous-stage model(s), during KD (Section 5.1), letting the student label its own training partition would reproduce the same pseudolabels that it was trained on. Using those pseudolabels to train a next-stage student can only be expected to yield very little improvement, if any. We overcome this problem by partitioning the unlabeled data so that each student is asked to label a partition that it was not trained on.

The final KD stage. In the last distillation stage, a single student model is trained on the entire $\mathcal{D}^{\text{unlabeled}}$ with the latest pseudolabels.

2.3 KD mechanism

For sequence level knowledge distillation, the teacher can provide multiple types of supervision signals for the student, such as, token-level logits, greedy beams (hard pseudolabels), and beams that are close to the ground truth. Kim and Rush (2016) have observed that greedy beams are typically the most effective, although the other two may marginally further increase the student performance. In this work, we adopt greedy beams generated by the teacher for KD, following the literature on KD from LLMs (Wang et al., 2021b; Yoo et al., 2021; Ding et al., 2022; Li et al., 2023; Gilardi et al., 2023). MCKD can thus be used to distill both open-source and closed-source LLMs where only top-1 beams are provided.

3 Experimental Setup

3.1 Data and evaluation

Tasks and datasets. We perform experiments on two sequence prediction tasks: constituency parsing and task-oriented semantic parsing, where a general-purpose LLM pretrained to generate natural language text has to undergo distribution shifts. For constituency parsing, we use the Penn Treebank (PTB) dataset (Marcus et al., 1993) from the news domain and the Colorado Richly Annotated Full-Text (CRAFT) corpus from the biomedical domain (Verspoor et al., 2012). Semi-supervised learning for biomedical tasks is especially valuable since data annotation usually requires expertise and is formidably expensive.[1] For task-oriented semantic parsing, we use the Airline Travel Information Systems (ATIS) dataset (Tur et al., 2010) and the Snips dataset.[2] These datasets have been used for joint intent classification and slot filling. We fo-

[1] For example, it took 80 annotators around 2.5 years to parse the 20k sentences in the CRAFT corpus.

[2] https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines

cus on the latter that is a sequence generation task, letting models input natural langauge queries and output sequences of per-word slot labels. More details about the datasets are shown in Appendix A.

Simulating low-resource scenarios. For the constituency parsing datasets, PTB and CRAFT, we randomly sample 250 examples from the training set as the pool of labeled data. In different experiments, we randomly sample 50 or 250 examples from this pool to be the labeled set $\mathcal{D}^{\text{labeled}}$. For the task-oriented semantic parsing datasets, ATIS and Snips, we randomly sample 50 examples as the labeled set $\mathcal{D}^{\text{labeled}}$ since the datasets are smaller.

We use the remaining training examples, excluding the 250 above for constituency parsing and 50 for semantic parsing, as the pool of unlabeled data. For the main results (Section 4), we randomly sample 20k unlabeled examples for PTB and use the entire unlabeled pool for CRAFT, ATIS, and Snips as the set $\mathcal{D}^{\text{unlabeled}}$. We randomly and evenly partition $\mathcal{D}^{\text{unlabeled}}$ into $\mathcal{D}_A^{\text{unlabeled}}$ and $\mathcal{D}_B^{\text{unlabeled}}$. For further analyses (Section 5), the amount of unlabeled data randomly sampled from the pool will be described in the respective contexts.

Evaluation. We compute token-level F1-score for each example and average over examples to measure model performance. For constituency parsing, we follow common practice to use bracketing but discard constituent tags in training and evaluation, and remove punctuation before computing parsing F1.[3] We minimally post-process model outputs, balancing brackets and fixing accidental word segmentation.[4] For semantic parsing, we use the evaluation function from Goo et al. (2018).[5]

3.2 Model, learning, and prediction

The LLM teacher. We use GPT-3.5 Turbo[6], a model supported by the ChatGPT API, to collect initial pseudolabels for $\mathcal{D}^{\text{unlabeled}}$. We access the model through the OpenAI API. We randomly and uniformly sample 30 labeled parsing examples from $\mathcal{D}^{\text{labeled}}$ to include in the prompt, since 50 or 250 examples will exceed the allowed maximum context length. Details of prompt design can be found in Appendix C.

The student model. Our student models are finetuned from T5-Base (220M parameters; Raffel et al., 2020). Our assumed low-resource semi-supervised setting implies that the amount of available labeled data is small and a reliably large validation set does not exist. Inspired by classic unsupervised algorithms such as K-Means, we train each student until a convergence criteria is met — we train the student until the training accuracy on teacher pseudolabels changes by less than 0.1% for three consecutive epochs. Generation is performed via greedy decoding. The other hyperparameters are learning rate 3×10^{-4}, batch size 32, and 21 maximum training epochs.[7] Results are from single runs unless otherwise stated.

3.3 Baselines

We compare MCKD with the few-shot prompted LLM, T5-Base finetuned using few-shot labeled data $\mathcal{D}^{\text{labeled}}$, and the following KD baselines.

Vanilla KD. We employ sequence-level knowledge distillation (Kim and Rush, 2016) where a student is trained on hard labels generated by GPT-3.5 Turbo for the unlabeled dataset $\mathcal{D}^{\text{unlabeled}}$.

KD + SD w/ filtering. We first train a student with teacher pseudolabels on the entire unlabeled dataset $\mathcal{D}^{\text{unlabeled}}$ through vanilla KD. Then we apply self-distillation (SD): letting the student predict pseudolabels for $\mathcal{D}^{\text{unlabeled}}$, filtering them by keeping $r\%$ of the data where the student has the highest average token log-probabilities, and retraining a student on the filtered pseudolabeled data. Confidence-based filtering is a common method to optimize learning from pseudolabels (Lang et al., 2022; Wang et al., 2021a; Mohananey et al., 2020; Vinyals et al., 2015; McClosky et al., 2006). We note that KD + SD without filtering is similar to vanilla KD, since the student pseudolabels on $\mathcal{D}^{\text{unlabeled}}$ can be expected to be almost identical with teacher pseudolabels. We do not report a baseline that similarly filters LLM pseudolabels before

[3] Decisions associated with punctuation are often arbitrary in ground truth parse trees and not informative of model performance.

[4] We do not constrain model generation. Early during training, T5 fails to consistently output well-formed parse trees. This is less of an issue after the first few epochs.

[5] https://github.com/MiuLab/SlotGated-SLU/blob/master/utils.py

[6] https://platform.openai.com/docs/models

[7] Oracle experiments on PTB show that the validation performance doesn't change significantly given a wide range of these hyperparameters, and thus we use a default set in all experiments. In scenarios where a labeled validation set is available, further tuning may prove even more beneficial due to optimization behavior such as double descent (Opper et al., 1989; LeCun et al., 1991; Maddox et al., 2020).

we train a student, since the GPT-3.5 Turbo API access does not provide logits.

4 Main Results

Baseline performance. As shown in Table 1, GPT-3.5 Turbo's few-shot performance is reasonably good but has room for improvement. Due to the maximum input length for GPT-3.5 Turbo, we always sample 30 in-context demonstrations. In most cases, sampling them from a set of 50 of 250 labeled data $\mathcal{D}^{\text{labeled}}$ does not make a big difference.

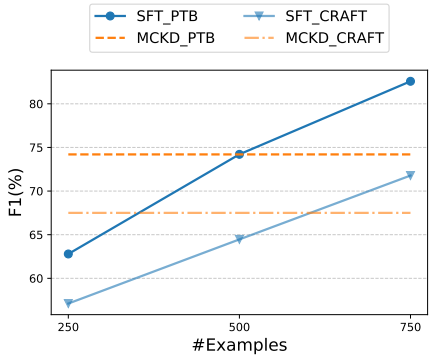
Supervised finetuning of T5-Base on $\mathcal{D}^{\text{labeled}}$ has the worst performance. This matches our assumption that the amount of available labeled data is too small to directly finetune a model. On both datasets for constituency parsing, 50 labeled examples completely fail to effectively finetune T5-Base to generate well-formed parse trees.

Vanilla KD yields students that outperform the teacher across all datasets. KD + SD with confidence-based filtering is shown to have limited performance improvement over vanilla KD and is subject to the selection of threshold for high-confidence labels for each case. For instance, keeping the 25% most confident pseudolabels gives the best performance for PTB with 50 labeled data, while keeping 75% performs the best for CRAFT with 50 labeled data. Selecting a proper threshold thus can be resource consuming and difficult in practice especially in low-data regimes with no labeled validation data.

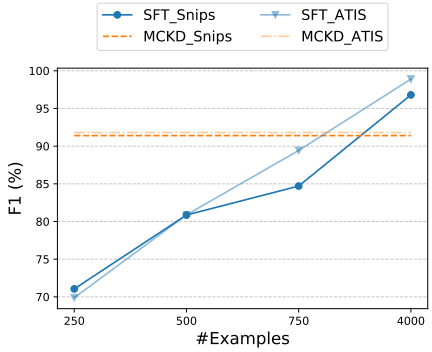
MCKD performance. Our MCKD approach substantially outperforms the few-shot prompted GPT-3.5 Turbo and KD baselines in all settings. For example, on CRAFT with 50 gold labeled training examples, 2-stage MCKD outperforms GPT-3.5 Turbo by 6.9% F1 and Vanilla KD by 3.1% F1. 3-stage MCKD further improves results, outperforming GPT-3.5 Turbo by 7.5% and Vanilla KD by 3.7%.

3-stage MCKD performs better than 2-stage MCKD consistently, but requires additional training time. The 4th-stage KD has marginal benefits in some settings. In practice, 2-stage MCKD is a good starting point and practitioners can increase the number of stages based on available resources.

Comparison with direct supervised finetuning. For constituency parsing, 3-stage MCKD using 50 labeled examples can match the performance of supervised finetuning (SFT) of T5-Base using 500 labeled examples on PTB, and outperform SFT using 500 labeled examples on CRAFT (Figure 2a). For task-oriented semantic parsing, 3-stage MCKD with 50 labeled examples can substantially outperform SFT on 750 labeled examples on both ATIS and Snips (Figure 2b). These results demonstrate the label-efficiency of MCKD and its effectiveness in semi-supervised sequence generation tasks where annotated data are scarce.



(a) Constituency parsing. The performance of MCKD using 50 annotations (shown in horizontal lines) is competitive with supervised finetuning (SFT) using 500 or more annotations on PTB and CRAFT.



(b) Task-oriented semantic parsing. The performance of MCKD using 50 annotations (shown in horizontal lines) outperforms supervised finetuning (SFT) with 750 annotations on ATIS and Snips.

Figure 2: The test F1 scores of supervised finetuned (SFT) models increase with more gold annotated data. MCKD needs much less labeled data to match the performance of SFT.

5 Analysis

This section presents additional analyses that motivate the design of MCKD and explore more deeply into why it performs well. All analyses are conducted on PTB data. For simplicity and without loss of generality, we examine 2-stage KD with a single student per stage; the takeaways can be generalized to multiple students and more stages which are advantageous in more applied settings.

Dataset	PTB		CRAFT		ATIS	Snips
N_{labeled}	50	250	50	250	50	50
GPT-3.5 Turbo	69.1	70.7	60.0	59.9	84.6	87.9
T5-Base	0	59.7	0	54.1	34.1	30.6
Vanilla KD	71.6	74.7	63.8	63.7	89.5	89.3
KD + SD	72.1	74.7	64.3	63.8	89.9	89.5
w/filter: keep top 75%	72.5	75.2	64.7	65.3	89.7	83.8
w/filter: keep top 50%	72.4	74.6	64.3	64.9	89.9	78.8
w/filter: keep top 25%	73.1	73.8	62.2	62.9	87.9	70.8
2-stage MCKD	73.9	76.6	66.9	67.5	91.4	90.9
3-stage MCKD	**74.2**	**76.9**	67.5	68.7	**91.8**	91.4
4-stage MCKD	**74.2**	**76.9**	**68.5**	**69.3**	**91.8**	**91.7**

Table 1: The test F1 on PTB, CRAFT, ATIS, and Snips datasets, where $N_{\text{labeled}} = 50$ or 250 gold annotated sentences are available. GPT-3.5 Turbo is accessed via an API using few-shot prompts; all other settings finetune T5-Base. MCKD outperforms baselines in all settings.

In each experiment, we sample subsets of $\mathcal{D}_A^{\text{unlabeled}}$ and $\mathcal{D}_B^{\text{unlabeled}}$ (see Section 3.1) and denote them as $\mathcal{D}_A$ and $\mathcal{D}_B$. We vary the sizes of $\mathcal{D}_A$ and $\mathcal{D}_B$ across experiments to study the effect of the distillation set size on performance. We train a stage-1 student s_1 using $\mathcal{D}_A$ with pseudolabels from an LLM teacher t and evaluate it on $\mathcal{D}_B$ against ground truth labels (assumed to be available only for analysis purposes and not during the training in MCKD). Then we train a stage-2 student s_2 using $\mathcal{D}_B$ with pseudolabels from s_1 and evaluate it on the test set.

5.1 A student can generalize better than its teacher

To understand the generalization properties of a prompted LLM teacher and its KD students, we first train student s_1 on 4k data $\mathcal{D}_A$ pseudolabeled by the LLM teacher t. We oberve that student model s_1 (1) approaches 100% F1-score on the pseudolabels of the teacher t, and (2) outperforms t on 4k held-out data $\mathcal{D}_B$ (Figure 3). The mean training F1 of s_1 on $\mathcal{D}_A$ plateaus around 97% while the median F1 is 100% as shown in the box plots, indicating that s_1 almost perfectly learns the pseudolabels of t, including potentially noisy ones. This does not weaken the generalization of s_1, however, as its performance continues to improve on $\mathcal{D}_B$ gold labels, outperforming t. In our experiments, this observation holds for various distillation set sizes (see Appendix D). Then we train a stage-2 student s_2 on 4k data $\mathcal{D}_B$ pseudolabeled by the stage-1 student s_1. We observe that the stage-2 student s_2 performance on the unseen test set reaches 72.47% F1 and surpasses that of the student directly trained by teacher pseudolabeled $\mathcal{D}_B$, which obtains 71.33% F1.

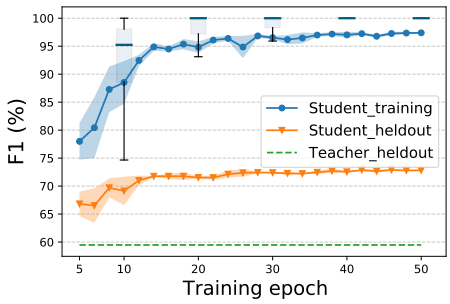


Figure 3: The held-out performance of the stage-1 student (F1 against gold annotations) improves with its training performance (F1 against teacher pseudolabels). Its held-out performance also surpasses that of the few-shot prompted LLM teacher.

Memorization vs. generalization. Why is the generalization of our student s_1 robust to the presence of noisy teacher pseudolabels? Feldman (2020) shows that the signal in a training example can either be internalized by a trained model as an instance of the general distribution that it learns or memorized separately. To find out if s_1 learns high and low-quality pseudolabels of teacher t differently in our context, we design the following experiment: We first identify the examples in the distillation set $\mathcal{D}_A$ where s_1 achieves a 100% F1 on pseudolabels of teacher t, *i.e.*, its training labels. From these well-learned training examples, we randomly select 100 *clean* ones on which pseudolabels of t have a 96.3% average F1 with respect to the ground truth labels, and denote them as $\mathcal{D}_{\text{clean}}$. We also select 100 *noisy* examples where pseudolabels of t have only a 31.8% average F1 with respect to ground truth, and denote them as $\mathcal{D}_{\text{noisy}}$. We then finetune a new T5-base student s_1' on the remaining data $\mathcal{D}_A - (\mathcal{D}_{\text{clean}} \cup \mathcal{D}_{\text{noisy}})$ and evaluate it on $\mathcal{D}_{\text{clean}}$ and $\mathcal{D}_{\text{noisy}}$ against pseudolabels of teacher t.

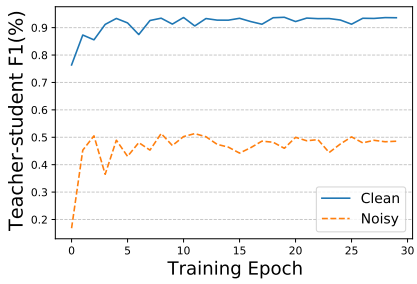


Figure 4: The student achieves a high F1 on held-out *clean* pseudolabels from an LLM teacher, but a low F1 on held-out *noisy* pseudolabels from the same teacher. The student tends to learn a general distribution over clean teacher pseudolabels while learning noisy ones via instance-wise memorization.

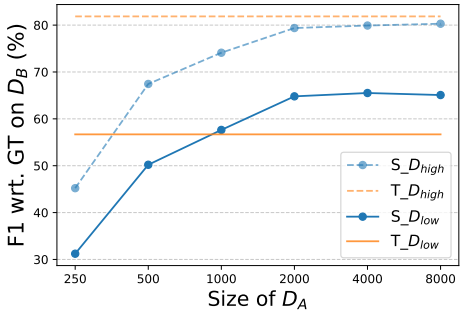


Figure 5: Performance (F1) on subsets of the held-out set $\mathcal{D}_B$ on which the teacher performs the best ($\mathcal{D}_{high}$) and the worst ($\mathcal{D}_{low}$). With sufficient amounts of pseudolabeled data, the student exhibits high agreement with the teacher on $\mathcal{D}_{high}$ while outperforming on $\mathcal{D}_{low}$.

As shown in Figure 4, s_1' obtains a high teacher-student F1 on $\mathcal{D}_{\text{clean}}$ but a low teacher-student F1 on $\mathcal{D}_{\text{noisy}}$. The original student s_1, with $\mathcal{D}_{\text{clean}}$ and $\mathcal{D}_{\text{noisy}}$ in its training data, was able to learn both sets well with a 97% teacher-student F1 in Figure 3. When both are removed from training (of s_1'), $\mathcal{D}_{\text{clean}}$ still resembles the general distribution learned by s_1' from the remaining training data, while $\mathcal{D}_{\text{noisy}}$ is now out-of-distribution. The s_1' results above thus confirm that our stage-1 MCKD student learns a general distribution over clean teacher pseudolabels while memorizing the noisy ones in a more instance-specific way. This explains why the generalization of the student is not affected by its learning of teacher errors in Figure 3.

5.2 The student corrects the teacher's errors

We have already observed the student s_1 trained on 4k instances of $\mathcal{D}_A$ pseudolabeled by the LLM teacher t to outperform t on the held-out set $\mathcal{D}_B$. To find out why, we split $\mathcal{D}_B$ evenly into two halves:

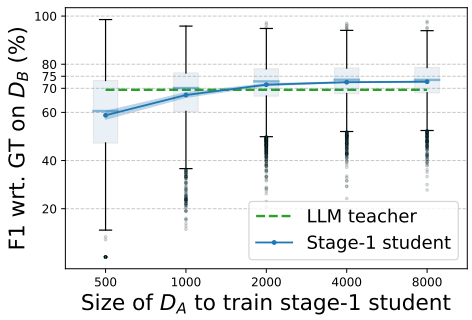


Figure 6: Performance (F1) of the stage-1 student on held-out set $\mathcal{D}_B$ when trained on varying amounts of pseudolabeled data from $\mathcal{D}_A$. With a sufficiently large distillation set, the student's F1 surpasses that of the teacher. We observe further improvements with multi-stage distillation (not shown in the figure).

the sentences on which t achieves the highest F1 ($\mathcal{D}_{high}$) and those on which it has the lowest F1 ($\mathcal{D}_{low}$). As illustrated in Figure 5, the performance of s_1 improves on both halves given more pseudolabeled training data $\mathcal{D}_A$. Given a large enough $\mathcal{D}_A$, the performance of s_1 approaches that of t on $\mathcal{D}_{high}$ and is better on $\mathcal{D}_{low}$. In other words, the student is able to correct the teacher's mistakes while almost completely retaining its true knowledge.

5.3 Large distillation sets are useful

Given an equal amount of labeled data $\mathcal{D}^{\text{labeled}}$, we investigate in this section if the stage-2 MCKD student s_2 can benefit from growing amounts of unlabeled data $\mathcal{D}^{\text{unlabeled}}$. First, Figure 6 shows that the performance of the stage-1 student s_1 on the held-out set $\mathcal{D}_B$ of 4k instances improves as the distillation set $\mathcal{D}_A$ grows larger. With a $\mathcal{D}_A$ of 2k or more instances, s_1 outperforms the few-shot prompted LLM teacher t on $\mathcal{D}_B$.

A stage-2 student s_2 is then trained similarly on $\mathcal{D}_B$ with pseudolabels from s_1 and evaluated on the test set. We examine the effect of the sizes of distillation sets $\mathcal{D}_A$ and $\mathcal{D}_B$ on the final test performance of s_2 in Figure 7. When the distillation set $\mathcal{D}_A$ for s_1 is too small, increasing the size of $\mathcal{D}_B$ by generating more pseudolabels with s_1 does not substantially improve performance for s_2, likely due to the pseudolabels being too noisy. Learning from an increasing amount of such noisy labels may even worsen the test set performance of s_2; for instance, with a $\mathcal{D}_A$ of 2k instances, enlarging $\mathcal{D}_B$ from 4k to 8k instances reduces the test F1 of s_2 from 71.3 to 69.0. On the other hand, when $\mathcal{D}_B$ is too small, making $\mathcal{D}_A$ larger alone does not improve the test

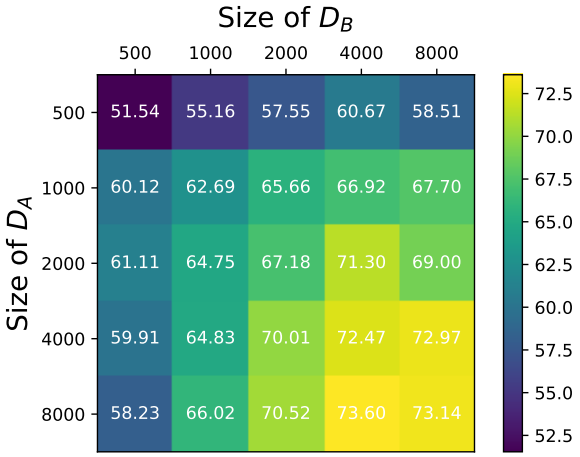


Figure 7: Test F1 against ground truth labels with varying amounts of stage-1 and stage-2 pseudolabeled data. The pseudolabels used in stage 1 ($\mathcal{D}_A$) are inferred by the teacher, and those used in stage 2 ($\mathcal{D}_B$) are inferred by the stage-1 student. It is generally better to use more data in both stages for stronger final performance, although performance can drop if the difference between the sizes of $\mathcal{D}_A$ and $\mathcal{D}_B$ is too large.

set performance of s_2 either, despite higher-quality pseudolabels in $\mathcal{D}_B$ from a better-trained s_1. Overall, the above results indicate that adequately large distillation sets are needed at every stage of MCKD to optimize the performance of the final student. In the original, more streamlined implementation of §2, we employ a pair of students at each stage to pseudolabel instances, so that the final student can take advantage of the most data available.

6 Related Work

Knowledge distillation (KD), apart from being used for model compression (Tang et al., 2019; Jiao et al., 2020; Bucila et al., 2006; Sun et al., 2020), has shown effectiveness in semi-supervised learning (Iliopoulos et al., 2022), where a teacher model annotates unlabeled data to train a student model. We study low-resource semi-supervised sequence generation with KD. We use hard pseudolabels from teachers to train student models, following common practice in sequence-level knowledge distillation (Kim and Rush, 2016), including scenarios when the teacher is an LLM (Ding et al., 2022; Yoo et al., 2021; Wang et al., 2021b; Gilardi et al., 2023; Shridhar et al., 2022; Ho et al., 2022; Li et al., 2023).

Self-distillation (SD) through training the model with its self-generated pseudolabels is often valuable in enhancing generalization, but finding reliable confidence thresholds to filter the pseudolabels and ensure their quality is often tricky (Furlanello et al., 2018; Liu et al., 2021). Instead, we improve the pseudolabel quality over stages by cross-partition labeling, taking advantage of the better generalization abilities of models over stages.

Some past works also leverage several student models to improve the distillation performance. Zhang et al. (2018) and Guo et al. (2020) introduce several student models that learn from each other's predicted logits. Song and Chai (2018) and Zhu et al. (2018) introduce extra branches of the student model (e.g., high-level layers or the classifier heads) to gather their multiple views as additional information for student training, which is shown helpful to improve the generalization. These works are mainly evaluated on image classification and do not analyze distillation from LLMs.

Co-training (Blum and Mitchell, 1998; Han et al., 2018; Lang et al., 2022) is relevant to our work since it involves using separate models to generate improved pseudolabels. In co-training, different learners are typically initialized with different views of the data. Then they provide confident predictions to each other for additional training. Wei et al. (2021) provides theoretical explanation on why generated pseudolables can be more credible than original labels for training. In contrast, our approach begins with an LLM teacher and trains better student models over stages.

7 Conclusion

Semi-supervised sequence generation in specialized low-resource settings can be challenging for NLP systems. This paper presents a novel solution to this problem that is able to leverage an LLM for training much smaller yet higher-accuracy models, despite the LLM itself performing only at a mediocre level. We first present the discovery that student models distilled from a few-shot prompted LLM can often correct the teacher's errors, thus showing superior generalization, on such tasks. Exploiting this phenomenon along with strategies such as data partitioning and cross-partition labeling over multiple stages, we then propose MCKD, a novel approach that utilizes knowledge distillation (KD) to train high-accuracy students. MCKD essentially leverages progressively better models to train future generations with even better performance. Extensive empirical evaluation verifies the effectiveness and label efficiency of MCKD.

8 Limitations

We have studied the scenario of semi-supervised sequence generation where labeled data are too scarce to train a model while a few-shot prompted LLM exhibits room for improvement. We therefore use an LLM as the initial teacher for knowledge distillation (KD) and follow common practice in sequence-level KD by using hard labels for KD. Future work may explore the effectiveness of the proposed multistage KD with cross-partition labeling (1) from a non-LLM teacher and (2) using soft logits or other mechanisms for KD.

The GPT-3.5 Turbo teacher model in our experiments is a company hosted service and many details are missing about how it was implemented and which data it was trained on. Relatively more information is available about T5-based. For instance, it was pretrained on large amounts of news data; this may explain why its performance on PTB is relatively higher than that on CRAFT.

References

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-training: A survey. *arXiv preprint arXiv:2202.12040*.

Avrim Blum and Tom M. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT' 98*.

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*.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Knowledge Discovery and Data Mining*.

Bosheng Ding, Chengwei Qin, Linlin Liu, Lidong Bing, Shafiq Joty, and Boyang Li. 2022. Is gpt-3 a good data annotator? *arXiv preprint arXiv:2212.10450*.

Vitaly Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *International Conference on Machine Learning*.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd-workers for text-annotation tasks. *arXiv preprint arXiv:2303.15056*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. 2020. Online knowledge distillation via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11020–11029.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Fotis Iliopoulos, Vasilis Kontonis, Cenk Baykal, Gaurav Menghani, Khoa Trinh, and Erik Vee. 2022. Weighted distillation with unlabeled examples. In *Advances in Neural Information Processing Systems*, volume 35.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327. Association for Computational Linguistics.

Vasilis Kontonis, Fotis Iliopoulos, Khoa Trinh, Cenk Baykal, Gaurav Menghani, and Erik Vee. 2023. Slam: Student-label mixing for distillation with unlabeled examples. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Tiffany H. Kung, Morgan Cheatham, ChatGPT, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, and Victor Tseng. 2022. Performance of chatgpt on usmle: Potential for ai-assisted medical education using large language models. *medRxiv*.

Hunter Lang, Monica N Agrawal, Yoon Kim, and David Sontag. 2022. Co-training improves prompt-based learning for large language models. In *International Conference on Machine Learning*, pages 11985–12003. PMLR.

Yann LeCun, Ido Kanter, and Sara A. Solla. 1991. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical review letters*, 66 18:2396–2399.

Zongjie Li, Chaozheng Wang, Pingchuan Ma, Chaowei Liu, Shuai Wang, Daoyuan Wu, and Cuiyun Gao. 2023. On the feasibility of specialized ability stealing for large language code models. *arXiv preprint arXiv:2303.03012*.

Yang Liu, Sheng Shen, and Mirella Lapata. 2021. Noisy self-knowledge distillation for text summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 692–703. Association for Computational Linguistics.

Wesley J. Maddox, Gregory W. Benton, and Andrew Gordon Wilson. 2020. Rethinking parameter counting in deep models: Effective dimensionality revisited. *ArXiv*, abs/2003.02139.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006*. The Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008*, pages 561–568.

Anhad Mohananey, Katharina Kann, and Samuel R. Bowman. 2020. Self-training for unsupervised parsing with PRPN. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 105–110. Association for Computational Linguistics.

Manfred Opper, J. Kleinz, H. Köhler, and Wolfgang Kinzel. 1989. Basins of attraction near the critical storage capacity for neural networks with constant stabilities. *Journal of Physics A*, 22.

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(1).

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2022. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*.

Guocong Song and Wei Chai. 2018. Collaborative learning for deep neural networks. *Advances in neural information processing systems*, 31.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170. Association for Computational Linguistics.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24. IEEE.

Karin Verspoor, Kevin Bretonnel Cohen, Arrick Lanfranchi, Colin Warner, Helen L Johnson, Christophe Roeder, Jinho D Choi, Christopher Funk, Yuriy Malenkiy, Miriam Eckert, et al. 2012. A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools. *BMC bioinformatics*, 13(1):1–26.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. *Advances in neural information processing systems*, 28.

Fusheng Wang, Jianhao Yan, Fandong Meng, and Jie Zhou. 2021a. Selective knowledge distillation for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6456–6466.

Ruida Wang, Wangchunshu Zhou, and Mrinmaya Sachan. 2023. Let's synthesize step by step: Iterative dataset synthesis with large language models by extrapolating errors from small models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11817–11831, Singapore. Association for Computational Linguistics.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021b. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics.

Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. 2021. Theoretical analysis of self-training with deep networks on unlabeled data. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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*, pages 38–45, Online. Association for Computational Linguistics.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328.

Xiatian Zhu, Shaogang Gong, et al. 2018. Knowledge distillation by on-the-fly native ensemble. *Advances in neural information processing systems*, 31.

...
Input:{input string of exemplar i}
Output:{output result of exemplar i}
...
Input:{input string of test data}
Output:

Table 2: The prompting format for LLM.

A Dataset Details

In Table 3, we show examples of the contituency and semantic parsing tasks that we work with. We use the training and testing sets described in 3.1. We assume a low-resource scenario where a labeled validation set is unavailable; we find default parameters that are not tuned on dataset-specific validation sets are effective across datasets. For PTB we adopt the standard data splits.

Constituency parsing. The standard split for the test set of PTB contains 2416 examples. For training, we randomly sample 20,000 examples as unlabeled training data. CRAFT contains 21,121 examples, but does not have standard splits; we randomly sample 50% examples as the training set and 40% examples as the test set. The remaining could be used for validation, although we assume a low-resource scenario where a labeled validation set is unavailable. For CRAFT, the test set contains 8448 examples, while the unlabeled training set contains 10560 examples.

Semantic parsing. For ATIS, the training set contains 4,478 utterances and the test set contains 893 utterances. There are 120 slot labels, i.e., labels for words. For Snips, the training set contains 13,084 utterances and the test set contains 700 utterances. There are 72 slot labels. We adopt dataset splits by Goo et al. (2018).

B Implementation Details

Hardware and software. Each experiment takes less than 6 hours on a single RTX 6000 GPU. Code is based on huggingface transformers (Wolf et al., 2020) and will be open-sourced.

C Prompting LLM for Pseudolabel Generation

We employ the standard random selection approach (Brown et al., 2020) to choose in-context exemplars from available labeled data to prompt the LLM. The prompt format is shown in Table 2.

For task-oriented semantic parsing, we use the following instruction for ATIS and Snips respectively: "Predict tag for each word in the sentence. 'O' for unrelated word. Focus on words related to the flight information, such as locations, time and airline."; "Predict tag for each word in the sentence. 'O' for unrelated word. Focus on words related to the music, restuarant, weather, book, playlist and searching." We also preprocess the output of in-context exemplars to interleave each predicted tag with its according slot word so as to improve the prompting performance for LLMs. For instance, for the example shown in Table 3, the output when prompting LLM is formatted as "list O the O fares O of O US B-airline_name Air B-airline_name flights from O Boston B-fromloc.city_name to O Philadelphia B-toloc.city_name".

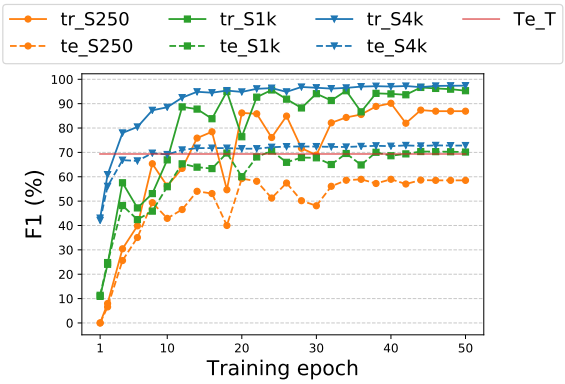


Figure 8: Student training F1 on teacher pseudolabels and student evaluation F1 on held-out data with ground truth labels using different distillation set sizes. E.g., "tr_S250" represents the training F1 of a student trained with 250 teacher pseudolabels, while "te_S250" is its generalization performance on held-out data. The generalization performance continues to improve when the student better learns teacher pseudolabels during training.

D Effects of Training Fidelity in Distillation Sets of Different Sizes

As discussed in Section 5.1, we find high training fidelity, can generally lead to a better student's performance on held-out data, despite the unreliability of teacher labels. This section investigates whether the observation still holds with different scales of the distillation set, especially when the size is substantially smaller. Results are shown in Figure 8. Similar patterns across different scales of distilla-

tion sets can be observed. Even if the distillation set only contains 250 teacher labels, the student can still generally have better performance on held-out data rather when the student memorizes more teacher labels.

Constituency parsing on PTB
Input: Some lousy earnings reports whacked the stock market but bond prices fell only slightly and the dollar rose a little against most major currencies.
Output: ((((Some) (lousy) (earnings) (reports)) ((whacked) ((the) (stock) (market)))) (but) (((bond) (prices)) ((fell) ((only) (slightly)))) (and) (((the) (dollar)) ((rose) ((a) (little)) ((against) ((most) (major) (currencies))))))

Constituency parsing on CRAFT
Input: Rather , the abnormal organization of Sertoli cells appears * to result from lack of Dmrt7 in the germ line.
Output: ((((Rather)) (,) (((the) (abnormal) (organization)) ((of) ((Sertoli) (cells)))) ((appears) (((*)) ((to) ((result) ((from) (((lack)) ((of) ((Dmrt7))) ((in) ((the) (germ) (line))))))))) (.)))

Semantic parsing on ATIS
Input: list the fares of US Air flights from Boston to Philadelphia
Output: O O O O B-airline_name B-airline_name O O B-fromloc.city_name O B-toloc.city_name

Semantic parsing on SNIPS
Input: add sabrina salerno to the grime instrumentals playlist
Output: O B-artist I-artist O O B-playlist I-playlist O

Table 3: Examples from our used datasets.