LMTurk: Few-Shot Learners as Crowdsourcing Workers in a Language-Model-as-a-Service Framework

Mengjie Zhao[†] **Fei Mi**[‡] **Yasheng Wang**[‡] **Minglei Li**[*]
Xin Jiang[‡] **Qun Liu**[‡] **Hinrich Schütze**[†]
[†]CIS, LMU Munich [‡]Huawei Noah's Ark Lab [*]Huawei Technologies Co., Ltd.
`mzhao@cis.lmu.de, {mifei2,wangyasheng,jiang.xin,qun.liu}@huawei.com`

Abstract

Vast efforts have been devoted to creating high-performance few-shot learners, i.e., large-scale pretrained language models (PLMs) that perform well with little downstream task training data. Training PLMs has incurred significant cost, but utilizing the few-shot learners is still challenging due to their enormous size. This work focuses on a crucial question: How to make effective use of these few-shot learners? We propose LMTurk, a novel approach that treats few-shot learners as crowdsourcing workers. The rationale is that crowdsourcing workers are in fact few-shot learners: They are shown a few illustrative examples to learn about a task and then start annotating. LMTurk employs few-shot learners built upon PLMs as workers. We show that the resulting annotations can be utilized to train models that solve the task well and are small enough to be deployable in practical scenarios. Active learning is integrated into LMTurk to reduce the amount of queries made to PLMs, minimizing the computational cost of running PLM inference passes. Altogether, LMTurk is an important step towards making effective use of current PLMs.[1]

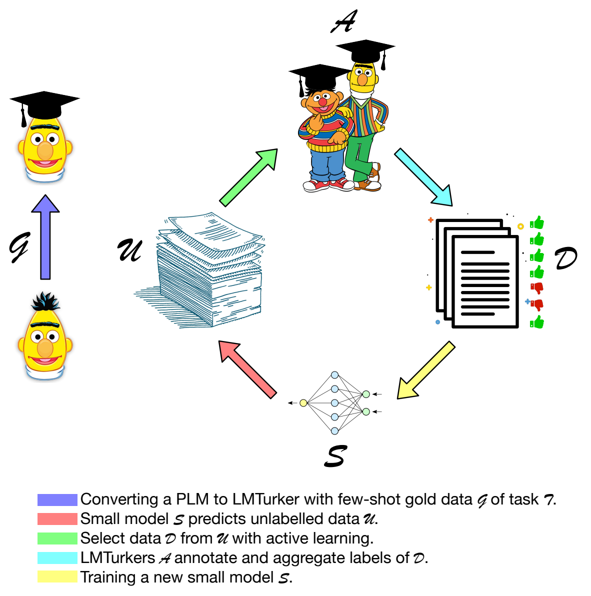
$\mathcal{A}$
$\mathcal{G}$
$\mathcal{U}$
$\mathcal{D}$
$\mathcal{S}$
🟦 Converting a PLM to LMTurker with few-shot gold data $\mathcal{G}$ of task $\mathcal{T}$.
🟥 Small model $\mathcal{S}$ predicts unlabelled data $\mathcal{U}$.
🟩 Select data $\mathcal{D}$ from $\mathcal{U}$ with active learning.
🟦 LMTurkers $\mathcal{A}$ annotate and aggregate labels of $\mathcal{D}$.
🟨 Training a new small model $\mathcal{S}$.

Figure 1: LMTurk overview; best viewed in color. We few-shot adapt PLMs to task $\mathcal{T}$ (left) and then use them as crowdsourcing workers in active learning. We show that these PLM workers are effective in training a small model $\mathcal{S}$ through a customized active learning loop (right). LMTurk is a novel way to take advantage of large-scale PLMs: It creates models small enough to be deployed in resource-limited real-world settings.

1 Introduction

Equipped with prolific linguistic features (Liu et al., 2019; Tenney et al., 2019; Belinkov and Glass, 2019; Rogers et al., 2020) and rich world knowledge (Petroni et al., 2019; Poerner et al., 2020; Kassner et al., 2021), large-scale pretrained language models (PLMs) have been shown to be versatile: They are now basic building blocks (Bommasani et al., 2021) of systems solving diverse NLP tasks in many languages (Wang et al., 2018, 2019; Hu et al., 2020; Xu et al., 2020; Khashabi et al., 2021; Park et al., 2021; Adelani et al., 2021).

Recent work shows that PLMs are effective *few-shot learners* (Brown et al., 2020; Schick and Schütze, 2021b; Gao et al., 2021; Tam et al., 2021) through *priming* (Brown et al., 2020; Tsimpoukelli et al., 2021) or *prompting* (Li and Liang, 2021; Liu et al., 2021b; Lester et al., 2021; Zhao and Schütze, 2021). Developing few-shot learners is crucial because current NLP systems require much more data than humans (Yin et al., 2020). Few-shot learners tend to perform well; however, they still fall behind systems trained with abundant data. Furthermore, the enormous size of PLMs hinders their deployment in practice. For example, it is challenging to fit the 11 billion T5-XXL (Raffel et al., 2020) model on a single regular GPU.

Our goal in this paper is to devise methods that make *more effective use of current few-shot learners*. This is crucial because an increasing number

[1]Resources are available at: `github.com/lmturk`

of gigantic few-shot learners are trained; how to use them effectively is thus an important question. In particular, we want an alternative to hard-to-deploy huge models. At the same time, we want to take full advantage of the PLMs' strengths: Their versatility ensures wide applicability across tasks; their vast store of knowledge about language and the world (learned in pretraining) manifests in the data efficiency of few-shot learners, reducing labor and time consumption in data annotation.

In this work, we propose **LMTurk**, **L**anguage **M**odel as mechanical **Turk**. Our basic idea (see Figure 1) is that, for an NLP task $\mathcal{T}$, *we treat few-shot learners as non-expert workers*, resembling crowdsourcing workers that annotate resources for human language technology. We are inspired by the fact that we can view a crowdsourcing worker as a type of few-shot learner: A few examples demonstrating $\mathcal{T}$ teach her enough about $\mathcal{T}$ to conduct effective annotation. For example, Snow et al. (2008) train workers with a few examples of annotating emotion; He et al. (2015) conduct short training sessions for workers before annotation; Lee et al. (2021) train workers with learning curricula.

Snow et al. (2008) pioneered crowdsourcing in NLP (Howe et al., 2006; Howe, 2008), motivated by the high cost of TreeBank annotation (Marcus et al., 1993; Miller et al., 1993). Crowdsourcing organizes human workers over the Web to annotate data. Workers need not be experts to be effective, resulting in reduced *per-label cost*. Active learning (Hachey et al., 2005; Felder and Brent, 2009) can be incorporated (Laws et al., 2011) to further decrease annotation cost, by lowering *the number of labels* to be annotated. LMTurk treats PLM-based few-shot learners as non-expert workers that produce training sets, which are then used to train a small machine learning model $\mathcal{S}$ specialized for $\mathcal{T}$. This scenario is analogous to active learning. We achieve two benefits: (i) low annotation cost because humans only need to annotate a few shots of data; (ii) solving practical NLP tasks with small models that are more real-world deployable.

LMTurk resonates with Laws et al. (2011)'s earlier idea of combining crowdsourcing and active learning. They consider human workers as "noisy annotators" while we explore the utilization of modern NLP few-shot learners (built upon machine learning models) as workers – which have the advantage of being free, instantly interactive, fast, responsive, and non-stopping.

Our **contributions**: (i) We propose LMTurk, a method that uses few-shot learners as crowdsourcing workers. Figure 1 shows the overview of LMTurk. (ii) We vary an array of important design choices, identifying strengths and weaknesses of LMTurk. (iii) Unlike much work on active learning in a synthetic oracle setting, we develop methods for handling the varying quality of annotation that does not come from an oracle. (iv) We extensively evaluate LMTurk on five datasets, showing that LMTurk can guide a small model $\mathcal{S}$ to progressively improve on $\mathcal{T}$. $\mathcal{S}$ can then be deployed in practical scenarios. (v) This is the first work showing that few-shot learners give rise to effective NLP models through crowdsourcing and active learning – with the benefits of low annotation cost and practical deployability.

2 Related Work

Few-shot learners in NLP. Significant progress has been made in developing (Devlin et al., 2019; Peters et al., 2018; Yang et al., 2019; Brown et al., 2020), understanding (Liu et al., 2019; Tenney et al., 2019; Belinkov and Glass, 2019; Hewitt and Liang, 2019; Hewitt and Manning, 2019; Zhao et al., 2020a; Rogers et al., 2020), and utilizing (Houlsby et al., 2019; Zhao et al., 2020b; Brown et al., 2020; Li and Liang, 2021; Schick and Schütze, 2021a; Lester et al., 2021; Mi et al., 2021a) PLMs. Brown et al. (2020), Schick and Schütze (2021a), and Liu et al. (2021b) show that PLMs can serve as data-efficient few-shot learners, through priming or prompting (Liu et al., 2021a). For example, GPT3 achieves near state-of-the-art performance on COPA (Roemmele et al., 2011) with only 32 annotated data.

However, little to no work discusses or explores the actual *practical utility* of these few-shot learners. We aim to develop effective methods of utilizing them in practical scenarios.

Crowdsourcing has a long history in human language technology (Alonso et al., 2008; Callison-Burch, 2009; Trautmann et al., 2020); specialized workshops were organized (Callison-Burch and Dredze, 2010; Paun and Hovy, 2019). It has numerous applications (Yuen et al., 2011), but we focus on its application as voting systems. To reduce *per-label* cost, crowdsourcing organizes non-expert human workers distributed across the Web for annotation, instead of employing linguistic experts (Jamison and Gurevych, 2015; Bhardwaj et al., 2019;

Nangia et al., 2021). Snow et al. (2008) show that averaging ten crowdsourced labels matches an expert-level label for recognizing textual entailment (Dagan et al., 2006). Paun et al. (2018) show that incorporating structure in annotation models is important. Measuring label disagreements is also crucial (Dumitrache et al., 2021).

LMTurk utilizes NLP few-shot learners as non-expert workers. The few-shot training data can be viewed as the examples shown to humans before annotating. The process is free, fast, responsive, and non-stopping.

Active learning (AL; Cohn et al. (1996); Settles (2009)) strives to reduce *the number of examples* to be annotated via identifying informative examples with acquisition functions. Settles and Craven (2008) evaluate AL algorithms for sequence labeling. Zhang et al. (2017); Shen et al. (2017); Siddhant and Lipton (2018) apply AL to deep neural networks. Simpson and Gurevych (2018) devise a scalable Bayesian preference learning method for identifying convincing arguments. Lee et al. (2020) propose to consider user feedback in AL systems. Ein-Dor et al. (2020) explore AL for BERT. Schröder and Niekler (2020) review text classification with AL. Liang et al. (2020); Margatina et al. (2021) integrate contrastive learning into AL. Zhang and Plank (2021) identify examples with datamap (Swayamdipta et al., 2020).

We incorporate AL in LMTurk to reduce the amount of examples to be annotated by PLMs, reducing the computational cost of running several inference passes. This contributes to a more environmentally friendly (Strubell et al., 2019; Schwartz et al., 2020; Patterson et al., 2021) scenario.

Perhaps closest to our work, Yoo et al. (2021) conduct data augmentation via priming GPT3 and Wang et al. (2021) mix human- and GPT3-annotated data, focusing on cost analysis. GPT3 is utilized in a Language-Model-as-a-Service form by OpenAI, which is not free.[2] Also, strategies of priming GPT3 may not generalize well to other PLMs. For example, priming strategies have to adapt to GPT3's maximum sequence length. However, maximum sequence length – as a hyperparameter – could vary across PLMs. In this work, we prompt publicly available free PLMs. This also makes the process more flexible; for example, the PLM can be updated with gradient descent.

[2]https://beta.openai.com/pricing

3 LMTurk

3.1 Training few-shot learners

We first adapt a PLM to task $\mathcal{T}$ with a few-shot human-labeled gold dataset $\mathcal{G} = \{\mathcal{G}_{train}; \mathcal{G}_{dev}\}$ of $\mathcal{T}$. This procedure mimics one of the initial but crucial steps in crowdsourcing: A few example annotations are shown to the workers, demonstrating $\mathcal{T}$; workers learn about the task and then start annotating (Snow et al., 2008; He et al., 2015; Roit et al., 2020; Trautmann et al., 2020; Lee et al., 2021).

We achieve this adaptation through P-Tuning (Liu et al., 2021b). Taking movie review classification as an example, the goal is to associate a binary label y from {-1, +1} to an input sentence $\mathbf{x} = (x_1, ..., x_n)$ where x_i refers to a token. Unlike finetuning and its variants (Devlin et al., 2019; Houlsby et al., 2019; Zhao et al., 2020b) that train a classifier head, P-Tuning reformulates a sentence into a cloze-style query; the PLM is then requested to respond to the query with an answer selected from a list of candidates. Concretely, an input pair

(**x**, y) = ("watching it leaves you giddy.", -1)

is reformulated to:

"$\underline{\text{[v]}}$ watching it leaves you giddy. $\underline{\text{It}}$ $\underline{\text{is}}$ [MASK] $\underline{\text{.}}$"

in which the $\underline{\text{underlined}}$ tokens are prompting words that give the model a hint about $\mathcal{T}$. "[v]" – whose trainable embedding vector is randomly initialized – is a prompting token injecting extra free parameters. The PLM is then requested to pick a word from {"bad", "good"} to fill in the position of "[MASK]". A mapping {"bad" $\rightarrow$ -1, "good" $\rightarrow$ +1} is used to transform the selected answer to a label such that standard evaluation measures like accuracy can be computed. Prompting has been shown to effectively adapt a PLM to $\mathcal{T}$ *with only a few annotations*; see (Liu et al., 2021a) for a comprehensive review of prompting. We refer to a PLM adapted to $\mathcal{T}$ as an **LMTurker** A.

We select prompting words and mappings based on the small development set $\mathcal{G}_{dev}$. §4.2 provides details on prompting and datasets.

3.2 Aggregating annotations

Individual workers are subject to annotation biases (Snow et al., 2008); therefore, crowdsourcing often collects labels from several workers (Yuen et al., 2011) for an example **x** and then aggregates them for quality control (Alonso et al., 2008). It is straightforward to obtain a group of LMTurkers

$\mathcal{A} = \{A_1, ..., A_k\}$, by adapting the PLM to $\mathcal{T}$ with k different prompts. A querying sentence $\mathbf{x}$ is then annotated by every LMTurker, resulting in a list of labels $\mathbf{y} = [y_1, ..., y_k]$. We evaluate different methods aggregating $\mathbf{y}$ to a single label $\hat{y}$.

BestWorker. Among the k LMTurkers, we pick the one performing best on the dev set $\mathcal{G}_{dev}$.

MajorityVoting. We select the most frequent label in $\mathbf{y} = [y_1, ..., y_k]$ as $\hat{y}$.

To estimate an LMTurker's confidence on label y_i, we compare the logits[3] computed by the PLM:

$$y_i = \arg\max(\text{logit}(y^1),..., \text{logit}(y^N)),$$

where N refers to the label set size, e.g., N=2 for y from {-1, +1}. We then can evaluate several methods of aggregating annotations according to PLM logits.

LogitVoting. We average the logits from all k LMTurkers $\{A_1, ..., A_k\}$ to compute $\hat{y}$:

$$\hat{y} = \arg\max(\tfrac{1}{k}\textstyle\sum_{i=1}^{k} \text{logit}(y_i^1),...,\tfrac{1}{k}\sum_{i=1}^{k} \text{logit}(y_i^N)).$$

WeightedLogitVoting. We use LMTurkers' performance on $\mathcal{G}_{dev}$ to weight their logits and then aggregate the predictions:

$$\hat{y} = \arg\max(\textstyle\sum_{i=1}^{k} w_i\text{logit}(y_i^1),...,\sum_{i=1}^{k} w_i\text{logit}(y_i^N))$$
$$w_i = f(A_i,\mathcal{G}_{dev})/\textstyle\sum_{i=1}^{k} f(A_i,\mathcal{G}_{dev})$$

where $f(A_i, \mathcal{G}_{dev})$ is the performance of the ith LMTurker A_i on $\mathcal{G}_{dev}$.

We collect and aggregate annotations from five LMTurkers, i.e., we use k=5 in our experiments.

3.3 Training a small model $\mathcal{S}$

After adapting LMTurkers to $\mathcal{T}$ through prompting with the few-shot gold dataset $\mathcal{G}$, we next train a small model $\mathcal{S}$ specialized to solve $\mathcal{T}$. Though large PLMs are versatile and strong performers, training and inference are faster and more efficient for small models: They are more deployable in resource-restricted scenarios, e.g., on edge devices (Jiao et al., 2020).

We mimic pool-based active learning (AL; Settles (2009)) to train $\mathcal{S}$. The motivation is to avoid frequent querying of LMTurkers $\mathcal{A}$ because energy and time consumption of PLM inference is costly when the number of queries and $|\mathcal{A}|$ are large.

Concretely, pool-based AL assumes a large collection of unlabeled data $\mathcal{U} = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ for $\mathcal{T}$. $\mathcal{S}$ is first trained with $\mathcal{G} = \{\mathcal{G}_{train}; \mathcal{G}_{dev}\}$. After that, a group of examples $\mathcal{B}$ from $\mathcal{U}$ is sampled (c.f. §3.3.1), which LMTurkers annotate. Next, the annotated and aggregated examples $\mathcal{B}'$ are concatenated with $\mathcal{G}$ to train $\mathcal{S}$. The procedure is repeated iteratively, such that the training data for $\mathcal{S}$ keeps expanding. We denote as $\mathcal{S}^j$ the model trained after the jth iteration. Note that $\mathcal{S}$ is trained from scratch in each iteration (Cohn et al., 1994).

3.3.1 AL acquisition function

At the beginning of the jth iteration, a straightforward strategy of sampling $\mathcal{B}$ from $\mathcal{U}$ is **random sampling**. AL promises to select a more informative $\mathcal{B}$ such that the trained $\mathcal{S}^j$ performs better, under the same budget. These strategies – or *acquisition functions* – rely on $\mathcal{S}^{j-1}$, i.e., $\mathcal{S}$ from the previous iteration: $\mathcal{S}^{j-1}$ is employed to infer $\mathcal{U}$ to obtain labels and logits $\mathcal{P}^{j-1} = \{(y_1, \mathbf{c}_1), ..., (y_M, \mathbf{c}_M)\}$; each $\mathbf{c}_i$ contains the logits of the N labels; $y_i = \arg\max(\mathbf{c}_i)$. We explore two common AL acquisition functions: Entropy (Roy and McCallum, 2001) and LeastConfident (Lewis and Gale, 1994).

Entropy selects from $\mathcal{P}^{j-1}$ examples with the largest prediction entropy, computed using $\mathbf{c}$. Large entropy of an example $\mathbf{x}$ implies that $\mathcal{S}^{j-1}$ is unsure about which label to select; $\mathbf{x}$ is then a query made to LMTurkers to obtain its label $\hat{y}$. $(\mathbf{x}, \hat{y})$ is subsequently added to $\mathcal{G}_{train}$ for training $\mathcal{S}^j$.

LeastConfident selects from $\mathcal{P}^{j-1}$ examples for which the maximum logit in $\mathbf{c}$ is the smallest. Selected examples are then annotated and added to $\mathcal{G}_{train}$ for training $\mathcal{S}^j$.

Our AL setup is fairly standard, both in terms of acquisition functions and iterative enlargement by new sampled data $\mathcal{B}$ at iteration j labeled by $\mathcal{S}^{j-1}$.

3.3.2 Considering annotation quality

As in any realistic AL scenario, annotations are not perfect: LMTurkers do not score perfectly on $\mathcal{T}$. As a result, *annotation quality of LMTurkers needs to be taken into consideration before training* $\mathcal{S}^j$. Denoting the training data of $\mathcal{S}^j$ as $\mathcal{D}^j$, we explore a strategy of processing $\mathcal{D}^j$, based on LMTurker logits $\mathbf{l}$.

InstanceTresholding. We preserve examples $(\mathbf{x}, \hat{y}, \mathbf{l}) \in \mathcal{D}^j$ for which entropy computed on $\mathbf{l}$ is smallest. $\mathcal{G}^{train}$ is always preserved because it is human-labeled gold data. Note that this is different from the strategy of sampling $\mathcal{B}$, where we select from $\mathcal{P}^{j-1}$ examples to which $\mathcal{S}^{j-1}$ is most unsure

[3]Calibration can be conducted to further improve the estimation (Guo et al., 2017). We leave this to future work.

(computed with $\mathbf{c}$). We evaluate[4] the effectiveness of processing $\mathcal{D}^j$ before training $\mathcal{S}^j$ in §5.6.

3.4 Summary of LMTurk

LMTurk can be viewed as intermediate between self training (Yarowsky, 1995; Abney, 2004; Lee et al., 2013; Mi et al., 2021b) and AL. Unlike self training, LMTurk employs *external* models provide labels to $\mathcal{S}$. Different from the artificial setup used in many AL experiments, the provided labels *do not have oracle quality*; so $\mathcal{S}$ must use the annotations more carefully. We next conduct experiments investigating the effectiveness of LMTurk.

4 Datasets and Setup

4.1 Dataset

We evaluate LMTurk on five datasets: Binary (SST2) and fine-grained (five classes) sentiment classification (SST5) with the Stanford Sentiment TreeBank (Socher et al., 2013); news article topic classification with the AG's News Corpus (AGNews; Zhang et al. (2015)); recognizing textual entailment (RTE; Dagan et al. (2006)); assessing linguistic acceptability (CoLA; Warstadt et al. (2019)). Appendix §A reports dataset statistics. SST2/SST5 and AGNews are widely used in crowdsourcing and AL (Laws et al., 2011; Ein-Dor et al., 2020; Margatina et al., 2021; Zhang and Plank, 2021). RTE and CoLA assess the models' ability to understand textual entailment and linguistic phenomena – as opposed to text categorization. We report Matthew's correlation coefficient for CoLA and accuracy for the others (Wang et al., 2018).

Few-shot datasets. Recall LMTurk uses a small human-annotated dataset $\mathcal{G} = \{\mathcal{G}_{train}; \mathcal{G}_{dev}\}$. Denoting n as the number of shots *per class*, we sample $\mathcal{G}^n_{train}$ and $\mathcal{G}^n_{dev}$ for each of $n \in \{8, 16, 32\}$. For SST2, RTE, and CoLA, we use the train and dev sets of GLUE (Wang et al., 2018); $\mathcal{G}^n_{train}$ and $\mathcal{G}^n_{dev}$ are sampled from the train set; the dev set is used as the test set. For SST5 and AGNews, we use the official datasets; $\mathcal{G}^n_{train}$ ($\mathcal{G}^n_{dev}$) is sampled from the train (dev) set; we report performance on the test set. We repeat the sampling process with three random seeds.

4.2 Training setup

Brown et al. (2020) show that large model size is necessary for strong few-shot performance. We use ALBERT-XXLarge-v2 (Lan et al., 2020) – of size 223M parameters – as our large PLM, which is adapted to be an LMTurker A of $\mathcal{T}$ with $\mathcal{G}$. With parameter reuse, ALBERT-XXLarge-v2 outperforms larger models like the 334M BERT-large (Devlin et al., 2019). In contrast, $\mathcal{S}$ must be small to be deployable in practical scenarios. We use TinyBERT-General-4L-312D (Jiao et al., 2020), which has 14.5M parameters.

We train – with prompting – the large PLM with $\mathcal{G}$ for 100 batch steps using batch size 16, AdamW (Loshchilov and Hutter, 2019) and learning rate 5e-4 with linear decay. We prompt the large PLM five times to obtain five LMTurkers; Appendix §C shows prompting details. At each iteration, we finetune $\mathcal{S}$ for 20 epochs using batch size 32, Adam (Kingma and Ba, 2015) and learning rate 5e-5. Each experiment is run with three different random seeds. We use PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020).

[4]Motivated by Wang et al. (2017), we also investigate the effectiveness of weighting training examples. However, we do not observe noticeable improvements of task performance. We list more details in Appendix §E.

	Schick and Schütze (2021a,b)	Gao et al. (2021)	Ours
SST2	n/a	93.0±0.6	93.08±0.62
SST5	n/a	49.5±1.7	46.70±0.93
RTE	69.8	71.1±5.3	70.88±1.70
AGN.	86.3±0.0	n/a	87.71±0.07
CoLA	n/a	21.8±15.9	19.71±1.89

Table 1: LMTurkers achieve comparable few-shot performance with the literature. We refer to *PET* results in Schick and Schütze (2021a,b) and results of *Prompt-based FT (auto) + demonstrations* in Gao et al. (2021).

5 Experiment

5.1 Few-shot performance (non-iterative)

We compare few-shot performance of LMTurkers and the small model $\mathcal{S}$ when *only $\mathcal{G}$ is used.* LMTurker performance is comparable to prior work (Schick and Schütze, 2021a,b; Gao et al., 2021) as shown in Table 1.

Figure 2 compares performance of LMTurkers and $\mathcal{S}$. Appendix §B Table 3 reports numeric values. LMTurkers perform clearly better than $\mathcal{S}$ on CoLA, SST5, AGNews, and SST2; e.g., for SST2, for train/dev size 16, LMTurker accuracy is 93.08% vs. 75.83% for $\mathcal{S}$. LMTurkers' superiority over $\mathcal{S}$ on RTE is modest. As an inference task, RTE is more challenging than classification (e.g., AGNews). We hypothesize that current few-shot learners require more data than $\mathcal{G}^{32}$ to process difficult tasks better than $\mathcal{S}$. Scaling up to even larger PLMs is also a promising direction (Brown et al., 2020;

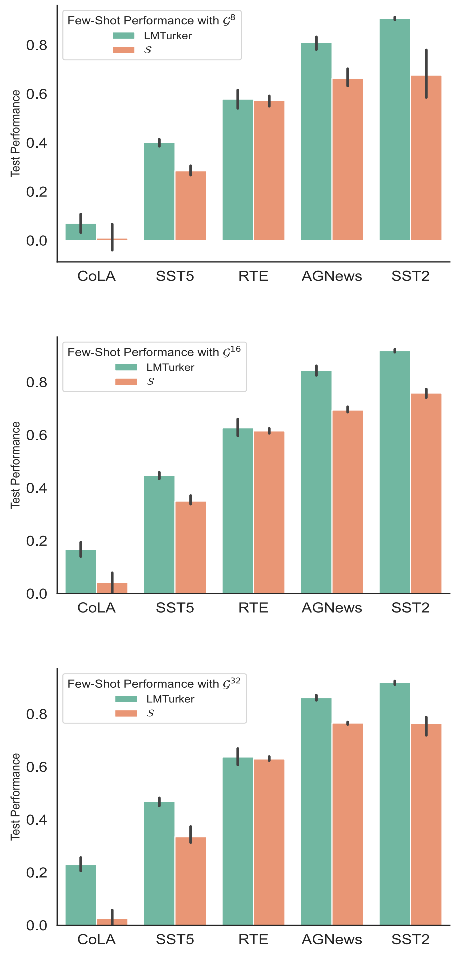

Figure 2: *Few-shot* test set performance of LMTurkers and $\mathcal{S}$. We use the few-shot gold datasets $\mathcal{G}^8$ (top), $\mathcal{G}^{16}$ (middle), and $\mathcal{G}^{32}$ (bottom).

Lester et al., 2021).

Overall, LMTurkers outperform $\mathcal{S}$ with clear margins, evidencing that their annotations can serve as supervisions for training $\mathcal{S}$. We next conduct iterative training to improve performance of $\mathcal{S}$ on $\mathcal{T}$ with supervisions from LMTurkers.

5.2 Iterative training

We investigate the effectiveness of LMTurk by simulating scenarios analogous to active learning. Concretely, we compare three schemes of annotating the sampled data $\mathcal{B}$ at each annotation iteration j:

- Active learning (AL). We use $\mathcal{B}$'s *gold labels* to show how $\mathcal{S}$ performs with expert annotations. Gold labels are ideal, but costly because expert annotators need to be employed.
- Self training (ST). We use $\mathcal{S}^{j-1}$, the model trained in the previous iteration, to annotate $\mathcal{B}$ (Yarowsky, 1995; Abney, 2004; Lee et al., 2013). ST trades supervision quality for annotation cost; no extra cost is introduced. Because there is no external supervision, ST is expected to be a baseline.

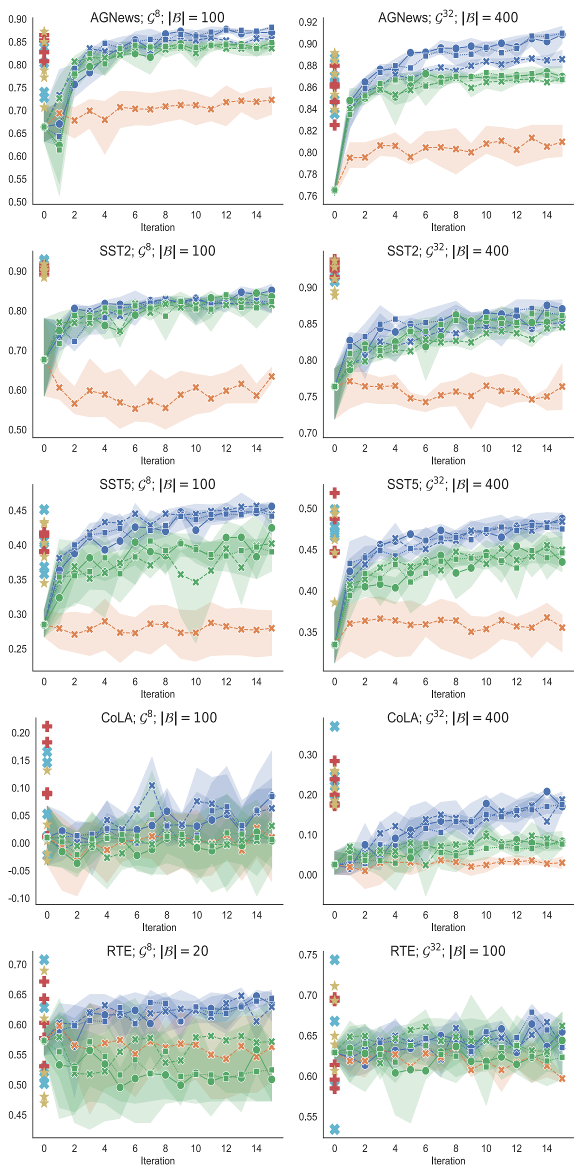

Figure 3: Improving $\mathcal{S}$ with active learning (blue), self training (orange), and LMTurk (green). Free markers at step zero show LMTurker performances; colors distinguish random seeds. Three acquisition functions are: Entropy (●), LeastConfident (■), random sampling (✖). At iteration j, each experiment is repeated three times; we show mean and standard deviation. Appendix Figure 9 visualizes more results.

- LMTurk. We query the LMTurkers to annotate $\mathcal{B}$. LMTurkers are machine learning models, so there is no human labor. Based on the findings in Figure 2, LMTurker supervisions are expected to have better quality than those of ST. Yet LMTurk could fall behind AL because LMTurker labels are not gold labels.

When sampling $\mathcal{B}$ from $\mathcal{U}$ at each iteration j, we consider the strategies described in §3.3. We employ Random for all three schemes and Entropy/LeastConfident for AL/LMTurk. Entropy and LeastConfident rely on $\mathcal{S}^{j-1}$. Regarding the number of sampled examples, we experiment with $|\mathcal{B}|$=100 and $|\mathcal{B}|$=400 for SST2, SST5, AGNews, CoLA. Due to RTE's small size, we use $|\mathcal{B}|$=20 and $|\mathcal{B}|$=100. We run for 15 iterations of improving $\mathcal{S}$. To aggregate annotations from LMTurkers, we use MajorityVoting (§3.2), which is widely used in crowdsourcing. See §5.3 for a comparison of various aggregation methods.

Figure 3 **compares AL, ST, and LMTurk.** ST (orange) noticeably helps $\mathcal{S}$ to perform progressively better on AGNews, e.g., when comparing $\mathcal{S}^{15}$ to $\mathcal{S}^{0}$ shown in the first row, especially when $|\mathcal{B}|$=400. However, we do not identify clear improvements when looking at other tasks. Except for RTE-$\mathcal{G}^8$, ST clearly falls behind AL and LMTurk. This inferior performance meets our expectation because there is no external supervision assisting $\mathcal{S}$ to perform better on $\mathcal{T}$. In what follows, we omit ST for clearer visualization and discussion.

AL (blue) performs the best in most experiments. However, this comes with extra costs that are not negligible: *At each iteration*, human annotators need to annotate 100–400 sentences.

LMTurk (green) holds a position between AL and ST on AGNews, SST2, SST5, and CoLA. Somehow surprisingly, LMTurk performs almost comparably to AL on SST2. Unlike AL, LMTurk requires very little human labor; the only human annotation throughout the entire process is the few-shot gold dataset $\mathcal{G}$. In contrast, AL has high human annotation cost, e.g., 1000–4000 examples by iteration ten. LMTurk also shows clear performance improvements over ST.

Results on RTE are noisy; we conjecture this is due to its very small test set (277 examples). We do not observe performance improvement of $\mathcal{S}$ along the iterations in experiment RTE-$\mathcal{G}^{32}$-$|\mathcal{B}|$=100, likely due to saturated task performance: TinyBERT-General-4L-312D ($\mathcal{S}$) achieves 66.6% on RTE for the full train set (Jiao et al., 2020).

Comparing sampling strategies. Entropy (●) and LeastConfident (■) outperform random sampling (✖) in AGNews and SST2 with noticeable margins – for both AL and LMTurk, especially when $|\mathcal{B}|$=400. They also surpass random sampling when using LMTurk for SST5 and CoLA with $\mathcal{G}^8$. In other words, Entropy and LeastConfident assist LMTurk to achieve the same performance as of using random sampling, but with fewer annotations. For example in AGNews-$\mathcal{G}^8$-$|\mathcal{B}|$=100, LeastConfident at iteration six already achieves comparable performance as random sampling at iteration eleven. This is economically and environmentally beneficial because the number of queries made to LMTurkers, i.e., the cost of running inference passes on the array of large PLMs, is significantly reduced.

Overall, we show that LMTurk can be used to create datasets for training a specialized model $\mathcal{S}$ of solving $\mathcal{T}$ in practical scenarios. To reduce computational cost, we use only Entropy in what follows.

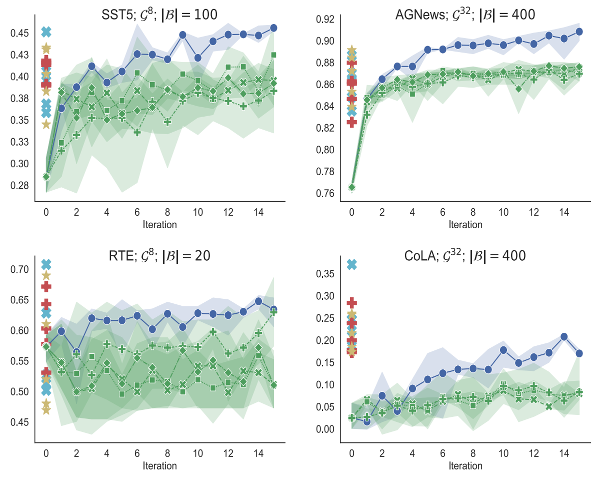

Figure 4: Comparing strategies of aggregating LMTurker annotations. We compare LMTurk (green) with AL (blue). Strategies: LogitVoting (✖), MajorityVoting (■), WeightedLogitVoting (♦), BestWorker (✚). AL uses gold labels without aggregation (●).

5.3 Design choice 1: Aggregation strategies

Figure 4 compares effectiveness of different strategies of aggregating LMTurker annotations (§3.2). Looking at SST5 and AGNews results (top two images), we observe that committee-style aggregation (LogitVoting (✖), MajorityVoting (■), and WeightedLogitVoting (♦)) generally outperforms BestWorker (✚), which simply relies on the LMTurker performing best on $\mathcal{G}_{dev}$. LMTurkers perform well on these two datasets as shown by the free markers at iteration zero; ensembling their predictions results in higher-quality datasets.

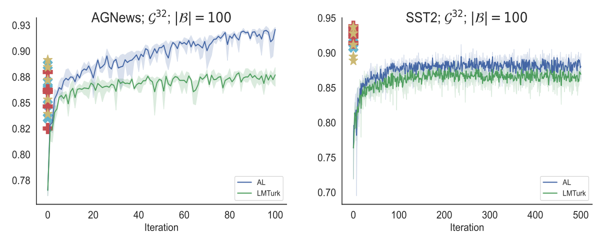

Figure 5: Running more iterations of improving $\mathcal{S}$ with AL and LMTurk. Sampling strategy Entropy is used for both methods; WeightedLogitVoting is used for aggregating LMTurker annotations.

In contrast, BestWorker (✚) has stellar performance on RTE (bottom-left), outperforming committee-style aggregation. Note that even the LMTurkers do not perform really well in this experiment, as shown by the free markers at iteration zero – some LMTurkers even perform worse than $\mathcal{S}$. Ensembling these low-quality annotations seems a worse option than simply relying on the best LMTurker. For CoLA, we observe comparable performance of different aggregation strategies.

5.4 Design choice 2: More iterations

We hypothesize that AL performance is an upper bound for performance when $\mathcal{S}$ is trained with LMTurker annotations – recall that the AL annotations are gold labels. Figure 5 compares AL and LMTurk when running 100 iterations of improving $\mathcal{S}$ on AGNews and 500 iterations on SST2. As expected, AL outperforms LMTurk because the pool of human-annotated data expands. The performance of $\mathcal{S}$ progressively approaches that of the LMTurkers; LMTurk performs comparably to AL in SST2, however, no human labor is required.

5.5 Design choice 3: Distilling logits

We can view LMTurk as a kind of distillation (Hinton et al., 2015): The ability of LMTurkers to solve $\mathcal{T}$ is progressively transferred to $\mathcal{S}$. In this section, we explore the utility of distillation: We train $\mathcal{S}$ with predicted logits[5] instead of discrete labels from LMTurkers. Concretely, we train $\mathcal{S}$ by reducing the KL divergence between its predicted probability distribution (over the label set) and the probability distribution from LMTurkers.

[5]Distilling with intermediate activations likely to further improve performance of $\mathcal{S}$. However, note that PLM intermediate activations are not always available in a Language-Model-as-a-Service framework.

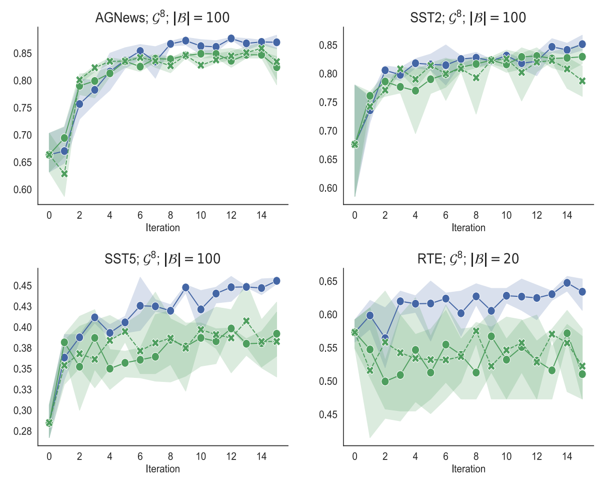

Figure 6: Performance of AL and LMTurk with discrete labels (●) vs. with KL divergence (✖). Entropy is used as the sampling strategy and WeightedLogitVoting is used to aggregate worker annotations.

Figure 6 shows that training $\mathcal{S}$ with KL divergence noticeably improves over discrete labels on AGNews and SST5. This is expected: AGNews and SST5 have larger label set size (four and five) such that the probability distribution over the label set is more informative than that of the binary classification tasks SST2 and RTE.

5.6 Design choice 4: Quality-based filtering

One key difference between AL and LMTurk is that LMTurkers are not oracles: Their labels are not perfect. Hence, it is reasonable to consider processing the training data, denoted as $\mathcal{D}^j$, for $\mathcal{S}^j$, instead of using it indiscriminately as in AL.

InstanceTresholding (§3.3.2) preserves annotations in $\mathcal{D}^j$ for which LMTurkers have the smallest prediction entropy. Concretely, we rank all annotations $(\mathbf{x}, \hat{y}, \mathbf{l}) \in \mathcal{D}^j$ by *entropy*$(\mathbf{l})$ and then keep the τ percent smallest. Note that we always preserve the human-labeled few-shot data $\mathcal{G}_{train}$. We experiment with $\tau \in \{10\%, \ldots, 90\%, 100\%\}$.

Figure 7 left shows the performance of $\mathcal{S}$; Figure 7 right tracks the status of $\mathcal{D}^j$. To measure quality, we compute the accuracy of LMTurker annotations on $\mathcal{D}^j$ (compared to gold labels); see the lineplots and the left y-axis. We also report the size of $\mathcal{D}^j$ as scatter plots (right y-axis).

We observe that τ=10%, i.e., keeping only the 10% most certain examples, gives the worst performance. This is most obvious at iteration three for SST2: The performance drops to near the majority baseline (≈50%). This is because $\mathcal{D}^3$ is small and

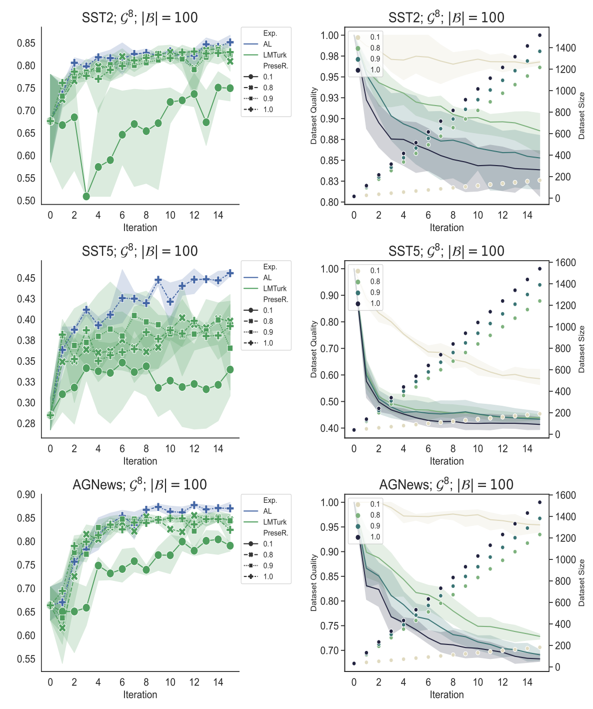

Figure 7: Training $\mathcal{S}$ with examples for which LMTurkers have low entropy. We report performance of $\mathcal{S}$ (left), number and quality (measured by accuracy) of the preserved examples (right) at each iteration.

unbalanced: It has eight negative (from $\mathcal{G}^{train}$) and 38 positive examples. However, using all the LMTurker annotations (τ=100%) may not be optimal either. This is noticeable when looking at SST5: τ=90% and τ=80% are better options.

We see that there is a trade-off between $\mathcal{D}^j$'s quality and size from Figure 7 right. Being conservative, i.e., preserving only a handful of annotations from LMTurkers, results in a small, but high-quality $\mathcal{D}^j$; using all the annotations indiscriminately leads to a large $\mathcal{D}^j$ with low quality. This experiment highlights a key difference between LMTurk and AL: LMTurker annotations are not perfect and taking the annotation quality into consideration when training $\mathcal{S}$ is crucial.

6 Conclusion

In this work, our focus is the research question: *How to make effective use of current few-shot learners?* We propose LMTurk, a simple yet effective method that considers PLM-based few-shot learners as non-expert annotators in crowdsourcing; active learning strategies are incorporated to reduce the cost of annotation. We further show that processing the annotations from LMTurkers can be beneficial.

Future work may combine LMTurker annotations with human annotators in a human-in-the-loop setup (Monarch, 2021) to increase the overall utility of invested resources (Bai et al., 2021). Scaling up to even larger PLMs likely to further boost model performances (Kaplan et al., 2020; Brown et al., 2020) Applying LMTurk to multilingual few-shot learners (Zhao et al., 2021; Winata et al., 2021; Lin et al., 2021) is also promising.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and suggestions. MZ and HS were supported by the European Research Council (ERC# 740516) and the German Federal Ministry of Education and Research (BMBF, grant #01IS18036A).

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365–395.

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen H. Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Rabiu Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. MasakhaNER: Named Entity Recognition for African Languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131.

Omar Alonso, Daniel E. Rose, and Benjamin Stewart. 2008. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15.

Fan Bai, Alan Ritter, and Wei Xu. 2021. Pre-train or annotate? domain adaptation with a constrained budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages

5002–5015, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Sangnie Bhardwaj, Samarth Aggarwal, and Mausam Mausam. 2019. CaRB: A crowdsourced benchmark for open IE. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6262–6267, Hong Kong, China. Association for Computational Linguistics.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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*, volume 33, pages 1877–1901. Curran Associates, Inc.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 286–295, Singapore. Association for Computational Linguistics.

Chris Callison-Burch and Mark Dredze, editors. 2010. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, Los Angeles.

David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine learning*, 15(2):201–221.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Anca Dumitrache, Oana Inel, Benjamin Timmermans, Carlos Ortiz, Robert-Jan Sips, Lora Aroyo, and Chris Welty. 2021. Empirical methodology for crowdsourcing ground truth. *Semantic Web*, 12(3):1–19.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Richard M Felder and Rebecca Brent. 2009. Active learning: An introduction. *ASQ higher education brief*, 2(4):1–5.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 144–151, Ann Arbor, Michigan. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Jeff Howe. 2008. *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House.

Jeff Howe et al. 2006. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Emily Jamison and Iryna Gurevych. 2015. Noise or additional information? leveraging crowdsource annotation item agreement for natural language tasks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 291–297, Lisbon, Portugal. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3250–3258, Online. Association for Computational Linguistics.

Daniel Khashabi, Arman Cohan, Siamak Shakeri, Pedram Hosseini, Pouya Pezeshkpour, Malihe Alikhani, Moin Aminnaseri, Marzieh Bitaab, Faeze Brahman, Sarik Ghazarian, Mozhdeh Gheini, Arman Kabiri, Rabeeh Karimi Mahabagdi, Omid Memarrast, Ahmadreza Mosallanezhad, Erfan Noury, Shahab Raji, Mohammad Sadegh Rasooli, Sepideh Sadeghi, Erfan Sadeqi Azer, Niloofar Safi Samghabadi, Mahsa Shafaei, Saber Sheybani, Ali Tazarv, and Yadollah Yaghoobzadeh. 2021. ParsiNLU: A suite of language understanding challenges for Persian. *Transactions of the Association for Computational Linguistics*, 9:1147–1162.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active learning with Amazon Mechanical Turk. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

Ji-Ung Lee, Jan-Christoph Klie, and Iryna Gurevych. 2021. Annotation curricula to implicitly train non-expert annotators. *arXiv preprint arXiv:2106.02382*.

Ji-Ung Lee, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4233–4247, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Weixin Liang, James Zou, and Zhou Yu. 2020. ALICE: Active learning with contrastive natural language explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4380–4391, Online. Association for Computational Linguistics.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2021. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fei Mi, Yitong Li, Yasheng Wang, Xin Jiang, and Qun Liu. 2021a. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. *arXiv preprint arXiv:2109.04645*.

Fei Mi, Wanhao Zhou, Lingjing Kong, Fengyu Cai, Minlie Huang, and Boi Faltings. 2021b. Self-training improves pre-training for few-shot learning in task-oriented dialog systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1887–1898.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Robert Munro Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.

Nikita Nangia, Saku Sugawara, Harsh Trivedi, Alex Warstadt, Clara Vania, and Samuel R. Bowman. 2021. What ingredients make for an effective crowdsourcing protocol for difficult NLU data collection tasks? 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 1221–1235, Online. Association for Computational Linguistics.

Sungjoon Park, Jihyung Moon, Sung-Dong Kim, Won Ik Cho, Jiyoon Han, Jangwon Park, Chisung Song, Junseong Kim, Yongsook Song, Tae Hwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Young kuk Jeong, Inkwon Lee, Sang gyu Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin Shin, Seonghyun Kim, Lucy Park, Alice H. Oh, Jung-Woo Ha, and Kyunghyun Cho. 2021. KLUE: Korean language understanding evaluation. *ArXiv*, abs/2105.09680.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Silviu Paun and Dirk Hovy, editors. 2019. *Proceedings of the First Workshop on Aggregating and Analysing Crowdsourced Annotations for NLP*. Association for Computational Linguistics, Hong Kong, China.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 803–818, Online. Association for Computational Linguistics.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2:441–448.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Christopher Schröder and Andreas Niekler. 2020. A survey of active learning for text classification using deep neural networks. *arXiv preprint arXiv:2008.07267*.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green AI. *Communications of the ACM*, 63(12):54–63.

Burr Settles. 2009. Active learning literature survey.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, Honolulu, Hawaii. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Aditya Siddhant and Zachary C. Lipton. 2018. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium. Association for Computational Linguistics.

Edwin Simpson and Iryna Gurevych. 2018. Finding convincing arguments using scalable Bayesian preference learning. *Transactions of the Association for Computational Linguistics*, 6:357–371.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Dietrich Trautmann, Johannes Daxenberger, Christian Stab, Hinrich Schütze, and Iryna Gurevych. 2020. Fine-grained argument unit recognition and classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9048–9056.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. In *Advances in Neural Information Processing Systems*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488, Copenhagen, Denmark. Association for Computational Linguistics.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. 2021. Language models are few-shot multilingual learners. *arXiv preprint arXiv:2109.07684*.

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.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239, 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*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2011. A survey of crowdsourcing systems. In *2011*

IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, pages 766–773.

Mike Zhang and Barbara Plank. 2021. Cartography active learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 395–406, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Ye Zhang, Matthew Lease, and Byron C. Wallace. 2017. Active discriminative text representation learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3386–3392. AAAI Press.

Mengjie Zhao, Philipp Dufter, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2020a. Quantifying the contextualization of word representations with semantic class probing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1219–1234, Online. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020b. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

Mengjie Zhao and Hinrich Schütze. 2021. Discrete and soft prompting for multilingual models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8547–8555, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mengjie Zhao, Yi Zhu, Ehsan Shareghi, Ivan Vulić, Roi Reichart, Anna Korhonen, and Hinrich Schütze. 2021. A closer look at few-shot crosslingual transfer: The choice of shots matters. 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 5751–5767, Online. Association for Computational Linguistics.

A Reproducibility Checklist

A.1 Computing infrastructure

We use four Tesla V100 GPUs to prompt each of the LMTurkers, and a single Tesla V100 GPU is used when finetuning the small model $\mathcal{S}$.

A.2 Datasets

For SST2, CoLA, and RTE, we use the official datasets available on the benchmark website gluebenchmark.com. We download SST5 dataset from nlp.stanford.edu/sentiment and AGNews from the link provided by Zhang et al. (2015).

The number of testing examples of each dataset is shown in Table 2. Note that for SST2, CoLA, and RTE, $\mathcal{G}^{dev}$ is sampled from the training set, and the dev set is used as the test set.

CoLA	SST5	RTE	AGNews	SST2
1042	2210	277	7600	872

Table 2: Number of testing examples.

B Numerical Results

Table 3 reports the numerical value of Figure 2.

C Prompting Details

For each task, we list the five prompts employed to adapt a PLM to a LMTurker. "[v]" is a prompting token whose trainable embedding vector is randomly initialized.

For **SST5**, we use following prompts:

- "[v] **x** It is [MASK]."
- "[v] **x** Such a [MASK] movie."
- "**x** [v] It is pretty [MASK]."
- "It is [MASK] because **x** [v]"
- "**x** So it is [MASK]. [v]"

and the PLM picks a word from {"crap", "bad", "normal", "good", "perfect"}. to fill the position of "[MASK]". The mapping {"crap" $\rightarrow$ 1, "bad" $\rightarrow$ 2, "normal" $\rightarrow$ 3, "good" $\rightarrow$ 4, "perfect" $\rightarrow$ 5 } is used to convert model predictions to numerical values.

For **SST2**, we use following prompts:

- "[v] **x** It is [MASK]."
- "[v] **x** Such a [MASK] movie."
- "**x** [v] It is pretty [MASK]."
- "It is [MASK] because **x** [v]"
- "**x** So it is [MASK]. [v]"

and the PLM picks a word from {"bad", "good"} to fill the position of "[MASK]". The mapping {"bad" $\rightarrow$ 0, "good" $\rightarrow$ 1} is used.

For **AGNews**, we use following prompts:

- "[v] **x** It is about [MASK]."
- "**x** [v] Topic: [MASK]."
- "**x** [v] The text is about [MASK]."
- "**x** Topic: [MASK]. [v]"
- "**x** [v] [MASK]."

and the PLM picks a word from {"world", "sports", "economy", "technology"} to fill the position of "[MASK]". The mapping {"world" $\rightarrow$ 1, "sports" $\rightarrow$ 2, "economy" $\rightarrow$ 3, "technology" $\rightarrow$ 4 } is used.

For **CoLA**, we use following prompts:

- "[v] **x** It sounds [MASK]."
- "[v] **x** The sentence is [MASK]."
- "[v] **x** It is a [MASK] sentence."
- "**x** [v] [MASK]."
- "[v] **x** [MASK]."

and the PLM picks a word from {"wrong", "ok"} to fill the position of "[MASK]". The mapping {"wrong" $\rightarrow$ 0, "okay" $\rightarrow$ 1} is used.

For **RTE**, we use following prompts:

- "**p** Question: **h**? [v] Answer: [MASK]."
- "**p** [SEP] **h**? [MASK]. [v]"
- "**p** [SEP] **h**? [v] answer: [MASK]."
- "**p** [SEP] In short **h**. [MASK]. [v]"
- "[v] **p** [SEP] In short **h**. [MASK]."

where **p** and **h** refer to premise and hypothesis. The PLM picks a word from {"No", "Yes"} to fill the position of "[MASK]". The mapping {"No" $\rightarrow$ 0, "Yes" $\rightarrow$ 1} is used.

	$\mathcal{G}^8$		$\mathcal{G}^{16}$		$\mathcal{G}^{32}$	
	Workers	$\mathcal{S}$	Workers	$\mathcal{S}$	Workers	$\mathcal{S}$
	91.13±0.52		91.93±1.09		91.97±0.83	
	91.63±0.68		93.08±0.62		91.70±1.78	
SST2	90.18±1.00	67.63±8.01	91.74±1.04	75.83±1.35	91.21±1.83	76.37±3.16
	90.83±0.58		90.79±0.47		91.13±0.24	
	90.52±1.84		91.67±1.36		93.23±0.37	
	41.37±1.55		45.16±2.13		45.91±0.96	
	42.32±2.04		45.96±2.12		48.64±0.59	
SST5	40.57±2.70	28.47±1.61	46.70±0.93	34.97±1.51	50.53±0.94	33.47±2.79
	37.69±1.34		42.53±2.43		43.32±3.42	
	38.05±2.60		42.96±0.69		45.72±1.43	
	68.95±1.47		68.35±2.29		71.72±1.96	
	54.99±3.76		57.64±3.23		58.48±3.59	
RTE	62.70±1.33	57.30±1.79	70.88±1.70	61.50±0.78	68.47±1.19	62.93±0.74
	50.42±2.07		58.60±1.62		59.33±4.72	
	51.99±4.45		57.88±2.83		60.41±2.47	
	75.39±5.25		83.06±0.83		84.92±0.28	
	85.40±1.43		87.71±0.07		87.79±1.08	
AGNews	78.83±4.77	66.37±2.95	83.59±2.96	69.40±0.93	87.39±1.29	76.53±0.41
	85.07±1.09		87.69±0.04		87.17±0.67	
	79.95±0.86		80.15±3.38		83.32±0.59	
	0.14±1.43		11.81±7.82		19.88±3.30	
	2.42±4.84		15.23±7.07		22.51±0.96	
CoLA	7.40±8.12	0.97±4.40	19.71±1.89	4.27±3.26	26.34±1.54	2.50±2.41
	9.91±7.98		17.14±2.48		18.15±0.63	
	15.33±2.15		19.66±0.48		27.58±7.09	

Table 3: Few-shot performance of the five LMTurkers and the small model $\mathcal{S}$. Each experiment is repeated three times and we report mean and standard deviation.

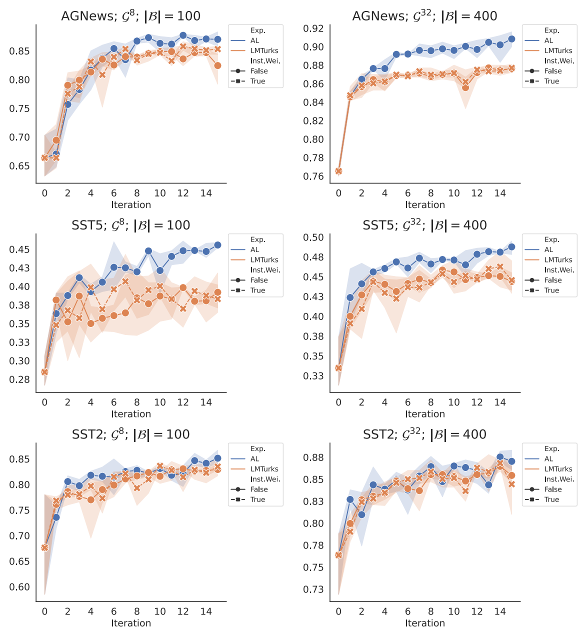

Figure 8: Weighting the training instances from LMTurkers.

D More Visualizations

Figure 9 visualizes the performance of $\mathcal{S}$ when different $|\mathcal{G}|$ and $|\mathcal{B}|$ are used.

E Instance Weighting

Following Wang et al. (2017), we associate each example $(\mathbf{x}, \hat{y}, \mathbf{l}) \in \mathcal{D}^j$ with weight 1-$entropy(\mathbf{l})$ when computing the loss during training $\mathcal{S}^j$. We can interpret this weight as a measure of the certainty of the LMTurkers ensemble.

Figure 8 reports the performance of $\mathcal{S}$ when using instance weighting, however, the impacts are less noticeable.

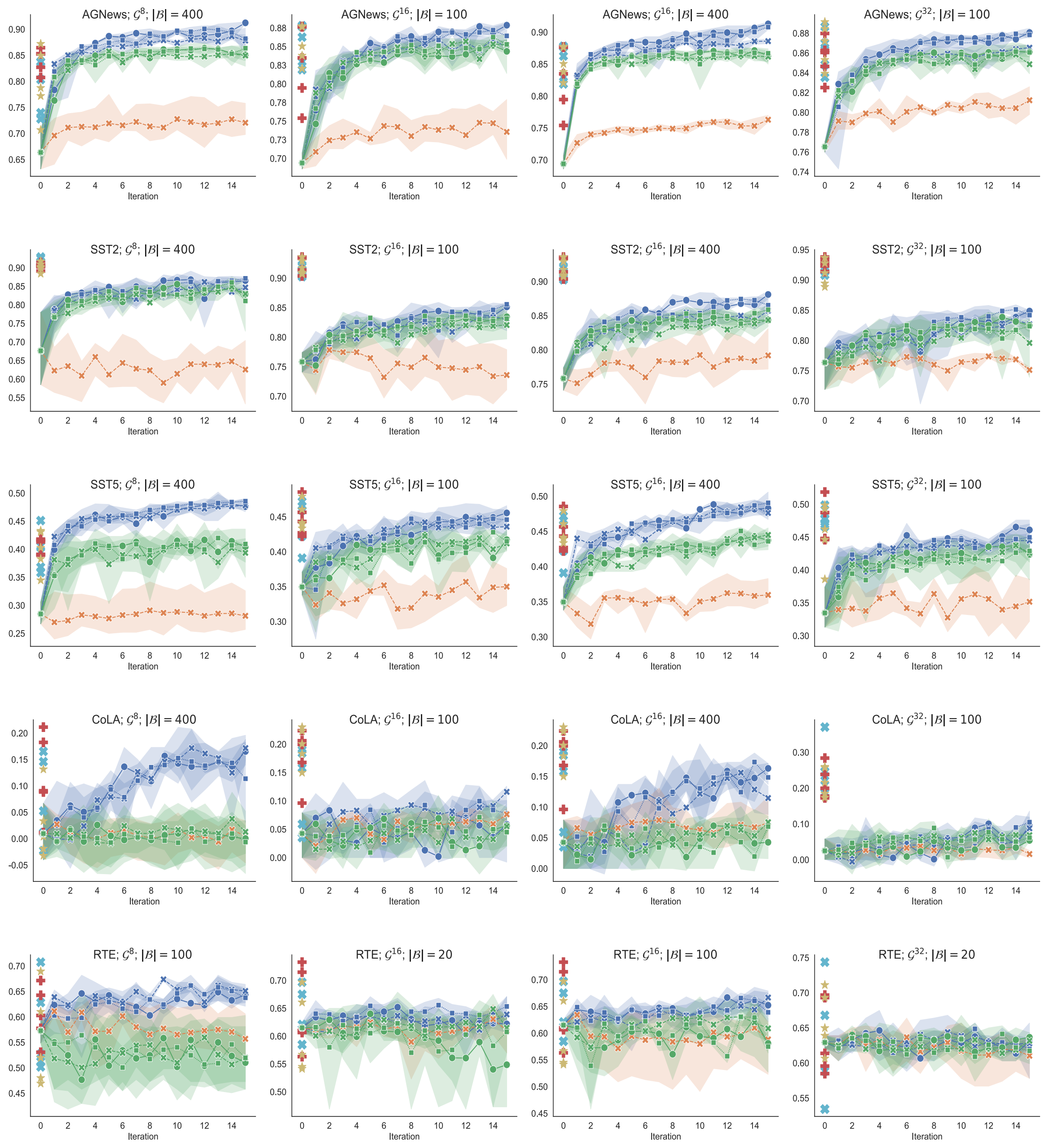

Figure 9: Improving $\mathcal{S}$ with active learning (blue), self training (orange), and LMTurk (green). Free markers at step zero show LMTurker performances; colors distinguish random seeds. Three acquisition functions are: Entropy (●), LeastConfident (■), random sampling (✖). At iteration j, each experiment is repeated three times; we show mean and standard deviation. We evaluate different $|\mathcal{G}|$ and $|\mathcal{B}|$.