Predicting generalization performance with correctness discriminators

Yuekun Yao and **Alexander Koller**
Department of Language Science and Technology
Saarland Informatics Campus
Saarland University, Saarbrücken, Germany
{ykyao, koller}@coli.uni-saarland.de

Abstract

The ability to predict an NLP model's accuracy on unseen, potentially out-of-distribution data is a prerequisite for trustworthiness. We present a novel model that establishes upper and lower bounds on the accuracy, without requiring gold labels for the unseen data. We achieve this by training a *discriminator* which predicts whether the output of a given sequence-to-sequence model is correct or not. We show across a variety of tagging, parsing, and semantic parsing tasks that the gold accuracy is reliably between the predicted upper and lower bounds, and that these bounds are remarkably close together.

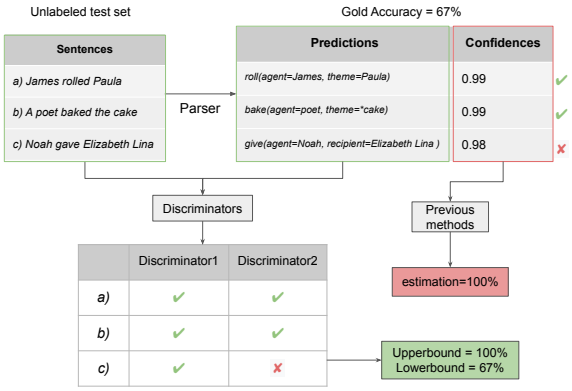


Figure 1: Comparison of our discriminators and confidence-based methods. Our method provides upper and lower bounds which can capture gold accuracy.

1 Introduction

In order for a user to trust that an NLP system performs its task with sufficient reliability, the user must be able to judge the system's accuracy on real-world tasks of interest. This ability is growing in importance with the rapidly increasing prominence of NLP technology in users' daily lives and the growing capability of this technology to solve high-level tasks, e.g. by orchestrating the use of external tools (Yao et al., 2023; Shinn et al., 2023). At the same time, even the best available models still struggle on out-of-distribution (OOD) test sets (Lake and Baroni, 2018; Li et al., 2023) and complex tasks on unseen domains (Zhou et al., 2024; Jimenez et al., 2024).

In realistic settings, the accuracy of an NLP model M needs to be estimated on *unlabeled* test data; it is plausible that the estimator has access to the user's inputs, but not to gold annotations that would capture the behavior the user intended. There is some previous work on estimating the accuracy of M on unlabeled test data, primarily for text or image classification tasks and based on M's confidence (Garg et al., 2022; Guillory et al., 2021). However, existing accuracy estimation models provide only point estimates for the accuracy of M, which hides their own uncertainty; a user cannot judge whether the accuracy estimator is confident about its estimates or whether they should be cautious about trusting them. Ultimately, there is an infinite hierarchy of accuracies: the true accuracy of M; the accuracy of the accuracy estimator; estimates of *that* accuracy; and so on.

In this paper, we take the first step up this hierarchy by offering a method for capturing the accuracy predictor's uncertainty about the estimation of M's accuracy. Instead of directly calculating a point estimate for M's accuracy, our method predicts *upper and lower bounds* for this accuracy, from unlabeled test data. We focus on estimating the accuracy of sequence-to-sequence models, applied to parsing, semantic parsing, and tagging tasks; these tasks have the advantage over other sequence generation tasks that there is a unique correct answer, which allows us to talk about accuracies.

We first train a *discriminator* to predict whether M's output on a given input is correct or not; we show that this can be done with remarkable accuracy across a range of tasks. We then run an ensemble of discriminators on M's predictions on the unlabeled test data and obtain upper and lower bounds through a voting mechanism (Figure 1). We

Findings of the Association for Computational Linguistics: EACL 2024, pages 11725–11739

show across a variety of in-distribution and OOD tasks that M's true accuracy is reliably between the upper and lower bounds, and that these bounds are quite tight. Finally, if forced to predict point estimates of the true accuracy, our model provides more precise estimates than earlier work on most datasets, by taking the mean of the upper and lower bounds.

We will release our code online [1].

2 Related work

Calibration. A neural model is called *well-calibrated* if its predicted probability (e.g. confidence) for its decision (e.g. label or sequence) aligns to the probability of the prediction correctness. Much prior work has attempted to improve the calibration of systems, often through either modifying training objectives or posthoc methods: Kong et al. (2020) add a regularization term into training objective to address in-distribution calibration and out-of-distribution detection for text classification; Desai and Durrett (2020) exploit temperature scaling (Guo et al., 2017) to normalize output logits with a scalar temperature parameter; Dong et al. (2018); Kamath et al. (2020) train an additional regressor to estimate the model confidence with designed features for semantic parsing; Jiang et al. (2021b) investigate all these methods and find that posthoc-based methods are universally helpful for question answering tasks.

Most calibration works above focus on in-distribution (ID) tasks and assume a development set as given, which allows them to estimate parameters (e.g. temperature) to yield the optimal confidence. However, according to Kamath et al. (2020), the predicted model confidence is an unreliable estimate of the correctness on OOD generalization tasks. Compared to such calibration works, our method applies just as easily to OOD as to ID tasks. Further, development sets from OOD distributions are usually difficult to access, which introduces additional challenges of applying calibration-based methods. Kamath et al. (2020) also consider distribution shift, but their calibrator requires a small amount of data from a known OOD distribution.

Predicting test accuracy from unlabeled data. Previous works have investigated predicting the model performance on an unannotated OOD test set for other tasks: Guillory et al. (2021) exploit the difference of confidences between training distribution and the OOD distribution as a useful feature; Jiang et al. (2021a) show that the test error of deep networks can be estimated by the disagreement of two models trained with the same architecture on the same training set but with two different runs; Yu et al. (2022) exploits the euclidean distance between model parameters trained on differently distributed data to predict generalization errors; Garg et al. (2022) estimate a threshold of model confidence from training data and predict the correctness of OOD data based on it; Fu et al. (2023) train an additional model to predict the accuracy of large language models on question answering tasks, which takes as input confidence scores and outputs the overall accuracy of the test set.

Works introduced above estimate the accuracy as a scalar value between 0 and 1. In contrast, our method explicitly judges the uncertainty of the estimated accuracy, providing upper and lower bounds for the estimated accuracy. Besides, previous works only consider image classification and natural language inference tasks. Our work shows that for sequence generation tasks like semantic parsing, the predicted sequence can serve as a good-enough feature to determine the prediction correctness on OOD data.

Quality estimation in NLP tasks. Automatic accuracy prediction has also been investigated for NLP tasks: Van Asch and Daelemans (2010) exploit similarity metrics between the training and test set to estimate POS tagger performances; Chatterjee et al. (2018) train regressors to predict BLEU (Papineni et al., 2002) scores of a machine translation system with given features; Opitz and Frank (2019) train regressors to predict F1 scores for subtasks of AMR (Banarescu et al., 2013). Compared to these works, our method does not require manually designed features and thus is easy to be adapted to any sequence generation tasks. Varshney and Baral (2023) also train a correctness discriminator to improve the coverage of a selective prediction system for question answering. In contrast to this work, we use discriminators to predict accuracies, and more specifically upper and lower bounds.

3 Correctness discriminator

The core of our approach is to construct and train a *correctness discriminator* model, which judges the correctness of a model prediction on unseen data. In this section, we first introduce how we design the discriminator model and collect training data

[1] https://github.com/coli-saar/discriminator

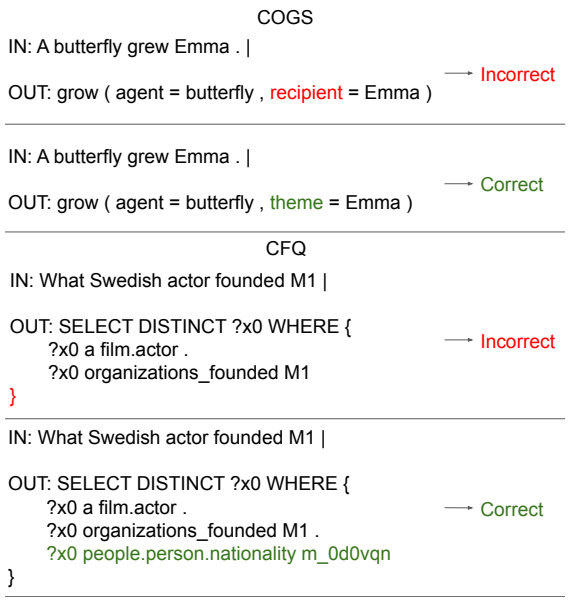


Figure 2: Examples of COGS and CFQ training data for the discriminator. *IN* refers to the input sentence and *OUT* refers to the predicted output sequence (e.g. logical form for COGS and SPARQL query for CFQ).

(Section 3.1), and then describe how to predict the upper bound and lower bounds accuracy (Section 3.2). To avoid confusion, we call the model for the original parsing or tagging tasks a *parser* and the model for predicting the parser performance a *discriminator*. Note that here we only assume that the parser solves a sequence-to-sequence task, but the task output can be any sequence – not just a linearized parse.

3.1 Discriminator design

The discriminator is designed as a binary classifier whose task is to determine whether a given predicted sequence is the correct output for a given natural language sentence. Formally, given a natural language sentence $X \in \mathcal{X}$ and a predicted symbolic sequence (e.g. meaning representation for semantic parsing tasks) $Y \in \mathcal{Y}$, the discriminator $F : \mathcal{X} \times \mathcal{Y} \rightarrow \{Correct, Incorrect\}$ maps them to a *Correct* or *Incorrect* label to represent its correctness.

In this paper, we explore different model architectures as the discriminators: encoder-only, encoder-decoder, and decoder-only. All three models take as input the concatenation of the input natural language sentence with the predicted sequence. For each architecture, we finetune an existing pretrained language model as the discriminator. For encoder-only (e.g. Roberta) discriminators, the input is first encoded into hidden representations and then fed into an additional multi-layer perceptron classifier, which determines the *Correct* or *Incorrect* label. For encoder-decoder (e.g. T5) and decoder-only (e.g. LLaMA) discriminators, the decoder directly generates the label.

Now we discuss how to collect training data for our discriminator. In principle, the training data should contain both positive and negative examples. For positive examples, we can always exploit the training set of the parser. However, it is non-trivial to obtain negative examples. Such examples can be synthesized by applying noise functions (e.g. replacement or deletion) to positive examples (Kim et al., 2021), but this requires prior knowledge about errors a parser tends to make. Another option is to collect errors a trained parser made on its training set, which is still challenging since parsers yield near-perfect accuracies on their training sets.

We therefore generate negative examples from intermediate checkpoints of our parser during its training. Specifically, we run the parser checkpoint on its training data. We take incorrect predictions from the decoder beam as the negative training data for the discriminator. Figure 2 gives examples of our training data.

Given the described discriminator, we can estimate the accuracy of our parser on any unseen test sets. Assuming a parser makes predictions on $|D_t|$ instances and the discriminator labels $|D_c|$ of predictions as *Correct*, the predicted accuracy can be calculated as Eq 1.

$$Acc_{pred} = \frac{|D_c|}{|D_t|} \quad (1)$$

3.2 Bounds prediction

We now introduce how to predict upper and lower bounds for accuracy with the discriminator described above. This is implemented by two voting mechanisms, *Ensemble_correct* and *Ensemble_incorrect*. These mechanisms aggregate outputs from multiple trained discriminators.

- *Ensemble_correct* predicts *Correct* if at least one discriminator predicts *Correct*; if all discriminators predict *Incorrect*, it also predicts *Incorrect*. This yields the most optimistic estimation, assuming correctness if at least one discriminator agrees.

- Conversely, *Ensemble_incorrect* predicts *Incorrect* if at least one discriminator predicts

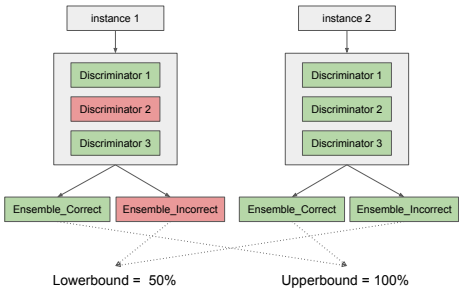


Figure 3: Example of calculating upper and lower bounds from discriminators. *Green* blocks mean the instance is predicted as *Correct* by the discriminator and *Red* blocks refer to *Incorrect*. *Ensemble_incorrect* predicts *Incorrect* for instance 1 and *Correct* for instance 2. Hence, the lower bound is $1/2 = 50\%$.

Incorrect; otherwise *Correct*. This mechanism is more cautious, predicting an instance as incorrect if any discriminator disagrees.

We calculate the upper bound of the accuracy as Acc_{pred} in Eq. 1, with $|D_c|$ determined by the output of the *Ensemble_correct* mechanism instead of a single discriminator. Similarly, the lower bound is calculated with $|D_c|$ determined by the output of *Ensemble_incorrect*. Figure 3 illustrates how the ensembles compute bounds.

Ensembles of neural networks have been shown effective for uncertainty quantification (Lakshminarayanan et al., 2017; Lukovnikov et al., 2021) by averaging confidence scores of individual models. Here we use voting mechanisms to calculate the upper and lower accuracy bounds.

4 Experiments

We introduce our datasets, model setup, evaluation metrics and experimental results in this section.

4.1 Datasets

We experiment with three tasks: semantic parsing, part-of-speech tagging and constituency parsing.

For semantic parsing, we consider two OOD generalization scenarios: compositional generalization and low-resource domain adaptation. We use the COGS (Kim and Linzen, 2020) and CFQ (Keysers et al., 2020) datasets to evaluate compositional generalization. For CFQ, we use its MCD1 and MCD2 splits. For low-resource domain adaptation, we use the TOPv2 (Chen et al., 2020) dataset. We also evaluate our method on in-distribution task with the AMR 2.0 dataset (Banarescu et al., 2013).

For part-of-speech (POS) tagging and constituency parsing tasks, we use the Penn Treebank 3 (PTB) dataset (Marcus et al., 1993). We train our parser on the WSJ training set and evaluate its in-domain performance on the WSJ test set and cross-domain performance on the Brown corpus. We predict the generalization performance for both the WSJ test set (i.e. in-distribution test set) and the Brown corpus (i.e. OOD test set), which we call Syn-WSJ, Syn-Brown (for parsing) and POS-WSJ, POS-Brown (for tagging) in this paper. In addition, we experiment with POS-COGS, a POS tagging dataset generated based on COGS (Yao and Koller, 2022), to evaluate compositional generalization in the POS tagging task. Details of our datasets are in Appendix A.

4.2 Setup

Parser. We finetune *T5-base* (Raffel et al., 2020) as the parser for all tasks described above. To do this, we convert all of our tasks into sequence generation tasks, where the output sequence can be a semantic meaning representation, POS tag sequence or linearized parse tree. All our parsers achieve the same or close performance as those reported in previous works using T5.

Discriminator. We experiment with three architectures as discriminators: (1) An encoder-only architecture consisting of a *Roberta-base* encoder (Liu et al., 2019) and an MLP classifier (2) An encoder-decoder architecture using *T5-base* (Raffel et al., 2020) and (3) A decoder-only architecture using *Vicuna-7B* (Zheng et al., 2023). We report results of the T5 and RoBERTa discriminators in Section 4.4, since the T5 discriminators share the same architecture as our parsers, and the RoBERTa discriminators have the fewest parameters. Results for the Vicuna discriminators in Appendix C. All three discriminators perform well across corpora.

To collect negative training examples, we validate the parser checkpoint every K steps on its training set, where K is a hyperparameter. Since our parser is an encoder-decoder model, we randomly sample incorrect predictions from the decoded beam predictions. For each task we train an ensemble of 5 discriminators with different random seeds. See Appendix B for more training details.

Comparable baseline. We also compare our methods with several previous methods.

MaxProb. Maxprob is a strong baseline shown in Kamath et al. (2020). Assuming we are given a threshold γ on the maximal prediction probability (e.g. confidence) of a parser, we can predict an instance as *Correct* if the parser confidence on

		T5						Roberta					
		Single		Upper		Lower		Single		Upper		Lower	
		CR	IR	CR	IR	CR	IR	CR	IR	CR	IR	CR	IR
OOD	MCD1	97.0	83.1	99.2	69.8	92.9	94.5	97.0	90.0	98.5	80.8	92.5	97.3
	MCD2	80.6	83.7	83.4	77.9	71.7	92.2	78.3	85.0	84.7	79.4	70.6	90.6
	COGS	98.5	96.6	99.8	89.4	98.5	96.9	98.9	87.7	99.8	86.6	97.8	90.8
	TOP	87.4	57.9	92.2	44.7	82.5	78.9	85.4	65.8	91.3	42.1	77.7	78.9
	POS-Brown	81.3	54.7	94.0	26.0	52.2	84.7	62.1	67.4	95.2	23.3	44.4	88.0
	POS-COGS	98.8	86.3	99.9	84.4	98.7	89.2	99.9	90.9	100	86.7	99.5	94.9
	Syn-Brown	33.8	90.0	69.3	73.9	29.7	96.0	17.5	74.3	70.6	38.4	4.2	96.3
ID	AMR 2.0	37.0	98.3	70.9	84.1	29.1	99.1	38.9	96.1	59.1	80.2	37.3	97.4
	POS-WSJ	80.2	53.6	93.0	26.5	52.4	84.8	64.5	65.2	96.0	21.7	49.6	87.6
	Syn-WSJ	44.5	89.2	66.2	73.1	20.5	97.0	27.1	66.0	74.3	34.6	6.5	94.7

Table 1: Results of our discriminators on different datasets. For each dataset, we report *Correct-Recall* (CR) and *Incorrect-Recall* (IR). *Single* refers to the results with predictions from a single discriminator. *Upper* refers to the results with discriminator predictions using *ensemble_incorrect*; *Lower* to *ensemble_correct*.

this instance is higher than γ, otherwise *Incorrect*. Since we have no prior knowledge about the OOD distribution, we set $\gamma = 0.5$ in our experiments.

Average Confidence (AC). We take the average confidence across the test set as the predicted accuracy. Different from previous works where the confidence is defined as the maximal softmax probability of the classifier, here we define the confidence as the probability of the most probable sequence in the beam, which is calculated by the product of softmax probabilities of each word in the sequence.

Difference Of Confidence (DOC). We also estimate the accuracy using DOC (Guillory et al., 2021). We start with a development set that follows the same distribution as the training data. We then subtract the difference in average confidence between the development set and the test set from the gold accuracy on the development set. The result is the estimated accuracy on the test set.

Average Thresholded Confidence (ATC) is a strong method recently proposed by Garg et al. (2022), which has been shown to be more effective than previous methods. Applying ATC consists of two steps. First, we estimate a threshold γ on parser confidence scores to make the number of errors made by the parser match the number of instances where the parser confidence is lower than γ; then we can obtain the predicted accuracy on the test set by calculating the fraction of unlabeled instances that obtain a score below γ.

Maxprob (Oracle). To compare with our predicted bounds, we calculate bounds based on Maxprob, where estimate γ such that the *Correct-Recall* calculated based on γ is equal to the one from the predicted upper bound calculated by our discriminators. This measures the reliability of the parser's confidence in recognizing correct instances compared to discriminators. Similarly, we calculate a lower bound by matching *Incorrect-Recall* scores. Note that this method requires annotated test sets, which are impractical for real-world applications.

4.3 Evaluation metrics

For all parsing tasks, we evaluate the exact match accuracy of our parser.

For discriminators, we need a metric to quantify the quality of the predicted upper and lower bounds. Intuitively, such a metric should reflect *whether the gold accuracy is within the bounds* (i.e. reliability) and *whether the bounds are tight* (i.e. tightness).

Previous work predicts point estimations for OOD test sets and evaluates their method with mean absolute estimation error (MAE) by calculating average absolute difference between the true accuracy on the target data and the estimated accuracy on the same unlabeled examples. Their results are averaged over multiple test sets for each classifier (e.g. parser in our tasks). In our setup, most tasks only have one OOD or ID test set, and thus we calculate the absolute estimation error (AE) to compare with previous works. Equation 2 defines the metric, where Acc_{gold} denotes the gold accuracy and Acc_{pred} denotes the predicted accuracy.

$$|Acc_{gold} - Acc_{pred}| \quad (2)$$

We calculate Acc_{pred} with two methods: (1) calculating the mean of estimated accuracies by all discriminators and (2) calculating the mean of estimated upper and lower bounds. Despite their simplicity, both methods perform well across tasks.

	OOD								ID	
	MCD1		MCD2		COGS		TOP		AMR 2.0	
	Acc	AE ↓	Acc	AE ↓	Acc	AE ↓	Acc	AE ↓	Acc	AE ↓
Maxprob	84.5	26.7	78.0	55.1	97.1	5.7	92.2	19.2	40.6	26.3
AC	82.5	24.7	74.0	51.1	96.6	5.2	85.9	12.9	38.0	23.7
DOC	82.9	25.1	74.3	51.4	96.6	5.2	89.3	16.3	32.8	18.5
ATC	73.0	15.2	56.9	34.0	100.0	8.6	66.0	7.0	15.0	0.7
Maxprob (Oracle)										
Upper.	86.4	-	51.3	-	96.7	-	85.8	-	17.9	-
Lower.	43.7	-	17.2	-	44.3	-	65.2	-	5.8	-
Mean	65.1	7.3	34.3	11.4	70.5	20.9	75.5	2.5	11.8	2.5
Ours (T5)										
$\text{Mean}_{\text{discrim}}$	63.0	5.2	28.8	**5.9**	91.4	**0.0**	75.3	2.3	8.6	5.7
Upper.	70.0	-	36.1	-	92.1	-	82.3	-	18.3	-
Lower.	56.0	-	22.4	-	90.2	-	66.0	-	3.4	-
$\text{Mean}_{\text{bounds}}$	63.0	5.2	29.3	6.4	91.2	0.3	74.2	1.2	10.9	3.4
Ours (Roberta)										
$\text{Mean}_{\text{discrim}}$	59.5	**1.7**	28.9	6.0	91.5	0.1	73.0	**0.0**	14.1	**0.2**
Upper.	65.0	-	35.3	-	92.3	-	82.3	-	25.5	-
Lower.	54.6	-	23.4	-	90.2	-	62.4	-	7.6	-
$\text{Mean}_{\text{bounds}}$	59.8	2.0	29.3	6.4	91.3	0.2	72.3	0.7	16.6	2.3
Gold	57.8	0.0	22.9	0.0	91.4	0.0	73.0	0.0	14.3	0.0

Table 2: Predicted test-set accuracy on semantic parsing tasks. *Upper.* and *Lower.* in the leftmost column refer to predicted upper bound and lower bound. $\textit{Mean}_{\textit{discrim}}$ refers to a point estimate obtained as *the mean of estimated accuracies by all discriminators*. $\textit{Mean}_{\textit{bounds}}$ refers to *the mean of estimated upper and lower bounds* as the point estimate. *Gold* refers to the accuracy evaluated with gold annotations. *Green* numbers refer to valid bounds that capture the gold accuracy, and *Red* numbers refer to invalid bounds.

In addition, we report the *Recall* of our discriminators. Specifically, we report the score for the *Correct* and *Incorrect* labels individually. We define *True Correct (TC)* as instances with an annotation being *Correct* and the prediction being *Correct*, *False Correct (FC)* as instances with an annotation being *incorrect* and the prediction being *correct*. Similarly, we can define *True Incorrect* (TI) and *False Incorrect* (FI). The *Correct-Recall* is calculated by Equation 3; the *Incorrect-Recall* is analogous.

$$CR = \frac{Count(TC)}{Count(TC) + Count(FI)} \quad (3)$$

These recall scores indicate how many correct or incorrect instances can be discriminated, but are not studied by previous works. We propose these metrics as a side contribution, which can be beneficial for downstream uses of the discriminator.

4.4 Results

Correctness of bounds prediction. We first report recall scores of our discriminators in Table 1. For both T5 and RoBERTa discriminators, we can observe that the upper bound achieves the highest *Correct-Recall* score, and the lower bound achieves the highest *Incorrect-Recall* score. This is because these bounds are based on voting mechanisms specifically designed to find correct or incorrect predictions. On many of our datasets, these recall scores approach 100%, which indicates the strong ability of our method to discriminate correctness.

Accuracy of bounds prediction. We then compare the predicted accuracy of our bounds in Table 2 (e.g. semantic parsing), Table 3 (e.g. tagging) and Table 4 (e.g. parsing). We can observe that our predicted upper and lower bounds accurately capture the gold accuracy (i.e. high reliability). This pattern holds for 9 of 10 datasets with T5 discriminators, and for 8 of 10 datasets with RoBERTa discriminators. Even for POS-COGS and MCD2, where this conclusion is not true, the gold accuracy only violates the bounds predicted by a small amount (i.e. 0.4% on POS-COGS and 0.5% on MCD2). Meanwhile, the predicted upper and lower bounds are usually close (i.e. high tightness). Comparing our predicted bounds with *Maxprob (Oracle)*, our bounds are more tight on OOD general-

	OOD				ID	
	POS-Brown		POS-COGS		POS-WSJ	
	Acc	AE	Acc	AE	Acc	AE
Maxprob	87.4	26.4	99.8	14.1	84.7	19.4
AC	80.5	19.5	100.0	14.3	77.4	12.1
DOC	88.0	27.0	98.8	13.1	85.0	19.7
ATC	68.0	7.0	100.0	14.3	61.6	3.7
Maxprob (Oracle)						
Upper.	83.6	-	99.6	-	82.4	-
Lower.	44.5	-	83.3	-	47.9	-
Mean	64.0	3.0	91.4	5.7	65.2	0.1
Ours (T5)						
$\text{Mean}_{\text{discrim}}$	64.1	3.1	87.1	1.4	65.6	**0.3**
Upper.	86.2	-	87.9	-	86.2	-
Lower.	37.8	-	86.2	-	39.5	-
$\text{Mean}_{\text{bounds}}$	62.0	**1.0**	87.1	1.4	62.9	2.4
Ours (Roberta)						
$\text{Mean}_{\text{discrim}}$	62.6	1.6	86.8	1.1	65.8	0.5
Upper.	88.0	-	87.6	-	89.9	-
Lower.	31.8	-	86.1	-	36.7	-
$\text{Mean}_{\text{bounds}}$	59.9	1.1	86.8	**1.1**	63.3	2.0
Gold	61.0	0.0	85.7	0.0	65.3	0.0

Table 3: Predicted accuracy on POS tagging tasks.

	OOD		ID	
	Syn-Brown		Syn-WSJ	
	Acc	AE	Acc	AE
Maxprob	48.3	14.5	50.8	13.2
AC	50.8	17.0	52.4	14.8
DOC	48.5	14.7	50.2	12.6
ATC	34.7	0.9	34.0	3.6
Maxprob (Oracle)				
Upper.	32.9	-	33.4	-
Lower.	17.7	-	16.5	-
Mean	25.3	8.5	24.9	12.7
Ours (T5)				
$\text{Mean}_{\text{discrim}}$	36.3	2.5	43.8	6.2
Upper.	57.5	-	64.2	-
Lower.	17.9	-	24.6	-
$\text{Mean}_{\text{bounds}}$	37.7	3.9	44.4	6.8
Ours (Roberta)				
$\text{Mean}_{\text{discrim}}$	40.2	6.4	34.8	2.8
Upper.	64.6	-	68.8	-
Lower.	3.8	-	5.8	-
$\text{Mean}_{\text{bounds}}$	34.2	**0.4**	37.3	**0.3**
Gold	33.8	0.0	37.6	0.0

Table 4: Predicted accuracy on parsing tasks.

ization tasks (e.g. MCD splits and COGS). Note that *Maxprob (Oracle)* can access gold annotations to find a proper bound, which is implausible in practice. Nonetheless, our method still provides better bounds than this oracle method, indicating the effectiveness of our method on OOD tasks.

Obtaining point estimates. We also compare our method with other point estimation methods with two heuristics (e.g. $Mean_{discrim}$ and $Mean_{bounds}$ rows in *Ours*). Although our methods are not specifically designed for point estimation, these estimates substantially outperform previous methods and achieve very low AE scores across all tasks. Our method is especially useful for OOD test sets, where confidence-based methods yield a much larger AE.

5 Discussion

Low performance on constituency parsing. Our method predicts loose bounds on PTB parsing tasks and sometimes yields high AE scores. We conjecture that this is because the PTB training set contains many long output sequences (e.g. linearized parse trees), whose lengths are much larger than the maximal encoding length (e.g. 512) of our language model discriminators. Encoding sequences longer than observed during pretraining has been shown challenging for transformer-based language models (Dai et al., 2019), which leads to an additional challenge for our discriminators. Nonetheless, the gold accuracy is still robustly between the predicted bounds.

The robustness of discriminators. We have seen that our predicted upper and lower bounds can capture the gold accuracy. However, this may not be enough to show the robustness of our method, since we only evaluated it on one overall test set for each parser, while previous works (Garg et al., 2022) collect multiple test sets for each classifier. To investigate the robustness of our method, we create multiple test sets by randomly sampling subsets from the original test set and plot the accuracy of T5 discriminators on OOD test sets in Figure 4. See Appendix D for full results of both T5 and RoBERTa discriminators.

According to the results, we can observe that our predicted bounds robustly capture the gold accuracy with regard to different sizes of randomly sampled test sets. On COGS, POS-COGS and TOP, a small test set gives a large confidence interval. We consider this is because their test sets contain some extremely difficult examples for the parser, which could result in a challenging subset and yield a low accuracy. Despite this, our discriminators capture the difficulty of such challenging subsets and shares similar confidence intervals as the gold accuracy.

(a) MCD1 (b) MCD2 (c) COGS

(d) TOP (e) POS-COGS (f) POS-Brown

Figure 4: Predicted accuracy w.r.t. the number of test-set instances. For each subset we randomly sample 50 times and show its confidence interval with 95% confidence.

Training models to predict another model's accuracy. The main idea of our method is to train one model (i.e. discriminator) to predict a base model's accuracy (i.e. parser). Previous work estimates model accuracy using the parser's confidence scores, which can be misleading in OOD tasks due to overconfidence, as shown in Section 4.4. On the other hand, Kim et al. (2021) find that despite the parser's poor performance on OOD data, categorizing the parsing task into a classification task enables their classifier to generalize to OOD data to some extents. Hence, we directly train a classifier (i.e. discriminator) to assess correctness of the prediction. Training discriminators to evaluate another model's prediction has been studied in various fields, including adversarial learning (Goodfellow et al., 2014), error detection (Chen et al., 2023) and reranking (Yin and Neubig, 2019), where discriminators are used to improve the base model's accuracy. Our work differs in using discriminators to predict bounds of the base model's accuracy.

Despite the impressive performance of our discriminators, training them relies on incorrect parser outputs (i.e. errors) on the parser's training set. This raises concerns about the generalization of discriminators to errors made in OOD data by the parser. Our results empirically show that discriminators can generalize to such errors, but further investigation is needed to understand the specific scenarios where this method fails and the underlying reasons. Since the discriminators' performance still lags behind on particular tasks, it is worth exploring discriminators' confidence scores to improve the estimated accuracy. We leave this for future study.

6 Conclusion

We propose a method to predict *upper and lower bounds* for the accuracy of a model on unlabeled and possibly OOD data. To do this, we first train multiple correctness discriminators by finetuning pretrained language models, and then ensemble discriminator predictions through a special voting mechanism. Our experiments show that our predicted bounds reliably capture gold accuracy across a variety of in-distribution and out-of-distribution tasks including semantic parsing, tagging and constituency parsing tasks, and the upper and lower bounds are usually tight. Although our method is not specifically designed for point estimation, simple heuristics (e.g. using the mean of bounds as estimated accuracy) based on our method can substantially outperform previous methods, which indicates the effectiveness of our method.

For the future, we will explore the use of our discriminators to improve model performance on tasks evaluated in this paper. For example, given unlabeled OOD sentences and a parser, our lower bound can be used to detect instances with a high *Correct-Precision*, as training data to improve the parser. It would be interesting to expand our predic-

tion of hard upper and lower bounds to a Bayesian model that predicts a probability distribution over accuracies. Finally, it would be useful to extend our method to predicting accuracy in terms of other metrics (e.g. parsing f-score or SMATCH).

Limitations

In order to train the discriminators, we extract negative training instances from the beam of a partially finetuned T5 model. Our method is therefore not applicable in situations where we cannot easily access the beam, e.g. when trying to estimate the accuracy of a closed language model.

Training the discriminators incurs a computational overhead, compared to training only the parser. In the experiments reported above, with five discriminators per ensemble, the training time is increased roughly by a factor of ten. However, once an ensemble has been trained, it can be applied across many unlabeled test sets for the same task.

Finally, the discriminator ensembles in our approach currently vote only at the level of entire test instances, which means that we can only use instance-level evaluation measures such as exact match. As we discussed in the conclusion, extending our approach to other evaluation measures seems like a very useful topic for future research.

Acknowledgements

We appreciate Mareike Hartmann, Megan Dare, Blerta Veseli and Xudong Hong for their insightful feedback to this paper. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) through the project KO 2916/2-2.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Frédéric Blain, and Lucia Specia. 2018. Combining quality estimation and automatic post-editing to enhance machine translation output. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 26–38, Boston, MA. Association for Machine Translation in the Americas.

Shijie Chen, Ziru Chen, Huan Sun, and Yu Su. 2023. Error detection for text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11730–11743, Singapore. Association for Computational Linguistics.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5090–5100, Online. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753, Melbourne, Australia. Association for Computational Linguistics.

Harvey Yiyun Fu, Qinyuan Ye, Albert Xu, Xiang Ren, and Robin Jia. 2023. Estimating large language model capabilities without labeled test data.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C Lipton, Behnam Neyshabur, and Hanie Sedghi. 2022. Leveraging unlabeled data to predict out-of-distribution performance. *arXiv preprint arXiv:2201.04234*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. 2021. Predicting with confidence on unseen distributions. In

Proceedings of the IEEE/CVF international conference on computer vision, pages 1134–1144.

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.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium. Association for Computational Linguistics.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. 2021a. Assessing generalization of sgd via disagreement. *arXiv preprint arXiv:2106.13799*.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021b. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696, Online. Association for Computational Linguistics.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*.

Juyong Kim, Pradeep Ravikumar, Joshua Ainslie, and Santiago Ontanon. 2021. Improving compositional generalization in classification tasks via structure annotations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–645, Online. Association for Computational Linguistics.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. 2020. Calibrated language model fine-tuning for in- and out-of-distribution data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1326–1340, Online. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882, Stockholmsmässan, Stockholm Sweden. PMLR.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Bingzhi Li, Lucia Donatelli, Alexander Koller, Tal Linzen, Yuekun Yao, and Najoung Kim. 2023. Slog: A structural generalization benchmark for semantic parsing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Denis Lukovnikov, Sina Daubener, and Asja Fischer. 2021. Detecting compositionally out-of-distribution examples in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 591–598, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Juri Opitz and Anette Frank. 2019. Automatic accuracy prediction for AMR parsing. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 212–223, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. 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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Vincent Van Asch and Walter Daelemans. 2010. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 31–36, Uppsala, Sweden. Association for Computational Linguistics.

Neeraj Varshney and Chitta Baral. 2023. Post-abstention: Towards reliably re-attempting the abstained instances in QA. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–982, Toronto, Canada. Association for Computational Linguistics.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Yuekun Yao and Alexander Koller. 2022. Structural generalization is hard for sequence-to-sequence models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5048–5062, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4553–4559, Florence, Italy. Association for Computational Linguistics.

Yaodong Yu, Zitong Yang, Alexander Wei, Yi Ma, and Jacob Steinhardt. 2022. Predicting out-of-distribution error with the projection norm. In *International Conference on Machine Learning*, pages 25721–25746. PMLR.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

A Dataset details

We introduce details of our used datasets here. Statistics of datasets are reported in Table 5. The license of datasets are reported in Table 6.

COGS (Kim and Linzen, 2020) is a synthetic English semantic parsing task. The task input is a sentence and the output is a logical form (e.g. *The baby on a tray in the house screamed.* → `scream(agent=*baby(nmod.on=tray(nmod.in=*house)))`). It provides a training set generated by a probabilistic context-free grammar (PCFG) and a OOD test set with 21-typed data, which are generated by different PCFGs to test the different generalization abilities of the parser.

POS-COGS (Yao and Koller, 2022) is a synthetic English part-of-speech tagging task generated based on COGS. The task input is a sentence and the output is the POS tag sequence (e.g. *The baby on a tray in the house screamed.* → `Det N P Det N P Det N V`). POS-COGS shares the same split of train and test sets as COGS.

CFQ (Keysers et al., 2020) is a synthetic English semantic parsing task. The task input is a sentence and the output is a SPARQL query (e.g. *Did M0 ' s writer write M1 and M2* → `SELECT count(*) WHERE {?x0 film.writer.film M0...}`). We use the MCD1 and MCD2 splits of CFQ, where the test set is designed to compositionally diverge from the training set but share similar atom distributions.

TOPv2 (Chen et al., 2020) is a natural English semantic parsing task. The task input is a sentence and the output is a hierarchical semantic representation (Gupta et al., 2018) (e.g. *Will there be snowfall this week?* → `[in:get_weather will there be [sl:weather_attribute snowfall] [sl:date_time this week] ?]`). The TOPv2 training set consists of data from multiple domains including two low-resource domains (e.g. *reminder* and *weather*), and the test set consists of data from the two domains to test low-resource domain

Dataset	Split	# train	# dev.	# test	# gen	Vocab. size	Train len.	Test len.	Gen len.
COGS	-	24155	3000	3000	21000	809	22/48	19/40	61/144
CFQ	MCD1	95743	11968	11968	-	171	29/133	30/103	-
	MCD2	95743	11968	11968	-	171	29/133	30/103	-
TOP	weather	84372	-	484	-	29462	51/82	21/60	-
AMR 2	-	36251	1368	1371	-	79651	216/615	159/583	-
POS-WSJ	-	39832	1700	2416	-	46378	141/141	67/67	-
POS-Brown	-	-	-	24243	-	29564	-	172/172	-
POS-COGS	-	24155	3000	3000	21000	753	22/21	22/21	61/60
Syn-WSJ	-	39832	1700	2416	-	46404	141/903	67/429	-
Syn-Brown	-	-	-	24243	-	29596	-	172/1135	-

Table 5: Statistics for all our datasets. # denotes the number of instances in the dataset. Vocab.size denotes the size of vocabulary for the dataset, which consists of input tokens and output tokens. Train.len denotes the maximum length of the input tokens and output tokens in the train set. Test.len and Gen.len denote the maximum length in the test and generalization set.

Dataset	License
COGS	MIT
CFQ	CC-BY
TOP	CC-BY-SA
AMR 2	LDC
PTB	LDC
POS-COGS	MIT

Table 6: Licenses for used datasets.

adaptation ability of the parser. We focused on the*weather* domain in our experiments.

AMR 2.0 (Banarescu et al., 2013) is an English semantic parsing task. The input is a sentence and output is an abstract meaning representation (e.g. *I will stick around until the end*→`(stick-around-03 :ARG0(i) :time(until :op1(end-01))))`.

Penn Treebank 3 (PTB) (Marcus et al., 1993) is an English constituency parsing task. The input is a sentence and the output is the constituency parse tree (e.g. *Vice President* → `(TOP(NP(NNP Vice)(NNP President))))`.

B Training details

B.1 Hyperparameters

Parser. We finetune *t5-base*[2] (220M parameters) as our parser for all tasks. We use Adam (Kingma and Ba, 2015) as the optimizer. For most tasks, the learning rate is set to 1e-5. For CFQ, AMR, PennTreebank tasks, the learning rate is set to 1e-4 to make the training faster. For tasks that provide a development set, early stopping is used and the best checkpoint is selected based on the evaluation metrics on the development set. Otherwise, the checkpoint at the end of training is used to report results. For AMR, the evaluation metric is Smatch F1 score. For syntactic parsing, the evaluation metric is EVALB F1 score[3]. For other tasks, exact match accuracy is used as the evaluation metric. We use weight decay 1e-3 for all datasets. No learning rate scheduler is used for all experiments. During evaluation, we use beam search with beam size 4.

Discriminator. We finetune *t5-base* (220M parameters), *roberta-base* (125M parameters) and *Vicuna-7b-v1.5*[4] (7B parameters) as our discriminators. To collect training data, we use the first 5 checkpoints of the parser and validate them on the parser's training set. We select negative examples from beam predictions of these checkpoints as the training data of the discriminator. If a task provides an in-distribution development set for the parser, we use the same method to create the development set for the discriminator.

For T5, we follow the same hyperparameter settings described for T5 parser. For RoBERTa, we adopt learning rate 1e-5 for all tasks except PTB. On PTB tasks, the learning rate is set to 5e-5. For both T5 and RoBERTa, we validate the AUC score on the development set to select the best checkpoint of the discriminator when a development set is available. Otherwise, we train the discriminator until its training loss converges with fixed steps. Note that although CFQ provides an out-of-distribution development set, we did not use it since we assume we do not have the access to the OOD data.

We use QLoRA (Dettmers et al., 2023) to finetune Vicuna discriminators. For datasets except POS-COGS, the learning rate is set to 3e-4. For POS-COGS dataset, the learning rate is set to 2e-5. We use linear scheduler with warmup as our learn-

[2] https://huggingface.co/t5-base

[3] https://nlp.cs.nyu.edu/evalb/

[4] https://huggingface.co/lmsys/vicuna-7b-v1.5

ing rate scheduler. We use weight decay 1e-3 for all datasets. For LoRA, we set the rank value to 8, the alpha value to 32 and the dropout value to 0.1.

B.2 Other details

We use AllenNLP (Gardner et al., 2018) to implement T5 and RoBERTa finetuning and Huggingface (Wolf et al., 2020) to implement Vicuna finetuning. Experiments are run on Tesla A100 GPU cards (80GB). Table 8 shows the training time cost to train a single discriminator on one GPU.

C Results of Vicuna-based discriminators

Here we report the results of Vicuna discriminators in Table 9. In this setting, we finetune *Vicuna-7B* on the same datasets we used in Section 4.1.

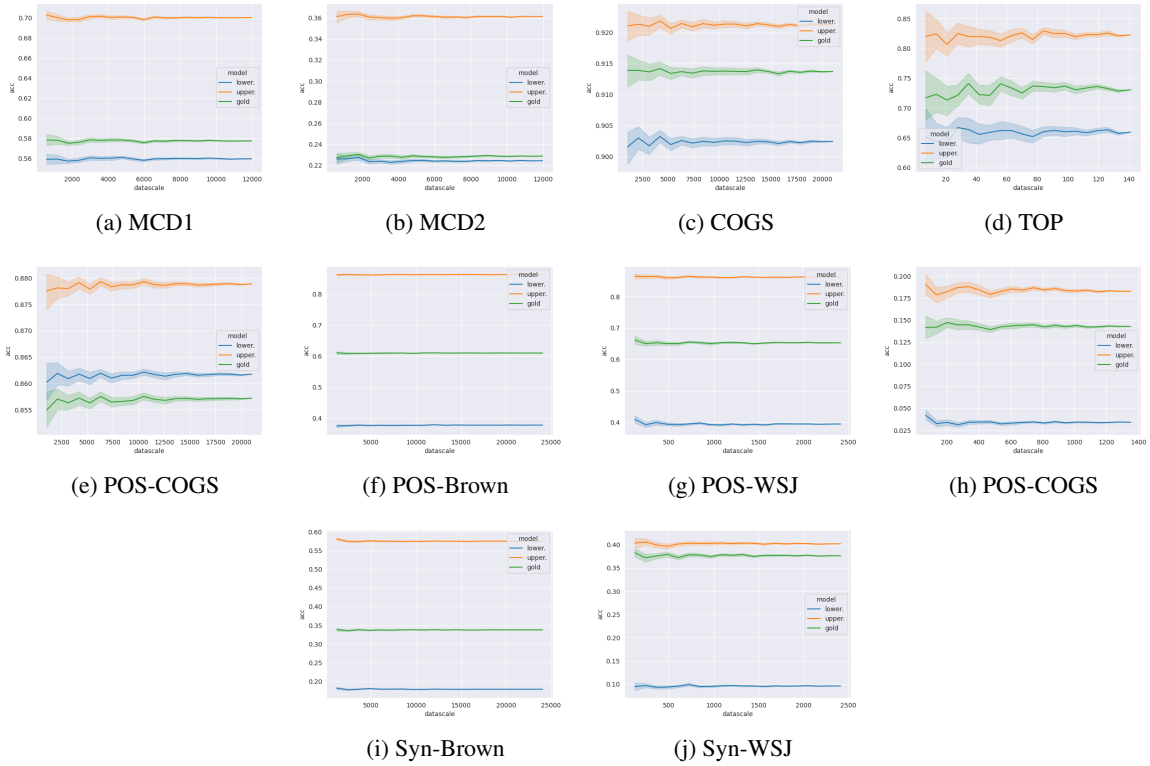
Similar to the observation in Section 4, Vicuna-based discriminators achieve high recall scores on most datasets, indicating that they can still make correct judgements for most instances. We also report estimation errors of this discriminator in Table 7, 10, 11. According to the results, we can observe that Vicuna-based discriminators still achieve very low estimation errors across different datasets. The predicted upper and lower bounds captures the gold accuracy on all datasets. These results are consistent with our observations when using T5 and RoBERTa discriminators, which suggests that our method is robust with regard to different discriminator architectures.

D Results of discriminators on subsets of test sets

We report results of T5 and RoBERTa discriminators on different subsets of the test set in Figure 5, 6. Both discriminators predict lower and upper bounds that capture the gold accuracy robustly, which is consistent with our observation in Section 5.

	OOD								ID	
	MCD1		MCD2		COGS		TOP		AMR 2.0	
	Acc	AE ↓	Acc	AE ↓	Acc	AE ↓	Acc	AE ↓	Acc	AE ↓
Ours (Vicuna)										
$\text{Mean}_{\text{discrim}}$	56.4	1.4	23.6	0.7	93.0	1.6	76.2	3.2	18.3	4.0
Upper.	60.0	-	28.1	-	94.7	-	87.9	-	28.0	-
Lower.	51.5	-	18.4	-	87.8	-	57.4	-	4.2	-
$\text{Mean}_{\text{bounds}}$	55.8	2.0	23.3	0.4	91.3	0.2	72.7	0.3	16.1	1.8
Gold	57.8	0.0	22.9	0.0	91.4	0.0	73.0	0.0	14.3	0.0

Table 7: Predicted test-set accuracy with Vicuna-based discriminators on semantic parsing tasks.



(a) MCD1 (b) MCD2 (c) COGS (d) TOP

(e) POS-COGS (f) POS-Brown (g) POS-WSJ (h) POS-COGS

(i) Syn-Brown (j) Syn-WSJ

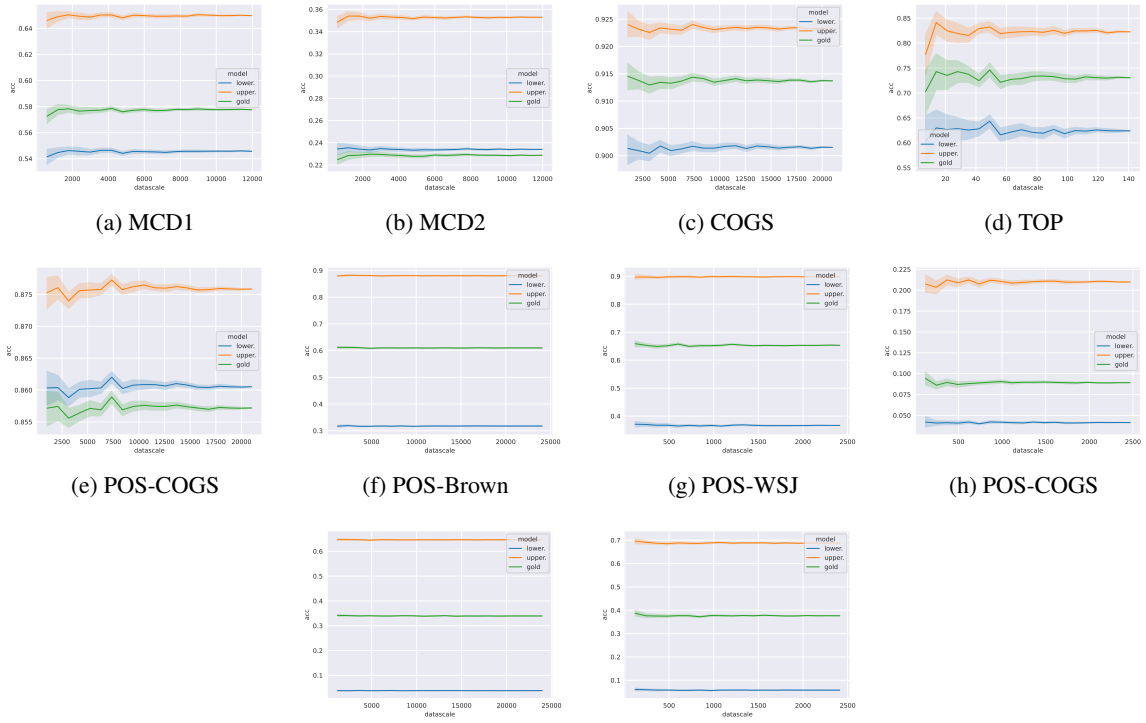
Figure 5: Predicted accuracy by T5 discriminators w.r.t. the number of test-set instances. For each subset we randomly sample 50 times and show its confidence interval with 95% confidence.

	Time (hours)		
Dataset	T5	Roberta	Vicuna
MCD1	10	8	12
MCD2	10	8	12
COGS	3	3	5
Top	2	5	2
AMR	16	18	50
POS-WSJ	16	16	100
POS-COGS	2	3	5
Syn-WSj	15	10	100

Table 8: Training time for our model on each dataset (1 run) in our experiments.

		Single		Upperbound		Lowerbound	
		CR	IR	CR	IR	CR	IR
OOD	MCD1	94.3	96.7	98.7	93.0	88.4	98.9
	MCD2	81.3	94.7	95.9	92.0	68.1	96.4
	COGS	98.8	78.0	99.9	60.2	95.6	95.5
	TOP	82.5	71.1	97.1	36.8	72.8	84.2
	POS-Brown	68.6	60.5	92.1	26.5	24.7	94.6
	POS-COGS	97.9	86.4	98.9	72.6	96.3	91.5
	Syn-Brown	56.9	59.3	88.0	27.5	25.3	75.0
ID	AMR 2.0	51.3	89.9	57.5	76.9	26.4	99.5
	POS-WSJ	70.4	58.2	91.2	26.6	33.6	93.0
	Syn-WSJ	63.9	52.4	91.7	23.8	32.2	68.1

Table 9: Correct recall and incorrect recall scores of Vicuna-based discriminators.



(a) MCD1 (b) MCD2 (c) COGS (d) TOP

(e) POS-COGS (f) POS-Brown (g) POS-WSJ (h) POS-COGS

(i) Syn-Brown (j) Syn-WSJ

Figure 6: Predicted accuracy by RoBERTa discriminators w.r.t. the number of test-set instances. For each subset we randomly sample 50 times and show its confidence interval with 95% confidence.

	OOD				ID	
	POS-Brown		POS-COGS		POS-WSJ	
	Acc	AE	Acc	AE	Acc	AE
Ours (Vicuna)						
$\text{Mean}_{\text{discrim}}$	51.5	9.5	86.8	1.1	55.2	10.1
Single	57.3	3.7	85.3	0.4	60.5	4.8
Upper.	84.8	-	86.6	-	85.0	-
Lower.	17.2	-	83.8	-	24.4	-
$\text{Mean}_{\text{bounds}}$	51.0	10.0	85.2	0.5	54.7	10.6
Gold	61.0	0.0	85.7	0.0	65.3	0.0

Table 10: Predicted accuracy on POS tagging tasks.

	OOD		ID	
	Syn-Brown		Syn-WSJ	
	Acc	AE	Acc	AE
Ours (Vicuna)				
$\text{Mean}_{\text{discrim}}$	51.5	17.7	56.7	19.1
Upper.	77.8	-	82.1	-
Lower.	25.1	-	31.2	-
$\text{Mean}_{\text{bounds}}$	51.5	17.7	56.7	19.1
Gold	33.8	0.0	37.6	0.0

Table 11: Predicted accuracy on parsing tasks.