Calibrated Interpretation: Confidence Estimation in Semantic Parsing

Elias Stengel-Eskin
Johns Hopkins University
elias@jhu.edu

Benjamin Van Durme
Johns Hopkins University
vandurme@jhu.edu

Abstract

Task-oriented semantic parsing is increasingly being used in user-facing applications, making measuring the calibration of parsing models especially important. We examine the calibration characteristics of six models across three model families on two common English semantic parsing datasets, finding that many models are reasonably well-calibrated and that there is a trade-off between calibration and performance. Based on confidence scores across three models, we propose and release new challenge splits of the two datasets we examine. We then illustrate the ways a calibrated model can be useful in balancing common trade-offs in task-oriented parsing. In a simulated annotator-in-the-loop experiment, we show that using model confidence allows us to improve performance by 9.6% (absolute) with interactions on only 2.2% of tokens. Using sequence-level confidence scores, we then examine how we can optimize trade-off between a parser's usability and safety. We show that confidence-based thresholding can reduce the number of incorrect low-confidence programs executed by 76%; however, this comes at a cost to usability. We propose the DidYouMean system (cf. Fig. 4) which balances usability and safety. We conclude by calling for calibration to be included in the evaluation of semantic parsing systems, and release a library for computing calibration metrics.

1 Introduction

Task-oriented dialogue systems (Gupta et al., 2018; Cheng et al., 2020; Semantic Machines et al., 2020) represent one path towards achieving the long-standing goal of using natural language as an API for controlling real-world systems. Such systems transform user requests into executable programs. In line with the principles for human-AI interaction proposed by Horvitz (1999), the goal of these systems is to augment a user's abilities by providing a natural language API for interaction, rather than automating user actions. A foundational principle from Horvitz (1999) is the ability to take rational actions in light of model uncertainty: we should be able to include the expected utilities of different actions in weighing what a system ought to do. When model confidence is low and the model is unlikely to succeed, we might prefer it to defer action or request clarification, while when confidence is high, actions like this may unnecessarily annoy a user.

This sort of reasoning presupposes the model's confidence and its probability of success are correlated, i.e. the model is *calibrated*. In a classification setting, this means the model's confidence is roughly equal to its average accuracy. In Section 4 we investigate the calibration profiles of several common semantic parsing models, asking how well-calibrated modern semantic parsing systems are. We examine six standard models from three model families on two common task-oriented dialogue datasets, using qualitative and quantitative evaluations to measure calibration at the token level. We find that most models are already fairly well-calibrated, but that there is a trade-off between model performance and calibration. In light of this trade-off – and the importance of calibration in semantic parsing applications – we introduce an open-source library for computing calibration metrics and plotting model confidences.[1] We also find that each model's low-confidence examples are challenging for other models; this leads us to propose a confidence-based challenge dataset for both SMCalFlow and TreeDST.[2]

Given the relatively well-calibrated nature of our models, we examine how they could be used in an annotation interface, with a view to balancing the trade-off between annotation cost and correctness. We simulate a human-in-the-loop (HITL) experi-

[1]Metric code: https://github.com/esteng/calibration_metric

[2]Code & data: https://github.com/esteng/calibration_miso

ment where low-confidence tokens trigger a dialogue with with an oracle annotator. The annotator either chooses the correct token from a top-K list or manually inserts it. We find that with interactions on only 2.2% of the total tokens, the oracle annotator can boost exact match accuracy by 9.6% absolute improvement (12.2% relative improvement). The vast majority (85.6%) of interactions involve choosing from the top-5 list, which we expect to be significantly faster than insertion.

A similar trade-off exists between usability and safety in task-oriented user interfaces. We examine how sequence-level model confidence scores can be used to balance this trade-off by reducing the number of incorrect programs executed while also minimizing the number of follow-up user interactions and their cognitive burden. We find that thresholding outputs based on model confidence (i.e. rejecting outputs falling below a tuned threshold) reduces the number of incorrect programs executed by 76% compared to the baseline. However, this comes at a cost to usability; the thresholded system results in about half of the correctly-predicted parses being thrown away. To strike a balance between safety and usability, we introduce the DidYouMean system (cf. Fig. 4). DidYouMean rephrases the original user input conditioned on the predicted parse and asks users to confirm that the paraphrase is equivalent to the original. In a user study of DidYouMean, we obtain an 36% improvement in usability over the thresholded system while still maintaining a 58% reduction in the number of incorrect programs executed compared to simply executing all predicted programs.

Our main contributions are:

- A large-scale examination of model calibration for commonly-used semantic parsing models, including pre-trained models.
- A package for computing calibration metrics and creating calibration plots.
- New EASY and HARD splits of TreeDST and SMCalFlow's validation and test data.
- Our simulated HITL experiment demonstrating the benefits of calibrated confidence scores in an annotation setting.
- The DidYouMean system and a user study demonstrating its potential for balancing usability and safety.

2 Related Work

Having been introduced in the context of weather forecasts (Brier et al., 1950), calibration and confidence estimation have seen a resurgence in machine learning and especially in the context of neural network models. A large body of research has focused on describing the calibration characteristics of different architectures and models, with some work finding neural networks to be relatively well-calibrated (Niculescu-Mizil and Caruana, 2005; Minderer et al., 2021; Carrell et al., 2022) and other research indicating they are not (Guo et al., 2017; Wang et al., 2020). Past work has examined a variety of classification problems, often focusing on binary or multi-class classification (Naeini et al., 2015; Guo et al., 2017; Minderer et al., 2021; Khojah et al., 2022). Some papers have addressed sequential NLP tasks: Jagannatha and Yu (2020) address calibration in structured prediction tasks. More related to our sequence-to-sequence (seq2seq) setting, Kumar and Sarawagi (2019) and Wang et al. (2020) examine calibration in machine translation, both finding models to be over-confident. Related to our focus on dialogue, Mielke et al. (2022) examine and improve text-based calibration (i.e. how much the model's text includes expressions of doubt for incorrect answers) in non-task-oriented dialogue models. In semantic parsing specifically, Dong et al. (2018) develop a method for confidence estimation based on features of the model and the input; our focus is on down-stream applications of model confidence confidence rather than its estimation *per se*. Another key difference is in the type of models used: Dong et al. (2018) use models trained from scratch on individual datasets, while we make extensive use of pre-trained encoders and models. Given Desai and Durrett (2020)'s results indicating that pre-trained encoders are often well-calibrated, it is worth investigating the effect of using such models in the context of semantic parsing, where they have become standard. Related to our use of model confidence in Section 5, Lin et al. (2022) employ a T5 model's token-level posterior probabilities for ensembling neural and grammar-based models for English Resource Grammar (Adolphs et al., 2008).

The DidYouMean system we present relates to the domain of interactive semantic parsing (Li and Jagadish, 2014; Chaurasia and Mooney, 2017; Su et al., 2018), where humans are included in the semantic parsing loop to resolve ambiguities and improve model outputs. Yao et al. (2019) introduce an interactive system in which a parsing agent can opt to ask a user for clarification and further information. The decision to ask for human help is based on model confidence, estimated from the model's posterior. DidYouMean follows in the same spirit, but asks the user to confirm an existing parse before execution rather than generating a question for the user to answer. Our system relates more broadly to the theme of abstention where a model is expected to abstain from making a decision or answering a question at low confidence (Chow, 1957; Xin et al., 2021; Whitehead et al., 2022). Finally, DidYouMean shares a motivation with Fang et al. (2022), who introduce a method for reliably generating summaries of SMCalFlow programs for users to approve or reject. One notable difference from our work is that their work aims to provide users with a post-hoc explanation of what the agent did (which can then be undone) while we focus on resolving misunderstandings *before* execution.

3 Methods

Datasets We examine model calibration on two English task-oriented semantic parsing datasets: SMCalFlow (Semantic Machines et al., 2020) and TreeDST (Cheng et al., 2020). For SMCalFlow, we use version 2.0. For both datasets, we use the data processed by Platanios et al. (2021), who converted TreeDST into a format shared with SMCalFlow. The data format resembles Lisp, with function calls nested by parentheses. Since the test set used by Stengel-Eskin et al. (2022) is not public, we use the SMCalFlow data splits given by Roy et al. (2022), who reserved the validation set for testing and used 10% of the examples from train for validation. This gives us 108,753 training, 12,271 validation, and 13,496 test examples. For TreeDST, we use the official splits given by Platanios et al. (2021), which have train/dev/test splits of 121,652/22,910/22,841.

Models We consider six model variants from three classes. These models fall into two broad paradigms: transductive and seq2seq. Transductive models (Zhang, 2020) treat the parsing problem as a sequence-to-graph task. While the executable programs found in SMCalFlow and TreeDST are expressed as sequential Lisp-like programs, the have an underlying execution graph. Rather than learning to generate the surface form, the transductive approach seeks to directly model the underlying directed acyclic graph (DAG), predicting a sequence of nodes as well as labeled edges. Zhang et al. (2019a) and Zhang et al. (2019b) introduced the MISO transductive parsing framework for predicting DAGs from text inputs. MISO consists of an encoder-decoder model for node decoding paired with a biaffine parser (Dozat and Manning, 2017) for edge prediction, and feature source and target copy operations allowing special tokens (such as names and numbers) to be copied from the input and previously generated tokens to be re-generated (in the case of re-entrancy in the execution graph). We use the state-of-the-art MISO model from Stengel-Eskin et al. (2022), which is based on the transformer-based implementation from Stengel-Eskin et al. (2021b) and has a tuned RoBERTa (Liu et al., 2019) encoder as a feature extractor. MISO has 127 million (M) parameters.

The seq2seq paradigm for semantic parsing instead directly models the output sequence. While predicting the syntactic nuances of a parse (e.g. generating the correct number of closing parentheses) can be challenging, seq2seq models are better able to leverage large pre-trained Transformers, often enabling them to outperform methods like MISO that have stronger inductive biases. We use the SMCalFlow and TreeDST parsers presented in BenchClamp framework Roy et al. (2022); specifically, we examine T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) model architectures. Both are large encoder-decoder Transformers (Vaswani et al., 2017) pre-trained with self-supervised objectives. We examine T5-small (60M parameters), T5-base (220M), T5-large (770M), BART-base (139M), and BART-large (406M). Note that while BenchClamp allows for constrained decoding according to an synchronous context-free grammar, restricting the model to producing only valid parses, we choose to decode in an unconstrained fashion. This choice is motivated by the fact that the constrained decoding process intervenes on the output logit space, zeroing out invalid continuations and renormalizing; this would affect its comparability to MISO in terms of calibration. Furthermore, in the high-resource setting we examine, Roy et al. (2022) obtained only minor performance benefits from constrained decoding.

For both MISO and BenchClamp models, we follow previous work in using the previous 2 turns as input if available. Thus, each datapoint consists of an input $X = (\mathcal{U}_0, \mathcal{A}_0, \mathcal{U}_1)$ and an output program $\mathcal{P}$, where $\mathcal{U}_0$ is the previous user utterance (if it exists), $\mathcal{A}_0$ is an automatically-generated agent response to the previous utterance, and $\mathcal{U}_1$ is the current user command.

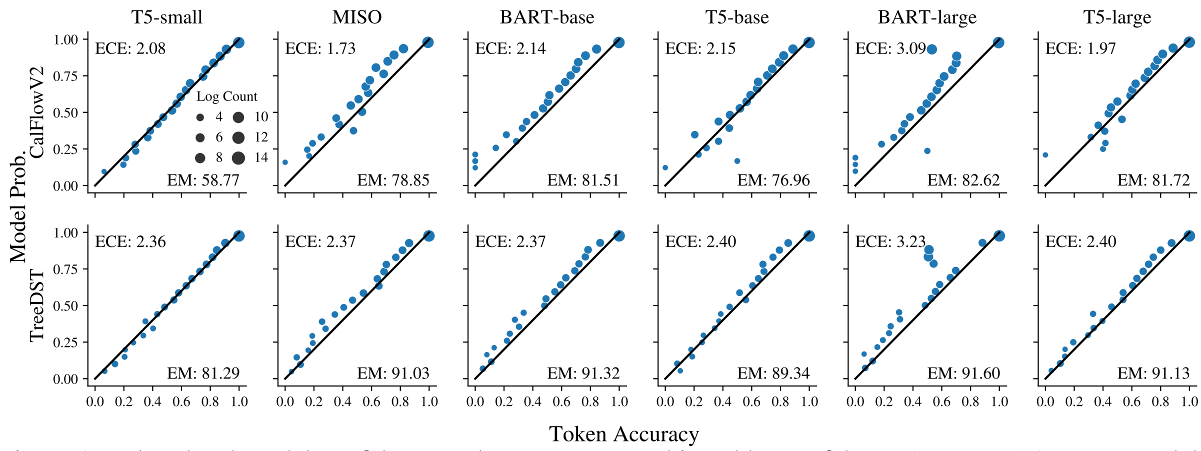

Figure 1: Token-level model confidence and mean accuracy, binned by confidence ($n_{bins} = 20$) across models (sorted by size) and datasets. Point size reflects the number of tokens in the bin. Points above the line reflect overconfidence, while those below reflect underconfidence. Exact Match accuracy (EM) and Expected Calibration Error (ECE) (lower is better) are given. All models show relatively low ECE; MISO has low error on both datasets.

Metrics We evaluate our models' semantic parsing ability using exact match accuracy. For calibration, we use expected calibration error (ECE) (Naeini et al., 2015; Guo et al., 2017). To compute ECE, predictions are binned by the model's confidence. Then, the average accuracy for each bin is computed; the difference of those bins from the line $y = x$, weighted by the size of the bin,(see Fig. 1) is the error. Let $\hat{\mathcal{Z}}$ be the distribution over the output vocabulary $\mathcal{V}$ predicted by the model, and let $\hat{C} = \max \hat{\mathcal{Z}}$, $\hat{Y} = \text{argmax } \hat{\mathcal{Z}}$. Let Y be the true class indices, define a binary accuracy vector A s.t. $a_i = \delta(\hat{y}_i, y_i)$. After binning $\hat{Y}$ into N bins $\mathcal{B}$, $ECE(\mathcal{B})$ is defined as:

$$ECE(\mathcal{B}) = \sum_{i=1}^{N} \frac{|\mathcal{B}_i|}{N} \left| \frac{\sum_{j \in \mathcal{B}_i} a_j}{|\mathcal{B}_i|} - \frac{\sum_{j \in \mathcal{B}_i} c_j}{|\mathcal{B}_i|} \right| \quad (1)$$

In other words, ECE is the mean absolute error between each bin's average confidence and average accuracy. In addition to ECE, we qualitatively analyze calibration by plotting the average accuracy against the bin confidence. To encourage calibration to be measured in semantic parsing, we release our metric and plotting library as a Python package.

4 Calibration and Accuracy

To be able to measure token-level accuracy against a reference program at time t, all predicted tokens at timesteps $1, \ldots, t-1$ must match the reference program's prefix; thus, we use forced decoding, feeding the model the gold prefix up to the current timestep. In Fig. 1 we show the token-level calibration plots and ECE scores for all models and datasets. Note that in our plots, the probability assigned by the model is shown on the y-axis, and the accuracy on the x-axis. While this is non-standard, it leads to a more natural interpretation of the plot w.r.t. the line $y = x$: all points above the line are overconfident bins (confidence > accuracy); those under the line are underconfident. The models here are ranked by size. The size of each point is based on the log of the number of elements in that bin (following Mielke et al. (2022)); by far the largest bin for all models is the most confident bin. The dominance of the largest bin is to be expected from the Exact Match (EM) accuracy results reported in Fig. 1; for a model to achieve high accuracy, all its output tokens must exactly match all of the reference tokens on most of the programs, i.e. the vast majority of tokens must be predicted correctly. We observe that the weighted nature of ECE leads to some counter-intuitive results: for example, that MISO is better-calibrated than T5-small despite the plot for T5-small appearing to be more aligned with $y = x$. The discrepancy is due to the dominance of the largest, most confident bin. This supports the use of both quantitative metrics and qualitative examination, and aligns previous critiques of ECE (Ovadia et al., 2019).

We note that all models are relatively well-calibrated. We see that the smallest variants of models with the same architecture and pre-training data (e.g. T5-small, BART-base) tend to be better-calibrated than their larger equivalents (T5-base, T-large, BART-large) but that this does not hold in general, as T5-base is worse than or equivalent to T5-large. In other words, calibration is not always inversely correlated with model scale. In Fig. 2, we plot the trade-off between accuracy and calibration error for SMCalFlow models on the

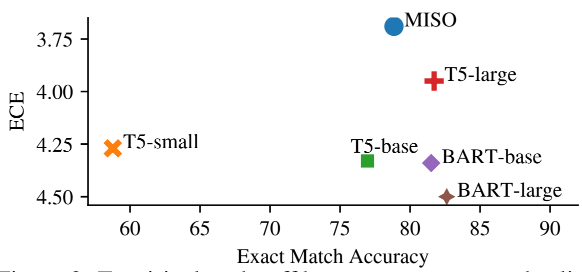

Figure 2: Empirical trade-off between accuracy and calibration. Note that the Y-axis is flipped (lower calibration error is better). Several models exist at the Pareto front.

test set. We see that models with high accuracy tend to also have higher calibration error; models with low error often have lower accuracy. Overall, MISO and T5-large perform best when considering both metrics. Although BART variants typically have the highest accuracy scores, they are also the least calibrated. Importantly, their calibration curves are non-monotonic, which is particularly bad. A monotonically-distorted model can be improved via isotonic regression on the validation set (Zadrozny and Elkan, 2002); this is not the case for non-monotonic distortions.

4.1 Easy and Hard Subsets

Fig. 1 shows that for many of the models we examine, token-level confidence is well-correlated with accuracy. However, we measure this confidence using a forced decode of the gold program prefix, which is unavailable at test-time. If sequence-level confidence is also well-estimated, then we can predict (on average) how likely an example is to be correct. Low-confidence examples thus represent programs that models are likely to get wrong. After measuring sequence-level confidence, we create EASY and HARD splits of high- and low-confidence programs. These splits follow in the spirit of adversarial filtering (Zellers et al., 2018) but are based on confidence rather than accuracy.

Sequence-level Calibration To obtain sequence-level confidence scores, we need a method for aggregating token-level confidence into sequence-level confidence. We explore two aggregation functions: `min` and `mean`. These operate over the token-level confidences generated during decoding. Note that here we measure exact match accuracy at the sequence level rather than the token level. Table 1 shows the ECE for sequence-level confidence scores with the two aggregation methods. Given the smaller number of sequences (compared to tokens) we reduce the number of bins to 10. Since that `min` is typically better, especially on BART-large and T5-large, where `mean` results high overconfidence, we adopt `min` moving forward.

Model	ECE (Min.)	ECE (Mean)
MISO	3.66	3.36
BART-large	5.23	16.38
T5-large	5.88	17.54

Table 1: Sequence-level ECE for our best models.

The HARD subset contains the union of examples for which the *sequence*-level confidence falls below a threshold of 0.6,[3] with all other examples marked EASY. We take the minimum probability across the decoded tokens for the sequence-level confidence.

Table 2 shows the percentage of test data below the threshold c for each model. On both datasets, MISO has by far the most examples below c and contributes more to the union of all low-confidence examples. TreeDST has greater overlap between the different models than SMCalFlow.

Model	HARD (SM)	HARD (Tr)
MISO	21.01%	13.09%
BART-large	6.09%	8.28%
T5-large	8.49%	7.68%
Union	26.64%	13.81%

Table 2: Percentage of test examples labeled as HARD by each model for SMCalFlow (SM) and TreeDST (Tr).

Table 3 shows the accuracy of each model on our subsets. We see much lower performance across all models on the HARD subset and much higher performance on EASY. The performance of MISO is much lower than that of the other models; the difference is much larger than the performance difference in Fig. 1. This is partly due to MISO contributing the largest percentage of low-confidence examples to HARD (cf. Table 2) – low-confidence examples are often more likely to be misclassified (Hendrycks and Gimpel, 2016). We release our HARD and EASY subsets of the validation and test data for both SMCalFlow and TreeDST to act as challenge datasets for future work.

These results show that MISO is much stronger on EASY examples than HARD examples; we explore this further in Appendix C, where we ensemble MISO and BART-large. Examining these splits more closely, we find that what makes examples hard differs between datasets. We consider three factors: the number of tokens in user utterance $\mathcal{U}_1$,

[3]This threshold was chosen based on the intuition that for a well-calibrated model, a cut-off of 0.6 represents getting roughly a 3/5 chance of being correct.

Dataset	Model	HARD	EASY
SMCalFlow	MISO	39.80	92.19
	BART-L	54.62	92.20
	T5-L	50.87	92.27
TreeDST	MISO	37.91	97.66
	BART-L	53.85	97.83
	T5-L	51.22	97.70

Table 3: Exact match accuracy on the EASY and HARD subsets for all models. L indicates "large" variant. All models perform significantly worse on HARD.

whether $\mathcal{U}_1$ is composed of multiple sentences, and the percentage of values in the target program that can be copied from the input (including the context $\mathcal{U}_0$ and $\mathcal{A}_0$). Table 4 shows the differences between EASY and HARD examples in both datasets; all differences were significant ($p < 0.05$, Welch's t-test). For SMCalFlow, the HARD examples have longer inputs with multiple sentences. For TreeDST on the other hand, HARD examples are more likely to have uncopiable tokens. This is likely due to the fact that TreeDST does not have salience operations, meaning that values in a program can refer to utterances outside of the three-turn context window, effectively making those values impossible to predict.

Dataset	Metric	EASY	HARD
SMCalFlow	input length	7.59	11.15
	$\% > 1$ sent.	3.94%	9.17%
	% copiable	47.2%	50.1%
TreeDST	input length	8.41	8.16
	# sents.	8.14%	10.40%
	% copiable	57.1%	38.4%

Table 4: Differences between EASY and HARD splits in input length, num. of input sentences, and perc. of copiable tokens.

5 Human-in-the-Loop Simulation

In the production settings they were designed for, datasets like SMCalFlow and TreeDST are not meant to be static objects: they are constantly evolving as new functionalities are added and shortcomings are revealed. In this evolution, expert annotators use custom tools to annotate new data for training and evaluating models. The expensive and time-consuming nature of this process can be mitigated by the use of predictive parsing models which suggest parses for new utterances. This can be thought of in similar terms as recent IDE-integrated predictive code models such as Codex (Chen et al., 2021). As with code, the output of the parsing model can be incorrect, especially given out-of-distribution inputs; debugging these outputs can be difficult and time-consuming. Given human tendency to accept automated decisions (Cummings, 2004), we also need to ensure that annotators are not introducing errors by placing too much trust in the model.

One solution is to rely on the confidence of a well-calibrated model as a proxy for how likely the prediction is to be correct. For example, we can alert annotators to low confidence predictions and ask them to intervene. Note that calibration here is key in order to set a valid threshold for alerting the annotator. Using a threshold, we can prioritize time or correctness: a higher threshold would result in more annotator-model interactions, increasing the amount of time spent on the task but also increasing the correctness of the programs – reducing the need for debugging – while a lower threshold would reduce the number of interactions but also lower the accuracy.

Since we do not have access to expert SMCalFlow annotators, we run a simulated human-in-the-loop (HITL) experiment. We simulate an oracle annotator who always provides a correct answer by using the gold annotations provided in the dataset. Specifically, for a given input, we decode the output tokens of a predicted program $o_0, \ldots o_n$ normally as long as predictions are confident (above a given threshold). If at time t the confidence $p(o_t)$ falls below a set threshold, we attempt to match the decoded prefix $o_0, \ldots, o_{t-1}$ to the gold prefix $g_0, \ldots g_{t-1}$. If the prefixes do not match, we count the example as incorrect. If they do match, we replace o_t with g_t, the gold prediction from our oracle annotator, and continue decoding. For our experiment, we use MISO, which is the best-calibrated model for SMCalFlow. We consider three metrics in this simulated experiment:

- The exact match accuracy of the decoded programs (higher is better).
- The percentage of total tokens for which we have to query an annotator (lower is better).
- The percentage of uncertain tokens (below the threshold) for which the gold token g_t is in the top 5 predictions at timestep t. Here, higher is better, as selecting a token from a list of candidates is typically faster and easier than manually writing out the token.

Results and Analysis Fig. 3 shows our three metrics as the threshold is increased in increments of 0.1. We see that accuracy grows exponentially with a higher threshold, and that the percentage of tokens for which an annotator intervention is required grows at roughly the same rate. The exponential growth reflects the distribution of token confidences, with most tokens having high confidences. Finally, we see that while at low confidences, most tokens must be manually inserted, the rate at which they are chosen from the top 5 list rapidly increases with the threshold.[4] This suggests that the increased number of annotator interactions required at higher thresholds may be mitigated by the fact that many of these interactions would result in a choice from the top 5 list, reducing the time annotators would need to spend.

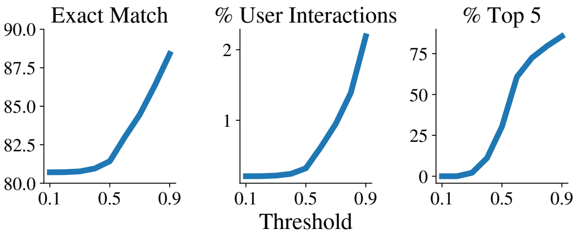

Figure 3: Simulated annotator-in-the-loop results across increasing confidence thresholds. Higher thresholds produce more correct programs but cost more in terms of annotator interactions. A greater percentage of high-confidence tokens can be selected from a top-5 list.

6 User Correction via DidYouMean

Section 5 showed that token-level confidence scores can be used to balance speed and correctness in an annotation interface. A similar trade-off exists for user interfaces using semantic parsing models between safety and usability. Here, we define safety as rejecting unsuccessful programs *before executing them*. This is a strict definition in the context of digital assistants (the domain SMCalFlow and TreeDST were designed for) as a mistaken action by the digital assistant can still be rejected or undone by the user. However, this assumption cannot always be made: imagine that rather than controlling a digital assistant, a user is guiding a robot via language commands (e.g. Winograd (1972); Lynch and Sermanet (2020); Stengel-Eskin et al. (2021a); Lynch et al. (2022); Nair et al. (2022)). In this setting, the agent may be unable to simulate actions which can have irreversible physical consequences. This could become dangerous to humans in other settings where robots are commonly deployed, such as warehouses and factories. Safety considerations need to be balanced with usability of the system: an agent that does nothing would be very safe but unusable. To increase usability, an agent might make follow-up requests to a user, like asking for clarification or confirmation. The types of requests the agent makes have varying cognitive load on the user: for example, providing confirmation is less work than rephrasing an instruction.

[4]Qualitatively, many of the lowest-confidence predictions are due to misspelled or OOD inputs. In these cases, the model's output distribution often diverges from the gold.

Motivated by these factors, we use SMCalFlow as a domain in which to measure how well we can reject incorrect programs before executing them. Success here is measured by precision, recall, and F score w.r.t. the correctness of the program; in this case, precision is the percentage of accepted programs which were correct, and recall is the percentage of all correct programs which were accepted. We consider F1 – the harmonic mean between precision and recall – as well as F0.5, which upweights precision by a factor of 2. The precision/recall trade-off here can be thought of as a trade-off between safety and usability: a high-precision, low-recall system would be safer have more false negatives, i.e. reject more correct programs, decreasing the usability. A low-precision, high-recall system might be more usable at the cost of false positives, i.e. executing incorrect programs. Note that we do not commit to setting an optimal threshold for this trade-off, since it is task-specific.

As a baseline, we consider the performance of a system that executes everything it predicts. As in Section 5, we can use the calibrated nature of the MISO model to greatly improve safety outcomes by setting a confidence threshold below which we reject programs. However, these safety improvements come at a cost to usability: threshold-based rejection leads to many *correct* programs being rejected. To recover some usability, we introduce the DidYouMean system, a more sophisticated method for filtering low-confidence programs. For a given utterance, DidYouMean presents the user with a paraphrase of the input; the user makes the decision to accept or reject the parse based on this paraphrase. This allows correctly-predicted low-confidence programs to still be accepted and executed, while reducing the user load: making a binary choice to accept a paraphrase is a receptive task, while rephrasing an instruction is a more costly productive task.

Glossing model Ideally, we would like to be be able to show a user a low-confidence parse and ask them if it corresponds to what they meant to say; however, since users are typically unfamiliar with formats like Lispress, we need to present the user with a natural language paraphrase of the candidate parse. We refer to this paraphrase as a *gloss* of the parse; To train a glossing model, we modify Roy et al. (2022)'s seq2seq BenchClamp model: rather than using the user utterance with the previous turn's context $(\mathcal{U}_0, \mathcal{A}_0, \mathcal{U}_1)$ as input and a program $\mathcal{P}$ as output, we take the context and *program* $(\mathcal{U}_0, \mathcal{A}_0, \mathcal{P})$ as the input and user instruction $\mathcal{U}_1$ as the output. We train this model with the same architectures and hyperparameters as the other BenchClamp models; we explore T5-base, T5-large, BART-base, and BART-large.

Evaluating the glossing model Instead of evaluating the glossing model using string metrics such as BLEU (Papineni et al., 2002) which can be noisy, we choose to evaluate the output programs using cycle-consistency. Specifically, we evaluated trained glossing models by glossing programs in the gold test set and then parsing those glosses with a fixed MISO parser. All accuracy scores are reported in Table 5 along with the baseline accuracy obtained by parsing the gold test inputs. The best-performing gloss model is BART-large.[5]

Model	Exact Match
No gloss	78.85
T5-base	79.51
T5-large	82.99
BART-base	82.78
BART-large	84.00

Table 5: Exact match for programs parsed from glossed inputs from the SMCalFlow test set.

DidYouMean system When a low-confidence parse is detected (using the minimum over predicted token confidences) DidYouMean triggers a dialogue with the user in order to recover some usability over simply rejecting all low-confidence parses. Fig. 4 shows the system workflow. High-confidence utterances are simply executed, while low-confidence parses trigger a simple dialogue. The dialogue shows the user a paraphrase of their utterance and asks them to accept or reject it. To find a good paraphrase of each predicted parse, we generate N glosses $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_N$ via beam search and take the one that yields the highest probability of the predicted parse $\hat{\mathcal{P}}$, i.e. $\hat{\mathcal{U}}^* = \text{argmax}_i P_{MISO}(\hat{\mathcal{P}}|\mathcal{U}_0, \mathcal{A}_0, \hat{\mathcal{U}}_i)$. DidYouMean then presents the original utterance $\mathcal{U}_1$ and $\hat{\mathcal{U}}^*$ to the user, who determines whether they are identical or not. If they accept a gloss, we optionally reparse the gloss rather than the original utterance, as glosses often remove misspellings, typos, and other idiosyncracies.

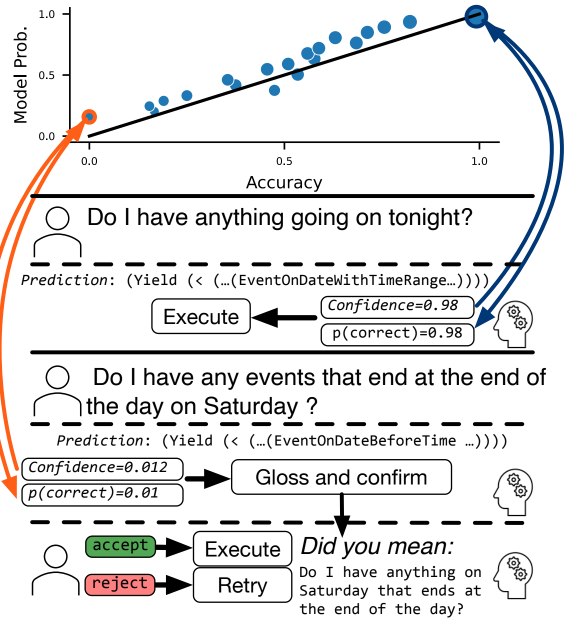

Figure 4: The DidYouMean system. When the model has high-confidence, we simply execute the predicted parse. At low confidences, the probability of a successful parse is low; DidYouMean rephrases the user query based on the predicted parse and asks a user to confirm the paraphrase. The program is only executed if the paraphrase is accepted.

User study We conduct a static user study of DidYouMean using examples from the SMCalFlow validation set. We sample 100 MISO predictions with a minimum confidence below 0.6, using the same threshold as in Section 4.1. This sample is stratified across 10 equally-spaced bins with 10 samples per bin. We then choose the top gloss for each example from 10 candidates generated by the BART-large glossing model. Annotators were recruited from a list of qualified annotators on Amazon Mechanical Turk and paid \$0.05 per example, averaging to roughly \$16 per hour. The template and instructions used can be seen in Appendix A. We obtained 3 judgments per example;

Annotation Statistics 8 annotators completed at least one judgement, with 4 completing the majority. All 3 annotators agreed on 79% examples,

[5]Note that all glossing models outperform MISO without glosses. This can be explained by the fact that we gloss the inputs from the gold program, which we then evaluate on, allowing for information leakage. We also hypothesize that the gloss model, having been trained on the entire dataset of utterances, averages out many annotator-specific idiosyncracies which may make inputs hard to parse. This result does *not* imply that glosses generated from *predicted* programs would yield better performance when parsed than the user input.

indicating the task is well-formulated. For the remaining examples, we use the majority decision to accept or reject. After majority voting, annotators accepted $68/100$ glosses and rejected 32.

Settings Our baseline is to accept all programs, resulting in the lowest-possible precision and highest-possible recall.[6] Because the model is well-calibrated, we can use a confidence threshold to reject potentially incorrect parses. We tune our threshold on the full validation set using F1; we explore the range $[0.0, 1.0)$ in increments of 0.01, finding 0.40 to be optimal. We also consider two DidYouMean settings: in the "chosen" setting, we use the predicted program $\hat{\mathcal{P}}$ corresponding to an accepted gloss $\hat{\mathcal{U}}^*$ and compare it against the reference program $\mathcal{P}$. In the "re-parsed" setting, we re-parse the accepted gloss $\hat{\mathcal{U}}^*$ and compare the exact match accuracy of that second parse to the reference program $\mathcal{P}$. We hypothesize that allowing users to accept and reject glosses will improve the balance between safety and usability (i.e. F1) over the thresholded system by allowing them to accept correct low-confidence parses.

Setting	FP	P	R	F1	F0.5
Accept	67	0.33	1.00	0.50	0.38
Tuned	16	0.50	0.49	0.49	0.50
Chosen	37	0.46	0.91	0.61	0.51
Re-parsed	28	0.59	0.76	0.66	0.62

Table 6: Number of false positives (FP, lower is better), Precision (P, higher is better), Recall (R, higher is better), and F1 (higher is better) for accepting correct parses and rejecting incorrect parses.

Table 6 shows the results of the user study. Tuning a threshold results yields the best safety outcomes. However, this safety comes at a cost to the usability of the system; a recall of 0.49 indicates that the tuned system accepts only about half of all correct programs. The "tuned" system's low usability is reflected in the F1 and F0.5 scores, which balance precision and recall. The "chosen" system, while better in F1, is comparable to the "tuned" system in F0.5, which takes both usability and safety into account but prioritizes safety at a 2:1 ratio. Users are slightly worse than thresholding in terms of precision but much better in terms of recall, indicating that asking for confirmation can greatly improve usability. However, looking at the rate of false positives, we see a large increase in the number of incorrect programs executed when glosses are chosen as compared to the tuned threshold. When the accepted glosses are re-parsed, we see a shift towards a lower-recall, higher-precision system favoring safety, with fewer incorrect programs being executed than in the "chosen" setting. For both F1 and F0.5, the "re-parsed" system best balances usability and safety.

These results show that a calibrated model can be used with a threshold to greatly improve safety, reducing the number of incorrect programs accepted by 76%. The DidYouMean system allows users to recover some low-confidence programs by accepting and rejecting programs based on their glosses, resulting in the best aggregated scores. Note also that the threshold was tuned on F1 score on the entire dev set. This means that the F1 performance of that tuned system is as high as possible for confidence-threshold-based system. Thus, DidYouMean achieves a balance which cannot be achieved by tuning: simply increasing the threshold would decrease safety and result in a lower F1 than the current threshold of 0.40.

7 Discussion

Section 4 indicates that models are relatively well-calibrated as-is, but we see in Fig. 1 that there is room for growth. Furthermore, Fig. 2 indicates a trade-off between performance and calibration; similar observations have been made in text classification (Khojah et al., 2022). Given the downstream utility of calibration scores, a model's calibration profile should be taken into account when it is evaluated; while Fig. 1 indicates that BART-large is the highest-performing model in terms of exact match, it does quite poorly in terms of calibration. Depending on the setting in which the parsing model is deployed, BART-large may in fact *not* be optimal. We advocate for considering calibration in the standard evaluation suite for semantic parsing models, and release our metric and visualization suite as a standalone package to facilitate these comparisons. We use confidence to identify EASY and HARD subsets, the latter of which are substantially more difficult than the original datasets.

The utility of a well-calibrated model is primarily in being able to rationally balance different factors based on their expected utility (EU). For example, in Section 5 we used token-level confidence scores to balance annotation cost (measured in the number of annotator interactions and the difficulty of those interactions) and correctness. In Section 6

[6]We do not include the opposite baseline (reject all outputs) as this would result in 0 true positives.

we used sequence-level confidence scores to balance usability and safety. If model confidence is a non-monotonic function (e.g. BART-large's confidence for SMCalFlow in Fig. 1) then setting a rational threshold becomes impossible. EU is a standard method for making decisions under uncertainty (Horvitz, 1999), allowing developers to consider the positive utility of correct actions as well as the negative utility of taking incorrect actions or of burdening the user. As Horvitz (1999) notes, "the utility of *asking* a user before performing a desired action is typically smaller than the utility of simply performing a desired action when the user indeed has the goal." How utilities are assigned is an empirical matter, as utilities may vary across application areas or even across users.

Limitations The experiments in Section 6 rely on a glossing model to translate predicted Lispress programs into natural language (NL), since users cannot read Lispress. We approached this by training a Lispress-to-NL model; this approach has several limitations. Neural text generation models often hallucinate outputs, i.e. glosses generated by the model may not be faithful to their corresponding programs. Fang et al. (2022) address via a grammar-based approach to response generation from programs. Their framework generates acceptable and faithful responses in SMCalFlow by introducing grammar-based constraints into the generation model's decoding process. We do not assume access to a grammar but note that DidYouMean does not preclude the use of a grammar for constrained decoding.

Our annotators in Section 6 face the additional challenge of interpreting and choosing glosses. SMCalFlow programs are nuanced and slight variations in input can result in different programs. These nuances are often obscured by the glossing model, resulting in two different programs glossing to roughly identical utterances. Annotators might mistakenly accept glosses from incorrect programs or reject correct glosses; this would be difficult to address even with a faithful translation method; we explore this further in Appendix B.

Our study is limited by the models, datasets, and languages we consider. Firstly, we examine only English datasets, limiting the impact of our results; future work may examine calibration across typologically diverse languages. Additionally, although we consider two datasets and six models, our datasets are limited to Lispress-based programs and our models to Transformer-based models. While these choices are representative of current standards in task-oriented semantic parsing, we encourage broader investigations spanning additional models and datasets. We also examine models only in the highest-resource settings and leave examining how calibration profiles change with the size of a dataset to future work.

In terms of metrics, exact match can often be an unnecessarily strict; a better choice might be execution accuracy, but this would require access to an execution engine. Additionally, ECE can be a brittle metric (Ovadia et al., 2019). It is sensitive to the number of bins, and as seen in Fig. 1 is dominated by the ECE of the largest bin. This obscures a model's true calibration characteristics; we attempt to balance this using qualitative assessments.

We make several limiting assumptions in Section 5 and Section 6. Foremost amongst these is the assumption of access to an oracle annotator in Section 5; clearly, no such annotator exists. Our results may vary when real annotators are brought into the loop. For one, we do not know exactly how choosing from the top-k list will compare to insertion w.r.t. speed. We also do not know how automation bias would affect the top-k list: given that the correct answer is often in the list, real annotators might overly rely on the list and prefer it to token insertion, resulting in incorrect programs.

8 Conclusion

We examined calibration on two datasets across three model families and six models, finding that many models are already well-calibrated and that there is often a trade-off between calibration and performance. We then used model confidence to create a HARD challenge split for SMCalFlow and TreeDST which is hard even for the best-performing models. In Section 5 we illustrated how token-level model confidences could be used in a simulated human-in-the-loop annotation task for a task-oriented parsing formalism. Our experiments in Section 6 extend these results to sequence-level confidences and non-expert users; we found that model confidence could be used to improve the usability-safety trade-off, and introduced the DidYouMean system, which improved usability by asking users to accept or reject predicted programs. We conclude with a call to include calibration in semantic parsing evaluation suites, and release a tool for facilitating this measurement.

References

Peter Adolphs, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Dan Flickinger, and Bernd Kiefer. 2008. Some fine points of hybrid natural language parsing. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

Glenn W Brier et al. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Annabelle Carrell, Neil Mallinar, James Lucas, and Preetum Nakkiran. 2022. The calibration generalization gap. *arXiv preprint arXiv:2210.01964*.

Shobhit Chaurasia and Raymond Mooney. 2017. Dialog for language to code. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 175–180.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders Johannsen. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

Chi-Keung Chow. 1957. An optimum character recognition system using decision functions. In *IRE Transactions on Electronic Computers*, 4, pages 247–254. IEEE.

Mary Cummings. 2004. Automation bias in intelligent time critical decision support systems. *AIAA 1st Intelligent Systems Technical Conference*.

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.

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.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Hao Fang, Anusha Balakrishnan, Harsh Jhamtani, John Bufe, Jean Crawford, Jayant Krishnamurthy, Adam Pauls, Jason Eisner, Jacob Andreas, and Dan Klein. 2022. The whole truth and nothing but the truth: Faithful and controllable dialogue response generation with dataflow transduction and constrained decoding. *arXiv preprint arXiv:2209.07800*.

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.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166.

Abhyuday Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2020, page 2078. NIH Public Access.

Ranim Khojah, Alexander Berman, and Staffan Larsson. 2022. Evaluating n-best calibration of natural language understanding for dialogue systems. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 582–594.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Fei Li and Hosagrahar V Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Zi Lin, Jeremiah Zhe Liu, and Jingbo Shang. 2022. Towards collaborative neural-symbolic graph semantic parsing via uncertainty. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4160–4173.

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. *arXiv preprint arXiv:1907.11692*.

Corey Lynch and Pierre Sermanet. 2020. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. 2022. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*.

Sabrina J Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. 2022. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. 2021. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34:15682–15694.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. 2022. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32.

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.

Emmanouil Antonios Platanios, Adam Pauls, Subhro Roy, Yuchen Zhang, Alexander Kyte, Alan Guo, Sam Thomson, Jayant Krishnamurthy, Jason Wolfe, Jacob Andreas, and Dan Klein. 2021. Value-agnostic conversational semantic parsing. 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 3666–3681, 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.

Subhro Roy, Sam Thomson, Tongfei Chen, Richard Shin, Adam Pauls, Jason Eisner, and Benjamin Van Durme. 2022. Benchclamp: A benchmark for evaluating language models on semantic parsing. *arXiv preprint arXiv:2206.10668*.

Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Elias Stengel-Eskin, Andrew Hundt, Zhuohong He, Aditya Murali, Nakul Gopalan, Matthew Gombolay, and Gregory D. Hager. 2021a. Guiding multi-step rearrangement tasks with natural language instructions. In *5th Annual Conference on Robot Learning*.

Elias Stengel-Eskin, Kenton Murray, Sheng Zhang, Aaron Steven White, and Benjamin Van Durme. 2021b. Joint universal syntactic and semantic parsing. *Transactions of the Association for Computational Linguistics*.

Elias Stengel-Eskin, Emmanouil Antonios Platanios, Adam Pauls, Sam Thomson, Hao Fang, Benjamin Van Durme, Jason Eisner, and Yu Su. 2022. When more data hurts: A troubling quirk in developing broad-coverage natural language understanding systems. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yu Su, Ahmed Hassan Awadallah, Miaosen Wang, and Ryen W White. 2018. Natural language interfaces with fine-grained user interaction: A case study on web apis. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 855–864.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079.

Spencer Whitehead, Suzanne Petryk, Vedaad Shakib, Joseph Gonzalez, Trevor Darrell, Anna Rohrbach, and Marcus Rohrbach. 2022. Reliable visual question answering: Abstain rather than answer incorrectly. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. The art of abstention: Selective prediction and error regularization for natural language processing. 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 1040–1051.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019. Model-based interactive semantic parsing: A unified framework and a text-to-sql case study. 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 5447–5458.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sheng Zhang. 2020. *Transductive Semantic Parsing*. Ph.D. thesis, The Johns Hopkins University.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. 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 3786–3798, Hong Kong, China. Association for Computational Linguistics.

A User study template

Fig. 5 shows the template for the confirmation HIT. The instructions asked users to read the paraphrase produced by the agent and determine whether the agent had correctly understood the user.

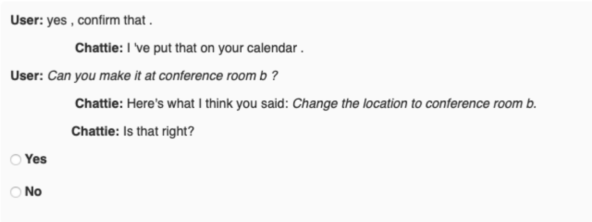

Figure 5: Template for the confirmation HIT.

B Selection HIT

In Section 6 we presented the DidYouMean system, which allowed users to confirm or reject a potential gloss. The gloss was chosen from a larger set of candidates; this naturally raises the question of whether users can directly choose a gloss from the candidate list, rather than accepting or rejecting a single gloss at a time. Here, we examine to what extent it is helpful to users to choose glosses from a list of options. We take a sample of validation programs for SMCalFlow, stratified across 10 evenly-spaced bins from 0.0 to 1.0. Note that this differs from the setting in Section 6, where the maximum bin was 0.6. We then present the top predictions using nucleus sampling (Holtzman et al., 2019); the number of predictions depends on the confidence of the model. Specifically, we set a cutoff of 0.85 and add predictions until the sum of the minimum token probabilities across predictions is greater than 0.85. We limit ourselves to a maximum of 10 predictions, even if the cutoff is not reached.

The template is seen in Fig. 6; each example gives the gloss as well as a rounded confidence score for the predicted program. Annotation was run with the same group of annotators as the experiments in Section 6; annotators were paid \$0.11 per example, or about \$16 per hour of annotation. Each example was annotated by a single annotator, all of whom had been vetted in a pilot annotation task. Annotators were instructed to help a robot named SvenBot, who had recently learned English and was not confident about its understanding, disambiguate between several options. The interface contained a text box where annotators could optionally manually re-write the input; this was only to be done in cases where *none* of the options reflected the intended meaning of the original utterance.

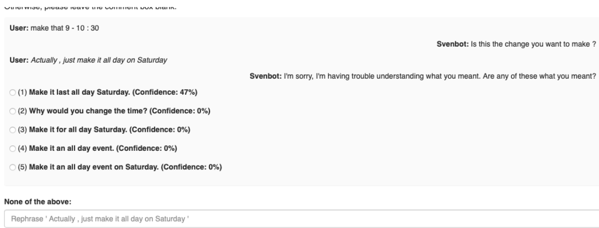

Figure 6: Template for the selection HIT.

Of the 100 examples sampled, annotators manually rewrote 7 and chose from the top list for the other 93. Ignoring the rewritten examples, 39 model predictions were incorrect and 54 were correct; by choosing glosses, annotators correct 5 incorrect predictions. However, they also inadvertently changed 4 correct predictions to incorrect glosses. Figure Fig. 7 shows the exact match accuracy before and after annotators selected glosses at each confidence bin. At low confidence, we see very minor increases on the order of a single program being corrected. At high confidences, annotators generally have only one or two options, and are able to choose the correct one, resulting in similar performance to nucleus decoding. However, at medium confidences, annotators often choose the wrong gloss, leading to lower performance. Qualitatively, these incorrect choices are primarily driven by glosses which appear to be paraphrases of each other, but in fact correspond to subtly different Lispress programs. This highlights the difficulty of determining the mapping between SMCalFlow programs and their glosses which we discuss in Section 7. Taken together, these results suggest that accepting and rejecting glosses is more promising than choosing them.

C Confidence-based Ensembling

The fact that MISO has high performance on the EASY subset suggests that two models could be used in a complementary fashion to balance each others' errors, following Lin et al. (2022). Specifically, we ensemble MISO and BART-large, using the sequence-level confidence as a threshold. When the confidence falls below the threshold (i.e. the example is in HARD) we use the prediction from BART-large, while when it is above the threshold we use MISO's prediction. We tune the threshold on the validation set, finding 0.775 to have the best

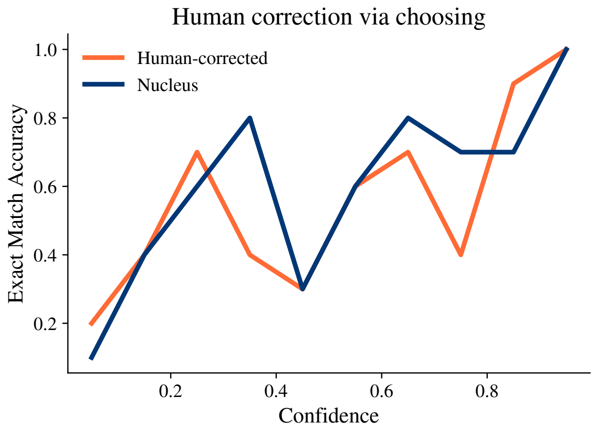

Figure 7: Selection experiment. Annotators sometimes select accurate glosses, but often have trouble deciding between seemingly invariant glosses, lowering performance especially at medium confidences.

accuracy. Table 7 reports the exact match results compared to previous state-of-the-art models on SMCalFlow. For BART-large, we report both the numbers from Roy et al. (2022) and our re-run of the same system.

Table 7 shows the exact match accuracies of the different models; the ensemble model improves over both component models separately, reaching the same performance as the previously-reported state-of-the-art results from Roy et al. (2022). The ensemble does offer some practical benefits over the purely BART-based model. MISO is smaller model than BART-large (127M parameters vs. 406M); with a cutoff of 0.775, MISO is used for 28% of predictions, opening the possibility for substantial computational savings. Surprisingly, the ensemble model is worse in terms of calibration than both its constituent models, with a higher ECE than BART or MISO alone.

Model	EM	ECE
BART (Roy et al., 2022)	83.0	–
BART (re-run)	82.6	5.23
MISO	78.9	3.66
Ensemble	83.0	6.57

Table 7: Exact match accuracy for previous SOTA models (MISO, BART-large) on SMCalFlow, using the test split from Roy et al. (2022). Confidence-based ensembling improves over each model individually in terms of exact match, and achieves parity with SOTA.