Calibrated Interpretation: Confidence Estimation in Semantic Parsing

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

Benjamin Van Durme
Johns Hopkins University, USA
vandurme@jhu.edu

Abstract

Sequence generation models are increasingly being used to translate natural language into programs, i.e., to perform executable semantic parsing. The fact that semantic parsing aims to predict programs that can lead to executed actions in the real world motivates developing safe systems. This in turn makes measuring calibration—a central component to safety—particularly important. We investigate the calibration of popular generation models across four popular semantic parsing datasets, finding that it varies across models and datasets. We then analyze factors associated with calibration error and release new confidence-based challenge splits of two parsing datasets. To facilitate the inclusion of calibration in semantic parsing evaluations, we release a library for computing calibration metrics.[1]

1 Introduction

When probabilistic models are used for decision-making or interaction, we not only want the model to be accurate but also to be *calibrated*, meaning that the probability the model assigns to a decision should roughly correspond to its likelihood of being correct. For example, if a model translates a user instruction into a program for a robot, the confidence that the program accurately captures the person's intent may inform whether the robot executes the program. Because of its importance to safety and usability, a large body of work has addressed calibration, with most research considering single-timestep classification models (i.e., one output decision per input). However, probabilistic models are also commonly applied to multi-timestep tasks. In these tasks, the model generates a *sequence* of decisions, with each subsequent decision dependent on the input and the previous decisions. For example, text generation models typically predict sequences of words one at a time, with each new word conditioned on the previous outputs. Although calibration has been measured in some text generation tasks like machine translation (Kumar and Sarawagi, 2019; Wang et al., 2020), determining the accuracy of generated text sequences can be challenging, making calibration—the relationship between accuracy and confidence—hard to estimate. As sequence generation models, especially large language models, play a role in a growing variety of tasks, measuring their calibration has become extremely important.

The task of executable semantic parsing, where a model predicts an executable program from a natural language instruction, is often modeled as a sequence generation task. Such parsing models are used in systems that interact with the real world, such as human–robot interfaces (Tellex et al., 2020) and digital assistants (Gupta et al., 2018; Cheng et al., 2020; Semantic Machines et al., 2020). Since actions in these domains—especially physical domains—can have irreversible effects, the importance of ensuring model safety cannot be understated. Thus, semantic parsing provides a clear motivation for having a well-calibrated sequence generation model: At low confidence, we may prefer for the system to defer action or request clarification, while when confidence is high, these actions may unnecessarily annoy a user and make the system unusable. This reasoning presupposes that the model's confidence is well-correlated with its probability of success.

Simultaneously, the constrained and executable nature of semantic parses makes accuracy easier to measure. In many text-based sequence generation tasks like machine translation, summarization, long-form question-answering (QA), and

[1]Models/Analysis/Data: https://github.com/esteng/calibration_miso, Metric: https://github.com/esteng/calibration_metric.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 1213–1231, 2023. https://doi.org/10.1162/tacl_a_00598
Action Editor: Katrin Elisabeth Erk. Submission batch: 1/2023; Revision batch: 5/2023; Published 9/2023.

open-ended dialogue, evaluating the quality and correctness of a generation poses a variety of challenges, often due to the fact that there are many ways of stating roughly the same proposition in language. Because calibration measures the relationship between accuracy and confidence, our ability to measure calibration will only be as good as our ability to measure accuracy. In executable semantic parsing, the model generates a program (rather than text) with a more restricted vocabulary and known syntactic rules. This generally limits the number of reasonable semantically equivalent outputs and makes quantifying accuracy easier. Furthermore, the executable nature of the programs allows us to measure accuracy via denotation (i.e., the result of program execution) rather than form. These factors make it an ideal domain for benchmarking the calibration of sequence generation models.

In Section 4, we conduct what is to our knowledge the first large-scale investigation of calibration in sequence generation models as applied to semantic parsing tasks. We examine a variety of commonly used models and measure their calibration across four popular semantic parsing datasets drawn from two different domains: task-oriented dialogue (TOD) and text-to-SQL. We first document the model's calibration profiles, asking how well-calibrated modern semantic parsing systems are; this includes a large pre-trained model queried in a few-shot setting, as well as more traditional fine-tuned models. Using qualitative and quantitative metrics, we find that most TOD models are already fairly well-calibrated. However, the same models are poorly calibrated on text-to-SQL datasets. In Section 5, we analyze various factors implicated in calibration, attempting to shed light on the differences documented in Section 4. We first find that dataset size does not account for the difference in calibration between TOD and SQL models. Using measures of input and program difficulty, we then explore the relationship between difficulty and calibration, finding that for SQL programs, the difficulty of the input and output are associated with poor calibration. Finally, we find that on TOD datasets, each model's low-confidence examples are challenging for other models; this leads us to propose two confidence-based challenge datasets. Because of calibration's importance to semantic parsing specifically—and sequence generation tasks generally—we introduce an open-source library for computing calibration metrics and plotting confidence, compatible with the Transformers library (Wolf et al., 2020).

2 Related Work

2.1 Calibration in NLP

Given the utility of calibrated models in decision-making, 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)—including neural encoders pre-trained on text data (Desai and Durrett, 2020)—and other research indicating they are not (Guo et al., 2017; Wang et al., 2020; Si et al., 2022). These mixed results preclude drawing general conclusions about neural models' calibration, and motivate studies like ours documenting the calibration characteristics of standard models.

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: For example, Jagannatha and Yu (2020) address calibration in structured prediction tasks. More related to our sequence generation setting, Kumar and Sarawagi (2019) and Wang et al. (2020) examine calibration in machine translation, both finding models to be over-confident. Measuring calibration in translation tasks is limited by the metrics used, which are noisy proxies for accuracy and have well-documented limitations (Callison-Burch et al., 2006; Mathur et al., 2020). In semantic parsing specifically, past work has focused on improving confidence estimation for certain parsers. Dong et al. (2018) propose a confidence estimation method based on model and input features. Similarly, Chen et al. (2022) introduce an additional confidence-estimation model for semantic parses. In our work, we extract confidence scores from the same model used for parsing, and focus on analyzing calibration rather than improving it.

Calibration in LLMs We focus on calibration in pre-trained large language models (LLMs), which has also been addressed in other lines of work, first by Mielke et al. (2022), who examine

Dataset	Train	Dev	Test	Example Input	Example Output
SMCalFlow	108,753	12,271	13,496	*Do I have anything going on tonight?*	`(Yield (> (size (QueryEventResponse. results(...))) 0L))`
TreeDST	121,652	22,910	22,841	*I want to book a flight to Paris*	`(plan (^ (Flight)... (Flight.dest...)))`
Spider	7,794	865	1,034	*What are the numbers of all flights coming from Los Angeles?*	`SELECT flno FROM flight WHERE origin = ''Los Angeles''`
CoSQL	6,598	745	1,007	*How many people are named Janessa? | Do you mean the number of people whose first name is Janessa? | Yes*	`SELECT ...AS T1 JOIN ... T2.first_name = ''Janessa''`

Table 1: Number of train, validation, and test examples per dataset and example inputs and outputs. For SQL tasks, column and table names are also included in the input (these are omitted here for readability).

calibration in large pre-trained dialogue models. They focus on ''linguistic calibration,'' describing a guided generation method for introducing verbalized statements of uncertainty (e.g., ''I believe'', ''I'm not sure, but...'', ''I am certain...'') into QA responses. Our experiments use logit-based numerical confidence estimates instead. Similarly, Lin et al. (2022) analyze both logit-based and verbalized uncertainty in GPT-3. Si et al. (2023) examine LLM calibration through the lens of reliability, focusing on QA data and using logit-based confidence estimation. Kadavath et al. (2022) examine whether LLMs are well-calibrated by measuring the probability an answer is true. Finally, Zhou et al. (2023) examine how teaching models to interpret and express certainty and uncertainty impacts calibration and performance. With the exception of Kadavath et al. (2022), these studies focus on single-prediction classification settings: Mielke et al. (2022) examine TriviaQA questions (Joshi et al., 2017), Lin et al. (2022) use math questions, and both Zhou et al. (2023) and Si et al. (2023) consider multiple QA benchmarks including TriviaQA. While studying these settings is valuable, our focus is on longer-form sequence generation, perhaps a more typical use-case for LLMs.

This divergence is also pointed out by Kadavath et al. (2022), who, in addition to several experiments on QA benchmarks like MMLU (Hendrycks et al., 2021), include experiments on HumanEval code generation examples (Chen et al., 2021) and their own dataset of Python code generation problems. Our experiments differ from theirs along a few axes. Firstly, although making claims about the calibration of ''language models'' broadly, Kadavath et al. only consider a single pre-trained model (of varying sizes) and a single program synthesis task; we consider several models across four datasets. Furthermore, while we consider both fine-tuned and few-shot models, Kadavath et al. measure calibration only in a few-shot setting. While we obtain confidence estimates from the token probabilities, Kadavath et al. extract their estimates via an additional prompt that asks the model to label a predicted program or answer as ''True'' or ''False'', where the confidence is taken to be $P(True)$. This is a natural formulation for few-shot models, but is less compatible with fine-tuned models, which are far more common in practice. Note that Kadavath et al.'s method incurs roughly twice the cost of program generation, as the generated program must be re-encoded to obtain a confidence estimate.

3 Methods

Our first goal is to determine the typical calibration characteristics of models applied to semantic parsing tasks. For this, we choose a range of standard semantic parsing tasks, datasets and models.

3.1 Tasks and Datasets

In total, we examine four datasets across two tasks; statistics and examples of each are provided in Table 1. All of the datasets we examine are

in English. Two are TOD parsing datasets: SMCalFlow (Semantic Machines et al., 2020) and TreeDST (Cheng et al., 2020).

In the TOD task, a user engages in a dialogue with a digital assistant in order to achieve some goal, e.g., booking a flight or scheduling a meeting. For each user turn, the agent predicts an executable program and provides a response to the user based on the outcome of the program's execution; thus, the modeling task is to predict a program from a user input and a dialogue history. As neither SMCalFlow nor TreeDST has an available execution suite, we measure correctness by exact match to the reference program. For both TOD datasets, we use the preprocessed data from Platanios et al. (2021), who converted TreeDST into a format shared with SMCalFlow that resembles Lisp, and use the SMCalFlow data splits given by Roy et al. (2022).

Our second task is database querying. As in TOD tasks, the modeling target is a program; however, the program here is a SQL query that can be executed against a database (DB) to return an answer. Crucially, this text-to-SQL task differs from the TOD task in that the programs are highly dependent on the structure of the execution environment. The DB's schema influences the program and changes depending on the DB being used. For the text-to-SQL task, we consider two popular datasets: Spider (Yu et al., 2018) and CoSQL (Yu et al., 2019). Both datasets contain queries over a shared set of 200 DBs; however, while Spider queries are single-turn, CoSQL queries are multi-turn dialogues between a user and an agent. For SQL, we do have access to an execution engine, allowing us to measure both execution accuracy and exact-match accuracy. Here also, we use the splits and preprocessing scripts from Roy et al. (2022).

3.2 Models

Given our goal of benchmarking the calibration characteristics of different approaches commonly in use in semantic parsing, we measure calibration via two extant modeling paradigms: fine-tuning and in-context learning (ICL).

Fine-tuning In the fine-tuning paradigm, we take a model pre-trained with a self-supervised objective on text or code data and continue to train it (i.e., fine-tune it) using a supervised objective on the training data for each dataset in Table 1.[2] We train seven models from two commonly used semantic parsing paradigms: transductive and sequence-to-sequence (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 Lisp-like sequences, they also have an underlying execution graph. Rather than learning to generate the surface form, the transductive approach seeks to directly model the underlying graph, predicting a sequence of nodes as well as labeled, directed edges. Zhang et al. (2019a) and Zhang et al. (2019b) introduced the MISO transductive parsing framework for predicting directed acyclic graphs from text inputs. MISO combines an encoder-decoder model for predicting nodes paired with a biaffine parser (Dozat and Manning, 2017) for edge prediction. It features 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). This kind of model represents an ''engineering-heavy'' approach where inductive bias for parsing is encoded in the model architecture. We take MISO as an example of this class, motivated by its application across a variety of semantic parsing tasks (Zhang et al., 2019a,b; Stengel-Eskin et al., 2020, 2021, 2022; Li et al., 2021). We use Stengel-Eskin et al.'s (2022) best model, which has a RoBERTa (Liu et al., 2019) encoder and contains 127 million (M) parameters.

The seq2seq paradigm 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 with stronger inductive biases, like MISO. We use the BenchCLAMP framework (Roy et al., 2022) to finetune seq2seq models for all TOD and SQL datasets; specifically, we examine the T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) architectures, both of which have frequently been applied to semantic parsing benchmarks (Shaw et al., 2021; Scholak et al., 2021; Desai and Aly, 2021; Banerjee et al.,

[2]Excepting MISO, for which only part of text encoder is pre-trained and the rest of the model is trained from scratch.

2022). Both are large encoder-decoder Transformers (Vaswani et al., 2017) pre-trained on text data with self-supervised objectives. Since SQL is used in many open-source applications (unlike SMCalFlow and TreeDST), code scraped from the web is likely to contain examples of it; thus, for Spider and CoSQL, we also examine Code-T5 (Wang et al., 2021), a T5 architecture pre-trained on large amounts of web-scraped code. As MISO is not commonly used in text-to-SQL tasks, we choose to omit it in our SQL experiments. We examine T5-small (60M parameters), T5-base (220M), T5-large (770M), BART-base (139M), and BART-large (406M), as well as Code-T5-base (220M). Note that while BenchCLAMP allows for constrained decoding according to a context-free grammar, restricting the model to producing only valid parses, we choose to decode in an unconstrained fashion. Because the constrained decoding process intervenes on the output logit space, zeroing out invalid continuations and renormalizing, it could affect the model's calibration characteristics.

In-Context Learning The ICL paradigm instead uses a model that has *only* been trained with a self-supervised objective. Brown et al. (2020) find that a sufficiently large model pre-trained on text data can perform many tasks after being shown a few examples, without any updates to the gradients. This has been extensively explored in the semantic parsing domain, where grammar-constrained decoding has been combined with ICL, allowing models to predict dataset-specific programs from only a few retrieved examples (Shin et al., 2021; Shin and Van Durme, 2022; Roy et al., 2022). Following this paradigm, for a given test query, we retrieve a set of relevant input-output examples from the training data and concatenate them into the model's input, or prompt, as instructive examples. The model then predicts a parse, constrained by the grammar of the domain language. Past work in ICL for semantic parsing has used OpenAI's Codex model (Chen et al., 2021), a LLM based on the GPT architecture and trained on web-scraped text and code. However, in addition to being costly, Codex was recently slated for removal from the OpenAI's public API. To ensure our results are not affected by similar decision in the future, we instead opt to use open-source publicly released models for our ICL experiments. We use Codegen (Nijkamp et al., 2022), which is available through the Transformers library (Wolf et al., 2020). Codegen models are also auto-regressive GPT-style models pre-trained on code, with strong performance across code generation tasks; we explore 4 model sizes: 350M parameters, 2B, 6B, and 16B.

For each test example, we construct a prompt by retrieving 5 similar training examples and concatenating them into the context.[3] We follow Roy et al. (2022) and use a BM25 retriever (Robertson and Zaragoza, 2009) over the full training set, indexed by the input utterance. Examples are ordered by similarity, with the most similar example appearing last (i.e., immediately before the test input).

Input Representation In TOD datasets like SMCalFlow and TreeDST, the user communicates with the agent over the course of a dialogue; we follow previous work in using the previous dialogue turn 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 utterance. Similarly, for CoSQL we include the previous dialogue turn in the input (if available). Correctly predicting a SQL program relies on knowledge of the DB schema; to inform the model of the schema, we follow past work in concatenating schema information (column and table names) into the input.

Confidence Estimation The models used (except for MISO) predict subword tokens, rather than the whitespace-delimited tokens of a programming language. For example, the SQL token `SELECT` is split into 3 subwords by the BART tokenizer; in other words, the model will produce 3 probabilities for this single program token. We estimate program token confidence by first obtaining estimates for subwords and then aggregating. When measuring token-level accuracy, we also use program tokens rather than subwords.

To estimate subword-level confidence, we use the baseline estimator introduced by Hendrycks

[3]The number of examples was chosen in light of hardware memory limitations when running the largest models. Similarly, all billion-parameter models were run at half precision, and memory constraints precluded running constrained decoding on the 16B model, so we do not provide sequence-level accuracies for this model.

and Gimpel (2016), which is robust across many tasks (Varshney et al., 2022). More specifically, we take the maximum probability across the output vocabulary at each timestep. We then aggregate the probabilities across subwords of a given token using `min`. Intuitively, a prediction is only as good as its weakest link. Given a sufficiently long sequence of subwords, aggregation methods like `mean` may obscure low-confidence decisions. In practice, using `mean` pooling on subwords gives qualitatively similar results, as the number of subwords per whitespace token tends to be small. However, in Section 4.4 we see that `mean` leads to high ECE on sequence-level calibration, indicating that `min` may be the better overall choice.

3.3 Metrics

For TOD and SQL datasets, we evaluate our models' semantic parsing ability using exact match accuracy (EM), where a prediction is considered correct it it exactly matches the reference program. This can be quite strict and lead to false negatives; for example, the snippet `x > 0 AND x < 5` is logically identical to the snippet `x < 5 AND x > 0` but would result in an EM score of 0. In the case of SQL, we could mitigate this by executing programs against the provided DBs and comparing the result of the execution to the reference result. This metric is perhaps too lenient, and can result in false positives, e.g., if the gold program yields a null result, then any program yielding a null result will be counted as correct, even if it does not correspond at all to the user's input. In light of this, Zhong et al. (2020) introduce test-suite accuracy, which executes each program against a suite of test DBs optimized for high code coverage. A program is counted as correct if it matches the gold program's denotations on all DBs in the suite. This reduces the false positive rate and provides a much tighter upper-bound on performance.

We follow past work in using expected calibration error (ECE) (Naeini et al., 2015; Guo et al., 2017) as our calibration metric. To compute ECE, predictions are binned by the model's confidence; since each prediction has an accuracy of either 0 or 1, accuracies must be averaged across examples falling in a confidence range to obtain an expected accuracy for a given confidence score. In its original formulation, the number of bins used is a hyperparameter. However, Ding et al. (2020) find that a fixed binning approach used in ECE can be suboptimal. Intuitively, because confidence scores are often distributed non-uniformly, a fixed binning strategy may result in bins containing few samples, leading to high-variance accuracy estimates within these bins. In general, there is a tradeoff between introducing many small bins (estimating accuracy with high variance) and maintaining a few large bins (estimating accuracy with high bias). They instead introduce *adaptive binning*, which correlates the number of samples in a bin with the bin's range: In regions where confidence estimates are sparse, adaptive binning introduces more bins, such that a smaller range is covered by each bin (reducing bias), while when estimates are dense, adaptive binning includes fewer bins, reducing the variance of the estimate. The number of samples for each bin is given by $n = 0.25\left(\frac{Z_{\alpha/2}}{\epsilon}\right)^2$, where $Z_{\alpha/2}$ is the standard normal distribution's Z-score, $1-\alpha$ is the confidence interval, and ϵ is a small positive value included for numerical stability.[4]

The expected calibration error is the difference between the average accuracy of each bin and its confidence, weighted by the size of the bin. Let $\hat{\mathcal{Z}}$ be the model's distribution over the output vocabulary $\mathcal{V}$, and let $\hat{C} = \max \hat{\mathcal{Z}}$, $\hat{Y} = \text{argmax}\, \hat{\mathcal{Z}}$. Let Y be the true class indices and 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:

$$ECE(\mathcal{B}) = 100 * \sum_{i=1}^{N} \frac{|\mathcal{B}_i|}{N} \left| \frac{\sum_{j \in \mathcal{B}_i} a_j}{|\mathcal{B}_i|} - \frac{\sum_{j \in \mathcal{B}_i} c_j}{|\mathcal{B}_i|} \right| \quad (1)$$

In other words, ECE is the mean absolute error between each bin's average confidence and average accuracy; we scale ECE by a factor of 100 for readability. 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 Benchmarking Calibration

Calibration can be measured at the token-level or the sequence-level. Sequence-level calibration is most relevant to safety; in a system using

[4] Like Ding et al. (2020), we found that the metric is not sensitive to changing hyperparameters.

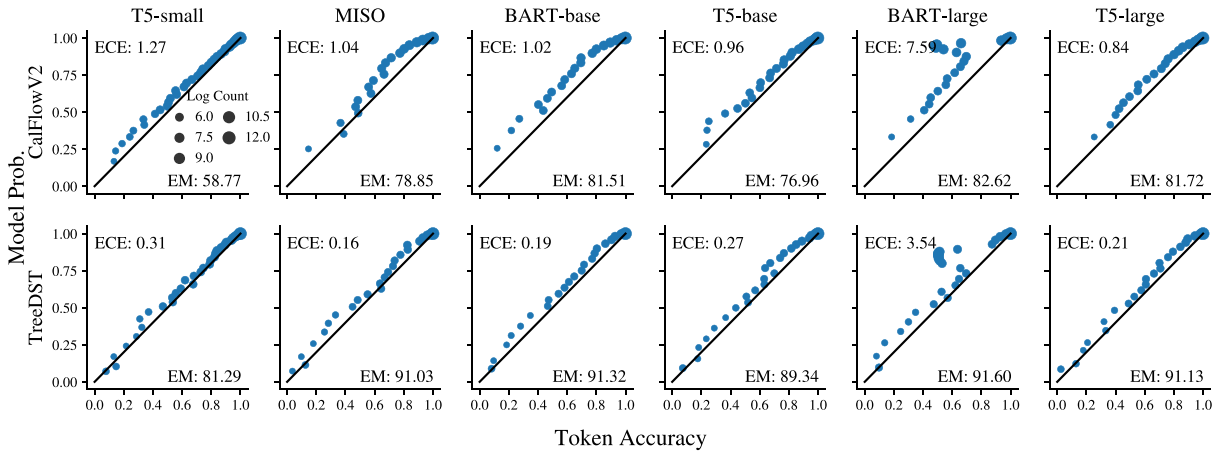

Figure 1: Token-level model confidence and mean accuracy, binned by confidence across models (sorted by size) for TOD datasets. Point size reflects the number of tokens in the bin. Points above the line reflect overconfidence, while those below reflect underconfidence. Exact Match accuracy (EM, higher is better) and Expected Calibration Error (ECE, lower is better) are given. All models show relatively low ECE.

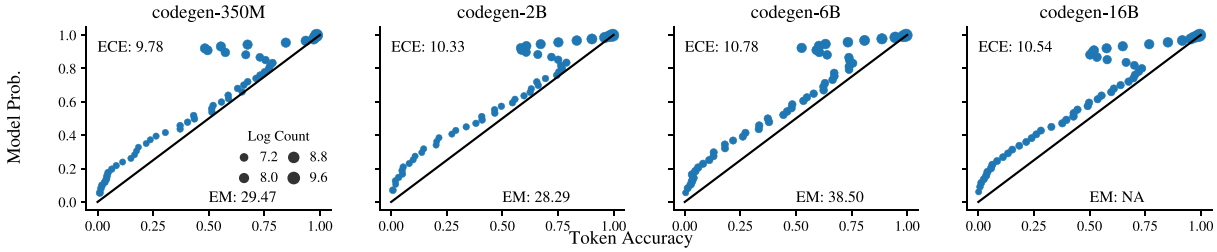

Figure 2: Token-level calibration curves for few-shot code models tested on SMCalFlow.

an executable parsing model, the predicted program will be executed as a whole (i.e., as a sequence). However, absent a (potentially expensive) external model, sequence-level confidence scores will be composed of token-level scores, just as sequences are composed of tokens. Moreover, token-level scores can reveal phenomena obscured by sequence-level scores. Rather than only measuring the model's accuracy and confidence on a whole sequence, it may be more informative to examine on which types of tokens the model makes mistakes or has high calibration error. For example, token-level confidence scores can allow us to examine which specific functions the model is struggling on.

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 teacher forcing, feeding the model the gold prefix up to the current timestep. That is, when predicting the confidence of token $\hat{y}_t$, we use tokens $y_1, \ldots, y_{t-1}$ taken from the *gold* program P, not the predicted program $\hat{P}$. Note that this differs from the setting used for computing EM accuracy in Figure 1, Figure 2, and Figure 4, where we compare the gold program $P = y_1, \ldots, y_T$ to the predicted program $\hat{P} = \hat{y}_1, \ldots, \hat{y}_T$. For sequence-level confidence, we use $\hat{P}$.

4.1 Task-oriented Dialogue Results

Figure 1 shows the token-level calibration plots and ECE scores for all fine-tuned models on TOD datasets. Note that in our plots, the model's confidence 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 $y = x$: Points above the line are overconfident bins (confidence $>$ accuracy); those under the line are underconfident. The models 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]); the largest bin for all models is the most confident bin. This is consistent with the Exact Match (EM) accuracy results reported

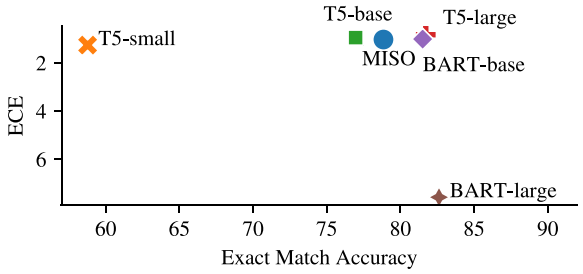

Figure 3: Empirical trade-off between accuracy and calibration. Note that the y-axis is flipped (lower calibration error is better). Several models exist at the Pareto front.

in Figure 1; for a model to achieve high accuracy, all its output tokens must exactly match the reference tokens on most programs, i.e., the vast majority of tokens must be predicted correctly.

We note that all models are relatively well-calibrated. For the T5 series, calibration improves with scale, while the opposite is true for BART. In Figure 3, we plot the trade-off between accuracy and calibration error for SMCalFlow models on the test set. Here, we observe a Pareto front, with BART models having higher accuracy but worse ECE than comparable T5 models. Importantly, BART's calibration curves are non-monotonic, which is particularly troubling. Given monotonic distortions (e.g., a sigmoid calibration curve), regression models can be fit to the validation set to correct the calibration curve (Zadrozny and Elkan, 2002). Such correction is much harder to do for non-monotonic distortions.

4.2 Few-shot Calibration

Figure 2 shows the calibration curves for ICL models with 5 prompt examples on the SMCalFlow test set. First, we note that the exact-match results here are obtained with constrained decoding, since without fine-tuning models are generally unable to produce syntactically-correct programs. However, the token-level confidence scores are obtained without constraints (as in Section 4.1), so the curves are comparable to those in Figure 1. Even with constraints, the exact match performance of these models is lower than that of the fine-tuned models; however, they are shown only five examples per input, while the fine-tuned models are trained on over $100,000$ examples. The ECE of the ICL models is higher than the ECE of finetuned models. While ECE increases initially with model scale, the ECE of the 16B model is slightly lower than the ECE of the 6B model, suggesting that the trend could be U-shaped, a phenomenon that has been documented in other large models (Wei et al., 2022). Qualitatively and quantitatively, the ICL calibration curves are remarkably similar to the BART-large curve in Figure 1, despite the fact that BART-large is fine-tuned on the training data. We qualitatively analyze the spikes seen in many BPE-based models like BART and Codegen in Section 5.1, finding that they are driven by common syntactic tokens.

4.3 SQL Results

Figure 4 shows the token-level calibration characteristics of models fine-tuned on SQL data. Adaptive binning results in fewer bins here due to the smaller sizes of the SQL test sets. We see both qualitatively and quantitatively that many of the same models which were well-calibrated on TOD data are poorly calibrated on SQL datasets. All models are substantially over-confident, and the trends between models here are different than in the TOD setting. BART-large is better-calibrated than BART-base, which is the worst-calibrated on both datasets. We also see each model has higher ECE on CoSQL than Spider; this may have to do with input complexity, since CoSQL is multi-turn. We also note that the EM scores for SQL are generally quite low; this tracks with past results finding that EM is an excessively strict metric for SQL (Zhong et al., 2020). We found the execution accuracies for the models to be in-line with those reported by Roy et al. (2022).

4.4 Sequence-level Calibration

The results in Figure 1 follow past calibration work in using confidence estimates at the level of individual classifications (i.e., tokens). However, unlike many of the NLP tasks where calibration has been explored in the past, the output in semantic parsing is sequential, i.e., a series of classifications, where each timestep is dependent on the preceding decisions. Thus, to obtain sequence-level confidence scores from token-level scores, we need a method for aggregating token-level estimates. We explore two aggregation functions: `min` and `mean`. These operate over the token-level confidence scores generated during decoding; note here the decoding takes place *without* teacher-forcing, i.e., using the predicted program $\hat{P}$. Table 2 shows the ECE for

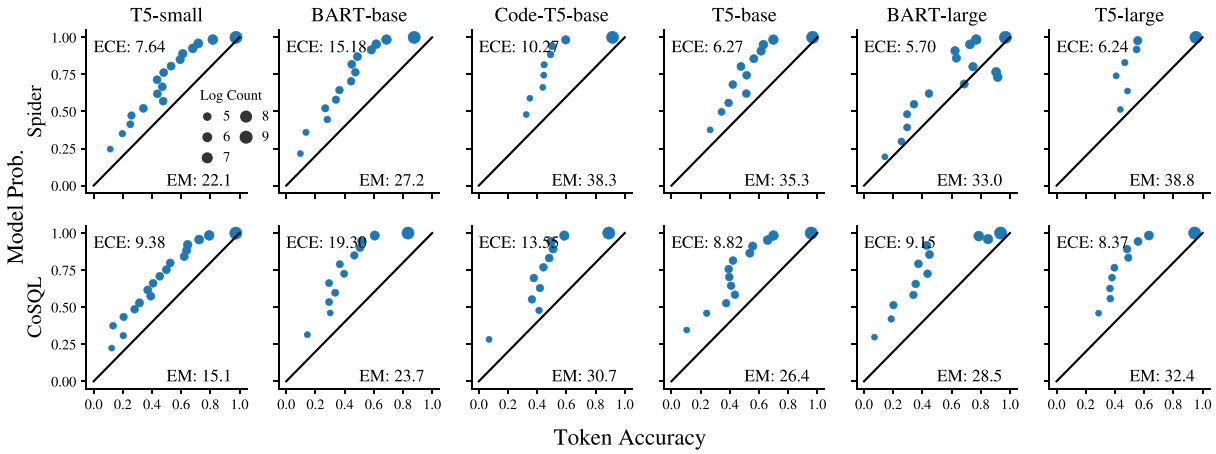

Figure 4: Token-level model confidence and accuracy for SQL datasets, binned by confidence across models and datasets. Unlike the models fine-tuned on TOD data (cf. Figure 1) models fine-tuned on SQL are poorly calibrated.

Model	ECE (Min.)	ECE (Mean)
MISO	5.57	5.49
BART-large	6.23	16.85
T5-large	8.29	18.01

Table 2: Sequence-level ECE for representative models on SMCalFlow.

sequence-level confidence scores with the two aggregation methods on the SMCalFlow dataset. Since `min` is typically better, especially on BART-large and T5-large, where `mean` results in high over-confidence, we adopt `min` moving forward. As mentioned in Section 3.2, `mean` may result in high ECE because over a long sequence, a single low-confidence prediction may be ‘‘washed out’’ by a sequence of high-confidence predictions. This is especially true in program prediction tasks, where syntactic constraints mean that many tokens can be easily predicted with high confidence (most SQL programs begin with `SELECT`, etc.).

5 Discussion and Analysis

In Section 4, we found that for a fixed dataset, different models have varying calibration characteristics, even when the models have similar architectures and pre-training data. Similarly, we saw that the same models often differ between datasets, and especially differ between domains (TOD vs. text-to-SQL). Finally, we found that some models show conspicuous spikes in calibration error in some regions of the confidence space, and that these spikes hold across models. In light of these differences and quirks, we conduct additional analyses on factors associated with model calibration.

5.1 Qualitative Analysis

Error Spikes In Figure 1 (for BART-large) and in Figure 2 (for all models), we see a spike in calibration error in the higher-confidence region. This spike generally becomes more pronounced as the size of the model increases. Given that these models all share a tokenizer and that the spikes are located in similar regions of the confidence space, we hypothesize that they might be related to the identity of the tokens in the bins. We isolate predictions in bins within each spike (i.e., bins with high confidence and lower accuracy, where the curve deviates from the $y = x$ line.) We find that many of the tokens are shared between models and datasets; while each bin for each dataset does contain ‘‘content’’ tokens (e.g., function names, value names), all bins for all models contain a large number of very frequent ‘‘syntactic’’ tokens (e.g., `(`, `)`, `,`, `=`, `,`, `>`, etc.) shared by SMCalFlow and TreeDST. Given how common these tokens—especially parentheses—are in the Lisp datasets, it is likely that the error spikes are driven by overconfidence on these syntactic tokens.

Common Errors We find that most models are over-confident; we can see over-confidence either as a failure of confidence estimation (i.e., the model predicts a higher confidence value than is appropriate) or a failure of prediction (i.e., the

model makes mistakes that lead to low accuracy). Following the second interpretation, we examine some common errors that models make on text-to-SQL and TOD datasets. For SQL datasets, we find that higher-confidence programs (> 0.5) sometimes are semantically correct, but fail according to exact match because of a syntactic difference. For example, some programs use single quotes rather than double quotes; while this does not affect the execution result, it counts as a failure for exact-match accuracy. We discuss the shortcomings of exact-match further in Section 5.4. In SMCalFlow, many high-confidence errors are due to mismatches in capitalization, and others are due to confusion between similar functions, e.g., substituting `AttendeeListHasRecipientConstraint` for `AttendeeListHasRecipient`.

We also see some errors due to ambiguous examples, where the model predicts a valid interpretation of the input which deviates from the reference. For example, given the input *''List the number of different series names and contents in the TV Channel table.''*, the BART-large text-to-SQL model trained on Spider predicts the program:

```
SELECT count ( DISTINCT
series_name ) , content FROM
tv_channel
```

which corresponds to the following valid parse of the input: *''[the number of different series names] and [contents]...''*. The reference however is:

```
SELECT count ( DISTINCT
series_name ) , count ( DISTINCT
content ) FROM tv_channel;
```

which corresponds to *''[the number of different [series names and contents]]...".*

5.2 Data Size

Given the dramatic differences in calibration between Figure 1 and Figure 4, a natural question to ask is *why* SQL models are so much less calibrated. One simple hypothesis is dataset size: Table 1 shows that the SQL datasets have between 14x and 18x *less* data available for fine-tuning than the TOD datasets. Could simply increasing the size of the dataset fix the poor SQL calibration seen in Figure 4?

If calibration were a result primarily of the dataset size, we would expect an SMCalFlow model trained on a small amount of data to also have high ECE. To test this, we trained a T5-small model on the first $7,794$ examples in the SMCalFlow dataset—the same exact number of training examples as are available for Spider. We find that the model's ECE increases from 1.27 to 2.40. While the ECE does increase, it is still far lower than the ECE on Spider (7.64), indicating that low dataset size is not the primary driver of poor calibration on text-to-SQL tasks.

5.3 Input and Output Difficulty

Having established that the difference in calibration between TOD and text-to-SQL models cannot be accounted for merely by dataset size, we explore other factors that could be associated with poor calibration. Specifically, we examine the difficulty of the input and output. For the input, we examine out-of-distribution (OOD) inputs, following past observations that models are typically over-confident on OOD inputs (Bui and Liu, 2023). We measure how OOD an input is via perplexity from an LSTM LM (Hochreiter and Schmidhuber, 1997). To exclude the possibility of data leakage, we train this model from scratch, without any pre-trained word embeddings. Inputs are tokenized using BPE (Sennrich et al., 2016) and the train split is used to build a vocabulary. We autoregressively train 2-layer unidirectional 256-dimensional LSTMs on the training splits of SMCalFlow and Spider, using an Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001.[5] Training ends when the validation perplexity fails to decrease for 5 consecutive epochs.

We consider sequence-level confidence bins from models shared between SMCalFlow and Spider (this excludes MISO and Code-T5). We compute the average perplexity of the inputs in each bin and plot the confidence and accuracy against the mean perplexity in Figure 5. In all cases, we expect to see accuracy decreasing with perplexity: Intuitively, as inputs become more OOD, they become harder for the model to parse. If models are robust to OOD inputs, the confidence should also decrease with perplexity, i.e., the model should be ''aware'' of the fact that it is worse at predicting programs for OOD inputs.

[5]LSTMs were chosen over Transformers here because of the relatively small size of the data combined with the desire to train the model from scratch.

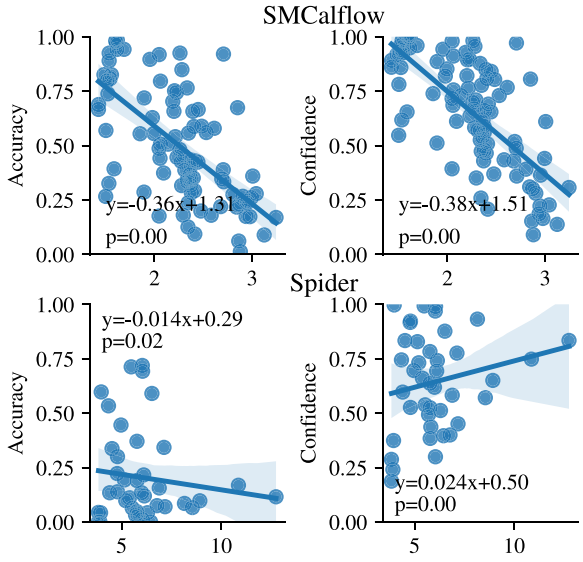

Figure 5: Confidence and accuracy (y-axis) plotted by perplexity of the input (x-axis).

On SMCalFlow, accuracy is correlated with input perplexity, decreasing on more OOD inputs. Confidence decreases with perplexity at a similar rate, indicating that the model has learned to identify OOD inputs and produce lower confidence values accordingly. That the slopes of the two lines are very close aligns with the low ECE seen on SMCalFlow across models: accuracy, confidence, and input perplexity are all ‘‘coupled’’.

For Spider, on the other hand, there is no such coupling. While accuracy is weakly correlated with perplexity, confidence is *positively* correlated with perplexity. In other words, as inputs become more OOD, the model becomes *more* over-confident in its predictions.[6] This suggests that the SQL models may not have learned to recognize OOD inputs.

We also consider output difficulty in the SQL domain, where use the difficulty labels provided by Yu et al. (2018), who classify SQL programs into *easy*, *medium*, *hard*, and *extra-hard* depending on the types of functions they involve. We use T5-large, which is the best-calibrated (and best-performing) model in Figure 4, and compute the sequence-level ECE for programs separated out by difficulty-type. We do note here that, due to the relatively small number of programs in the SQL test set, the bins here are often sparse, introducing variance into the estimates. To account for this, we train and evaluate models using 3 random seeds and plot the average performance. These results are described in Figure 6, where ECE generally increases with difficulty. The extra-hard programs have roughly twice the ECE of easy programs. Notably, accuracy decreases as programs get harder; the high ECE values for harder programs indicates that the model does not lower the confidence appropriately for these programs.

[6]Note that for Spider, the two outlier points with high average perplexity (> 10) contain many tokens not seen in training, like names of countries. Removing these outliers does not have a qualitative impact on the results.

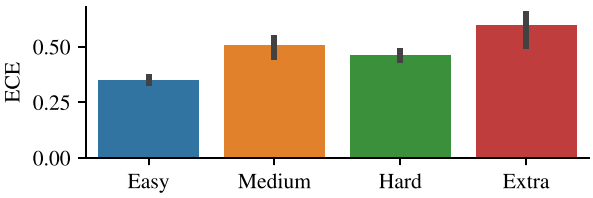

Figure 6: T5-large ECE by target program difficulty on Spider (as defined by Yu et al. [2018]).

5.4 Execution Accuracy

Because SQL is executable, we can additionally examine whether models are well-calibrated with respect to execution accuracy. This is especially relevant in light of the results in Figure 4, indicating that models are over-confident. Execution accuracy is typically more lenient than exact match accuracy, which is prone to *false negatives*; a program may vary syntactically from a reference but still execute to the same result, i.e., have the same denotation.

Another more lenient form of accuracy is accuracy@k (Acc@k, which measures whether the correct program is in the top k programs returned by the model after beam search. Intuitively, if there are two equally valid programs for a given input, we would expect Acc@2 to capture both, while Acc@1 (i.e., standard EM accuracy) would only have a 50% chance of being correct. While Acc@k is more lenient, it is not a realistic metric, since in practice we always need to choose one single program to execute.

Figure 7 shows Acc@k and the execution accuracy for T5-base on the Spider test set. The dotted line connects Acc@1 to execution accuracy, showing the range. First, we note that while all bins are over-confident, the execution accuracy is generally better-calibrated (less over-confident) than any of the Acc@k values. Note

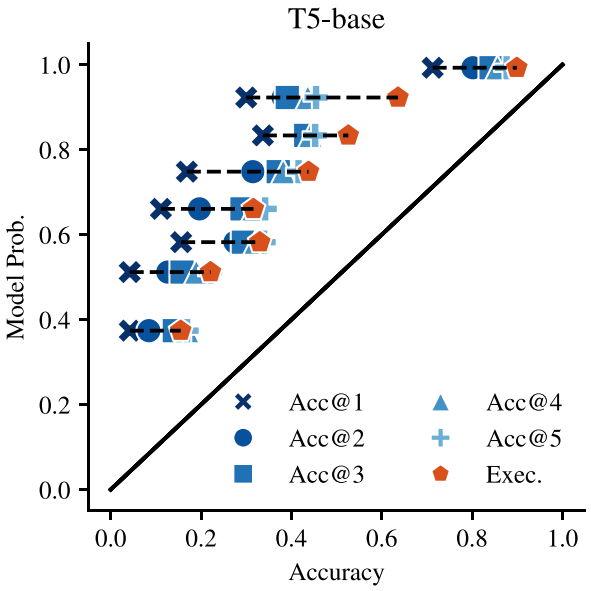

Figure 7: Execution accuracy and exact match accuracy@k. All accuracies result in over-confidence, but execution accuracy is generally the best-calibrated.

that the model's training signal only provides indirect information about execution accuracy, since the loss is computed against a single reference program (which ostensibly would execute correctly), not against a set of equivalent programs. These results are promising: while the T5-base model is very poorly-calibrated according to exact match, it is not nearly as bad when considering test-suite eecution accuracy, which has been demonstrated to be the more informative metric (Zhong et al., 2020).

5.5 Easy and Hard TOD Subsets

Figure 1 shows that for many of the TOD models we examine, token-level confidence is well-correlated with accuracy; in Table 2, using `min` to aggregate token-level confidence scores, we can obtain relatively low ECE at the sequence-level as well, meaning we can predict (on average) how likely a test example is to be correct. The ECE values in Table 2 tell us that the lower a model's `min` confidence is, the more likely it is to make a mistake. Since different models have qualitatively different calibration curves, and may have complementary errors, we use an ensemble of three different models (MISO, T5-large, BART-large) to extract 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.

Model	HARD (SM)	HARD (Tr)
MISO	33.31%	18.58%
BART-large	18.54%	44.50%
T5-large	23.15%	11.93%
Union	40.38%	50.50%

Table 3: Percentage of test examples labeled as HARD by each model for SMCalFlow (SM) and TreeDST (Tr).

The HARD subset contains the union of examples for which the *sequence*-level confidence falls below a threshold – all other examples are in EASY. The threshold for each dataset is chosen by computing the 25^{th} percentile over all of the sequence-level probabilities across the three models. This threshold is 0.86 for SMCalFlow and 0.85 for TreeDST. The union is taken across the three models, i.e., if any one model assigns an example a confidence below the threshold, the example is considered hard.

Table 3 shows the percentage of test data below the threshold for each model. Note that, because the models do not all assign low confidence to the same examples, the union of examples exceeds 25% of the data. Similarly, because the threshold is computed using the aggregated data from across all 3 models, it is possible for a single model to have more than 25% HARD examples, as long as the average across all 3 models is 25%. For SMCalFlow, MISO contributes the most HARD examples, while for TreeDST, BART-large contributes more.

Table 4 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. MISO's performance is lower than that of the other models; the difference is larger than the performance difference in Figure 1. This is partly due to MISO contributing a large percentage of low-confidence examples to HARD (cf. Table 3)—low-confidence examples are often more likely to be misclassified (Hendrycks and Gimpel, 2016). We release our HARD and EASY subsets for both SMCalFlow and TreeDST, to act as challenge datasets for future work.

Dataset	Model	HARD	EASY
SMCalFlow	MISO	53.43	96.05
	BART-L	62.66	96.15
	T5-L	60.28	96.25
TreeDST	MISO	80.32	94.42
	BART-L	84.97	98.67
	T5-L	84.10	98.61

Table 4: Exact match accuracy on the EASY and HARD subsets for all models. L indicates ''large'' variant. All models perform significantly worse on HARD.

5.6 Limitations

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 multiple datasets and models, our datasets are drawn from two domains and programming paradigms, and our models are limited to Transformer-based architectures. While these choices are representative of current standards in executable semantic parsing, we encourage broader investigations spanning additional models and datasets. Specifically, future work may address programs that are interleaved into text-based interactions. As sequence-models become integrated into chat-bot interfaces, increasing attention has been paid to augmenting these models with ''tools'', i.e., the ability to call APIs by predicting short programs (Schick et al., 2023; Mialon et al., 2023). Examining calibration in these settings is particularly relevant. We examine only the highest-resource settings and leave examining how calibration profiles change with dataset size to future work.

We are also limited in how we measure calibration. ECE can be a brittle metric (Ovadia et al., 2019; Si et al., 2022); we have tried to mitigate this by using adaptive binning. However, there are still hyperparameters and design choices involved in measuring ECE, and raw ECE scores can obscure a model's true calibration characteristics. We attempt to balance this using qualitative assessments. Similarly, while accuracy is easier to measure in semantic parsing than other text generation tasks, we have noted some of the shortcomings of exact-match accuracy metrics.

6 Conclusion

Broadly, our results show that calibration is a complex phenomenon with a multitude of influences. On a single dataset, different models vary in their calibration error; moreover, the same model often varies drastically between different datasets, indicating that calibration is a function of both the model and the dataset. Taken together, these results point to the complexity of measuring calibration, and suggest that care should be taken when making claims about classes of models based on evidence from a limited number of models or datasets. Given the utility of calibrated models and the importance of calibration to safety, 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.

Acknowledgments

We would like to thank Zhengping Jiang, Anthony Platanios, Chenglei Si, Subhro Roy, Kate Sanders, Yu Su, Dan Klein, Matt Gardner, Anqi Liu, and Daniel Khashabi for their feedback and pointers. We also thank the TACL reviewers and the Action Editor, who provided valuable feedback. Elias Stengel-Eskin is supported by an NSF GRFP, and this work was additionally supported by NSF #1749025.

References

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. 2022. Modern baselines for sparql semantic parsing. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2260–2265. https://doi.org/10.1145/3477495.3531841

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter,

Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Ha Manh Bui and Anqi Liu. 2023. Density-softmax: Scalable and distance-aware uncertainty estimation under distribution shifts. *arXiv preprint arXiv:2302.06495*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Shijie Chen, Ziru Chen, Huan Sun, and Yu Su. 2022. Error detection for interactive text-to-sql semantic parsing. In *Proceedings of the Second Workshop on Interactive Learning for Natural Language Processing*.

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. https://doi.org/10.18653/v1/2020.emnlp-main.651

Shrey Desai and Ahmed Aly. 2021. Diagnosing transformers in task-oriented semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 57–62. https://doi.org/10.18653/v1/2021.findings-acl.5

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. https://doi.org/10.18653/v1/2020.emnlp-main.21

Yukun Ding, Jinglan Liu, Jinjun Xiong, and Yiyu Shi. 2020. Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 4–5. https://doi.org/10.1109/CVPRW50498.2020.00010

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. https://doi.org/10.18653/v1/P18-1069

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.

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. https://doi.org/10.18653/v1/D18-1300

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735, PubMed: 9377276

Abhyuday Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *Proceedings of the Association for Computational Linguistics Meeting*, volume 2020, page 2078. NIH Public Access. https://doi.org/10.18653/v1/2020.acl-main.188, PubMed: 33612961

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1147

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

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.

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*.

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. https://doi.org/10.18653/v1/2020.acl-main.703

Zhuowan Li, Elias Stengel-Eskin, Yixiao Zhang, Cihang Xie, Quan Tran, Benjamin Van Durme, and Alan Yuille. 2021. Calibrating concepts and operations: Towards symbolic reasoning on real images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/ICCV48922.2021.01464

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*.

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*.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. *arXiv preprint arXiv:2006.06264*. https://doi.org/10.18653/v1/2020.acl-main.448

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: A survey. *arXiv preprint arXiv:2302.07842*.

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. https://doi.org/10.1162/tacl_a_00494

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*. https://doi.org/10.1609/aaai.v29i1.9602

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. https://doi.org/10.1145/1102351.1102430

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

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.

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. https://doi.org/10.18653/v1/2021.acl-long.284

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.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389. https://doi.org/10.1561/1500000019

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*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.779

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. https://doi.org/10.1162/tacl_a_00333

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1162

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? 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 922–938, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.75

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen Jr., Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715. https://doi.org/10.18653/v1/2021.emnlp-main.608

Richard Shin and Benjamin Van Durme. 2022. Few-shot semantic parsing with language models trained on code. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5417–5425. https://doi.org/10.18653/v1/2022.naacl-main.396

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023. Prompting GPT-3 to be reliable. *International Conference on Learning Representations*.

Chenglei Si, Chen Zhao, Sewon Min, and Jordan Boyd-Graber. 2022. Re-examining calibration: The case of question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2814–2829, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Elias Stengel-Eskin, Kenton Murray, Sheng Zhang, Aaron Steven White, and Benjamin Van Durme. 2021. Joint universal syntactic and semantic parsing. *Transactions of the Association for Computational Linguistics*. https://doi.org/10.1162/tacl_a_00396

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)*. https://doi.org/10.18653/v1/2022.emnlp-main.789

Elias Stengel-Eskin, Aaron Steven White, Sheng Zhang, and Benjamin Van Durme. 2020. Universal decompositional semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8427–8439. https://doi.org/10.18653/v1/2020.acl-main.746

Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. 2020. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55. https://doi.org/10.1146/annurev-control-101119-071628

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022. Investigating selective prediction approaches across several tasks in iid, ood, and

adversarial settings. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1995–2002. https://doi.org/10.18653/v1/2022.findings-acl.158

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. https://doi.org/10.18653/v1/2020.acl-main.278

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C. H. Hoi. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708. https://doi.org/10.18653/v1/2021.emnlp-main.685

Jason Wei, Najoung Kim, Yi Tay, and Quoc V. Le. 2022. Inverse scaling can become u-shaped. *arXiv preprint arXiv:2211.02011*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 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. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. 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 1962–1979. https://doi.org/10.18653/v1/D19-1204

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921. https://doi.org/10.18653/v1/D18-1425

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. https://doi.org/10.1145/775047.775151

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)*. https://doi.org/10.18653/v1/D18-1009

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. https://doi.org/10.18653/v1/P19-1009

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. https://doi.org/10.18653/v1/D19-1392

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411. https://doi.org/10.18653/v1/2020.emnlp-main.29

Kaitlyn Zhou, Dan Jurafsky, and Tatsunori Hashimoto. 2023. Navigating the grey area: Expressions of overconfidence and uncertainty in language models. *arXiv preprint arXiv: 2302.13439*.