Addressing Resource and Privacy Constraints in Semantic Parsing Through Data Augmentation

Anonymous ACL submission

Abstract

We introduce a novel setup for low-resource task-oriented semantic parsing which incorporates several constraints that may arise in real-world scenarios: (1) lack of similar datasets/models from a related domain, (2) inability to sample useful logical forms directly from a grammar, and (3) privacy requirements for unlabeled natural utterances. Our goal is to improve a low-resource semantic parser using utterances collected through user interactions. In this highly challenging but realistic setting, we investigate data augmentation approaches involving generating a set of structured canonical utterances corresponding to logical forms, before simulating corresponding natural language and filtering the resulting pairs. We find that such approaches are effective despite our restrictive setup: in a low-resource setting on the complex SMCalFlow calendaring dataset (Andreas et al., 2020), we observe 33% relative improvement over a non-data-augmented baseline in top-1 match.

1 Introduction

We aim to improve the performance of a semantic parser based on previous user interactions, but without making use of their direct utterances to the system nor any associated personal identifiable information (PII). Such privacy requirements are common in practical deployment (Kannan et al., 2016), and semantic parsers are commonly used in real-world systems such as Siri and Alexa, converting natural language into structured queries to be executed downstream (Kamath and Das, 2018).

Constructing semantic parsers can also be resource-intensive: annotating training data consisting of natural language-logical form pairs often requires trained experts. Two complementary lines of recent work address this issue. First, several works (Zhong et al., 2020; Cao et al., 2020) tackle *low-resource* semantic parsing via approaches such as data augmentation. A second line of work (Wang et al., 2015; Xiao et al., 2016) explores *canonical utterances:* structured language which maps one-to-one to logical forms, but which resembles natural language (Table 1). Representing logical forms as canonical utterances lowers the difficulty of parsing natural utterances not only for humans, but for models (Shin et al., 2021; Wu et al., 2021).

Natural	*When is Allison's birthday?*
Logical	(Yield :output (:start (singleton (:results (FindEventWrapperWithDefaults :constraint (Constraint[Event] :subject (? = #(String "Allison's birthday"))))))))
Canonical	start time of find event called something like "Allison's birthday"

Table 1: An example of natural language, logical form, and canonical form in the SMCalFlow domain. The event title, *"Allison's birthday,"* is PII.

We consider low-resource semantic parsing with further resource and privacy constraints which may arise in practical deployment: beyond a small gold dataset of labeled pairs, we assume only unlabeled natural utterances which must be masked for PII. Unlike many prior works, we assume that (1) we do not have a large dataset of related logical forms in a different domain, (2) we cannot sample arbitrarily many useful logical forms, and (3) we must preserve privacy of user utterances.

We propose several approaches which are compatible with our imposed restrictions, broadly following three steps: (1) generate a set of privacy-preserving canonical utterances; (2) simulate corresponding natural utterances; and (3) filter the resulting canonical-natural utterance pairs to yield additional "silver" data for training. We double the performance of a non-data-augmented baseline on the ATIS domain (Hemphill et al., 1990), and achieve a relative improvement of 33% on the more realistic SMCalFlow domain (Andreas et al., 2020). We hope this work motivates further research interest in methods for parser improvement in realistic scenarios.

2 Semantic Parsing in Practice

Our setup assumes access exclusively to:

1. a small "seed" dataset $\mathcal{D}$ of natural utterance with corresponding parses, and
2. a larger set of unlabeled natural utterances $\mathcal{U}$, for which PII must be masked before use.

In a real-world setting, one might hand-annotate the seed dataset $\mathcal{D}$ to train a system for initial deployment, while then leveraging $\mathcal{U}$ to refine a future version of the system.

While our setting is highly restrictive, we argue that it reflects practical constraints. For example, in practice, the grammar for logical forms—as well as the synchronous context-free grammar (SCFG) that maps them to canonical utterances—will often be written from scratch, precluding transfer learning methods which leverage a large quantity of similar data in another domain. Moreover, in complex domains, one cannot expect to *sample* useful logical forms directly from a grammar if the grammar is designed for *coverage* as in e.g., SMCalFlow (Andreas et al., 2020). Therefore, other than $\mathcal{D}$, the only additional data (excluding additional manual annotation) are subsequent user inputs in the form of $\mathcal{U}$, with PII masked to preserve privacy.

3 Related Work

Compared to prior work in low-resource semantic parsing, our task setup's constraints require different approaches.

First, we consider semantic parsing on an entirely *new* grammar for logical forms, rather than adapting to new domains starting from a *preexisting* grammar (Zhao et al., 2019; Zhong et al., 2020; Burnyshev et al., 2021; Kim et al., 2021; Tseng et al., 2021). For example, Zhong et al. (2020) takes a natural-language-to-SQL model for one database to propose language-SQL training examples for another database.

Second, we assume one cannot sample useful canonical utterances directly from the grammar, unlike Zhong et al. (2020) and Cao et al. (2020). For example, Cao et al. (2020) use a backtranslation-esque approach leveraging large numbers of unlabeled natural and canonical utterances.

Moreover, we do not even assume direct access to unlabeled natural utterances, due to real-world privacy considerations (Kannan et al., 2016; Campagna et al., 2017). Many works on low-resource semantic parsing, such as those mentioned previously, do not consider the privacy aspect.

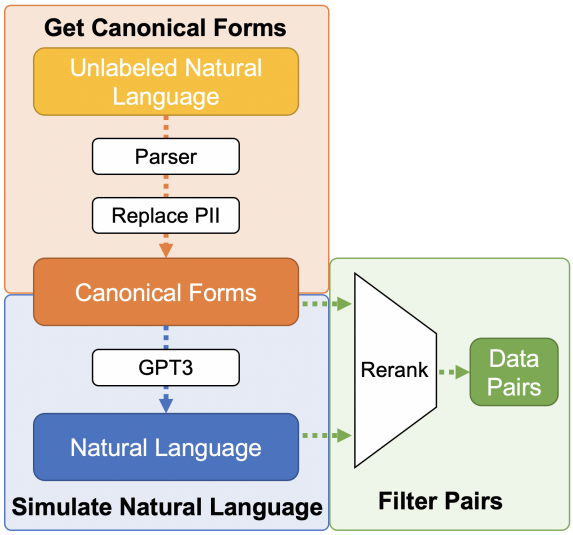

Figure 1: Illustration of one of our proposed methods for data augmentation (USER-RANK) in low-resource semantic parsing. We first obtain canonical forms from unlabeled user data using a parser trained on seed data, replacing PII. Next, we simulate corresponding natural language for the generated canonical forms. Finally, we filter the canonical-natural pairs to obtain our final silver data pairs for augmentation.

Nevertheless, recent work (Shin et al., 2021; Wu et al., 2021; Yin et al., 2021; Schucher et al., 2021) has demonstrated decent performance given just a small seed dataset $\mathcal{D}$, by combining pretrained language models with constrained decoding. For example, Shin et al. (2021) use only 300 labeled examples in the complex SMCalFlow dialogue domain (Andreas et al., 2020). However, using pretrained models to directly *generate* silver training data, with a method such as DINO (Schick and Schütze, 2021), is unsuitable in semantic parsing: the models are unaware of either the underlying grammar or the space of parse-able queries. One of our contributions is to explore more effective uses of pretrained models for data augmentation in a practical semantic parsing scenario.

4 Data Augmentation for Practical Semantic Parsing

While finetuning a pretrained model on the seed dataset $\mathcal{D}$ can yield a reasonable parser P (Shin et al., 2021; Wu et al., 2021), we aim to increase performance via data augmentation. However, our realistic setup precludes many prior approaches. We propose to generate silver data via three main

steps, shown in Figure 1: (1) generate a set $\mathcal{C}$ of canonical utterances c, (2) simulate a set $\mathcal{N}$ of corresponding natural utterances n, and (3) filter the resulting (c, n) pairs. We suggest multiple approaches for these steps, and benchmark their efficacy in Sec. 5. The entire procedure can be iterated multiple times as the parser improves.

4.1 Generating Canonical Utterances

First, we generate canonical utterances c. In principle, one could sample directly from a task-specific grammar, but the results may not be useful in practice (Sec. 5). The remaining options are to generate c conditioned on either unlabeled natural utterances $\mathcal{U}$ or the seed data $\mathcal{D}$.

Generation conditioned on $\mathcal{U}$ (USER). We need to mask all PII, but this is difficult to guarantee in the original natural language domain. Therefore, we first train a parser P on $\mathcal{D}$, and parse each utterance in $\mathcal{U}$ to obtain a set of canonical utterances $\mathcal{C}'$. In the more structured domain of $\mathcal{C}'$ we can guarantee masking and replacing all PII to yield the final set $\mathcal{C}$. Critically, it is not necessary that the initial $\mathcal{C}'$ are correct parses of $\mathcal{U}$; we only need a realistic distribution over canonical utterances, and the initial $\mathcal{U}$ is no longer parallel to the final $\mathcal{C}$ anyway due to replacing PII. Hence it is acceptable if the parser P's errors are numerous but unbiased. In any case, the final $\mathcal{C}$ will be somewhat tied to the true distribution of user utterances in $\mathcal{U}$.

Generation conditioned on $\mathcal{D}$ (GPT). A second method of generating $\mathcal{C}$ is SCFG-constrained decoding on an autoregressive language model,[1] prompting with the seed data $\mathcal{D}$. Specifically, we prompt with a random concatenation of plans from $\mathcal{D}$, separated by newlines. The SCFG that defines canonical utterances constrains the decoding, forcing the model to output a valid canonical utterance.

4.2 Simulated Natural Utterances

For each canonical c in $\mathcal{C}$, we now re-generate a natural utterance n. While other methods (e.g., fine-tuning) are possible, here we employ a prompting approach using GPT3 (Brown et al., 2020). We use a prompt containing $\mathcal{D}$'s canonical-natural pairs, ending with the canonical utterance c for which we want to sample a corresponding n.

[1] Ideally we would use GPT3 (Brown et al., 2020), and we do so in the ATIS domain, but API limitations in GPT3 together with the requirements of our constrained decoding force us to use GPT2-XL (Radford et al., 2019) in SMCalFlow.

4.3 Filtering Silver Data

Many (c, n) pairs we generate may be low-quality, depending on the task and seed data $\mathcal{D}$ available. To obtain more high-quality pairs, we simulate 20 natural utterances n for each c. We must then filter the resulting pairs, which we do based on either reranking or cycle consistency.

Reranking (RANK). We accept the best of 20 simulated n for each c, and add this (c, n) to our training data. The reranker combines two scores: (1) the log-probability that the original $\mathcal{D}$-trained parser P parses n back to the original canonical c, and (2) the edit distance between n and c (capped based on the length of c), which should intuitively be *maximized* to encourage linguistic diversity.

Cycle consistency (CYC). We accept a (c, n) pair if the original parser P parses n back to c. This assures the resulting pairs' quality, but may skew the distribution toward easier examples, which are less helpful in downstream training.

5 Experiments

Tasks. We evaluate on two domains:

1. ATIS (Hemphill et al., 1990), a flight booking dataset. We use the Break (Wolfson et al., 2020) subset.[2]
2. SMCalFlow (Andreas et al., 2020), a calendaring dataset, which we view as the most complex and realistic.

In each domain, we assume a seed data $\mathcal{D}$ of just 30 pairs, conducting several trials with different samples of seed data to mitigate noise from this selection. We sample 300 unlabeled natural utterances $\mathcal{U}$ from the dataset, which must be parsed to canonical forms (using the grammar and SCFG of Shin et al. (2021)) and then PII-masked before use. See Appendix A for details on PII masking.

Methods. We evaluate several methods on each task, listed below.

1. BASE, a supervised baseline which finetunes BART (Lewis et al., 2019) on the seed $\mathcal{D}$ following Shin et al. (2021), discarding $\mathcal{U}$.
2. USER-RANK, a data augmentation approach following the USER and RANK methods de-

[2] We also ran preliminary experiments on the DROP (Dua et al., 2019) and NLVR2 (Suhr et al., 2018) subsets of Break, but found that the canonical utterances were too unnatural for any method to perform reasonably (Appendix B).

Method	ATIS	SMCalFlow
BASE	6.8 ± 3.5	13.2 ± 3.4
USER-RANK	13.4 ± 4.1	15.5 ± 3.7
GPT-RANK	13.7 ± 3.2	15.9 ± 2.7
USER-CYC	6.0 ± 2.3	15.0 ± 4.0
GRAM-RANK		13.4 ± 2.8
USER-RANK-3X		$\mathbf{17.6 \pm 4.6}$
GPT-RANK-3X		16.1 ± 3.0

Table 2: *Main results on ATIS and SMCalFlow for different methods.* Top-1 parsing match percentage evaluated over 5 (ATIS) or 10 (SMCalFlow) trials on different seed datasets $\mathcal{D}$. For the two highest-performing methods, USER-RANK and GPT-RANK, we iterate data augmentation 3 times on SMCalFlow, yielding USER-RANK-3X and GPT-RANK-3X. USER-RANK-3X performs best overall.

scribed in Sec. 4.1 and 4.3 respectively, and depicted in Figure 1.

3. GPT-RANK, a similar approach which generates c following GPT from Sec. 4.1 instead.
4. USER-CYC, a version which filters (c, n) pairs via cycle consistency (Sec. 4.3).
5. GRAM-RANK, a weak baseline that samples initial c directly from the grammar, which we run only on SMCalFlow since our ATIS grammar is too loosely specified for sampling.

Results. We observe that our best data augmentation methods (USER-RANK, GPT-RANK) double the performance of the baseline finetuning method BASE on ATIS, and outperform it on SMCalFlow by up to 33% relative gain (Table 2),[3] although absolute performance remains low due to the tiny amount of seed data. Interestingly, GPT-RANK outperforms BASE despite using only the seed $\mathcal{D}$, and not extra unlabeled $\mathcal{U}$. Moreover, iterating the data augmentation procedure can further improve performance, by improving the initial parser P used for parsing unlabeled $\mathcal{U}$ or for filtering pairs (c, n). However, USER-CYC performs poorly on ATIS, indicating that the CYC filtering is perhaps too restrictive. GRAM-RANK is also poor: sampling plans directly from a grammar is ineffective in a complex, realistic domain like SMCalFlow.

5.1 Analysis

We conduct additional analyses on SMCalFlow.

Reranking. First, we run ablations on reranking in USER-RANK (Table 3). While our edit distance heuristic described in Sec. 4.3 makes little difference, reranking of some form is crucial. Meanwhile, there are many possibilities for other reranking procedures.

[3] Although the standard deviations appear large, the variation between trials is largely due to randomness in selecting the seed data $\mathcal{D}$. For example, USER-RANK is better than BASE on SMCalFlow with $p = .0004$ on a paired t-test.

Method	SMCalFlow
BASE	13.2 ± 3.4
USER-RANK-3X	17.6 ± 4.6
USER-NOEDITRANK-3X	17.3 ± 4.7
USER-NORANK	12.8 ± 3.5

Table 3: *SMCalFlow reranking ablations.* Since the version without reranking (USER-NORANK) is no better than the baseline, we do not iterate the data augmentation procedure. The edit distance heuristic makes little difference in this case (USER-NOEDITRANK-3X vs. USER-RANK-3X), but reranking is crucial.

Method	SMCalFlow (100 seed data)
BASE	31.6 ± 0.3
USER-RANK	31.7 ± 1.0

Table 4: *SMCalFlow results with more seed data.* We use a seed dataset $\mathcal{D}$ of size 100 rather than 30, with 3 trials per method. The gains from data augmentation largely disappear at this scale, so we do not do additional augmentation iterations.

Additional Seed Data. We explore using a larger seed dataset $\mathcal{D}$ of size 100. On SMCalFlow, we observe that USER-RANK's gains over the baseline largely disappear (Table 4). Thus, improved data augmentation methods which still yield gains with larger seed datasets are an important direction for future exploration.

6 Discussion

We have presented a difficult setting for semantic parsing based on real-world resource and privacy constraints. In addition to a seed dataset, the only resources allowed are unlabeled natural utterances which must be PII-masked. Nevertheless, we observe that data augmentation approaches leveraging pretrained language models can still improve over supervised baselines which use only the seed dataset. At the same time, substantial room remains for additional improvement: there are many alternatives to our reranking procedure for silver data, and our method loses some effectiveness when more labeled data is provided. We hope that our observations in this challenging but realistic task setup can lay a foundation for further exploration in data augmentation approaches for semantic parsing.

Ethical Considerations

We believe our work makes a positive impact by focusing heavily on the need for privacy considerations when exploring low-resource settings for semantic parsing. However, as our methods rely heavily on large pretrained language models such as GPT3, we may inherit similar biases which such models are known for (Brown et al., 2020).

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, et al. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Pavel Burnyshev, Valentin Malykh, Andrey Bout, Ekaterina Artemova, and Irina Piontkovskaya. 2021. A single example can improve zero-shot data generation. *arXiv preprint arXiv:2108.06991*.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, pages 341–350.

Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. 2020. Unsupervised dual paraphrasing for two-stage semantic parsing. *arXiv preprint arXiv:2005.13485*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978*.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964.

Sungdong Kim, Minsuk Chang, and Sang-Woo Lee. 2021. Neuralwoz: Learning to collect task-oriented dialogue via model-based simulation. *arXiv preprint arXiv:2105.14454*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. *arXiv preprint arXiv:2104.07540*.

Nathan Schucher, Siva Reddy, and Harm de Vries. 2021. The power of prompt tuning for low-resource semantic parsing. *arXiv preprint arXiv:2110.08525*.

Richard Shin, Christopher H Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. *arXiv preprint arXiv:2104.08768*.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2018. A corpus for reasoning about natural language grounded in photographs. *arXiv preprint arXiv:1811.00491*.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable dialogue systems and user simulators. *arXiv preprint arXiv:2107.11904*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Shan Wu, Bo Chen, Chunlei Xin, Xianpei Han, Le Sun, Weipeng Zhang, Jiansong Chen, Fan Yang, and Xunliang Cai. 2021. From paraphrasing to semantic parsing: Unsupervised semantic parsing via synchronous semantic decoding. *arXiv preprint arXiv:2106.06228*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350.

Pengcheng Yin, John Wieting, Avirup Sil, and Graham Neubig. 2021. On the ingredients of an effective zero-shot semantic parser. *arXiv preprint arXiv:2110.08381*.

Zijian Zhao, Su Zhu, and Kai Yu. 2019. Data augmentation with atomic templates for spoken language understanding. *arXiv preprint arXiv:1908.10770*.

Victor Zhong, Mike Lewis, Sida I Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. *arXiv preprint arXiv:2009.07396*.

A Masking and Replacing Personal Identifiable Information

A.1 ATIS

The ATIS grammar is somewhat loosely defined and does not clearly indicate the instances of PII. This would be problematic in a real production setting due to making it difficult to guarantee masking out all PII. However, for our experiments we simply truecase the data and apply named entity recognition using spaCy (Honnibal et al., 2020), which we find is highly successful from a qualitative inspection. We treat detected named entities as PII.

To sample new values for replacing PII, we prompt GPT3 using the masked utterance together with the prefix up to where the PII string appears. We additionally contextualize with examples of unmasked-masked pairs from the seed data.

A.2 SMCalFlow

Since the SMCalFlow grammar is type-annotated, we define three categories of PII: names, event titles, and locations. Each category is easily identifiable from the logical form, so it suffices to sample a new value from the same category in the logical form to guarantee that PII is replaced.

We sample names from a distribution balanced for ethnicity and gender. For event titles and locations, we sample them from GPT3 by prompting with seed data canonical forms containing event titles and/or locations, and then prefixing the generation with `find event called something like "` (event titles) or a mix of `weather at "` and `find event at "` (locations). We cut off the generation once the next `"` appears.

B Preliminary Experiments on Other Break Subsets

We additionally ran preliminary experiments on the DROP (Dua et al., 2019) (reading comprehension) and NLVR2 (Suhr et al., 2018) (language-vision reasoning) subsets of Break (Wolfson et al., 2020). We used a similar setup to our ATIS and SMCalFlow experiments, with 30 initial seed data $\mathcal{D}$ and 300 unlabeled user utterances $\mathcal{U}$.

However, across multiple trials of multiple methods (BASE, USER-RANK, GPT-RANK, USER-CYC), we never observed performance above 2% on either domain. This may be partially due to the diversity of the data; for example, DROP is an amalgamation of data from several sources. However, we hypothesize that this across-the-board poor performance is primarily the result of an SCFG for canonical utterances which results in somewhat unnatural language (Table 5), and that performance could be greatly improved with a better SCFG. Given the current form of our canonical utterances in DROP and NLVR2, it is challenging to learn the task given just 30 seed examples. In comparison, the SMCalFlow canonical utterances (Table 1 in the main text) are much more natural.

DROP Natural	*Which player had the shortest touchdown reception of the game?*
DROP Canonical	return touchdown receptions ;return shortest of #1 ;return player of #2
NLVR2 Natural	*If there are two carts, but only one of them has a canopy.*
NLVR2 Canonical	return carts ;return number of #1 ;return if #2 is equal to two ;return canopy ;return #1 that has #4 ;return number of #5 ;return if #6 is equal to one ;return if both #3 and #7 are true

Table 5: Examples of natural utterances with corresponding canonical utterances for DROP and NLVR2 domains. The language of the canonical utterances is relatively unnatural.

We additionally inspect some inaccurate example predictions by BASE on DROP and NLVR2, which are often wildly incorrect (Table 6). We also show some example (c, n) pairs generated by our data augmentation procedure, demonstrating the failure to propose good natural language n given the limited data and unnatural canonical c (Table 7).

DROP Natural	*Which player threw more yards in the game, Young or Manning?*
DROP Top-1 Parse	return that was the highest ;return that was more of #1 ;return number of #2 for each #1 ;return #1 where #3 is lower than one ;return number of #4
NLVR2 Natural	*If there are bananas with stickers on them*
NLVR2 Top-1 Parse	return, ;return number of #1 ;return if #2 is equal to one

Table 6: Predictions by BASE on DROP and NLVR2 which are wildly incorrect. Our data augmentation methods fare no better.

DROP Canonical	return the five
DROP Simulated Natural	Fact-checkers failed to catch five factual errors.
NLVR2 Canonical	return left image ;return #1 that are dirty ;return if #2 is in one of the images
NLVR2 Simulated Natural	If any of the trucks are dirty.

Table 7: Example simulated natural utterances generated by prompting GPT3 on DROP and NLVR2, after reranking and selecting the best of 20 generations. The correspondence between canonical and simulated natural utterances remains imperfect.