1 Dataset Details

1.1 Pizza

Pizza[1] is a recently introduced dataset consisting of English utterances that represent orders of pizzas and drinks. The target parse is a LF that specifies the various components of the relevant pizza and drink orders. Examples from this dataset can be seen in Table 1. Since our system uses canonical forms as targets instead of LFs, we defined a canonicalization scheme for pizza and drink orders via a rule-based parser that can go from the canonical form to the LF and conversely (details in the next section).

The original Pizza dataset contains 2.5M synthetic training examples, 348 dev examples, and 1357 test examples. For our experiments, we ignore the synthetic training data and use the 348 dev examples as the training set to choose sets of 16, 32, and 48 examples for low-resource training.

To create the mask-infilling data, we include utterances from the unselected examples. For the denoising task, we randomly sample 10k target parses from the original synthetic training set of 2.5M examples and construct their canonical forms. This is akin to generating random pizza orders since that is how the synthetic dataset was created in the first place.

1.2 Overnight

Overnight is a popular semantic parsing dataset that consists of 13,682 examples across eight domains. The task is to convert natural language utterances to database queries, which are then executed on a fixed database to obtain the results for the user utterances. This dataset was originally generated by first creating canonical utterances and their parses (database queries) and then paraphrasing the canonical utterances using crowd sourcing to obtain the natural utterance. As a result, we have access to the utterance, canonical form and the corresponding database query for all examples. An example from the basketball domain is the utterance *which team did kobe bryant play on in 2004*, whose canonical target is *team of player kobe bryant whose season is 2004*.

Just like with the Pizza dataset, we use the remaining utterances to create mask-infilling data. To generate queries for the denoising task, we use the SEMPRE toolkit, upon which the Overnight dataset was built, to generate sample queries for each domain from its canonical grammar, consisting of around 100 general and 20-30 per-domain rules. For paraphrase augmentation, for both datasets, we generate four paraphrases for each utterance in the training set. We use the BART-Large model trained on ParaNMT data and take the top four sequences from beam search decoding at inference.

For constrained decoding, we construct a large trie that contains all the canonical form sequences, and use it to look up valid next tokens given a prefix.

2 Pizza Canonical Forms

The Pizza dataset consists of natural-language utterances representing pizza (and/or drink) orders, along with corresponding LFs. For our experiments, however, we needed natural-language sentences as the target outputs. Unlike the Overnight dataset, Pizza doesn't come with a canonical-form grammar. Accordingly, we created our own grammar and rules to convert LFs to and from canonical utterances. We describe these below.

Every target LF in the Pizza dataset consists of one or more pizza and/or drink orders. Each pizza order contains various attributes such as number, size, style, toppings, and so on. Some of these attributes, such as complex toppings (which contain a topping and a quantity qualifier, like *extra* cheese) are nested. Likewise, each drink order has attributes such as number, size or volume, container type (can or bottle), and so on. Given some LF tree t with orders $o_1, o_2 \ldots o_n$, we express the canonical form of t, $CF(t)$, in terms of the canonical forms of the individual components of t as follows:

$$CF(t) = \textbf{i want } CF(o_1), CF(o_2), \ldots \textbf{ and } CF(o_n)$$

Each pizza/drink component order is further naturalized to create the canonical form sequence specified by the above expression. For a pizza order, this string captures the pizza attributes, while for a drink order it captures the drink attributes. The following expressions roughly describe how these strings are laid out for a pizza order p and a drink order d in terms of their various attributes, represented in angle brackets $\langle\rangle$ (multi-valued attributes have a starred superscript).

$$\begin{aligned} CF(p) &= \langle \textit{number} \rangle \langle \textit{size} \rangle \langle \textit{style} \rangle^* \textbf{pizza with} \\ &\quad \langle \textit{topping} \rangle^*, \textbf{and no } \langle \textit{topping} \rangle^*, \\ &\quad \textbf{not } \langle \textit{style} \rangle^* \textbf{style} \\ CF(d) &= \langle \textit{number} \rangle \langle \textit{size} \rangle \langle \textit{volume} \rangle \\ &\quad \langle \textit{drink name} \rangle \langle \textit{container} \rangle \end{aligned}$$

We simply skip filling an attribute value if it doesn't exist for an order. For further nesting such as with complex toppings, the canonical string is a concatenation of the values of all its attributes. The constructed canonical forms for all examples in the train, dev, and test sets are available in the data zip file. Table 1 provides a few sample canonical forms and their corresponding utterances for reference.

3 Further experimental details

We provide a few more details on our experimental settings here. Note that we didn't perform extensive hyper-parameter tuning. This section is simply to serve as a guideline for reproducing results reported in this paper.

3.1 Auxiliary Tasks

For the mask infilling task, we use all the available unannotated utterances to create source and target sequences. For our experiments, we upsample by $10\times$ and mask a random span of 25% tokens in each example. So for the pizza dataset for example, the size of this data in the $n = 16$ setting is $(348 - 16) \times 10 = 3320$. The same holds for the Overnight dataset. The mask infilling source and target sequence files for both the Pizza and Overnight datasets and all the reported experimental settings are available in the data zip file.

[1] https://github.com/amazon-research/pizza-semantic-parsing-dataset

Utterance	Canonical Form
get me three pepsis five medium diet sprites and a coke	i want one coke , five medium diet sprite , and three pepsi
i need a medium ham pizza with extra cheese and pesto	i want one medium pizza with extra cheese , ham , and pesto
can i have a large pizza along with onions tuna and add some thin crust	i want one large thin crust pizza with onions and tuna
good evening how are you do me a favor and get me a large pizza with ham and peppers i definitely do not want thin crust thanks	i want one large pizza with ham and peppers , not thin crust style
i wish to have one pie in large size along with olives and chicken but without ham	i want one large pizza with chicken and olives and no ham

Table 1: Example utterances and canonical form for the Pizza dataset

For the denoising task, we sample 10k random LFs from the synthetic training data for the Pizza data and apply noise. For the Overnight dataset, we use SEMPRE to generate canonical forms and upsample them until they reach 10k and then apply noise. So the dataset size is always 10K in all our experiments. The source and target sequence files for the denoising task are available in the data zip file.

3.2 Computational Resources

We train our BART Paraphrasing model on a 8x32GB GPU. We use a large GPU here since the training dataset contains around 5 million examples. For all the semantic parsing models reported in the paper, both the baselines and our models, we use a single 16GB GPU.

4 Full Analysis

In this section we analyze our results in more detail and also explain various design choices and the empirical results that motivated them. For most of the analysis experiments, we use the Pizza dataset, since, as the Results Section shows, the results mostly generalize across both datasets. The Pizza dataset also has a larger test set than any of the Overnight domains, which allowed us to see performance differences better.

4.1 Joint training vs Two-stage Fine-Tuning

Our approach employs joint training, where we combine the auxiliary task data with the annotated data and jointly train our model. Our intuition here was that this technique would make the training more robust and act as a regularizer. One could instead use the auxiliary tasks to pretrain the model and then fine-tune it on just the annotated data. We found that this two-stage training does not improve the model. Table 2 reports the results across the three data sizes on the pizza dataset for the JT and 2-stage fine-tuned models. We see a noticeable drop in performance with the extra fine-tuning step for 32 and 48 examples, and no significant boost for 16 examples.

4.2 Importance of the canonical form

We found canonical targets to work better in low-resource settings than LF targets, which stands to reason given that they are natural language sentences that can better leverage the pretraining of LMs. Moreover, for our JT technique, the canonical form provides us with an easy way to add meaningful noise without modifying the content tokens for the denoising auxiliary task. We can simply perform token level operations without worrying about the target structure. However, if the targets are parse trees, adding noise is trickier, since most of the tokens in the parse represent content and meaningful operations need to be performed at the tree level. Even more importantly, the target sequences for the mask prediction task are in natural language and are better aligned with the canonical form targets than the parse trees. This potentially allows for better knowledge transfer during joint training.

	Unordered EM Accuracy		
	n=16	n=32	n=48
Two-stage Fine-Tuning	**43.66**	59.47	67.87
Joint Training	42.23	**64.70**	**70.30**

Table 2: Comparing joint training to two-stage fine-tuning.

We performed a JT experiment with a model that predicts LFs instead of canonical forms for the Pizza dataset. We created the source sequences for the denoising auxiliary task using tree-level noise operations such as switching entities, dropping brackets, and inserting random tokens. We found that the resulting models achieved significantly lower scores than the models that use canonical targets. Table 3 reports these numbers. The numbers for LF targets are actually very close to those of their base architectures (for LF-based BART). So, as hypothesized, directly using LF trees as targets is not conducive to our joint training approach. We require canonical-form targets in the base architecture for JT to work effectively.

4.3 Other auxiliary tasks apart from denoising

The goal of our auxiliary tasks was to provide the model with a challenging objective. Using unlabeled utterances, we create the mask infilling task in the style of the BART pretraining objective. This is a standard domain adaptation technique that is well explored in the literature.

To train the decoder, we use the synthetically generated target sequences so that the decoder can train on, and learn to generate, a variety of valid canonical forms. To create a challenge for the decoder, we noise the targets to obtain corrupted source sequences and create a denoising task. This is

	Unordered EM Accuracy		
	n=16	n=32	n=48
Logical Form	19.90	57.26	59.10
Canonical Form	**42.23**	**64.70**	**70.30**

Table 3: Comparing canonical form targets to parse trees for the denoising task.

	Unordered EM Accuracy		
	n=16	n=32	n=48
Denoising	42.23	64.70	70.30
Synthetic Utterances	72.37	78.11	82.31

Table 4: Comparing denoising to synthetically generated semantic parsing as the auxiliary task.

a simple task that can be constructed to provide the decoder a challenge without requiring any external effort.

There are other possible tasks. One could create rules to generate synthetic utterances given the target parses. This synthetic data could then be used to train the decoder. That approach, however, requires manual effort and depends on the quality and diversity of the synthetic data. For Pizza, we already have access to synthetic data, since the entire training set is synthetic. Assuming we have access to a system that can generate such synthetic utterances given randomly generated target parses, we could replace our denoising task with the synthetic examples. We perform this experiment to compare these two auxiliary tasks. Table 4 shows these results. The synthetic parsing auxiliary task performs better than denoising but, as mentioned earlier, it requires a lot of manual effort to create a synthetic utterance grammar. Our JT approach is directly applicable to both tasks.