LINGUIST: Language Model Instruction Tuning to Generate Annotated Utterances for Intent Classification and Slot Tagging

Andy Rosenbaum* Amazon, Cambridge, USA andros@amazon.com
Saleh Soltan Amazon, New York, USA ssoltan@amazon.com
Wael Hamza Amazon, Dallas, USA waelhamz@amazon.com
Yannick Versley Amazon, Aachen, Germany yversley@amazon.de
Markus Boese Amazon, Aachen, Germany boesem@amazon.de

Abstract

We present LINGUIST, a method for generating annotated data for Intent Classification and Slot Tagging (IC+ST), via fine-tuning AlexaTM 5B, a 5-billion-parameter multilingual sequence-to-sequence (seq2seq) model, on a flexible instruction prompt. In a 10-shot novel intent setting for the SNIPS dataset, LINGUIST surpasses state-of-the-art approaches (Back-Translation and Example Extrapolation) by a wide margin, showing absolute improvement for the target intents of +1.9 points on IC Recall and +2.5 points on ST F1 Score. In the zero-shot cross-lingual setting of the mATIS++ dataset, LINGUIST outperforms a strong baseline of Machine Translation with Slot Alignment by +4.14 points absolute on ST F1 Score across 6 languages, while matching performance on IC. Finally, we verify our results on an internal large-scale multilingual dataset for conversational agent IC+ST and show significant improvements over a baseline which uses Back-Translation, Paraphrasing and Slot Catalog Resampling. To our knowledge, we are the first to demonstrate instruction fine-tuning of a large-scale seq2seq model to control the outputs of multilingual intent- and slot-labeled data generation.

1 Introduction

Conversational agents typically rely on large quantities of labeled training data to understand user requests through Intent Classification and Slot Tagging (IC+ST) (Tur and De Mori, 2011). Such data is plentiful for existing usage patterns (although costly to annotate), yet scarce for new intents/slots and new languages. A growing trend to address this problem is to generate synthetic training data, e.g. via Paraphrasing, Back-Translation (BT), slot replacement, and Example Extrapolation (Ex2). (Jolly et al., 2020; Xie et al., 2020; Zhang et al., 2020; Lee et al., 2021). In this work, we propose a novel data generation method called Language model INstruction tuning to Generate annotated Utterances for Intent classification and Slot Tagging (LINGUIST).

Our LINGUIST method addresses several important gaps in the existing literature: (1) controlling the generated data to include specific slot types and values, (2) cross-lingual and multilingual data generation, and (3) ability to leverage the intent

*Correspondence Author: <andros@amazon.com>. Author contributions are listed in Appendix A.
and slot names to inform the generation. The key to these achievements is our design of a novel instruction prompt (Figure 1), consisting of natural language descriptions for the desired model outputs. We first fine-tune a large pre-trained seq2seq Transformer (Vaswani et al., 2017) model to learn how to generate annotated utterances following the prompt instructions. Then, for a novel intent or slot with only a few or even zero training examples, we apply the model to generate similar data, which we add to the training set for an IC+ST model.

We demonstrate the effectiveness of LINGUIST on three datasets by showing substantial improvements over strong baselines. (i) On a 10-shot novel intent setting with English SNIPS (Coucke et al., 2018), LINGUIST improves over Back-Translation and Ex2 by +1.9 points absolute on IC and +2.5 points absolute on ST. (ii) On cross-lingual mATIS++ (Xu et al., 2020), LINGUIST out-performs the best Machine Translation plus slot alignment reported by Xu et al., by +4.14 points in ST across 6 languages, while matching performance on IC. (iii) Finally, to demonstrate the success of our method on a real-world conversational agent system, we apply LINGUIST on an internal dataset containing hundreds of intents and slot types across 4 languages, and show large improvements over a baseline which uses Back-Translation and Paraphrasing.

We also show LINGUIST can generate IC+ST-annotated data from zero examples, using only the natural language intent and slot label names. LINGUIST achieves 80.0 IC Recall and 56.9 ST F1 score on new SNIPS intents despite never seeing a single example for the new intents. To our knowledge, LINGUIST is the first system capable of generating IC+ST-annotated data in this setting.

2 Related Work

Large-scale Language Models (LLMs) such as GPT (Radford et al., 2019; Brown et al., 2020) and AlexaTM 20B (Soltan et al., 2022) excel at performing novel tasks with only a few examples via in-context learning, i.e. without requiring any model parameter updates. For example, Sahu et al. (2022) apply GPT-3 to generate variations of examples from a given class. Wang et al. (2021) generate both text and class label together via GPT-3, towards eliminating the need for human labeling.

Pre-training then fine-tuning of seq2seq models was introduced in English BART and multilingual mBART (Lewis et al., 2020a; Liu et al., 2020). T5 and mT5 (Raffel et al., 2020; Xue et al., 2021) extended the idea by framing more downstream tasks as text-to-text. FLAN (Wei et al., 2022) introduced instruction tuning, where a large-scale seq2seq model is fine-tuned on instruction prompts from a variety of tasks, in order to generalize to new tasks without any further parameter updates.

Prior work also explores conditioning generation on intent and slot labels. Ding et al. (2020) train a conditional language model on a mixture of annotated and unannotated text, allowing to sample novel annotated utterances. Malandrakis et al. (2019) train a seq2seq model from interpretation-text pairs, applying variational auto-encoders for more diversity. Jolly et al. (2020) expand on this, exploring different sampling strategies, adding more variety by shuffling slot names, and examining the behavior where a new intent is introduced with limited training data. Panda et al. (2021) extend this to the multilingual setting. Generative Insertion Transformers (Kumar et al., 2022) generate carrier phrases for a target intent and containing specific entities. A limitation of all these approaches is that the trained model cannot generalize to novel intents and slots at inference time.

Generative Conversational Networks (Papangelis et al., 2021) are trained via reinforcement learning to generate annotated data from seed examples, studying English IC+ST and other tasks.

The closest relative of LINGUIST is Example Extrapolation (Ex2) (Lee et al., 2021), which generates annotated IC+ST data using a seq2seq model and provided seed examples. We compare LINGUIST and Ex2 in more detail in section 3.3.

A widely used paraphrasing method is Back-Translation (BT), i.e. translating text from one language into another “pivot” language, then back again. Bannard and Callison-Burch (2005) extract paraphrases directly from parallel corpora. Senrich et al. (2016) and Edunov et al. (2018) use BT for Machine Translation and Xie et al. (2020) for data augmentation on classification tasks.

Other approaches directly target the Paraphrasing task: Prakash et al. (2016) learns an LSTM model by supervised training on a paraphrase corpus, whereas Kumar et al. (2020) use an unsupervised denoising task, in both cases only using text and not covering slot labels. Cho et al. (2019) explore Paraphrasing via Semi-Supervised Learning.

A different approach to data augmentation is
We control the generation output via a novel prompting scheme, as shown in Figure 1. The prompt contains five blocks: (i) the output <language>, (ii) the <intent> name, (iii) which slot types and values to <include> in the output, (iv) a mapping from <labels> to numbers, and (v) up to 10 <examples>, each belonging to the same intent, and including zero or more of the available labels.

The <include> block instructs LINGUIST which slot types and values to generate, such as [3 snow], where the number corresponds to the slot type (in this case, 3=condition_description), and the content inside the brackets is the value to use for that slot (here “snow”). To increase the diversity of outputs, the user may also instruct the model to generate a value for a slot, using the “wildcard” token, e.g. [1 * ] indicates that the model should sample a value for slot number 1.

As shown in Figure 1, LINGUIST learns to produce a rich sample of values even for intents and slot types it never saw during fine-tuning. For example, in this case the wildcard is for [1=geographic_poi] and the model outputs sensible values such as “Lake Tahoe”, despite this phrase never appearing the fine-tuning data.

3 LINGUIST Data Generator Model

LINGUIST provides three key innovations compared to prior data generation work: (1) controls for slot types and values (either user-supplied or model-generated) to include in the outputs; (2) multilingual and cross-lingual generation; and (3) ability to leverage the natural language label names to inform generation, enabling a new “Label Names Only” setting (Section 4.2.2).

3.1 LINGUIST Prompt Design

We fine-tune a pre-trained seq2seq model on pairs of LINGUIST instruction prompts and corresponding annotated target utterances, derived from a de-duplicated IC+ST task dataset. Specifically, we format an instruction prompt \( p_i \) targeting each utterance \( t_i \in R \), including in \( p_i \) up to 10 other example utterances \( E = \{ e_j \}_{j=1}^{10} \in R \) s.t. \( \forall j, e_j \neq t_i \) and \( \text{intent}(e_j) = \text{intent}(t_i) \). To make the generation robust to the number of provided examples, we do not always include all 10 in the prompt, but instead randomly select \( k \) examples from \( E \), with \( k \) chosen randomly between 0 and 10, or the number of utterances available that share the same intent as \( t_i \), whichever is smaller. We never duplicate utterances in the prompt. Finally, we produce a corpus of training prompts equal in size to the original IC+ST training set.

To reduce the tendency for the model to overfit on the intent and slot labels (as observed by Lee et al., 2021), we drop out the label names for both at a rate of 0.2, replacing the label name e.g. GetWeather with a random sequence of between 1 and 5 letters like A_0_Y. (Ablation in Appendix E.) The intuition for masking the labels rather than skipping the <intent> and <labels> blocks is to provide the model a consistent signal for position embeddings, and always allowing it to attend to these tags if it wishes to.

To jointly teach the model both to copy user-supplied slot values like [3 snow ] and to produce appropriate values for the wildcard [1 * ] we format the training prompts with examples of both. For the prompt \( p_i \) targeting utterance \( t_i \), we randomly select from a Geometric distribution \( d \sim \text{Geo}(0.5) \) (0 \( \leq d \leq \# \) slots in \( t_i \)) slots and replace their values with "*" in the <include> block of the prompt. The effect is that approximately 50% of utterances have all slot values replaced by the wildcard token, 25% of utterances keep one slot value, etc.

We do not add the tags <intent>, [1, etc. to the model’s sentencepiece (Kudo and Richardson, 2018) tokenizer vocabulary (Appendix C.2).

3.2 Training the LINGUIST model

3.3 Comparing LINGUIST to Ex2

The closest relative to our approach is Example Extrapolation (Ex2, Lee et al., 2021), which also produces slot-labeled text from seed examples. The novelty of LINGUIST compared to Ex2 is threefold: (i) instructions to control the slot types and values
generated, (ii) multilingual and cross-lingual, and (iii) the ability to include label and slot names in the prompt. In particular, EX2 showed that anonymizing the labels improved on IC however hurt ST. Our LINGUIST model improves over our implementation of Ex2 on both IC and ST, and furthermore the labels enable LINGUIST to perform one-shot and zero-shot for new intents and slots, as we show in Section 5.1.3 and Appendix B.

4 Experimental Setup

This section describes the datasets, tasks, IC+ST model, baseline gata generation methods, and metrics that we use to evaluate LINGUIST.

4.1 Datasets

4.1.1 SNIPS Dataset

The SNIPS dataset (Coucke et al., 2018) is a public IC+ST benchmark consisting of 7 intents, each with between 2 and 14 slot types (39 unique slot types in total). It includes around 2k training utterances and 100 validation utterances per intent. In order to avoid overfitting our method on the small validation set, at the beginning of our experiments, we partition the training set into 97% Train and 3% Validation. We use our Development set split for iterating on all modeling and data processing decisions, including the hyperparameters for LINGUIST and hyperparameters and selection of best checkpoint for the encoder fine-tuning on IC+ST. Only at the very end of our experiments, we evaluate and report on the Validation set. See Table 1 for counts of Train/Dev/Valid utterances.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Train</th>
<th>Dev</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>1884</td>
<td>58</td>
<td>100</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>1914</td>
<td>59</td>
<td>100</td>
</tr>
<tr>
<td>GetWeather</td>
<td>1940</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>1940</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>RateBook</td>
<td>1898</td>
<td>58</td>
<td>100</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>1896</td>
<td>58</td>
<td>100</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>1901</td>
<td>58</td>
<td>100</td>
</tr>
<tr>
<td>Total</td>
<td>13373</td>
<td>411</td>
<td>700</td>
</tr>
</tbody>
</table>

Table 1: Data counts per intent for SNIPS.

4.1.2 Multilingual ATIS++

For cross-lingual experiments we evaluate on mA-TIS++ (Xu et al., 2020), which consists of human-translated text and annotations from the original English travel information requests ATIS dataset (Hemphill et al., 1990) plus Hindi and Turkish translations from mATIS (Upadhyay et al., 2018). Our experiments cover the 7 languages that mA-TIS++ shares with our pretrained model: English, Spanish, German, French, Portuguese, Japanese, and Hindi, with 4488 (HI: 1440) training utterances covering 18 (HI: 17) intents and 84 (HI: 75) slots.

To demonstrate the cross-domain adaptation of the LINGUIST method, we use the MASSIVE dataset (FitzGerald et al., 2022b) covering 51 languages with parallel versions of the (English-only) SLU Resource Package (Bastianelli et al., 2020) utterances, covering 20k utterances per language across 18 domains, 60 intents and 55 slots. We use MASSIVE only to train a LINGUIST model, then apply the model to mATIS++. In order to keep mATIS++ as a novel domain, we exclude the somewhat related transport domain from MASSIVE when we train the LINGUIST model.

4.1.3 Internal Dataset

To demonstrate the value of our method to a real-world setting, we benchmark on an internal large-scale multilingual dataset representative of requests to a conversational agent. We consider five portions of the dataset, known as features, namely: CameraControl, ClockSettings, HomeSecurity, Music, and Timers, each containing one or more intents, and one or more associated slots. For each feature, there is a “starter” training set comprised of a few dozens of annotated utterances which were curated for the new feature, and a test set containing hundreds of annotated utterances pertaining to that new feature. Additionally, there is a large training dataset $E$ of annotated utterances from existing features. The Existing Features training data $E$ does not contain examples of any of the new features.

4.2 Evaluation Tasks

4.2.1 New-Intent Few-Shot (NIFS)

As shown in Figure 2, we simulate the introduction of a new intent into an existing well-resourced dataset. Given a training dataset $R = \bigcup_{j=1}^{m} D_j$ containing data $D_j$ for $m$ intents $j = 1, \ldots, m$, we select an intent $i \in \{1, \ldots, m\}$, and reduce its training data to only a small number $K$ of “starter” utterances $S_i \subset D_i$. We apply various data augmentation techniques on $S_i$ to create augmented data $A_i$. Finally, we train an IC+ST model using $R'_i = S_i \cup A_i \cup \{D_j\}_{j \neq i}$, i.e. the concatenation of starter and augmented data for intent $i$ with the unmodified data for all other intents.

The internal dataset is already split in this way, however at the feature rather than intent level.
4.3 Models

Our experiments rely on two pre-trained models: (1) **AlexaTM 5B** (described next in Section 4.3.1) which we fine-tune on the LINGUIST prompts and use it to generate IC+ST training data, and (2) **xlm-roberta-base** which we fine-tune on the IC+ST task including the data generated from LINGUIST.

4.3.1 AlexaTM 5B

AlexaTM 5B is a multilingual seq2seq Transformer (Vaswani et al., 2017) model pre-trained similar to AlexaTM 20B (Soltan et al., 2022), however with denoising objective only (i.e. without Causal Language Modeling objective). Like AlexaTM 20B, the architecture is derived from the HuggingFace (Wolf et al., 2020) BART (Lewis et al., 2020b) class, and consists of 29 encoder and 24 decoder layers, hidden dimension 2560, and 32 attention heads. The model is trained on 900B tokens of Wikipedia and mC4 (Xue et al., 2021) of 12 languages as used in AlexaTM 20B. We used a maximum sequence length of 512 and a batch size of 1M tokens. The encoder weights of AlexaTM 5B are initialized with the 2.3B-parameter Alexa Teacher Model encoder (FitzGerald et al., 2022a) trained on MLM task on the same data, frozen during the first half of the AlexaTM 5B pre-training. Hyperparameters used for fine-tuning AlexaTM 5B on LINGUIST are described in Appendix C.

4.3.2 IC+ST Fine-tuning

For SNIPS and mATIS++, following Chen et al. (2019), we fine-tune a BERT-style model for joint IC+ST. On top of the encoder hidden states, we attach two separate classification heads, one for IC and another for ST. Each head consists of two layers of 256 hidden dimension, with gelu activation, dropout 0.2, and layer norm. The IC head utilizes representation from the first token of the sequence ([CLS]), while the ST head utilizes the first subword token of each word.

For our encoder, we use **xlm-roberta-base** (Conneau et al., 2020) (12 layers, 768 hidden dimension), from the HuggingFace (Wolf et al., 2020) implementation. We fine-tune with batch size 128 for 3k updates (i.e. 30 epochs for the full-size data). We freeze the embedding layer; all other parameters are free to update during training. We use Adam (Kingma and Ba, 2015) with peak learning rate 3e-5, increased linearly from 0 to 600 updates, then decayed linearly to 0 until the end of training.
To avoid over-fitting on the official SNIPS Validation dataset, we use our Development split (Section 4.1.1) for early stopping, selecting the checkpoint with best performance on ST. All of our IC+ST fine-tuning runs for SNIPS use identical hyper-parameters, regardless of the data generation method being explored. For each data generation method, we train and test 7 different Joint IC+ST models \( \{M_i\}_{i=1}^7 \) in NIFS setting: each using a combination of the modified data for intent \( i \), and the unmodified data for all other intents.

For mATIS++, we follow the same model architecture settings and train for 2k updates (64 epochs for English only data, or 9 epochs when using data from all the 7 languages.) We select the checkpoint with best ST F1 Score on the English dev set only.

For our internal benchmark, we use similar settings, however with a smaller internal Transformer-based encoder for fine-tuning on the IC+ST task.

4.4 Baseline Data Generation Methods

The Interpretation-Conditioned Language Model (ICLM) Jolly et al. (2020) generates unlabeled text conditioned on intent and provided slot values, with a separate label projection step to recover the full slot annotation. ICLM does not generate novel slot values. Our implementation uses a small Transformer (Vaswani et al., 2017) architecture with \( \sim 37 \text{M} \) parameters, and a simple character-level Levenshtein distance measure to project the slot labels. We produce 50 outputs per input, then filter/de-duplicate (see Appendix J).

We apply Back-Translation (BT) using two separate MT systems to show the influence of the translation model. The first uses the open-source Sockeye toolkit (Hieber et al., 2018) and a small (91M parameters) Transformer which has been fine-tuned on around 10k utterances of annotated parallel data. We use fast_align (Dyer et al., 2013) to project the slot labels to the generated utterances. We call this system “BT-Small”. We use \( M=1 \) forward and \( N=10 \) backward translations to obtain 10 paraphrases, and then filter and deduplicate (see Appendix K). We use French (SNIPS) or English (Internal) respectively as pivot languages.

For a stronger BT baseline “BT-5B”, we build an MT system by fine-tuning AlexaTM 5B on WMT14 (retrieved from HuggingFace datasets) jointly on en→fr and fr→en using an instruction prompt (prefixing the input text with Translate to French: or Translate to English:, respectively) to control the translation direction. We use SimAlign (Jalili Sabet et al., 2020) to project the slot labels to the paraphrased text. For SNIPS, we use French as the pivot language, with beam search 10 both forward and backward, producing 100 outputs per original sentence, then filter and de-duplicate the outputs (Appendix L). BT-5B was not available for the Internal Benchmark.

For SNIPS, we implement Example Extrapolation (Ex2, Lee et al. (2021)) with the default “fully anonymized labels” setting, again fine-tuning from AlexaTM 5B, training a separate version for each intent’s experiment, as described in 4.2.1.

Slot Catalog Resampling is a simple approach to data augmentation which samples entities from a catalog for a particular label. For example, given an utterance like “play jason mraz” we might sample “weeezer” from a catalog of artist names, to get “play weeezer”. We only use Slot Catalog Resampling for the Internal Benchmark, as there are no slot catalogs available for SNIPS or mATIS++.

4.5 Metrics

4.5.1 Metrics for SNIPS and mATIS++

We use separate metrics to measure (1) support for the new intent, while (2) not harming the overall performance across all intents. For (1), we run the model on a test set containing only the new intent. We refer to this as the Local Intent Recall (IR), and Local ST F1 Score. To measure (2), we run the model on the combined test set of all intents together, and call this the Global Intent Accuracy (IA) and Global ST F1 Score. In both cases, for ST F1 Score, we ignore the “O” (non-entity) tag, using the seqeval (Nakayama, 2018) implementation.

When training data is modified for a particular intent \( i \), the Local metrics for \( i \) change across methods as expected, whereas changes in Global metrics (see Appendix G) are very small for all methods.

For the cross-lingual mATIS++ experiments, we report (Global) intent accuracy and Slot F1, since we are doing cross-lingual transfer for the whole dataset, and not targeting specific intents.

4.5.2 Metrics for Internal Benchmark

For the internal benchmark, we only evaluate in the Local setting. We measure Semantic Error Rate (SemER: Su et al., 2018 or Appendix O) which jointly evaluates the IC and ST performance. Lower SemER indicates improvement to the system. We report relative reduction in SemER, where a negative number indicates improvement.
We train 7 versions of the \( \text{INGUIST} \) model one for each heldout intent, as described in section 4.2.1. We train a joint IC+ST encoder on the combination of data from intent \( i \) modified according to strategy \( j \), and all other intents’ data unmodified. “Full” is trained on the full dataset without any modifications; for “s10-NoUps”, the data for intent \( i \) is reduced to only 10 “starter” examples, and are Not Up-sampled; for “s10”, the starter utterances are up-sampled to \( N_i \), the original data size for intent \( i \). For the remaining columns, the up-sampled starter utterances for intent \( i \) are mixed with augmented data derived from them using a particular method, which is re-sampled to \( N_i \) in size. “s10+X” uses ICLM. BT-Small, BT-5B, respectively. “s10+Ex2” uses our internal 5B seq2seq model with Ex2, “s10+LINGUIST” uses data generated by our LINGUIST method. We bold (underline) the mean for the method with best (second best) results. Experiments are run across five random seeds, as mean ± standard deviation.

5 Results

5.1 SNIPS Results

The main results are presented in Table 2a for Local Intent Recall and Table 2b for Local ST F1 Score.

5.1.1 Baseline Results on SNIPS

An upper bound for the New-Intent Few-Shot (NIFS) setting, is a model trained on the full dataset, which we train and report (“Full” in the tables) at 99.2 for Local Intent Recall and 96.6 for Local ST F1 Score. Reducing to 10 utterances (“s10-NoUps”) harms both IC and ST, (although ST more substantially), however simply up-sampling (duplicating) the starter utterances (“s10”) recovers a sizeable portion of the performance lost.

The rest of the columns use a mix (weighted 0.5/0.5) of the up-sampled 10 starter utterances, plus augmented data derived from them via the specified methods. In all cases, we resample the final amount of data for the target intent to match the count in the original unmodified dataset.

We find that ICLM and BT-Small do not improve on Local Intent Recall or Local ST F1 Score compared to “s10”, whereas BT-5B is a strong baseline, achieving 90.1 vs 88.2 for IC and 79.2 vs 77.7 for ST. Compared to BT-5B, Ex2 matches for IC at 90.0 and only slightly improves for ST at 79.8.

5.1.2 LINGUIST Results on SNIPS

We train 7 versions of the LINGUIST model one for each heldout intent, as described in section 4.2.1.

Utilizing the ability of LINGUIST to both copy slot values and produce novel values, we format multiple prompt versions \( p_{ik} \) from each starter utterance \( s_i \). The first, dubbed “copy-all” instructs LINGUIST to copy all the slot values, while producing new carrier phrases. Note that LINGUIST may also re-order the slots in the sentence.

Then, for each slot type \( k \), we create a new version of the prompt replacing the value for \( k \) with the wildcard “*”, instructing LINGUIST to produce a new value for the slot, while copying the other slot values as they are, and generating a suitable carrier phrase. We refer to this strategy as “sample-each”. We use top_\( k \) sampling with \( k = 50 \) and tempera-
0.3 to generate 100 utterances per prompt.

After filtering the generated utterances (see Appendix M for details), we mix the up-sampled 10 starter utterances with the LINGUIST-generated data. We use identical settings for LINGUIST fine-tuning and generation across all 35 runs (7 intents times 5 random seeds) for the SNIPS-NIFS benchmark. Following the setting of the other baselines, we fine-tune the IC+ST model on the concatenation of the augmented and mixed data for intent \( i \) with the original data for all other intents.

Compared to Ex2 ("s10+Ex2"), LINGUIST improves by +2.0 points absolute on Local Intent Recall (from 90.0 to 92.0), and +2.5 points absolute on Local ST F1 Score (from 79.8 to 82.3).

Finally, we show that the improvements in Local metrics for the new intent do not cause harm to the overall system, and in fact provide a small improvement. As shown in Table 15a and Table 15b (Appendix G.1), "s10+LINGUIST" improves upon "s10+Ex2" by +0.3 points absolute on both Global Intent Accuracy and Global Slot F1 Score.

5.1.3 LINGUIST Results on SNIPS (LNO)

We report on the Label Names Only (LNO) setting described in Section 4.2.2. For these results, we used LINGUIST models trained without label name dropout, which we found to perform significantly better (ablation shown in Appendix E).

As shown in Table 3 for IC and Table 4 for ST, despite having zero real examples for the new intents, LINGUIST achieves 80.0 on Local Intent Recall and 56.9 on Local ST F1 Score. (Global metrics are shown in Appendix G.2.) While this is still far behind using the real text and annotation from these 10 examples ("s10"), it represents significant progress towards true zero-shot development of new intents and slots in IC+ST systems.

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>s10</th>
<th>LINGUIST (via s10 LNO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>98.3 ± 2.1</td>
<td>68.6 ± 18.7</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>93.5 ± 2.4</td>
<td>92.1 ± 5.3</td>
</tr>
<tr>
<td>GetWeather</td>
<td>98.8 ± 0.4</td>
<td>99.6 ± 0.5</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>77.1 ± 6.6</td>
<td>85.4 ± 3.1</td>
</tr>
<tr>
<td>RateBook</td>
<td>99.6 ± 0.5</td>
<td>100.0 ± 0.0</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>76.9 ± 9.3</td>
<td>66.7 ± 5.8</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>73.1 ± 8.2</td>
<td>47.7 ± 9.0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>88.2 ± 1.9</strong></td>
<td><strong>80.0 ± 2.6</strong></td>
</tr>
</tbody>
</table>

Table 3: Local Intent Recall results on SNIPS in the New Intent Few-Shot Label Names Only (NIFS-LNO) setting. "s10" results are copied from Table 2a.

5.2 mATIS++ Results

Our mATIS++ results are shown in Tables 5 (Intent Accuracy), and 6 (Slot F1). The main focus is "avg-0S", the average zero-shot performance across the 6 non-en languages (de, es, fr, hi, ja, pt).

5.2.1 Baseline Results on mATIS++

An upper bound for zero-shot cross-lingual IC+ST is multilingual training, where a model is trained jointly on the real data for all languages, ("all") which achieves 97.17 for IC and 90.72 for ST. Reducing to English only data ("en") harms average zero-shot Intent by 5.0 points, and Slot F1 by 23.6 points. As our baseline, we report the numbers from the best cross-lingual system ("MT+soft-align") in (Xu et al., 2020), which uses a specialized transformer architecture for slot alignments, achieving 94.88 on IC, and 79.84 on ST.

5.2.2 Fine-tuning LINGUIST on MASSIVE

We first fine-tune a LINGUIST model on the MASSIVE dataset, following the process from Section 3.2. We formulate monolingual prompts for each of the 7 languages, and cross-lingual prompts from English to the other 6 languages, which is straightforward: for each training utterance in e.g. French, we select up to 10 English training examples that have the same intent, to include in the prompt with the French utterance as the target, setting “French” in the <language> block of the prompt.

To demonstrate not only cross-lingual and cross-schema (mATIS++ label names and annotations conventions are different from MASSIVE) but also cross-domain transfer of LINGUIST, we exclude the transport domain from MASSIVE, as it has some overlap with the travel information domain of mATIS++. We also exclude two other domains for validation early stopping (Appendix C).
5.2.3 LINGUIST Results on mATIS++

Then, for inference on mATIS++, we first create monolingual English prompts, then use a cloud-based MT system to translate the slot values into the target language, set the <language> tag in the prompt, and generate 10 annotated utterances. See Figure A5 (Appendix B.2) for an example.

We select the output with lowest perplexity, and use the English IC+ST model to verify its intent, discarding it if the prediction mismatches the intent from the prompt, in which case we simply copy over an English utterance from the same intent, to maintain the class distribution. (See Appendix F.)

The final dataset contains $N$ original English examples, and $N$ examples for each other language.

Compared to the “MT+soft-align” method of Xu et al. (2020), LINGUIST is on-par for IC (from 94.88 to 95.06), and improves ST F1 Score by 4.14 points absolute (from 79.84 to 83.98). We note that ST, being a structured prediction task, is inherently more challenging than IC, so our ST results are of particular interest. Moreover, the improvement on both IC and ST is particularly large for Japanese, which tends to be challenging for alignment with English, since the two languages having very different linguistic characteristics.

<table>
<thead>
<tr>
<th>Feature/Lang</th>
<th>de</th>
<th>es</th>
<th>fr</th>
<th>ja</th>
</tr>
</thead>
<tbody>
<tr>
<td>CameraControl</td>
<td>-33.3%</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ClockSettings</td>
<td>-1.6%</td>
<td>-12.3%</td>
<td>+1.8%</td>
<td>-</td>
</tr>
<tr>
<td>HomeSecurity</td>
<td>-</td>
<td>-</td>
<td>-31.5%</td>
<td>-</td>
</tr>
<tr>
<td>Music</td>
<td>-36.8%</td>
<td>-30.6%</td>
<td>-12.3%</td>
<td>-</td>
</tr>
<tr>
<td>Timers</td>
<td>-27.5%</td>
<td>-15.4%</td>
<td>-20.0%</td>
<td>-</td>
</tr>
<tr>
<td>Average</td>
<td>-11.8%</td>
<td>-20.8%</td>
<td>-7.9%</td>
<td>-25.2%</td>
</tr>
</tbody>
</table>

Table 7: Internal Dataset Results. Each number is relative reduction in SemER, from baseline (combined Slot Catalog Resampling, BT-Small, and ICLM) to LINGUIST. A negative number indicates improvement.

6 Conclusion and Future Work

We introduced LINGUIST, a novel method for annotated data generation, via fine-tuning a large-scale pre-trained multilingual seq2seq model. Our method generalizes to new intents and slots in challenging few-shot, zero-shot, and cross-lingual settings, which we have shown on three datasets.

In future work, we wish to explore ways to improve the generation output, e.g. human-in-the-loop and reinforcement learning. We would also like to include more controls in the prompt such as text style, and explore generation for multi-turn dialogues, and more complex and nested semantics.

Acknowledgements

We thank Christophe Dupuy, Weitong Ruan, and the anonymous reviewers for feedback on our work.
References


Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with
over 100 billion parameters. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, KDD ’20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.


A Author Contributions

Andy proposed and implemented LINGUIST; performed the experiments on SNIPS, MASSIVE, and mATIS++; fine-tuned LINGUIST for the internal dataset; and pre-trained the Alexa Teacher Model 2.3B encoder to warm-start AlexaTM 5B seq2seq.

Saleh implemented the seq2seq pre-training and fine-tuning code, pre-trained AlexaTM 5B, provided guidance on experiments and modeling choices for LINGUIST, and helped write the paper.

Wael drove the strategic vision for the project, provided deep technical feedback throughout, and implemented the backbone of our codebase.

Yannick and Markus assisted with the internal dataset preparation and evaluation, the baseline methods ICLM and BT-Small for SNIPS, and contributed to the paper write-up and technical review. Yannick also suggested MASSIVE as the dataset for transferring LINGUIST to mATIS++.

B Sample Model Outputs

B.1 English Outputs

This section uses a LINGUIST model from 6 SNIPS intents: AddToPlaylist, BookRestaurant, PlayMusic, RateBook, SearchCreativeWork, and SearchScreeningEvent.

B.1.1 One-shot Novel Ambiguous Intent

We show (Figure A1) the LINGUIST model generating annotated utterances for a new intent not seen in training data, given only a single example. Furthermore, when the example is ambiguous or under-specified, the model can leverage the intent label to generate more relevant outputs.

(a) The prompt with DownloadGame intent.

```plaintext
INPUT:
<language>English</language>
<intent>DownloadGame</intent>
<include>[1 zelda]</include>
<labels>[1=game]</labels>
<examples>get [1 zelda]</examples>

OUTPUTS:
1. I’d like to play [1 zelda]
2. I want to play [1 zelda]
3. I want to download [1 zelda]
4. I need to download [1 zelda]
5. Where can I download [1 zelda]?
```

(b) The prompt with PurchaseGame intent.

```plaintext
INPUT:
<language>English</language>
<intent>PurchaseGame</intent>
<include>[1 zelda]</include>
<labels>[1=game]</labels>
<examples>get [1 zelda]</examples>

OUTPUTS:
1. I’d like to buy the game [1 zelda]
2. I’d like to buy [1 zelda]
3. I want to buy [1 zelda]
4. Where can I buy [1 zelda]?
5. Please buy [1 zelda]
```

Figure A1: For the same example and slots, LINGUIST can leverage the intent name to generate different outputs.

B.1.2 One-shot Novel Ambiguous Slot

Similarly, (Figure A2) from a single example for a novel slot, LINGUIST can generate novel values, following the label name to disambiguate an ambiguous word such as “bat” (animal or sports equipment).

(a) The prompt with animal slot.

```plaintext
INPUT:
<language>English</language>
<intent>DisplayImage</intent>
<include>[1 •]</include>
<labels>[1=animal]</labels>
<examples>show me a picture of a [1 bat]</examples>

OUTPUTS:
1. I’d like to see a picture of a [1 dog]
2. Please show me a picture of a [1 deer].
3. Show me a photo of a [1 horse].
4. Show me a picture of a [1 zebra].
5. Show me a [1 elephant] picture.
```

(b) The prompt with sports_equipment slot.

```plaintext
INPUT:
<language>English</language>
<intent>DisplayImage</intent>
<include>[1 •]</include>
<labels>[1=sports_equipment]</labels>
<examples>show me a picture of a [1 bat]</examples>

OUTPUTS:
1. I’d like to see a picture of a [1 volleyball]
2. Show me a photo of a [1 soccer ball]
3. Show me a [1 football net]
4. show me a [1 puck]
5. Show me a [1 golf club]
```

Figure A2: For the same example and slots, LINGUIST can leverage the slot name to generate different outputs.
B.1.3 Zero-shot Novel Intent and Slot

We show (Figure A3) that LINGUIST can also generate in a zero-shot setting, prompted only on the intent and slot names, without any full text examples. By changing the slot type from `baseball_team_name` to `hockey_team_name`, the model can generate for the requested sports team. Remarkably, although the model never saw utterances about sports during fine-tuning, it can accurately sample correct team names, presumably relying on the world knowledge it acquired during pre-training. In the outputs, all baseball teams are actual MLB, and the hockey team names are NHL, with one exception of the Wildcats being basketball. The slot `year` appeared in the SNIPS fine-tuning data, so the model may be using values either from world knowledge or from other SNIPS intents.


(a) The prompt with `baseball_team_name` intent.


(b) The prompt with `hockey_team_name` intent.

Figure A3: LINGUIST can perform zero-shot generation based only on the intent and slot names, with no full text example. It can also leverage world knowledge from pre-training to generate novel slot values that incorporate world knowledge such as valid team names for professional sports.

B.1.4 Label Names Only

Finally, as shown in Figure A4, the most ambiguous generation scenario we explore is “Label Names Only” (introduced in Section 4.2.2), where the model sees no examples, and all slot values are open-ended via the wildcard instruction * . Remarkably, LINGUIST can still generate useful outputs in this case, although with some more noise: for example, the text of Output 2 does not correspond to GetWeather intent, and the word `time` is mistakenly tagged as a slot value `timeRange`. Nevertheless, as shown in Section 5.1.3, training with LINGUIST-generated data in this setting can achieve reasonable accuracy, representing significant progress towards true zero-shot data generation for novel intents and slots in IC+ST systems.

B.2 Cross-Lingual Novel Intent and Slots

In Figure A5 we show an example of LINGUIST performing few-shot cross-lingual annotated data generation on a novel domain, where it sees only 7 examples in English, and can generate diverse, fluent, and correctly annotated outputs in French.

The model here was trained on MASSIVE both monolingual and cross-lingual prompts, as described in Section 5.2.2. We discarded three of the 18 MASSIVE domains during training, namely audio and cooking to use as validation sets, and transport, in order to keep mATIS++, which covers travel information, as a novel domain.
Figure A4: LINGUIST generation in Label Names Only (LNO) setting, where there are no examples or slot values, and the model must rely entirely on the label names to generate outputs.

Figure A5: LINGUIST generating on a Cross-lingual English to French example from mATIS++.
C LINGUIST Training Details

C.1 Hyperparameters and Early Stopping

As shown in Figure A6, we find that the model converges after a very small number of updates. Specifically, we train with batch size 512 for 400 updates (i.e. around 18 epochs for SNIPS, around 2.5 epochs for MASSIVE), using a very small learning rate $5e^{-7}$ with Adam (Kingma and Ba, 2015), warmed up over the first 100 updates, then kept constant for the rest of training. (The internal dataset is much larger, so for that, we train for 4k updates instead, and use a larger learning rate of $1e^{-6}$.)

For MASSIVE, we removed two additional small domains audio and cooking from training and early stop once Token Accuracy plateaus on these domains, which occurs around 400 updates. The Token Accuracy is the percentage of subword tokens for which the model’s top-1 hypothesis matches the ground truth (higher is better). It requires only a single forward pass to compute, without needing auto-regressive decoding. In early experiments, we found Token Accuracy to be more reliable than perplexity at predicting downstream performance.

As shown in Figure A6, the Token Accuracy continues to improve for the domains seen during training, however plateaus after 400 updates for the two novel domains, suggesting overfitting beyond 400 updates. The token accuracy is similar for same-language (left) and cross-lingual (right) prompts, suggesting that the model can jointly learn both tasks to a similar level of performance.

For the SNIPS runs, the data is so limited to only 6 intents per run, so removing another intent to check for early stopping could harm the model performance. Therefore, we simply use 400 updates again, as that worked for MASSIVE.

We use DeepSpeed (Rasley et al., 2020) ZeRO Stage 2 to accelerate training.

![Figure A6: Validation Accuracy across updates of fine-tuning LINGUIST on the MASSIVE dataset.](image)

(a) Validation Token Accuracy on same-language prompts. (b) Validation Token Accuracy on cross-lingual prompts.

C.2 Tokenizer Choices

As mentioned in Section 3.2, we keep the original sentencepiece (Kudo and Richardson, 2018) tokenizer of the model; we do not explicitly add vocabulary items for $<\text{intent}>$, [1, etc. This choice is motivated by two intuitions:

1. We hypothesize that these tags’ resemblance to markup languages like HTML/XML may help the model learn that they are instructions rather than content words, by relying on data seen during pre-training which was formatted similarly. An earlier version of the LINGUIST prompt had only the open tags such as $<\text{intent PlayMusic}>$, and we found that the pre-trained model produced matching closing tags such as $</\text{intent}>$ before any fine-tuning, suggesting it had learned some knowledge of the tag structure from pre-training.
(2) We hypothesize that the model may be able to generalize at inference time to utterances with a larger number of slots than seen during fine-tuning, by tokenizing the numbered brackets. For example, the largest numbered bracket seen when fine-tuning on MASSIVE is \([10\text{, tokenized as } '_[', '10']\). For inference on mATIS++, 47% of the prompts contain numbered brackets between \([11\text{ and } 24\text{. If we had added vocabulary items for these, they would be stuck as randomly initialized tokens at inference time, and therefore unlikely to produce high quality generation. Instead, we see in the generated outputs many examples where the model handles these larger numbered brackets without a problem.}

D Impact of Model Size

To evaluate the impact of the model size used for LINGUIST data generation, we evaluate on mATIS++ with a 10x smaller model, following the same procedure of first fine-tuning on MASSIVE (Section 5.2.2), then running cross-lingual inference on mATIS++ (Section 5.2.3).

We use **AlexaTM-Large 500M**, which is trained using the same data as AlexaTM 20B (Soltan et al., 2022) and AlexaTM 5B (Section 4.3.1). Like AlexaTM 5B, AlexaTM-Large 500M uses only the denoising objective (no Causal Language Modeling). The architecture contains 12 encoder and 12 decoder layers, with hidden size 1024, the same as (m)BART (Lewis et al., 2020b; Liu et al., 2020).

As show in Tables 8 for IC and 9 for ST, switching to the smaller model loses 1.62 points on IC (from 95.06 to 93.44), and 1.74 points on ST (from 83.98 to 82.24). While LINGUIST with the smaller model under-performs the baseline “en+MT soft-align” on IC by 1.44 points (93.44 compared to 94.88), it still out-performs on ST by 2.40 points (82.24 compared to 79.84), showing the value of the LINGUIST on the more challenging ST task, even with a smaller model.

<table>
<thead>
<tr>
<th>Lang</th>
<th>all</th>
<th>en</th>
<th>en+MT soft-align</th>
<th>en+LINGUIST</th>
<th>en+LINGUIST (AlexaTM-Large 500M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>98.10</td>
<td>97.77</td>
<td>-</td>
<td>97.77</td>
<td>97.38</td>
</tr>
<tr>
<td>de</td>
<td>97.32</td>
<td>90.51</td>
<td>96.66</td>
<td>94.08</td>
<td>95.65</td>
</tr>
<tr>
<td>es</td>
<td>97.21</td>
<td>95.20</td>
<td>97.72</td>
<td>97.10</td>
<td>95.87</td>
</tr>
<tr>
<td>fr</td>
<td>98.10</td>
<td>93.64</td>
<td>97.49</td>
<td>96.88</td>
<td>95.98</td>
</tr>
<tr>
<td>hi</td>
<td>95.20</td>
<td>88.62</td>
<td>92.81</td>
<td>94.08</td>
<td>86.94</td>
</tr>
<tr>
<td>ja</td>
<td>97.86</td>
<td>90.99</td>
<td>88.33</td>
<td>95.38</td>
<td>91.55</td>
</tr>
<tr>
<td>pt</td>
<td>97.32</td>
<td>93.97</td>
<td>96.78</td>
<td>92.86</td>
<td>94.64</td>
</tr>
<tr>
<td>avg-OS</td>
<td>97.17</td>
<td>92.16</td>
<td>94.88</td>
<td><strong>95.06</strong></td>
<td>93.44</td>
</tr>
</tbody>
</table>

Table 8: Results on mATIS++ Intent Accuracy, showing impact of model size. All except “en+LINGUIST (AlexaTM-Large 500M)” are copied from Table 5.

<table>
<thead>
<tr>
<th>Lang</th>
<th>all</th>
<th>en</th>
<th>en+MT soft-align</th>
<th>en+LINGUIST</th>
<th>en+LINGUIST (AlexaTM-Large 500M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>95.26</td>
<td>95.96</td>
<td>-</td>
<td>95.07</td>
<td>95.51</td>
</tr>
<tr>
<td>de</td>
<td>94.54</td>
<td>80.15</td>
<td>89</td>
<td>84.61</td>
<td>83.02</td>
</tr>
<tr>
<td>es</td>
<td>88.27</td>
<td>81.24</td>
<td>76.42</td>
<td>86.89</td>
<td>85.45</td>
</tr>
<tr>
<td>fr</td>
<td>92.69</td>
<td>77.29</td>
<td>79.64</td>
<td>83.83</td>
<td>82.19</td>
</tr>
<tr>
<td>hi</td>
<td>85.58</td>
<td>62.61</td>
<td>78.56</td>
<td>76.61</td>
<td>75.36</td>
</tr>
<tr>
<td>ja</td>
<td>92.76</td>
<td>24.52</td>
<td>79.1</td>
<td>86.32</td>
<td>83.23</td>
</tr>
<tr>
<td>pt</td>
<td>90.49</td>
<td>76.64</td>
<td>76.3</td>
<td>85.63</td>
<td>84.18</td>
</tr>
<tr>
<td>avg-OS</td>
<td>90.72</td>
<td>67.08</td>
<td>79.84</td>
<td><strong>83.98</strong></td>
<td>82.24</td>
</tr>
</tbody>
</table>

Table 9: Results on mATIS++ Slot F1, showing impact of model size. All except “en+LINGUIST (AlexaTM-Large 500M)” are copied from Table 6.

E Ablation Study on Label Name Dropout

In early experiments, we found Label Name Dropout (LNDrop, described in Section 3.2) helped reduce the model’s tendency to overfit on the label names seen during training. However, at the end of our
We present an ablation study on filtering the outputs of LINGUIST for mATIS++, with results presented in Table 13 for Intent Accuracy, and Table 14 for Slot F1.

### Table 11: Local ST F1 Score results on SNIPS comparing with and without Label Name Dropout (LNDrop), in settings NIFS vanilla (left two columns) and NIFS-LNO (Label Names Only, Section 4.2.2) (right two columns).

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>s10 +LINGUIST LNDrop: yes</th>
<th>s10 +LINGUIST LNDrop: no</th>
<th>LINGUIST (via s10 LNO) LNDrop: yes</th>
<th>LINGUIST (via s10 LNO) LNDrop: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>93.9 ± 3.2</td>
<td>92.3 ± 7.8</td>
<td>20.0 ± 11.6</td>
<td>68.6 ± 18.7</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>94.6 ± 1.5</td>
<td>93.8 ± 2.3</td>
<td>86.5 ± 4.9</td>
<td>92.1 ± 5.3</td>
</tr>
<tr>
<td>GetWeather</td>
<td>100.0 ± 0.0</td>
<td>99.6 ± 0.5</td>
<td>99.2 ± 0.8</td>
<td>99.6 ± 0.5</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>90.4 ± 4.7</td>
<td>90.0 ± 3.1</td>
<td>76.2 ± 6.0</td>
<td>85.4 ± 3.1</td>
</tr>
<tr>
<td>RateBook</td>
<td>100.0 ± 0.0</td>
<td>99.8 ± 0.4</td>
<td>99.6 ± 0.5</td>
<td>100.0 ± 0.0</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>83.3 ± 6.9</td>
<td>92.5 ± 4.0</td>
<td>66.1 ± 13.4</td>
<td>66.7 ± 5.8</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>81.9 ± 3.9</td>
<td>79.4 ± 2.8</td>
<td>44.2 ± 6.7</td>
<td>47.7 ± 9.0</td>
</tr>
<tr>
<td>Average</td>
<td>92.0 ± 0.8</td>
<td><strong>92.5 ± 1.5</strong></td>
<td>70.3 ± 2.6</td>
<td><strong>80.0 ± 2.6</strong></td>
</tr>
</tbody>
</table>

Table 10: Local Intent Recall results on SNIPS comparing with and without Label Name Dropout (LNDrop), in settings NIFS vanilla (left two columns) and NIFS-LNO (Label Names Only, Section 4.2.2) (right two columns). “s10+LINGUIST (LNDrop: yes)” results are copied from Table 2a and “LINGUIST (via s10 LNO) LNDrop: no” results are copied from Table 3.

### Table 11: Local ST F1 Score results on SNIPS comparing with and without Label Name Dropout (LNDrop), in settings NIFS vanilla (left two columns) and NIFS-LNO (Label Names Only, Section 4.2.2) (right two columns).

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>s10 +LINGUIST LNDrop: yes</th>
<th>s10 +LINGUIST LNDrop: no</th>
<th>LINGUIST (via s10 LNO) LNDrop: yes</th>
<th>LINGUIST (via s10 LNO) LNDrop: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>80.9 ± 3.4</td>
<td>80.4 ± 2.1</td>
<td>56.6 ± 8.0</td>
<td>45.9 ± 9.1</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>83.4 ± 1.7</td>
<td>82.8 ± 4.0</td>
<td>70.7 ± 3.5</td>
<td>74.8 ± 3.5</td>
</tr>
<tr>
<td>GetWeather</td>
<td>85.4 ± 2.8</td>
<td>83.1 ± 3.6</td>
<td>70.0 ± 3.4</td>
<td>71.2 ± 3.8</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>70.1 ± 1.8</td>
<td>69.3 ± 2.5</td>
<td>54.2 ± 3.2</td>
<td>55.6 ± 3.1</td>
</tr>
<tr>
<td>RateBook</td>
<td>94.8 ± 1.7</td>
<td>96.1 ± 1.0</td>
<td>51.0 ± 11.1</td>
<td>55.0 ± 6.3</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>79.3 ± 5.0</td>
<td>85.3 ± 3.7</td>
<td>55.5 ± 11.2</td>
<td>59.3 ± 5.7</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>82.3 ± 3.4</td>
<td>80.4 ± 1.4</td>
<td>29.7 ± 2.9</td>
<td>36.6 ± 6.1</td>
</tr>
<tr>
<td>Average</td>
<td>82.3 ± 1.3</td>
<td><strong>82.5 ± 1.7</strong></td>
<td>55.4 ± 2.3</td>
<td><strong>56.9 ± 2.5</strong></td>
</tr>
</tbody>
</table>

Table 10: Local Intent Recall results on SNIPS comparing with and without Label Name Dropout (LNDrop), in settings NIFS vanilla (left two columns) and NIFS-LNO (Label Names Only, Section 4.2.2) (right two columns). “s10+LINGUIST (LNDrop: yes)” results are copied from Table 2a and “LINGUIST (via s10 LNO) LNDrop: no” results are copied from Table 3.

### Ablation Study on Filtering Generated Annotated Utterances for mATIS++

We present an ablation study on filtering the outputs of LINGUIST for mATIS++, with results presented in Table 13 for Intent Accuracy, and Table 14 for Slot F1.

#### F.1 Filtering Methods

As introduced in Section 5.2.3, for each annotated English utterance in mATIS++, we generate 10 annotated utterances in each of the zero-shot languages (German, Spanish, French, Hindi, Japanese, Portuguese). Then, for each language, we select the single\(^1\) generated annotated utterance with lowest perplexity which also passes Valid-Filter, described next.

\(^1\)In future work, we would like to evaluate the impact on mATIS++ of including more than one output per input prompt, as we have done for SNIPS and for the Internal Dataset.
We apply two filtering methods, (1) Valid-Filter, and (2) English-IC-Filter, and report the “Pass Rate” of each in Table 12, as the portion of utterances that pass the filter.

For Valid-Filter we discard utterances that have invalid brackets like `[2 [ ] ]`, or do not respect the prompt, by either generating too few or too many slots, or not copying the value when requested. The Pass Rate for Valid-Filter is 71.5%, averaged across the languages.

For English-IC-Filter, we classify the intent of the generated utterance’s text, using the English-only IC+ST model, and discard the utterance if the predicted intent disagrees with the intent from the LINGUIST prompt. We note that the English-only IC+ST model (“en” in Table 13) already performs quite well on IC for other languages, achieving 92.16 Intent Accuracy, so we expect it to contain a strong signal to filter out noisy generated utterances. The Pass Rate for English-IC-Filter is 83.8%, suggesting that the remaining 16.2% of the utterances are likely to correspond to an intent other than what was requested in the prompt, which we discuss further below (Section F.3).

After cascading the two filters, the overall Pass Rate is 59.9%.

As a final step, observing that some intents may have lost more data than others, we apply a simple fix which we call Balance-Classes to recover the original per-intent class distribution: we simply copy over English utterances from the intents that lack enough data.

<table>
<thead>
<tr>
<th>Lang</th>
<th>Valid-Filter Pass Rate</th>
<th>English-IC-Filter Pass Rate</th>
<th>Cascaded Pass Rate</th>
<th>Num Outputs from LINGUIST</th>
<th>Num Copied from English</th>
<th>Total Num Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>de</td>
<td>73.8</td>
<td>77.1</td>
<td>56.9</td>
<td>2393</td>
<td>1816</td>
<td>4209</td>
</tr>
<tr>
<td>es</td>
<td>77.2</td>
<td>88.2</td>
<td>68.1</td>
<td>2867</td>
<td>1342</td>
<td>4209</td>
</tr>
<tr>
<td>fr</td>
<td>77.3</td>
<td>80.2</td>
<td>62.0</td>
<td>2609</td>
<td>1600</td>
<td>4209</td>
</tr>
<tr>
<td>hi</td>
<td>75.4</td>
<td>85.6</td>
<td>64.6</td>
<td>2717</td>
<td>1492</td>
<td>4209</td>
</tr>
<tr>
<td>ja</td>
<td>50.5</td>
<td>84.8</td>
<td>42.8</td>
<td>1801</td>
<td>2408</td>
<td>4209</td>
</tr>
<tr>
<td>pt</td>
<td>74.7</td>
<td>87.2</td>
<td>65.2</td>
<td>2744</td>
<td>1465</td>
<td>4209</td>
</tr>
<tr>
<td>avg</td>
<td>71.5</td>
<td>83.8</td>
<td>59.9</td>
<td>2522</td>
<td>1687</td>
<td>4209</td>
</tr>
</tbody>
</table>

Table 12: Pass Rate of LINGUIST generated utterances for mATIS++

F.2 Impact of Filtering
The impact of filtering is presented in Table 13 for Intent Accuracy, and Table 14 for Slot F1, where our main result is “avg-0S”, the average of the zero-shot languages (de, es, fr, hi, ja, pt). We observe that English-IC-Filter improves IC by 0.89 points absolute (from 93.09 to 93.98) and Balance-Classes improves IC by a further 1.08 points absolute (from 93.98 to 95.06), with both methods having minimal impact on Slot F1.

<table>
<thead>
<tr>
<th>Lang</th>
<th>all</th>
<th>en</th>
<th>en+MT soft-align</th>
<th>en+ LINGUIST (NoFilter)</th>
<th>en+ LINGUIST (+English-IC-Filter)</th>
<th>en+ LINGUIST (+English-IC-Filter) (+Balance-Classes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>98.10</td>
<td>97.77</td>
<td>–</td>
<td>97.21</td>
<td>97.77</td>
<td>97.77</td>
</tr>
<tr>
<td>de</td>
<td>97.32</td>
<td>90.51</td>
<td>96.66</td>
<td>94.31</td>
<td>93.64</td>
<td>94.08</td>
</tr>
<tr>
<td>es</td>
<td>97.21</td>
<td>95.20</td>
<td>97.2</td>
<td>94.75</td>
<td>96.88</td>
<td>97.10</td>
</tr>
<tr>
<td>fr</td>
<td>98.10</td>
<td>93.64</td>
<td>97.49</td>
<td>93.97</td>
<td>96.43</td>
<td>96.88</td>
</tr>
<tr>
<td>hi</td>
<td>95.20</td>
<td>88.62</td>
<td>92.81</td>
<td>98.17</td>
<td>92.19</td>
<td>94.08</td>
</tr>
<tr>
<td>ja</td>
<td>97.86</td>
<td>90.99</td>
<td>88.33</td>
<td>91.78</td>
<td>91.44</td>
<td>95.38</td>
</tr>
<tr>
<td>pt</td>
<td>97.32</td>
<td>93.97</td>
<td>96.78</td>
<td>95.54</td>
<td>93.30</td>
<td>92.86</td>
</tr>
<tr>
<td>avg-0S</td>
<td>97.17</td>
<td>92.16</td>
<td>94.88</td>
<td>93.09</td>
<td>93.98</td>
<td>95.06</td>
</tr>
</tbody>
</table>

Table 13: Intent Accuracy results on ablation study of Filtering for mATIS++.

F.3 Intent Mismatch Discussion
We discuss an intuition about why LINGUIST in a cross-domain setting such as MASSIVE to mATIS++ might produce outputs that do not exactly match the prompted intent. Notice that the prompt contains only 10 examples of the target intent, and no examples of other intents from the new domain. For example,
mATIS++ contains several closely related intents, such as “flight” which asks to list and book flights, “flight_time” which asks about the time of a flight, and “airfare” which asks about the price of a flight. We find that when the prompts contain only “flight” examples, the model tends to over-generalize and produce some requests asking for time or price instead, which harms IC, necessitating a post-processing method such as English-IC-Filter. In future work, we plan to explore methods to incorporate few-shot data from other intents in the domain while generating for the target intent, to mitigate this problem.

## G SNIPS Results on Global Metrics

We report the results on SNIPS Global Metrics as mentioned in Section 5.1.

### G.1 SNIPS Results on Global Metrics: NIFS

Our main results are on the New-Intent Few-Shot setting (NIFS, described in Section 4.2.1). Table 15a shows Global Intent Accuracy, and Table 15b shows Global ST F1 Score. These correspond to the Local metrics shown in Tables 2a for Local IC and 2b for Local ST.

As all our methods target a new-intent setting, as expected, they do not substantially impact the Global metrics. Nonetheless, LINGUIST does provide a small improvement of +0.3 points absolute on both IC and ST compared to Ex2.

### Table 14: Slot F1 results on ablation study of Filtering for mATIS++.

<table>
<thead>
<tr>
<th>Lang</th>
<th>all</th>
<th>en</th>
<th>en+MT soft-align</th>
<th>en+LINGUIST (NoFilter)</th>
<th>en+LINGUIST (+English-IC-Filter)</th>
<th>en+LINGUIST (+English-IC-Filter) (+Balance-Classes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>95.26</td>
<td>95.96</td>
<td>-</td>
<td>94.16</td>
<td>95.01</td>
<td>95.07</td>
</tr>
<tr>
<td>de</td>
<td>94.34</td>
<td>80.15</td>
<td>89</td>
<td>83.40</td>
<td>85.26</td>
<td>84.61</td>
</tr>
<tr>
<td>es</td>
<td>88.27</td>
<td>81.24</td>
<td>76.42</td>
<td>85.92</td>
<td>85.47</td>
<td>86.89</td>
</tr>
<tr>
<td>fr</td>
<td>92.69</td>
<td>77.29</td>
<td>79.64</td>
<td>84.65</td>
<td>84.71</td>
<td>83.83</td>
</tr>
<tr>
<td>hi</td>
<td>85.58</td>
<td>62.61</td>
<td>78.56</td>
<td>78.35</td>
<td>75.89</td>
<td>76.61</td>
</tr>
<tr>
<td>ja</td>
<td>92.76</td>
<td>24.52</td>
<td>79.1</td>
<td>85.72</td>
<td>85.38</td>
<td>86.32</td>
</tr>
<tr>
<td>pt</td>
<td>90.49</td>
<td>76.64</td>
<td>76.3</td>
<td>84.45</td>
<td>85.06</td>
<td>85.63</td>
</tr>
<tr>
<td>avg-OS</td>
<td>90.72</td>
<td>67.08</td>
<td>79.84</td>
<td>83.75</td>
<td>83.63</td>
<td>83.98</td>
</tr>
</tbody>
</table>

### Table 15: Our results on SNIPS for the Global metrics, showing that the gains for Local metrics shown in Tables 2a and 2b do not cause harm to the system overall. See Section 5.1 for details.

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>Full</th>
<th>s10-NoUps</th>
<th>s10</th>
<th>s10 +ICLM</th>
<th>s10 +BT-Small</th>
<th>s10 +BT-5B</th>
<th>s10 +Ex2</th>
<th>s10 +LINGUIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>99.1</td>
<td>98.7 ±0.1</td>
<td>98.9 ±0.3</td>
<td>98.9 ±0.3</td>
<td>98.9 ±0.3</td>
<td>99.1 ±0.1</td>
<td>99.0 ±0.3</td>
<td>98.4 ±0.6</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>99.1</td>
<td>97.7 ±0.4</td>
<td>98.2 ±0.3</td>
<td>98.2 ±0.2</td>
<td>97.9 ±0.2</td>
<td>98.2 ±0.1</td>
<td>97.9 ±0.8</td>
<td>98.2 ±0.3</td>
</tr>
<tr>
<td>GetWeather</td>
<td>99.1</td>
<td>98.9 ±0.1</td>
<td>98.9 ±0.1</td>
<td>98.9 ±0.2</td>
<td>99.0 ±0.0</td>
<td>99.0 ±0.1</td>
<td>99.1 ±0.2</td>
<td>99.1 ±0.1</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>99.1</td>
<td>95.3 ±1.5</td>
<td>96.1 ±1.0</td>
<td>96.4 ±1.3</td>
<td>96.0 ±0.8</td>
<td>97.0 ±0.4</td>
<td>97.1 ±1.1</td>
<td>97.9 ±0.6</td>
</tr>
<tr>
<td>RateBook</td>
<td>99.1</td>
<td>99.0 ±0.1</td>
<td>98.9 ±0.1</td>
<td>99.0 ±0.1</td>
<td>99.0 ±0.1</td>
<td>99.0 ±0.0</td>
<td>99.0 ±0.0</td>
<td>99.0 ±0.0</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>99.1</td>
<td>95.2 ±1.1</td>
<td>96.5 ±1.2</td>
<td>96.0 ±1.2</td>
<td>95.8 ±1.8</td>
<td>96.7 ±1.4</td>
<td>96.7 ±0.8</td>
<td>97.3 ±1.0</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>99.1</td>
<td>95.3 ±0.8</td>
<td>96.3 ±1.1</td>
<td>95.9 ±0.5</td>
<td>95.8 ±0.8</td>
<td>96.2 ±1.5</td>
<td>96.8 ±1.2</td>
<td>97.4 ±0.5</td>
</tr>
<tr>
<td>Average</td>
<td>99.1</td>
<td>97.2 ±0.4</td>
<td>97.7 ±0.2</td>
<td>97.6 ±0.4</td>
<td>97.5 ±0.3</td>
<td>97.9 ±0.2</td>
<td>97.9 ±0.3</td>
<td>98.2 ±0.1</td>
</tr>
</tbody>
</table>

(a) SNIPS New-Intent few-shot results on Global Intent Accuracy.

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>Full</th>
<th>s10-NoUps</th>
<th>s10</th>
<th>s10 +ICLM</th>
<th>s10 +BT-Small</th>
<th>s10 +BT-5B</th>
<th>s10 +Ex2</th>
<th>s10 +LINGUIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>96.7</td>
<td>94.0 ±0.4</td>
<td>94.2 ±0.5</td>
<td>94.2 ±0.4</td>
<td>94.9 ±0.4</td>
<td>94.5 ±0.3</td>
<td>94.7 ±0.4</td>
<td>94.6 ±0.6</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>96.7</td>
<td>92.7 ±0.4</td>
<td>94.1 ±0.6</td>
<td>94.1 ±0.3</td>
<td>94.2 ±0.3</td>
<td>94.5 ±0.3</td>
<td>93.9 ±1.0</td>
<td>94.6 ±0.4</td>
</tr>
<tr>
<td>GetWeather</td>
<td>96.7</td>
<td>93.8 ±0.5</td>
<td>94.9 ±0.7</td>
<td>94.7 ±0.6</td>
<td>94.6 ±0.4</td>
<td>95.0 ±0.3</td>
<td>94.7 ±0.7</td>
<td>95.0 ±0.4</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>96.7</td>
<td>91.3 ±0.4</td>
<td>92.9 ±0.3</td>
<td>93.1 ±0.6</td>
<td>92.8 ±0.2</td>
<td>93.8 ±0.4</td>
<td>93.9 ±0.6</td>
<td>94.2 ±0.2</td>
</tr>
<tr>
<td>RateBook</td>
<td>96.7</td>
<td>95.0 ±0.3</td>
<td>95.8 ±0.4</td>
<td>95.8 ±0.3</td>
<td>95.5 ±0.6</td>
<td>95.5 ±0.4</td>
<td>96.0 ±0.2</td>
<td>96.1 ±0.4</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>96.7</td>
<td>92.9 ±1.0</td>
<td>94.2 ±0.9</td>
<td>93.9 ±1.0</td>
<td>94.2 ±1.0</td>
<td>94.1 ±1.1</td>
<td>94.6 ±0.8</td>
<td>95.0 ±0.5</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>96.7</td>
<td>92.3 ±0.8</td>
<td>94.0 ±0.6</td>
<td>94.0 ±0.6</td>
<td>93.5 ±0.6</td>
<td>94.2 ±0.4</td>
<td>94.8 ±0.7</td>
<td>95.3 ±0.3</td>
</tr>
<tr>
<td>Average</td>
<td>96.7</td>
<td>93.1 ±0.2</td>
<td>94.3 ±0.2</td>
<td>94.2 ±0.2</td>
<td>94.3 ±0.3</td>
<td>94.5 ±0.3</td>
<td>94.7 ±0.3</td>
<td>95.0 ±0.2</td>
</tr>
</tbody>
</table>

(b) SNIPS New-Intent few-shot results on Global ST F1 Score.
G.2 SNIPS Results on Global Metrics: NIFS Label Names Only (LNO)

We show Global metrics for SNIPS in the NIFS-LNO setting (New-Intent Few-Shot Label Names Only, Section 4.2.2) in Tables 16a and 16b for Intent Accuracy and Slot F1 Score, respectively. These correspond to the Local metrics shown in Tables 3 for Local IC and 4 for Local ST. The numbers for “LINGUIST (via s10 LNO)” are only a few points behind “s10”, indicating that even when no real data is available for the novel intent, LINGUIST generated data can provide some support for the new intent, bringing the overall system performance close to where it would be with 10 real examples for that new intent.

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>s10</th>
<th>LINGUIST (via s10 LNO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>98.9 ± 0.3</td>
<td>94.5 ± 2.8</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>98.2 ± 0.3</td>
<td>98.0 ± 0.7</td>
</tr>
<tr>
<td>GetWeather</td>
<td>98.9 ± 0.1</td>
<td>99.1 ± 0.1</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>96.1 ± 1.0</td>
<td>97.3 ± 0.5</td>
</tr>
<tr>
<td>RateBook</td>
<td>98.9 ± 0.1</td>
<td>98.9 ± 0.1</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>96.5 ± 1.2</td>
<td>94.8 ± 0.8</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>96.3 ± 1.1</td>
<td>92.6 ± 1.3</td>
</tr>
<tr>
<td>Average</td>
<td>97.7 ± 0.2</td>
<td>96.5 ± 0.4</td>
</tr>
</tbody>
</table>

(a) SNIPS NIFS-LNO results on Global Intent Accuracy.

<table>
<thead>
<tr>
<th>Modified Intent / Data</th>
<th>s10</th>
<th>LINGUIST (via s10 LNO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>94.2 ± 0.5</td>
<td>88.8 ± 1.9</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>94.1 ± 0.6</td>
<td>93.1 ± 0.9</td>
</tr>
<tr>
<td>GetWeather</td>
<td>94.9 ± 0.7</td>
<td>93.1 ± 0.6</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>92.9 ± 0.3</td>
<td>92.6 ± 0.5</td>
</tr>
<tr>
<td>RateBook</td>
<td>95.8 ± 0.4</td>
<td>88.1 ± 1.1</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>94.2 ± 0.9</td>
<td>93.1 ± 0.5</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>94.0 ± 0.6</td>
<td>90.0 ± 0.8</td>
</tr>
<tr>
<td>Average</td>
<td>94.3 ± 0.2</td>
<td>91.2 ± 0.4</td>
</tr>
</tbody>
</table>

(b) SNIPS NIFS-LNO results on Global Slot F1 Score.

Table 16: Global metrics (Intent Accuracy, (a), left; Slot F1 Score, (b), right) for SNIPS in the NIFS-LNO setting (New-Intent Few-Shot Label Names Only, Section 4.2.2). The numbers for “s10” are copied from Tables 15a and 15b, for IC and ST, respectively.

H Intent Bleeding Case Study

As described in Ex2 (Lee et al., 2021), we also observed intent “bleeding”, where the model would produce outputs like one of the fine-tuning intents, despite the prompt and examples being from a novel intent. We noticed this particularly strongly when generating for AddToPlaylist intent on SNIPS, where the model had a strong tendency to return utterances starting with “play”, which overlaps with the closely related PlayMusicIntent from fine-tuning. Consequently, this harmed IC results (Table 2a) for this intent. A very simple fix is to use “n-gram blocking”, where the model is prevented from generating phrases like “play”, “I want to play”, etc. during generation. We found that this mitigates the issue, and we can get near 100% on Intent Recall for AddToPlaylist. However custom designing which n-grams to block requires effort from human experts, and does not scale to a large number of new intents and languages, so in future work, we would like to explore more automated and scalable solutions.

I Generation Hyperparameters

For all three datasets, we use top-k sampling (Fan et al., 2018). For SNIPS, we use top_k=50, temperature=0.3, and produce 100 outputs per input. For mATIS++, we use top_k=50, temperature=0.3, and produce 10 outputs per input. For the internal dataset, we top_k=20, temperature=1.0, and produce 20 outputs per input. We observe that when LINGUIST is trained on the much larger internal dataset compared to SNIPS, it produces less noisy outputs. Thus, we allow a higher temperature of 1.0. We use the same settings for all intents. We also benchmarked beam search and nucleus sampling (Holtzman et al., 2019) for generation, and found both to perform worse overall on the internal datasets and on SNIPS compared to top-k sampling.

J Filtering ICLM Outputs

We discard any outputs containing the <unk> token, which happens less than 1% of the time. The number of outputs (after de-duplication) are reported in Table 17.

K Filtering BT-Small Outputs

The small model has a fair amount of noise in its outputs, so we heuristically filter them, discarding any which contain repeated bigrams such as play the song halo the song and/or any trigram of
Table 17: The number of filtered and de-duplicated outputs from ICLM per intent. All numbers are averaged across the five random seeds.

<table>
<thead>
<tr>
<th>Modified Intent</th>
<th>Num outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>296</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>347</td>
</tr>
<tr>
<td>GetWeather</td>
<td>322</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>255</td>
</tr>
<tr>
<td>RateBook</td>
<td>288</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>295</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>273</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>297</strong></td>
</tr>
</tbody>
</table>

the same word such as of of of. Success rate and number of outputs (after de-duplication) are reported in Table 18.

<table>
<thead>
<tr>
<th>Modified Intent</th>
<th>SuccessRate</th>
<th>NumOutputs</th>
<th>AvgNumSlots</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>70.2</td>
<td>64</td>
<td>2.7</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>72.8</td>
<td>73</td>
<td>3.2</td>
</tr>
<tr>
<td>GetWeather</td>
<td>60.4</td>
<td>60</td>
<td>2.3</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>53.6</td>
<td>52</td>
<td>2.2</td>
</tr>
<tr>
<td>RateBook</td>
<td>70.8</td>
<td>71</td>
<td>3.8</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>41.6</td>
<td>42</td>
<td>1.8</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>69.6</td>
<td>70</td>
<td>2.2</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>62.7</strong></td>
<td><strong>62</strong></td>
<td><strong>2.6</strong></td>
</tr>
</tbody>
</table>

Table 18: For each intent, the Success Rate of Back-Translation with the Small model, and Number of Generated Outputs, both averaged across the five random seeds. For reference, we also show the Average Number of Slots in the training data per intent.

L Filtering BT-5B outputs

The Back-Translated text with the 5B model is significantly cleaner than with the smaller model, so we do not apply any filtering on the output text itself. We do heuristically discard any outputs where we suspect the augmented utterance is missing a slot. Specifically, SimAlign in ArgMax mode only returns alignments across words that have mutual argmax between source and target. For any source word that is an entity tag (i.e., not “O”), if it is not aligned to an output word, then we consider the output invalid. For example, an input like rate this book 5 out of 6 with a Back-Translated output give this book a rating of 5 would typically have no output word aligned to the source word “6” (best_rating slot label), so the output would be discarded.

Success rate and number of outputs (after de-duplication) for BT-5B are reported in Table 19.

<table>
<thead>
<tr>
<th>Modified Intent</th>
<th>SuccessRate</th>
<th>NumOutputs</th>
<th>AvgNumSlots</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>66.2</td>
<td>411</td>
<td>2.7</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>82.8</td>
<td>423</td>
<td>3.2</td>
</tr>
<tr>
<td>GetWeather</td>
<td>72.0</td>
<td>311</td>
<td>2.3</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>89.0</td>
<td>455</td>
<td>2.2</td>
</tr>
<tr>
<td>RateBook</td>
<td>79.2</td>
<td>478</td>
<td>3.8</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>85.5</td>
<td>451</td>
<td>1.8</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>72.0</td>
<td>330</td>
<td>2.2</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>78.1</strong></td>
<td><strong>408</strong></td>
<td><strong>2.6</strong></td>
</tr>
</tbody>
</table>

Table 19: For each intent, the Success Rate of Back-Translation with the 5B model, and the number of outputs, both averaged across the five random seeds. For reference, we also show the Average Number of Slots in the training data per intent.

M Filtering LINGUIST Outputs

We apply heuristic filtering by discarding outputs which meet any of the following criteria: (1) copy one of the examples from the prompt verbatim; (2) fail to follow the prompt instructions, by not copying the
instructed slot value or by producing repeated, missing, extra, or malformed slot-tag numbers; (3) produce the literal wildcard instruction " * "; or (4) produce a content word containing a punctuation character in the set of {<>[]()}{};}.²

In Table 20, we report the Success Rate as the portion of generated utterances which remain after filtering, and show the total number of generated utterances per intent. We observe a trend that success rate is generally lower when the prompt contains more slots, which is intuitive as the generation task is more challenging and has more chances to make a mistake. The success rates vary significantly by intent from 75.1 for BookRestaurant to 96.9 for GetWeather, with an average of 87.4 across the 7 intents.

<table>
<thead>
<tr>
<th>Modified Intent</th>
<th>Success Rate</th>
<th>#Outputs</th>
<th>Average #Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddToPlaylist</td>
<td>95.1</td>
<td>1230</td>
<td>2.7</td>
</tr>
<tr>
<td>BookRestaurant</td>
<td>75.1</td>
<td>2124</td>
<td>3.2</td>
</tr>
<tr>
<td>GetWeather</td>
<td>96.9</td>
<td>1197</td>
<td>2.3</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>82.3</td>
<td>622</td>
<td>2.2</td>
</tr>
<tr>
<td>RateBook</td>
<td>78.0</td>
<td>1729</td>
<td>3.8</td>
</tr>
<tr>
<td>SearchCreativeWork</td>
<td>91.3</td>
<td>1154</td>
<td>1.8</td>
</tr>
<tr>
<td>SearchScreeningEvent</td>
<td>93.0</td>
<td>1370</td>
<td>2.2</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>87.4</strong></td>
<td><strong>1346</strong></td>
<td><strong>2.6</strong></td>
</tr>
</tbody>
</table>

Table 20: For each intent, the Success Rate of Generation, and Number of Generated Outputs, both averaged across the five random seeds. For reference, we also show the Average Number of Slots in the training data per intent.

N SNIPS Dataset Details

We retrieve the SNIPS dataset from https://github.com/sonos/nlu-benchmark/tree/master/2017-06-custom-intent-engines. For each intent, we use “full” training set file, e.g. AddToPlaylist/train_AddToPlaylist_full.json to split into Train and Development sets (described in Section 4.1.1). The validation data comes from the “validate” file for each intent, e.g. AddToPlaylist/validate_AddToPlaylist.json.

In PlayMusic/train_PlayMusic_full.json, as of the time of publishing, there is a bug in row 461 (0-based), where we replace "Pop Punk Perfection <non_utf8_chars>" with "Pop Punk Perfection" before processing the dataset.

We provide in Table 21 the row IDs (0-based) and md5sum of the training data subsets we use for the “s10” New-Intent Few-Shot (NIFS) setting, described in Section 4.2.1.

O SemER Metric

For the internal IC+ST benchmark (Sections 4.1.3, 4.5.2, and 5.3.2), we report on Semantic Error Rate (SemER) (Su et al., 2018) which jointly evaluates Intent Classification and Slot Filling. SemER is defined as follows: comparing a reference of tokens and their accompanying labels, count each of these operations: (1) Correct slots, where the slot name and slot value is correctly identified, (2) Deletion errors, where the slot name is present in the reference but not in the hypothesis, (3) Insertion errors, where extraneous slot names are included in the hypothesis, (4) Substitution errors, where slot names from the hypothesis are included but with an incorrect slot value. Intent classification errors are substitution errors. Then, apply Equation 1 to compute the SemER.

\[
\text{SemER} = \frac{\# \text{ Del} + \# \text{ Ins} + \# \text{ Sub}}{\# \text{ Cor} + \# \text{ Del} + \# \text{ Sub}}
\] (1)

²These characters do not appear in the text of any of the original training data, so are considered to be generation mistakes.
<table>
<thead>
<tr>
<th>Seed</th>
<th>Intent</th>
<th>row IDs</th>
<th>md5sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AddToPlaylist</td>
<td>81 271 314 495 561 636 858 1285 1615 1702</td>
<td>ade55e42e48183c6617298d300758d8</td>
</tr>
<tr>
<td></td>
<td>BookRestaurant</td>
<td>122 438 574 739 950 1252 1615 1702 1702</td>
<td>606c7e4ad848f3c7cfaedb0b2b20f58</td>
</tr>
<tr>
<td></td>
<td>GetWeather</td>
<td>163 348 454 529 870 936 1286 1766</td>
<td>f37/c6a4040e81d57e046719e8bda10</td>
</tr>
<tr>
<td>1</td>
<td>PlayMusic</td>
<td>348 454 529 808 870 966 1286 1766</td>
<td>a9899270bccc0d35985d9b7149179616</td>
</tr>
<tr>
<td></td>
<td>RateBook</td>
<td>125 129 181 690 739 1243 1250 1600 1658</td>
<td>bcf6306351e156738669da436087284</td>
</tr>
<tr>
<td></td>
<td>SearchCreativeWork</td>
<td>75 76 126 256 272 412 712 1216</td>
<td>c8db82b2a2da4b343ca07</td>
</tr>
<tr>
<td></td>
<td>SearchScreeningEvent</td>
<td>76 117 236 261 411 785 919 1583</td>
<td>a32972f7f50a7779d2b0d625a30f10f7</td>
</tr>
<tr>
<td>2</td>
<td>AddToPlaylist</td>
<td>15 26 80 91 459 637 723</td>
<td>8463a2c95d6104cea85e096b1c9abb0f</td>
</tr>
<tr>
<td></td>
<td>BookRestaurant</td>
<td>172 246 829 999 1061 1203 1717</td>
<td>50f8c85c58c14d4387861968b88e721</td>
</tr>
<tr>
<td></td>
<td>GetWeather</td>
<td>155 466 857 957 1514 1673 1748 1810</td>
<td>4252f8c48f17062a003e73438254bdf8</td>
</tr>
<tr>
<td>3</td>
<td>PlayMusic</td>
<td>155 466 857 957 1514 1673 1748 1810 1930</td>
<td>94aa305f4a0e97b15f019fed0525</td>
</tr>
<tr>
<td></td>
<td>RateBook</td>
<td>210 349 506 527 596 717 1241 1299 1434</td>
<td>0678d112f3073a31f6bb6b406e6d4f0c</td>
</tr>
<tr>
<td></td>
<td>SearchCreativeWork</td>
<td>209 348 506 528 600 750 1241 1299</td>
<td>329937eeef31009e1029553e7368513</td>
</tr>
<tr>
<td></td>
<td>SearchScreeningEvent</td>
<td>210 522 660 862 880 951 983 1314</td>
<td>cdb4eb65ce6742a310622d1c2b04add</td>
</tr>
<tr>
<td>4</td>
<td>AddToPlaylist</td>
<td>177 244 912 1044 1218 1306 1374 1423 1541</td>
<td>781b6e76a79c496078c89e1e9593d41e</td>
</tr>
<tr>
<td></td>
<td>BookRestaurant</td>
<td>252 469 555 849 969 1053 1241 1324 1570</td>
<td>26ad74c585d8b8e458d1ed96f612e21</td>
</tr>
<tr>
<td></td>
<td>GetWeather</td>
<td>16 40 345 384 611 735 1071 1128</td>
<td>8a4d667e6c269c020b9a9c74924f9d</td>
</tr>
<tr>
<td>5</td>
<td>PlayMusic</td>
<td>16 40 345 611 735 1071 1094 1128 1587</td>
<td>45b220408499789b59505905f3535661</td>
</tr>
<tr>
<td></td>
<td>RateBook</td>
<td>561 840 937 1156 1234 1401 1314 1578</td>
<td>65b0a5b8c8a0fda441821926d299</td>
</tr>
<tr>
<td></td>
<td>SearchCreativeWork</td>
<td>263 402 515 598 819 877</td>
<td>8d4f37ebd8f9e23a48f4e6a191303c</td>
</tr>
<tr>
<td></td>
<td>SearchScreeningEvent</td>
<td>562 844 941 1160 1241 1253 1389 1437</td>
<td>4d4193a72a2407a099b04383d9bbad</td>
</tr>
</tbody>
</table>

Table 21: The Row IDs (0-based) used for the “s10” splits of the SNIPS dataset.