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 **L**anguage model **IN**struction tuning to **G**enerate annotated **U**tterances for **I**ntent classification and **S**lot **T**agging (**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

```
INPUT:
<language> English </language>
<intent> GetWeather </intent>
<include>
 [1 * ] , [3 snow ] [5 tomorrow ]
</include>
<labels>
 [1=geographic_poi , [2=country ,
 [3=condition_description , [4=city ,
 [5=timeRange
</labels>
<examples>
 Will the weather be okay in
  [1 Yellowstone National Park ]
  [5 one week from now ] ? <br>
 will it [3 rain ] at the [1 Statue of Liberty ]
  at [5 noon ] <br>
 What's the weather like
  at [1 Disneyworld ] in [5 november ] <br>
 I need the weather info for
  the [1 Guggenheim Museum ] in [2 Spain ] <br>
  What is the weather forecast
  for [5 October 12, 2022 ] in [4 Gyeongju ]
</examples>
OUTPUTS:

1. Give the [1 National Wildlife Refuge ]
   forecast for [3 snow ] [5 tomorrow ]

2. Can I get the [3 snow ] forecast
   for [1 Lake Tahoe ] [5 tomorrow ]?

3. I want to know if it will [3 snow ]
   [5 tomorrow ] at
   [1 Mount Rainier State Park ].
```

Figure 1: LINGUIST uses an instruction prompt to generate data with both user-requested slot values ("snow") and model-generated values ("*"). This model **has not seen any training data for GetWeather intent, or for the slot tag geographic_poi**: it was fine-tuned only on the other 6 SNIPS intents.

*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 Extrapoloation (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. Sennrich 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

token replacement: SeqMix (Zhang et al., 2020) replaces tokens with the nearest neighbor in the embedding space, Dai and Adel (2020) replaces slots with synonyms, or mentions from other instances of the same label. Easy Data Augmentation (Wei and Zou, 2019) includes synonym replacement, random insertion, random swap, and random deletion for classification task. Zheng et al. (2021) benchmark the success of LLMs on few-shot settings.

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 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.

3.2 Training the LINGUIST model

We fine-tune a pre-trained seq2seq model on pairs of LINGUIST instruction prompts and corresponding annotated target utterances, derived from a deduplicated IC+ST task dataset R. 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_Q_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 $d \sim \text{Geo}(0.5)$ $(0 \leq d \leq \#\text{ slots in } t_i)$ 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 sentencepice (Kudo and Richardson, 2018) tokenizer vocabulary (Appendix C.2).

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% Development. 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.

Intent	Train	Dev.	Valid.
AddToPlaylist	1884	58	100
BookRestaurant	1914	59	100
GetWeather	1940	60	100
PlayMusic	1940	60	100
RateBook	1898	58	100
SearchCreativeWork	1896	58	100
SearchScreeningEvent	1901	58	100
Total	13373	411	700

Table 1: Data counts per intent for SNIPS.

4.1.2 Multilingual ATIS++

For cross-lingual experiments we evaluate on mATIS++ (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 mATIS++ 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.

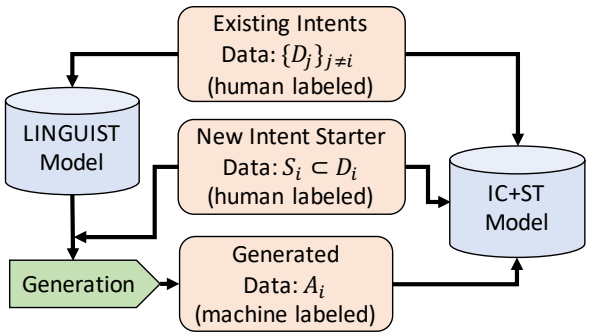


Figure 2: New-Intent Few-Shot (NIFS) Setup.

For SNIPS, we create 7 NIFS settings, one for each intent, reducing its training data down to only K=10 starter utterances S_i. We create 5 versions, each with a different random seed for selecting S_i, and always including at least one example for all slot types that occur for intent i.

To demonstrate the ability of LINGUIST to generalize to new intents and slots at inference time, **we exclude the new intent's starter utterances from fine-tuning**. For each intent i, we train a LINGUIST model on the other 6 intents $\{D_j\}_{j \neq i}$. Then, *during inference*, we forumlate prompts with the starter utterances between `<example>` and `</example>`, and generate more data. Note, this generation step **does not require any model parameter updates**.

4.2.2 NIFS Label Names Only (LNO)

In this more challenging variant of NIFS, *only the intent and slot label names* are available for the starter utterances, not their text or annotation. This is useful when developing new intents as we need only specify which slot types can go together, and need not curate or annotate any real examples. Notably, to the best of our knowledge, **LINGUIST is the first system capable of generating intent- and slot-annotated data in this setting** (as shown in Figure A4, Appendix B.1.4), by attending to the natural language label names in the prompt.

4.2.3 Zero-Shot Cross-Lingual

For mATIS++, we evaluate in the *zero-shot cross-lingual* setting, where real training data is available only for English. We fine-tune an IC+ST model on the English training data plus any augmented examples generated from this data, and evaluate the model on the test sets from all languages.

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 generaiton 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 $\sim$ 37M 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 weezer". 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.

Modified Intent / Data	Full	s10-NoUps	s10	s10 +ICLM	s10 +BT-Small	s10 +BT-5B	s10 +Ex2	s10 +LINGUIST
AddToPlaylist	100.0	95.6 ±6.8	98.3 ±2.1	97.5 ±2.2	98.3 ±2.4	**99.4** ±0.5	97.5 ±1.6	93.9 ±3.2
BookRestaurant	100.0	91.0 ±3.4	93.5 ±2.4	93.8 ±1.8	92.5 ±1.7	**94.6** ±1.1	91.3 ±5.4	**94.6** ±1.5
GetWeather	100.0	98.8 ±0.4	98.8 ±0.4	99.4 ±0.5	99.6 ±0.5	99.8 ±0.4	99.8 ±0.4	**100.0** ±0.0
PlayMusic	99.0	70.8 ±10.1	77.1 ±6.6	79.8 ±8.4	76.5 ±5.8	84.0 ±2.8	83.8 ±7.5	**90.4** ±4.7
RateBook	100.0	99.0 ±0.0	99.6 ±0.5	**100.0** ±0.0	99.8 ±0.4	99.6 ±0.5	99.8 ±0.4	**100.0** ±0.0
SearchCreativeWork	100.0	69.0 ±8.8	76.9 ±9.3	74.2 ±9.1	73.3 ±13.6	79.4 ±11.3	80.8 ±3.9	**83.3** ±6.9
SearchScreeningEvent	95.2	66.5 ±5.6	73.1 ±8.2	72.1 ±3.7	71.5 ±6.1	73.5 ±10.8	77.3 ±9.1	**81.9** ±3.9
Average	99.2	84.4 ±2.9	88.2 ±1.9	88.1 ±2.4	87.4 ±2.9	90.1 ±1.6	90.0 ±2.4	**92.0** ±0.8

(a) SNIPS New-Intent Few-Shot (NIFS) results on **Local Intent Recall**.

Modified Intent / Data	Full	s10-NoUps	s10	s10 +ICLM	s10 +BT-Small	s10 +BT-5B	s10 +Ex2	s10 +LINGUIST
AddToPlaylist	94.1	76.8 ±2.9	81.2 ±2.5	78.4 ±2.4	**82.0** ±1.8	81.3 ±1.7	80.6 ±3.1	80.9 ±3.4
BookRestaurant	96.4	71.9 ±2.3	81.3 ±2.1	80.4 ±1.4	81.2 ±1.2	83.3 ±2.5	78.6 ±4.5	**83.4** ±1.7
GetWeather	97.8	74.7 ±3.9	84.9 ±5.4	82.9 ±4.8	82.3 ±4.5	84.0 ±2.8	82.9 ±4.2	**85.4** ±2.8
PlayMusic	91.7	42.0 ±4.3	59.2 ±2.2	58.0 ±3.4	56.1 ±3.1	65.4 ±4.2	67.6 ±6.2	**70.1** ±1.8
RateBook	99.7	89.4 ±1.5	95.0 ±0.9	**95.4** ±0.8	93.5 ±3.3	93.6 ±1.5	94.7 ±0.6	94.8 ±1.7
SearchCreativeWork	100.0	56.2 ±10.5	70.9 ±11.3	67.6 ±9.8	68.9 ±11.1	72.9 ±12.1	75.2 ±5.2	**79.3** ±5.0
SearchScreeningEvent	96.6	56.4 ±7.4	71.6 ±3.6	72.8 ±4.6	69.8 ±4.6	74.2 ±3.6	79.0 ±4.3	**82.3** ±3.4
Average	96.6	66.8 ±2.2	77.7 ±2.0	76.5 ±2.1	76.3 ±2.5	79.2 ±2.8	79.8 ±2.1	**82.3** ±1.3

(b) SNIPS New-Intent Few-Shot (NIFS) results on **Local ST F1 Score**.

Table 2: Our main results on SNIPS Validation set (Section 4.1.1). For each cell (i, j), 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 re-sample 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-

ture 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 show 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.

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

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.

Modified Intent / Data	s10	LINGUIST (via s10 LNO)
AddToPlaylist	81.2 ± 2.5	45.9 ± 9.1
BookRestaurant	81.3 ± 2.1	74.8 ± 3.5
GetWeather	84.9 ± 5.4	71.2 ± 3.8
PlayMusic	59.2 ± 2.2	55.6 ± 3.1
RateBook	95.0 ± 0.9	55.0 ± 6.3
SearchCreativeWork	70.9 ± 11.3	59.3 ± 5.7
SearchScreeningEvent	71.6 ± 3.6	36.6 ± 6.1
Average	77.7 ± 2.0	**56.9** ± 2.5

Table 4: Local ST F1 Score results on SNIPS in the New Intent Few-Shot Label Names Only (NIFS-LNO) setting. "s10" results are copied from Table 2b.

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.

Lang	all	en	en+MT soft-align	en+ LINGUIST
en	98.10	97.77	–	97.77
de	97.32	90.51	96.66	94.08
es	97.21	95.20	97.20	97.10
fr	98.10	93.64	97.49	96.88
hi	95.20	88.62	92.81	94.08
ja	97.86	90.99	88.33	95.38
pt	97.32	93.97	96.78	92.86
avg-0S	97.17	92.16	94.88	**95.06**

Table 5: Results on mATIS++ Intent Accuracy.

Lang	all	en	en+MT soft-align	en+ LINGUIST
en	95.26	95.96	–	95.07
de	94.54	80.15	89.00	84.61
es	88.27	81.24	76.42	86.89
fr	92.69	77.29	79.64	83.83
hi	85.58	62.61	78.56	76.61
ja	92.76	24.52	79.10	86.32
pt	90.49	76.64	76.30	85.63
avg-0S	90.72	67.08	79.84	**83.98**

Table 6: Results on mATIS++ Slot F1.

5.3 Internal Dataset Results

Table 7 shows SemER on our Internal Dataset.

5.3.1 Baseline Results on Internal Dataset

For each feature i we train an IC+ST model M_i combining the Existing Features data E, upsampled starter utterances S_i, augmented utterances A_i produced from S_i via Slot Catalog Resampling, ICLM, and BT-Small. We evaluate on the feature's test set T_i, reporting Local SemER.

5.3.2 LINGUIST Results on Internal Dataset

We fine-tune a single LINGUIST model on instruction prompts formatted from the Existing Features dataset E (Section 4.1.3.), which *does not contain any examples of the new features*. Then, for each feature, i, following a similar procedure described in Section 5.1.2, we format prompts from the starter utterances S_i and apply LINGUIST to generate more data G_i. Finally, we follow the same data mixing, training, and testing procedure for each feature i as in the baseline (Section 5.3.1). As shown in Table 7, LINGUIST results in 7.9% to 25.2% relative SemER reduction across four languages compared to the baseline of combined Catalog Resampling, ICLM, and BT-Small.

Feature/Lang	**de**	**es**	**fr**	**ja**
CameraControl	-	-33.3%	-	-
ClockSettings	-1.6%	-12.3%	+1.8%	-
HomeSecurity	-	-	-	-31.5%
Music	-36.8%	-	-30.6%	-12.3%
Timers	-27.5%	-15.4%	-20.0%	-
Average	**-11.8%**	**-20.8%**	**-7.9%**	**-25.2%**

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

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan. Association for Computational Linguistics.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. SLURP: A spoken language understanding resource package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7252–7262, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling.

Eunah Cho, He Xie, and William M. Campbell. 2019. Paraphrase generation for semi-supervised learning in NLU. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 45–54, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3861–3867, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation.

Jack FitzGerald, Shankar Ananthakrishnan, Konstantine Arkoudas, Davide Bernardi, Abhishek Bhagia, Claudio Delli Bovi, Jin Cao, RAKESH CHADA, Amit Chauhan, Luoxin Chen, Anurag Dwarakanath, Satyam Dwivedi, Turan Gojayev, Karthik Gopalakrishnan, Thomas Gueudre, Dilek Hakkani-Tur, Wael Hamza, Jonathan Hueser, Kevin Martin Jose, Haidar Khan, Beiye Liu, Jianhua Lu, Alessandro Manzotti, Pradeep Natarajan, Karolina Owczarzak, Gokmen Oz, Enrico Palumbo, Charith Peris, Chandana Satya Prakash, Stephen Rawls, Andy Rosenbaum, Anjali Shenoy, Saleh Soltan, Mukund Harakere, Liz Tan, Fabian Triefenbach, Pan Wei, Haiyang Yu, Shuai Zheng, Gokhan Tur, and Prem Natarajan. 2022a. Alexa teacher model: Pretraining and distilling multi-billion-parameter encoders for natural language understanding systems. In *KDD 2022*.

Jack FitzGerald, Christopher Leo Hench, Charith S. Peris, Scott Mackie, Kay Rottmann, A. Patricia Domínguez Sánchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, L. Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and P. Natarajan. 2022b. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *ArXiv*, abs/2204.08582.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In *HLT*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2018. The sockeye neural machine translation toolkit at AMTA 2018. In *Proceedings of the 13th*

Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track), pages 200–207, Boston, MA. Association for Machine Translation in the Americas.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1627–1643, Online. Association for Computational Linguistics.

Shailza Jolly, Tobias Falke, Caglar Tirkaz, and Daniil Sorokin. 2020. Data-efficient paraphrase generation to bootstrap intent classification and slot labeling for new features in task-oriented dialog systems. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 10–20, Online. International Committee on Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Manoj Kumar, Yuval Merhav, Haidar Khan, Rahul Gupta, Anna Rumshisky, and Wael Hamza. 2022. Controlled data generation via insertion operations for NLU. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 54–61, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Kenton Lee, Kelvin Guu, Luheng He, Timothy Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation. *ArXiv*, abs/2102.01335.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

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

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation.

Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled text generation for data augmentation in intelligent artificial agents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 90–98, Hong Kong. Association for Computational Linguistics.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Subhadarshi Panda, Caglar Tirkaz, Tobias Falke, and Patrick Lehnen. 2021. Multilingual Paraphrase Generation For Bootstrapping New Features in Task-Oriented Dialog Systems. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 30–39, Online.

Alexandros Papangelis, Karthik Gopalakrishnan, Aishwarya Padmakumar, Seokhwan Kim, Gokhan Tur, and Dilek Hakkani-Tur. 2021. Generative conversational networks.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Paper s*, pages 2923–2934, Osaka, Japan. The COLING 2016 Organizing Committee.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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.

Gaurav Sahu, Pau Rodríguez López, Issam H. Laradji, Parmida Atighehchian, David Vázquez, and Dzmitry Bahdanau. 2022. Data augmentation for intent classification with off-the-shelf large language models. In *4th ACL Workshop on NLP for Conversational AI*, volume abs/2204.01959.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Saleh Soltan, Shankar Ananthakrishnan, Jack G. M. FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith S. Peris, Stephen Rawls, Andrew Rosenbaum, Anna Rumshisky, Chandan Prakash, Mukund Sridhar, Fabian Triefenbach, Apurv Verma, Gokhan Tur, and Premkumar Natarajan. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *ArXiv*, abs/2208.01448.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyridon Matsoukas. 2018. A re-ranker scheme for integrating large scale nlu models. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 670–676.

Gokhan Tur and Renato De Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Wiley.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tur, Dilek Z. Hakkani-Tur, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *CoRR*, abs/2109.09193.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual nlu.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rongzhi Zhang, Yue Yu, and Chao Zhang. 2020. SeqMix: Augmenting active sequence labeling via sequence mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8566–8579, Online. Association for Computational Linguistics.

Yanan Zheng, Jing Zhou, Yujie Qian, Ming Ding, Jian Li, Ruslan Salakhutdinov, Jie Tang, Sebastian Ruder, and Zhilin Yang. 2021. Fewnlu: Benchmarking state-of-the-art methods for few-shot natural language understanding.

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

The Examples in this section use a LINGUIST model fine-tuned only on 6 of the SNIPS intents, namely `AddToPlaylist`, `BookRestaurant`, `PlayMusic`, `RateBook`, `SearchCreativeWork`, and `SearchScreeningEvent`.

B.1.1 One-shot Novel Ambiguous Intent

We show (Figure A1) the LINGUIST model generating high quality 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, such as `get [1 zelda ]`, the model can leverage the intent label such as `DownloadGame` or `PurchaseGame`, to generate more relevant outputs.

```
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 ] ?
```

(a) The prompt with `DownloadGame` intent.

```
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 ]
```

(b) The prompt with `PurchaseGame` intent.

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

B.1.2 One-shot Novel Ambiguous Slot

Similarly, as shown in Figure A2, from a single example for a slot not seen in training data, LINGUIST can generate novel values, following the label name to disambiguate an ambiguous word such as "bat" (animal or sports equipment).

```
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 .
```

(a) The prompt with `animal` slot.

```
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 ]
```

(b) The prompt with `sports_equipment` slot.

Figure A2: For the exact same example and slots, LINGUIST can leverage the slot name in the prompt 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. Remarkbly, although the model never saw utterances about sports during fine-tuning, it can accuratey 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.

```
INPUT:
<language> English </language>
<intent> DraftPlayer </intent>
<include>
 [1 * ] , [2 Patrice Bergeron ] , [3 * ]
</include>
<labels>
 [1=baseball_team_name , [2=player, [3=year
</labels>
<examples> </examples>

OUTPUTS:

1. I want to draft [2 Patrice Bergeron ] from
   [1 Houston Astros ] for the [3 20th season ].

2. I want to draft [2 Patrice Bergeron ] for the
   [1 Washington Nationals ] in [3 2008 ].

3. I'd like to draft [2 Patrice Bergeron ] from
   [1 the Boston Red Sox ] in
   [3 2019 ]

4. I want to draft [2 Patrice Bergeron ] for
   [1 San Francisco Giants ] in [3 2007 ]

5. I need to draft [2 Patrice Bergeron ] from
   [1 Toronto Blue Jays ] for [3 2016 ]
```

(a) The prompt with `baseball_team_name` intent.

```
INPUT:
<language> English </language>
<intent> DraftPlayer </intent>
<include>
 [1 * ] , [2 Patrice Bergeron ] , [3 * ]
</include>
<labels>
 [1=hockey_team_name , [2=player, [3=year
</labels>
<examples> </examples>

OUTPUTS:

1. I would like to draft [2 Patrice Bergeron ] from
   [1 Vancouver Canucks ] in [3 2015 ]

2. I want to draft [2 Patrice Bergeron ] from
   [1 New Jersey Devils ] in [3 2023 ]

3. Draft the [2 Patrice Bergeron ] from
   [1 the St. Louis Blues ] for the upcoming
   [3 2013 ] season.

4. I want to add [2 Patrice Bergeron ] to
   [1 the Wildcats ] roster for [3 2012 ].

5. Draft [2 Patrice Bergeron ] for
   [1 Boston Bruins ] in [3 2014 ]
```

(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.

INPUT:
<language> English </language>
<intent> GetWeather </intent>
<include>
[1 *] , [2 *] , [3 *]
</include>
<labels>
[1=city , [2=timeRange , [3=state
</labels>
<examples> </examples>

OUTPUTS:

1. What is the weather like for [1 Oyster Bay] [3 AK] at [2 midnight]

2. What is the [2 time] for [1 Red Bank], [3 West Virginia]

3. What is the weather like in [1 North Lake Tahoe] [3 North Dakota]
   at [2 1 minute from now]

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.

INPUT:
<language> French </language>
<domain> travelinfo </domain>
<intent> flight </intent>
<include> [2 lundi] , [3 baltimore] , [4 dallas] </include>
<labels>
[1=depart_time.period_of_day , [2=depart_date.day_name ,
[3=toloc.city_name , [4=fromloc.city_name
</labels>
<examples>
show me the flights from [4 atlanta] to [3 denver] on [2 friday]

 what flights go from [4 charlotte] to [3 baltimore] in the [1 morning]

 show me all the flights from [4 philadelphia] to [3 cincinnati]

 flights on [2 friday] from [4 newark] to [3 tampa]

 what flights leave from [4 chicago] to [3 seattle] on [2 saturday]

 list flights from [4 detroit] to [3 san diego] on [2 thursday]

 [4 dallas] to [3 baltimore] [2 monday]
</examples>

OUTPUTS:

1. combien de vols y a-t-il entre [4 dallas] et [3 baltimore] le [2 lundi]

2. quels vols y a-t-il [2 lundi] entre [4 dallas] et [3 baltimore]

3. quel est le meilleur vol pour [2 lundi] entre [4 dallas] et [3 baltimore]

4. s'il vous plait trouvez les vols partant de [4 dallas] le [2 lundi]
   à destination de [3 baltimore]

5. quelles sont les vols qui partent le [2 lundi] de [4 dallas]
   à [3 baltimore]

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

C LINGUIST Training Details

C.1 Hyperparamters and Early Stopping

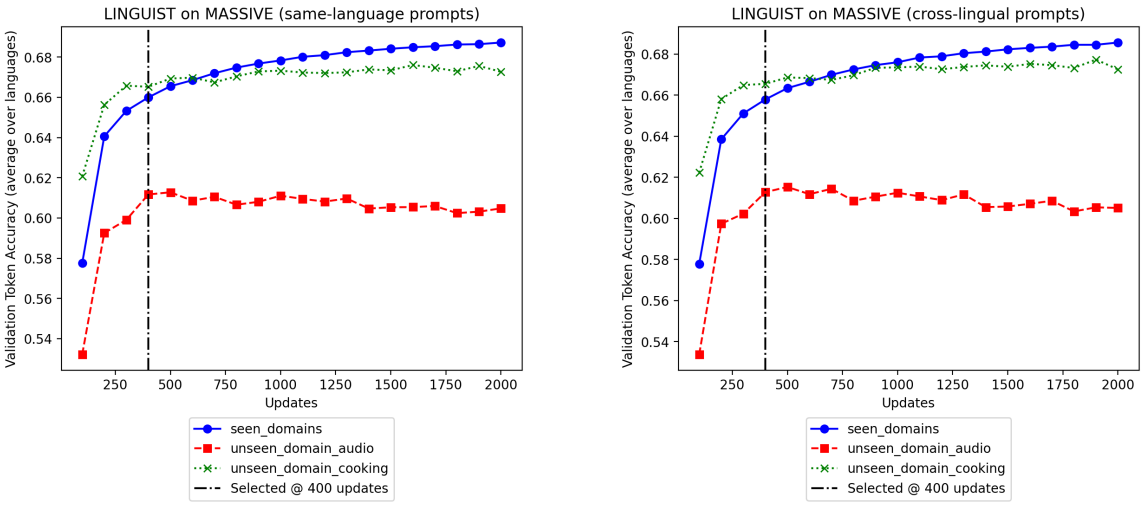
As shown in Figure A6, we find that the model converges after a very small number of updates. Specifically, we train with batch size 512 for 400 updates (i.e. around 18 epochs for SNIPS, around 2.5 epochs for MASSIVE), using a very small learning rate 5e-7 with Adam (Kingma and Ba, 2015), warmed up over 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.



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

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

Figure A6: Validation Accuracy across updates of fine-tuning LINGUIST on the MASSIVE dataset.

C.2 Tokenizer Choices

As mentioned in Section 3.2, we keep the original sentencepice (Kudo and Richardson, 2018) tokenizer of the model; we do not explicitly add vocabulary items for `<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 `<intent PlayMusic>`, and we found that the pre-trained model produced matching closing tags such as `</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`, tokenized as `['_[', '10']`. For inference on mATIS++, 47% of the prompts contain numbered brackets between `[11` and `[24`. 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.

Lang	all	en	en+MT soft-align	en+ LINGUIST	en+ LINGUIST (AlexaTM-Large 500M)
en	98.10	97.77	–	97.77	97.88
de	97.32	90.51	96.66	94.08	95.65
es	97.21	95.20	97.2	97.10	95.87
fr	98.10	93.64	97.49	96.88	95.98
hi	95.20	88.62	92.81	94.08	86.94
ja	97.86	90.99	88.33	95.38	91.55
pt	97.32	93.97	96.78	92.86	94.64
avg-0S	97.17	92.16	94.88	**95.06**	93.44

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

Lang	all	en	en+MT soft-align	en+ LINGUIST	en+ LINGUIST (AlexaTM-Large 500M)
en	95.26	95.96	–	95.07	95.51
de	94.54	80.15	89	84.61	83.02
es	88.27	81.24	76.42	86.89	85.45
fr	92.69	77.29	79.64	83.83	82.19
hi	85.58	62.61	78.56	76.61	75.36
ja	92.76	24.52	79.1	86.32	83.23
pt	90.49	76.64	76.3	85.63	84.18
avg-0S	90.72	67.08	79.84	**83.98**	82.24

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

experiments, the ablation study in this section (Table 10 for Local IC Recall and Table 11 for Local ST F1 Score) shows an improvement from removing the label dropout. The impact is small on "s10" (where the model sees both the label names and the annotated text for the 10 starter examples), however is quite larger in "LINGUIST (via s10 LNO)" (i.e. Label Names Only, Section 4.2.2), where the model sees only the label names.

We hypothesize that other changes we made along the way such as reducing the number of model updates via early stopping (Section C.1) and reducing the learning rate may have introduced regularization which makes label name dropout less necessary, and may surface a side effect of adding undesirable noise. In future work, we would like to study this more thoroughly, and investigate whether label dropout helps in cases where at inference time, the label names are either not available, or are not descriptive of the utterance semantics.

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

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.

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

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). "s10+LINGUIST (LNDrop: yes)" results are copied from Table 2b and "LINGUIST (via s10 LNO) LNDrop: no" results are copied from Table 4.

F 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` 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.

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

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.

Lang	all	en	en+MT soft-align	en+ LINGUIST (NoFilter)	en+ LINGUIST (+English-IC-Filter)	en+ LINGUIST (+English-IC-Filter) (+Balance-Classes)
en	98.10	97.77	–	97.21	97.77	97.77
de	97.32	90.51	96.66	94.31	93.64	94.08
es	97.21	95.20	97.2	94.75	96.88	97.10
fr	98.10	93.64	97.49	93.97	96.43	96.88
hi	95.20	88.62	92.81	88.17	92.19	94.08
ja	97.86	90.99	88.33	91.78	91.44	95.38
pt	97.32	93.97	96.78	95.54	93.30	92.86
avg-0S	97.17	92.16	94.88	93.09	93.98	**95.06**

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,

Lang	all	en	en+MT soft-align	en+ LINGUIST (NoFilter)	en+ LINGUIST (+English-IC-Filter)	en+ LINGUIST (+English-IC-Filter) (+Balance-Classes)
en	95.26	95.96	–	94.16	95.01	95.07
de	94.54	80.15	89	83.40	85.26	84.61
es	88.27	81.24	76.42	85.92	85.47	86.89
fr	92.69	77.29	79.64	84.65	84.71	83.83
hi	85.58	62.61	78.56	78.35	75.89	76.61
ja	92.76	24.52	79.1	85.72	85.38	86.32
pt	90.49	76.64	76.3	84.45	85.06	85.63
avg-0S	90.72	67.08	79.84	83.75	83.63	**83.98**

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

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 resutls 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.

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

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

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

(b) SNIPS New-Intent few-shot results on **Global ST F1 Score**.

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.

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 show 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.

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

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

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

(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, respecitvely.

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

Modified Intent	Num outputs
AddToPlaylist	296
BookRestaurant	347
GetWeather	322
PlayMusic	255
RateBook	288
SearchCreativeWork	295
SearchScreeningEvent	273
Average	297

Table 17: The number of filtered and de-duplicated outputs from ICLM per intent. All numbers are averaged across the five random seeds.

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

Modified Intent	SuccessRate	NumOutputs	AvgNumSlots
AddToPlaylist	70.2	64	2.7
BookRestaurant	72.8	73	3.2
GetWeather	60.4	60	2.3
PlayMusic	53.6	52	2.2
RateBook	70.8	71	3.8
SearchCreativeWork	41.6	42	1.8
SearchScreeningEvent	69.6	70	2.2
Average	62.7	62	2.6

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.

Modified Intent	SuccessRate	NumOutputs	AvgNumSlots
AddToPlaylist	66.2	411	2.7
BookRestaurant	82.8	423	3.2
GetWeather	72.0	311	2.3
PlayMusic	89.0	455	2.2
RateBook	79.2	478	3.8
SearchCreativeWork	85.5	451	1.8
SearchScreeningEvent	72.0	330	2.2
Average	78.1	408	2.6

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.

Modified Intent	Success Rate	#Outputs	Average #Slots
AddToPlaylist	95.1	1230	2.7
BookRestaurant	75.1	2124	3.2
GetWeather	96.9	1197	2.3
PlayMusic	82.3	622	2.2
RateBook	78.0	1729	3.8
SearchCreativeWork	91.3	1154	1.8
SearchScreeningEvent	93.0	1370	2.2
Average	87.4	1346	2.6

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 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.

$$\text{SemER} = \frac{\#\text{ Del} + \#\text{ Ins} + \#\text{ Sub}}{\#\text{ Cor} + \#\text{ Del} + \#\text{ Sub}} \tag{1}$$

[2]These characters do not appear in the text of any of the original training data, so are considered to be generation mistakes.

Seed	Intent	row IDs										md5sum
0	AddToPlaylist	81	271	314	495	561	636	856	1285	1615	1702	ade55e42e481f83c6617298d300758d8
	BookRestaurant	122	438	574	739	950	1252	1401	1420	1578	1728	60f6c7e4af848f3c7cfaedb02bb2f058
	GetWeather	163	348	454	529	870	932	966	1286	1368	1766	f37c6a4040e861d517e046719eb8da10
	PlayMusic	348	454	529	808	827	870	966	1286	1368	1766	a9899270bcfc3de3985d9b7149176916
	RateBook	125	129	181	690	739	1100	1243	1250	1600	1658	bcfa3063511c156738669da43f657284
	SearchCreativeWork	75	76	126	256	272	412	611	712	1216	1301	c8db82b42da972e2f3fbba4fb343ca97
	SearchScreeningEvent	76	117	236	261	411	785	919	1523	1856	1866	a32972f750a077709d22bd062b3a10f7
1	AddToPlaylist	15	26	80	91	459	637	723	735	844	1306	8463a2c95d6104cea85e096b1c9abb0f
	BookRestaurant	172	246	829	999	1061	1100	1203	1602	1717	1901	50cf89c558c1fd4387861968fbb8ea72
	GetWeather	155	466	857	957	1514	1673	1687	1748	1810	1930	4252f8c48f17062a003e73438254bdf8
	PlayMusic	155	466	857	957	1514	1683	1687	1748	1810	1930	94aa305f40be0d971b5fbf019fed0525
	RateBook	210	349	506	527	596	745	1174	1241	1295	1426	0b87d112f3073a31f6bb6b406e0d4f0c
	SearchCreativeWork	209	348	506	528	600	750	1175	1241	1299	1434	329937eef631008e1029f53ee7368513
	SearchScreeningEvent	210	522	660	862	880	951	983	1314	1766	1925	cdb4eb65ce6742a3106222d1e2b04add
2	AddToPlaylist	177	244	912	1044	1047	1218	1306	1374	1423	1541	f718676eafd6e2285ab11b2e35359aee
	BookRestaurant	252	469	555	849	969	1053	1113	1324	1570	1800	269ad74c558d8be458d1ee9dff612e21
	GetWeather	16	40	345	384	611	694	735	1071	1128	1587	8a4d66e79fc26c900c20baf99c7f429d
	PlayMusic	16	40	345	611	735	1071	1094	1128	1587	1840	45b22040849978b9590509ff53553661
	RateBook	561	840	937	1156	1234	1246	1314	1383	1401	1719	65b6a5bbc8ca80f4da4418362c16d299
	SearchCreativeWork	263	402	515	598	819	877	1116	1296	1657	1791	8d4f37ebd8f93e23a48f4ae60191303c
	SearchScreeningEvent	562	844	941	1160	1241	1253	1389	1393	1457	1635	4d4193a6a724b07a090bb483db8baded
3	AddToPlaylist	381	491	885	909	1341	1451	1459	1580	1778	1873	160fadf3bf9b76ce0af603992ae3a025
	BookRestaurant	42	245	570	702	1059	1227	1283	1330	1465	1887	319bd8b3fad9666ae1ebc4a3fa4bb9de
	GetWeather	90	127	502	522	759	910	957	1013	1242	1337	2188dd29d307e6200201eb0936b16a88
	PlayMusic	127	502	522	620	759	910	957	1093	1242	1840	3b1bb6d63a58ebe40b697aa1a7d9f1d7
	RateBook	42	70	372	447	768	1180	1594	1705	1838	1932	76fe7286f19552b820f0b9e4061bfe80
	SearchCreativeWork	42	70	372	447	772	1182	1207	1600	1711	1766	76de043a76b2c96d7e782ca890fb24d7
	SearchScreeningEvent	42	70	179	274	454	889	957	1058	1061	1256	6651011e32bbadaf55e169251d1f3a31
4	AddToPlaylist	26	58	276	328	403	574	834	1069	1644	1891	a2ce834b5ca753d67f88ff42530568f0
	BookRestaurant	228	270	519	946	1361	1482	1508	1832	1927	1936	d3d8c652313485e537b56e3279dddf80
	GetWeather	11	213	371	442	948	1040	1140	1280	1659	1835	b002ada961f3af1e95f34c46e2c3d047
	PlayMusic	11	213	277	371	442	948	1140	1280	1691	1835	81ed81653e9b7f8dbc0f05f38a97ecbb
	RateBook	58	128	208	815	876	891	941	1772	1784	1879	bce0c7d2d5b70a26378022ce3dba98fd
	SearchCreativeWork	58	127	207	817	894	943	1640	1699	1788	1878	54caeb85f764adac5856dd006d819418
	SearchScreeningEvent	58	128	596	894	952	962	1140	1365	1693	1928	767a9b5e0c83b4d043c74d57dcbba12e

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