CLASP: Few-Shot Cross-Lingual Data Augmentation for Semantic Parsing

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

Amir Saffari
Amazon, Cambridge, UK
amsafari@amazon.co.uk

Marco Damonte
Amazon, Cambridge, UK
dammarco@amazon.co.uk

Isabel Groves
Amazon, Cambridge, UK
isabeg@amazon.co.uk

Abstract

A bottleneck to developing Semantic Parsing (SP) models is the need for a large volume of human-labeled training data. Given the complexity and cost of human annotation for SP, labeled data is often scarce, particularly in multilingual settings. Large Language Models (LLMs) excel at SP given only a few examples, however LLMs are unsuitable for runtime systems which require low latency. In this work, we propose CLASP, a simple method to improve low-resource SP for moderate-sized models: we generate synthetic data from AlexaTM 20B to augment the training set for a model 40x smaller (500M parameters). We evaluate on two datasets in low-resource settings: English PIZZA, containing either 348 or 16 real examples, and mTOP cross-lingual zero-shot, where training data is available only in English, and the model must generalize to four new languages. On both datasets, we show significant improvements over strong baseline methods.

1 Introduction and Related Work

Semantic Parsing (SP) is the task of mapping a natural language sentence to a structured representation of its meaning. SP enables conversational agents to handle requests such as ordering pizza, creating reminders, and playing music. A bottleneck to developing SP models is their reliance on a large amount of human annotated training data, which is difficult and expensive to curate (particularly for multilingual settings) due to the complexity of the annotation task (Section 2). While Large Language Models (LLMs) perform well at SP given limited data (Shin et al., 2021), they are unsuitable for runtime systems which require low latency.

Data Augmentation (DA) is a common approach to mitigating data scarcity, and recently LLMs are shown to excel at in-context (Brown et al., 2020) training data generation for sentence-level tasks
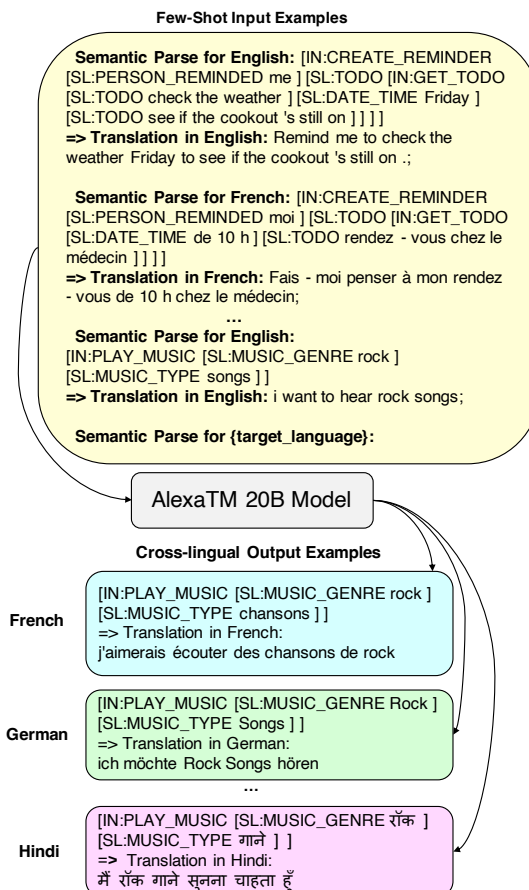

Figure 1: Cross-lingual Data Augmentation: AlexaTM 20B sees only a few examples of mTOP Semantic Parsing and can generate data in multiple languages.

(Sahu et al., 2022; Schick and Schütze, 2021; Wang et al., 2021). Fine-tuned LLMs can also generate data for English slot tagging (Lee et al., 2021) and multilingual intent classification and slot tagging (Rosenbaum et al., 2022). As we discuss in Section 2, SP poses unique challenges for DA, and remains relatively under-explored in the field. Prior work is either limited to heuristic re-combination of the training data (Andreas 2020; Jia and Liang 2016) or else assumes the availability of large-scale unannotated natural data (Yang et al., 2022). Furthermore, there is a gap in the literature on multilingual DA

---

[*]Corresponding Author

444

```
Input:          large pizza with extra    cheese  and pineapple hold the ham    and two    sprites    please
--------------------------------------------------------------------------------------------------------------------
Slot Tags:      Size  O    O    Quantity Topping  O   Topping   Not  O   Topping  O   Number DrinkType  O
Intent Classes: PizzaOrder , DrinkOrder
--------------------------------------------------------------------------------------------------------------------
Semantic Parse: (Order (Pizzaorder (Number 1 ) (Size large ) (Complex_topping (Quantity extra ) (Topping cheese ) )
                (Topping pineapple ) (Not (Topping ham ) ) ) (Drinkorder (Number 2 ) (Drinktype sprite ) ) ) )
```

Figure 2: Comparing "flat" semantics (Slot Tagging and Intent Classification, upper) to Semantic Parsing (lower).

for SP, as most existing work covers only English.

In this work, we extend the general example of DA via LLM prompting to the SP task. Using AlexaTM 20B (Soltan et al., 2022), we generate synthetic training examples for SP, to augment low-resource settings for moderate-sized models.

We evaluate on two datasets: English PIZZA (Arkoudas et al., 2021) and cross-lingual mTOP (Li et al., 2021). On **PIZZA**, we first establish a new SOTA baseline by improving upon the Canonical Form targets of Rongali et al. (2022) and tuning the amount of grammar-generated training data, then show that **our method improves by 4.79 points (from 80.40 to 85.19)** on the few-shot n=16 setting on Unordered Exact Match (Arkoudas et al., 2021). On **mTOP**, we demonstrate **6.1 points improvement (from 60.3 to 66.4)** on Exact Match in the cross-lingual one-shot setting, compared to machine translation with slot alignment.

2 Motivation

2.1 Why Semantic Parsing?

Consider an example from PIZZA (Arkoudas et al., 2021): "*large pizza with extra cheese and pineapple hold the ham and two sprites please*". As shown in Figure 2, SP evolves beyond "flat" semantics to exctract complex information such as the implicit Number slot, the scope of modifiers Quantity and Not, and the association between slots and intents.

2.2 Data Augmentation Challenges for SP

The core of many standard DA methods is to modify the text from an existing annotated sample, assume the same label applies, and accept the novel text-label pair as training data. For example, a model might paraphrase "order a pizza with basil" to "order a pizza with *extra* basil", which would no longer match the original Semantic Parse.

Similarly, in cross-lingual settings (i.e., data is available in one language and the model must perform the task on other languages), a standard approach for sentence-level tasks is to translate the text and keep the label. For SP however, the target parse must also be updated with the translated slot values. Li et al. (2021) translate the text then align words to recover the parse. However, this second alignment step may introduce errors (Appendix I).

3 CLASP Methods

To address the challenge of maintaining text-label agreement when generating SP training data, we propose CLASP (Cross-Lingual data Augmentation for Semantic Parsing). CLASP consists of four methods for prompting LLMs to generate training data, either in the Same Language [SL] or Cross-Lingually [CL]: (1) **RS**: Replace Slots, Generate Text [SL]; (2) **TS**: Translate Slots, Generate Text [CL]; (3) **GB**: Generate Both Parse and Text [SL]; and (4) **TB**: Translate Both Parse and Text [CL].

3.1 RS: Replace Slots, Generate Text [SL]

As shown in Figure 3 (Appendix A.1), we start with a real training example, $e_i = (x_i, y_i)$ such as with input text x_i = "*i need to get five small mushroom and bacon pizzas with a pepsi*", and target ground-truth parse y_i = "*(Pizzaorder ... (Topping mushroom) ...)*". To create a novel training example $e_i' = (x_i', y_i')$ we apply a modification $F(\cdot)$ on the parse y_i to obtain $y_i' = F(y_i)$, then prompt a LLM to generate a corresponding text x_i'.

Specifically, $F(\cdot)$ randomly selects one slot (leaf nodes in the parse tree) of y_i, and **replaces the slot value in the parse** with a different value from a catalog. In this instance, we replace the Topping "*mushroom*" with "*spinach*", giving y_i' = "*(Pizzaorder ... (Topping spinach) ...)*". To help the model understand how to **generate the text** x_i', we include in the prompt 4 other context examples $\{c_j = (x_j, y_j)\}_{j=1}^4$ followed by the original example e_j, each verbalized as `Semantic Parse:` y_i `Translation in English:` x_i.

3.2 TS: Translate Slots, Generate Text [CL]

This method extends the idea of CLASP-RS to cross-lingual data generation: we **translate each slot value** into the target language l and prompt the LLM to **generate the corresponding text** in l. See an example in Figure 4 (Appendix A.2).

3.3 GB: Generate Both Parse and Text [SL]

CLASP-RS provides control over the slot values, but cannot add or remove slot or intents. Instead, CLASP-GB **generates both** the parse and text together, giving the model flexibility to generate more diverse outputs (Figure 5 in Appendix A.3).

3.4 TB: Translate Both Parse and Text [CL]

Given the difficulty of translating a slot value out of context, which may lead to cascading errors, we propose to apply the LLM to **translate both** the parse and the corresponding text (Figure 1).

4 Experimental Setup

4.1 Datasets

We evaluate CLASP on two datasets: PIZZA (Arkoudas et al., 2021) and mTOP (Li et al., 2021).

PIZZA is a challenging English dataset of SP for the food ordering domain. We follow the setting of Rongali et al. (2022), namely converting the parse targets to a Canonical Form (CF) closer to natural language; training on the annotated "dev" portion, either full (n=348) or few-shot (n=16); and reporting on the "test" portion of 1,357 utterances. We use 10% of the test set for checkpoint selection.

We iterate upon the CF targets used for training, by naturalizing from TOP-style parse to CF while *preserving the order of sibling slots and intents from the original text*. (Appendix B.1). Note that this applies only at training time.[1]

PIZZA also provides 2.5M grammar-generated "train" samples, and catalogs of values for each slot.

mTOP (Li et al., 2021) is a larger-scale multilingual SP dataset covering 11 domains and 6 languages. The splits are "train" (15,667 English, 10k-11k others), "validation" (2,235 English, 1k-2k others), and "test" (4,386 English, 2k-3k others). We follow a cross-lingual one-shot setting: full training and validation data is available for English only, we use one training example from each other language for in-context prompts (Appendix G), and we test on all languages, however excluding Thai which is not supported by our pre-trained LMs. The mTOP dataset provides two options for the input text, either "Utterance", or "tokens". We use space-joined "tokens" which resolves many (although not all) string matching anomalies (Appendix B.2).

---

[1]We release the alternate PIZZA dataset used in this paper at https://github.com/amazon-research/pizza-semantic-parsing-dataset/tree/main/data/alternate-canonical.

4.2 Baselines

For **PIZZA**, we cite Rongali et al. (2022), who fine-tune BART (Lewis et al., 2020), including joint training with auxiliary tasks and constrained decoding. We also explore using various amounts m of (grammar-generated) train data, both in isolation and mixed with the (annotated) dev set. Selecting the best-performing m from values between 348 and 174,000 (Appendix E) we use m=69,600 for train in isolation. For combining with dev n=348 / n=16, we use train m=3,480 / m=104,400. For combining with dev and CLASP, we always use train m=348.

For **mTOP**, we implement machine translation of the text, via Opus MT (Tiedemann and Thottingal, 2020) and via the 20B model (using a one-shot in-context prompt, Figure 6 in Appendix A.4). We use Sim-Align (Jalili Sabet et al., 2020) (Appendix J) to align the translated sentence to the original English, to recover the target-language parse.

4.3 CLASP Settings

For **PIZZA**, we apply two CLASP methods: CLASP-RS (Sec. 3.1) and CLASP-GB (Sec. 3.3) to generate novel training data based on the dev set. For each method, we generate k=3,480 samples We also try including the union of data from the two CLASP methods, referred to as CLASP-{RS,GB}.

For **mTOP**, we use CLASP-TS (Sec. 3.2) and CLASP-TB (Sec. 3.4) to generate training data in other languages from the English source. We select a single example from each of the four target languages (de, es, fr, and hi; shown in Appendix G) to use in one-shot prompts for generation. We filter the outputs as described in Appendix H.

Regardless of which and how much data we add, we always up-sample the non-synthetic data source (dev for PIZZA, English data for mTOP) to account for 50% of the mass of utterances seen during training, and scale down the number of epochs to fix the total number of *model updates* across experiments.

4.4 Metrics

We use the form of Exact Match (EM) standard for each dataset: Unordered Exact Match (UEM) (Arkoudas et al., 2021) for PIZZA, which is invariant to different order of sibling nodes in parses; and Space- and Case-Insensitive Exact Match (SCIEM) (Appendix C) for mTOP, which is invariant to different spacing and casing of slot values.

4.5 Models

For CLASP data generation, we leverage in-context learning with AlexaTM 20B (Soltan et al., 2022).

For Semantic Parsing fine-tuning (Rongali et al. (2020), details in Appendix F), we use AlexaTM-Large 500M, a 500-million-parameter seq2seq Transformer (Vaswani et al., 2017) pre-trained similarly to AlexaTM 20B (Soltan et al., 2022), however with denoising objective only (no Causal Language Modeling). This model has 12 encoder, 12 decoder layers, and 1024 hidden size (same as (m)BART (Liu et al., 2020)). For mTOP we use sentinel words (Raman et al., 2022) which function similarly to pointers (Appendix B.2.2). At test-time inference, we use the top-1 hypothesis from beam search 4 (Appendix D).

5 Results

5.1 PIZZA Results

Results are presented in Table 1. We first note that applying our Fixed Canonical Form to dev-only provides a very large boost in performance, from 82.54/21.00 for n=348/n=16 to 90.05/58.00, an improvement of 7.51 and 37.00 points, respectively. For n=16, dev-only with Fixed CF already outperforms the best system reported by Rongali et al.. We show that training data (which is grammar-generated) on its own under-performs at 59.84, however it can help a lot when combined with the dev set, providing 92.70/80.40.

Both CLASP methods improve significantly over dev-only: CLASP-RS provides 92.04/60.65 and CLASP-GB provides 93.52/77.75. Combining data from the CLASP methods (CLASP-{RS,GB}) shows a slight improvement on n=348, however is 2.14 points behind CLASP-GB alone on n=16. Finally, our best performing system uses the fixed Canonincal Form with data from dev, train, and both CLASP methods together, obtaining a new SOTA by a wide margin: **95.06 for n=348** setting, and **85.19 for n=16** setting.

5.2 mTOP Results

Results are presented in Table 2, where the main focus is on "avg-0s" ("average-zero-shot"), the average across the non-English languages. Training on English data only ("en-only") is a lower bound of 45.3, and training on all languages together ("ALL") is an upper bound of 73.5, i.e. a gap of 28.2 points. The baseline MT with Slot-Alignment ("MT-Opus") provides 15.0 points improvement

Data	Unordered EM	
Original CF	n=348	n=16
dev-only (ours)	82.54	21.00
dev-only (Rongali et al.)	87.25	16.95
Rongali et al. best	–	49.89
Fixed CF (all ours)		
dev-only	90.05	58.00
train-only	59.84	59.84
dev+train	92.70	<u>80.40</u>
dev+CLASP-RS	92.04	60.65
dev+CLASP-GB	93.52	77.75
dev+CLASP-{RS,GB}	<u>93.81</u>	75.61
dev+train+CLASP-{RS,GB}	**95.06**	**85.19**

Table 1: Results on PIZZA dataset with Unordered Exact Match (UEM) metric. The best and second-best numbers are bolded and underlined, respectively. Original CF is the Canonical Form of Rongali et al. (2022). Fixed CF is our fixed Canonical Form (Sec. 4.1), and n is the number of samples available from the dev set.

over "en-only", from 45.3 to 60.3. Scaling up the MT model size ("MT-20B") does not provide improvement, matching "MT-Opus" at 60.3.

Non-en data	en	de	es	fr	hi	avg 0s
Lower/Upper Bounds and Baseline						
en-only	83.1	47.3	51.0	54.8	28.2	45.3
ALL	83.3	70.3	77.3	75.9	70.5	73.5
MT-Opus	83.0	63.8	65.0	65.1	47.4	60.3
Single Methods						
MT-20B	83.3	63.8	64.3	65.2	47.8	60.3
CLASP-TS	82.9	62.8	62.6	67.2	57.9	62.6
CLASP-TB	83.3	65.4	64.4	66.3	54.7	62.7
Combination of Methods						
CLASP-{TS,TB}	83.4	64.2	63.7	68.4	<u>59.2</u>	63.9
CLASP-{TS,TB} +MT-20B	<u>83.8</u>	<u>66.3</u>	<u>65.9</u>	<u>69.0</u>	**59.7**	<u>65.2</u>
CLASP-{TS,TB} +MT-20B +MT-Opus	**84.4**	**66.7**	**68.1**	**72.6**	58.1	**66.4**
CRISS with Pointers (Li et al., 2021) (for reference only)						
en-only	84.2	36.1	48.6	46.6	31.2	40.6
ALL	84.1	74.4	79.1	77.7	74.7	76.5
MT	84.2	62.8	73.3	71.7	63.2	67.8

Table 2: Our mTOP results, where 'avg-0s' is averaged across the non-en languages. Li et al. (2021) is cited for reference only, and are not directly comparable due to using a stronger backbone model (CRISS, (Tran et al., 2020)) with a higher upper bound ("ALL"). Our best result is bolded, and our second best is underlined.

CLASP-TS and CLASP-TB provide 62.6 and 62.7, respectively, while their combination (CLASP-{TS,TB}) improves further to 63.9. Adding data from "MT-20B" increases to 65.2, and finally by combining data from both CLASP meth-

ods and both MT models, our best result is 66.4, i.e. **6.1 points improvement** over the baseline. The gain is particularly large for **Hindi: 12.3 points improvement** over the baseline (from 47.4 to 59.7).

6 Conclusion and Future Work

We have demonstrated CLASP, a simple method to generate synthetic training data for multi-lingual Semantic Parsing by prompting a frozen Large Language Model. In very low-resource (n=16 and n=1) settings, on two datasets covering five languages, we show significant improvements over strong baseline methods. In future work, we would like to evaluate on more languages and datasets, combine our method with CRISS style pre-training, and extend our method to more tasks such as Text-to-SQL and Code Generation.

References

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Konstantine Arkoudas, Nicolas Guenon des Mesnards, Melanie Rubino, Sandesh Swamy, Saarthak Khanna, and Weiqi Sun. 2021. Pizza: a task-oriented semantic parsing dataset.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for 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.

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. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, 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. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. Zero: Memory optimizations toward training trillion parameter models.

Karthik Raman, Iftekhar Naim, Jiecao Chen, Kazuma Hashimoto, Kiran Yalasangi, and Krishna Srinivasan. 2022. Transforming sequence tagging into a seq2seq task.

Subendhu Rongali, Konstantine Arkoudas, Melanie Rubino, and Wael Hamza. 2022. Training naturalized semantic parsers with very little data. *arXiv preprint arXiv:2204.14243*.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. *Proceedings of The Web Conference 2020*.

Andy Rosenbaum, Saleh Soltan, Wael Hamza, Yannick Versley, and Markus Boese. 2022. Linguist: Language model instruction tuning to generate annotated utterances for intent classification and slot tagging.

Gaurav Sahu, Pau Rodriguez, Issam Laradji, Parmida Atighehchian, David Vazquez, and Dzmitry Bahdanau. 2022. Data augmentation for intent classification with off-the-shelf large language models. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 47–57, Dublin, Ireland. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. 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.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.

Chau Tran, Yuqing Tang, Xian Li, and Jiatao Gu. 2020. Cross-lingual retrieval for iterative self-supervised training.

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. *ArXiv*, abs/2109.09193.

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.

Kevin Yang, Olivia Deng, Charles Chen, Richard Shin, Subhro Roy, and Benjamin Van Durme. 2022. Addressing resource and privacy constraints in semantic parsing through data augmentation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3685–3695, Dublin, Ireland. Association for Computational Linguistics.

A Sample Model outputs

A.1 Example of CLASP-RS: Replace Slots and Generate Text

We show an example of CLASP-RS (Replace Slots and Generate Text) in Figure 3.

```
INPUT:
[CLM] Semantic Parse: (Order
  (Pizzaorder (Number a ) (Size medium ) (Style supreme ) )
  (Drinkorder (Number a ) (Drinktype sprite ) ) );
Translation in English:
  order me a medium supreme pizza and a sprite;
Semantic Parse: (Order
  (Pizzaorder (Number two ) (Topping bacon ) (Topping onion ) )
  (Drinkorder (Number a ) (Size large ) (Drinktype mountain dew ) ) );
Translation in English:
  put in my order for two bacon and onion pizzas and include a large mountain dew;
Semantic Parse: (Order
  (Pizzaorder (Number two ) (Size large ) (Topping pepperoni ) (Topping mushrooms ) )
  (Drinkorder (Number four ) (Size large ) (Drinktype cherry cokes ) ) );
Translation in English:
  two large pizzas with pepperoni and mushrooms and four large cherry cokes;
Semantic Parse: (Order
  (Pizzaorder (Number one ) (Size small ) (Topping yellow peppers ) (Topping olives ) )
  (Drinkorder (Number two ) (Containertype cans ) (Drinktype coke ) ) );
Translation in English:
  place an order for one small pizza with yellow peppers and olives
  and also include two cans of coke with it;
Semantic Parse: (Order
  (Pizzaorder (Number five ) (Size small ) (Topping mushroom ) (Topping bacon ) )
  (Drinkorder (Number a ) (Drinktype pepsi ) ) );
Translation in English:
  i need to get five small mushroom and bacon pizzas with a pepsi;
Semantic Parse: (Order
  (Pizzaorder (Number five ) (Size small ) (Topping spinach ) (Topping bacon ) )
  (Drinkorder (Number a ) (Drinktype pepsi ) ) );
Translation in English:

OUTPUTS:

0: five small spinach and bacon pizzas with a pepsi

1: put in my order for five small spinach and bacon pizzas and include a pepsi

2: five small spinach and bacon pizzas and a pepsi

3: please place my order for five small spinach and bacon pizzas with a pepsi

4: put my order in for five small spinach and bacon pizzas with a pepsi
```

Figure 3: CLASP-RS: Replace Slots and Generate Text. In this example from the PIZZA dataset, we have replaced the value of Topping mushroom with Topping **spinach**. The model sees c=5 context examples, the last of which is the original utterance, and is prompted to generate text matching the parse with the replaced slot. The model generates reasonable paraphrases, including the requested slots. In particular, the model can both mix and match carrier phrase components from the prompted examples (e.g. "include a pepsi") and generate novel carrier phrases, (e.g. "please place my order") presumably relying on general language knowledge acquired during unsupervised pre-training. Note that "[CLM]" is a special token which the model expects during in-context learning.

A.2 Example of CLASP-TS: Translate Slots and Generate Text

We show an example of CLASP-TS (Translate Slots and Generate Text) in Figure 4.

```
INPUT:
[CLM] Semantic Parse: [IN:CREATE_REMINDER
    [SL:PERSON_REMINDED me ]
    [SL:TODO [IN:GET_TODO [SL:DATE_TIME 10 : 00 am ] [SL:TODO doctor 's appointment ] ] ] ];
Translation in English:
  Remind me of my 10 : 00 am doctor 's appointment;
Semantic Parse: [IN:CREATE_REMINDER
    [SL:PERSON_REMINDED moi ]
    [SL:TODO [IN:GET_TODO [SL:DATE_TIME de 10 h ] [SL:TODO rendez - vous chez le médecin ] ] ] ];
Translation in French:
  Fais - moi penser à mon rendez - vous de 10 h chez le médecin;
Semantic Parse: [IN:SEND_MESSAGE
    [SL:RECIPIENT [IN:GET_CONTACT [SL:CONTACT_RELATED my ] [SL:TYPE_RELATION husband ] ] ]
    [SL:CONTENT_EXACT pick up bread ];
Translation in English:
  Send a message to my husband reminding him to pick up bread;
Semantic Parse: [IN:SEND_MESSAGE
    [SL:RECIPIENT [IN:GET_CONTACT [SL:CONTACT_RELATED mon ] [SL:TYPE_RELATION mari ] ] ]
    [SL:CONTENT_EXACT prendre du pain ] ];
Translation in French:

OUTPUTs:

0: Envoie un message à mon mari pour lui rappeler de prendre du pain

1: Envoie un message à mon mari lui demandant de prendre du pain

2: Envoie un message à mon mari lui rappelant de prendre du pain
```

Figure 4: CLASP-TS (Translate Slots and Generate Text). We first translate the slots replacing them in the parse, then use a one-shot in-context prompt to instruct the 20B model to generate a matching sentence in the target language.

A.3 Example of CLASP-GB: Generate Both Parse and Text

An example is shown in Figure 5

```
INPUT:
[CLM] Semantic Parse: (Order
  (Pizzaorder (Number a ) (Size small ) (Topping peppers ) (Topping sausage ) (Topping pineapple ) ) )
=> Translation in English:
  can you get me a small pizza with peppers and sausage and pineapple please;
Semantic Parse: (Order
  (Pizzaorder (Number a ) (Size large ) (Topping olives )
    (Complex_topping (Quantity extra ) (Topping cheese ) ) (Topping chicken ) ) )
=> Translation in English:
  i need a large pizza and i want olives and extra cheese as well as chicken on it thanks a lot;
Semantic Parse: (Order
  (Pizzaorder (Number a ) (Size medium ) (Topping onions ) (Topping tuna ) (Topping ham ) ) )
=> Translation in English:
  i'd like a medium pizza with onions tuna and ham;
Semantic Parse: (Order
  (Pizzaorder (Number two ) (Topping olive ) (Topping pineapple ) (Topping mushroom ) ) )
=> Translation in English:
  i want two olive pineapple and mushroom pies;
Semantic Parse: (Order
  (Pizzaorder (Number a ) (Size large ) (Topping ham ) (Topping peppers ) (Not (Style thin crust ) ) ) )
=> Translation in English:
  good evening how are you do me a favor and get me a large pizza with ham and peppers
    i definitely do not want thin crust thanks;
Semantic Parse:

OUTPUTS:

0: (Order (Pizzaorder (Number two ) (Topping olive ) (Topping pineapple ) (Topping mushroom )
    (Not (Style thin crust ) ) ) =>
  Translation in English:
    can you get me two olive pineapple and mushroom pies please no thin crust;

1: (Order (Pizzaorder (Number a ) (Size medium ) (Topping tuna ) (Topping chicken ) ) ) =>
  Translation in English:
    hello how are you i want a medium pizza with tuna and chicken on it thanks;

2: (Order (Pizzaorder (Number a ) (Size small ) (Topping chicken ) (Topping bacon ) ) ) =>
  Translation in English:
    can you please bring me a small pizza with chicken and bacon on it thanks;

3: (Order (Pizzaorder (Number a ) (Size large ) (Topping mushroom )
    (Topping pepperoni ) (Topping green pepper ) ) ) =>
  Translation in English:
    how are you today i want a large pizza with mushrooms pepperoni green peppers and cheese thanks;
```

Figure 5: CLASP-GB: AlexaTM 20B generates both the parse and matching text remarkably well given that it has only seen five examples in the prompt. Intuitively this is this most ambiguous of the tasks, as the model has freedom to generate both a new parse and a new text. Therefore, we see a bit noisier generation in this mode, as shown in the example where an extra slot value "cheese" appears in the text but not in the parse. We filter out such examples with a heuristic string match check.

A.4 Example of In-Context Sentence Translation

An example is shown in Figure 6.

```
INPUT:
[CLM] Sentence: remind me to call tim after work tomorrow at 6 pm;
Translation in Hindi: कल काम से आने के बाद शाम 6 बजे टिम को कॉल करने के लिए मुझे रिमाइंड करवाएँ;
Sentence: What are the most popular recipes on Food Network;
Translation in Hindi:

OUTPUTS:

0: फूड नेटवर्क पर सबसे लोकप्रिय रेसिपी क्या है;
```

Figure 6: An example of in-context Sentence text translation from English to Hindi.

B Data Preprocessing

We discuss preprocessing for each of our datasets.

452

B.1 Data Preprocessing for Pizza

We provide more details about our modified Canonical Form (CF) training data, as introduced in Section 4.1. We compare the Canonical Forms released by Rongali et al. (2022) [2] with the original pizza text, TOP, and EXR released by Arkoudas et al. (2021).[3] An example is shown in Figure 7, where we see that in the original data release, EXR does not preserve the sibling order of nodes in the tree. It appears that the CF of Rongali et al. (2022) follows the EXR, so it inherits this mismatch.

We hypothesize that this mismatch in sibling order creates an extra challenge for the model to learn at training time, and limits the power of the naturalization approach proposed by Rongali et al. (2022). In particular, in the 16-shot setting, we find that 12 out of the 16 utterances have a canonical form that does not match the original sibling order.

Observing that the sibling order is still faithfully represented in the "TOP" field of the corresponding utterances in the Pizza dataset (Arkoudas et al., 2021), we re-produce the CF from TOP directly, using the same codebase as Rongali et al. (2022).[4] Note, we only perform this change during training time. At testing time, we follow (Rongali et al., 2022) and use the standard grammar to parse the model output and compare using Unordered Exact Match (UEM) against the ground-truth EXR (entity resolved) format.

As shown in Section 5.1, our fixed Canonical Form provides a very large improvement across all runs, in particular increasing UEM from 82.54/21.00 to 90.05/58.00 on n=348/n=16, respectively. This represents 7.51/37.00 points absolute improvement, respectively.

```
Text in Arkoudas et al. (2021):
  can you get me a pizza with peppers and sausage and pineapple please

TOP in Arkoudas et al. (2021):
  (ORDER can you get me (PIZZAORDER (NUMBER a ) (SIZE small ) pizza with
      (TOPPING peppers ) and (TOPPING sausage ) and (TOPPING pineapple ) ) please )

TOP-Decoupled we produced using code at Arkoudas et al. (2021):
  (ORDER (PIZZAORDER (NUMBER a ) (SIZE small )
      (TOPPING peppers ) (TOPPING sausage ) (TOPPING pineapple ) ) )

EXR in Arkoudas et al. (2021):
  (ORDER (PIZZAORDER (NUMBER 1 ) (SIZE SMALL )
      (TOPPING PEPPERS ) (TOPPING PINEAPPLE ) (TOPPING SAUSAGE ) ) )

Rongali et al. (2022) CF for this utterance:
  i want one small pizza with peppers , pineapple , and sausage

Our CF:
  i want one small pizza with peppers , sausage , and pineapple
```

Figure 7: Comparing our "Fixed" Canonical Form ("Our CF") to the original provided by Rongali et al. (2022). We use the same code to resolve, we just start with the TOP and TOP-Decoupled versions provided in the dataset, which maintain the ordering of slots in the original.

B.2 Data Preprocessing for mTOP

We describe two data pre-processing steps for mTOP: (1) Space-joined Tokens, and (2) Sentinel Words. As shown in Table 3, these steps have minimal impact on non-English languages when training on ALL data (from 73.4 to 73.5), however **improve lower bound cross-lingual zero-shot by 17.0 points** (from 28.3 to 45.3). Furthermore, our data pre-processing provides a moderate improvement on English, of 0.8 points (from 82.3 to 83.1) when training on en-only data, and 0.9 points (from 82.4 to 83.3) when training on ALL data.

Data	Input Source	Word Sentinels	en	de	es	fr	hi	avg-0s
	Utterance	no	82.3	31.8	28.5	32.7	20.3	28.3
en-only	Space-joined Tokens	no	82.9	34.6	36.6	39.8	22.8	33.4
	Space-joined Tokens	**yes**	**83.1**	47.3	51.0	54.8	28.2	**45.3**
	Utterance	no	82.4	71.3	77.4	74.9	70.1	73.4
ALL	Space-joined Tokens	no	82.0	71.7	76.7	75.1	68.7	73.0
	Space-joined Tokens	**yes**	**83.3**	70.3	77.3	75.9	70.5	**73.5**

Table 3: Results for cross-lingual zero-shot and ALL languages training on mTOP, comparing using Utterance or space-joined tokens as input text. In each case, the same format is used at both train and test time.

B.2.1 Space-joined Tokens for mTOP

As noted in section 4.1, the mTOP dataset[5] provides two options for the input: raw "Utterance", as well as "tokens", which according to the README file: "This is a JSON string representing the tokenization used for all experiments in the paper." We opt for using the provided tokens JSON, and joining the tokens on spaces. This fixes many (although not all) spacing and other anomalies with exact match and token copying which occur in as much as 30% of utterances the non-English datasets. An example for French is shown in Figure 8.

We encourage the community to continue a deep dive into anomalies in the mTOP dataset, and develop a standard setting, perhaps even releasing a standardized / cleaned mTOP-v2. As it stands, we still consider mTOP a highly useful dataset to evaluate experiments within the same publication or research team, however comparisons across publications and groups should be taken with a grain of salt.

```
Utterance field in mTOP French:
  Donne-moi la liste des salons de l'automobile prévus à Atlanta le week-end prochain

Ground-truth parse:
  [IN:GET_EVENT [SL:CATEGORY_EVENT salons de l' automobile ] [SL:LOCATION Atlanta ]
    [SL:DATE_TIME le week – end prochain ] ]

Space-joined tokens field (our models use this version):
  Donne – moi la liste des salons de l' automobile prévus à Atlanta le week – end prochain
```

Figure 8: Comparing "space-joined tokens" input versus "Utterance" input format for mTOP. As shown, the "space-joined-tokens" resolves various spacing anomalies which improves cross-lingual zero-shot performance.

B.2.2 Sentinel Words for mTOP

Following Raman et al. (2022), we use "sentinel words" which we show greatly improves the cross-lingual zero-shot performance. An example is shown in Figure 9.

As noted in section B.2, we use *Space-joined Tokens* as input, which resolves many spacing anomalies occurring in the ground-truth annotation for a large portion (up to 30% of non-English) of the data. Still, approximately 3% of the non-English data has unresolved spacing and casing anomalies (see also, Appendix C). In those cases, we simply discard the original training utterances which cannot be converted into sentinel form. When an unresolved spacing or casing anomaly occurs in a test utterance, we do not discard the the utterance, but rather use a metric which makes it possible for the model to recover the correct answer (see Appendix C).

---

[5] https://fb.me/mtop_dataset

We do not add these sentinel words to the vocabulary, but rather simply allow the sentencepiece (Kudo and Richardson, 2018) tokenizer to split them into subwords, such as ['_word', '0']. We hypothesize that this could allow the model to generalize at inference time to inputs longer than those seen during training. However, this choice makes the input and output sequences longer than necessary, which could impact latency. In future work, we would like to explore adding the sentinel words to the vocabulary and measure this trade-off explicitly.

```
## English example ##
Original Text:
  are there thunder storms on the forecast this weekend

Original Parse:
  [IN:GET_WEATHER [SL:WEATHER_ATTRIBUTE thunder storms ] [SL:DATE_TIME this weekend ] ]
----------
Sentinel Words Text:
  word0 are word1 there word2 thunder word3 storms word4 on word5 the
    word6 forecast word7 this word8 weekend

Sentinel Words Parse:
  [IN:GET_WEATHER [SL:WEATHER_ATTRIBUTE word2 word3 ] [SL:DATE_TIME word7 word8 ] ]
----------

## German example ##
Original Text:
  Sind für dieses Wochenende Gewitter vorhergesagt ?

Original Parse:
  [IN:GET_WEATHER [SL:WEATHER_ATTRIBUTE Gewitter ] [SL:DATE_TIME für dieses Wochenende ] ]
----------
Sentinel Words Text:
  word0 Sind word1 für word2 dieses word3 Wochenende word4 Gewitter word5 vorhergesagt word6 ?

Sentinel Words Parse:
  [IN:GET_WEATHER [SL:WEATHER_ATTRIBUTE word4 ] [SL:DATE_TIME word1 word2 word3 ] ]
----------
```

Figure 9: An example of the input and output formats when using sentinel words.

C   Space- and Case-Insensitive Exact Match (SCIEM) Metric for mTOP

We define the variant of Exact Match we use for mTOP, which we call Space- and Case-Insensitive Exact Match (SCIEM). SCIEM is *insensitive to spacing and casing of text words in the parse* (excluding the parse elements such as the intent and slot names). Python code is provided in Figure 10 and an example is shown in Figure 11. **We encourage the research community to adopt these standard settings for mTOP: Space-joined Tokens as Input, and SCIEM metric.**

We compare results using Verbatim Exact Match vs. SCIEM, with greedy decoding ("Greedy"), in Table 4. As show in the table, SCIEM provides a small boost in performance on the non-English languages, of 0.5 points on the lower bound "en-only" (from 44.5 to 45.0), 0.9 points on the upper bound "ALL" (from 72.4 to 73.3), 0.7 points on our baseline method "MT-Opus" (from 59.5 to 60.2), and 0.8 points on our best-performing combination of methods "Our Best" (from 65.4 to 66.2).

Note, however, that the difference is unequal across languages, e.g. in the "en-only" setting, switching from Verbatim Exact Match to SCIEM improves French ("fr") by 1.1 points (from 53.1 to 54.2) however does not impact Hindi ("hi") at all. Finally, SCIEM has minimal impact on "en" results, with "ALL" improving by 0.2 points (from 83.1 to 83.3) and the other settings matching exactly.

These trends match with our observations in Appendices B.2.1 and B.2.2, that even after using space-joined tokens and sentinel words for the input, there remain a small number of spacing and casing anomalies, some of which are resolved by using the SCIEM metric.

D   Impact of Test-Time Decoding Strategy

In Table 4 (Appendix C), we also compare the impact of our choice of Decoding Strategy. As show in the Table, across settings Beam4 provides only a small boost over Greedy decoding, between 0.1 and 0.3 points on "avg-0s", and either exactly the same or 0.1 points improvement on "en".

```python
def get_sciem_key(model_output):
    pieces = model_output.strip().split()
    new_pieces = []
    for piece in pieces:
        if piece.startswith('[IN:') or piece.startswith('[SL:') or piece == ']':
            new_pieces.append(piece)
        else:
            new_pieces.append(piece.lower())
    return ''.join(new_pieces)

>>> model_output = "[IN:GET_WEATHER [SL:DATE_TIME para el Domingo de Pascua a las 14 : 00] ]"
>>> get_sciem_key(model_output)
'[IN:GET_WEATHER[SL:DATE_TIMEparaeldomingodepascuaalas14:00]]'
```

Figure 10: Python code for SCIEM metric.

```
Example from mTOP Spanish
----------
Utterance Input:
  Dime el pronóstico para el Domingo de Pascua a las 14:00.

Space-joined Tokens Input:
  Di me el pronóstico para el Domingo de Pascua a las 14 : 00 .
----------
Model hypothesis when using Utterance:
  [IN:GET_WEATHER [SL:DATE_TIME para el Domingo de Pascua a las 14:00 ] ]

Model hypothesis when using Space-joined Tokens:
  [IN:GET_WEATHER [SL:DATE_TIME para el Domingo de Pascua a las 14 : 00 ] ]

Ground-truth Parse Original:
  [IN:GET_WEATHER [SL:DATE_TIME para el domingo de Pascua a las 14 : 00 ] ]
----------
Model Hypothesis (in both cases) For Space- and Case-Insensitive Exact Match (SCIEM):
  [IN:GET_WEATHER[SL:DATE_TIMEparaeldomingodepascuaalas14:00]]

Ground-truth Parse For Space- and Case-Insensitive Exact Match (SCIEM):
  [IN:GET_WEATHER[SL:DATE_TIMEparaeldomingodepascuaalas14:00]]
----------
Verbatim Exact Match? NO
SCIEM Exact Match? YES
```

Figure 11: An example of Space- and Case-Insensitive Exact Match (SCIEM). The original *Utterance* input has both a spacing ("14:00" vs. "14 : 00") and a casing ("Domingo" vs. "domingo") anomaly compared to the Ground-truth Parse. While using *Space-joined Tokens* as input solves the spacing issue, the casing issue remains. In both cases, SCIEM corrects for the anomalies in the test set by counting the model's hypothesis as correct.

Data	Decoding	Exact Match Type	en	de	es	fr	hi	avg-0s
	Greedy	Verbatim	83.1	46.9	50.0	53.1	27.9	44.5
en-only	Greedy	SCIEM	83.1	47.2	50.8	54.2	27.9	45.0
	Beam4	**SCIEM**	83.1	47.3	51.0	54.8	28.2	**45.3**
	Greedy	Verbatim	83.1	69.7	75.7	73.8	70.5	72.4
ALL	Greedy	SCIEM	83.3	70.2	77.0	75.6	70.5	73.3
	Beam4	**SCIEM**	83.3	70.3	77.3	75.8	70.5	**73.5**
	Greedy	Verbatim	82.9	63.2	63.9	63.5	47.3	59.5
MT-Opus	Greedy	SCIEM	82.9	63.5	64.9	64.9	47.3	60.2
	Beam4	**SCIEM**	83.0	63.8	65.0	65.1	47.4	**60.3**
(Our Best) CLASP-{TS,TB}	Greedy	Verbatim	84.4	66.1	66.7	70.8	57.9	65.4
+MT-20B +MT-Opus	Greedy	SCIEM	84.4	66.5	67.9	72.4	57.9	66.2
	Beam4	**SCIEM**	84.4	66.7	68.1	72.6	58.1	**66.4**

Table 4: The impact of SCIEM (vs. Verbatim Exact Match) and Beam4 decoding (vs. Greedy decoding) on lower bound ("en-only"), upper bound ("ALL"), baseline ("MT-Opus"), and our best-performing ("Our Best") combination of methods.

E Impact of Adding Grammar-Generated Train Data for PIZZA

For PIZZA, we show the impact on tuning the amount of grammar-generated training data, as described in Section 4.2. As show in Figure 12, the best-performing option for train (m) in isolation is m=69,600, and when mixed with dev (n=16) + train (m), m=104,400 is best. These correspond to the rows "train-only"

456

and "dev+train", respectively, in table 1. Note, as described in Section 4.1, to avoid overfitting on the test set which contains only 1,357 utterances, we extract a 10% subset of the test set, referred to as the "validation" set to use for hyperparameter tuning and early stopping.
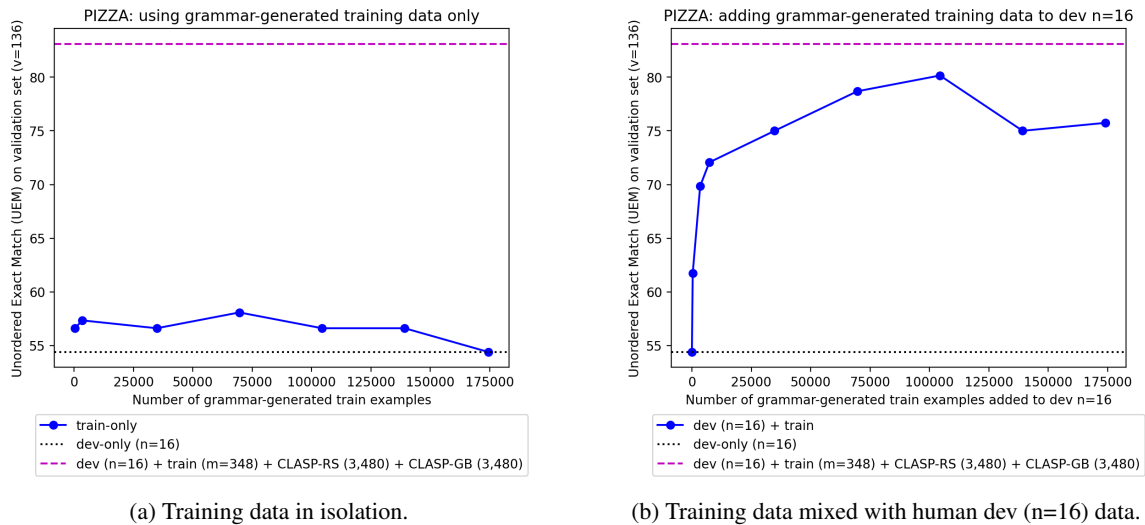

(a) Training data in isolation.

(b) Training data mixed with human dev (n=16) data.

Figure 12: Learning Curve of increasing amount of (grammar-generated) training data for PIZZA. Left (a) in isolation; Right (b) mixed with (human-curated) dev n=16.

F   Hyperparameters

We fine-tune with Adam (Kingma and Ba, 2015) using a learning rate $1e-5$, dropout 0.1, and batch size 128. We fix the number of update steps to u=2,500 (1,000 epochs for dev n=348 or 20,000 epochs for dev n=16) for PIZZA, and u=12,000 (100 epochs) for mTOP. Fine-tuning takes takes one hour for PIZZA and four hours for mTOP on an AWS p3.24xlarge instance, using DeepSpeed ZeRO (Rajbhandari et al., 2019) Stage 1 to save GPU memory and speed up training. Our models are built on top of HuggingFace (Wolf et al., 2020).

When generating data with AlexaTM 20B, we use either sampling or greedy decoding, described in Appendix H.

G   mTOP Utterances Used for Prompting

The utterances we use for all mTOP in-context generation prompts are shown in Figure 13.

Figure 13: The one-shot examples from mTOP which we use for all in-context prompts.

H Filtering CLASP Outputs

Our filtering logic starts from the following two Validation Principles: *VP1 (Valid Parse)*: the parse must be valid according to the task format and the specific instructions contained in the generation prompt (e.g. including a particular slot); *VP2 (All Slots Present)*: each slot value in the parse must appear in the sentence text.

H.1 Filtering CLASP Outputs for PIZZA

For **PIZZA**, we generate 4 outputs with sampling[6] (settings: $top\_k = 50$ (Fan et al., 2018), $top\_p = 0.9$ (Holtzman et al., 2020), and $temperature = 0.9$), discard any which are invalid according to certain heuristic Failure Modes (described below), then select the remaining one with lowest perplexity. In cases where there is no acceptable output utterance, we duplicate an utterance from the prompt back into the training set to maintain the per-class distribution.

We define the **Success Rate (Inputs) as the percentage of input prompts which result in at least one valid output**. In early experiments, we used the Success Rate (Inputs) metric to iterate on settings such as the the number of input examples, the prompt format, and the sampling hyperparameters. Our final settings produce a Success Rate (Inputs) of **81.1% for CLASP-RS** (Replace Slots then Generate Text; Section 3.1) and **77.6% for CLASP-GB** (Generate Both Parse and Text; Section 3.3) (Table 5).

The lower Success Rate (Inputs) for CLASP-GB reflects the greater degree of ambiguity for this CLASP method, as the model must *generate both* the the parse and text. We also measure the Success Rate (Outputs) as the percentage of all *outputs* which are valid, and see a similar trend.

We identify a total of seven common Failure Modes, which are (non-mutually exclusive) criteria for discarding a generated utterance. The occurrence rate for each is shown in Table 5, where the denominator is the total number of *outputs* produced.

The most common Failure Mode is "Missing Slot", where the output is missing one of the requested slot values, occurring 25.8%/30.0% of the time for CLASP-RS/CLASP-GB. "Untagged Slot" occurs when a slot word from the catalog, such as "pepperoni" appears in the outputs, but is not tagged in any slot, occurring for 1.6%/7.1% of outputs. Invalid Separators (semicolon or arrow "=>" is missing from or mis-placed or duplicated in the output) occurs for 0.1%/2.1% of outputs. 3.4%/0.8% of the outputs are discarded due to copying an input example.

We discard *Duplicate Outputs*, occurring for 39.3% of the CLASP-RS and 3.6% of the CLASP-GB outputs, respectively. The higher (lower) portion of duplicates for CLASP-RS (CLASP-GB) reflects how the method is more (less) constrained, resulting the model's ability to produce less (more) diverse outputs.

Finally, for CLASP-GB, we discard outputs which have an Invalid Parse or Unk. (Unknown) Entity according to the catalog. The **Invalid Parse percentage is remarkably low, just 0.9%, suggesting that the CLASP-GB method is effective at teaching the LLM to produce valid Semantic Parsing training data from very few examples**.

The Unknown Entity portion of 6.3% may represent an opportunity to expand the catalog, either automatically or via a human annotation pipeline. For example, in one case the model produced "lemonade" as a Drinktype, which is reasonable, however was discarded since it does not appear in the slot catalogs.

Future work can discover more failure modes to filter out, and explore methods to improve the quality of outputs so that less filtering is required.

CLASP Method	Success Rate (Inputs)	Success Rate (Outputs)	Failure Modes						
			Missing Slot	Untagged Slot	Invalid Separators	Copy Example	Duplicate Output	Invalid Parse	Unk. Entity
CLASP-RS	**81.1**	66.2	25.8	1.6	0.1	3.4	39.3	–	–
CLASP-GB	**77.6**	34.9	30.0	7.1	2.1	0.8	3.6	0.9	6.3

Table 5: Success rate (percentage) and occurrence of Failure Modes (percentage) when generating data for PIZZA using the CLASP methods, CLASP-RS and CLASP-GB. The Success rate (Inputs) for each line is bolded.

---

[6]We refer the reader to this guide: https://huggingface.co/blog/how-to-generate .

H.2 Filtering CLASP Outputs for mTOP

For **mTOP**, we use greedy search which returns only one output per input prompt. Then, similar to our setup for PIZZA, we discard outputs which exhibit one or more Failure Modes (described below), and when there is no acceptable output utterance, we duplicate an utterance from the prompt back into the training set to maintain the per-class distribution.

We define **Success Rate as the percentage of inputs which result in a valid output** after filtering. As show in Table 6, the overall Success Rate (averaged across the four non-English languages) is **87.9% for CLASP-TS** (Translate Slots then Generate Parse, Section 3.2) and **76.3% for CLASP-TB** (Translate both Parse and Text, Section 3.4). We further analyze the Success Rate by three Success Modes: "Clean" (77.3%/64.4% for CLASP-TS/CLASP-TB) where no post-processing is needed, and two heuristic recovery methods, "Slot n-best" and "Fix Casing", described in the next section.

Given that CLASP-TB is more challenging (the model must generate not only the text but also the parse), it is not surprising to find that the Success Rate is lower for this method compared to the CLASP-TS. However, as show in Section 5.2, the two methods provide similar downstream performance. This suggests that although CLASP-TB provides a smaller volume of viable data than CLASP-TS, the data from CLASP-TB is of higher quality (perhaps due to avoiding the noise of translating slots a priori).

The most common Failure Mode is "Missing Slot", described above for PIZZA in Appendix H.1. While the model rarely copies an input example verbatim, Invalid Separators (=> and semicolon) occur for 12.4% of for Hindi outputs, discussed in more detail in Appendix I.

Finally, while the model rarely outputs invalid parses, we observe a high rate of the "Mismatch Parse" failure mode, where the output parse does not match the input example structure.[7] We find the majority of these occur when the model copies part of one of the input examples, as show in Figure 14. In early experiments, we found that adding more examples to the prompt exacerbated this problem, so we decided to always use just one example.

Future work can explore how to reduce the occurrence of these failure modes to extract even more performance boost from CLASP.

CLASP Method	Language	Success Rate	Success Modes			Failure Modes				
			Clean	Slot n-best	Fix Casing	Missing Slot	Copy Example	Invalid Separators	Invalid Parse	Mismatch Parse
TS	de	84.4	72.8	8.7	2.9	15.4	0.2	–	–	–
	es	86.5	76.0	6.7	3.7	13.0	0.5	–	–	–
	fr	90.4	78.8	7.2	4.4	9.4	0.2	–	–	–
	hi	90.3	81.6	8.7	0.0	9.7	0.0	–	–	–
	avg	**87.9**	77.3	7.8	2.8	–	–	–	–	–
TB	de	78.8	70.9	1.8	6.2	11.7	0.6	0.6	0.9	7.3
	es	82.2	61.5	18.3	2.4	14.3	1.6	0.7	0.0	1.1
	fr	76.7	62.6	13.0	1.2	8.0	1.1	1.0	0.5	12.6
	hi	67.5	62.8	4.6	0.1	6.0	0.1	12.4	1.6	12.4
	avg	**76.3**	64.4	9.4	2.5	–	–	–	–	–

Table 6: Success Rate and occurrence of various Success Modes and Failure Modes when generating data for mTOP using the CLASP methods, CLASP-TS and CLASP-TB. All numbers represent percentage of occurrence. The average across the four languages for each CLASP method is bolded.

H.2.1 Slot N-Best and Casing Recovery for mTOP

There is inherent ambiguity of word choice in cross-lingual data generation. When a slot has a different form in the parse vs. in the text, the example is considered invalid (VP2, above), and would need to be discarded. However, we identify two modes, "Slot n-best" and "Fix Casing", where it is possible to recover from this mismatch by simply replacing the slot value in the parse with a readily available alternative.

For "**Slot n-best**", we *a priori* create an n-best list of all slot translations, using an in-context prompt with AlexaTM 20B (see Figure 15) and beam search 4 outputs. Then, as show in Figure 16, if we find that

---

[7]Note for mTOP, our goal is not to generate novel parse structures, but rather to create a parallel dataset from English to the other languages.

```
INPUT:
[CLM] Semantic Parse for English:
  [IN:CREATE_REMINDER [SL:PERSON_REMINDED me ] [SL:TODO
    [IN:GET_TODO [SL:DATE_TIME 10 : 00 am ] [SL:TODO doctor 's appointment ] ] ] ]
=> Translation in English:
  Remind me of my 10 : 00 am doctor 's appointment;
Semantic Parse for French:
  [IN:CREATE_REMINDER [SL:PERSON_REMINDED moi ] [SL:TODO
    [IN:GET_TODO [SL:DATE_TIME de 10 h ] [SL:TODO rendez - vous chez le médecin ] ] ] ]
=> Translation in French:
  Fais - moi penser à mon rendez - vous de 10 h chez le médecin;
Semantic Parse for English:
  [IN:SET_RSVP_NO ]
=> Translation in English:
  RSVP no to this event;
Semantic Parse for French:

OUTPUT:
[IN:SET_RSVP_NO
  [SL:PERSON_REMINDED moi ] [SL:TODO [IN:GET_TODO [SL:DATE_TIME de 10 h ]
  [SL:TODO rendez - vous chez le médecin ] ] ] ]
=> Translation in French: Fais - moi penser à mon rendez - vous de 10 h chez le médecin;
```

Figure 14: Example of Failure Mode Mismatch Parse for CLASP-TB. While the output parse is technically valid according to the mTOP specification, it does not match the requested parse format. In particular, in this case, it copies part of the other example's prompt verbatim.

a slot is missing from the text, we check for the presence of another version of the slot from the n-best list, and if found, update the parse with the new value, and accept the generated training example. As show in Table 6, this allows us to recover 7.8%/9.4% of Success Rate for CLASP-TS/CLASP-TB.

Similarly, for "**Fix Casing**" (see Figure 17) if we find that a slot is missing from the text, we check for a case-insensitive match in the text, and if found, replace the slot in the parse. This allows us to recover 2.8%/2.5% of Success Rate for CLASP-TS/CLASP-TB (Table 6).

```
INPUT:
[CLM] Slot: me; Translation in Spanish: me;
Slot: 30 minutes before; Translation in Spanish: 30 minutos antes;
Slot: 2 : 00; Translation in Spanish: 14 : 00;
Slot: appointment; Translation in Spanish: cita;
Slot: all; Translation in Spanish:

OUTPUTS:

0: todo;
1: todos;
2: todas;
3: todos los;
```

Figure 15: An example of in-context Slot text translation from English to Spanish.

I   Filtering Machine Translation Outputs

For mTOP Machine Translation experiments (either using Opus or the 20B LLM, described in Section 4.2), we filter the outputs using heuristics to avoid noisy alignments.[8]

We first apply Sim-Align (Jalili Sabet et al., 2020) to align the translated sentence back to the original English source, in order to compute the parse in the target language. We discard outputs which exhibit any of four Failure Modes. The first two Failure Modes are related to slots: (i) Missing Slot Value (Figure 18); or (ii) Discontiguous Target (Figure 19). We also discard outputs which: (iii) Copy the Original input text verbatim, and in the case of translation with the 20B model, (iv) contain the word "Sentence", i.e. fail to end with a semicolon as prompted (Figure 20).

We define the "Success Rate" as the percentage of remaining outputs after filtering. As show in Table 7, the success rate is far from 100%, e.g. for Opus MT varying from 86.7 for German ("de") down to 62.2 for Hindi ("hi"). This reflects the *difficulty of the alignment task, a fundamental limitation of the*

---

[8]Early experiments showed these filtering mechanisms to provide significant improvement over using the alignment as-is. Future work can continue to explore cleaning and filtering methods for MT alignment.

```
INPUT:
[CLM] Semantic Parse:
  [IN:CREATE_REMINDER [SL:PERSON_REMINDED me ] [SL:DATE_TIME 30 minutes before ]
    [SL:TODO [IN:GET_TODO [SL:DATE_TIME 2 : 00 ] [SL:TODO appointment ] ] ] ];
Translation in English:
  It would be great if you could remind me 30 minutes before my 2 : 00 appointment .;
Semantic Parse:
  [IN:CREATE_REMINDER [SL:PERSON_REMINDED me ] [SL:DATE_TIME 30 minutos antes ]
    [SL:TODO [IN:GET_TODO [SL:DATE_TIME 14 : 00 ] [SL:TODO cita ] ] ] ];
Translation in Spanish:
  Sería genial que me recordaras 30 minutos antes de mi cita de las 14 : 00 .;
Semantic Parse:
  [IN:GET_ALARM [SL:AMOUNT all ] [SL:DATE_TIME for Friday ] ];
Translation in English:
  I want to see all alarms for Friday;
Semantic Parse:
  [IN:GET_ALARM [SL:AMOUNT todo ] [SL:DATE_TIME viernes ] ];
Translation in Spanish:

OUTPUT:
Quiero ver todas las alarmas para el viernes.;

SLOT N-BEST RECOVERY:
INFO: Recovered parse for slot n-best match: 'all': ['todo', 'todos', 'todas', 'todos los']
INFO: src_parse: [IN:GET_ALARM [SL:AMOUNT all ] [SL:DATE_TIME for Friday ] ]
INFO: old_parse: [IN:GET_ALARM [SL:AMOUNT todo ] [SL:DATE_TIME viernes ] ]
INFO: new_parse: [IN:GET_ALARM [SL:AMOUNT todas ] [SL:DATE_TIME viernes ] ]
```

Figure 16: Example of Success Mode "Slot n-best" for CLASP-TS. The sentence generated by the model uses a different word for the slot "all" than was set during *a priori* slot translation. (Here, the feminine plural form "todas" instead of the masculine singular form "todo".) Instead of discarding this example with Missing Slot failure mode, we can use our pre-computed n-best slot mapping to recover a version of the target-language parse which matches the words in the model's output.

```
INPUT:
[CLM] Semantic Parse for English:
  [IN:CREATE_REMINDER [SL:PERSON_REMINDED me ] [SL:TODO
    [IN:GET_TODO [SL:TODO check the weather ] [SL:DATE_TIME Friday ]
    [SL:TODO see if the cookout 's still on ] ] ] ]
=> Translation in English:
  Remind me to check the weather Friday to see if the cookout 's still on .;
Semantic Parse for German:
  [IN:CREATE_REMINDER [SL:PERSON_REMINDED mich ] [SL:TODO
    [IN:GET_TODO [SL:TODO das Wetter überprüfen ] [SL:DATE_TIME am Freitag ]
    [SL:TODO sehen ob die Grillparty noch stattfindet ] ] ] ]
=> Translation in German:
  Erinnere mich am Freitag das Wetter zu überprüfen um zu sehen ob die Grillparty noch stattfindet .;
Semantic Parse for English:
  [IN:UPDATE_CALL [SL:CONTACT_ADDED nicole ] ]
=> Translation in English:
  join nicole to this call;
Semantic Parse for German:

OUTPUT:
[IN:UPDATE_CALL [SL:CONTACT_ADDED nicole ] ]
=> Translation in German: Nicole zu diesem Anruf hinzufügen;

FIX CASING RECOVERY:
INFO: Recovered parse via Fix Casing: changed 'nicole' to 'Nicole'
INFO: old_parse: [IN:UPDATE_CALL [SL:CONTACT_ADDED nicole ] ]
INFO: new_parse: [IN:UPDATE_CALL [SL:CONTACT_ADDED Nicole ] ]
```

Figure 17: Example of Success Mode "Fix Casing" for CLASP-TB. The model generates both the parse and text, however the casing for the slot 'Nicole' does not match. Instead of discarding this example as Missing Slot failure mode, we recover the correct parse by finding a case-insensitive match for the slot in the text, and updating the parse to match.

baseline approach of Machine Translation with slot alignment, particularly between distant language pairs such as English and Hindi.

Also of note, when using the 20B model for translation, 13.4% of the prompts for Hindi were discarded due to producing the word "Sentence", i.e. not ending with a semicolon as instructed. (See an example in Figure 20, compared to Figure 6.) We hypothesize this could be caused by using a semicolon as the

461

separator, which might be less common in Hindi than the other languages which use the Latin alphabet. Future work could explore using language-agnostic separators such as `<br>`.

MT Model	Language	Success Rate	Failure Modes			
			Missing Slot Value	Discontiguous Target	Copy Original	Contains "Sentence"
Opus	de	86.7	4.6	8.6	0.1	–
	es	74.2	4.9	20.8	0.2	–
	fr	82.3	3.9	13.7	0.1	–
	hi	<u>62.2</u>	19.4	18.4	0.0	–
	avg	**76.4**	–	–	–	–
20B	de	85.5	4.2	9.8	0.3	0.1
	es	70.9	6.6	21.5	0.6	0.3
	fr	77.4	4.7	17.2	0.1	0.5
	hi	<u>58.3</u>	12.2	16.0	0.1	13.4
	avg	**73.0**	–	–	–	–

Table 7: Success Rate (percentage) and occurrence of failure cases (percentage) of Machine Translation (MT) with with alignment across MT models and languages. The average across the four languages is bolded, and the language with lowest (i.e., worst) Success Rate for each model is underlined.
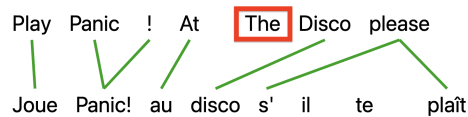

Figure 18: Example of a translation alignment discarded due to "Missing Slot Value", where a source-side slot word ("The") is not aligned to any output word. The parse for the English utterance is `[IN:PLAY_MUSIC [SL:MUSIC_ARTIST_NAME Panic !  At The Disco ] ]`. (Via https://simalign.cis.lmu.de/)

Figure 19: Example of a translation alignment discarded due to "Discontiguous Target", where a source-side slot ("playlist") aligns to a discontiguous set of words in the target ("liste" and "lecture", missing "de"). The parse for the English utterance is `[IN:DELETE_PLAYLIST_MUSIC [SL:MUSIC_TYPE playlist ] ]`. (Via https://simalign.cis.lmu.de/)

Figure 20: Example of a translation output from the 20B model, discarded due to Contains "Sentence".

J Sim-Align Settings

We explore four settings for Sim-Align, using either (multilingual) "bert" (Devlin et al., 2019) or "xlm-roberta-base" (Conneau et al., 2020) each with either "ArgMax" or "IterMax" as the alignment method. *We choose "bert" with with "IterMax"* as we find it has the highest Success Rate (defined in Appendix I).