An Empirical Analysis of Leveraging Knowledge for Low-Resource Task-Oriented Semantic Parsing

Mayank Kulkarni[1], **Aoxiao Zhong**[2*], **Nicolas Guenon des mesnards**[1], **Sahar Movaghati**[1], **Mukund Sridhar**[1], **He Xie**[1], **Jianhua Lu**[1]
[1]Amazon Alexa AI [2]Harvard University
{maykul,mesnarn,movas,harakere,hexie,jianhual}@amazon.com
aoxiaozhong@g.harvard.edu

Abstract

Task-oriented semantic parsing has drawn a lot of interest from the NLP community, and especially the voice assistant use-cases as it enables representing the meaning of user requests with arbitrarily nested semantics, including multiple intents and compound entities. SOTA models are large seq2seq transformers and require hundreds of thousands of annotated examples to be trained. However annotating such data to bootstrap new domains or languages is expensive and error-prone, especially for requests made of nested semantics. In addition large models easily break the tight latency constraints imposed in a user-facing production environment. As part of this work we explore leveraging external knowledge as a replacement for additional annotated data in order to improve model accuracy in low-resource and low-compute settings. We demonstrate that using knowledge-enhanced encoders inside seq2seq models does not result in performance gains by itself, but multitask learning to uncover entities in addition to the parse generation is a simple yet effective way of improving performance across the domains and data regimes. We show this is especially true in the low-compute low-data setting and for entity-rich domains, with relative gains up to 74.48% in some cases on the TOPv2 dataset.

1 Introduction

Fostered by NLP advances, virtual assistants such as Google Home or Alexa are becoming increasingly competent to address complex yet natural, everyday user needs. While requests as simple as "turn off the living room lights when the movie starts" could not be fulfilled with legacy systems that assigned a single user intent to each utterance and a single slot label to each token in an utterance (Mesnil et al., 2013; Liu and Lane, 2016), recent works on task-oriented semantic parsing (Gupta et al., 2018; Aghajanyan et al., 2020) represent utterance semantics with arbitrarily nested trees (Figure 1), thus handling the above use-case among others (e.g. multiple intents, cross-domain intents, compound entities, etc.). The research community tackles this task with success by treating it as a seq2seq generation task where a linearized semantic tree is predicted iteratively (Rongali et al., 2020), but such approaches fall short when confronted by real-life constraints such as strict run-time latency and scarcity of quality training data. Manual data annotation of training examples is a costly and error-prone process, which is exacerbated as utterance target representations become richer (more nested). The impact of data scarcity has been quantified in recent years with the introduction of the TOPv2 benchmark (Chen et al., 2020) that provides low-resource scenarios for task-oriented parsing.

Popular approaches to overcome data scarcity include synthetic data augmentation (Feng et al., 2021; Jia and Liang, 2016; Schick and Schütze, 2021), transfer learning (Ruder et al., 2019; Fan et al., 2017), and meta-learning (Gu et al., 2018; Huang et al., 2018; Wang et al., 2020). In this paper, we explore if we can model richer token representations for mentions by leveraging external knowledge, as mentions are fundamental to generating the correct parse. The backbone motivation lies in the observation that several everyday NLP applications involve real-life entities referenced in knowledge bases (for e.g. street names, sports events, or public figures). This information can be utilized for enhancing downstream NLP tasks. For example the request "play the *green line*" could refer to either a movie name or a song name, modeling this mention appropriately for the decoder could improve performance while generating a parse. This is particularly appealing in the low-data regime, for which rare entities are unlikely to be represented in the training data at all. Additionally, building entity embeddings through entity-focused modeling

[*] The work was done while at Amazon Alexa AI.

Findings of the Association for Computational Linguistics: ACL 2023, pages 1955–1969
July 9-14, 2023 ©2023 Association for Computational Linguistics

objectives has shown promising results in entity based NLP tasks such as named entity recognition (Yamada et al., 2020) and entity linking (Wu et al., 2020).

While there has been prior work to leverage knowledge for generation tasks (Guu et al., 2020; Izacard et al., 2022; Cao et al., 2020) this largely focused on unstructured text generation tasks such as Question-Answering or Entity Linking. To the best of our knowledge, we are the first to investigate its use in seq2seq models for task-oriented semantic parsing, a complex and structured text generation task.

We present an empirical analysis of using knowledge to improve accuracy of semantic parsing models, with a special focus on low-latency models such as small-decoder seq2seq models and non-auto-regressive models like RINE (Mansimov and Zhang, 2022). Our contributions are as follow:

- We benchmark three popular Knowledge-Enhanced encoders inside seq2seq models and show this way of leveraging knowledge does not consistently improve accuracy in the low-data regimes for task-oriented semantic parsing generation. However when reformulated as a classification task we see promising results with knowledge-enhanced encoders.

- We propose a joint training objective combining semantic parsing and mention detection as a simple and effective approach to leverage external knowledge and improve accuracy. We find up to 74.48% relative gains over baselines for low-data settings and entity-rich domains.

- We quantify the benefits of source training for regular, knowledge-enhanced and low-latency models, in gradually increasing low-data scenarios.

2 Related Work

Task-oriented Semantic Parsing Semantic parsing refers to the task of mapping natural language queries into machine-executable representations. Voice assistants typically transform a voice recording into text, that is further mapped to a backend exploitable representation containing the semantics of the request: the user intent, the invoked entities, relations between those entities, etc. Task-oriented parsing was popularized with the introduction of the TOP dataset (Gupta et al., 2018), and is usually treated as a seq2seq task where utterance tokens are copied into a semantic tree constructed auto-regressively (Rongali et al., 2020; Arkoudas et al., 2022). However such models are not always applicable in production environments with strict memory and latency constraints. This limitation is commonly addressed by reducing model sizes (Jiao et al., 2019; Kasai et al., 2020) and leveraging non-auto-regressive modeling (Gu et al., 2017; Zhu et al., 2020; Mansimov and Zhang, 2022).

Knowledge-Enhanced LMs Retrieval-based seq2seq models such as REALM (Guu et al., 2020) and ATLAS (Izacard et al., 2022) leverage factual knowledge from a corpus or knowledge-graph during training and inference, hence incur a considerable latency cost, despite attempts to make the retrieval more efficient (Wu et al., 2022). Given our low-latency setup, we focus on parametric knowledge that is learnt during the pre-training or fine-tuning process of large language models (LLMs), resulting in embeddings that do not require explicit knowledge retrieval at inference.

Knowledge-enhanced pretraining focuses on modeling entities: WKLM (Xiong et al., 2019) learns to determine if an entity was replaced with another entity of the same type in addition to Masked Language Modeling (MLM) and shows gains on downstream knowledge-intensive tasks such as Question-Answering (QA) and Relation Extraction (RE). LUKE (Yamada et al., 2020) explicitly models entity-embeddings through entity-embedding prediction during MLM and entity-entity self-attention layers during fine-tuning, with gains on Named Entity Recognition (NER), QA and RE. KBIR (Kulkarni et al., 2022) learns to reconstruct keyphrases in a combination and extension of WKLM and SpanBERT (Joshi et al., 2020), improving keyphrase extraction/generation tasks. Lastly, BLINK (Wu et al., 2020) learns entity-disambiguation by aligning entity surface forms to their descriptions resulting in rich entity embeddings. Work in the area of parametric knowledge-enhanced seq2seq models is limited to KeyBART (Kulkarni et al., 2022) for Keyphrase Generation and GENRE (Cao et al., 2020) for Entity Disambiguation.

3 Methods

We explore two complementary methods for leveraging knowledge: (1) fine-tuning knowledge-enhanced encoders for task-oriented semantic pars-

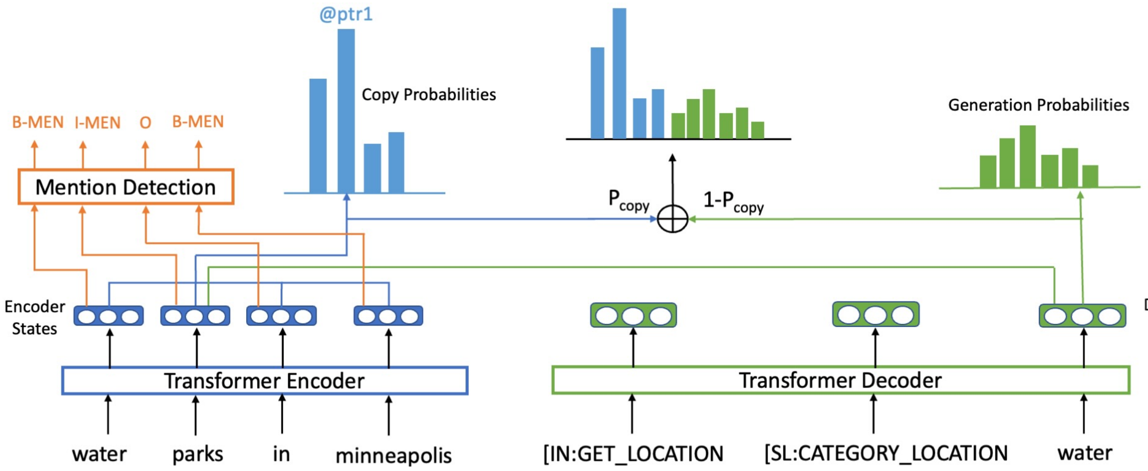

Figure 1: Multitask architecture that detects mentions of ‘water parks‘ (CATEGORY_LOCATION slot) and ‘minneapolis‘ (*Location* entity), and also generates the semantic parse by leveraging jointly learned encoder token embeddings.

ing inside seq2seq models, and (2) multi-tasking the parse generation with a mention detection task.

Task formulation We follow the task formulation of the Seq2Seq-PTR model as a sequence-to-sequence generation setup (Rongali et al., 2020). The source sequence is an utterance and the target sequence is a linearized representation of the semantic parse. The target sequence is modified to contain only intent and slot labels or pointers to tokens in the utterance. Following (Aghajanyan et al., 2020) and subsequent work we use the *decoupled* format that limits prediction to tokens that are leaves of slots[1] as it yielded better downstream performance in previous work. We illustrate the format used with an example from the TOPv2 dataset below:

```
Source: water parks in minneapolis
Target: [IN:GET_LOCATION
[SL:CATEGORY_LOCATION @ptr0 @ptr1 ]
[SL:LOCATION_MODIFIER @ptr3]]
```

Each $@ptr_i$ token here points to the i^{th} token in the source sequence. Here $@ptr_3$ corresponds to the word *minneapolis*.

Proposed Architecture Based on the observation that many slot-values present in our task are actual real-life entities, we hypothesize that learning more effective representations of these slot-values may result in generating more accurate semantic parses as mentions play a critical role in understanding the utterance. We use knowledge-enhanced pre-trained encoders (as described in Section 2) inside the Seq2Seq-PTR architecture used in Rongali et al. (2020), extended to multitask training of parse generation and training of the encoder to perform token classification (mention detection), as it aligns with classification-based pre-training of the encoder. We anticipate that the multitask training will allow the knowledge-enhanced encoder representations to be attended and leveraged more effectively by the decoder generating the parse. Further, by modeling mentions inherently present in the annotated data, this serves well for low-resource use cases since we maximize the potential to learn from the data available.

Figure 1 illustrates our proposed architecture, whereby for a given input utterance $[x_1, .., x_n]$ we obtain encoder representation $[e_1, ..., e_n]$, from which we jointly learn two tasks: a) Mention Detection and b) Parse Generation.

Mention Detection We frame this as a token classification task to identify spans corresponding to mentions using the BIO tagging schema. Given the input sequence containing two mention spans $[x_0, x_1]$ and $[x_3]$, the corresponding target labels are [B-MEN, I-MEN, O, B-MEN], where B represents the beginning of the span, I represents an intermediate label within the mention span and O represents a non-mention span token. We only use this coarse-grained single entity-type label (MEN) as this is not used for inference but rather only to guide learning better encoder representations to be used by the decoder. We use a cross-entropy loss to learn these model parameters:

[1] The remaining tokens do not contribute to the semantics beyond what is already captured in their parent intent node.

$$L_m = -\sum_{c=1}^{3} y_{o,c} \log(p_{o,c})$$

Parse Generation Given the first $t-1$ generated tokens, the decoder generates the token at step t as follows: the decoder first produces a hidden state d_t through a multi-layer, multi-head self-attention (MHA) on the encoder hidden states and the decoder states so far, in line with the transformer decoder from Vaswani et al. (2017). The hidden state d_t is fed into a dense layer to produce scores over the target vocabulary and weights are learnt using a reconstruction loss L_r.

As the loss scales are similar, we use an equally weighted joint loss combining the losses from both the task to update the model parameters.

$$L_\theta = L_r + L_m$$

4 Experimental Setup

Dataset We use a crowdsourced dataset called TOPv2 (Chen et al., 2020) for this empirical analysis. The dataset maps user queries to hierarchical representation as exemplified in Figure 1. The dataset contains 8 domains, such as Reminder (used to set alarms, reminders) and Navigation (used to get driving directions, traffic information). Some domains are more complex than others, by having larger catalogs and overall more nested semantics. TOPv2 is a relevant testbed for virtual assistant understanding models in low-data settings, as it comes with different data regimes called Samples Per Intent and Slot (SPIS), for example 10 SPIS which means that each intent and slot label is present in only 10 different annotations.

Mention Distribution We use the FLAIR (Akbik et al., 2019, 2018) NER model[2] to tag entities and then leverage BLINK[3] (Wu et al., 2020) to link entities to get their canonical surface form when available. Entity-type information is only used to facilitate linking. Table 1 shows the entity distribution across the various domains of the TOPv2 dataset. This leads us to pick the following domains for our analysis:

- **Event**, which has the highest percentage of utterances that contain entities, serving as an ideal candidate to test our hypothesis.
- **Navigation**, which has the second highest entity presence and happens to be the domain with the most complex semantics (deepest trees, large catalogs).
- **Reminder**, which has the second least number of entities per utterance. We consider this domain to evaluate the impact of our proposed method for entity-scarce domains[4].

Because FLAIR NER tagger is limited to identify only three types of entities: Organizations (ORG), Persons (PER) and Locations (LOC), we extend our entity set by using slot-values present in the TOPv2 annotations. We manually select slots labels that are close to real-life entity types, but which slot values might not be recognized by the NER tagger. We describe the slots used for each domain in Appendix A.2.

The updated mention distribution is illustrated in Table 2. We see that trends between domains stay relatively the same, however there are significantly more utterances now containing entities. Event and Navigation almost double the number of average entities present in their utterances: from 1.04 to 1.76 for Event, and 1.31 to 1.86 for Navigation. For Reminder it remains more or less the same as before (1.03 vs 1.07). Even by adding those slots there isn't a lot of salient information to be captured in the form of entities in Reminder.

Our experiments show that using a combination of the entities tagged by FLAIR NER + BLINK and those tagged by the slot-matching mechanism described in A.2, was more effective than using either of these methods independently. We consider the spans of the tagged entities as labels. In the case both systems flag overlapping spans of text, longer spans override the shorter spans in case of nested entities as shown in A.3.

Source Training A common scenario for deployed production systems that serve N domains is to scale to a new N+1th domain. We assume the existing N domains have longer established, larger datasets that we can use as training data to bootstrap the new domain, on which we want to fine-tune and perform evaluation.

Models Given our resource-constrained setting, all models we evaluate are *base* variants of the

[2] https://huggingface.co/flair/ner-english-large

[3] https://github.com/facebookresearch/BLINK

[4] The number of entities is small but not zero, as having zero would not be different from simple (non-multitask) training.

Domain	Alarm		Event		Messaging		Music		Navigation		Reminder		Timer		Weather	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Avg Entities (All Utt.)	0.00	0.00	*0.37*	*0.37*	0.16	0.16	0.06	0.05	**0.37**	**0.38**	0.04	0.03	0.00	0.00	0.21	0.20
Avg Entities (Utt. w/ entity)	1.00	1.00	*1.04*	*1.03*	1.08	1.09	1.01	1.01	**1.31**	**1.31**	1.03	1.03	1.00	1.00	1.05	1.04
% utterances w/ entities	0%	0%	**36%**	**36%**	15%	14%	6%	5%	*28%*	*29%*	4%	3%	0%	0%	20%	19%
Total Utterances	20,430	7,123	9,170	2,654	10,018	3,048	11,563	4,184	20,998	6,075	17,840	5,767	11,524	4,252	23,054	5,682

Table 1: Entity distributions (FLAIR NER and BLINK Entity Disambiguation) across domains in the TOPv2 dataset.

Domain	Event		Navigation		Reminder	
	Train	Test	Train	Test	Train	Test
Avg Entities (All Utt.)	**1.46**	**1.50**	*1.23*	*1.23*	0.72	0.70
Avg Entities (Utt. w/ entity)	*1.76*	*1.80*	**1.86**	**1.88**	1.07	1.06
% utterances w/ entities	**83%**	**83%**	*66%*	*66%*	67%	66%

Table 2: Updated mention distributions after manually adding some of domain's slot labels to valid entity types.

publicly available models, unless specified otherwise. We work with both seq2seq pre-trained transformer models and pre-trained transformer encoders stitched with a transformer decoder as done in Rongali et al. (2020). We primarily experiment with:

- BART: We use the pre-trained encoder-decoder BART-base[5] (Lewis et al., 2020) as our baseline for the sequence generation task.
- RoBERTa2BART: We use the RoBERTa-base[6](Liu et al., 2019) as the encoder and randomly initialize a six layer decoder in the same configuration as the BART-base decoder. This largely serves as a baseline to LUKE as a parametric non-knowledge-enhanced encoder i.e. a vanilla encoder.
- LUKE2BART: We use the LUKE-base[7] as the encoder and randomly initialize a six layer decoder in the same configuration as the BART-base decoder. LUKE[8] serves as our parametric knowledge-enhanced encoder in evaluations.

Lightweight Architecture Variants As we explore the computation constrained setting with limited latency budget, we also implemented our models using a Single Layer Decoder (SLD) while maintaining the same size encoder. We do this as the largest portion of the latency footprint comes from the passes through the decoder, since auto-regressive decoding requires token representation to travel all their way up to the decoder as many times as there are tokens to generate. As such we propose BART2SLD, RoBERTa2SLD, and LUKE2SLD variants with a randomly initialized single layer decoder. Another angle to latency reduction is to use non-auto-regressive modeling, such as RINE (Mansimov and Zhang, 2022), a RoBERTa-based approach that achieve state-of-the-art accuracy on low and high-resource TOP dataset while being 2-3.5 times faster than auto-regressive counterparts. In this work we experiment with *rine-roberta* (the original RINE model), and *rine-luke*, where we instead initialize the encoder model weights with the LUKE-base parameters.

Implementation Details We use HuggingFace Transformers (Wolf et al., 2020) for seq2seq modeling architecture to ensure reproducibility. We do not tokenize intent and slot tags, but instead learn embeddings from scratch. For all our experiments we use 8 V100 NVIDIA GPUs, with batch sizes of 32 per GPU with a gradient accumulation of 2 with FP16 enabled. Source training uses a learning rate of $1e^{-5}$ over 100 epochs and fine-tuning uses a learning rate of $8e^{-5}$ over 50 epochs. Both use the Adam optimizer (Kingma and Ba, 2015). We use beam search decoding with beam size 3, and a maximum generation length of 128.

Evaluation We report Exact Match (EM) accuracy score metrics in line with previous literature (Chen et al., 2020; Aghajanyan et al., 2020; Rongali et al., 2020). Exact match accuracy is the most important metric to report as it strictly penalizes any incorrectly generated intermediate tokens as the end-performance of a semantic parsing system would result in a failure even for partially correct answers.

5 Results

All our results are *source trained* + *fine-tuned*, unless specified otherwise. We perform 3 runs across each experiment setting and report average scores

[5] https://huggingface.co/facebook/bart-base

[6] https://huggingface.co/roberta-base

[7] https://huggingface.co/studio-ousia/luke-base

[8] It is directly comparable to ROBERTA in architecture and size since we use only the token embeddings, and not the entity-entity self-attention layers. For results including these too see Section 6.

and standard deviations. Our findings are as follows:

Knowledge-enhanced encoders don't improve generative semantic parsing Table 3 shows results for the six layer (full) decoder setting and Table 4 shows results for a single layer decoder. In both the Multitasking and Non-Multitask setting, we see that the best performing model across data-regimes and domains is **not** consistently the knowledge-enhanced encoder LUKE. In the full decoder setting, LUKE-encoder based models perform on par but no better than the vanilla RoBERTa-encoder based models. We also note that both these model underperform BART, but that the gap bridges as we add more training samples. In the light-decoder setting, we also see similar trends, however an interesting finding is that BART tends to underperform when compared to RoBERTa and LUKE, even in the full data setting. This could be attributed to the smaller encoder size for BART.

The above findings are contrary to expected performance improvements typically seen using knowledge-enhanced encoders for other entity-related tasks such as NER, RE and QA. We believe the reason for this is that the aforementioned tasks are all classification-based tasks that are able to leverage the entity representations in making decisions on class-types, but in contrast Task-Oriented Semantic Parsing is a complex generation task. Even though entities play a critical role, the entity representations are not able to effectively guide the from-scratch decoder. This problem is alleviated to a certain extent through the Multitask training that we hypothesize is able to jointly learn representations of entities that will guide the decoder, but these jointly learnt representations do not necessarily benefit from the knowledge-enhanced encoder. Further, the application of Source Training potentially wipes out any gains the knowledge-enhanced encoder had over the vanilla counterparts as they have seen sufficient data to negate the gains through knowledge-enhancements as discussed in Section 6.

However knowledge-enhanced encoders can bring gains when reformulating parsing as a classification task as shown in Table 5 with the RINE approach that inserts utterance tokens in a semantic tree by recursively predicting triplets $(label, start\ position, end\ position)$ until it predicts termination. We do not penalize misplaced non-semantic tokens in metric calculation. Recasting the generation task to a classification task serves to be more in-line with how LUKE was pre-trained. Further, we also do not require any form of source training in this setting. We observe that *rine-luke* outperforms *rine-roberta* in most scenarios for the two entity-rich domains, but not on the entity-poor domain Reminder.

Multitasking with mention detection is an efficient way to leverage knowledge and improves performance across the board on the two TOPv2 domains with strong entity presence (Navigation and Event), especially in the lightweight decoder setting (up to 74.48%, Table 4), but also non-negligible in the full decoder setting (up to 8.60%, Table 3). When trained in domains with a weak entity presence (Reminder) multitasking serves as noise in the loss and results in a worse performing model for both full (-31.14%) and lightweight decoder (-82.83%). We also observe minor regression on 10 SPIS in Event but not in other data regimes for the domain, leading us to believe this may be an aberration. We find that while for certain settings such as Navigation+Lightweight decoder trained w/ MT knowledge-enhanced encoders outperform their vanilla counterparts, this behavior is not consistent across domains and decoder settings. Hence while the gains through multitasking remain consistent throughout, KE encoders do not play a large role in these gains. However, we also find that in the full decoder setting in the Navigation domain, LUKE seems to benefit the most from the Multitasking across all data regimes albeit performing slightly worse than RoBERTa still. Finally we also observe that as more data is added to the training set, the effectiveness of the Multitask learning reduces drastically. We believe this helps demonstrates that Multitask learning is most effective in the lower-data regime by leveraging knowledge available in the data.

Source-training is essential as shown in Table 8 in which KE models on their own are not sufficient to reach reasonable accuracy, as is the case for BART and was reported in Chen et al. (2020). We show that source-training improves accuracy by up to 86.36% in full data regimes, with larger percentage gains for LUKE and RoBERTa when compared to BART, further demonstrating that Source Training is required to tune the encoders to the generation task as knowledge-enhanced pre-training

Data Regime	10 SPIS			25SPIS			50SPIS			Full Data		
Training	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv
Navigation												
bart	*50.28 ± 3.33*	**51.86 ± 1.11**	3.14%	*56.58 ± 3.12*	**58.07 ± 0.25**	2.63%	60.8 ± 2.82	**61.9 ± 0.56**	1.81%	83.7 ± 0.43	83.69 ± 0.1	-0.01%
roberta2bart	44.48 ± 2.46	46.77 ± 1.95	5.15%	53.21 ± 1.39	54.75 ± 1.35	2.89%	61.04 ± 0.96	*61.8 ± 1.68*	1.25%	83.95 ± 0.47	**84.62 ± 0.16**	**0.80%**
luke2bart	43.35 ± 1.83	47.08 ± 0.81	**8.60%**	53.11 ± 0.58	56.16 ± 1.61	**5.74%**	58.33 ± 2.57	60.81 ± 2.8	**4.25%**	83.39 ± 1	*83.96 ± 0.28*	0.68%
Event												
bart	*63.85 ± 1.17*	61.77 ± 0.14	-3.26%	67.39 ± 1.51	67.81 ± 0.66	0.62%	71.31 ± 0.51	*72.14 ± 0.58*	**1.16%**	83.71 ± 0.62	83.32 ± 0.44	-0.47%
roberta2bart	**65.12 ± 2.68**	61.29 ± 2.45	-5.88%	67.13 ± 0.85	**68.74 ± 0.47**	**2.40%**	71.06 ± 0.68	71.29 ± 0.38	0.32%	*84.29 ± 0.39*	83.9 ± 0.46	-0.46%
luke2bart	63.07 ± 3.83	61.09 ± 3.3	**-3.14%**	67.6 ± 0.97	*68.05 ± 0.3*	0.67%	**72.24 ± 0.89**	71.57 ± 0.78	-0.93%	84.05 ± 0.38	**84.39 ± 0.27**	**0.40%**
Reminder												
bart	52.29 ± 1.4	39.97 ± 1.61	**-23.56%**	62.23 ± 0.99	47.24 ± 2.74	-24.09%	68.04 ± 1.38	59.83 ± 1.08	**-12.07%**	*82.88 ± 0.38*	82.59 ± 0.25	-0.35%
roberta2bart	*54.5 ± 1.35*	37.53 ± 1.74	-31.14%	*65.16 ± 1.11*	50.04 ± 1.73	-23.20%	*69.28 ± 1.36*	59.71 ± 2.67	-13.81%	82.69 ± 0.11	82.64 ± 0.27	-0.06%
luke2bart	**54.69 ± 1.22**	40.33 ± 1.53	-26.26%	**66.12 ± 1.43**	52.52 ± 1	**-20.57%**	**70.39 ± 1.19**	61.55 ± 0.67	-12.56%	82.35 ± 0.11	**82.94 ± 0.18**	**0.72%**

Table 3: The impact of Multitask (MT) training on Exact Match (EM) performance across models and domains of the TOPv2 dataset in a Full Decoder setting. **Bold** is best performing and *Italic* is second best.

Data Regime	10 SPIS			25SPIS			50SPIS			Full Data		
Training	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv
Navigation												
bart2SLD	5.59 ± 1.74	5.88 ± 1.62	5.19%	16.88 ± 1.46	19.03 ± 0.75	12.74%	28.2 ± 5.57	27.54 ± 0.94	-2.34%	78.69 ± 0.35	78.13 ± 0.71	-0.71%
roberta2SLD	5.02 ± 1.64	*8.63 ± 3.16*	**71.91%**	14.03 ± 2.27	**24.48 ± 2.66**	**74.48%**	27.58 ± 1.09	*35.82 ± 5.63*	**29.88%**	80.44 ± 0.27	*80.9 ± 0.25*	0.57%
luke2SLD	6.76 ± 1.11	**9.13 ± 3.48**	35.06%	18.47 ± 0.76	*20.81 ± 3.55*	12.67%	30.74 ± 3.33	**39.42 ± 5.5**	28.24%	80.61 ± 0.5	**82.29 ± 0.63**	**2.08%**
Event												
bart2SLD	**24.84 ± 4.49**	*17.65 ± 1.37*	-28.95%	**42.17 ± 1.92**	*37.72 ± 4.34*	-10.55%	*56.92 ± 1.68*	54.39 ± 1.07	-4.44%	79.88 ± 0.65	80.13 ± 0.25	0.31%
roberta2SLD	12.93 ± 0.79	14.9 ± 0.33	**15.24%**	24.04 ± 0.9	30.09 ± 7.32	25.17%	41.26 ± 4.9	**57.13 ± 2.86**	**38.46%**	*81.15 ± 0.28*	**81.96 ± 0.64**	1.00%
luke2SLD	12.95 ± 5.88	13.62 ± 4.54	5.17%	22.74 ± 10.19	31.68 ± 5.74	39.31%	45.09 ± 9.12	54.28 ± 2.73	20.38%	80.04 ± 1.88	80.99 ± 0.28	**1.19%**
Reminder												
bart2SLD	27.54 ± 1.96	7.56 ± 3.64	**-72.55%**	43.72 ± 2.03	22.11 ± 3.57	**-49.43%**	57.27 ± 0.81	37.71 ± 1.39	**-34.15%**	76.31 ± 0.72	76.15 ± 0.71	-0.21%
roberta2SLD	*35.82 ± 2.74*	6.15 ± 2.51	-82.83%	*51.31 ± 4.46*	24.46 ± 5.65	-52.33%	*62.77 ± 2.25*	39 ± 1.36	-37.87%	79.85 ± 0.55	**80.07 ± 0.64**	**0.28%**
luke2SLD	**39.42 ± 3.96**	8.6 ± 2.73	-78.18%	**54.03 ± 1.08**	23.11 ± 2.55	-57.23%	**64.1 ± 1.58**	38.96 ± 5.04	-39.22%	80.03 ± 0.3	*80.04 ± 0.36*	0.01%

Table 4: The impact of Multitask (MT) training on Exact Match (EM) performance across models and domains of the TOPv2 dataset in the Light Decoder setting. **Bold** is best performing and *Italic* is second best.

Data Regime	10 SPIS	25 SPIS	50 SPIS	Full Data
Navigation				
rine-roberta	**37.63 ± 2.21**	55.33 ± 0.44	61.15 ± 1.11	80.01 ± 0.13
rine-luke	37.22 ± 0.82	**56.88 ± 1.91**	**62.85 ± 1.12**	**80.02 ± 0.36**
Event				
rine-roberta	26.91 ± 2.46	43.50 ± 0.41	**65.12 ± 2.48**	79.98 ± 4.87
rine-luke	**30.40 ± 3.42**	**46.82 ± 4.94**	64.98 ± 1.59	**82.97 ± 0.10**
Reminder				
rine-roberta	34.47 ± 2.90	**54.26 ± 1.38**	**64.63 ± 1.23**	**83.45 ± 0.61**
rine-luke	**34.79 ± 3.19**	52.54 ± 2.78	64.23 ± 1.12	83.20 ± 0.23

Table 5: RINE model EM using RoBERTa-base encoder (rine-roberta) and LUKE-base encoder (rine-luke).

is typically classification-based. We also find that source-training drastically improves performance especially in low-data regimes with gains of up to 1262.20%. However, as more training data is made available, the impact of Source Training also drops quickly. In the absence of further pretraining of KE models, source training is a required step, and can actually be viewed as pretraining step. We also explored if using a pre-trained decoder from BART-base helps in improving performance but found no significant gains hence skipped the results for brevity.

Data Regime	10 SPIS	25 SPIS
Navigation		
luke2bart	43.35 ± 1.83	**53.11 ± 0.58**
luke2bart + linked entities	**45.85 ± 2.35**	*52.91 ± 2.16*
luke2bart + unlinked entities	*44.91 ± 1.68*	51.75 ± 1.51
luke2bart + unlinked mentions	42.49 ± 3.52	51.72 ± 0.71
luke2bart + MHA	40.14 ± 1.54	50.04 ± 2.79

Table 6: Exact Match (EM) performance improvements and degradations in an effort to further augment the knowledge-encoder LUKE on the Navigation domain of TOPv2.

6 Case Study on Knowledge-enhanced encoders

To better understand the lack of performance boost by KE encoders we propose a deeper dive on using LUKE as well as two alternative KE encoders.

Further enhancements to LUKE only result in limited gains For our previous experiments we restrict to using only LUKE's token embeddings to make a fair comparison with RoBERTa. However the original LUKE encoder is armed with many more parameters, including the entity-entity self-attention that allows us to leverage richer entity embeddings. We explore using the entity em-

Data Regime	10 SPIS			25SPIS		
Training	w/o MT	w/ MT	rel improv	w/o MT	w/ MT	rel improv
Navigation						
roberta2bart	44.48 ± 2.46	*46.77 ± 1.95*	5.15%	53.21 ± 1.39	*54.75 ± 1.35*	2.89%
luke2bart	43.35 ± 1.83	**47.08 ± 0.81**	8.60%	53.11 ± 0.58	**56.16 ± 1.61**	5.74%
kbir2bart*	41.42 ± 0.91	43.21 ± 1.28	4.32%	51.29 ± 0.54	52.42 ± 0.63	2.20%
blink2bart*	33.08 ± 3.77	40.75 ± 0.84	**23.19%**	45.57 ± 2.28	50.16 ± 0.88	**10.07%**

Table 7: Exact Match (EM) performance by leveraging other knowledge-enhanced encoders on the Navigation domain of TOPv2. *Only large variants of models are available publicly.

Data Regime	10 SPIS			25SPIS			50SPIS		
Training	w/o ST	w/ ST	rel improv	w/o ST	w/ ST	rel improv	w/o ST	w/ ST	rel improv
Navigation									
bart	10.65	50.28	372.11%	40.25	56.58	40.57%	50.67	60.8	19.99%
roberta2bart	4.25	44.48	946.59%	24.3	53.21	118.97%	39.05	61.04	56.31%
luke2bart	6.12	43.35	608.33%	24.15	53.11	119.92%	37.55	58.33	55.34%
Events									
bart	7.27	63.85	778.27%	25.77	67.39	161.51%	50.9	71.31	40.10%
roberta2bart	4.86	65.12	1239.92%	10.32	67.13	550.48%	38.13	71.06	86.36%
luke2bart	4.63	63.07	1262.20%	13.53	67.6	399.63%	39.68	72.24	82.06%

Table 8: The impact of Source Training (ST) on Exact Match (EM) performance across models and domains of the TOPv2 dataset

beddings in various forms and methods as we report in Table 6. *luke2bart+linked entities* finds the corresponding entity representation from LUKE's entity vocab and concatenates the embedding to the token representation. We also explore the approach *luke2bart+unlinked entities* that does not rely on finding a match in LUKE's entity vocabulary but rather generates the entity embedding based only on the given surface form. While the two aformentioned approaches are run only on entities tagged by FLAIR NER and linked with BLINK, we also try *luke2bart+multitask entities*, where the setup is similar to *luke2bart+unlinked entities* but leverages a larger entity set, which is actually the entity set used for the Multitasking, and uses entity embeddings for each surface form. We find that *luke2bart+linked entities* is the most effective methodology for 10 SPIS (+2.5 EM), however gains are neutralized as data is added (-0.2 EM). *luke2bart+unlinked entities* serves as a slightly more resource efficient way of improving performance as it skips the need to link entities before using them (+1.56 EM). Most interestingly, in contrast to the multitask learning setup we find that only concatentating representations of the slot-values in *luke2bart+unlinked mentions* actually hurts model performance (-0.86 EM). We believe the reason for this is that without the jointly learnt embeddings a higher number of concatenations to token representations introduces more noise than useful information, especially in low-data settings where there is insufficient data to learn across many parameters. Lastly, along the same lines of having too many parameters to learn from too few data, we made the additional finding that in the pointer generator network used by the decoder, using Dot Product Attention (DPA) is more effective than Multi-Head Attention (MHA) as it contains fewer parameters to learn.

Other KE encoders than LUKE lead to similar conclusions We explore using other knowledge-enhanced encoders: KBIR and BLINK. KBIR is potentially better suited as it is pre-trained to exploit keyphrases, which are closer to slot-values than entities. However Table 7 shows that KBIR performance is worse than its LUKE and RoBERTa counterparts (-3.87 EM). Using BLINK as the pretrained encoder also results in sub-par performance (-6.33 EM). This further strengthens our claim that the knowledge-enhanced encoders do not automatically enhance model performance. However, we see that Multitasking still continues to largely benefit both these encoders too, with BLINK making the largest gains of up to 23.19%.

Any potential KE encoder gains are diluted by Source Training We further investigated if KE encoders could have had a larger impact with less source training, for e.g. over fewer training epochs. We plot training curves for all our settings as seen in Figure 2. Our main observation here is that in the multitask setting LUKE outperforms RoBERTa in the single layer decoder setups early in training. However, as we train over more steps, the performance from both models converge. Further, in all other settings LUKE shows no discernible edge over RoBERTa during Source Training.

7 Conclusion & Future Work

We presented an empirical analysis of how we can leverage external knowledge for task-oriented semantic parsing in the low-resource and low-compute settings, by conducting a rigorous set of experiments. We demonstrated that simply using a knowledge-enhanced encoder is not sufficient to improve performance over baselines for the complex task of sequence generation, but shows promising result when the task is reformulated as a classification task. We presented a multitask learning framework that leverages external knowledge and requires little to no extra data annotation, and demonstrated its effectiveness in the low-data and low-compute settings. Future work could probe the type of knowledge learned by this method, and attempt to apply it to other entity-rich tasks, across model architectures. It could also explore an in-depth error analysis of where knowledge-enhanced encoders fail in order to address these shortcomings. Further we could extend this work for retrieval-based seq2seq models to improve task-oriented semantic parsing.

Limitations

We concede that there are differences in the number of parameters between the BART models when compared to the RoBERTa and LUKE counterparts. However, as per our result discussions and observations, the gains are orthogonal to the encoder used and the differences in the base models are not as significant when comparing the larger counterparts. We note that we also explored seq2seq pre-trained knowledge-enhanced models like KeyBART and GENRE, however both resulted in underwhelming performance compared to BART. Further exploration is required in improving performance for such models. We also note that while we demonstrate gains by switching to a classification-based approach in RINE, such models are limited in other generation task capabilities such as translation or summarization. We will release the data and code used for this work, but emphasize that some processing was done over the raw TOPv2 dataset, namely reconstructing source utterances directly from the provided target instead of using the provided source, as we encountered mismatches when constructing pointers. The source was then lowercased.

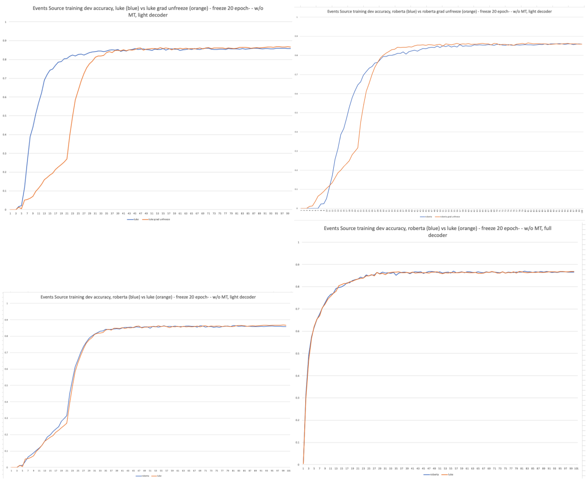

Figure 2: Training Curves for Source Training on the Navigation domain across various settings.

Ethics Statement

We use publicly available data sets in our experiments with permissive licenses for research experiments. We do not release new data or annotations as part of this work.

Acknowledgements

We would like to thank Ryan Gabbard, Amir Saffari, Kai-Wei Chang, Haidar Khan, Thomas Gueudre and Chandana Prakash for insightful discussions and feedback during the development of this work.

References

Armen Aghajanyan, Jean Maillard, Akshat Shrivastava, Keith Diedrick, Mike Haeger, Haoran Li, Yashar Mehdad, Ves Stoyanov, Anuj Kumar, Mike Lewis, et al. 2020. Conversational semantic parsing. *arXiv preprint arXiv:2009.13655*.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Konstantine Arkoudas, Nicolas Guenon des Mesnards, Melanie Rubino, Sandesh Swamy, Saarthak Khanna, Weiqi Sun, and Khan Haidar. 2022. Pizza: A new benchmark for complex end-to-end task-oriented parsing. *arXiv preprint arXiv:2212.00265*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *CoRR*, abs/2010.00904.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5090–5100, Online. Association for Computational Linguistics.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. *arXiv preprint arXiv:1706.04326*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. 2018. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. *arXiv preprint arXiv:1803.02400*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. 2020. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. *arXiv preprint arXiv:2006.10369*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning rich representation of keyphrases from text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 891–906, Seattle, United States. Association for Computational Linguistics.

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.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Elman Mansimov and Yi Zhang. 2022. Semantic parsing in task-oriented dialog with recursive insertion-based encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11067–11075.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. In *Proceedings of The Web Conference 2020*, pages 2962–2968.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2020. Meta-learning for domain generalization in semantic parsing. *arXiv preprint arXiv:2010.11988*.

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.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2022. An efficient memory-augmented transformer for knowledge-intensive nlp tasks.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. *CoRR*, abs/1912.09637.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Qile Zhu, Haidar Khan, Saleh Soltan, Stephen Rawls, and Wael Hamza. 2020. Don't parse, insert: Multilingual semantic parsing with insertion based decoding. *arXiv preprint arXiv:2010.03714*.

A Appendix

A.1 RINE Implementation Details

RINE uses an encoder, in this case RoBERTa-base, to encode the input sequence into hidden vectors, then uses a sequence classification head to predict the output label. It uses attention probabilities from the first and second attention head of the last attention layer to predict the begin and end positions, respectively. Finally, it trains the model by optimizing the combined three objectives, label loss, start position loss and end position loss.

The training data for RINE is different from seq2seq models. Unlike seq2seq models, RINE predicts a label to insert into the input sequence. Hence, to train the model we need to create a dataset with partial parses, where each training example corresponds to inserting one more label into a partial linearized parse, creating a new non-terminal semantic node in the parse tree. Similar to RINE paper, we follow a top-down generation ordering to create pairs of partially constructed trees.

A.2 Slot Matching Schema

Table 9 shows the slots we have used from each domain in TOPv2 while generating all for the slot-value augmentation of the FLAIR and BLINK recognized entities. Note we need these slot label schemas for all domain as Source Training is conducted across all domain except one (the target domain) and thus we require this information during Source Training. We had two authors define this schema based on inter-annotator agreement and data analysis.

A.3 Multitask Entity Labeling Example

For the utterance, "How long is the drive to 401 North Highway". In the case where FLAIR NER identifies "401 North" as a *Location (LOC)* entity-type, whereas our slot-matching schema identifies "401 North Highway" as it corresponds to the *Destination* Slot. Since these are overlapping spans from two systems we consider the longer span, which in this case leads to "401 North Highway" span tagged as an entity and "401 North" discarded.

Domain	Entity Slots
ALARM	N/A
EVENT	LOCATION, ORGANIZER_EVENT, CATEGORY_EVENT, NAME_EVENT, CATEGORY_LOCATION, ATTENDEE_EVENT, POINT_ON_MAP
MESSAGING	CATEGORY_LOCATION, CATEGORY_EVENT, RECIPIENT, RESOURCE, LOCATION, CONTACT, SENDER
MUSIC	MUSIC_PLAYLIST_TITLE, MUSIC_PROVIDER_NAME, MUSIC_TRACK_TITLE, MUSIC_ARTIST_NAME, MUSIC_ALBUM_TITLE
NAVIGATION	LOCATION, DESTINATION, SOURCE, POINT_ON_MAP, CATEGORY_LOCATION, MUTUAL_LOCATION, LOCATION_WORK, LOCATION_CURRENT, NAME_EVENT, PATH, PATH_AVOID
REMINDER	PERSON_REMINDED, ORGANIZER_EVENT, CATEGORY_EVENT, ATTENDEE_EVENT, RECIPIENT, ATTENDEE, CONTACT, SENDER
TIMER	N/A
WEATHER	LOCATION, CONTACT

Table 9: Slots schema matching mechanism to detect mentions in all the TOPv2 Domains.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Ethics

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 2 and Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Ethics

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Ethics

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Ethics

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Table 1

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Table 1 and Table 2

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4 - Implementation Details

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 - Implementation Details

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4 - Implementation Details

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.