Retrieve-and-Fill for Scenario-based Task-Oriented Semantic Parsing

Akshat Shrivastava **Shrey Desai** **Anchit Gupta** **Ali Elkahky**
Aleksandr Livshits **Alexander Zotov** **Ahmed Aly**
Meta
{akshats, shreyd, alielk, anchit,
alll, alexzotov, ahhegazy}@meta.com

Abstract

Task-oriented semantic parsing models have achieved strong results in recent years, but unfortunately do not strike an appealing balance between model size, runtime latency, and cross-domain generalizability. We tackle this problem by introducing scenario-based semantic parsing: a variant of the original task which separates disambiguating an utterance's "scenario" (an intent-slot template with variable leaf spans) and generating its frame, complete with ontology and utterance tokens. This formulation closely ties to the data collection process where the scenarios are first designed followed by crowd sourced utterance annotation. Concretely, we create a Retrieve-and-Fill (RAF) architecture comprised of (1) a retrieval module which ranks the best scenario given an utterance and (2) a filling module which imputes spans into the scenario to create the frame. Our model is modular, differentiable, interpretable, and allows us to garner extra supervision from scenarios. RAF achieves strong results in high-resource, low-resource, and multilingual settings, outperforming recent approaches despite, using base pre-trained encoders and efficient decoding.

1 Introduction

Task-oriented conversational assistants typically first use semantic parsers to map textual utterances into structured frames for language understanding (Hemphill et al., 1990; Coucke et al., 2018; Gupta et al., 2018; Rongali et al., 2020; Aghajanyan et al., 2020). While these parsers achieve strong performance with rich supervision, they often face obstacles adapting to novel settings, especially ones with distinct semantics and scarce data. Recent approaches address this by improving parsers' data efficiency, such as by using pre-trained representations (Aghajanyan et al., 2020; Rongali et al., 2020), optimizing loss functions (Chen et al., 2020b), and supplying natural language prompts (Desai et al., 2021). However, these approaches typically rely on larger models and longer contexts, impeding their applicability in real-world conversational assistants. Even though non-autoregressive models can alleviate some concerns (Babu et al., 2021; Shrivastava et al., 2021; Zhu et al., 2020), to the best of our knowledge, there exists no approach which strikes an appealing balance between model size, runtime latency, and cross-domain generalizability.

We begin tackling this problem by introducing **scenario-based task-oriented semantic parsing** which is more closely tied to how new task domains are developed. A slight variation on original semantic parsing, we are given access to all supported scenarios apriori and have to parse the utterance given this scenario bank. Here, a scenario is akin to a *incomplete* frame; it is, precisely, an intent-slot template with variables as leaf spans (e.g., `IN:GET_WEATHER [SL:LOCATION` x_1 `] ]`), indicating it maps to a family of linguistically similar utterances. As domain development and data-collection usually starts out with designing the set of supported scenarios, our intuition is that by giving the model access to this bank we can train it to more explicitly reason about scenarios and improve performance especially in the low data regime.

Concretely, we propose RAF (Retrieve-and-Fill), a modular yet differentiable architecture for scenario-based task-oriented semantic parsing. Guided by the definition of our task, RAF also proceeds in two steps: (1) given an utterance, a **retrieval** module finds the highest ranking scenario and (2) given the utterance and retrieved scenario, a **filling** module imputes spans into the scenario, creating the final frame. This approach requires no extra supervision despite performing auxiliary inference: utterances and frames are typically provided, and scenarios are obtained by stripping leaf text in frames. We design RAF to capitalize on the advantages of prior work but avoid their disadvantages; using base pre-trained encoders across-the-

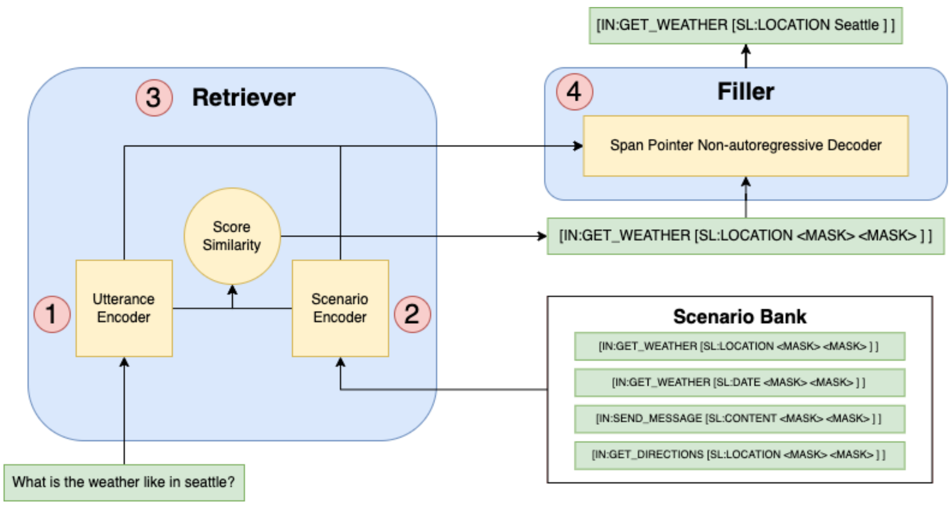
[IN:GET_WEATHER [SL:LOCATION Seattle]]
3 Retriever
4 Filler
Span Pointer Non-autoregressive Decoder
Score Similarity
[IN:GET_WEATHER [SL:LOCATION <MASK> <MASK>]]
1 Utterance Encoder
Scenario Encoder 2
Scenario Bank
[IN:GET_WEATHER [SL:LOCATION <MASK> <MASK>]]
[IN:GET_WEATHER [SL:DATE <MASK> <MASK>]]
[IN:SEND_MESSAGE [SL:CONTENT <MASK> <MASK>]]
[IN:GET_DIRECTIONS [SL:LOCATION <MASK> <MASK>]]
What is the weather like in seattle?

Figure 1: **High-Level Overview.** Retrieve-and-Fill (RAF) consists of 4 steps: (1) Encode the utterance via an utterance encoder (e.g., RoBERTa); (2) Encode all scenarios (Desai et al., 2021) in the scenario bank into a cached index via a scenario encoder (e.g., RoBERTa); (3) Compute the dot product similarity amongst the incoming utterance and the index of all scenarios, obtaining the top-n candidates; (4) For each retrieved scenario, leverage the non-autoregressive span pointer decoder (Shrivastava et al., 2021) to impute each scenario's spans.

board, our retrieval module caches intrinsic representations during inference and our filling module non-autoregressively decodes leaf spans in a generalizable fashion.

We evaluate our approach in high-resource, low-resource, and multilingual settings using standard task-oriented semantic parsing datasets. RAF achieves 87.52% EM on TOPv2 (Chen et al., 2020b) and 86.14% EM on TOP (Gupta et al., 2018), outperforming recent autoregressive and non-autoregressive models. RAF also excels in weakly supervised settings: on TOPv2-DA, we outperform Inventory (Desai et al., 2021) on 4 domains (alarm, music, timer, weather) despite using <128 token sequence lengths and 2-4x less parameters, on TOPv2 low resource, we outperform BART (Lewis et al., 2020), RINE (Mansimov and Zhang, 2021), and RoBERTa + Span Pointer (Shrivastava et al., 2021), and on MTOP (Li et al., 2021), we outperform XLM-R + Span Pointer, achieving 42% EM averaged across en$\rightarrow${es, fr, de, hi, th} transfer tasks.

To summarize, our contributions are: (1) Introducing scenario-based task-oriented semantic parsing, a novel task which requires disambiguating scenarios during typical utterance$\rightarrow$frame predictions; (2) Creating RAF (Retrieve-and-Fill), a modular yet differentiable architecture composed of a retrieval and filling module for solving scenario-based task-oriented semantic parsing; and (3) Achieving strong results in high-resource, low-resource, and multilingual settings, outperforming recent models such as Span Pointer, Inventory, and RINE by large margins, while also optimizing for model size, runtime latency, and cross-domain generalizability.

2 Scenario-based Semantic Parsing

We formally introduce the task of scenario-based semantic parsing. Task-oriented conversational assistants support a wide range of domains (e.g., calling, reminders, weather) to maximize coverage over users' needs. Over time, in response to requests and feedback, developers often iterate on assistants' skill-sets by adding new domains. Though the crowdsourcing process of collecting and annotating samples—utterances and frames—for new domains can be accomplished in many ways, we propose a two-step methodology where developers (a) develop scenarios which roughly describe a family of samples and (b) collect linguistically-varied samples consistent with each scenario. We elaborate more on our scenario-based methodology below.

Scenario Definition. We define a **scenario** as an intent-slot rule which abstracts away linguistic variation in utterances. More specifically, it is a decoupled semantic frame (Aghajanyan et al., 2020) with variables in leaf spans, indicating it can subclass utterances with similar syntactic and semantic structure, an example is showing in table 1. Our no-

Utterance	Frame
Scenario: [IN:GET_WEATHER [SL:LOCATION x_1]]	
what's the weather in seattle	[IN:GET_WEATHER [SL:LOCATION seattle]]
how's the forecast in sf	[IN:GET_WEATHER [SL:LOCATION sf]]
Scenario: [IN:GET_WEATHER [SL:LOCATION x_1] [SL:DATE_TIME x_2]]	
what's the weather in seattle tomorrow	[IN:GET_WEATHER [SL:LOCATION seattle] [SL:DATE_TIME tomorrow]]
how's the forecast in sf at 8pm	[IN:GET_WEATHER [SL:LOCATION sf] [SL:DATE_TIME 8pm]]

Table 1: **Scenario Description.** Scenarios are intent-slot templates with missing slot text, suggesting they subclass linguistically similar utterances. We show examples of utterances, scenarios, and frames in the weather domain; each scenario consists of multiple (utterance, frame) pairs.

tion of a scenario is inspired by production rules in constituency parsing (Chomsky, 1959), but the parallel is not exact given our scenarios have semantic not syntactic types.

Using scenarios, we can effectively quantize the space of possible utterances by identifying and defining slices of user requests. Our solution also offers fine-grained control over precision and recall, which is important in real-world systems; we can collect more paraphrases to improve precision and we can create more scenarios to improve recall.

We collect scenarios by taking train, eval, and test frames from our datasets, and stripping out the utterance text, to yield a decoupled semantic frame. As discussed later in this manuscript, our approach requires knowing, beforehand, the space of possible scenarios, in order to perform global inference.

Case Study: Weather. Table 1 shows an example setting where we crowdsource weather domain samples using the scenario-based methodology outlined above. To begin, we may envision building a weather system which supports requests with location and/or date-time information. Therefore, we can define two scenarios: (1) a family of samples with one location span; and (2) a family of samples with one location span and one date-time span. Each scenario explicitly outlines the intents and slots that must be present, as scenarios are not "one-size-fits-all" rules with optional slotting. So, as an example, the utterance "how's the forecast in sf" would not be compatible with the scenario [IN:GET_WEATHER [SL:LOCATION x_1] [SL:DATE_TIME x_2]] since it does not have a date-time span as specified by the x_2 variable.

Task Definition. Finally, we precisely define our task of scenario-based semantic parsing. For the typical task of semantic parsing, we define U as a random variable over utterances, F as a random variable over frames, and model $P(F|U)$: find the most likely frame given the utterance. However, because we introduce scenarios as a coarse, intermediate representation of frames, we additionally define S as the set of all supported scenarios given a priori, and model $P(F|U, S)$.

3 RAF: Retrieve and Fill

We propose a model called RAF (Retrieve and Fill) for scenario-based semantic parsing that naturally decomposes the task into a coarse-to-fine objective where we (a) find the most likely scenario given the utterance and (b) find the most likely frame given the utterance and scenario. More concretely given an utterance u and scenarios $s_1, \cdots, s_n$, as well as a gold scenario s^* and gold frame f^*, we learn our model as follows:

1. Retrieve (Coarse Step; §3.1): A **retrieval module** maximizes $P(S = s^*|U = u)$ by learning to retrieve scenario s_i given utterance u, e.g., "what's the weather in seattle" $\rightarrow$ [IN:GET_WEATHER [SL:LOCATION x_1]].

2. Fill (Fine Step; §3.2): A **filling module** maximizes $P(F = f^*|U = u, S = s^*)$ by decoding the most likely frame f given the structure of scenario s_i^* and spans of utterance u, e.g., [IN:GET_WEATHER [SL:LOCATION x_1]] $\rightarrow$ [IN:GET_WEATHER [SL:LOCATION seattle]].

RAF is a composable yet differentiable model which implements coarse-to-fine processing: we first develop a coarse-grained sketch of an utterance's frame, then impute fine-grained details to achieve the final frame. In these types of approaches, there often exists a trade-off when creating the intermediate representation; if it is "too coarse", the filling module suffers, but if it is "too fine", the retrieval module suffers. We find scenarios, as defined in §2, offer the most appealing solution, as the retrieval module unearths rough syntactic-semantic structure, while the filling module focuses in on imputing exact leaf spans.

In the following sub-sections, we discuss the technical details behind the retrieval and filling modules, as well as describe the training and inference procedures.

3.1 Retrieval Module

First, we discuss the **coarse-grained step** of RAF, which aims to find the best-fitting scenario for an utterance. We formulate this task as a metric learning problem. Here, we can maximize the similarity $\mathrm{sim}(u, s^*)$ between an utterance u and scenario s^* as judged by a scalar metric. This offers numerous advantages: we can explore ad-hoc encodings of utterances and scenarios, adjust the output space dynamically during inference, and compute exact conditional probabilities by leveraging a (tractable) partition function.

3.1.1 Bi-Encoder Retrieval

Following retrieval modeling in open-domain QA (Karpukhin et al., 2020), we specifically leverage pre-trained encoders (E_U for utterances and E_S for scenarios) to compute dense vector representations, then maximize the dot product similarity $\mathrm{sim}(u, s^*) = E_U(u)^\top E_S(R(s^*))$ between utterance-scenario pairs (u, s^*); the precise nature of R is discussed in §3.1.3. To learn such a metric space and avoid degenerate solutions we need to train the encoders to *pull* positive pairs together and *push* negative pairs apart. Hence, we need access to both positive (gold; (u, s^+)) and negative (non-gold; (u, s^-)) pairs.

3.1.2 Negatives Sampling

The choice of negatives has a large impact on retrieval performance, which is consistent with findings in information retrieval (Zhan et al., 2021; Karpukhin et al., 2020). We explore two types of negative sampling to improve retrieval performance: in-batch negatives and model-based negatives.

In-Batch Negatives. We mine positive and negative pairs from each training batch using in-batch negatives (Karpukhin et al., 2020). Let $\mathbf{U}$ and $\mathbf{S}$ be the utterance and scenario matrices, each being a $(B \times d)$ matrix consisting of d-dimensional embeddings up to batch size B. We obtain B^2 similarity scores upon computing the similarity matrix $\mathbf{M} = \mathbf{U}\mathbf{S}^\top$, where $\mathbf{M}_{i=j}$ consists of positive scores and $\mathbf{M}_{i\neq j}$ consists of negative scores. For each positive utterance-scenario pair (u_i, s_i^+), we now have $(B-1)$ negative pairs $\{(u_i, s_{ij}^-)\}_{i\neq j}$. Having collected multiple negative pairs per positive pair, we leverage the contrastive loss for utterance i is defined in Karpukhin et al. (2020); Chen et al. (2020a):

$$\mathcal{L}^i_{\text{retrieval}}(u_i, s_1, \cdots, s_b) = \log \frac{e^{\mathrm{sim}(u_i, s_i)}}{\sum_{j=0}^{b} e^{\mathrm{sim}(u_i, s_{ij})}} \quad (1)$$

Since our set of scenarios isn't huge this can result in conflicts so we additionally implement **identity masking** while training which ensures that for every utterance each scenario is present at-most once in it's positive/negative set.

Model-Based Negatives. Following prior work in IR (Xiong et al., 2020; Oğuz et al., 2021) we train an initial retrieval model that uses only In-Batch negatives. This model is used to rank all scenarios for each utterance and the top K ranked scenarios which are not the gold scenario are selected as additional negative examples. Those negatives examples are expected to be harder examples since they were ranked higher by the initial model. We train a new model using those added k scenarios as explicit negative examples for each positive example.

3.1.3 Frame Representation

While we have covered scenario-based retrieval above, we have not yet precisely described how dense vectors for scenarios are computed. Recall our definition of utterance-scenario similarity in §3.1.1: our objective is to maximize $\mathrm{sim}(u, s) = E_U(u)^\top E_S(R(s))$, where R is a string transformation applied to scenarios.

Following Desai et al. (2021), we leverage **intrinsic modeling** to rewrite intents and slots as a composition of their intrinsic parts in a single string. Desai et al. (2021) define two, in particular: the categorical type (e.g., "intent" or "slot") and language span (e.g., "get weather" or "location"). Using this methodology, we can transform the scenario `IN:GET_WEATHER [SL:LOCATION` x_1 `] ]` $\rightarrow$ `[ intent | get weather [ slot | location` x_1 `] ]`, which is inherently more natural and descriptive.

Guided by the general concept of intrinsic modeling, our goal here is to define R such that

$E_U(u)^\top E_S(s) \leq E_U(u)^\top E_S(R(s))$, all else being equal. We discuss our approach in detail in the following sub-sections.

3.1.4 Language Spans

Desai et al. (2021) chiefly use an **automatic** method to extract language spans from ontology labels. For example, using standard string processing functions, we can extract "get weather" from `IN:GET_WEATHER`. While this method is a surprisingly strong baseline, it heavily relies on a third-party developers' notion of ontology nomenclature, which may not always be pragmatically useful. In TOPv2 (Chen et al., 2020b), our principal evaluation dataset, there exists ambiguous labels like `SL:AGE`—is this referring to the age of a person, place, or thing?

Therefore, to improve consistency and descriptiveness, we propose a **handmade** method where we manually design language spans for each ontology label.[1] The most frequent techniques we use are (1) using the label as-is (e.g., `IN:GET_WEATHER` $\rightarrow$ "get weather"); (2) inserting or rearranging prepositions (e.g., `IN:ADD_TO_PLAYLIST_MUSIC` $\rightarrow$ "add music to playlist"; and (3) elaborating using domain knowledge (e.g., `IN:UNSUPPORTED_ALARM` $\rightarrow$ "unsupported alarm request").

3.1.5 Example Priming

Despite using curation to improve ontology label descriptions, there are still many labels which remain ambiguous. One such example is `SL:SOURCE`; this could refer to a travel source or messaging source, but without seeing its exact manifestations, it is challenging to fully grasp its meaning. This motivates us to explore **example priming**: augmenting scenario representations with randomly sampled, dataset-specific slot values. This can help our model further narrow down the set of spans each slot maps to during parsing. Furthermore, our examples are just spans, so they are straightforward to incorporate into our representation. For example, for the slot `SL:WEATHER_TEMPERATURE_UNIT`, we can augment and contextualize its representation "slot | unit of weather temperature" with "slot | unit of weather temperature | F / C" where "F" and "C" are examples which appear in our dataset.

[1]See Appendix §D for our curated intent and slot descriptions, respectively.

3.1.6 Representation Sampling

Our scenario-based retrieval task performs reasoning over a compact set of scenarios, unlike large-scale, open-domain tasks such as information retrieval and question answering which inculcate millions of documents. As such, **the chance our system overfits to a particular representation is much greater**, no matter what it is set to. So, we instead make R stochastic by uniformly sampling unique frame representations for encoding scenarios, and only using the handmade representations during inference; Table 12 enumerates the complete set of outcomes.

3.2 Filling Module

We use Span pointer model (Shrivastava et al., 2021; Nicosia et al., 2021) as the basis of our infilling model. Figure 2.A shows the how the model works. In our case, we use scenario tokens from the retrieved scenario as input tokens to decoder as shown in figure 2.B. We also tried to pass the scenario encoder output as input to the decoder to replace the discrete input to the decoder and called this **scenario fusion** as shown in figure 2.C.

Our final objective is $\mathcal{L}_{\text{filling}} = \text{NLL}(f^*, f) + \alpha\text{LS}(f)$ where LS refers to label smoothing (Pereyra et al., 2017).

4 Experiments and Results

We evaluate RAF in three settings: a high-resource setting (100,000+ training samples), low-resource setting (1-1,000 training samples), and multilingual setting (0 training samples). Hyper parameter details are described in Appendix §E. Our goal here is to show that our system both achieves competitive performance on established benchmarks and offers substantial benefits in resource-constrained environments where training samples are limited.

4.1 Datasets for Evaluation

Following prior work in task-oriented semantic parsing, we use 5 datasets for evaluation: TOP (Gupta et al., 2018), TOPv2 (Chen et al., 2020b), TOPv2-LR (Low Resource; Chen et al. (2020b)), TOPv2-DA (Domain Adaptation; Desai et al. (2021)), and MTOP (Li et al., 2021). TOP and TOPv2 are used for high-resource experiments, TOPv2-LR and TOPv2-DA are used for low-resource experiments, and MTOP is used in multilingual experiments.

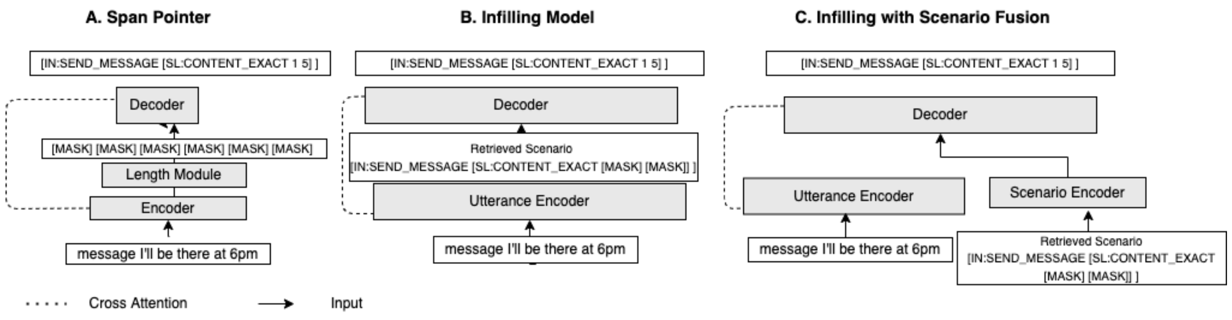
A. Span Pointer
B. Infilling Model
C. Infilling with Scenario Fusion
[IN:SEND_MESSAGE [SL:CONTENT_EXACT 1 5]]
Decoder
[MASK] [MASK] [MASK] [MASK] [MASK] [MASK]
Length Module
Encoder
message I'll be there at 6pm
Retrieved Scenario
[IN:SEND_MESSAGE [SL:CONTENT_EXACT [MASK] [MASK]]]
Utterance Encoder
Scenario Encoder
Cross Attention
Input

Figure 2: **Infilling Model.** A. the Span Pointer model in (Shrivastava et al., 2021). B. the Infilling model using tokens from retrieved scenario and C. the infilling model with scenario fusion where scenario encoder output is connected to the decoder directly. Note in all cases the decoder apply cross attention on the utterance encoder output.

4.2 Systems for Comparison

We compare against multiple task-oriented semantic parsing models, which cover autoregressive (AR), and non-autoregressive (NAR) training. See Aghajanyan et al. (2020); Mansimov and Zhang (2021); Babu et al. (2021); Shrivastava et al. (2021) for detailed descriptions of these models.

The autoregressive models consist of BART (Lewis et al., 2020) and RoBERTa (Liu et al., 2019), and RINE (Mansimov and Zhang, 2021), and the non-autoregressive models are RoBERTa NAR (Babu et al., 2021) and RoBERTa NAR + Span Pointer (Shrivastava et al., 2021). These models are applicable to both high-resource and low-resource settings; though, for the latter, we also add baselines from Desai et al. (2021): CopyGen (BART + copy-gen decoder) and Inventory (BART + intrinsic modeling). The multilingual setting only requires swapping RoBERTa with XLM-R (Conneau et al., 2020).

We denote our system as RAF in our experiments. Unless noted otherwise, we use RoBERTa$_{\text{BASE}}$ for the utterance encoder θ_U and secnario θ_S and a random-init, copy-gen, transformer decoder for the frame decoder θ_F. As alluded to before, we swap RoBERTa$_{\text{BASE}}$ with XLM-R$_{\text{BASE}}$ for multilingual experiments.

4.3 High-Resource Setting

First, we evaluate RAF in a high-resource setting where hundreds of thousands are samples are available for supervised training; Table 2 shows the results. RAF achieves strong results across-the-board, using both base and large pre-trained encoders: **RAF$_{\text{BASE}}$ consistently outperforms other base variants by 0.25-0.5 EM and RAF$_{\text{LARGE}}$ comparatively achieves the best results on TOPv2.**

Model	TOPv2	TOP
Type: Autoregressive Modeling (Prior)		
RoBERTa$_{\text{BASE}}$	86.62	83.17
RoBERTa$_{\text{LARGE}}$	86.25	82.24
BART$_{\text{BASE}}$	86.73	84.33
BART$_{\text{LARGE}}$	87.48	85.71
RINE$_{\text{BASE}}$	—	87.14
RINE$_{\text{LARGE}}$	—	**87.57**
Type: Non-Autoregressive Modeling (Prior)		
RoBERTa$_{\text{BASE}}$	85.78	82.37
+ Span Pointer	86.93	84.45
RoBERTa$_{\text{LARGE}}$	86.25	83.40
+ Span Pointer	87.37	85.07
Type: Scenario Modeling (Ours)		
RAF$_{\text{BASE}}$	87.11	86.00
RAF$_{\text{LARGE}}$	**87.52**	86.14

Table 2: **High-Resource Results.** Exact Match (EM) on TOPv2 (Chen et al., 2020b) and TOP (Gupta et al., 2018). We compare various semantic parsing paradigms: autoregressive, non-autoregressive, and scenario. RAF achieves strong performance on TOPv2 and TOP, illustrating its competitiveness with state-of-the-art models.

4.4 Low-Resource Setting

Having established our system is competitive in high-resource settings, we now turn towards evaluating it in low-resource settings, where training samples are not as readily available. Here, we chiefly consider two setting types: a **high difficulty** setting (TOPv2-DA) with 1-10 samples and a **medium difficulty** setting (TOPv2-LR) with 100-1,000 samples. The exact number of samples in a few-shot training subset depend on both the subset's cardinality and sampling algorithm.

Tables 3 and 4 show results on the high and medium difficulty settings, respectively. **RAF achieves competitive results in the high-difficulty setting, outperforming both CopyGen$_{\text{BASE}}$ and Inventory$_{\text{BASE}}$ by large**

	alarm	music	timer	weather
CopyGen$_{\text{BASE}}$	47.24	25.58	16.62	47.24
CopyGen$_{\text{LARGE}}$	36.91	23.84	32.64	53.08
Inventory$_{\text{BASE}}$	62.13	23.00	28.92	54.53
Inventory$_{\text{LARGE}}$	**67.25**	**38.68**	48.45	**61.77**
RAF$_{\text{BASE}}$ (ours)	62.71	35.47	**55.06**	61.05

Table 3: **High-Difficulty Low-Resource Results.** EM on the 1 SPIS split of TOPv2-DA (Desai et al., 2021). Compared to Inventory and CopyGen baselines, RAF achieves competitive performance with a fraction of parameter usage.

	Weather Domain (SPIS)					
	10	25	50	100	500	1000
Type: Autoregressive Modeling (Prior)						
RoBERTa$_{\text{BASE}}$ AR	69.71	74.90	77.02	78.69	—	86.36
BART$_{\text{BASE}}$ AR	73.34	73.35	76.58	79.16	—	86.25
RINE$_{\text{BASE}}$	—	74.53	—	—	87.80	—
RINE$_{\text{LARGE}}$	—	77.03	—	—	87.50	—
Type: Non-Autoregressive Modeling (Prior)						
RoBERTa$_{\text{BASE}}$ NAR	59.01	72.12	73.41	78.48	—	87.42
+ Span Pointer	72.03	74.74	74.85	78.14	—	**88.47**
Type: Scenario Modeling (Ours)						
RAF$_{\text{BASE}}$	**75.10**	**78.74**	**77.53**	**79.67**	**87.91**	88.17

Table 4: **Medium-Difficulty Low-Resource Results.** EM on various SPIS splits of the TOPv2 (Chen et al., 2020b) weather domain. RAF largely outperforms autoregressive and non-autoregressive models, trailing RoBERTa-Base + Span Pointer only in a high-resource split.

margins; notably, on timer, we nearly double Inventory$_{\text{BASE}}$'s exact match score. RAF also performs well in the medium-difficulty setting; our system consistently outperforms prior autoregressive, and non-autoregressive models.

4.5 Multilingual Setting

Finally, we consider a multilingual setting, where a model trained on English samples undergoes zero-shot transfer to non-English samples. In Table 5, we see that, **compared to XLM-R$_{\text{BASE}}$ NAR + Span Pointer, RAF achieves +2.3 EM averaged across all 5 non-English languages.** Upon inspecting this result more closely, RAF's performance is strong across both typologically *similar* languages (+4.8 EM on Spanish, +3.5 EM on French) and *distinct* languages (+2.7 EM on Hindi and Thai).

5 Ablations and Analysis

We perform model ablations on RAF, removing core retrieval- and filling-related components we originally introduced in §3 to better understand the design decisions. From Tables 6 and Table 7, we draw the following conclusions:

Negatives are important for accurate scenario retrieval. The metric learning objective for retrieval, as introduced in §3.1.2, precisely delineates between positive and negative samples. Our ablations show model-based negatives and identity masking are critical to achieve best performance; when removing model-based negatives, for example, retrieval accuracy drops by 3%+. We also investigate training RAF with heuristic-based negatives: a simple algorithm which finds top-k similar scenarios with string-based edit distance (Appendix C). However, heuristic-based negatives regress both retrieval-only and end-to-end approach, suggesting model-based negatives are more informative.

Sharing parameters between retrieval encoders improves quality. Our retrieval module has two encoders: an utterance encoder E_U and a scenario encoder E_S. An important design decision we make is tying both encoders' parameters together with RoBERTa (Liu et al., 2019); this improves end-to-end performance by roughly +0.6 EM. We believe that parameter sharing among retrieval encoders improves generalizability: because there are more vastly more unique utterances than scenarios, the scenario encoder may overfit to a select set of scenarios, so weight tying enables the joint optimization of both encoders.

Scenario fusion enables better end-to-end modeling. Because RAF is composed of two neural modules—the retrieval and filling modules—chaining them together arbitrarily may result in information loss. The filling module chiefly uses scenario token embeddings to reason over the retrieval module's outputs. Our results show that scenario fusion, initializing these embeddings using the scenario encoder's final state, improves upon random init by +0.77 EM and +0.65 EM-S.

Intrinsic representations improve low-resource performance. When comparing the high-level scenario representation, we see that canonical (e.g., "[IN:GET_WEATHER [SL:LOCATION]") underperforms intrinsic (e.g., "[intent | get weather [slot | location]]") by a wide margin (-4.49%) in the target domain. This implies RAF *better* understands scenarios' natural language descriptions even if they contain unseen, domain-specific terms. **Furthermore, we see leveraging all concepts (handmade, automatic, examples) achieves both competitive source and target performance.** Even though excluding handmade improves target perfor-

		Zero-Shot Evaluation					
	en	en→es	en→fr	en→de	en→hi	en→th	Avg
XLM-R$_{\text{BASE}}$ NAR	78.3	35.2	32.2	23.6	18.1	16.7	25.2
+ Span Pointer	**83.0**	51.2	51.4	**42.0**	29.6	27.3	40.3
RAF$_{\text{BASE}}$ (ours)	81.1	**56.0**	**54.9**	40.1	**32.1**	**30.0**	**42.6**

Table 5: **Multilingual Results.** We perform zero-shot experiments where we fine-tune a parser on English (en), then evaluate it a non-English language—Spanish (es), French (fr), German (de), Hindi (hi), and Thai (th)—without fine-tuning. Average EM (Avg) is taken over the five non-English languages. RAF outperforms XLM-R-Base + Span Pointer by +2.3% on average.

Model	EM	EM-S
Classify and Fill	84.80	87.16
RAF$_{\text{BASE}}$	**87.03**	**89.34**
- Hard Negatives	83.69	86.00
- Identity Masking	85.87	88.23
- Scenario Fusion	86.26	88.69
- Parameter sharing	86.76	89.10
- Repr. Sampling	86.70	89.05
+ Heuristic negatives	85.66	89.10

Table 6: **Model Ablations.** We assess several components of our model, individually removing them and evaluating EM (Exact Match) and EM-S (Exact Match of Scenarios, i.e., intent-slot templates without slot text) on TOPv2 (Chen et al., 2020b) validation set.

	Source Fine-Tuning		Target Fine-Tuning	
Model	EM	EM-S	EM	EM-S
Canonical Repr.	86.67	88.74	74.25	76.80
Intrinsic Repr.	**87.10**	**89.15**	78.74	81.70
- Automatic	87.02	89.06	78.92	81.80
- Handmade	86.85	88.87	**79.27**	**82.29**
- Examples	86.91	88.90	78.25	80.89

Table 7: Scenario representation ablation, where the target domain is a 25 SPIS split of the TOPv2 (Chen et al., 2020b) weather. Following the typical few-shot fine-tuning methodology, we perform source fine-tuning on all domains except weather and reminder, then perform target fine-tuning on a specific split.

mance (+0.53%), it regresses source performance (-0.28%), suggesting sampling all representations is more generalizable.

Runtime Latency Empirically we find that RAF retains the runtime latency of Span Pointer, due to it's non-autoregressive decoding nature despite leveraging the retrieval module. We find that the difference is within 3ms P99 and significantly faster than the autoregressive counter part (189ms). We report the full experiment details in A.2.

Interpretablity RAF offers a more interpretable modeling paradigm Due to (1) RAF encoding utterances and scenarios into a joint embedding space which we can directly visualize. We present two case studies leveraging these visualizations: domain development (B.1) and error analysis (B.2). (2) RAF retains modularity of components to isolate retrieval vs filling issues. In A.1 we show the comparisons of retrieval vs filling to highlight potential of each module, in particular on the TOPv2 eval dataset, oracle retrieval improves performance +9.5% where oracle filling improves performance +2.3% indicating the importance of retrieval.

5.1 Extra-Scenario Generalization

A core difference between scenario-based and non-scenario-based (seq2seq; autoregressive or non-autoregressive) models is that scenario-based models "know" of all scenarios beforehand, while seq2seq models do not, and therefore have to purely rely on generalization. We further quantify the impact that this has by dividing overall EM using two groups: (1) Known vs. Unknown - i.e scenarios in the training dataset vs. scenarios only in the test dataset and (2) In-Domain vs. Out-of-Domain - scenarios with a supported intent vs. unsupported intent (e.g., IN:UNSUPPORTED_*).

From the results in Table 8, we draw a couple of conclusions. First on Unknown EM, even though Span pointer, BART models are capable of generating novel scenarios not part of the train set they do so poorly. RAF does much better on this given the caveat that it adds the novel scenarios to its frame index beforehand but it hasn't seen any paired utterance to them. Second, RAF outperforms on In-Domain EM but underperforms on Out-of-Domain EM. Because RAF leverages intrinsic descriptions of scenarios, the word "unsupported" may not precisely capture what it means for an utterance to be in- vs. out-of-domain.

6 Related Work

Scaling Semantic Parsing A critical theme in semantic parsing is reducing data requirements to stand up new domains and scenarios. Existing works rely on leveraging large language models such as BART (Lewis et al., 2020) with augmenta-

Model	EM	Known EM	Unknown EM	ID EM	OOD EM
RAF$_{\text{BASE}}$	**87.14**	88.30	**59.96**	**88.67**	44.11
SpanPointer$_{\text{BASE}}$	86.76	**88.50**	46.20	88.07	**49.70**
BART$_{\text{BASE}}$	86.72	88.33	49.15	88.03	49.69

Table 8: Comparing EM on Known vs. Unknown and In-Domain (ID) vs. Out-of-Domain (OD) frames. RAF performs better on unknown frames, but struggles with out-of-domain frames.

tions for scaling. In particular Chen et al. (2020b) introduce a meta-learning approach to improve domain scaling in the low-resource setting. Other works such as (Liu et al., 2021; Zhu et al., 2020; Mansimov and Zhang, 2021) aim to improve scaling through new decoding formulations. Desai et al. (2021) introduce the concept of **intrinsic modeling** where we provide a human-readable version of the semantic parsing ontology as context to encoding to improve few-shot generalization.

Our work leverages the intrinsic modeling paradigm by building a function R to convert each intent-slot scenario into a readable representation via intent slot descriptions and example priming. Furthermore, our bi-encoder based retrieval setup allows us to inject additional context into each scenario and cache it to an index in order to retain inference efficiency.

Retrieval Based Semantic Parsing Finally, there has been a recent trend towards dense retrieval in various NLP domains such as machine translation (Cai et al., 2021), question answering (Karpukhin et al., 2020), text generation (Cai et al., 2019) and language modeling (Borgeaud et al., 2018). Recent works also introduce retrieval-based semantic parsing: RetroNLU (Gupta et al., 2021) and CASPER (Pasupat et al., 2021) both leverage a retrieval step to provide examples as context to seq2seq models.

Our approach differs in two ways: (1) We phrase our problem as utterance-to-scenario retrieval rather than utterance-to-utterance retrieval. This allows us to look into supporting new scenarios with minimal-to-no-data required for retrieval. (2) Prior work leverage a separate module (Pasupat et al., 2021) or separate iteration (Gupta et al., 2021) for retrieval. We conduct our retrieval after encoding but prior to decoding as an intermediate step for non-autoregressive parsing. This allows our model to retain similar inference speed to one shot non-autoregressive decoding despite leveraging retrieval.

7 Conclusion

In this paper, we tackle scenario-based semantic parsing with retrieve-and-fill (RAF), a coarse-to-fine model which (a) retrieves a scenario with the best alignment to an utterance and (b) fills the scenario with utterance spans in leaf positions. Experiments show our model achieves strong results in high-resource, low-resource, and multilingual settings. The modular nature of our architecture also lends itself well to interpretability and debuggability; we perform several case studies uncovering the inner-workings of our approach.

8 Limitations

Although RAF has shown promising results on task oriented parsing, we identify the following limitations.

Use of Scenario Index Critically we introduce the scenario based parsing task, a problem where we assume knowledge of all possible scenarios across train/test/eval. This new task is inspired by how domain development usually happens 1)Define scenarios 2) collect and annotate utterances. While powerful and applicable when grammars and downstream applications can provide a scenario index, it prohibits generating novel scenarios or intent/slot combination that have not been indexed, where as prior seq2seq approaches are capable of this. We argue that such generalization is not a priority in pipelined task oriented assistant systems as the downstream application usually handles a set of pre-defined scenarios. Refer to §5.1 for more analysis/commentary on this.

Training Cost RAF has a high training cost due to the use of dense retrieval models and model based negative sampling. While inference is cheap due to a cached index and non-autoregressive decoding. During training, we must compute forward/backward over 2 large encoders (utterance encoder/scenario encoder) which uses significant memory and prohibits larger batch sizes. Additionally, model based negative samples requires two training iterations: first with in-batch negatives, second using the first stage model to identify hard negative samples from the model.

References

Armen Aghajanyan, Jean Maillard, Akshat Shrivastava, Keith Diedrick, Michael Haeger, Haoran Li, Yashar Mehdad, Veselin Stoyanov, Anuj Kumar, Mike Lewis, and Sonal Gupta. 2020. Conversational Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better Fine-tuning by Reducing Representational Collapse. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Arun Babu, Akshat Shrivastava, Armen Aghajanyan, Ahmed Aly, Angela Fan, and Marjan Ghazvininej. 2021. Non-Autoregressive Semantic Parsing for Compositional Task-Oriented Dialog. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2018. Improving Language Models by Retrieving from Trillions of Tokens. *arXiv preprint arXiv:2112.04426*.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1866–1875, Hong Kong, China. Association for Computational Linguistics.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural Machine Translation with Monolingual Translation Memory. *arXiv preprint arXiv:2105.11269*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020a. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv preprint arXiv:2002.05709*.

Xilun Chen, Ashish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020b. Low-Resource Domain Adaptation for Compositional Task-Oriented Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Noam Chomsky. 1959. *On Certain Formal Properties of Grammars*. Elsevier.

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 Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips Voice Platform: An Embedded Spoken Langauge Understanding System for Private-by-Design Voice Interfaces. *arXiv preprint arXiv:1805.10190*.

Shrey Desai, Akshat Shrivastava, Alexander Zotov, and Ahmed Aly. 2021. Low-resource task-oriented semantic parsing via intrinsic modeling. *arXiv preprint arXiv:2104.07224*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic Parsing for Task Oriented Dialog using Hierarchical Representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vivek Gupta, Akshat Shrivastava, Adithya Sagar, Armen Aghajanyan, and Denis Savenkov. 2021. Retronlu: Retrieval augmented task-oriented semantic parsing. *arXiv preprint arXiv:2109.10410*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS Spoken Language Systems Pilot Corpus. In *Proceedings of the Workshop on Speech and Natural Language*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2004.04906*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for

Natural Language Generation, Translation, and Comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A Comprehensive Multilingual Task-Oriented Semantic Parsing Benchmark. *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

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. *arXiv preprint arXiv:1907.11692*.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2021. X2Parser: Cross-Lingual and Cross-Domain Framework for Task-Oriented Compositional Semantic Parsing. *arXiv preprint arXiv:2106.03777*.

Elman Mansimov and Yi Zhang. 2021. Semantic Parsing in Task-Oriented Dialog with Recursive Insertion-based Encoder. *arXiv preprint arXiv:2109.04500*.

Massimo Nicosia, Zhongdi Qu, and Yasemin Altun. 2021. Translate & fill: Improving zero-shot multilingual semantic parsing with synthetic data. *CoRR*.

Barlas Oğuz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen tau Yih, Sonal Gupta, and Yashar Mehdad. 2021. Domain-matched pre-training tasks for dense retrieval. *arXiv preprint arXiv:2107.13602*.

Panupong Pasupat, Yuan Zhang, and Kelvin Guu. 2021. Controllable semantic parsing via retrieval augmentation. *arXiv preprint arXiv:2110.08458*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *Proceedings of the International Conference on Learning Representations (ICLR): Workshop Track*.

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 (WWW)*.

Akshat Shrivastava, Pierce Chuang, Arun Babu, Shrey Desai, Abhinav Arora, Alexander Zotov, and Ahmed Aly. 2021. Span Pointer Networks for Non-Autoregressive Task-Oriented Semantic Parsing. In *Proceedings of the Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. *arXiv preprint arXiv:2104.08051*.

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 Analysis

A.1 Retrieval vs. Filling

We now turn towards better understanding the aspects our model struggles with. Because RAF jointly optimizes both the retrieval and filling modules, one question we pose is whether the retrieval or filling task is more difficult. We create three versions of RAF: (1) standard retrieval + standard filling, (2) oracle retrieval + standard filling, and (3) standard retrieval + oracle filling. By comparing models (1), (2), and (3), we can judge the relative difficulty of each task.

We begin by evaluating these models in a high-resource setting; on the TOPv2 eval dataset, the standard model gets 87.03%, retrieval oracle gets 96.56%, and filling oracle gets 89.34%. **Here, the gap between models (1)-(2) is +9.53%, while the gap between models (1)-(3) is +2.31%, indicating the retrieval module is the main performance bottleneck.** We also perform experiments in low-resource and multilingual settings, displaying results in Tables 9 and 10, respectively. These results also confirm the same trend: in both settings, the retrieval oracle achieves the best performance, notably achieving +18.3% averaged across 5 multilingual transfer experiments.

Despite retrieval having the most room for improvement, we also see some evidence filling struggles in certain multilingual transfer cases; for example, providing gold spans can improve en$\rightarrow$th transfer by +16.7%. As such, there is ample opportunity for optimizing the retrieval and filling modules in future work.

A.2 Runtime Latency

We compare the latency of the non-autoregressive Span Pointer Networks against RAF to show that there is minimal degradation, despite the improved generalization from RAF. In order to measure latency, we measure wall clock time (ms) of the SpanPointer$_{\text{BASE}}$, RAF$_{\text{BASE}}$, and Autoregressive$_{\text{BASE}}$, all models use a 12-layer RoBERTa$_{\text{BASE}}$ encoder with a 1 layer transformer decoder. We run the benchmark on the TOPv2 source domain evaluation set (17k utterances across 6 domains) on a Tesla V100 GPU, using batch size of 1. In table 11 we report P50, P90, and P99. Our results find that there is less than a 3ms in latency increase in P99 for RAF compared to SpanPointer due to the non-autoregressive nature, despite leveraging our retrieval based model in RAF, and both models are significantly faster than the autoregressive counter part.

B Visualizations

Because RAF encodes utterances and scenarios into a joint embedding space, we can directly visualize this space to further understand our models' inner-workings. We present two case studies: domain development (§B.1) and error analysis (§B.2).

B.1 Domain Development

Figure 3 presents an example of performing domain development on the weather domain. Here, we train RAF with 4 dataset sizes (0 SPIS, 10 SPIS, 25 SPIS, and 1,000 SPIS) to simulate zero-shot, few-shot, low-resource, and high-resource settings, respectively. Each utterance (from the high-resource split) is projected using an utterance encoder and colored according to its gold scenario. Interestingly, the zero-shot setting has multiple, apparent clusters, but the overall performance is poor given many scenarios overlap with each other. The clusters spread further apart as the dataset size increases, suggesting the scenarios become more well-defined.

B.2 Error Analysis

While we have demonstrated how RAF refines the utterance-scenario space as we increase dataset size, we now dive deeper into how each space can be used to further analyze domain semantics. In figure 4, using our high-resource-trained RAF model, we create multiple scenario spaces: each utterance is projected using an utterance encoder and colored according to its predicted frame. We use these scenario spaces in several debugging exercises:

- **Slot Ambiguity:** In Figure 4 (a), we investigate the scenario `[IN:CREATE_ALARM [SL:ALARM_NAME ] [SL:DATE_TIME ]`. Here, we notice a cluster of predictions with the frame `[IN:CREATE_ALARM [SL:DATE_TIME ]` missing the `[SL:ALARM_NAME ]`. These map to utterances such as "I want to wake up at 7 am" where the annotation has "wake up" is `SL:ALARM_NAME`; however, our model does not identify this. There are other examples, such as "wake me up at 7 am", which are annotated without `SL:ALARM_NAME`, leading to ambiguity of whether or not "wake up" is an alarm name.

- **Incorrect Annotations:** In Figure 4 (b), we investigate the scenario `[IN:PLAY_MUSIC`

	Weather Domain (SPIS)							
	0	10	25	50	100	500	1000	Avg
RAF$_{\text{BASE}}$								
Standard Retrieval + Standard Filling	26.19	75.10	78.74	77.53	79.67	87.91	88.17	73.33
Oracle Retrieval + Standard Filling	**81.68**	**90.43**	**92.13**	**91.97**	**93.35**	**95.74**	**96.27**	**91.65**
Standard Retrieval + Oracle Filling	27.67	77.84	81.70	79.92	81.78	89.88	90.44	75.60

Table 9: Evaluating whether retrieval or filling is the most challenging components of RAF in low-resource settings. We fine-tune several variants of RAF, using either a standard / oracle retriever and a standard / oracle filler, on various SPIS splits of the TOPv2 (Chen et al., 2020b) weather domain. RAF with oracle retrieval achieves the best performance, suggesting utterance→scenario retrieval is the most difficult piece to model.

		Zero-Shot Evaluation					
	`en`	`en→es`	`en→fr`	`en→de`	`en→hi`	`en→th`	Avg
RAF$_{\text{BASE}}$							
Standard Retrieval + Standard Filling	81.1	56.0	54.9	40.1	32.1	30.0	42.6
Oracle Retrieval + Standard Filling	**91.0**	**68.9**	**71.6**	**67.5**	**47.5**	**48.9**	**60.9**
Standard Retrieval + Oracle Filling	83.8	66.5	62.8	45.5	40.2	46.7	52.3

Table 10: Evaluating whether retrieval or filling is the most challenging components of RAF in low-resource settings. See Table 9 for a description of our methodology; we use MTOP (Li et al., 2021) for evaluation instead.

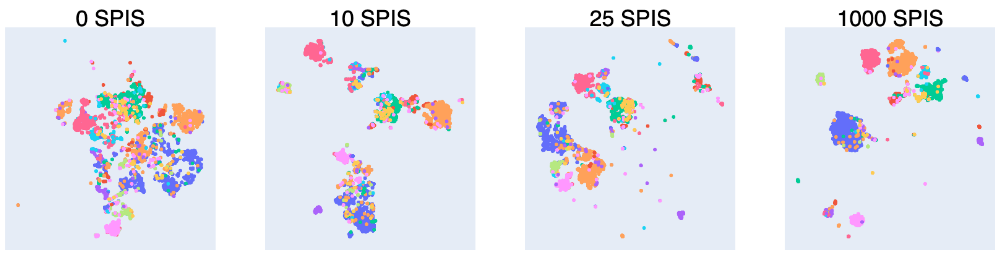

Figure 3: Visualizing the semantic space for the weather domain as the model is trained on more and more data. We visualize the index prior to any training (0 SPIS) to low-resource (10, 25 SPIS) and high resource (1000 SPIS). The graphs are TSNE projections of the utterance vectors used for retrieval color coded by the scenario each utterance belongs to.

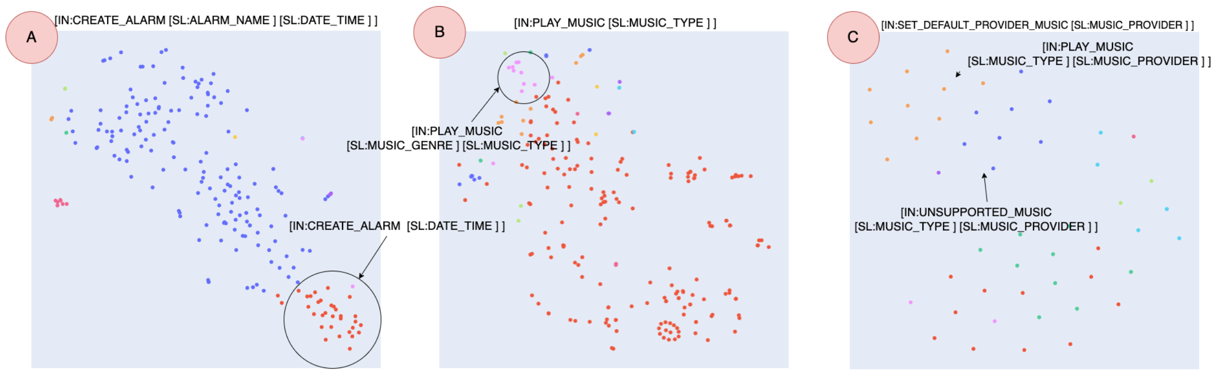

Figure 4: Depicting scenario visualizations, where each is a TSNE projection of utterances belonging to the specified scenario color coded by the predicted scenario. (A) covers the scenario `[IN:CREATE_ALARM [SL:ALARM_NAME ] [SL:DATE_TIME ]` showing a case of ambiguity of alarm names. (B) covers the scenario `[IN:PLAY_MUSIC [SL:MUSIC_TYPE ]` where annotations are missing the slot "music type". (C) covers the scenario `[IN:SET_DEFAULT_PROVIDER_MUSIC [SL:MUSIC_PROVIDER ]` showing how our model requires more support here as the predictions for this scenario span OOD and music domain.

Model	P50	P90	P99
SpanPointer$_{\text{BASE}}$	28.30	29.62	37.78
RAF$_{\text{BASE}}$	28.92	31.40	39.94
Autoregressive$_{\text{BASE}}$	90.29	155.11	228.92

Table 11: Runtime latency (ms) comparison of SpanPointer$_{\text{BASE}}$, RAF$_{\text{BASE}}$, and Autoregressive$_{\text{BASE}}$ using batch size 1 on the TOPv2 source evaluation set.

`[SL:MUSIC_TYPE ]`. Here, we notice a cluster of predictions with the frame `[IN:PLAY_MUSIC [SL:MUSIC_GENRE ] [SL:MUSIC_TYPE ]` adding the `[SL:MUSIC_GENRE ]`. These map to utterances such as “Play 1960s music” where here the annotation only has “music” as `SL:MUSIC_TYPE`, but our model predicts “1960s” as `SL:MUSIC_GENRE`. We believe this is an incorrect annotation in this cluster.

- **Underfitting:** In Figure 4 (c), we investigate the scenario `[IN:SET_DEFAULT_PROVIDER_MUSIC [SL:MUSIC_PROVIDER ]`. This cluster is highly diverse, consisting of predictions from other music intents (e.g., `IN:PLAY_MUSIC`) and out-of-domain intents (e.g., `IN:UNSUPPORTED_MUSIC`). This scenario may need more data in order to be more properly defined.

C Heuristic negatives

In order to understand the importance of model based hard negatives, we develop a simple heuristic to curate a set of hard negative scenarios for each gold scenario. Our heuristic involves selecting the top-N scenarios that share the top level intent but have the lowest Levenshtein edit distance (Levenshtein, 1966) compared to the gold scenario. The full algorithm is described in Algorithm 1.

In §5 and Table 6 we present the full results depicting the importance of model based negative sampling. We show that while our heuristic improves on top of no hard negatives (in-batch negatives only), it still lags behind model based hard negatives (-1.37%).

D Intrinsic Descriptions

In table 12 we provide brief examples of the various representation functions used in RAF training.

Algorithm 1 Heuristic-based negative sampling via edit distance.

```
1: procedure EDIT DISTANCE NEGATIVES
2:     S ← all scenarios
3:     s* ← current scenario
4:     S_same intent ← Scenarios with the same top level intent as Si
5:     heap ← min heap of score and structure
6:     for $S_i \in S_{\text{same intent}}$ do
7:         score ← LevenshteinDistance($s^*, S_i$)
8:         heappush(heap, score, $S_i$)
       return heap
```

In tables 13, 14, 15, and 16 we present the intrinsic hand made descriptions used for each intent/slot on TOPv2 (Chen et al., 2020b) and MTOP (Li et al., 2021) respectively. §3.1.3 describes the various scenario representations used in full detail.

E Hyperparameters

In this section we describe the hyper parameters for training our various RAF models.

Architecture Parameters. For our RAF architectures we leverage a shared RoBERTa (Liu et al., 2019) or XLM-R (Conneau et al., 2020) encoder for both the utterance and scenario encoders. We augment each of these encoders with an additional projection layer with a hidden dimension of 768 (base models) or 1024 (large models). For our span pointer decoder we leverage a 1 layer transformer decoder with the same hidden dimension as the respective encoder (1L, 768/1024H, 16/24A).

Optimization Parameters. We train our models with the Adam (Kingma and Ba, 2015) optimizer along with a warmup and linear decay. We train our models across 8 GPUs with 32GB memory each. Additionally we optionally augment our models with the R3F loss (Aghajanyan et al., 2021) based on validation set tuning in each setting. To determine hyperparameters, we conduct hyperparameter sweeps with 56 iterations each. The hyperparameters for the high resource runs on TOPv2 (Chen et al., 2020b) are described in Table 17.

R Type	R Example
type-only	[intent [slot]]
automatic-span	[add time timer [measurement unit]]
automatic-type-span	[intent | add time timer [slot | measurement unit]]
automatic-type-span-exs	[intent | add time timer [slot | measurement unit | sec / min / hr]]
curated-span	[add time to timer [unit of measurement]]
curated-type-span	[intent | add time to timer [slot | unit of measurement]]
curated-type-span-exs	[intent | add time to timer [slot | unit of measurement | sec / min / hr]]

Table 12: List of intrinsic representations used for encoding scenarios in RAF.

Intent Token	Description
IN:ADD_TIME_TIMER	add time to timer
IN:ADD_TO_PLAYLIST_MUSIC	add music to playlist
IN:CANCEL_MESSAGE	cancel message
IN:CREATE_ALARM	create alarm
IN:CREATE_PLAYLIST_MUSIC	create playlist music
IN:CREATE_REMINDER	create reminder
IN:CREATE_TIMER	create timer
IN:DELETE_ALARM	delete alarm
IN:DELETE_REMINDER	delete reminder
IN:DELETE_TIMER	delete timer
IN:DISLIKE_MUSIC	dislike music
IN:GET_ALARM	get alarm
IN:GET_BIRTHDAY	get birthday
IN:GET_CONTACT	get contact
IN:GET_DIRECTIONS	get directions
IN:GET_DISTANCE	get distance between locations
IN:GET_ESTIMATED_ARRIVAL	get estimated arrival time
IN:GET_ESTIMATED_DEPARTURE	get estimated departure time
IN:GET_ESTIMATED_DURATION	get estimated duration of travel
IN:GET_EVENT	get event
IN:GET_EVENT_ATTENDEE	get event attendee
IN:GET_EVENT_ATTENDEE_AMOUNT	get amount of event attendees
IN:GET_EVENT_ORGANIZER	get organizer of event
IN:GET_INFO_CONTACT	get info of contact
IN:GET_INFO_ROAD_CONDITION	get info of road condition
IN:GET_INFO_ROUTE	get info of route
IN:GET_INFO_TRAFFIC	get info of traffic
IN:GET_LOCATION	get location
IN:GET_LOCATION_HOME	get location of my home
IN:GET_LOCATION_HOMETOWN	get location of my hometown
IN:GET_LOCATION_SCHOOL	get location of school
IN:GET_LOCATION_WORK	get location of work
IN:GET_MESSAGE	get message
IN:GET_RECURRING_DATE_TIME	get recurring date or time
IN:GET_REMINDER	get reminder
IN:GET_REMINDER_AMOUNT	get amount of reminders
IN:GET_REMINDER_DATE_TIME	get date or time of reminder
IN:GET_REMINDER_LOCATION	get location of reminder
IN:GET_SUNRISE	get info of sunrise
IN:GET_SUNSET	get info of sunset
IN:GET_TIME	get time
IN:GET_TIMER	get timer
IN:GET_TODO	get todo item
IN:GET_WEATHER	get weather
IN:HELP_REMINDER	get help reminder
IN:IGNORE_MESSAGE	ignore message
IN:LIKE_MUSIC	like music
IN:LOOP_MUSIC	loop music
IN:NEGATION	negate
IN:PAUSE_MUSIC	pause music
IN:PAUSE_TIMER	pause timer
IN:PLAY_MUSIC	play music
IN:PREVIOUS_TRACK_MUSIC	play previous music track
IN:REACT_MESSAGE	react to message
IN:REMOVE_FROM_PLAYLIST_MUSIC	remove from music playlist
IN:REPLAY_MUSIC	replay music
IN:PREVIOUS_TRACK_MUSIC	play previous music track
IN:REACT_MESSAGE	react to message
IN:REMOVE_FROM_PLAYLIST_MUSIC	remove from music playlist
IN:REPLAY_MUSIC	replay music
IN:REPLY_MESSAGE	reply to message
IN:RESTART_TIMER	restart timer
IN:RESUME_TIMER	resume timer
IN:SELECT_ITEM	select item
IN:SEND_MESSAGE	send message
IN:SEND_TEXT_MESSAGE	send text message
IN:SET_DEFAULT_PROVIDER_MUSIC	set default music provider
IN:SILENCE_ALARM	silence alarm
IN:SKIP_TRACK_MUSIC	skip music track
IN:SNOOZE_ALARM	snooze alarm
IN:START_SHUFFLE_MUSIC	start shuffling music
IN:STOP_MUSIC	stop music
IN:SUBTRACT_TIME_TIMER	subtract time from timer
IN:UNSUPPORTED_ALARM	unsupported alarm request
IN:UNSUPPORTED_EVENT	unsupported event request
IN:UNSUPPORTED_MESSAGING	unsupported messaging request
IN:UNSUPPORTED_MUSIC	unsupported music request
IN:UNSUPPORTED_NAVIGATION	unsupported navigation request
IN:UNSUPPORTED_TIMER	unsupported timer request
IN:UNSUPPORTED_WEATHER	unsupported weather request
IN:UPDATE_ALARM	update alarm
IN:UPDATE_DIRECTIONS	update directions
IN:UPDATE_REMINDER	update reminder
IN:UPDATE_REMINDER_DATE_TIMER	update date time of reminder
IN:UPDATE_REMINDER_TODO	update todo of reminder
IN:UPDATE_TIMER	update timer

Table 13: List of intrinsic handmade descriptions for intents in TOPv2 (Chen et al., 2020b).

Slot Token	Description
SL:AGE	age of person
SL:ALARM_NAME	alarm name
SL:AMOUNT	amount
SL:ATTENDEE	attendee
SL:ATTENDEE_ADDED	attendee to be added
SL:ATTENDEE_EVENT	attendee of event
SL:ATTENDEE_REMOVED	attendee to be removed
SL:ATTRIBUTE_EVENT	attribute of event
SL:BIRTHDAY	birthday
SL:CATEGORY_EVENT	category of event
SL:CATEGORY_LOCATION	category of location
SL:CONTACT	contact
SL:CONTACT_RELATED	contact related
SL:CONTENT_EMOJI	content text with emoji
SL:CONTEnT_EXACT	content text
SL:DATE_TIME	date or time
SL:DATE_TIME_ARRIVAL	date or time of arrival
SL:DATE_TIME_BIRTHDAY	date or time of birthday
SL:DATE_TIME_DEPARTURE	date or time of departure
SL:DATE_TIME_NEW	new date or time
SL:DATE_TIME_RECURRING	recurring date or time
SL:DESTINATION	travel destination
SL:DURATION	duration
SL:FREQUENCY	frequency
SL:GROUP	group
SL:JOB	job
SL:LOCATION	location
SL:LOCATION_CURRENT	current location
SL:LOCATION_HOME	location of my home
SL:LOCATION_MODIFIER	location modifier
SL:LOCATION_USER	location of user
SL:LOCATION_WORK	location of work
SL:MEASUREMENT_UNIT	unit of measurement
SL:METHOD_RETRIEVAL_REMINDER	method of retrieving reminder
SL:METHOD_TIMER	method of timer
SL:METHOD_TRAVEL	method of traveling
SL:MUSIC_ALBUM_TITLE	title of music album
SL:MUSIC_ARIST_NAME	name of music artist
SL:MUSIC_GENRE	genre of music
SL:MUSIC_PLAYLIST_TITLE	title of music playlist
SL:MUSIC_PROVIDER_NAME	name of music provider
SL:MUSIC_RADIO_ID	id of music radio
SL:MUSIC_TRACK_TITLE	title of music track
SL:MUSIC_TYPE	type of music
SL:MUTUAL_EMPLOYER	mutual employer
SL:MUTUAL_LOCATION	mutual location
SL:MUTUAL_SCHOOL	mutual school
SL:NAME_APP	name of app
SL:NAME_EVENT	name of event
SL:OBSTRUCTION_AVOID	obstruction to avoid
SL:ORDINAL	ordinal
SL:ORGANIZER_EVENT	obstruction to avoid
SL:ORDINAL	ordinal
SL:ORGANIZER_EVENT	organizer of event
SL:PATH	path
SL:PATH_AVOID	path to avoid
SL:PERIOD	time period
SL:PERSON_REMINDED	person to be reminded
SL:PERSON_REMINDED_ADDED	added person to be reminded
SL:PERSON_REMINDED_REMOVED	removed person to be reminded
SL:POINT_ON_MAP	point on map
SL:RECIPIENT	message recipient
SL:RECURRING_DATE_TIME	recurring date or time
SL:RECURRING_DATE_TIME_NEW	new recurring date or time
SL:RESOURCE	resource
SL:ROAD_CONDITION	road condition
SL:ROAD_CONDITION_AVOID	road condition to avoid
SL:SEARCH_RADIUS	search radius
SL:SENDER	message sender
SL:SOURCE	travel source
SL:TAG_MESSGE	tag of message
SL:TIMER_NAME	timer name
SL:TIME_ZONE	time zone
SL:TODO	todo item
SL:TODO_NEW	new todo item
SL:TYPE_CONTACT	contact type
SL:TYPE_CONTENT	content type
SL:TYPE_INFO	info type
SL:TYPE_REACTION	reaction type
SL:TYPE_RELATION	relation type
SL:UNIT_DISTANCE	unit of distance
SL:WAYPOINT	waypoint
SL:WAYPOINT_ADDED	waypoint to be added
SL:WAYPOINT_AVOID	waypoint to avoid
SL:WEATHER_ATTRIBUTE	weather attribute
SL:WEATHER_TEMPERATURE_UNIT	unit of temperature

Table 14: List of intrinsic handmade descriptions for slots in TOPv2 (Chen et al., 2020b).

Intent Token	Description
IN:FOLLOW_MUSIC	follow music
IN:GET_JOB	get job
IN:GET_GENDER	get gender
IN:GET_UNDERGRAD	get undergrad education
IN:GET_MAJOR	get college major
IN:DELETE_PLAYLIST_MUSIC	delete playlist
IN:GET_EDUCATION_DEGREE	get education degree of person
IN:GET_AGE	get age of person
IN:DISPREFER	dislike item
IN:RESUME_MUSIC	resume music
IN:QUESTION_MUSIC	question about music
IN:CREATE_CALL	create a caoo
IN:GET_AIRQUALITY	get airquality
IN:GET_CALL_CONTACT	get contact for caller
IN:SET_UNAVAILABLE	set status to unavailable
IN:END_CALL	end call
IN:STOP_SHUFFLE_MUSIC	stop shuffle of music
IN:PREFER	prefer item
IN:GET_LANGUAGE	get language
IN:SET_AVAILABLE	set available
IN:GET_GROUP	get group
IN:ANSWER_CALL	answer call
IN:GET_CONTACT_METHOD	get method to contact
IN:UPDATE_METHOD_CALL	update method of call
IN:GET_ATTENDEE_EVENT	get attendee for event
IN:UPDATE_CALL	update call
IN:GET_LIFE_EVENT	get life event
IN:REPEAT_ALL_MUSIC	repeat all music
IN:GET_EDUCATION_TIME	get education time
IN:QUESTION_NEWS	question about news
IN:GET_EMPLOYER	get employer
IN:IGNORE_CALL	ignore call
IN:REPEAT_ALL_OFF_MUSIC	turn of repeat
IN:UNLOOP_MUSIC	turn loop off
IN:SET_DEFAULT_PROVIDER_CALLING	set default provider for calling
IN:GET_AVAILABILITY	get avalability of contact
IN:HOLD_CALL	hold call
IN:GET_LIFE_EVENT_TIME	get time of life event
IN:SHARE_EVENT	share event
IN:CANCEL_CALL	cancel call
IN:SET_RSVP_YES	set rsvp to yes
IN:PLAY_MEDIA	play media
IN:GET_TRACK_INFO_MUSIC	get information about the current track
IN:GET_DATE_TIME_EVENT	get the date time of the event
IN:SET_RSVP_NO	set rsvp to no
IN:MERGE_CALL	marge call
IN:UPDATE_REMINDER_LOCATION	update the location of the reminder
IN:GET_MUTUAL_FRIENDS	get mutual friends
IN:GET_MESSAGE_CONTACT	get information about message contact
IN:GET_LYRICS_MUSIC	get lyrics about the song
IN:GET_INFO_RECIPES	get information about recipe
IN:GET_DETAILS_NEWS	get news details
IN:GET_EMPLOYMENT_TIME	get employment time
IN:GET_RECIPES	get a recipe
IN:GET_CALL	get call
IN:GET_CALL_TIME	get time of the call
IN:GET_CATEGORY_EVENT	get the category of the event
IN:RESUME_CALL	resume the call
IN:IS_TRUE_RECIPES	ask question about recipes
IN:SET_RSVP_INTERESTED	set rsvp to interested
IN:GET_STORIES_NEWS	get news stories
IN:SWITCH_CALL	switch call
IN:REWIND_MUSIC	rewind the song
IN:FAST_FORWARD_MUSIC	forward the song

Table 15: List of intrinsic handmade descriptions for intents in MTOP (Li et al., 2021).

Slot Token	Description
SL:GENDER	gender of person
SL:RECIPES_TIME_PREPARATION	time to prepare recipe
SL:RECIPES_EXCLUDED_INGREDIENT	exclude ingredient for recipe
SL:USER_ATTENDEE_EVENT	attendee of event
SL:MAJOR	major
SL:RECIPES_TYPE	type of recipe
SL:SCHOOL	school
SL:TITLE_EVENT	title of event
SL:MUSIC_ALBUM_MODIFIER	type of album
SL:RECIPES_DISH	recipe dish
SL:NEWS_TYPE	type of news
SL:RECIPES_SOURCE	source of recipe
SL:RECIPES_DIET	diet of recipe
SL:RECIPES_UNIT_NUTRITION	nutrition unit of recipe
SL:MUSIC_REWIND_TIME	time to rewind music
SL:RECIPES_TYPE_NUTRITION	nutrition type of recipe
SL:CONTACT_METHOD	method to contact
SL:SIMILARITY	similarity
SL:PHONE_NUMBER	phone number
SL:NEWS_CATEGORY	category of news
SL:RECIPES_INCLUDED_INGREDIENT	ingredient in recipe
SL:EDUCATION_DEGREE	education degree
SL:RECIPES_RATING	rating of recipe
SL:CONTACT_REMOVED	removed contact
SL:NEWS_REFERENCE	news reference
SL:METHOD_RECIPES	method of recipe
SL:LIFE_EVENT	life event
SL:RECIPES_MEAL	recipe meal
SL:NEWS_TOPIC	news topic
SL:RECIPES_ATTRIBUTE	recipe attribute
SL:EMPLOYER	employer
SL:RECIPES_COOKING_METHOD	cooking method of recipe
SL:RECIPES_CUISINE	cuisine of recipe
SL:MUSIC_PLAYLIST_MODIFIER	music playlist modifier
SL:RECIPES_QUALIFIER_NUTRITION	nutrition qualifier of recipe
SL:METHOD_MESSAGE	method to send message
SL:RECIPES_UNIT_MEASUREMENT	unit of measurement in recipe
SL:CONTACT_ADDED	added contact
SL:NEWS_SOURCE	news source

Table 16: List of intrinsic handmade descriptions for slots in MTOP (Li et al., 2021).

Parameter	RAF Models		
	RoBERTa$_{\text{BASE}}$	RoBERTa$_{\text{LARGE}}$	XLM-R$_{\text{BASE}}$
Params	134M	372M	288M
Epochs		40	
Optimizer		Adam	
Weight Decay		0.01	
ϵ		1e-8	
Warmup Period (steps)		1000	
Learning Rate Scheduler		Linear Decay	
Learning Rate	0.00002	0.00003	0.00002
Batch Size	40	12	16
Sampled Negatives	1	1	1
$\beta\mathcal{L}_{\text{Retrieval}}$	2.69	4	2.69
$\mathcal{L}_{\text{filling}}$Label Smoothing Penalty		0.2	
# GPU		8	
GPU Memory	32GB	32GB	32GB

Table 17: Hyperparameter values for RAF architectures.