LOW RESOURCE LANGUAGE UNDERSTANDING IN VOICE ASSISTANTS

A Dissertation Outline Presented

by

SUBENDHU RONGALI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2022

Robert and Donna Manning College of
Information and Computer Sciences

LOW RESOURCE LANGUAGE UNDERSTANDING IN VOICE ASSISTANTS

A Dissertation Outline Presented

by

SUBENDHU RONGALI

Approved as to style and content by:

Andrew McCallum, Chair

Mohit Iyyer, Member

Andrew Lan, Member

Konstantine Arkoudas, Member

James Allan, Chair of the Faculty
Robert and Donna Manning College of
Information and Computer Sciences

ABSTRACT

LOW RESOURCE LANGUAGE UNDERSTANDING IN VOICE ASSISTANTS

SEPTEMBER 2022

SUBENDHU RONGALI

B.Tech., INDIAN INSTITUTE OF TECHNOLOGY MADRAS

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew McCallum

Voice assistants such as Amazon Alexa, Apple Siri, and Google Assistant have become ubiquitous. They rely on spoken language understanding, which typically consists of an Automatic Speech Recognition (ASR) component and a Natural Language Understanding (NLU) component. ASR takes user speech as input and generates a text transcription. NLU takes the text transcription as input and generates a semantic parse to identify the requested actions, called intents (play music, turn on lights, etc.) and any relevant entities, called slots (which song to play? which lights to turn on?).

These components require massive amounts of training data to achieve good performance. In this dissertation, I identify and explore various data-related challenges to improve language understanding in voice assistants, specifically, the NLU component and the pipelined ASR-NLU architecture.

I first present a state-of-the-art NLU system based on sequence-to-sequence neural models that simplifies the traditional semantic parsing architecture, while also allow-

ing it to handle complex user utterances consisting of multiple nested intents and slots. This work serves as an anchor for future data-constraint work. Next, I present an architecture to completely replace the pipelined ASR-NLU system with a fully end-to-end system. Our system is jointly trained on multiple speech-to-text and text-to-text tasks, allowing for transfer learning and also creating a shared representation for both speech and text. It outperforms previous pipelined and end-to-end systems, and performs end-to-end semantic parsing on a new domain by only training on a few text-to-text annotated NLU examples. Finally, I demonstrate how to train large sequence-to-sequence NLU systems using a handful of examples by using auxiliary tasks to pre-train various components of the system.

In upcoming work, I propose to explore the paradigm of universal semantic parsing, especially zero-shot domain adaptation. The task of zero-shot domain adaptation aims to parse utterances from a new domain using only documentary information about the new domain but without any additional training data. I propose multiple research directions to build models that can perform zero-shot semantic parsing on a new domain in a fast and efficient manner. I present initial results from this work and describe a research plan to address remaining challenges.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Adoption of intelligent voice assistants such as Amazon Alexa, Apple Siri, and Google Assistant has increased dramatically among consumers in the past few years: as of early 2019, it is estimated that 21% of U.S. adults own a smart speaker, a 78% year-over-year growth [55]. Latest data shows that there are 4.2 billion digital voice assistants worldwide, with the market valued at $10.7 billion [32].

The core intelligence that helps voice assistants function comes from the system that enables them to understand and interpret spoken language. This language understanding system is built to take user speech as input and parse it into an interpretable logical form that captures the semantics of the user request: the actions requested by the user (play music, turn on lights etc.), called intents, as well as any entities that further refine the action to perform (which song to play? which lights to turn on?), called slots.

In current voice assistants, the language understanding system is build as a two-stage pipeline. The first stage is Automatic Speech Recognition (ASR), where user speech is transcribed into a text utterance. The second stage is Natural Language Understanding (NLU), where the text transcription is parsed into the logical form. Figure 1.1 shows this process in detail along with an example text utterance and logical form.

Several advances in ASR and NLU research have contributed to the success of voice assistants and their usability. In NLU particularly, there has been a lot of work in building systems for the style of semantic parsing describe above, referred to as task-

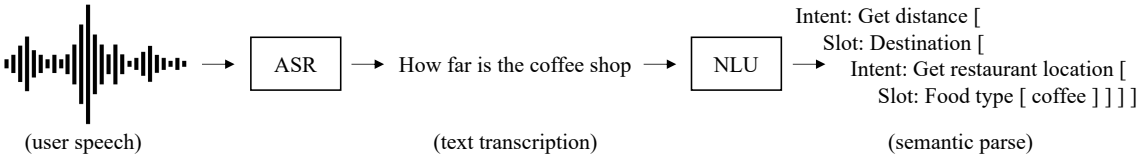

Figure 1.1. Two-stage pipelined ASR-NLU architecture for language understanding in voice assistants. ASR transcribes user speech into text and NLU parses the text into an actionable logical form.

oriented semantic parsing. Current semantic parsing models perform remarkably well after being trained on large amounts of annotated data consisting of user utterances and logical forms. Data annotation is typically a product of extensive manual effort from many crowd workers, making it expensive and time-consuming.

Despite advances, current ASR and NLU systems, and consequently the two-stage voice assistant pipeline, remain extremely data dependant. NLU models still require thousands of annotated examples to achieve good performance. In addition, converting the two-stage pipeline to a single, potentially more efficient, end-to-end model that takes speech as input and produces the logical form is held back to due to data constraints. Further, to increase a voice assistant's capabilities by adding new domains to its capabilities, we require lots of annotated data in the new domain. This task, referred to as domain adaptation, could also potentially entail retraining the whole system.

In this dissertation, I explore how to build good language understanding systems for voice assistants in low-resource settings i.e. with very few annotated examples (few-shot) or no annotated examples (zero-shot).

I first present a state-of-the-art NLU system for task-oriented semantic parsing based on sequence-to-sequence models. This system simplifies traditional semantic parsing architectures that consist of structured generation components by converting the parsing problem it into a sequence-to-sequence task. It hence allows us to seam-

lessly harness the power of large pre-trained language models and provide a base for efficient transfer learning for future low-resource semantic parsing work. Chapter 2, adapted from our published work [69], describes this work in detail.

Next, I present a fully end-to-end language understanding system to replace the two-stage ASR-NLU pipelined architecture in voice assistants. A fully end-to-end system has a number advantages over the two-stage system such as improved latency, minimal error propagation, and better end-to-end model optimization. It was however not previously explored due to the amount of annotated end-to-end data that would be required to get it to work well. Our system, which we call Audio-Text All-Task (AT-AT), is built to jointly train on multiple speech-to-text and text-to-text tasks, allowing it to learn from existing ASR and NLU data in addition to any annotated end-to-end data. Apart from transfer learning, our model is also built to construct a shared representation for speech and text, allowing it to be trained with a few text examples to bootstrap an end-to-end model for a new task or domain. Details of the AT-AT model and demonstration of its ability to perform end-to-end semantic parsing are described in Chapter 3, adapted from our published work [68].

In Chapter 4, I present how to train semantic parsers with very little data i.e. fewer than 50 annotated examples. Our architecture harnesses the power of transfer learning by training various components of the parsing model using easily-available auxiliary tasks. It also leverages ideas from recent semantic parsing work that converts the target logical form into a controlled natural language fragment of text, from which the logical form can be trivially extracted, before training the parsing model. This *naturalization* of the logical form helps us better incorporate pre-trained language models that also contain a pre-trained decoder and amplify transfer learning from auxiliary tasks. This chapter is adapted from our previously published work [67].

I describe proposed upcoming work in Chapter 5. I propose to explore the problem of low-resource NLU domain adaptation, which is a first step towards universal

semantic parsing. My proposed approaches aim to tackle domain adaptation in the zero-shot setting. Specifically, given an existing model and annotated training data for some known domains, we aim to parse utterances from a new domain using only documentary information about the new domain but without any additional training data. Our proposed models are built on our past work on sequence-to-sequence models and auxiliary task transfer learning and are designed to be fast and efficient. I present initial results from this work and describe a research plan to address remaining challenges.

CHAPTER 2

A STATE-OF-THE-ART NATURAL LANGUAGE UNDERSTANDING SYSTEM FOR SEMANTIC PARSING

2.1 Introduction

Traditional approaches for task-oriented semantic dialog parsing frame the problem as a slot filling task. For example, given the query *Play the song don't stop believin by Journey*, a traditional slot filling system parses it in two independent steps: (*i*) It first classifies the *intent* of the user utterance as `PlaySongIntent`, and then (*ii*) identifies relevant *named entities* and tags those slots, such as *don't stop believin* as a `SongName` and *Journey* as an `ArtistName`. Traditional semantic parsing can therefore be reduced to a text classification and a sequence tagging problem, which is a standard architecture for many proposed approaches in literature [43, 49, 38]. This is shown in Figure 2.1.

With increasing expectations of users from virtual assistants, there is a need for the systems to handle more complex queries – ones that are composed of multiple intents and nested slots or contain conditional logic. For example, the query *Are there any movie in the park events nearby?* involves first finding the location of *parks* that are *nearby* and then finding relevant *movie* events in them. This is not straightforward in traditional slot filling systems. Gupta et al. [23] and Einolghozati et al. [14] proposed multiple approaches for this using a Shift-reduce parser based on Recurrent Neural Network Grammars [13] that performs the tagging.

In this chapter, we propose a unified approach to tackle semantic parsing for natural language understanding based on Transformer Sequence to Sequence models

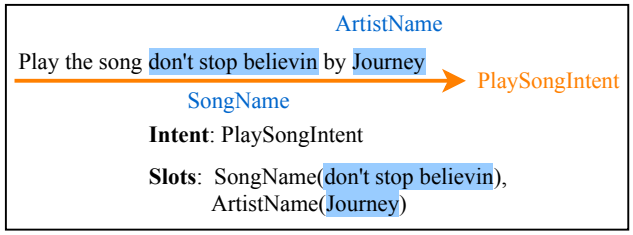

Figure 2.1. Semantic parsing of a "simple" query. Simple queries define single action (intent) and can be decomposed into a set of non-overlapping entities (slots).

[81] and a Pointer Generator Network [82, 74]. Furthermore, we demonstrate how our approach can leverage pre-trained resources, such as neural language models, to achieve state of the art performance on several datasets. In particular, we obtain relative improvements between 3.3% and 7.7% over the best single systems on three public datasets (SNIPS [7], ATIS [63] and TOP [23], the last consisting of complex queries); on two internal datasets, we show relative improvements of up to 4.9%. The sequence to sequence architecture also allows us to easily adapt the architecture to perform transfer learning using additional auxiliary tasks to help train models more effectively in low-resource scenarios.

Furthermore, our architecture can be easily used to parse queries that do not conform to the grammar of either the slot filling or RNNG systems. Some examples include semantic entities that correspond to overlapping spans in the query, and entities comprising of non-consecutive spans. We do not report any results on these kinds of datasets but we explain how to formulate the problems using our architecture.

In summary, our contributions are as follows.

- We propose a new architecture based on Sequence to Sequence models and a Pointer Generator Network to solve the task of semantic parsing for understanding user queries.

- We describe how to formulate different kinds of queries in our architecture. Our formulation is unified across queries with different kinds of tagging.

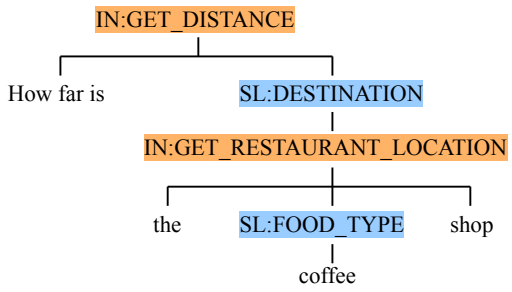

Figure 2.2. Semantic parse for a "complex" query in the Facebook TOP dataset. This complex query is represented as a tree containing two nested intents and slots.

- We achieve state-of-the-art results on three public datasets and two internal datasets.

2.2 Methodology

We propose a unified architecture to solve the task of semantic parsing for both simple and complex queries. This architecture can also be adapted to handle queries containing slots with overlapping spans. It consists of a Sequence to Sequence model and a Pointer Generator Network. We choose a pretrained BERT [10] model as our encoder. Our decoder is modeled after the transformer decoder described in Vaswani et al. [81] and is augmented with a Pointer Generator Network [82, 29] which allows us to learn to generate pointers to the source sequence in our target sequence. Figure 2.3 shows this architecture parsing an example query. We train the model using a cross-entropy loss function with label smoothing.

In this section, we first describe how we formulate queries and their semantic parses as sequences with pointers for our architecture. We then describe our encoder and decoder components.

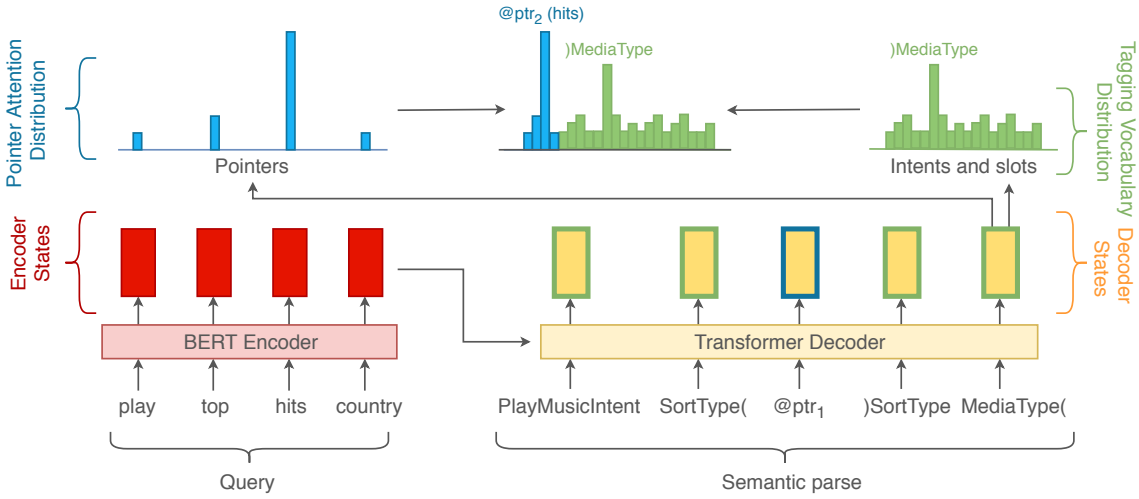

Figure 2.3. Our architecture - Sequence to Sequence model with Pointer Generator Network (Seq2Seq-Ptr). The model is currently decoding the symbol after MediaType(by looking at the scores over the tagging vocabulary and the attentions over the source pointers. It generates $@ptr_2$ since it has the highest overall score.

2.2.1 Query Formulation

A Sequence to Sequence architecture is trained on samples with a source sequence and a target sequence. When some words in the target sequence are contained in the source sequence, they can be replaced with a separate *pointer* token that points to that word in the source to be able to apply the Pointer Generator Network.

Take the example query from Figure 2.1. In our architecture, we use the query as our source sequence. The target sequence is constructed by combining the intent with all the slots, in order, with each slot also containing its source words. The source and target sequences now look as follows.

Source: *play the song don't stop believin by journey*
Target: *PlaySongIntent SongName(* $@ptr_3$ $@ptr_4$ $@ptr_5$ *)SongName*
ArtistName($@ptr_7$ *)ArtistName*

Here, each token $@ptr_i$ is a pointer to the i^{th} word in the source sequence. So $@ptr_3$, $@ptr_4$ and $@ptr_5$ point to the song words *don't stop believin*, and $@ptr_7$ points to the artist word *journey*. The slots have open and close tags since they are enclosing a consecutive span of source tokens. The intent is just represented as a single tag at

the beginning of the target sequence. We can do this for simple queries since they consist of just one intent. The target vocabulary hence consists of all the available intents, two times the number of different slots, and the pointers.

Complex queries with multiple intents and nested slots can also be transformed easily into this formulation. Figure 2.2 shows an example from the Facebook TOP dataset along with its parse tree. This query *How far is the coffee shop* can be converted into our formulation as follows.

Source: *How far is the coffee shop*
Target: [*IN*:*GET_DISTANCE* @ptr_0 @ptr_1 @ptr_2
[*SL*:*DESTINATION* [*IN*:*GET_RESTAURANT_LOCATION* @ptr_3
[*SL*:*TYPE_FOOD* @ptr_4 *SL*:*TYPE_FOOD*] @ptr_5
IN:*GET_RESTAURANT_LOCATION*] *SL*:*DESTINATION*] *IN*:*GET_DISTANCE*]

We made a minor modification to the reference parses from the TOP dataset for our formulation. We replaced the end-brackets with custom end-brackets corresponding to the intent or slot they close. We found that this formulation helped our models perform better.

Finally, we show how we can express queries from datasets that don't conform to either the slot-filling or Shift-reduce systems. Take the following example from the healthcare domain, where the task is to extract a patient diagnosis and related information from a clinician's notes.

Source: *The pt. was diagnosed with GI upper bleed today.*
Annotations: *Bleeding_Event (GI bleed),*
Anatomical_Site (upper)

A traditional slot filling system wouldn't know which non consecutive slots to combine, while a shift-reduce parser cannot split the middle word into a separate tag. In our architecture, we simply formulate the target sequence as follows.

Target: *Bleeding_Event(* @ptr_5 @ptr_7 *)Bleeding_Event,*
Anatomical_Site(@ptr_6 *)Anatomical_Site*

2.2.2 BERT Encoder

Language model pretraining has been shown to improve the downstream performance on many NLP tasks [60, 64, 10]. The idea is to train a language model on a large amount of text using a next word prediction objective to learn good representations for each of the words. These representations can then be fine-tuned on a given NLP task to improve the performance of an existing model. Pretrained models improve the performance of task models since they already contain a lot of useful semantic information learned through the pretraining phase. This has even more significance when the task-specific dataset is fairly small. Some examples of pretrained models in literature include word embeddings such as Word2Vec [51] and Glove [59], and contextualized representations such as ELMo [60], OpenAI-GPT [64], and BERT [10].

We choose BERT to encode the source sequence in our architecture. BERT (Bidirectional Encoder Representations from Transformers) is a language representation model architecture based on Transformers [81]. The original publicly available model was pretrained on a millions of lines of text from BooksCorpus and English Wikipedia. Unlike other language models (ELMo, OpenAI-GPT), which are trained to predict the next token given the previous sequence of words, BERT uses a composite objective that combines masked word prediction and next sentence prediction.

BERT's architecture is based on a multi-layer bidirectional Transformer, originally implemented in Vaswani et al. [81]. The detailed implementation of this architecture can be found in Devlin et al. [10]. For our experiments, we use three different variants of BERT.

For the three public datasets, we used the checkpoint released by Devlin et al. [10]. Experiments on the two internal datasets were carried out using a model we pretrained over a large sample of queries from the live traffic of Amazon Alexa. We also experimented with a publicly-available variant of BERT called RoBERTa [44].

RoBERTa (A Robustly Optimized BERT Pretraining Approach) uses the same architecture as BERT but changes the pretraining process. The next sentence prediction objective is removed and a dynamic masking scheme is used instead of a static one like the original BERT implementation. RoBERTa was also trained with longer sequences, higher batch-sizes, and for a longer time, and was reported to match or exceed the performance of BERT in several NLP benchmarks. The detailed implementation can be found in Liu et al. [44]. Finally as an ablation study, we experimented with an encoder with no pretrained weights.

2.2.3 Decoder with Pointer Generator Network

We use the transformer decoder proposed in Vaswani et al. [81] in our architecture. The self-attention mechanism in the decoder learns to attend to target words before the current step, as well as all the source words in the encoder.

We set up the decoder with different numbers of units, layers, and attention-heads for different tasks based on the size and complexity of the queries. These details are provided in the experiments section.

In a traditional Sequence to Sequence model, the target words are generated from the decoder hidden states through a feed-forward layer that obtains unnormalized scores over a target vocabulary distribution. In our architecture, we use a Pointer Generator Network to generate two different kinds of target words: words from the target vocabulary consisting of parse symbols (the intent and slot delimiters), and words that are simply pointers to the source sequence. Our Pointer Generator Network is based on the models in Vinyals et al. [82] and See at al. [74], and is closest in implementation to Jia and Liang [29].

We now describe our decoding process. For each input source sequence $[x_1 \dots x_n]$, we use the BERT encoder to encode it into a sequence of encoder hidden states

$[e_1 \dots e_n]$. Having generated the first $t-1$ output tokens, the transformer decoder generates the token at step t as follows.

First, the decoder produces the decoder hidden state at time t, d_t by building multi-layer multi-head self-attention on the encoded output as well as the embeddings of the previously generated output sequence as described in Vaswani et al. [81]. We feed d_t through a dense layer to produce scores $[s_1, \dots s_{|V|}]$ for each word in the vocabulary V. V contains all symbols necessary for the parse (intents, slots) but not regular words appearing on the source side.

We also use d_t as a query and compute unnormalized attention scores $[a_1 \dots a_n]$ with the encoded sequence. Concatenating the unnormalized attention scores (size n) and the output of the dense layer (size $|V|$), we obtain an unnormalized distribution over $|V|+n$ tokens, the first $|V|$ of which are the output parsing vocabulary and the last n of which are the $@ptr_i$ $(0 < i < n)$ words pointing to the source tokens. We then feed this through a softmax layer to obtain the final probability distribution. This probability is used in the loss function during training and will be used to choose the next token to generate during inference. Since the transformer decoder uses embeddings of previously generated tokens, we use a set of special embeddings to represent $@ptr_i$ tokens.

In the example in Figure 2.3, we are trying to predict a target word after the token `'MediaType('` at step 5. As shown in the figure, we compute the scores $[a_1 \dots a_4]$ (blue, left) over each of the source tokens, and the scores $[s_1 \dots s_{|V|}]$ (green, right) over the parsing vocabulary. We expect the model to produce the highest score for a_3, which corresponds to $@ptr_2$, representing the word *hits*.

2.3 Evaluation

We evaluate our models on multiple datasets and compare it against strong baselines. This sections covers information about the datasets, baselines, experimental methodology, and results.

2.3.1 Datasets

We test our approach on five different datasets (three publicly available, two interal), which we describe in this section.

2.3.1.1 Facebook TOP

The Task Oriented Parsing (TOP) [23] dataset contains complex hierarchical and nested queries that make the task of semantic parsing more challenging. It contains around 45k annotations with 25 intents and 36 slots, randomly split into 31k training, 5k validation and 9k test utterances. The dataset mainly consists of user queries about navigation and various public events. An example from this dataset can be seen in Figure 2.2.

The `IN:` prefix stands for intent while `SL:` is for slot. We can see how there are multiple intents and nested slots in the semantic interpretation. This makes the query much harder to interpret and parse using a simple slot tagging model that tags each word with a single slot.

2.3.1.2 SNIPS

The SNIPS dataset [7] is a public dataset that is used for training and testing semantic parsing models for voice assistants. It consists of utterances that belong to seven different intents: SearchCreativeWork, GetWeather, BookRestaurant, PlayMusic, AddToPlaylist, RateBook, and SearchScreeningEvent. Each intent contains around 2000 examples to train and 100 to test.

This dataset contains only simple queries with single intents and flat slots. An example is *Will there be fog in Tahquamenon Falls State Park*, where the intent is `GetWeather` and the slots are `condition_description` for *fog* and `geographic_poi` for *Tahquamenon Falls State Park*.

The dataset was originally used to evaluate models in the Snips Voice Platform. It has since been a widely used dataset to benchmark the performance of various task-oriented parsing models.

2.3.1.3 ATIS

The Airline Travel Information System (ATIS) [63] corpus is a widely used dataset in spoken language understanding. It was built by collecting and transcribing audio recordings of people making flight reservations in the early 90s. It consists of simple queries.

There are seventeen different goals or intents such as `Flight` or `Aircraft capacity`. This distribution is however skewed, with the `Flight` intent covering about 70% of the total queries. An example from this dataset consists of the query *How much is the cheapest flight from Boston to New York tomorrow morning?* The intent is `Airfare`, while the slots tag important information like the departure and arrival cities, and the departure times.

The ATIS corpus has supported research in the field of spoken language understanding for more than twenty years. Some researchers have performed extensive error analysis on the state of the art discriminative models for this dataset and reported that despite really low error rates, there exist many unseen categories and sequences in the dataset that can benefit from incorporating linguistically motivated features [79]. This supports the continued utility of ATIS as a research corpus.

2.3.1.4 Internal Datasets

Our internal datasets consist of millions of user utterances that are used to train and test Amazon Alexa. For our experiments, we sampled two datasets of utterances, one from the music domain and the other from video domain. Utterances in these domains naturally included a large amount of entities (e.g. artists and albums names, movie and video titles), and thus represent a good benchmark for the ability of any neural model to generalize over a diverse set of queries. The example in Figure 2.3 is from the music domain.

The sampled music domain dataset contains 6.2M training and 200k test utterances, with 23 intents and 100 slots. The video domain dataset contains 1M training and just 5k test utterances; parses in this dataset are comprised of 24 distinct intents and 59 slots.

2.3.2 Baseline Models

We benchmark our performance on the internal datasets by comparing it to a well tuned RNN based model. The model learns to perform joint intent and slot tagging using a bidirectional LSTM and a Conditional Random Field (CRF) [28]. We further enhanced this baseline by replacing its embedding and encoder layers with a language model pretrained on a subset of Alexa's live traffic. These components were fine-tuned on the two datasets described in Section 2.3.1.4.

For the ATIS and SNIPS datasets, we use the top four performing methods reported by Zhang et al. [95] as baselines. All these models perform joint intent and slot tagging. There are two variants that use RNNs: a simple RNN based model, and an RNN model augmented with attention. There is also a model that works completely with just attention, the slot gated full attention model. The final baseline, CapsuleNLU, uses Capsule Networks [71].

For the TOP dataset, we pick a model based on Recurrent Neural Network Grammars (RNNG) [13], the Shift Reduce Parser. We provide a brief overview of this model as described in Gupta et al. [23] - the parse tree is constructed using a sequence of transitions, or *actions*. The transitions are defined as a set of SHIFT, REDUCE, and the generation of intent and slot labels. SHIFT action consumes an input token (that is, adds the token as a child of the right most *open* sub-tree node) and REDUCE closes a sub-tree. The third set of actions is generating non-terminals: the slot and intent labels. The model learns to perform one of these actions at each step in time.

We report scores of three experimental setups with the shift reduce parser from Einolghozati et al. [14]: a simple shift reduce parser, a shift reduce parser augmented with ELMo embeddings, and an ensemble of these models augmented with ELMo and an SVM language model reranker.

2.3.3 Experimental Setup

All our models were trained on a machine with 8 NVIDIA Tesla V100 GPUs, each with 16GB of memory. When using pretrained encoders, we leveraged gradual unfreezing to effectively tune the language model layers on our datasets. We used the "BASE" variant of BERT and RoBERTa encoders, which uses 768-dimensional embeddings, 12 layers, 12 heads, and 3072 hidden units. When training from scratch, we used a smaller encoder consisting of 512-dimensional embeddings, 6 layers, 8 heads, and 1024 hidden units.

Depending on the dataset, we used either a 128 units, 4 layers, 3 heads, and 512 hidden units decoder (Facebook TOP, ATIS, SNIPS) or a larger 512 units, 6 layers, 8 heads, and 1024 hidden units decoder (internal Music and Video datasets). We used bi-linear product attention to score the source words in the Pointer Network.

While training, the cross entropy loss function was modified with label smoothing with $\epsilon = 0.1$. We used the Adam [35] optimizer with noam learning rate schedule

Method	Accuracy	
	exact match	intent
Shift Reduce (SR) Parser [14]	80.86	–
SR with ELMo embeddings [14]	83.93	–
SR ensemble + ELMo + SVMRank [14]	**87.25**	–
SEQ2SEQ-PTR (no pretraining)	79.25	97.43
SEQ2SEQ-PTR (BERT encoder)	83.13	97.91
SEQ2SEQ-PTR (RoBERTa encoder)	86.67	**98.13**

Table 2.1. Results on **TOP** [23]. Our system (SEQ2SEQ-PTR) outperforms the best single method (SR + ELMo) by 3.3%. It is close to the best ensemble approach (SR + ELMo + SVRank).

[81], each adjusted differently for different datasets. At inference time, we used beam search decoding with a beam size of 4.

2.3.4 Results and Discussion

We use exact match (EM) accuracy as the main metric to measure the performance of our models across all datasets. Under this metric, the entire semantic parse for a query has to match the reference parse to be counted as correct. Because EM is generally more challenging than slot-level precision and recall or semantic error rate [78], it is better suited to compare high performing systems like the ones studied in this work. For completeness, we also report the intent classification accuracy for our models.

The results from our experiments are documented in Tables 2.1-2.5. Our models match or beat the baselines across all datasets on both exact match and intent classification accuracies. We see significant improvements on both simple and complex datasets.

2.3.4.1 Complex Queries

We achieve an improvement of 2.7 (+3.3%) EM accuracy points on the TOP dataset over the state-of-the-art single model on this dataset (Table 2.1). Our SEQ2SEQ-

Method	Accuracy	
	exact match	intent
Joint BiRNN [25]	73.20	96.90
Attention BiRNN [43]	74.10	96.70
Slot Gated Full Attention [21]	75.50	97.00
CapsuleNLU [95]	80.90	97.30
Seq2Seq-Ptr (no pretraining)	85.43	97.00
Seq2Seq-Ptr (BERT encoder)	86.29	**98.29**
Seq2Seq-Ptr (RoBERTa encoder)	**87.14**	98.00

Table 2.2. Results on **SNIPS** [7]. Our system (Seq2Seq-Ptr) outperforms the previous state of the art by 7.7%.

Ptrmodel with RoBERTa encoder is only surpassed by the ensemble model reported in Einolghozati et al. [14] (+0.6% EM accuracy points.)

In addition, we find that even without specifying any hard requirements for the grammar of the parse trees in the complex queries, 98% of the generated parses are well formatted. For the simple query datasets, it was greater than 99% but difference is expected since the grammar is easier to learn there.

During error analysis, we found an interesting example in the TOP dataset where we believe our model generates a valid, more meaningful parse than the reference annotation. For the query *What time do I need to leave to get to Helen by 8pm*, our model parses *Helen* as `[SL:DESTINATION [IN:GET_ LOCATION_HOME [SL:CONTACT Helen ] ] ]`, while it is annotated as `[SL:DESTINATION Helen ]`. Our parse resolves the query as finding the estimated departure time to get to a location that is the home location of a contact named Helen, while the reference annotation suggests that the correct interpretation is to find the estimated departure time to get to a destination named *Helen*. We believe our parse is more likely to be correct given that Helen is most likely the name of a person.

Method	Accuracy	
	exact match	intent
Joint BiRNN [25]	80.70	92.60
Attention BiRNN [43]	78.90	91.10
Slot Gated Full Attention [21]	82.20	93.60
CapsuleNLU [95]	83.40	95.00
Seq2Seq-Ptr (no pretraining)	81.08	95.18
Seq2Seq-Ptr (BERT encoder)	86.37	97.42
Seq2Seq-Ptr (RoBERTa encoder)	**87.12**	**97.42**

Table 2.3. Results on **ATIS** [63]. Our system (Seq2Seq-Ptr) outperforms the previous best method by 4.5%.

2.3.4.2 Simple Queries

We report results of our sequence to sequence model (Seq2Seq-Ptr) on four datasets (SNIPS, ATIS, internal music, internal video) that contain simple queries in Tables 2.2, 2.3, 2.4, and 2.5.

On the SNIPS and ATIS datasets, we note that the best version of our method (Seq2Seq-Ptrwith RoBERTa encoder) achieves a significant improvement in EM accuracy over existing baselines (+7.7% and +4.5% respectively.) Using a BERT encoder causes a slight decrease in performance, but still achieves a meaningful improvement over the previous state of the art [95]; this is consistent with what has been observed on other NLP tasks [44]. If no pretraining is used, performance is further reduced but it is notable that this variant still beats all the baselines on the SNIPS dataset.

For our internal Alexa datasets, we note that the proposed Seq2Seq-Ptrmethod obtains comparable results to a BiLSTM-CRF tagger on the music domain, and slightly better EM accuracy (+1.9%) on the video domain. We believe our model wasn't able to outperform the baseline on the music domain because the entities in this domain are very diverse, especially song or album names. Sequence tagging methods therefore benefit from having to solve a simpler task of having to tag each word in the sequence, as opposed to our unconstrained model. We would however

Method	Accuracy	
	exact match	intent
BiLSTM-CRF (no pretraining)	*baseline*	
BiLSTM-CRF (pretrained LM)	**+3.0%**	**+0.1%**
Seq2Seq-Ptr (no pretraining)	-0.3%	-0.7%
Seq2Seq-Ptr (BERT encoder)	-2.2%	-0.8%
Seq2Seq-Ptr (RoBERTa encoder)	-3.5%	-0.7%

Table 2.4. Results on an internal Amazon dataset (music). The best configuration of our method (Seq2Seq-Ptr) is comparable to a BiLSTM-CRF pretrained on a large conversational dataset.

Method	Accuracy	
	exact match	intent
BiLSTM-CRF (no pretraining)	*baseline*	
BiLSTM-CRF (pretrained LM)	+3.0%	**+0.1%**
Seq2Seq-Ptr (no pretraining)	+2.9%	-0.1%
Seq2Seq-Ptr (BERT encoder)	+0.1%	-0.2%
Seq2Seq-Ptr (RoBERTa encoder)	**+4.9%**	-0.2%

Table 2.5. Results on an internal Amazon dataset (video). The best configuration of our method (Seq2Seq-Ptr) outperforms a pretrained BiLSTM-CRF network by 1.9%.

like to note that our from-scratch variants beat the from-scratch baselines on both domains. Also curiously, the performance of Seq2Seq-Ptrwith a BERT encoder fell behind that of a sequence to sequence model trained from scratch. Since the scratch model uses a smaller transformer encoder (6 layers with 8 heads per layer instead of 12/12), we believe it was able to converge more effectively than the BERT encoder.

2.4 Related Work

The task of semantic parsing for intent and slot detection is well established in literature. Traditionally, this was done with slot filling systems that classify the query and then label each word in the query. There were a few approaches that followed this system, using Recurrent Neural Networks [43, 49]. Researchers have also experimented with Convolutional Neural Networks and showed good results [34] and more recently, Capsule Networks [71, 95].

Prior to the advent of deep learning models, the task of sequence labeling was tackled with the use of Conditional Random Fields (CRF) [38, 31, 58]. CRFs learn pairwise potentials on labeling subsequent words which allow models to find more probable label sequences for a given query.

Most of this work is valid for semantic parsing for simple queries which boils down to a sequence labeling task. To handle more complex cases with hierarchical slots such as the example in Figure 2.2, researchers have experimented with Sequence to Sequence models and models based on Recurrent Neural Network Grammars (RNNG) [13]. RNNGs were shown to perform better on complex queries than RNN or Transformer-based Sequence to Sequence models [23]. Researchers have also explored models involving logical forms and discourse for language representation [42, 94, 80]. The Pointer Generator Network in our architecture was introduced in Vinyals et al. [82]. It was used in NLP applications where some words from the source sequence reappeared in the target sequence such as text summarization and style transfer [74, 57, 62]. They were also used to copy out of vocabulary words from the source to target in machine translation [36]. Our implementation of the Pointer Network is closest to the architecture in Jia and Liang [29]. By using pointers to represent the source tokens and imposing no particular logical form over our target sequence, we can handle any kind of queries for parsing. This makes our architecture as expressive as logical forms, while also being able to learn as easily as simple slot tagging systems.

2.5 Contributions

We propose a unified architecture for the task of semantic parsing for different kinds of queries. We show that our architecture matches or outperforms existing approaches across multiple datasets : internal music and video datasets, SNIPS,

ATIS, and Facebook TOP. We significantly outperform the current state of the art models on the public datasets TOP (3.3%), SNIPS (7.7%), and ATIS (4.5%).

We describe how to apply this architecture to both simple queries and complex queries with hierarchical and nested slots. We also describe how to formulate any set of queries with non-conforming grammars to work with our architecture, making this model applicable to many different types of semantic parsing. By creating a sequence to sequence architecture for semantic parsing, we pave way for future low resource work using transfer learning and auxiliary tasks.

2.6 Collaboration Statement

This chapter is adapted from Rongali et al. [69]. The work was done in collaboration with Luca Soldaini, Emilio Monti, and Wael Hamza from Amazon Alexa AI.

CHAPTER 3

AT-AT: AN END-TO-END SYSTEM FOR SPOKEN LANGUAGE UNDERSTANDING

3.1 Introduction

Voice assistants are built on complex Spoken Language Understanding (SLU) systems that are typically too large to store on an edge device such as a mobile phone or a smart speaker. Hence, user traffic is routed through a cloud server to process requests. This has led to privacy concerns and fueled the push for tiny AI and edge processing, where the user requests are processed on the device itself.

Traditional SLU systems consist of a two-stage pipeline, an Automatic Speech Recognition (ASR) component that processes customer speech and generates a text transcription *(ex. play the song watermelon sugar)*, followed by a Natural Language Understanding (NLU) component that maps the transcription to an actionable hypothesis consisting of intents and slots *(ex. Intent: PlaySong, Slots: SongName - watermelon sugar)*. An end-to-end (E2E) system that goes directly from speech to the hypothesis would help make the SLU system smaller and faster, allowing it to be stored on an edge device. It could potentially also be better optimized than a pipeline since it eliminates cascading errors.

However, E2E systems are not used in practice because they have some key issues. These systems are hard to build since they consist of large neural components such as transformers and require massive amounts of E2E training data. They also don't make use of the vastly available training data for the ASR and NLU components that could be used to enhance their performance, because the examples in these

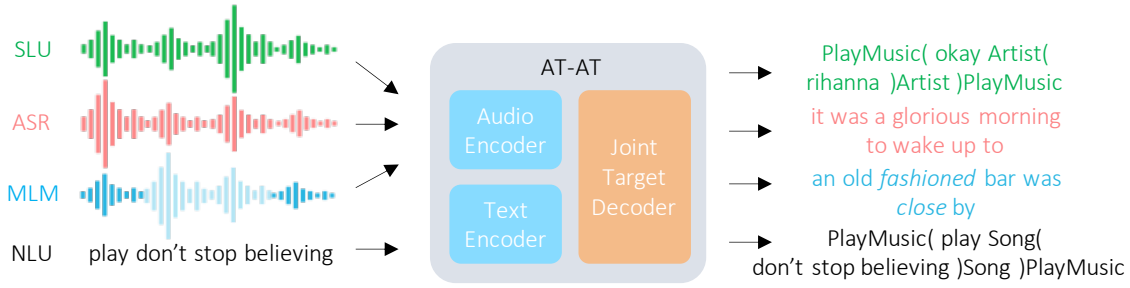

Figure 3.1. Pretraining AT-AT with audio-to-text and text-to-text tasks. The audio and text inputs go to separate encoders but share a joint decoder, which decodes the target sequence based on the task. Task labels are passed as BOS tokens while decoding.

datasets may not be aligned to create an E2E training sample. Another issue is feature expansion, a scenario where a new domain, with new intents and slots, is added to the voice assistant's capabilities. Here, developers typically only have access to some synthetically generated text-hypothesis examples. Speech data isn't readily available and it is very expensive to collect. E2E models thus fail as they require lots of new audio and hypothesis data to learn this new domain.

In this work, we build an E2E model that mitigates these issues using transfer learning. We call it the Audio-Text All-Task (AT-AT) Model. AT-AT is an E2E transformer-based model that is jointly trained on multiple audio-to-text and text-to-text tasks. Examples of these tasks include speech recognition (ASR), hypothesis prediction from speech (SLU), masked LM prediction (MLM), and hypothesis prediction from text (NLU). Our model achieves this by converting data from all these tasks into a single audio-to-text or text-to-text format. Figure 3.1 shows this joint training phase in detail. Our findings indicate that there is significant knowledge transfer taking place from multiple tasks, which in turn helps in downstream model performance. We see that the AT-AT pretrained model shows improved performance on SLU hypothesis prediction on internal data collected from Alexa traffic. We also

report state-of-the-art results on two public datasets: FluentSpeech [46], and SNIPS Audio [70].

Furthermore, since our model contains a text encoder, it can consume both audio and text inputs to generate a target sequence. By jointly training on both audio-to-text and text-to-text tasks, we hypothesize that this model learns a shared representation for audio and text inputs. This allows us to simply train on new text-to-text data and get audio-to-text performance for free, giving us a way to do E2E hypothesis prediction in a zero-shot fashion during feature expansion. We test this approach on an internal dataset from Alexa traffic, and an external dataset, Facebook TOP [23]. Since TOP consists of only text data, we collected speech data for the test split using an internal tool at Amazon. We release this dataset.

In summary, our contributions are as follows.

- We developed an E2E SLU model that is jointly trained on multiple audio-to-text and text-to-text tasks and shows knowledge transfer and SLU performance improvements.
- We report state-of-the-art results on two public SLU datasets, FluentSpeech and SNIPS Audio.
- We show how to perform zero-shot E2E hypothesis prediction with our model.
- We report a new benchmark for zeroshot E2E SLU on the Facebook TOP dataset and release the test data.

3.2 Related Work

The architecture of prior E2E SLU models is taken from neural speech recognition literature. Speech recognition was originally performed using hidden Markov models that predict acoustic features, followed by word-level language models [17]. More recently, deep learning models have become more popular for this task [27]. Deep

learning models solve this task by posing it as a sequence-to-sequence problem [22, 54]. With the success of transformer-based sequence-to-sequence models on text based tasks [81], researchers have explored and shown success in applying them for speech recognition [52, 33]. Our architecture is based on these models.

Other end-to-end SLU models also closely resemble this sequence-to-sequence encoder-decoder framework [24, 46]. The slot-filling task for SLU is formulated as a target text sequence by wrapping the target English tokens with intent and slot tags, which was shown to achieve state of the art results [69]. Our approach improves upon these models by introducing transfer learning. The transfer learning paradigm we adopt here is similar to prior efforts that use multiple tasks or pretraining to improve SLU performance [84, 30]. The audio-text shared training idea also has prior work. However, these efforts require parallel audio-text data [8], or are evaluated on a simpler classification task [72].

Zeroshot E2E SLU, where we only have text NLU training data but no audio has also been explored. Recently, [45] approached this task using speech synthesis. They generate synthetic speech from text using a Text to Speech (TTS) system and use the resultant audio to train their models. While this approach is simple and intuitive, its success greatly depends on access to a good TTS system. We propose a method that can perform this task, end-to-end, without any TTS system, and can also be used in conjunction with a TTS system to further improve performance.

Finally, an important part of all these models is the representation of audio. The raw audio waveform is typically converted into higher level features before being passed to the actual models. While Mel-Frequency Cepstral Coefficitents (MFCC) have been the traditional choice for this conversion, Log-filterbank features (LFB) have become more popular recently [15]. We use LFB features here.

3.3 The AT-AT Model

In this section, we explain the design of our proposed Audio-Text All-Task (AT-AT) model. AT-AT is trained to jointly perform multiple audio-to-text and text-to-text tasks.

We hypothesize that AT-AT will benefit from potential knowledge transfer in a multi-task setting. This is in line with findings in a recent work [65] that converts a variety of text based natural language tasks into source and target text sequences and shows knowledge transfer by using a single shared sequence-to-sequence model. AT-AT can also be used as a pretrained checkpoint to build end-to-end models on new datasets to achieve better performance. Finally, we believe that AT-AT is a powerful audio-text shared representation model that would allow us to do E2E zeroshot prediction using just text data.

When training AT-AT with audio tasks, the input audio signal is pre-processed to obtain a sequence of LFB features, which is taken as the source sequence. For text tasks, the source sequence is simply the text input tokens. The target consists of a sequence of tokens corresponding to the task being solved. For example, the target sequence is a sequence of words if the task is speech recognition. If the task is SLU or NLU hypothesis prediction, the target consists of the intent and slot tags as well as the words within them, a formulation based on recent work that solves this task as a sequence-to-sequence problem [69]. An example set of source-target sequences for tasks is shown in Figure 3.1. We pass the task label as the beginning-of-sequence (BOS) token in the target decoder. This way, the model can conditionally decode the target sequence based on the observed input and the task being solved. Note that previous multi-task text-to-text models [65] add this information to the source sequence itself. Since our source sequence can be in the audio space, we add the task label at the start of the target sequence.

While the audio encoder trained on multiple audio-to-text tasks presents an obvious transfer learning advantage in SLU, our reasons for incorporating a text encoder in this model are two-fold; first, we can add more text-to-text tasks in the pretraining phase, and second, more importantly, this would enable us to train on a task with only text-to-text data and expect good audio-to-text performance. AT-AT thus has the ability to do zero-shot end-to-end SLU by training on only annotated text data, an important ability that comes in handy during feature expansion, where new intents and slots need to be added to the model without any audio data available. This situation arises because the text data for new intents and slots can be synthetically generated but the audio data is not readily available and is expensive to collect.

A model can develop the zero-shot ability if the audio and text inputs share a common space from which the target sequence is generated. A common way to learn a shared space from two input sources is to explicitly impose an L2 loss penalty on the hidden state vectors of the two aligned input sources [8]. This is however infeasible in our setup because the hidden states from the audio and text input sequences are not single vectors, but sequences of vectors of different lengths and resolution. While we can pool these vectors to get a single vector, doing so would result in a huge information bottleneck which makes the decoder incapable of decoding the target sequence well. We resolve this problem by avoiding the explicit vector alignment altogether, hence eliminating any need to pool the encoder hidden states. We use a single shared decoder to process the hidden state vectors of both the audio and text encoder. By constraining the complexity of this decoder, we force it to learn a shared representation between audio and text so that it can solve both tasks without solving them separately.

AT-AT consists of two phases of training: 1) the pretraining phase, where we train our model on multiple audio-to-text and text-to-text tasks, and 2) the finetuning

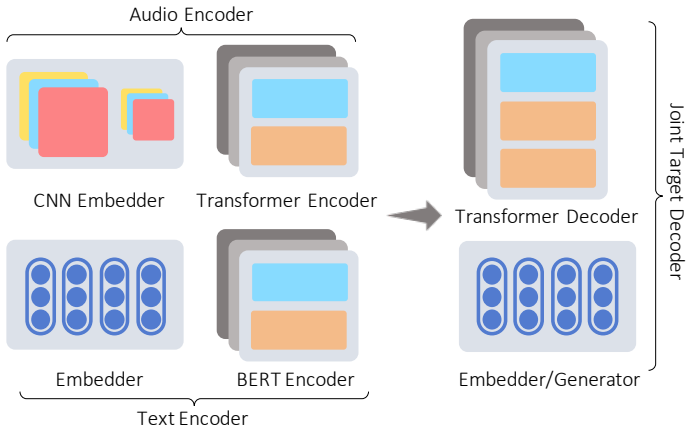

Figure 3.2. Model Components of AT-AT. The audio encoder consists of a CNN embedding layer and a transformer encoder to encode the audio spectrograms. The text encoder consists of a token embedding layer and a transformer encoder, similar to most pretrained language models such as BERT. There is only one decoder that decodes text from both encoders and it is a transformer decoder with tied embedding/generator weights.

phase, where we finetune our model on a single downstream task. The architecture of AT-AT, these two phases, and our zeroshot end-to-end approach are described below.

3.3.1 Architecture

AT-AT has an architecture similar to many transformer-based speech recognition models proposed recently [33, 52], which contain an encoder-decoder framework to process a source audio sequence and decode the target text sequence. In addition to the audio encoder, our model also contains a text encoder to process text sequences.

The audio encoder consists of multiple convolutional and max pooling layers to contextually embed the audio LFB spectrogram. This is followed by a transformer encoder [81]. These convert the input audio sequence into a much shorter sequence of hidden states to be consumed by the decoder. For the text encoder, we use BERT [10], which consists of an embedder to embed the input tokens and their positions, followed by a transformer encoder. The text encoder typically produces hidden states that are larger in size than the audio encoder so we use a projection layer to project

the text hidden states down to match the dimensionality of the audio hidden states. Once this is done, both the text and audio sequences generate a sequence of hidden states of the same size.

We use a single transformer decoder to decode the targets from the sequence of encoder hidden states. Both the text and audio inputs go through the same generation process, which allows the model to learn a shared representation without any explicit loss penalty to align them.

We use byte-pair encoding (BPE) to split the target words into smaller pieces. We only split the target English words, not any tokens corresponding to intent and slot tags. The target sequence tokens are embedded using a standard embedding matrix. The transformer decoder consumes the current token embedding and performs a multi-head multi-layer attention over the encoder hidden states to generate a decoder hidden state. The decoder hidden state is passed through a generator layer that shares weights with the embedding matrix. The generator layer assigns a probability mass to each token in the target vocabulary, representing the probability of that token being generated next. Further details on this decoder framework are beyond the scope of this paper and can be found in [81]. Note that instead of a fixed BOS token to start decoding as usual, we use the task label as the BOS token. Figure 3.2 lays out these components.

3.3.2 Pretraining Phase

The pretraining phase of AT-AT consists of training with multiple audio-to-text and text-to-text sequence-to-sequence tasks. Examples from all tasks are randomly sampled in each batch during pretraining. Figure 3.1 shows the pretraining phase in action, where we train with three audio-to-text tasks: SLU hypothesis prediction (SLU), automatic speech recognition (ASR), masked-audio LM prediction (MLM), and one text-to-text task: NLU hypothesis prediction (NLU). For the MLM task,

the audio corresponding to certain words is masked out and the model is trained to predict the whole target sequence. We perform audio-word alignment prior to the masking using an external tool; more details on this are in the datasets section. We require at least one audio-to-text and one text-to-text task if the model will be used to do zeroshot E2E prediction.

3.3.3 Finetuning Phase

In the finetuning phase, we start from the pretrained model and train it on a specific downstream task, such as SLU. We hypothesize that pretraining with multiple tasks allows the model to transfer knowledge from different tasks, allowing it to be better regularized and obtain a warm start for optimization for the downstream task.

When the pretrained model is used as a starting point for new datasets with new intents and slots, unseen target token embeddings are randomly initialized. The model is first trained by freezing all pretrained parameters so that these new parameters get to a good optimization point. They are then gradually unfrozen over time as the model is finetuned.

3.3.4 Zeroshot End-to-End

In the zeroshot scenario, we have access to a new annotated text-to-text dataset and we want to construct an E2E model capable of predicting the target sequence given audio input. It is a common occurrence in the feature expansion phase in voice assistants, where a new domain is added to the voice assistant's capabilities. For example, say a voice assistant is currently capable of handling user requests in music and shopping domains. We want to add the capability for it to handle requests in a new domain, say books, such as reading a book. In this case, developers usually write down some launch phrases and annotate them to perform a certain task in the new domain. The audio data for these phrases doesn't exist yet. The goal is to

bootstrap an E2E model that can process audio data and generate the hypothesis by just training on the text data.

AT-AT allows us to do this easily when it is pretrained on a certain task from both audio and text inputs. In the voice assistant feature expansion case for example, the pretraining phase is carried out with an SLU task on existing domains, an NLU task on existing domains, and any other tasks we want to add such as ASR and MLM. Once the pretraining is complete, we simply finetune the model using the annotated text NLU data from the new domain and test on audio data.

While this is one way to train E2E models without audio data, another way is to simply generate the missing audio data using a Text-to-Speech (TTS) system and use it for training. This approach is however contingent on the availability of a good TTS system. With AT-AT, we can perform zeroshot prediction without a TTS system. Moreover, when we do have access to a TTS system, we can add the generated synthetic audio to the finetuning phase and finetune AT-AT on both the synthetic audio and text. We hypothesize that this is better than simple E2E training since the text NLU data helps train the language model within the decoder even better, allowing AT-AT to work harmoniously with synthetic audio to further improve performance.

3.4 Evaluation

3.4.1 Datasets

Our experiments are carried out on a combination of internal and publicly available datasets. We describe them here.

3.4.1.1 Internal SLU Data

Our internal dataset is created by sampling utterances from user traffic of our voice assistant, Alexa. This is done in compliance with user commitments with regards to

privacy and anonymity. We select only utterances from the music domain for the first set of experiments. This dataset contains about 3M training utterances, 100k validation, and 100k testing utterances comprising 23 intents and 95 slots. Each utterance here contains the audio, text transcript, and the SLU hypothesis.

For our low-resource experiments, we sample 10% of utterances from the above dataset and select the audio and hypotheses. We pick the text transcriptions from the rest to create data for the ASR task during AT-AT pretraining.

3.4.1.2 LibriSpeech ASR Data

We also compile an ASR dataset by downloading all splits of the publicly available LibriSpeech dataset [56], giving us ∼1000 hours of data. This data is comprised of multiple speakers reading sentences from audio books in the LibriVox project.

3.4.1.3 MLM Data

We create the dataset for the MLM task by modifying the LibriSpeech dataset. We first use an external audio alignment tool, Gentle[1] that is built on the Kaldi framework [61]. Once this is done, we mask 15% of the words in each transcript and the corresponding audio in the audio file. This masked audio is then processed to produce the LFB features to produce the audio input and the target sequence is the entire transcript.

3.4.1.4 Public SLU Datasets

We also evaluate AT-AT on public SLU datasets to compare with the state-of-the-art results. We use two public datasets: FluentSpeech [46], and SNIPS Audio [70] in our evaluation. The FluentSpeech dataset consists of target sequences that are 3-tuples, not sequences. We convert them into target sequences using some pre-

[1]https://github.com/lowerquality/gentle

processing rules to create data in the required format. More details on data processing are given in Appendix A. There are about 23k train, 3k valid, and 4k test examples in this dataset.

The annotations in SNIPS are in the form of intents and slots, so can be trivially converted into target sequences in the required format. We use the smart-lights close-field and far-field datasets from the SNIPS dataset for our experiments and report results with 5-fold cross validation since there are no explicitly delineated train-test splits. These dataset are extremely small, each consisting of a total of 1660 examples.

3.4.1.5 Zeroshot SLU Datasets

For our zeroshot experiments, we require text NLU training data and audio SLU test data on an unseen domain. We collect two datasets for this. First is an internal dataset that consists of utterances sampled from Alexa traffic in the books domain. We extract around 200k text NLU training examples, and 10k audio SLU test examples comprising 21 intents and 47 slots.

We also construct a zeroshot dataset from the publicly available Facebook TOP [23] dataset. This is a challenging dataset that contains complex utterances with nested intents and slots. It contains ∼32k train, 4k eval, and 9k test utterances. We want to evaluate the performance of AT-AT on this dataset to show its effectiveness in a complete domain shift. With TOP, we use the training and validation data splits as is. Using an internal utterance collection tool, we collected audio data for a fraction of the test split, about 1915 utterances from multiple speakers, to test zeroshot performance. We release this dataset[2].

For the zeroshot experiments, one of our baselines is an E2E model built by generating synthetic speech data from a TTS System. We use Amazon Polly[3] as our

[2]https://subendhurongali.netlify.app/publication/atat/

[3]https://aws.amazon.com/polly/

TTS system. We use 9 randomly selected speakers and the neural engine to create speech data for utterances. Further details on synthetic data generation can be found in Appendix A.

3.4.2 Experimental Details

We use 80-dim LFB features to process the audio signals. The target English words were tokenized using byte-pair encoding to obtain a final vocabulary of 5k.

We use a 2-layer 2D CNN with 256 final units and a transformer encoder with 12 layers, 4 heads, 256 units, and 2048 hidden units as our audio encoder. The text encoder is the standard BERT-base encoder [10]. The target decoder consists of a 256-dim tied embedding/generator matrix and a transformer decoder with 6 layers, 4 heads, 256 units, and 2048 hidden units. We use noam learning rate schedule with 4000 warm-up steps and an adam optimizer with learning rate 1. We use cross entropy loss with label smoothing ($\epsilon = 0.1$) as our loss function. During inference, we use beam search with a beam-size of 4.

When finetuning with gradual unfreezing, we use a learning rate multiplier of 0 for the first 500 steps, and 0.2, 0.5, 0.7 for the next 100 steps each, finally reaching 1 after 800 steps and training normally from there on. We didn't perform extensive hyper-parameter tuning for our experiments.

3.4.3 AT-AT in Low Resource Settings

Our first set of experiments evaluate the effect of AT-AT multi-task training on improving the performance of an E2E model trained on a low-resource annotated dataset. To simulate the low resource setting, we take our internal music SLU dataset and sample 10% of the data to obtain the speech and SLU annotations. For the rest of the examples, we obtain the ASR transcripts to create the ASR dataset for the multi-task training. Our AT-AT model is pretrained on these two tasks. We evaluate this

Method	SemER	EM Accuracy
E2E Model with 100% data	*baseline*	*baseline*
E2E Model with 10% data	+8.63	-11.82
AT-AT, Pretrained (10%)	+2.13	-2.52
AT-AT, Finetuned (10%)	**+1.45**	**-1.49**

Table 3.1. Results on an internal Amazon music dataset in the low-resource setting. Our AT-AT models vastly improve over a simple E2E model trained on 10% of the data and almost catch up to a model trained on 100% data.

model's performance on the test set immediately after pretraining. We then perform the finetuning step on just the 10% SLU data and perform another evaluation.

3.4.3.1 Baselines

We train two E2E models as baselines. These models have the same architecture as our AT-AT model, without the multi-task component or the text encoder. The first model is trained on the full internal music SLU dataset. The second model is trained on the extracted 10% dataset. We expect our AT-AT model, that makes use of the additional ASR data from music to recuperate any drop in performance between these two models.

3.4.3.2 Results

Table 3.1 shows the results of these experiments. We report two metrics here, the semantic error rate (SemER), and the exact match (EM) accuracy. Exact match accuracy simply corresponds to the accuracy obtained by matching the entire predicted hypothesis to the gold hypothesis. SemER is a more slot-filling oriented metric that rewards partially correct hypotheses. It is an internal metric that is used to evaluate the performance of SLU models built for Alexa. Given the number of insertion (I), deletion (D), and substitution (S) errors in the predicted hypothesis, it is given by $\frac{\text{S+I+D}}{\text{\# total slots + 1 (for intent)}}$. We want a lower SemER and a higher EM accuracy. Due to

internal regulations, we do not report the absolute numbers on internal datasets. For this experiment, we use the performance numbers of the E2E model trained on 100% data as the baseline and report the remaining numbers relative to it.

We observe that there is a big drop in performance when we train a model on 100% data vs 10% data. The SemER increases by 8.63 absolute points. However, our AT-AT model, pretrained with additional ASR data recuperates most of this performance, mitigating this increase in error to only 2.23 points. Finetuning on the SLU data further improves performance, giving us a error increase of just 1.45 points. We see a similar trend in the exact match accuracy scores as well where our models lose the least number of accuracy points. These results show that multi-task training with additional ASR data is hugely beneficial in a low-resource scenario.

3.4.4 Building Better E2E Models with AT-AT

The previous experiment showed that the performance of an E2E model trained on a low-resource dataset (10% data) can be improved by adding additional ASR data and training in a multi-task setting with AT-AT. In this experiment, we want to take this a step further and evaluate if we can improve the performance of a model trained on the full 100% dataset using any available external data. We pretrain AT-AT with the full 100% SLU dataset and in addition, include two more tasks: ASR and MLM. We use the LibriSpeech ASR and MLM datasets as described in the datasets section for these two tasks. Note that these datasets are from a completely different domain than music. We want to determine whether we can improve the performance of an E2E model by adding tasks from other domains with transferable knowledge.

We evaluate our model in two settings. The first setting consists of pretraining with 2 tasks, SLU and ASR, followed by finetuning on the 100% SLU dataset. The second setting's pretraining phase consists of 3 tasks, SLU, ASR, and MLM, followed by finetuning again on the 100% SLU dataset. Our baseline is the E2E model trained

Method	SemER	EM Accuracy
E2E Model w. 100% data	*baseline*	*baseline*
Production ASR, linear chain CRF	+0.61	-1.02
Production ASR, BiLSTM + CRF	-0.45	+0.70
AT-AT, SLU + ASR	**-1.16**	**+1.39**
AT-AT, SLU + ASR + MLM	-1.01	+1.23

Table 3.2. Results on an internal Amazon music dataset with 100% data. Our AT-AT models trained with multiple tasks outperform a simple E2E model and other production baselines.

on 100% music SLU data from the previous set of experiments. For context, we also report numbers from two 2-stage pipeline models for SLU. We use a production-level ASR system from Amazon for the first stage. For the second (NLU) stage, we experiment with a linear chain CRF and a pretrained BiLSTM + CRF (SOTA). The BiLSTM + CRF model beats transformer-based models for this dataset [69].

3.4.4.1 Results

We report the results of these experiments in Table 2.2. We again report relative numbers here since this is in an internal dataset. We use the performance of the E2E 100% model as the baseline. We see that adding LibriSpeech ASR data and pretraining with AT-AT improves SemER on the internal music SLU test set by 1.16 points, representing a significant relative error reduction. The exact match accuracy also improves by 1.4 absolute points here. With all three tasks, we see that the SemER improves by 1.01 points, slightly worse than the previous number. We believe the lack of further improvement from the MLM task might be because it doesn't contribute new information to the model.

3.4.5 AT-AT on Public Datasets

In this set of experiments, we evaluate how AT-AT's pretraining can help improve performance on other datasets. We selected the publicly available FluentSpeech and SNIPS Audio datasets to compare to state-of-the-art models.

We use the AT-AT model pretrained with 2 tasks from the previous experiment and finetune it on the FluentSpeech and SNIPS datasets. We also trained end-to-end models from scratch on these two datasets. To perform an ablation on the AT-AT finetuning approach, we report an additional number on the SNIPS dataset, for a model that uses a pretrained AT-AT audio encoder. This model, compared to the full AT-AT model would give us an idea of how much the decoder pretraining helps, in addition to the encoder pretraining.

3.4.5.1 Baselines

For the FluentSpeech dataset, we compare to two SOTA models. The first model is the best model from [46]. It is a multi-layer RNN-based network, with lower layers trained to predict aligned word targets from the LibriSpeech dataset. The final task is formulated as a 3-way classification task, not a generation task like our AT-AT model. The second model is a transformer-based pretrained model from [84].

For the SNIPS Audio dataset, we compare with the two models reported in [70], SNIPS and Google. The SNIPS model consists of a pipe-lined approach with an acoustic model for ASR, followed by a language model, and slot tagging model for NLU. The Google model is from Google's DialogFlow cloud service[4].

3.4.5.2 Results

Table 3.3 reports the results on the FluentSpeech dataset. We report error rate on the complete hypothesis (HypER), exact match accuracy on the hypothesis, and the

[4]https://cloud.google.com/dialogflow

Method	HypER	EM Accuracy	
		Hyp	Full
E2E Model	8.3	91.7	83.4
SOTA 1 [46]	1.2	98.8	–
SOTA 2 [84]	1.0	99.0	–
AT-AT	**0.5**	**99.5**	**99.0**

Table 3.3. Results on **FluentSpeech** [46]. AT-AT outperforms the previous state-of-the-art approaches and achieves 50% error reduction.

Method	EM Accuracy	
	Hyp	Full
Close Field		
SNIPS [70]	84.22	–
Google [70]	79.27	–
E2E Model	*no convergence*	
E2E Model with pretrained. AT-AT encoder	81.87	53.90
AT-AT	**84.88**	**66.51**
Far Field		
SNIPS [70]	71.67	–
Google [70]	73.43	–
E2E Model	*no convergence*	
E2E Model with pretrained AT-AT encoder	67.83	38.92
AT-AT	**74.64**	**53.25**

Table 3.4. Results on **SNIPS Audio** [70]. AT-AT achieves state-of-the-art performance, beating pipeline-based approaches.

Method	EM Accuracy	Precision	Recall	F1	Tree validity
Synthetic Test Set					
E2E Model with Synthetic Audio	78.73	80.84	79.27	80.05	99.13
AT-AT zeroshot	56.52	62.47	57.98	60.14	98.42
AT-AT zeroshot + Synthetic Audio	**80.37**	**82.35**	**81.30**	**81.82**	**99.56**
Real Test Set					
E2E Model with Synthetic Audio	69.19	67.24	65.15	66.18	98.85
AT-AT zeroshot	51.54	51.31	49.80	50.55	98.96
AT-AT zeroshot + Synthetic Audio	**70.60**	**67.98**	**66.39**	**67.18**	**99.37**

Table 3.5. Results on the TOP audio test set we compiled. On both the synthetic and real test utterances, our AT-AT model shows remarkable zeroshot performance and when trained with additional synthetic audio data, it outperforms a simple E2E model trained with the same data.

exact match accuracy on the full target sequence. While the end-to-end model doesn't perform too well from scratch, our AT-AT finetuned model beats the state-of-the-art model by 0.5 accuracy points. This corresponds to a 50% error reduction.

Table 3.4 contains the results on the SNIPS dataset. We report the exact match accuracy on the hypothesis and the full target sequence here. We see that our AT-AT pretrained models have the best performance on both the close-field and far-field sets with a 5-fold cross validation setup. They beat both Google and SNIPS models' numbers previously reported. We also see that the AT-AT model is vastly superior to an end-to-end model with a pretrained audio encoder. This is especially evident with the accuracy scores on the full target sequence where the AT-AT model beats it by 10-15 absolute points. Note that we weren't able to train an end-to-end model from scratch due to extremely small dataset size.

3.4.6 Zeroshot E2E with AT-AT

In the final experiments, we evaluate the performance of AT-AT on zeroshot end-to-end tasks. Here, we only have text training data and we want to evaluate on speech.

Method	SemER	EM Accuracy
E2E Model with Real Audio	*baseline*	*baseline*
E2E Model with Synthetic Audio	+5.05	-9.58
AT-AT zeroshot	+11.90	-15.14
AT-AT zeroshot + Synthetic Audio	**+3.31**	**-5.20**

Table 3.6. Results on an internal Amazon books dataset. Our AT-AT model again achieves remarkable zeroshot performance and when trained with additional synthetic data, beats a simple E2E model trained with the same data.

We first pretrained AT-AT on 4 tasks: SLU (speech-to-hypothesis), ASR, MLM, and NLU (text-to-hypothesis). We use data from the internal music dataset (for SLU and NLU), and the LibriSpeech dataset for ASR and MLM. This model is then finetuned on the internal books dataset and the Facebook TOP dataset with the text NLU training data as described in the architecture section. We also finetune AT-AT in another setting, using text NLU training data along with the synthetic speech data from our TTS system. We want to show that in addition to performing zeroshot prediction without access to a TTS system, we can also work with an existing TTS system to further improve performance.

3.4.6.1 Baselines

For the internal books dataset, we built an E2E model on real audio training data to obtain a rough upper bound and gauge the zeroshot performance. In addition to this, we also trained another E2E model on the synthetic dataset constructed from our TTS system.

For the TOP dataset, we don't have real audio training data, so our baseline was an E2E model trained on the synthetic training data. For a fair comparison to AT-AT, we use the same pretrained audio encoder for these E2E models.

3.4.6.2 Results

Tables 3.5 and 3.6 report results of the zeroshot experiments. On the internal books dataset, we report relative numbers for SemER and EM accuracy. We use the performance numbers of an E2E model trained on real speech data as baselines. Using synthetic training data gives us a SemER of *baseline* + 5.05 points . AT-AT achieves a zeroshot SemER of *baseline* + 11.90 without access to a TTS system, a respectable number compared to the aforementioned model. AT-AT when finetuned with additional synthetic speech data beats an E2E model trained on only synthetic data, obtaining a SemER of *baseline* + 3.31 (lowest increase in error).

On the TOP dataset, we report all the recommended metrics given in the dataset but we are primarily interested in exact match accuracy. Note that our test set was compiled by recording speech for a fraction of the full test set. We observe the same trend here that we observe on the books dataset. An E2E model trained on synthetic data achieves an accuracy of 69.19 while AT-AT achieves 51.54. While there is a significant drop, it is to be noted that AT-AT sees absolutely no new labeled audio instances, giving it a significant disadvantage while switching input models during inference. It also doesn't require a TTS system for this training. The E2E model is however beaten by the AT-AT model finetuned with additional synthetic data, which achieves an accuracy of 70.60 (2% relative improvement).

3.5 Contributions

We propose the Audio-Text All-Task (AT-AT) model that uses transfer learning to improve the performance on end-to-end SLU. AT-AT beat the performance of E2E models on our internal music data, both in the full and low-resource settings. It also achieved state-of-the-art performance on the FluentSpeech (99.5% EM Accuracy) and SNIPS audio datasets (84.88% close-field, 74.64% far-field EM) with significant improvements over prior models. AT-AT also demonstrated its ability to perform

zeroshot E2E SLU, without access to a TTS system, and by learning a shared audio-text representation without any explicit loss penalty to force the audio and text hidden states into the same space. We also showed how AT-AT can work in conjunction with a TTS system to further improve E2E performance. It achieves a zeroshot E2E EM Accuracy of 70.60 on the TOP dataset.

On a closing note, we would like to remark that AT-AT somewhat mimics actual human learning. We typically read a lot more words than we hear. But when we hear a word for the first time, we transfer our knowledge of that word from when we read it. AT-AT similarly learns to understand and perform NLU tagging from text and then applies this knowledge when it is given speech.

3.6 Collaboration Statement

This chapter is adapted from Rongali et al. [68]. The work was done in collaboration with Beiye Liu, Liwei Cai, Konstantine Arkoudas, Chengwei Su, and Wael Hamza from Amazon Alexa AI.

CHAPTER 4

TRAINING SEMANTIC PARSERS WITH VERY LITTLE DATA

4.1 Introduction

Semantic parsing is the task of mapping a natural-language utterance to a structured representation of the meaning of the utterance. Often, the output meaning representation is a formula in an artificial language such as SQL or some type of formal logic. Current SOTA semantic parsers are seq2seq architectures based on very large language models (LMs) that have been pretrained on vast amounts of natural-language text [69, 14]. To better capitalize on that pretraining, various researchers have proposed to reformulate semantic parsing so that the output meaning representation is itself expressed in natural—instead of a formal—language, albeit a controlled (or "canonical") fragment of natural language that can then be readily parsed into a conventional logical form (LF). We refer to this reformulation as the *naturalization* of semantic parsing.

Naturalizing a semantic parser has significant advantages because the reformulated task involves natural language on both the input and the output space, making it better aligned with the pretraining LM objective. However, even with large-scale LM pretraining, fine-tuning these models requires lots of data, and producing complex annotations for semantic parsing is expensive. There has hence been great interest in *few-shot* semantic parsing, where we only have access to a few annotated examples [75, 90].

Techniques such as in-context learning and prompting [75], where the parsing task is posed as a structured natural language input to an LM, have shown very promising

results in few-shot scenarios when used with extremely large (and access-restricted) pretrained LMs such as GPT-3 (175B parameters). However, the size and inaccessibility of these models makes their use infeasible at present. Task-specific fine-tuning of smaller LMs remains the best-performing approach that is practically feasible, and is the one we pursue in this paper. We propose a simple but highly effective methodology for few-shot training of naturalized semantic parsers that can be used with smaller and more ecologically friendly LMs (we use BART-Large, which has fewer than 0.5B parameters) and can be quickly applied to bootstrap a high-performing semantic parser with less than 50 annotated examples and a modest number of unlabeled examples, which are typically readily available (and can be synthesized).

Our methodology is based on a judicious composition of four techniques: joint training of the semantic parsing task with masking and denoising LM objectives; constrained decoding, made possible because the canonical fragment of natural language is generated by a simple grammar; self-training; and paraphrasing. For training dataset sizes ranging from $n = 16$ to $n = 200$, our method consistently outperforms previous BART-based and GPT-2-based few-shot SOTA results on all domains of the Overnight dataset, in some cases delivering relative improvements exceeding 100%. For $n = 200$, our method catches up to and slightly outperforms in-context learning with GPT-3. We also provide results on Pizza, a new semantic parsing dataset, where we demonstrate relative improvements over BART-based SOTA architectures ranging from 20% to 190%.

We start with the best-performing finetuned naturalized model from Shin et al. [75] as our baseline. This model is based on BART [40], which was chosen by the authors because both the encoder and the decoder are pretrained. They also constrain their decoder to produce only valid canonical forms, using a method that filters valid next tokens. The authors showed that these techniques greatly improve model robustness and allowed models to train with just a few hundred examples.

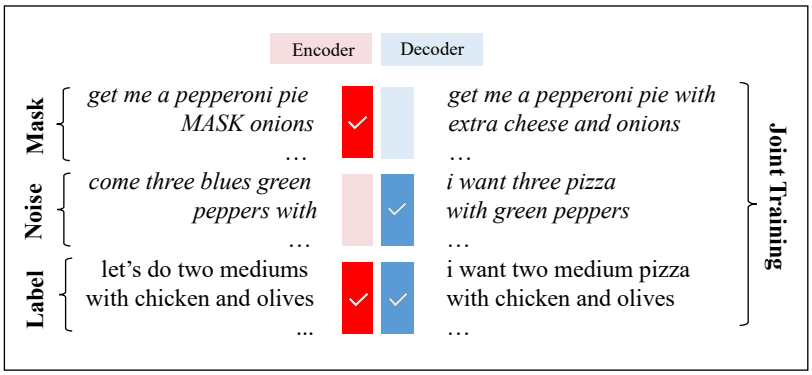

Figure 4.1. Jointly training a seq2seq model using mask prediction, denoising, and supervised semantic parsing examples. The mask prediction examples help train the encoder and the denoising examples help train the decoder, in addition to the supervised examples that train the full model.

We pursue the same general direction here but propose a general methodology that leverages modest amounts of unannotated data to deliver very significant improvements over that baseline model without needing additional effort from model developers. Specifically, we use unlabeled user utterances to create a masked prediction task, which allows the encoder to see and learn to encode utterances of interest. We then add random noise to a generated target dataset to produce noisy source sequences and create an additional denoising task. This task trains the decoder to produce canonical forms effectively. We merge the source and target sequences from both of these tasks along with the original labeled set and train a BART model. Figure 4.1 illustrates this process. During inference, we use constrained decoding, which ensures that we only generate valid canonical forms. By augmenting the dataset with additional examples that effectively adapt the encoder and decoder, we observe massive improvements in semantic parsing accuracy over the baseline models that are only fine-tuned on the labeled dataset. Apart from joint training (JT for short), our method uses self-training [48, 20], or ST for short, and paraphrase augmentation [16, 90]. Here, we take the model from the JT step and label all the unlabeled utterances with constrained decoding. We also paraphrase all our utterances to create

more data and label them in the same way. We then repeat the JT step with this enlarged self-labeled dataset, the original golden labeled dataset, and the masked and noised datasets. Since the self-labeling is done in a constrained manner, the labels are corrected if our model slightly strays from the golden parses. Injecting this knowledge back into the model helps it improve even further.

4.2 Methodology

4.2.1 Base model

Our starting architecture is based on the best fine-tuning model reported by Shin et al. [75], which is a BART-Large [40] seq2seq model with canonical-form targets and constrained decoding. Since this architecture uses canonical-form targets, both the inputs and the outputs are English sentences. As an example from the Pizza dataset, an input utterance like *could i have a medium pie along with ham sausage but please avoid bacon* is mapped to the output target *i want one medium pizza with ham and sausage and no bacon*. Canonical forms are defined by the domain developers and are designed so that they can be easily parsed using simple rules to obtain a conventional target LF.

Since the target canonical forms are user-defined and generated by a fixed grammar from which the ultimate meaning representations can be recovered, we can constrain the decoding in the seq2seq model to only produce valid sequences adhering to that grammar. We do this by defining a `validNextTokens` function that takes the tokens generated so far as input and returns the valid set of next tokens. During beam search, we adjust the logits to filter out invalid tokens.

4.2.2 Joint Training

We now describe our novel JT technique. While the base architecture was shown to perform well with dataset sizes of around 200, we observed that there is a lot of

room for improvement when the number of annotated examples falls further (to 48, 32, and 16 examples). Our key idea was to introduce auxiliary tasks constructed from easily-obtainable unsupervised data, and jointly train the model on these tasks, in addition to the semantic parsing task with a very small number of labeled examples. While labeling utterances is expensive, one can assume access to a larger set of unlabeled utterances. This assumption holds especially true for commercial voice assistants, which can record de-identified live traffic from participating customers. But even when bootstrapping new semantic parsers in a cold-start scenario, it is much easier to come up with utterances that need to be supported than it is to annotate these utterances. We can also generate a lot of target parse trees or canonical forms automatically, by sampling and generating from the target grammar. For example, we can generate sample pizza orders and create corresponding canonical forms. Given such data, we construct two tasks, Mask Prediction and Denoising, to augment the regular task.

4.2.2.1 Mask Prediction

Our first auxiliary task is focused on improving the encoder. We would like the encoder to see and learn to encode real source utterance sequences. To accomplish this, we use the unlabeled user utterances to construct an infilling-style mask prediction task. We mask spans of tokens in the same style as the BART pretraining objective. As an example where we mask a span containing roughly 25% of the tokens, the source is *i'll go for five pizzas along with MASK but avoid sausage* and the target is *i'll go for five pizzas along with mushrooms and onions but avoid sausage*. This task can be viewed as a form of domain adaptation, where the BART pretraining is continued on domain-specific data. It hence acts as a potential regularizer that stabilizes training with a small downstream task dataset. However, as we will show

later, integrating it with the regular task via JT is more effective than first adapting and then only fine-tuning on the labeled data.

4.2.2.2 Denoising

Our second auxiliary task is focused on improving the decoder. For this, we use the synthesized target canonical forms. These target canonical forms can be synthesized easily by randomly sampling from the target grammar. With the pizza dataset for example, this just corresponds to randomly creating various pizza orders and constructing their canonical forms. With a dataset like Overnight, it corresponds to generating random database queries from the query grammar.

Once we have a large set of random targets, we create a noisy version to use as the source sequences for a denoising task. We only add noise to the non-content tokens, i.e., tokens that do not interfere with entity names or intents. We do this to ensure that the model does not hallucinate. The choice of canonical forms which contain natural language instead of parse trees is also important here, as it allows us to easily add such noise. The noise itself consists of a set of manipulations on tokens. We randomly choose from the five following operations to apply to tokens with a certain probability:

- Delete: Delete a token.
- Replace: Replace a token with a token randomly sampled from the vocabulary.
- Swap: Swap two consecutive tokens.
- Insert: Insert a randomly sampled token.
- Duplicate: Duplicate a token.

An example of a noisy source sequence is *dishes want pizza one notified banana peppers uty pickles*, where the target is *i want one pizza with banana peppers and*

pickles. Once we construct the mask prediction and denoising datasets, we combine them with the labeled semantic parsing examples. We then shuffle the entire dataset and train the BART seq2seq model. Note that we do not introduce any weights or custom loss functions. We simply use the original sequence prediction loss to train on the new augmented dataset, as shown in Figure 4.1. We also do not explicitly differentiate between different task examples. The model learns to do mask prediction when it sees a *MASK* token. If not, it tries to generate a canonical form target sequence. We further ensure this is the case during inference by using constrained decoding.

4.2.3 Self-Training and Paraphrasing

To further improve upon JT, we introduce two enhancements: self-training and paraphrase augmentation. While both have been previously explored in isolation, we show that they work better in tandem with JT.

Self-training is a popular semi-supervised learning technique that has been explored across a wide range of applications to improve models with limited annotated data [48, 50]. The key idea is to first build a model with the existing labeled data and then use it to annotate an unlabeled dataset in order to obtain noisy annotations (silver labels). The model is then retrained with the combination of the original golden plus the silver data. This approach typically works well in low-resource scenarios for classification-style tasks or tasks with limited annotation diversity. It also requires a reasonable initial checkpoint to obtain the silver annotations.

We use our joint trained model as the initial checkpoint to label data. We make the self-training approach more effective for our generation-style task with some important design choices. The constrained decoding improves the label quality of the silver annotations and injects additional knowledge to retrain the model. We also add the mask prediction and denoising datasets to better retrain the model. Note that

we do not perform any confidence-based filtering or re-ranking on the silver labels since partially correct data might still help the decoder and confidence scores aren't reliable [12], especially with constrained decoding. We simply obtain predictions for all unlabeled data and use them to retrain the model, making this a straightforward enhancement.

A significant improvement comes from the data diversity introduced by self-training. By labeling unannotated utterances and augmenting the training dataset, the retrained model sees a larger variety of utterances. To further increase this variety, we propose paraphrase augmentation.

Paraphrasing is an effective way to obtain similar sentences with different surface forms. Since most neural paraphrasing models are noisy, especially when applied to out-of-domain data, we cannot assume that the semantics of the paraphrases are still captured by the original golden annotations. Instead, we rely on the self-training approach and use the JT model to label the newly generated paraphrases. We found that the diversity from the silver labeled utterances from these techniques is useful upto a certain size, at which point the label noise overpowers the diversity gains.

For our experiments, we built a paraphrasing model by training a BART-Large model on 5m examples from the ParaNMT dataset [86] for two epochs. As an example, *how many all season fouls did kobe bryant have as an la laker* is paraphrased as *how many fouls did kobe bryant have as a lakers player*. These are not exact paraphrases but still serve as new utterances for self-training.

4.2.4 Bringing it all together

To summarize, we start with a BART-Large seq2seq model. We convert the target LFs into canonical natural-language forms and implement constrained decoding to ensure that the generated tokens represent valid canonical forms. This is our base architecture. We train this base model using JT with mask infilling and denoising

as additional auxiliary tasks, along with semantic parsing using the limited labeled data. This produces our first model.

We use that model to label any available unannotated utterances. We also train a paraphrasing model and use it to paraphrase all the utterances, and we label the paraphrases with the same model. We then augment the JT data with these newly labeled examples and retrain the model. We get two more models in this step, one that uses the paraphrased data and one that does not.

4.3 Evaluation

4.3.1 Datasets

We evaluate our techniques on two datasets: Pizza[1] and Overnight [85]. We use three low-resource settings: 16, 32, and 48 labeled examples. These are randomly sampled from the full original datasets.

Pizza is a recently introduced dataset consisting of English utterances that represent orders of pizzas and drinks. The target parse is a LF that specifies the various components of the relevant pizza and drink orders. An example from this dataset was given in Section 4.2.1. We defined a canonicalization scheme for pizza and drink orders via a rule-based parser that can go from the canonical form to the LF and conversely.

The original Pizza dataset contains a synthetic training set, and real dev and test sets. For our experiments, we use the dev set to choose example for low-resource training. For the denoising task, we randomly sample 10k target parses from the original synthetic training set and construct their canonical forms to simulate random pizza orders.

[1]https://github.com/amazon-research/pizza-semantic-parsing-dataset

Overnight is a popular semantic parsing dataset that consists of 13,682 examples across eight domains. The task is to convert natural language utterances to database queries, which are then executed to obtain the results for the user utterances. We have access to the utterance, canonical form and the corresponding database query for all examples. An example from the basketball domain is the utterance *which team did kobe bryant play on in 2004*, whose canonical target is *team of player kobe bryant whose season is 2004*.

To generate queries for the denoising task, we use the SEMPRE toolkit [2], upon which the Overnight dataset was built, to generate sample queries for each domain from its canonical grammar, consisting of around 100 general and 20-30 per-domain rules.

For both datasets, for paraphrase augmentation, we generate four paraphrases for each utterance in the training set. We use the BART-Large model trained on ParaNMT data and take the top four sequences from beam search decoding at inference. For constrained decoding, we follow the approach of Shin et al. [75] and construct a large trie that contains all the canonical form sequences, and use it to look up valid next tokens given a prefix.

4.3.2 Baseline Models

We compare our models to the best fine-tuned model from Shin et al. [75], a BART-Large seq2seq model with canonical-form targets and constrained decoding, which we use as our base architecture. This model is trained on the same data as our JT models in the low-resource settings. We also compare to a fully trained version of this model, trained on all available training data.

4.3.3 Metrics

We report the recommended variants of exact match (EM) accuracy for both Pizza and Overnight datasets.

Unordered Exact Match Accuracy: For Pizza, we report unordered EM accuracy. This accounts for parses which have identical semantics but vary in their linearized representations due to differences in sibling order.

Denotation Accuracy: For Overnight, we report denotation accuracy. We execute the golden and predicted queries on the database and check for an exact match on the results. This accounts for any surface-level differences in the database queries that disappear upon actual execution.

4.3.4 Model Details

We use BART-Large as our base architecture. It contains 12 transformer encoder and decoder layers, 16 attention heads, and embeddings of size 1024 ($\sim$ 458 million parameters).

We train all our models with sequence cross entropy loss using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e-9$ and the Noam LR scheduler [81] with 500 warmup steps and a learning rate scale factor of 0.15. JT models are trained for 10 epochs, while base models are trained for 100 to 1000 epochs on the low-resource data. We fix the batch size to 512 tokens for all models. We use dropout of 0.1 and freeze the encoder token and position embeddings during training. During inference, we use beam search decoding with beam size 4. We did not perform any explicit hyperparameter tuning. Additional details, including the pizza canonicalization scheme, are provided in Appendix B. Data files can be found on our project page[2].

4.3.5 Results

Table 4.1 shows the performance of our models and baselines on Pizza. A fully trained BART model with canonical-form targets and constrained decoding (trained on the full dataset of 348 examples) achieves an unordered EM accuracy of 87.25%.

[2]https://github.com/amazon-research/resource-constrained-naturalized-semantic-parsing

	Unordered EM Accuracy		
	n=16	n=32	n=48
Baselines			
BART Canonical	16.95	53.35	58.36
Our models			
JT	42.23	**64.70**	70.30
JT + Self Training	**49.89**	63.23	72.07
JT + Self Training + Paraphrasing	48.19	64.55	**73.10**
Reference			
Full BART Canonical	87.25 (n = 348)		

Table 4.1. Results on **Pizza**. Our models consistently outperform baselines across all data sizes and bridge the gap to a fully trained model. The improvement is especially stark with 16 examples.

That is the SOTA result on this dataset. However, when the training data is reduced to 16 examples, the score drops to 16.95%. We see similar significant drops with 32 and 48 examples, with scores of 53.35% and 58.36% respectively. JT gives a huge boost to all three settings. With 16 examples, the accuracy jumps to 42.23%, with 32 examples to 64.70%, and with 48 examples to 70.30%. Self-training and paraphrase augmentation provide further boosts in the 16 and 48 example settings. Overall, we see that our best scores greatly improve the performance of the base architecture. This effect is most apparent in the 16 example setting, where we obtain an almost 3× improvement.

We see similar result trends with the Overnight dataset. Table 4.2 shows the denotation accuracies of all models across all eight domains. Our JT models attain a significant improvement over the baselines and bridge the gap towards fully trained models across all domains. The trend is especially noticeable in the calendar and recipes domains in the 16 example setting, where the denotation accuracies jump from 23.21% and 28.70% to 56.55% and 53.70% for our best models, respectively, a 2× boost on average.

	Basketball			Calendar			Publications			Restaurants		
	n=16	n=32	n=48	n=16	n=32	n=48	n=16	n=32	n=48	n=16	n=32	n=48
Baselines												
BART Canonical	41.43	58.57	68.29	23.21	35.71	63.10	29.81	42.86	55.90	20.18	48.19	55.12
Our models												
JT	48.85	63.17	68.29	51.19	**58.33**	62.50	52.80	49.69	59.63	58.43	64.46	66.57
JT + ST	56.52	**66.75**	**71.61**	**56.55**	57.74	66.67	**60.87**	**57.76**	62.11	**65.96**	**71.99**	65.36
JT + ST + Paraphrasing	**58.06**	66.24	70.84	55.95	55.95	**67.26**	59.63	55.90	**62.73**	65.66	69.58	**69.28**
Reference												
Full BART Canonical	89.51 (n = 1561)			85.12 (n = 669)			84.72 (n = 864)			82.18 (n = 3535)		
	Blocks			**Housing**			**Recipes**			**Social**		
	n=16	n=32	n=48	n=16	n=32	n=48	n=16	n=32	n=48	n=16	n=32	n=48
Baselines												
BART Canonical	27.07	21.05	30.33	15.87	43.92	37.57	28.70	38.43	46.76	28.73	40.50	45.02
Our models												
JT	36.34	36.84	48.37	45.50	58.73	60.32	48.61	**60.19**	63.89	24.21	39.48	53.17
JT + ST	38.85	38.60	**51.38**	**52.91**	64.02	**61.38**	52.78	58.80	65.74	**36.09**	**48.30**	57.01
JT + ST + Paraphrasing	**39.35**	**39.60**	49.87	52.38	**64.55**	**61.38**	**53.70**	**60.19**	**66.20**	31.45	45.25	**58.14**
Reference												
Full BART Canonical	69.67 (n = 1596)			80.42 (n = 752)			83.85 (n = 640)			88.25 (n = 1325)		

Table 4.2. Results on **Overnight** [85]. We report the denotation accuracy here. We again see that our models consistently outperform the baselines across all data sizes and domains, sometimes by up to 40% in the 16-example setting.

We wanted to analyze how far these improvements hold, so we repeated the experiment with 200 examples on the Overnight dataset. This also allows us to make a more direct comparison with some prior works that reported results for this setting. Table 4.3 presents these results. We see that our proposed techniques improve the denotation accuracies across all the domains over our baseline BART model by roughly 5 absolute points on average. Overall, they even slightly outperform the much larger and access-restricted GPT-3 model reported by Shin et al. [75].

4.3.6 Analysis

We analyzed some of our design decisions with experiments on the Pizza dataset. A detailed analysis can be found in the Appendix B. We summarize some of our findings here.

	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social
Baselines								
BART Canonical (our version)	84.7	55.4	77.4	68.3	73.3	75.5	75.3	69.7
BART Canonical [75]	86.4	55.4	78.0	67.2	75.8	80.1	80.1	66.6
GPT-3 [75]	85.9	**63.4**	79.2	**74.1**	77.6	79.2	**84.0**	68.7
Cao et al. [5]	77.2	42.9	61.3	55.0	69.6	67.1	63.9	56.6
Our models								
JT	84.9	62.7	81.0	**74.1**	78.3	79.6	79.8	69.0
JT + ST	**87.7**	62.4	82.1	**74.1**	79.5	**81.9**	80.7	69.2
JT + ST + Paraphrasing	86.7	**63.4**	**83.3**	73.5	**80.1**	78.7	81.0	**69.9**

Table 4.3. Results on **Overnight** [85] with 200 training examples. We outperform or match prior models on most domains, even outperforming prompting on the massive GPT-3 model which has almost 400x parameters.

Two-stage Finetuning: Our JT approach, while being simpler, does at-least as well or better than a two-stage finetuning process, where the auxiliary tasks are first used to pretrain the model and then the annotated data is used to finetune it. We see a noticeable drop in accuracy with the extra fine-tuning step for 32 (65% $\rightarrow$ 59%) and 48 (70% $\rightarrow$ 67%) examples, and no significant boost for 16 examples.

Importance of the canonical form: For our JT technique, the canonical form provides us with an easy way to add meaningful noise without modifying the content tokens for the denoising auxiliary task. We can simply perform token level operations without worrying about the target structure. If the targets are parse trees, adding noise is trickier, since most of the tokens in the parse represent content and meaningful operations need to be performed at the tree level. Further, the target sequences for the mask prediction task are in natural language and are better aligned with the canonical form targets than the parse trees. This potentially allows for better knowledge transfer during joint training.

We performed a JT experiment with a model that predicts tree LFs instead of canonical forms. We created the source sequences for the denoising auxiliary task using tree-level noise operations such as switching entities, dropping brackets, and inserting random tokens. We found that the resulting models achieved significantly

lower scores than the models that use canonical targets. For the 48 example case, the LF model achieves 59% accuracy compared to our JT model's 70%.

Synthetic data auxiliary task: The goal of our auxiliary tasks was to provide the model with a challenging objective. To train the decoder, we use the synthetically generated target sequences so that the decoder can train on, and learn to generate, a variety of valid canonical forms. To create a challenge for the decoder, we noise the targets to obtain corrupted source sequences and create a denoising task.

However, there are other possible tasks. One could create rules to generate synthetic utterances given the target parses. This synthetic data could then be used to train the decoder. This approach, however, requires manual effort and depends on the quality and diversity of the synthetic data. For Pizza, we already have access to synthetic data, since the entire training set is synthetic. Assuming we have access to a system that can generate such synthetic utterances given randomly generated target parses, we could replace our denoising task with the synthetic examples. We perform this experiment to compare these two auxiliary tasks. The synthetic model achieves 82% accuracy compared our denoising model's 70% in the 48 example case. The synthetic parsing auxiliary task performs better than denoising but requires lots of manual effort to create a synthetic utterance grammar. Our JT approach is directly applicable to both tasks.

4.4 Related Work

Naturalized semantic parsing can be traced back to work by Berant and Liang [3], who introduced the idea of canonical natural-language formulations of utterances. Our base architecture is based on work by Shin et al. [75]. There have been other approaches that explored low-resource semantic parsing in the past, which used concepts from meta-learning, self-training, and synthetic data generation [20, 90, 48].

Our model, however, is designed to be applicable to extremely small data sizes without requiring any external manual effort.

Wu et al. [87] have explored solving unsupervised semantic parsing as paraphrasing by decoding fixed paraphrases using a synchronous grammar. They hence require a carefully crafted synchronous grammar, unlike our method which relies on readily available data for auxiliary tasks.

Recently, there has also been an upward trend towards in-context learning or "prompting" approaches in low-resource settings [4, 75]. In these approaches, massive LMs are directly used to solve tasks without any training by framing the task as a prompt in the style of the pretraining objective, with a few task demonstrations selected from the handful of annotated examples. However, only GPT-3 has been shown to work well with a generation-style parsing task; smaller architectures, such as GPT-2, could not replicate the performance [75]. GPT-3 is a 175-billion parameter model that is currently not accessible by the entire research community.

Our JT technique can be seen as a mixture of domain adaptation of the pretrained LM and a regularizer. GRAPPA [93] is a recent effort that improves table semantic parsing using a separate pretraining phase, where the model is trained on synthetic parsing data and table-related utterances for domain adaptation before fine-tuning on a small annotated dataset of around 10k examples. Our work is similar to GRAPPA but focuses on much smaller training datasets, which requires us to train our model jointly with auxiliary tasks to make it more robust. We also show that denoising canonical forms is a reasonable auxiliary task.

At a high level, our approach also has some similarities to the work of Schick and Schutze [73], who also aim to show that smaller—and greener—LMs can be effective few-shot learners. They also utilize unlabeled data and a form of self-learning to augment a small amount of golden annotations. However, they focus on classification rather than generation tasks (reducing classification tasks to MLM).

Constraining the decoder of a neural semantic parser so that beam search only considers paths that adhere to various syntactic or semantic constraints has been widely explored over the last few years [37, 91]. Xiao et al. [89] show that constrained decoding can result in significant latency improvements.

4.5 Contributions

Our key idea is the application of joint training to train models with auxiliary tasks constructed from easily available unlabeled data in addition to the semantic parsing task with the limited annotated dataset. We also introduce a self-training step and a paraphrase augmentation step to augment the joint training data and further improve model performance.

We start with a strong baseline architecture that uses a BART-Large model, canonical-form targets, and constrained decoding, and show that our techniques provide massive improvements, in the order of 2–3× on EM scores. We evaluate our models on two datasets, Pizza and Overnight (the latter containing eight separate domains), and on three data sizes: 16, 32, and 48 examples. Models trained with our techniques consistently show improvements over baseline architectures across all datasets and size settings. The improvements are especially notable in the scarcest setting with 16 annotated examples. We analyze our model design and results in another series of experiments and show the effectiveness of our approach in constructing a robust, well-performing model.

4.6 Collaboration Statement

This chapter is adapted from Rongali et al. [67]. The work was done in collaboration with Konstantine Arkoudas, Melanie Rubino, and Wael Hamza from Amazon Alexa AI.

CHAPTER 5

ZERO-SHOT DOMAIN ADAPTATION AND UNIVERSAL SEMANTIC PARSING

5.1 Introduction

In this chapter, I describe proposed upcoming work. I plan to explore approaches for low-resource domain adaptation for task-oriented semantic parsing, specifically in the zero-shot setting.

Current semantic parsing models are trained on large amounts of annotated data from the predetermined set of domains. To increase the capabilities of the voice assistant to handle a new domain, we need to collect lots of annotated data for the new domain and re-train its semantic parsing model. This process is expensive and time-consuming and needs to be performed for each set of new domains. To combat this problem, researchers have proposed semantic parsing models that can be efficiently trained with fewer examples (few-shot) [76, 47, 19, 75, 9]. While this is a great first step, these models don't eliminate the problem entirely. They are typically evaluated on utterances from the new domain by first pretraining on data from existing domains and then finetuning on a few examples from the new domain. The performance of these models on the original domains drops in this scenario and this could be detrimental in production for a voice assistant. Furthermore, these models still require some examples from the new domain to perform finetuning. They cannot perform universal semantic parsing, where the voice assistant is given a large intent-slot schema covering many different aspects and requested to parse an utterance into that schema.

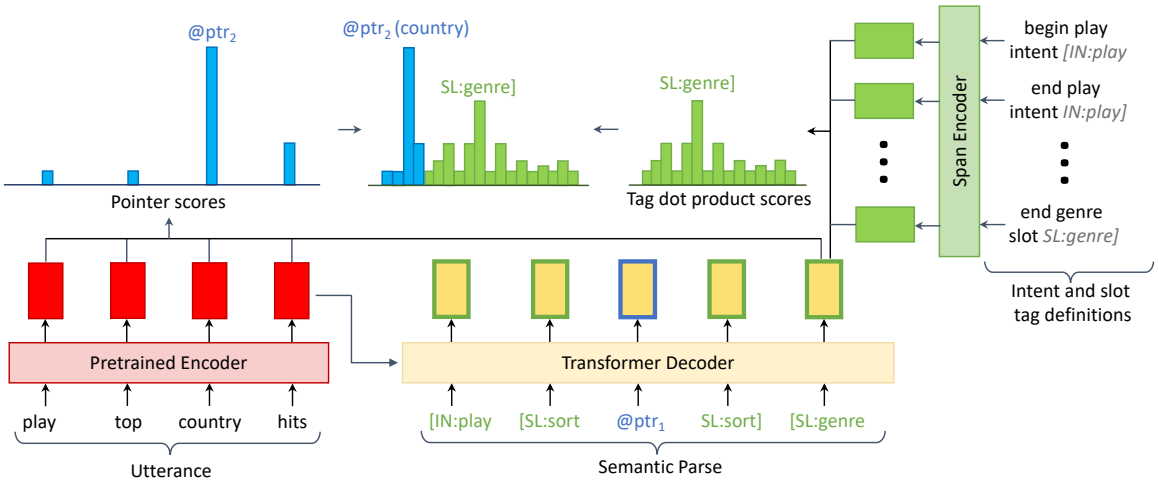

Figure 5.1. Proposed seq2seq model for Zeroshot Semantic Parsing for new domains. The target embedding and output layers are tied and replaced by embeddings from a span encoder that encodes the intent and slot tags from the new domain.

To achieve universal semantic parsing while maintaining high performance on domains with annotated data, we require models that can perform completely unsupervised (zero-shot) domain adaptation. In this scenario, the goal is to build a parsing model from just the annotated data from some domains that when given access to documentary information from a new domain, can effectively parse utterances in the new domain without any new annotated examples. To this end, I propose a model called Concept-Seq2Seq. Concept-Seq2Seq is based on our state-of-the-art semantic parsing model from Chapter 2, Seq2Seq-Ptr [69] that uses sequence to sequence models and a pointer generator network to decode the target semantic parse. We augment Seq2Seq-Ptr with a concept encoder that encodes intents and slots from the schema and uses those encodings to conditionally decode the semantic parse. Figure 5.2 shows the architecture of our proposed model.

Concept-Seq2Seq can be trained on annotated data from the given domains and during inference, all intents and slots from the schema, including new, unseen ones, can simply be encoded into the learned concept space, to decode the target parse. This model has the same time complexity as the original Seq2Seq-Ptr model but

comes with the added benefit of now being able to effectively parse utterances from unseen domains without any re-training. I also propose a pre-training scheme for Concept-Seq2Seq using an entity-centric processed version of Wikidata [83] called WikiWiki [41] to help it better encode unseen concepts.

There have been a few zero-shot semantic parsing approaches proposed in the past but they either only covered simple slot-filling style utterances [1, 39] or compositional utterances that also came with carefully crafted intermediate representations and context-free grammars [26, 88]. Our proposed model can perform zero-shot domain adaptation for compositional semantic parsing for parses with nested intents and slots but also doesn't require any grammars, whose construction effort, one could argue, is possibly more than the effort required to annotate a few examples.

We plan to evaluate Concept-Seq2Seq on two datasets: TOP v2 [6] and SNIPS [7]. Our goal is to report the first zero-shot performance number on the TOP dataset, which consists of complex utterances, and also show that our model achieves comparable zero-shot performance to other state-of-the-art models on the SNIPS dataset. We also plan to evaluate our model on in low-resource settings and compare to previous best-performing models in literature

5.2 Related Work

Zeroshot domain adaptation for task-oriented semantic parsing has been previously explored for simple flat queries with single intents and disjoint, non-overlapping slots. Bapna et al. [1] and Lee and Jha [39] encode the lexical tag features and create a token-tagging schema to create the final semantic parses. Yu et al. [92] solve the task using a retrieve-and-fill mechanism. We plan to use a baseline model for simple queries (SNIPS dataset) based on these approaches.

For complex utterances with nested structures, zeroshot semantic parsing has been explored using intermediate, concept-agnostic logical forms [26, 11, 66] or natural

language canonical forms [88]. These approaches apply to semantic parsing datasets which have context free grammars and specified rules, such as database or knowledge graph queries. The effort to craft these grammars for task-oriented semantic parsing in a voice assistant setting could quite possibly be greater than annotating utterances.

A more relevant class of approaches for our proposed work are ones that solve task-oriented semantic parsing for complex utterances in a few-shot setting using lexical tag features. Shrivastava et al. [76] and Mansimov and Zhang [47] modify the seq2seq architecture from Rongali et al. [69] to perform non-autoregressive style decoding and show that their models perform better in a few-shot setting. Ghoshal et al. [19] use adaptive label smoothing, a model-agnostic technique. Shin et al. [75] proposed a prompting-style approach where custom instructional prompts filled with handful of annotated examples and an unsolved utterance are fed as input to GPT-3 to directly produce a semantic parse. Their approach is extremely slow and cannot be easily adapted into a zeroshot framework. Shrivastava et al. [77] explore a retrieve-and-fill style approach where they retrieve the best *scenario*, an intermediate logical form consisting of the semantic frame and abstracted out tags, from a scenario bank of all supported semantic parses. Their approach is contingent on the availability of this scenario bank which could possibly entail more effort than annotating utterances. Mueller et al. [53] and Desai et al. [9] use lexical features from intent and slot names to create an *inventory* and use it as input to train semantic parsers for new domains. Mueller et al. [53] also pretrain their model to improve generalizability but only evaluate it on an intent classification task. Desai et al. [9] evaluate their model for full sequences and our model is similar to theirs. However, we plan to use our *inventory* to create custom decoder embeddings in a seq2seq model, which removes any input size issues that their model will encounter with large inventories. We also plan to pre-train our model with Wikidata and evaluate it in a completely zero-shot setting, in addition to few-shot.

5.3 Proposed Methodology

In this section, I describe our proposed model, Concept-Seq2Seq in detail. Like Seq2Seq-Ptr, it consists of a sequence-to-sequence encoder-decoder component, augmented with a pointer generator network to constrain the target decoding vocabulary. Since our task at hand is to perform zero-shot semantic parsing with just documentary information about the new domain, we modify the architecture of Seq2Seq-Ptr to incorporate information about new intents and slots from new domains by adding a concept encoder. Section 5.3.2 describes this proposed architecture in detail. To help our model learn to parse utterances from unseen domains better, we also propose a pre-training scheme to incorporate general concept parsing knowledge into it. Section 5.3.3 describes this concept pre-training scheme. Before we get to these sections, we first recall the source and target sequence formulation for the semantic parsing task from Chapter 2 below.

5.3.1 Task Formulation

Our proposed model solves semantic parsing as a sequence-to-sequence task, where the source sequence is the utterances and the target sequence is a linearized representation of the semantic parse. We modify the target sequence to only contain intent/slot tags or pointers to utterances tokens. An example source and target sequence from the TOP dataset are given below.

Source: *How far is the coffee shop*
Target: [*IN*:*GET_DISTANCE* @ptr_0 @ptr_1 @ptr_2 [*SL*:*DESTINATION*
[*IN*:*GET_RESTAURANT_LOCATION* @ptr_3 [*SL*:*TYPE_FOOD* @ptr_4
SL:*TYPE_FOOD*] @ptr_5 *IN*:*GET_RESTAURANT_LOCATION*]
SL:*DESTINATION*] *IN*:*GET_DISTANCE*]

Each @ptr_i token here points to the i^{th} token in the source sequence. So @ptr_4 corresponds to the word *coffee*.

5.3.2 Model Architecture

CONCEPT-SEQ2SEQ consists of three main components: an encoder, a decoder, and a concept encoder. Just like in traditional sequence-to-sequence models, the encoder encodes the source sequence, and the decoder autoregressively decodes the target sequence. However, since the target sequence in zero-shot parsing can contain intent and slot tags that the model hasn't seen during training, our models needs to be able to incorporate new intents and slots, or concepts, and decode the target sequence accordingly. The concept encoder helps us do this by encoding documentary information about new concepts and creating vector representations that we can use while decoding the target sequence.

Specifically, for an input sequence $[x_1 \dots x_n]$, we first encode it using the encoder into a sequence of hidden states $e_1 \dots e_n$. Then, having generated the first $t-1$ tokens, the decoder generates the token at step t as follows. It first produces the decoder hidden state at time t, d_t by building a multi-layer, multi-head self-attention on the encoder hidden states and the decoder states so far. This step is based on the transformer decoder from [81]. In a traditional sequence-to-sequence generation task, d_t is then fed into a dense layer to produce scores over the target vocabulary.

Our target vocabulary consists of pointer tokens and the concept tags. Since we do not have access to all concept tags at the time of training, we train our model to incorporate documentary information about concepts instead of using a fixed-size dense layer. To do this, we encode intent and slot concepts using a **concept encoder**. We first create a description for each token using the documentary information we have at hand. We include multiple features such as, is it a begin or end tag, is it an intent or a slot token, any textual description, and if available, a few examples. The textual description could be as simple as just making the tags more *natural* by removing special characters. For example, the description for the intent tag token `[IN:GET_DISTANCE` looks something like "*begin get distance intent*". Similarly, for

`SL:DESTINATION]`, it looks something like "*end travel destination slot*". We describe all the concept descriptions we use for different experimental settings in the evaluation section and include them our repository.

Given m concept tokens and their descriptions, the concept encoder encodes each of them to produce concept vector representations $[c_1 \dots c_m]$. We then use the computed decoder hidden state at t, d_t, as the query and compute unnormalized attention scores $[s_1 \dots s_m]$ with $[c_1 \dots c_m]$, and $[a_1 \dots a_n]$ with $[e_1 \dots e_n]$. Concatenating all these scores, we obtain an unnormalized distribution over $m+n$ tokens, the first m of which are the intent and slot tagging tokens from the concepts, and the last n of which are the $@ptr_i (0 < i < n)$ tokens pointing to the source sequence. We feed this through as softmax layer to obtain the final probability distribution. This probability distribution is used in the loss function during training and to choose the next token to generate during inference. For the target token embeddings in the decoder, we use a set of special embeddings to represent the $@ptr_i$ tokens and $[c_1 \dots c_m]$ to represent the concept embedding.

Figure 5.1 shows this process in action on a toy example. The model is decoding the next token after `SL:genre` at step 5. To do this, the model computes the pointer attention scores $[a_1 \dots a_n]$ (blue, left) and the concept token attention scores $[s_1 \dots s_n]$ (green, right). The highest overall score is for the token $@ptr_2$, corresponding to the word *country* in the source sequence, so the next predicted token is *country*.

5.3.3 Concept Pre-training

Concept-Seq2Seq has the ability to incorporate new, unseen concepts while parsing using the concept encoder. However, if the unseen concepts are semantically very different from ones seen during training, the model might struggle with overfitting. To allow the model to generalize better, we propose a pre-training phase, where the we train our model to encode a multitude of concepts and use them to parse

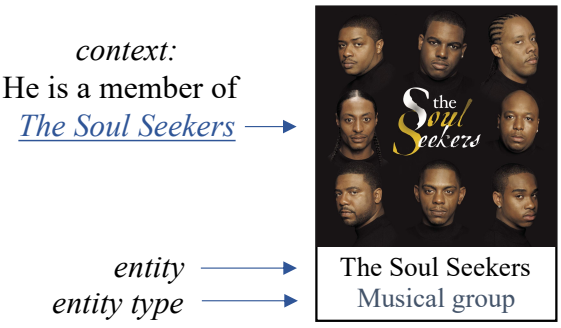

Figure 5.2. An example sentence from the Wikiwiki dataset with the associated mention, entity, and type fields. The full hyperlinked sub-span is extracted as the mention and the entity and type are extracted from the target page.

sentences. We use an entity-centric dataset created from Wikidata called Wikiwiki [41] for this purpose.

The Wikiwiki dataset curates mentions, entities, and entity types from 10M Wikipedia documents using hyperlink information linking sub-spans of text in sentences to other Wikipedia pages. The hyperlink is considered the mention, and the entity and the type information are extracted from the new page. For further details on this processing, please refer to Li et al. [41]. This dataset contains around 2m entities and 40K entity types.

Each example in the Wikiwiki dataset consists of a *context*, which is a paragraph from a wiki page, *mentions*, which are sub-spans of text that link to another page, *entities*, which correspond to each mention, and *entity types*, which describe the type of the entity. We extract individual sentences from this dataset and use them to train Concept-Seq2Seq to learn to encode a wide variety of concepts using the entity type fields as descriptions and tag the relevant mentions in the sentence. Figure 5.2 shows an example sentence from this dataset and the different fields. The source and target sequences for pre-training, and the descriptions of the concept tags for this example are given below.

Source: *He is a member of The Soul Seekers*
Target: @ptr_0 @ptr_1 @ptr_2 @ptr_3 @ptr_4 *[Q215380* @ptr_5 @ptr_6 @ptr_7

```
        Q215380]
Concept descriptions:
        [Q215380: begin musical group
        Q215380]: end musical group
```

During training, we collect all the concept tokens within a training batch and use them to create in-batch negatives for decoding task. We do this since it is extremely inefficient to encode all 40K $\times$ 2 concept token descriptions (begin and end) from Wikiwiki in every step.

5.4 Initial Results

We ran some initial experiments on the TOP v2 dataset. For these experiments, we only performed the training step on a fixed set of domains. We didn't perform the concept pre-training step. Also, we simply used tag names from the dataset as descriptions. We plan to repeat the experiments by augmenting the names with manual descriptions (we will create them) and example spans in the near future.

TOP has eight domains. We used a leave-one-out approach where given n domains, we train models on annotated data from $n - 1$ of them and evaluate on utterances from the left out domain for zero-shot domain adaptation. We report exact match (EM) accuracy and F1.

We use a transformer encoder, initialized from a *roberta-base* checkpoint, for Concept-Seq2Seq. The decoder is a transformer decoder initialized from scratch and it contains 6 layers, 8 heads, and a hidden state size of 768. The concept encoder is also a transformer encoder and it is initialized from a *bert-base-uncased* checkpoint. We choose a BERT-based model here since it is pre-trained to compute a vector for the whole sequence using the CLS token, which is what we need for encoding a concept consisting of a multi-word description.

	Alarm	Timer	Music
Zero-shot			
F1 score	71.04	55.65	14.34
EM Accuracy	53.43	0.51	0.12
Fully-trained			
EM Accuracy	-	86.03	80.89

Table 5.1. Initial results on TOP v2. Vanilla CONCEPT-SEQ2SEQ does well on the alarm domain, but currently fails on timer and music domains.

We train our models using sequence cross entropy loss and an Adam optimizer with learning rate $2e^{-5}$ and $\epsilon = 1e^{-8}$, warm-up proportion 0.1, weight decay 0.01. The number of epochs is set to 100 but we use early stopping with a patience of 5.

Table 5.1 contains some initial results of CONCEPT-SEQ2SEQ on three domains of TOP v2. For the fully-trained results, we use the checkpoint trained without alarm utterances, meaning it has been trained on timer and music domains and we see exact match accuracy scores comparable to current fully-trained state-of-the-art semantic parsing models. Table 5.2 contains some generated parses by CONCEPT-SEQ2SEQ.

In the zero-shot setting, we see that CONCEPT-SEQ2SEQ does well on the alarm domain. It achieves an exact match accuracy of 53.43 in the vanilla setting without pre-training or manual descriptions. This is pretty close to the score of 62.13 EM, reported in a few-shot setting in Desai et al. [9], where the model is additionally fine-tuned on around fifty examples (one sample per intent/slot tag).

In the timer domain, the model achieves a decent F1 score but very low EM accuracy. Upon manual examination of the generated parses, we found that in almost all utterances in this domain, there exists a span describing the type of timer such as *stopwatch* or *counter* and it needs to be tagged as `SL:METHOD_TIMER`. CONCEPT-SEQ2SEQ doesn't tag any spans with the `SL:METHOD_TIMER` tag since the description *method timer* is too generic for it to learn the parsing rule. It however, does a

	Alarm
Utterance	cancel all alarms for thursday
Gold	[IN:DELETE_ALARM cancel [SL:AMOUNT all SL:AMOUNT] alarms [SL:ALARM_NAME [IN:GET_TIME [SL:DATE_TIME for thursday SL:DATE_TIME] IN:GET_TIME] SL:ALARM_NAME] IN:DELETE_ALARM]
Predicted	[IN:DELETE_ALARM cancel [SL:AMOUNT all SL:AMOUNT] alarms [SL:DATE_TIME for thursday SL:DATE_TIME] IN:DELETE_ALARM]
Utterance	Set an alarm for 2 hours
Gold	[IN:CREATE_ALARM Set an alarm [SL:DATE_TIME for 2 hours SL:DATE_TIME] IN:CREATE_ALARM]
Predicted	[IN:CREATE_ALARM Set an alarm [SL:DATE_TIME for 2 hours SL:DATE_TIME] IN:CREATE_ALARM]
	Timer
Utterance	cancel all timers
Gold	[IN:DELETE_TIMER cancel [SL:AMOUNT all SL:AMOUNT] [SL:METHOD_TIMER timers SL:METHOD_TIMER] IN:DELETE_TIMER]
Predicted	[IN:DELETE_TIMER cancel [SL:AMOUNT all SL:AMOUNT] timers IN:DELETE_TIMER]
Utterance	Set timer for 30 minutes and 20 seconds
Gold	[IN:CREATE_TIMER Set [SL:METHOD_TIMER timer SL:METHOD_TIMER] [SL:DATE_TIME for 30 minutes and 20 seconds SL:DATE_TIME] IN:CREATE_TIMER]
Predicted	[IN:CREATE_TIMER Set timer [SL:DATE_TIME for 30 minutes and 20 seconds SL:DATE_TIME] IN:CREATE_TIMER]
	Music
Utterance	This song is the pits
Gold	[IN:DISLIKE_MUSIC This [SL:MUSIC_TYPE song SL:MUSIC_TYPE] is the pits IN:DISLIKE_MUSIC]
Predicted	[IN:CREATE_PLAYLIST_MUSIC [SL:MUSIC_GENRE This song is the pits SL:MUSIC_PROVIDER_NAME] IN:PLAY_MUSIC]
Utterance	Turn on music and play jazz
Gold	[IN:PLAY_MUSIC Turn on [SL:MUSIC_TYPE music SL:MUSIC_TYPE] and play [SL:MUSIC_GENRE jazz SL:MUSIC_GENRE] IN:PLAY_MUSIC]
Predicted	[IN:CREATE_PLAYLIST_MUSIC Turn on [SL:MUSIC_GENRE music and play jazz SL:MUSIC_GENRE] IN:CREATE_PLAYLIST_MUSIC]

Table 5.2. Some parses generated by Concept-Seq2Seq.

good job figuring out the other slots and the intents such as IN:PAUSE_TIMER or IN:BEGIN_TIMER which are adequately described in just their names.

Our model fails on the music domain. It learns to open and close tags but it mostly generates gibberish tags. We believe this is due to the extreme semantic difference between a domain such as music and the trained domains such as weather, timer, alarm etc. Also, the tags names in music are sometimes too close to each other, for example IN:PLAY_MUSIC and IN:REPLAY_MUSIC.

5.5 Remaining Challenges

Based on our initial results, we have identified some challenges to be addressed going forward. We also briefly describe proposed solutions to address them.

- **Special parsing rules**: From the results on the timer domain in TOP, we see that the data sometimes contains special parsing rules that can not be learned from just the tag names alone. For example, there is no way for the model to link the slot SL:METHOD_TIMER to the spans *timer* or *stopwatch* with the description *method timer*. Perhaps a better description would have been something such as *the type of timer such as stopwatch or counter*. We believe adding such manual descriptions and span examples to just the tag names would help the model do better here. We plan to manually create these descriptions and extract some high frequency examples for each of the tags.

- **Semantic differences in new domains**: We believe the main reason that Concept-Seq2Seq currently doesn't work on the music domain is the semantic gap between music and the other domains using during training. As described, we believe the Wikiwiki pre-training scheme will address this issue by training on a lot of generic concepts.

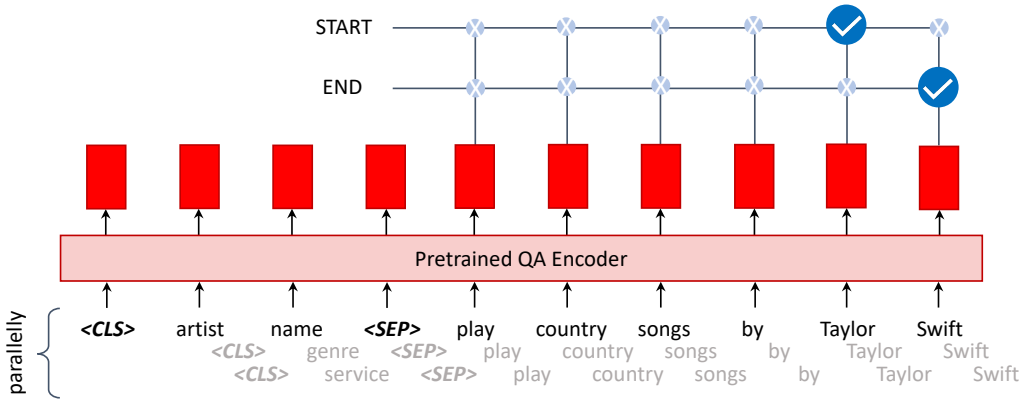

Figure 5.3. Proposed zero-shot extractive QA-style approach for domain adaptation. We plan to explore this approach if we don't achieve the expected performance from CONCEPT-SEQ2SEQ

- **Speed**: Right now, our decoder in CONCEPT-SEQ2SEQ is fully auto-regressive i.e. it generates tokens one by one. The inference speed is similar to SEQ2SEQ-PTR but for real-life applications in a voice assistant, we need to be even faster. To improve the speed of decoding, we plan to explore schemes such as insertion decoding [96], where tag tokens are inserted into their place, or parallel decoding [18] with a length module, where all the tokens are decoded at once, after a length module predicts the target length.

- **Alternate approach**: If the proposed approach doesn't achieve expected performance, we alternatively plan to explore this problem of zero-shot domain adaptation for semantic parsing using question-answering (QA) style models. A brief sketch of this approach is as follows: Train a extractive QA model with intent and slot descriptions posed as questions, the utterances as context, and the gold-span for the intent or slot tag as the answer. During inference with tags from a new domain, find the highest scoring spans for each new tag posed as a question and combine the results. Figure 5.3 shows this approach.

5.6 Research Plan

- CONCEPT-SEQ2SEQ design and evaluation

  Timeline: May 2022, In Progress

  Priority: High

- Wikiwiki data processing for pre-training

  Timeline: May 2022, In Progress

  Priority: High

- Manual schema descriptions, example creation

  Timeline: May 2022

  Priority: High

- Concept pre-training

  Timeline: May 2022

  Priority: High

- Non-autoregressive decoding for speed

  Timeline: June 2022

  Priority: Medium

- Alternate QA-style approach

  Timeline: June-July 2022

  Priority: Low

- Dissertation writing

  Timeline: July 2022

  Priority: High

- Dissertation Defense

  Timeline: August 2022

  Priority: High

CHAPTER 6

CONCLUSION

In this dissertation, I explore various approaches to improve low resource language understanding in voice assistants. We identify aspects of the current language understanding architecture and their data-related challenges, and propose novel architectures based on sequence-to-sequence models, large-scale pretraining, and transfer learning to solve those challenges.

In Chapter 2, we describe a state-of-the-art natural language understanding system, SEQ2SEQ-PTR, to solve compositional task-oriented semantic parsing, the bedrock intelligence system in voice assistants. Our approach uses a sequence-to-sequence model, initialized with a large pre-trained encoder and augmented with a pointer generator network, and has been shown to achieve state-of-the-art results on popular semantic parsing datasets such as TOP, ATIS, and SNIPS. This work laid the foundation for future low-resource work.

Chapter 3 describes an end-to-end spoken language understanding system that replaces the current two-stage pipelined ASR-NLU architecture. Our system, called AT-AT, directly takes speech as input and produces a semantic parse, without generating an intermediate text transcription. It uses sequence-to-sequence models, initialized with pre-trained checkpoints, and creates a shared audio-text representation space by training on multiple speech-to-text and text-to-text tasks. We show that AT-AT outperforms previous system, including pipelined ones, on spoken language understanding on datasets such as SNIPS and FluentSpeech. It is also shown to be capable of performing zeroshot end-to-end semantic parsing by only using a few text-

to-text annotated examples and no speech data by taking advantage of its shared audio-text hidden space.

In the final chapters, we cover approaches for few-shot and zero-shot semantic parsing. In Chapter 4, we describe approaches to perform few-shot semantic parsing by training large sequence-to-sequence semantic parsing models on very few annotated examples and carefully crafted auxiliary tasks from other easily available data. Our approach also relies on *naturalization* of the target semantic parse i.e. converting the target logical form into a controlled fragment of natural language. These approaches are hence best applied to semantic parsing datasets that have a well-describe grammar for the target logical forms such as database queries or food orders.

In Chapter 5, we explore few-shot and zero-shot approaches for domain adaptation in semantic parsing i.e. adapting the model to parse data from new domains with new logical form features. We propose a model called Concept-Seq2Seq, based on Seq2Seq-Ptr, and augmented with a concept encoder that encodes concepts from new domains. We also propose a pre-training scheme for our model to transfer concept knowledge from Wikipedia data. We show promising initial results and describe the remaining challenges, and next steps to evaluate and analyze our approach.

Together, our approaches lay the foundation for a comprehensive architecture and training scheme for low-resource language understanding. We show how to leverage the power or large sequence-to-sequence models and transfer learning to build robust and high-performing language understanding systems.

APPENDIX A

AT-AT DETAILS

A.1 Data Processing

This section describes the data processing that is done on different datasets to obtain target SLU sequences in the required format.

A.1.1 SNIPS Audio and Internal Datasets

These datasets contain simple utterances with a single intent and consecutive non-overlapping slots. The SLU hypothesis consists of a single intent and consecutive, non-overlapping words tagged as slots. While this hypothesis prediction is traditionally solved as a joint intent classification and slot-filling task, it can be solved as a sequence generation task. The hypothesis is converted into a target sequence with intents and slot tags enclosing the spoken English words. An example of this conversion is shown below.

```
Audio : Play songs by Iron Maiden
Intent: PlayMusic
Slots : Other Other Other ArtistName ArtistName
Target: PlayMusic( Play songs by ArtistName(
        Iron Maiden )ArtistName )PlayMusic
```

A.1.2 Facebook TOP

The TOP dataset consists of complex utterances with multiple nested intents and slots. The dataset already consists of the target hypothesis in a target sequence format. We simply add custom end brackets instead of one single closing bracket to end all slots. An example from the TOP dataset in the final format is given below.

```
Audio : How far is the coffee shop
Target: GetDistance( How far is Destination(
    GetRestaurantLocation( the TypeFood(
    coffee )TypeFood shop )GetRestaurantLocation
    )Destination )GetDistance
```

A.1.3 FluentSpeech

The FluentSpeech dataset consists of utterances where the SLU hypothesis is a 3-tuple containing an action, object, and a location. It was initially solved as a 3-way classification task. We use some fixed rules to convert this 3-tuple into a tagged sequence of the desired format as described with the previous datasets.

- The action tags are like intent tags for us. We wrap the whole transcript with action open and end tags.
- The location is usually found in the transcript. We tag the appropriate word with location open and end tags. The exception is when the location is washroom and the transcript word is bathroom. We manually account for these examples.
- For the object word, if it exists in the transcript, is is tagged with object open and end tags. If it doesn't, we wrap the transcript with open and end tags corresponding to the object token. Examples given below will make these rules more clear.

```
Audio : Turn on the kitchen lights
Tuple : activate, lights, kitchen
Target: Activate( Turn on the Location( kitchen
        )Location Object( lights )Object )Activate

Audio : Far too loud
Tuple : decrease, volume, none
Target: Decrease( Volume( Far too loud )Volume
        )Decrease

Audio : Turn up the bathroom temperature
Tuple : increase, heat, washroom
Target: Increase( Heat( turn up the Location(
```

```
        bathroom )Location temperature )Heat
        )Increase
```

A.2 Synthetic Audio Generation

For our zeroshot end-to-end experiments, we used models trained on synthetically generated audio data as one of our baselines. We generate synthetic audio for the internal books dataset and the TOP dataset. We used an external TTS system called Amazon Polly[1] for this. Polly is an AWS service that can be used to generate synthetic audio given a transcript, with options for different speakers, accents, and generation engines.

For our generation, we use a set of speakers that have US English accents and a neural engine. This set consists of the following voices: Joanna, Matthew, Salli, Justin, Kendra, Joey, Kimberly, Ivy, and Kevin. For each utterance in the dataset, we randomly choose a speaker from this set and synthesize speech from the transcript. Once the audio is generated, we process it in the same way as real audio, by extracting 80-dim log-filterbank features.

[1] https://aws.amazon.com/polly/

APPENDIX B

NATURALIZED SEMANTIC PARSING DETAILS

B.1 Dataset Details

B.1.1 Pizza

Pizza[1] is a recently introduced dataset consisting of English utterances that represent orders of pizzas and drinks. The target parse is a LF that specifies the various components of the relevant pizza and drink orders. Examples from this dataset can be seen in Table B.1. Since our system uses canonical forms as targets instead of LFs, we defined a canonicalization scheme for pizza and drink orders via a rule-based parser that can go from the canonical form to the LF and conversely (details in the next section).

The original Pizza dataset contains 2.5M synthetic training examples, 348 dev examples, and 1357 test examples. For our experiments, we ignore the synthetic training data and use the 348 dev examples as the training set to choose sets of 16, 32, and 48 examples for low-resource training.

To create the mask-infilling data, we include utterances from the unselected examples. For the denoising task, we randomly sample 10k target parses from the original synthetic training set of 2.5M examples and construct their canonical forms. This is akin to generating random pizza orders since that is how the synthetic dataset was created in the first place.

[1]https://github.com/amazon-research/pizza-semantic-parsing-dataset

Utterance	Canonical Form
get me three pepsis five medium diet sprites and a coke	i want one coke , five medium diet sprite , and three pepsi
i need a medium ham pizza with extra cheese and pesto	i want one medium pizza with extra cheese , ham , and pesto
can i have a large pizza along with onions tuna and add some thin crust	i want one large thin crust pizza with onions and tuna
good evening how are you do me a favor and get me a large pizza with ham and peppers i definitely do not want thin crust thanks	i want one large pizza with ham and peppers , not thin crust style
i wish to have one pie in large size along with olives and chicken but without ham	i want one large pizza with chicken and olives and no ham

Table B.1. Example utterances and canonical form for the Pizza dataset

B.1.2 Overnight

Overnight is a popular semantic parsing dataset that consists of 13,682 examples across eight domains. The task is to convert natural language utterances to database queries, which are then executed on a fixed database to obtain the results for the user utterances. This dataset was originally generated by first creating canonical utterances and their parses (database queries) and then paraphrasing the canonical utterances using crowd sourcing to obtain the natural utterance. As a result, we have access to the utterance, canonical form and the corresponding database query for all examples. An example from the basketball domain is the utterance *which team did kobe bryant play on in 2004*, whose canonical target is *team of player kobe bryant whose season is 2004*.

Just like with the Pizza dataset, we use the remaining utterances to create mask-infilling data. To generate queries for the denoising task, we use the SEMPRE toolkit, upon which the Overnight dataset was built, to generate sample queries for each domain from its canonical grammar, consisting of around 100 general and 20-30 per-domain rules. For paraphrase augmentation, for both datasets, we generate four

paraphrases for each utterance in the training set. We use the BART-Large model trained on ParaNMT data and take the top four sequences from beam search decoding at inference.

For constrained decoding, we construct a large trie that contains all the canonical form sequences, and use it to look up valid next tokens given a prefix.

B.2 Pizza Canonical Forms

The Pizza dataset consists of natural-language utterances representing pizza (and / or drink) orders, along with corresponding LFs. For our experiments, however, we needed natural-language sentences as the target outputs. Unlike the Overnight dataset, Pizza doesn't come with a canonical-form grammar. Accordingly, we created our own grammar and rules to convert LFs to and from canonical utterances. We describe these below.

Every target LF in the Pizza dataset consists of one or more pizza and/or drink orders. Each pizza order contains various attributes such as number, size, style, toppings, and so on. Some of these attributes, such as complex toppings (which contain a topping and a quantity qualifier, like *extra* cheese) are nested. Likewise, each drink order has attributes such as number, size or volume, container type (can or bottle), and so on. Given some LF tree t with orders $o_1, o_2 \ldots o_n$, we express the canonical form of t, $CF(t)$, in terms of the canonical forms of the individual components of t as follows:

$$CF(t) = \textbf{i want } CF(o_1), CF(o_2), \ldots \textbf{and } CF(o_n)$$

Each pizza/drink component order is further naturalized to create the canonical form sequence specified by the above expression. For a pizza order, this string captures the pizza attributes, while for a drink order it captures the drink attributes. The

following expressions roughly describe how these strings are laid out for a pizza order p and a drink order d in terms of their various attributes, represented in angle brackets $\langle\rangle$ (multi-valued attributes have a starred superscript).

$$
\begin{aligned}
CF(p) \;=\; & \langle number\rangle\langle size\rangle\langle style\rangle^{*}\textbf{pizza with} \\
& \langle topping\rangle^{*}, \textbf{and no } \langle topping\rangle^{*}, \\
& \textbf{not } \langle style\rangle^{*}\textbf{style} \\
CF(d) \;=\; & \langle number\rangle\langle size\rangle\langle volume\rangle \\
& \langle drink\ name\rangle\langle container\rangle
\end{aligned}
$$

We simply skip filling an attribute value if it doesn't exist for an order. For further nesting such as with complex toppings, the canonical string is a concatenation of the values of all its attributes. The constructed canonical forms for all examples in the train, dev, and test sets are available in the data zip file. Table B.1 provides a few sample canonical forms and their corresponding utterances for reference.

B.3 Further experimental details

We provide a few more details on our experimental settings here. Note that we didn't perform extensive hyper-parameter tuning. This section is simply to serve as a guideline for reproducing results reported in this paper.

B.3.1 Auxiliary Tasks

For the mask infilling task, we use all the available unannotated utterances to create source and target sequences. For our experiments, we upsample by $10\times$ and mask a random span of 25% tokens in each example. So for the pizza dataset for example, the size of this data in the $n = 16$ setting is $(348 - 16) \times 10 = 3320$. The

same holds for the Overnight dataset. The mask infilling source and target sequence files for both the Pizza and Overnight datasets and all the reported experimental settings are available in the data zip file.

For the denoising task, we sample 10k random LFs from the synthetic training data for the Pizza data and apply noise. For the Overnight dataset, we use SEMPRE to generate canonical forms and upsample them until they reach 10k and then apply noise. So the dataset size is always 10K in all our experiments. The source and target sequence files for the denoising task are available in the data zip file.

B.3.2 Computational Resources

We train our BART Paraphrasing model on a 8x32GB GPU. We use a large GPU here since the training dataset contains around 5 million examples. For all the semantic parsing models reported in the paper, both the baselines and our models, we use a single 16GB GPU.

B.4 Full Analysis

In this section we analyze our results in more detail and also explain various design choices and the empirical results that motivated them. For most of the analysis experiments, we use the Pizza dataset, since, as the Results Section shows, the results mostly generalize across both datasets. The Pizza dataset also has a larger test set than any of the Overnight domains, which allowed us to see performance differences better.

B.4.1 Joint training vs Two-stage Fine-Tuning

Our approach employs joint training, where we combine the auxiliary task data with the annotated data and jointly train our model. Our intuition here was that this technique would make the training more robust and act as a regularizer. One could instead use the auxiliary tasks to pretrain the model and then fine-tune it on

	Unordered EM Accuracy		
	n=16	n=32	n=48
Two-stage Fine-Tuning	**43.66**	59.47	67.87
Joint Training	42.23	**64.70**	**70.30**

Table B.2. Comparing joint training to two-stage fine-tuning.

just the annotated data. We found that this two-stage training does not improve the model. Table B.2 reports the results across the three data sizes on the pizza dataset for the JT and 2-stage fine-tuned models. We see a noticeable drop in performance with the extra fine-tuning step for 32 and 48 examples, and no significant boost for 16 examples.

B.4.2 Importance of the canonical form

We found canonical targets to work better in low-resource settings than LF targets, which stands to reason given that they are natural language sentences that can better leverage the pretraining of LMs. Moreover, for our JT technique, the canonical form provides us with an easy way to add meaningful noise without modifying the content tokens for the denoising auxiliary task. We can simply perform token level operations without worrying about the target structure. However, if the targets are parse trees, adding noise is trickier, since most of the tokens in the parse represent content and meaningful operations need to be performed at the tree level. Even more importantly, the target sequences for the mask prediction task are in natural language and are better aligned with the canonical form targets than the parse trees. This potentially allows for better knowledge transfer during joint training.

We performed a JT experiment with a model that predicts LFs instead of canonical forms for the Pizza dataset. We created the source sequences for the denoising auxiliary task using tree-level noise operations such as switching entities, dropping brackets, and inserting random tokens. We found that the resulting models achieved

	Unordered EM Accuracy		
	n=16	n=32	n=48
Logical Form	19.90	57.26	59.10
Canonical Form	**42.23**	**64.70**	**70.30**

Table B.3. Comparing canonical form targets to parse trees for the denoising task.

	Unordered EM Accuracy		
	n=16	n=32	n=48
Denoising	42.23	64.70	70.30
Synthetic Utterances	72.37	78.11	82.31

Table B.4. Comparing denoising to synthetically generated semantic parsing as the auxiliary task.

significantly lower scores than the models that use canonical targets. Table B.3 reports these numbers. The numbers for LF targets are actually very close to those of their base architectures (for LF-based BART). So, as hypothesized, directly using LF trees as targets is not conducive to our joint training approach. We require canonical-form targets in the base architecture for JT to work effectively.

B.4.3 Other auxiliary tasks apart from denoising

The goal of our auxiliary tasks was to provide the model with a challenging objective. Using unlabeled utterances, we create the mask infilling task in the style of the BART pretraining objective. This is a standard domain adaptation technique that is well explored in the literature.

To train the decoder, we use the synthetically generated target sequences so that the decoder can train on, and learn to generate, a variety of valid canonical forms. To create a challenge for the decoder, we noise the targets to obtain corrupted source sequences and create a denoising task. This is a simple task that can be constructed to provide the decoder a challenge without requiring any external effort.

There are other possible tasks. One could create rules to generate synthetic utterances given the target parses. This synthetic data could then be used to train the decoder. That approach, however, requires manual effort and depends on the quality and diversity of the synthetic data. For Pizza, we already have access to synthetic data, since the entire training set is synthetic. Assuming we have access to a system that can generate such synthetic utterances given randomly generated target parses, we could replace our denoising task with the synthetic examples. We perform this experiment to compare these two auxiliary tasks. Table B.4 shows these results. The synthetic parsing auxiliary task performs better than denoising but, as mentioned earlier, it requires a lot of manual effort to create a synthetic utterance grammar. Our JT approach is directly applicable to both tasks.

BIBLIOGRAPHY

[1] Bapna, Ankur, Tur, Gokhan, Hakkani-Tur, Dilek, and Heck, Larry. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363* (2017).

[2] Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)* (2013).

[3] Berant, Jonathan, and Liang, Percy. Semantic parsing via paraphrasing. In *ACL (1)* (2014), ACL (Association for Computer Linguistics), pp. 1415–1425.

[4] Brown, Tom B, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[5] Cao, Ruisheng, Zhu, Su, Liu, Chen, Li, Jieyu, and Yu, Kai. Semantic parsing with dual learning. *arXiv preprint arXiv:1907.05343* (2019).

[6] Chen, Xilun, Ghoshal, Asish, Mehdad, Yashar, Zettlemoyer, Luke, and Gupta, Sonal. Low-resource domain adaptation for compositional task-oriented semantic parsing. *arXiv preprint arXiv:2010.03546* (2020).

[7] Coucke, Alice, Saade, Alaa, Ball, Adrien, Bluche, Théodore, Caulier, Alexandre, Leroy, David, Doumouro, Clément, Gisselbrecht, Thibault, Caltagirone, Francesco, Lavril, Thibaut, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190* (2018).

[8] Denisov, Pavel, and Vu, Ngoc Thang. Pretrained semantic speech embeddings for end-to-end spoken language understanding via cross-modal teacher-student learning. *arXiv preprint arXiv:2007.01836* (2020).

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

[10] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Dong, Li, and Lapata, Mirella. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793* (2018).

[12] Dong, Li, Quirk, Chris, and Lapata, Mirella. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 743–753.

[13] Dyer, Chris, Kuncoro, Adhiguna, Ballesteros, Miguel, and Smith, Noah A. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776* (2016).

[14] Einolghozati, Arash, Pasupat, Panupong, Gupta, Sonal, Shah, Rushin, Mohit, Mrinal, Lewis, Mike, and Zettlemoyer, Luke. Improving semantic parsing for task oriented dialog. *arXiv preprint arXiv:1902.06000* (2019).

[15] Fayek, Haytham M. Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between, 2016.

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

[17] Furui, Sadaoki. Digital speech processing: Synthesis, and recognition, second edition,, 2000.

[18] Ghazvininejad, Marjan, Levy, Omer, Liu, Yinhan, and Zettlemoyer, Luke. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324* (2019).

[19] Ghoshal, Asish, Chen, Xilun, Gupta, Sonal, Zettlemoyer, Luke, and Mehdad, Yashar. Learning better structured representations using low-rank adaptive label smoothing. In *International Conference on Learning Representations* (2020).

[20] Goldwasser, Dan, Reichart, Roi, Clarke, James, and Roth, Dan. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (2011), pp. 1486–1495.

[21] Goo, Chih-Wen, Gao, Guang, Hsu, Yun-Kai, Huo, Chih-Li, Chen, Tsung-Chieh, Hsu, Keng-Wei, and Chen, Yun-Nung. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 753–757.

[22] Graves, Alex, Mohamed, Abdelrahman, and Hinton, Geoffrey E. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), 6645–6649.

[23] Gupta, Sonal, Shah, Rushin, Mohit, Mrinal, Kumar, Anuj, and Lewis, Mike. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 2787–2792.

[24] Haghani, Parisa, Narayanan, Arun, Bacchiani, Michiel, Chuang, Galen, Gaur, Neeraj, Moreno, Pedro J., Prabhavalkar, Rohit, Qu, Zhongdi, and Waters, Austin. From audio to semantics: Approaches to end-to-end spoken language understanding. *2018 IEEE Spoken Language Technology Workshop (SLT)* (2018), 720–726.

[25] Hakkani-Tür, Dilek, Celikyilmaz, Asli, Chen, Yun-Nung, Gao, Jianfeng, Deng, Li, and Wang, Ye-Yi. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech* (2016).

[26] Herzig, Jonathan, and Berant, Jonathan. Decoupling structure and lexicon for zero-shot semantic parsing. *arXiv preprint arXiv:1804.07918* (2018).

[27] Hinton, Geoffrey E., Deng, Li, Yu, Dong, Dahl, George E., Mohamed, Abdelrahman, Jaitly, Navdeep, Senior, Andrew W., Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara, and Kingsbury, Brian. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine 29* (2012), 82.

[28] Huang, Zhiheng, Xu, Wei, and Yu, Kai. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[29] Jia, Robin, and Liang, Percy. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622* (2016).

[30] Jia, Xueli, Wang, Jianzong, Zhang, Zhiyong, Cheng, Ning, and Xiao, Jing. Large-scale transfer learning for low-resource spoken language understanding. *arXiv preprint arXiv:2008.05671* (2020).

[31] Jiao, Feng, Wang, Shaojun, Lee, Chi-Hoon, Greiner, Russell, and Schuurmans, Dale. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (2006), Association for Computational Linguistics, pp. 209–216.

[32] Josh Wardini. Voice Search Statistics: Smart Speakers, Voice Assistants, and Users in 2022, Jan 2022.

[33] Karita, Shigeki, Chen, Nanxin, Hayashi, Tomoki, Hori, Takaaki, Inaguma, Hirofumi, Jiang, Ziyan, Someki, Masao, Soplin, Nelson Enrique Yalta, Yamamoto, Ryuichi, Wang, Xiaofei, Watanabe, Shinji, Yoshimura, Takenori, and Zhang, Wangyou. A comparative study on transformer vs rnn in speech applications. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (2019), 449–456.

[34] Kim, Yoon. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[35] Kingma, Diederik P, and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[36] Klein, Guillaume, Kim, Yoon, Deng, Yuntian, Senellart, Jean, and Rush, Alexander M. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810* (2017).

[37] Krishnamurthy, Jayant, Dasigi, Pradeep, and Gardner, Matt. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Copenhagen, Denmark, Sept. 2017), Association for Computational Linguistics, pp. 1516–1526.

[38] Lafferty, John D, McCallum, Andrew, and Pereira, Fernando CN. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (2001), pp. 282–289.

[39] Lee, Sungjin, and Jha, Rahul. Zero-shot adaptive transfer for conversational language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 6642–6649.

[40] Lewis, Mike, Liu, Yinhan, Goyal, Naman, Ghazvininejad, Marjan, Mohamed, Abdelrahman, Levy, Omer, Stoyanov, Ves, and Zettlemoyer, Luke. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[41] Li, Shuyang, Sridhar, Mukund, Prakash, Chandana Satya, Cao, Jin, Hamza, Wael, and McAuley, Julian. Instilling type knowledge in language models via multi-task qa. In *Findings of the 2022 North American Chapter of the Association for Computational Linguistics (NAACL)* (2022).

[42] Liang, Percy. Learning executable semantic parsers for natural language understanding. *arXiv preprint arXiv:1603.06677* (2016).

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

[44] Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, and Stoyanov, Veselin. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[45] Lugosch, Loren, Meyer, Brett H., Nowrouzezahrai, Derek, and Ravanelli, Mirco. Using speech synthesis to train end-to-end spoken language understanding models. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), 8499–8503.

[46] Lugosch, Loren, Ravanelli, Mirco, Ignoto, Patrick, Tomar, Vikrant Singh, and Bengio, Yoshua. Speech model pre-training for end-to-end spoken language understanding. *ArXiv abs/1904.03670* (2019).

[47] Mansimov, Elman, and Zhang, Yi. Semantic parsing in task-oriented dialog with recursive insertion-based encoder. *arXiv preprint arXiv:2109.04500* (2021).

[48] McClosky, David, Charniak, Eugene, and Johnson, Mark. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference* (2006), pp. 152–159.

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

[50] Mihalcea, Rada. Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004* (2004), pp. 33–40.

[51] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[52] Mohamed, Abdelrahman, Okhonko, Dmytro, and Zettlemoyer, Luke. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660* (2019).

[53] Mueller, Aaron, Krone, Jason, Romeo, Salvatore, Mansour, Saab, Mansimov, Elman, Zhang, Yi, and Roth, Dan. Label semantic aware pre-training for few-shot text classification. *arXiv preprint arXiv:2204.07128* (2022).

[54] Nassif, Ali Bou, Shahin, Ismail, Attili, Imtinan B., Azzeh, Mohammad, and Shaalan, Khaled. Speech recognition using deep neural networks: A systematic review. *IEEE Access 7* (2019), 19143–19165.

[55] National Public Radio. Smart Speakers See 78% Increase YOY, Jan 2019.

[56] Panayotov, Vassil, Chen, Guoguo, Povey, Daniel, and Khudanpur, Sanjeev. Librispeech: An asr corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), 5206–5210.

[57] Paulus, Romain, Xiong, Caiming, and Socher, Richard. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* (2017).

[58] Peng, Fuchun, Feng, Fangfang, and McCallum, Andrew. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics* (2004), Association for Computational Linguistics, p. 562.

[59] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[60] Peters, Matthew E, Neumann, Mark, Iyyer, Mohit, Gardner, Matt, Clark, Christopher, Lee, Kenton, and Zettlemoyer, Luke. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[61] Povey, Daniel, Ghoshal, Arnab, Boulianne, Gilles, Burget, Lukás, Glembek, Ondrej, Goel, Nagendra Kumar, Hannemann, Mirko, Motlícek, Petr, Qian, Yanmin, Schwarz, Petr, Silovský, Jan, Stemmer, Georg, and Veselý, Karel. The kaldi speech recognition toolkit, 2011.

[62] Prabhumoye, Shrimai, Tsvetkov, Yulia, Salakhutdinov, Ruslan, and Black, Alan W. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000* (2018).

[63] Price, Patti J. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990* (1990).

[64] Radford, Alec, Wu, Jeffrey, Child, Rewon, Luan, David, Amodei, Dario, and Sutskever, Ilya. Language models are unsupervised multitask learners. *OpenAI Blog 1*, 8 (2019).

[65] Raffel, Colin, Shazeer, Noam, Roberts, Adam, Lee, Katherine, Narang, Sharan, Matena, Michael, Zhou, Yanqi, Li, Wei, and Liu, Peter J. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv abs/1910.10683* (2019).

[66] Reddy, Siva, Täckström, Oscar, Petrov, Slav, Steedman, Mark, and Lapata, Mirella. Universal semantic parsing. *arXiv preprint arXiv:1702.03196* (2017).

[67] Rongali, Subendhu, Arkoudas, Konstantine, Rubino, Melanie, and Hamza, Wael. Training naturalized semantic parsers with very little data. In *To appear in Proceedings of the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence* (2022).

[68] Rongali, Subendhu, Liu, Beiye, Cai, Liwei, Arkoudas, Konstantine, Su, Chengwei, and Hamza, Wael. Exploring transfer learning for end-to-end spoken language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence 35*, 15 (May 2021), 13754–13761.

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

[70] Saade, Alaa, Coucke, Alice, Caulier, Alexandre, Dureau, Joseph, Ball, Adrien, Bluche, Théodore, Leroy, David, Doumouro, Clément, Gisselbrecht, Thibault, Caltagirone, Francesco, Lavril, Thibaut, and Primet, Maël. Spoken language understanding on the edge. *ArXiv abs/1810.12735* (2018).

[71] Sabour, Sara, Frosst, Nicholas, and Hinton, Geoffrey E. Dynamic routing between capsules. In *Advances in neural information processing systems* (2017), pp. 3856–3866.

[72] Sarı, Leda, Thomas, Samuel, and Hasegawa-Johnson, Mark. Training spoken language understanding systems with non-parallel speech and text. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), IEEE, pp. 8109–8113.

[73] Schick, Timo, and Schütze, Hinrich. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (June 2021), Association for Computational Linguistics, pp. 2339–2352.

[74] See, Abigail, Liu, Peter J, and Manning, Christopher D. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* (2017).

[75] Shin, Richard, Lin, Christopher H, Thomson, Sam, Chen, Charles, Roy, Subhro, Platanios, Emmanouil Antonios, Pauls, Adam, Klein, Dan, Eisner, Jason, and Van Durme, Benjamin. Constrained language models yield few-shot semantic parsers. *arXiv preprint arXiv:2104.08768* (2021).

[76] Shrivastava, Akshat, Chuang, Pierce, Babu, Arun, Desai, Shrey, Arora, Abhinav, Zotov, Alexander, and Aly, Ahmed. Span pointer networks for non-autoregressive task-oriented semantic parsing. *arXiv preprint arXiv:2104.07275* (2021).

[77] Shrivastava, Akshat, Desai, Shrey, Gupta, Anchit, Elkahky, Ali, Livshits, Aleksandr, Zotov, Alexander, and Aly, Ahmed. Retrieve-and-fill for scenario-based task-oriented semantic parsing. *arXiv preprint arXiv:2202.00901* (2022).

[78] Thomson, B., Gasic, M., Henderson, M., Tsiakoulis, P., and Young, S. N-best error simulation for training spoken dialogue systems. In *2012 IEEE Spoken Language Technology Workshop (SLT)* (Dec 2012), pp. 37–42.

[79] Tur, Gokhan, Hakkani-Tür, Dilek, and Heck, Larry. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop* (2010), IEEE, pp. 19–24.

[80] van Noord, Rik, Abzianidze, Lasha, Toral, Antonio, and Bos, Johan. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics 6* (2018), 619–633.

[81] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[82] Vinyals, Oriol, Fortunato, Meire, and Jaitly, Navdeep. Pointer networks. In *Advances in Neural Information Processing Systems* (2015), pp. 2692–2700.

[83] Vrandečić, Denny, and Krötzsch, Markus. Wikidata: a free collaborative knowledgebase. *Communications of the ACM 57*, 10 (2014), 78–85.

[84] Wang, Pengwei, Wei, Liangchen, Cao, Yong, Xie, Jinghui, and Nie, Zaiqing. Large-scale unsupervised pre-training for end-to-end spoken language understanding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), IEEE, pp. 7999–8003.

[85] Wang, Yushi, Berant, Jonathan, and Liang, Percy. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2015), pp. 1332–1342.

[86] Wieting, John, and Gimpel, Kevin. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732* (2017).

[87] Wu, Shan, Chen, Bo, Xin, Chunlei, Han, Xianpei, Sun, Le, Zhang, Weipeng, Chen, Jiansong, Yang, Fan, and Cai, Xunliang. From paraphrasing to semantic parsing: unsupervised semantic parsing via synchronous semantic decoding. *arXiv preprint arXiv:2106.06228* (2021).

[88] Wu, Shan, Chen, Bo, Xin, Chunlei, Han, Xianpei, Sun, Le, Zhang, Weipeng, Chen, Jiansong, Yang, Fan, and Cai, Xunliang. From paraphrasing to semantic parsing: Unsupervised semantic parsing via synchronous semantic decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Online, Aug. 2021), Association for Computational Linguistics, pp. 5110–5121.

[89] Xiao, Chunyang, Teichmann, Christoph, and Arkoudas, Konstantine. Grammatical sequence prediction for real-time neural semantic parsing. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges* (Florence, Aug. 2019), ACL (Association for Computational Linguistics), pp. 14–23.

[90] Xu, Silei, Semnani, Sina J, Campagna, Giovanni, and Lam, Monica S. Autoqa: From databases to qa semantic parsers with only synthetic training data. *arXiv preprint arXiv:2010.04806* (2020).

[91] Yin, Pengcheng, and Neubig, Graham. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers* (2017), Regina Barzilay and Min-Yen Kan, Eds., Association for Computational Linguistics, pp. 440–450.

[92] Yu, Dian, He, Luheng, Zhang, Yuan, Du, Xinya, Pasupat, Panupong, and Li, Qi. Few-shot intent classification and slot filling with retrieved examples. *arXiv preprint arXiv:2104.05763* (2021).

[93] Yu, Tao, Wu, Chien-Sheng, Lin, Xi Victoria, Wang, Bailin, Tan, Yi Chern, Yang, Xinyi, Radev, Dragomir, Socher, Richard, and Xiong, Caiming. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845* (2020).

[94] Zettlemoyer, Luke S, and Collins, Michael. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420* (2012).

[95] Zhang, Chenwei, Li, Yaliang, Du, Nan, Fan, Wei, and Yu, Philip. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 5259–5267.

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