ConVEx: Data-Efficient and Few-Shot Slot Labeling

Matthew Henderson[1] **and Ivan Vulić**[1,2]
[1] PolyAI Ltd, London, UK
[2] Language Technology Lab, University of Cambridge, UK
matt@polyai.com

Abstract

We propose **ConVEx** (**Con**versational **V**alue **Ex**tractor), an efficient pretraining and fine-tuning neural approach for slot-labeling dialog tasks. Instead of relying on more general pretraining objectives from prior work (e.g., language modeling, response selection), ConVEx's pretraining objective, a novel *pairwise cloze* task using Reddit data, is well aligned with its intended usage on sequence labeling tasks. This enables learning domain-specific slot labelers by simply fine-tuning decoding layers of the pretrained general-purpose sequence labeling model, while the majority of the pretrained model's parameters are kept frozen. We report state-of-the-art performance of ConVEx across a range of diverse domains and data sets for dialog slot-labeling, with the largest gains in the most challenging, few-shot setups. We believe that ConVEx's reduced pretraining times (i.e., only 18 hours on 12 GPUs) and cost, along with its efficient fine-tuning and strong performance, promise wider portability and scalability for data-efficient sequence-labeling tasks in general.

1 Introduction

Slot labeling or slot filling is a critical natural language understanding (NLU) component of any task-oriented dialog system (Young, 2002, 2010; Tür and De Mori, 2011, *inter alia*). Its goal is to fill the correct *values* associated with predefined *slots*: e.g., a dialog system for restaurant bookings is expected to fill slots such as *date*, *time*, and *the number of guests* with the values extracted from a user utterance (e.g., *next Thursday*, *7pm*, *4 people*).

Setting up task-oriented dialog systems, as well as slot labeling methods in particular, to support new tasks and domains is highly challenging due to inherent scarcity of expensive expert-annotated data for a plethora of intended use scenarios (Williams, 2014; Henderson et al., 2014; Budzianowski et al., 2018; Zhao et al., 2019). One plausible and promising solution is the creation of *data-efficient* models that learn from only a handful annotated examples in *few-shot scenarios*. This approach has recently been shown promising for learning intent detectors (Casanueva et al., 2020; Krone et al., 2020; Bunk et al., 2020) as well as for slot-filling methods (Hou et al., 2020; Coope et al., 2020) with limited annotated data.

The dominant paradigm followed by the existing models of few-shot slot labeling is *transfer learning* (Ruder et al., 2019): **1)** they rely on representations from models *pretrained* on large data collections in a self-supervised manner on some general NLP tasks such as (masked) language modeling (Devlin et al., 2019; Conneau et al., 2020; Brown et al., 2020) or response selection (Henderson et al., 2019b, 2020; Cer et al., 2018); and then **2)** add additional task-specific layers for modeling the input sequences. However, we detect several gaps with the existing setup, and set to address them in this work. First, recent work in NLP has validated that a stronger alignment between a pretraining task and an end task can yield performance gains for tasks such as extractive question answering (Glass et al., 2020) and paraphrase and translation (Lewis et al., 2020). We ask whether it is possible to design a pretraining task which is more suitable for slot labeling in conversational applications. Second, is it possible to bypass learning sequence-level layers from scratch, and simply *fine-tune* them after pretraining instead? Third, is it possible to build a generally applicable model which fine-tunes pretrained "general" sequence-level layers instead of requiring specialized slot labeling algorithms from prior work (Krone et al., 2020; Hou et al., 2020)?

Inspired by these challenges, we propose **ConVEx** (**Con**versational **V**alue **Ex**tractor), a novel Transformer-based neural model which can be pretrained on large quantities of natural language data

(e.g., Reddit) and then directly fine-tuned to a variety of slot-labeling tasks. Similar to prior work (Rastogi et al., 2019; Coope et al., 2020), ConVEx casts slot labeling as a span-based extraction task. For ConVEx, we introduce a new pretraining objective, termed *pairwise cloze*. This objective aligns well with the target downstream task: slot labeling for dialog, and emulates slot labeling relying on unlabeled sentence pairs from natural language data which share a keyphrase (i.e., a "value" for a specific "slot"). Instead of learning them from scratch as in prior work (Coope et al., 2020), ConVEx's pretrained Conditional Random Fields (CRF) layers for sequence modeling are fine-tuned using a small number of labeled in-domain examples.

We evaluate ConVEx on a range of diverse dialog slot labeling data sets spanning different domains: DSTC8 data sets (Rastogi et al., 2019), RESTAURANTS-8K (Coope et al., 2020), and SNIPS (Coucke et al., 2018). ConVEx yields state-of-the-art performance across all evaluation data sets, but its true usefulness and robustness come to the fore in the few-shot scenarios. For instance, it increases average F_1 scores on RESTAURANTS-8K over the current state-of-the-art Span-ConveRT model (Coope et al., 2020) from 57.6 to 76.0 with only 128 labeled examples available; the gains are even more pronounced with 64 labeled examples, from 40.5 to 71.7. Similar findings are observed with DSTC8, and we also report state-of-the-art performance in the 5-shot slot labeling task on SNIPS.

In summary, our results validate the benefits of *task-aligned pretraining from raw natural language data*, with particular gains for data-efficient slot labeling given a limited number of annotated examples, which is a scenario typically met in production. They also clearly demonstrate that competitive performance can be achieved via quick fine-tuning, without heavily engineered specialized methods from prior work (Hou et al., 2020). Further, we validate that learning sequence-level layers from scratch is inferior to fine-tuning from pretrained layers. From a broader perspective, we hope that this research will inspire further work on task-aligned pretraining objectives for other NLP tasks beyond slot labeling. From a more focused perspective, we hope that it will guide new approaches to data-efficient slot labeling for dialog.

2 Methodology

Before we delve deeper into the description of the ConVEx model in §2.3, in §2.1 we first describe a novel sentence-pair value extraction pretraining task used by ConVEx, called *pairwise cloze*, and then in §2.2 a procedure that converts "raw" unlabeled natural language data into training examples for the ConVEx pairwise cloze task.

2.1 Pretraining Task: Pairwise Cloze

Why Pairwise Cloze? Top performing natural language understanding models typically make use of neural nets pretrained on large scale data sets with unsupervised objectives such as language modeling (Devlin et al., 2019; Liu et al., 2019) or response selection (Henderson et al., 2020; Humeau et al., 2020). For sequential tasks such as slot labeling, this involves adding new layers and training them from scratch, as the pretraining procedure does not involve any sequential decoding; therefore, current unsupervised pretraining objectives are suboptimal for sequence-labeling tasks. With ConVEx, we introduce a new pretraining task with the following properties: **1)** it is more closely related to the target slot-labeling task, and **2)** it facilitates training all the necessary layers for slot-labeling, so these can be fine-tuned rather than learned from scratch.

What is Pairwise Cloze? In a nutshell, given a pair of sentences that have a *keyphrase* in common, the task treats one sentence as a *template sentence* and the other as its corresponding *input sentence*. For the template sentence, the keyphrase is masked out and replaced with a special *BLANK* token. The model must then read the tokens of both sentences, and predict which tokens in the input sentence constitute the masked phrase. Some examples of such sentence pairs extracted from Reddit are provided in Table 1. The main idea behind this task is teaching the model an implicit space of *slots* and *values*[1] that can be fine-tuned later to fit domain-specific slot labeling data.

The pairwise cloze task for pretraining has been inspired by the recent span selection objective applied to extractive question answering by Glass et al. (2020): they create examples emulating extractive QA pairs with long passages and short question sentences. In contrast, our work seeks to emulate slot labeling in a dialog system by creating examples from short conversational utterances.

[1] During self-supervised pretraining, slots are represented as the contexts in which a value might occur.

Template Sentence	*Input* Sentence
I get frustrated everytime I browse /r/all. I stick to my *BLANK* most of the time.	/r/misleadingpuddles Saw it on the **frontpage**, plenty of content if you like the premise.
Why Puerto Rico? It's Memphis at Dallas, which is in Texas where *BLANK* hit	**Hurricane Harvey**. Just a weird coincidence.
It really sucks, as the V30 only has *BLANK* . Maybe the Oreo update will add this.	Thanks for the input, but **64GB** is plenty for me :)
I took *BLANK*, cut it to about 2 feet long and duct taped Vive controllers on each end. Works perfect	Yeah, I just duct taped mine to **a broom stick**. You can only play no arrows mode but it's really fun.
I had *BLANK* and won the last game and ended up with 23/20 and still didn't get it.	I know how you feel my friend and I got **19/20** on the tournament today

Table 1: Sample data from Reddit converted to sentence pairs for the ConVEx pretraining via the pairwise cloze task. Target spans in the input sentence are denoted with bold, and are *"BLANKed"* in the template sentence.

2.2 Pairwise Cloze Data Preparation

Input Data. We assume working with the English language throughout the paper. Reddit has been shown to provide natural conversational English data for learning semantic representations that work well in downstream tasks related to dialog and conversation (Al-Rfou et al., 2016; Cer et al., 2018; Henderson et al., 2019b,a, 2020; Casanueva et al., 2020; Coope et al., 2020). Therefore, following recent work, we start with the 3.7B comments in the large Reddit corpus from 2015-2018 (inclusive) (Henderson et al., 2019a), filtering it to comments between 9 and 127 characters in length. This yields a total of almost 2B filtered comments.

Keyphrase Identification. Sentence-pair value extraction examples are extracted from unlabeled text based on their shared keyphrases. Therefore, in the first step, we must identify plausible candidate keyphrases. To this end, the filtered Reddit sentences are tokenized with a simple word tokenizer, and word frequencies are counted. The score of a candidate keyphrase $(w_1, w_2, \ldots, w_n)$ is computed as a function of the individual word counts: $(\prod_{i=1}^{n} \text{count}(w_i))^{1/n^\alpha}$.

This simple scoring function selects phrases that have informative low-frequency words. The factor α controls the length of the identified keyphrases: e.g., setting it to $\alpha = 0.8$, which is default in our experiments later, encourages selecting longer phrases. Given a sentence, the keyphrases are selected as those unigrams, bigrams and trigrams whose score exceeds a predefined threshold.

The keyphrase identification procedure is run for all sentences from the filtered Reddit sentences. At most two keyphrases are extracted per sentence, and keyphrases spanning more than 50% of the sentence text are ignored. Keyphrases that occur more than once in the sentence are also ignored.

Sentence-Pair Data Extraction. In the next step, sentences from the same subreddit are paired by keyphrase to create paired data, 1.2 billion examples in total,[2] where one sentence acts as the input sentence and another as the template sentence (see Table 1 again). Table 2 summarizes statistics from the entire pretraining data preparation procedure.

Total comments	3,680,746,776
Comments filtered by length	1,993,294,538
Extracted keyphrases	3,296,519,827
Training set size	1,172,174,919
Test set size	61,696,649
Mean number of words per keyphrase	1.3

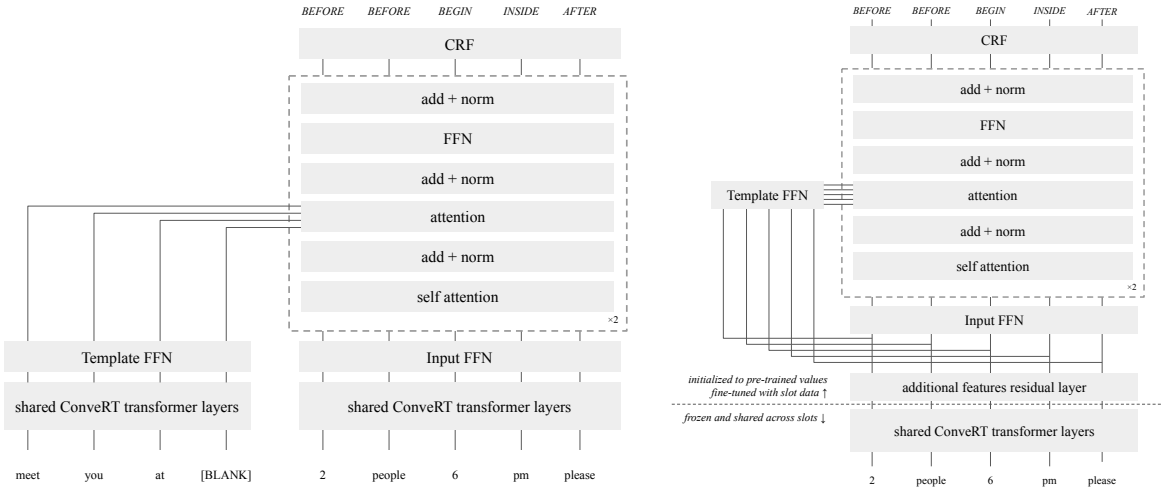
Table 2: Statistics of the pairwise cloze pretraining data from Reddit.

2.3 The ConVEx Framework

We now present *ConVEx*, a pretraining and fine-tuning framework that can be applied to a wide spectrum of slot-labeling tasks. ConVEx is pretrained on the pairwise cloze task (§2.1), relying on sentence-pair data extracted from Reddit (§2.2). Similar to prior work (Coope et al., 2020), we frame slot labeling as a span extraction task: spans are represented using a sequence of *tags*. These tags indicate which members of the sequence are in the span. We use the same tag representation as Coope et al. (2020), which is similar to the standard IOB format: the span is annotated with a sequence of *BEFORE*, *BEGIN*, *INSIDE* and *AFTER* tags. The ConVEx pretraining and fine-tuning architectures are illustrated in Figures 1a and 1b respectively, and we describe them in what follows.

ConVEx: Pretraining. The ConVEx model encodes the template and input sentences using exactly the same Transformer layer architecture

[2] We also expand keyphrases inside paired sentences if there is additional text on either side of the keyphrase that is the same in both sentences. For instance, the original keyphrase *"Star Wars"* will be expanded to the keyphrase *"Star Wars movie"* if the following sentences constitute a pair: *"I really enjoyed the latest Star Wars movie."* – *"We could not stand any Star Wars movie."*



(a) ConVEx: Pretraining

(b) ConVEx: Fine-tuning

Figure 1: An overview of the ConVEx model structure at: **(a)** pretraining, and **(b)** fine-tuning. The full description of each component of the model at both stages is provided in §2.3.

(Vaswani et al., 2017) as the lightweight and highly optimized ConveRT sentence encoder (Henderson et al., 2020): we refer the reader to the original work for all architectural and technical details. This model structure is very compact and resource-efficient (i.e., it is 59MB in size and can be trained in 18 hours on 12 GPUs) while achieving state-of-the-art performance on a range of conversational tasks (Casanueva et al., 2020; Coope et al., 2020; Bunk et al., 2020). The weights in the ConveRT Transformer layers are shared for both sentences. The input to the ConVEx pretraining also closely follows ConveRT's tokenization process: the final subword vocabulary contains 31,476 subword tokens plus 1,000 buckets reserved for out-of-vocabulary tokens. Input text is split into subwords following a simple left-to-right greedy prefix matching (Vaswani et al., 2018; Henderson et al., 2020), and we tokenize both input sentences and template sentences the same way.

The 512-dimensional output representations from the ConveRT layers are projected down to 128-dimensional representations using two separate feed-forward networks (FFNs), one for the template and one for the input sentence. The projected contextual subword representations of the input sentence are then enriched using two blocks of self attention, attention over the projected template sentence representations, and FFN layers. This provides features for every token in the input sentence that take into account the context of both the input sentence and the template sentence. A final linear layer computes Conditional Random Field (CRF) parameters for tagging the value span using the 4 *BEFORE*, *BEGIN*, *INSIDE*, and *AFTER* labels.

More formally, for each step t, corresponding to a subword token in the input sentence, the network outputs a 4×4 matrix of transition scores $\mathbf{W}_t$ and a 4-dimensional vector of unary potentials $\mathbf{u}_t$. Under the CRF model, the probability of a predicted tag sequence $\mathbf{y}$ is then computed as:

$$p(\mathbf{y}|\mathbf{v}) \propto \prod_{t=1}^{T-1} \exp\left(\mathbf{W}_t | y_{t+1}, y_t\right) \prod_{t=1}^{T} \exp\left(\mathbf{u}_t | y_t\right)$$

The loss is the negative log-likelihood, which is equal to the negative sum of the transition scores and unary potentials that correspond to the true tag labels, up to a normalization term. The top scoring tag sequences are computed efficiently using the Viterbi algorithm (Sutton and McCallum, 2012).

In addition to the CRF loss, an *auxiliary dot-product loss* can be added. This loss encourages the model to pair template sentences with the corresponding (semantically similar) input sentences. Let $\mathbf{f}_i^T$ be the d-dimensional encoding of the beginning-of-sentence (BOS) token for the i^{th} template sentence, and $\mathbf{f}_i^I$ be the encoding of the BOS token for the i^{th} (corrresponding) input sentence. As the encodings are contextual, the BOS representations can encapsulate the entire sequence. The auxiliary dot-product loss is then computed as:

$$-\sum_{i=1}^{N} C\left\langle \mathbf{f}_i^T,\ \mathbf{f}_i^I \right\rangle + \sum_{i=1}^{N} \log \sum_{j=1}^{N} e^{C\left\langle \mathbf{f}_i^T,\, \mathbf{f}_j^I \right\rangle} \quad (1)$$

where $\langle \cdot, \cdot \rangle$ is cosine similarity and C is an annealing factor that linearly increases from 0 to $\sqrt{d}$ over the first 10K training batches as in previous work (Henderson et al., 2020). The auxiliary loss is inspired by the dot-product loss typically used in retrieval tasks such as response selection (Henderson et al., 2017). Note that this loss does not necessitate any additional model parameters, and does not significantly increase the computational complexity of the pretraining procedure. Later in §4 we evaluate the efficacy of pretraining with and without the auxiliary loss.

ConVEx: Fine-tuning. In the ConVEx model, the majority of the computation and parameters are in the shared ConveRT Transformer encoder layers: they comprise 30M parameters, while the decoder layers comprise only 800K parameters. At fine-tuning, the shared ConveRT transformer layers of the pretrained ConVEx model are frozen: the expensive Transformer operations can be shared across slots, while the fine-tuned slot-specific models are small in memory and fast to run.

To apply the ConVEx model to slot-labeling for a specific slot, the user utterance is treated both as the input sentence and the template sentence (note that at fine-tuning and inference the user input does not contain any *BLANK* token) – see Figure 1b. This effectively makes the attention layers in the decoder act like additional self-attention layers. For some domains, additional context features such as the binary *is_requested* feature need to be incorporated (Coope et al., 2020): this is modeled through a residual layer that computes a term to add to the ConveRT output encoding, given the encoding itself and the additional features – see Figure 1b.

We note that, except for the residual layer, no new layers are added between pretraining and fine-tuning; this implies that the model bypasses learning from scratch any potential complicated dynamics related to the application task, and is directly applicable to various slot-labeling scenarios.

3 Experimental Setup

Pretraining: Technical Details. The parameters of the ConVEx model at pretraining are randomly initialized, including the ConveRT layer parameters, and the model is pretrained on the pairwise cloze Reddit data. Pretraining proceeds in batches of 256 examples, 64 of which are randomly paired sentences where no value should be extracted, and the remaining being paired examples from the training data. This teaches the model that sometimes no value should be predicted, a scenario frequently encountered with slot labeling. Table 3 provides a concise summary of these and other pretraining hyper-parameters.

Activation	Fast GELU approximation (Hendrycks and Gimpel, 2016)
Total batch size	256
Negatives per batch	64
Learning rate	0.3
Optimizer	Adadelta with $\rho = 0.9$ (Zeiler, 2012)
ConveRT layers	Same as in (Henderson et al., 2020)
Input / template FFN layer size	512 , 128
Decoder FFN size	256
Decoder attention projections	16

Table 3: ConVEx: Hyper-parameters at pretraining.

Computational Efficiency and Tractability. ConVEx is pretrained for 18 hours on 12 Tesla K80 GPUs; this is typically sufficient to reach convergence. The total pretraining cost is roughly $85 on Google Cloud Platform. This pretraining regime is orders of magnitude cheaper and more efficient than the prevalent pretrained NLP models such as BERT (Devlin et al., 2019), GPT models (Brown et al., 2020), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), etc. The reduced pretraining cost allows for wider experimentation, and aligns with recent ongoing initiatives on improving fairness and inclusion in NLP/ML research and practice (Strubell et al., 2019; Schwartz et al., 2019).

Fine-tuning: Technical Details. We use the same fine-tuning procedure for all fine-tuning experiments on all evaluation data sets. It proceeds for 4,000 steps of batches of size 64, stopping early if the loss drops below 0.001.[3] The ConveRT layers are frozen, while the other layers are initialized to their pretrained values and optimized with Adam (Kingma and Ba, 2015), with a learning rate of 0.001 that decays to 10^{-6} over the first 3,500 steps using cosine decay (Loshchilov and Hutter, 2017). Dropout is applied to the output of the ConveRT layers with a rate of 0.5: it decays to 0 over 4,000 steps also using cosine decay. The residual layer for additional features (e.g., *is_requested*, *token_is_numeric*, *token_is_word_boundary*) consists of a single hidden layer of dimensionality 1024. As we demonstrate later in §4, this procedure works well across a variety of data settings. The

[3] We train by enforcing that exactly 20% of examples in each batch contain a value, and 80% contain no value. Further, the batch size is reduced to smaller than 64 in few-shot scenarios if the training set is too small to meet this ratio without introducing duplicate examples.

early stopping and dropout are intended to prevent overfitting on very small data sets.

Fine-tuning and Evaluation: Data and Setup. We rely on several diverse slot-labeling data sets, used as established benchmarks in previous work. First, we evaluate on a recent data set from Coope et al. (2020): RESTAURANTS-8K, which comprises conversations from a commercial restaurant booking system. It covers 5 slots required for the booking task: *date*, *time*, *people*, *first name*, and *last name*. Second, we use the Schema-Guided Dialog Dataset (SGDD) (Rastogi et al., 2019), originally released for DSTC8, in the same way as prior work (Coope et al., 2020), extracting span annotated data sets from SGDD in four different domains. The particulars of the RESTAURANTS-8K and DSTC8 evaluation data are provided in Appendix A.

In order to investigate the model behavior in low-data regimes, for both data sets we simulate few-shot scenarios, and measure performance on smaller sets sampled from the full data. We (randomly) subsample the training sets of various sizes while maintaining the same test set.

Furthermore, we also evaluate ConVEx in the 5-shot evaluation task on the SNIPS data (Coucke et al., 2018), following the exact setup of Hou et al. (2020), which covers 7 diverse domains, ranging from *Weather* to *Creative Work* (see Table 6 later for the list of domains). The statistics of the SNIPS evaluation setting are also provided in Appendix A.

Since the SNIPS evaluation task slightly differs from RESTAURANTS-8K and DSTC8, we also provide additional details related to fine-tuning and evaluation procedure on SNIPS, replicating the setup of Hou et al. (2020). Each of the 7 domains in turn acts as a held-out test domain, and the other 6 can be used for training. From the held-out test domain, episodes are generated that contain around 5 examples, covering all the slots in the domain. For each domain, we first further pretrain the ConVEx decoder layers (the ones that get fine-tuned) on the other 6 domains: we append the slot name to the template sentence input, which allows training on all the slots. This gives a single updated fine-tuned ConVEx decoder model, trained on all slots of all other domains. For each evaluation episode, for each slot in the target domain we fine-tune 3 ConVEx decoders. The resulting predictions are *ensembled* by averaging probabilities to give final predictions. This helps reduce variability and improves prediction quality. We later investigate whether such model ensembling also helps in few-shot scenarios for RESTAURANTS-8K and DSTC8.

Baseline Models. For RESTAURANTS-8K and DSTC8, we compare ConVEx to the approaches from Coope et al. (2020) which displayed strongest performance on the two evaluation sets: Span-BERT and Span-ConveRT. Both models rely on the same CNN+CRF architecture[4] applied on top of the subword representations transferred from a pretrained BERT(-Base/Large) model (Devlin et al., 2019) (Span-BERT), or from a pretrained ConveRT model (Henderson et al., 2020).[5] For SNIPS, we compare ConVEx to a wide spectrum of different few-shot learning models proposed and compared by Hou et al. (2020).[6]

One crucial difference between our approach and the methods evaluated by Hou et al. (2020) is as follows: we treat each slot independently, using separate ConVEx decoders for each, while the their methods train a single CRF decoder that models all slots jointly. One model per slot is simpler, easier for practical use (e.g., it is possible to keep and manage data sets for each slot independently), and makes pretraining conceptually easier.[7]

Evaluation Measure. Following previous work (Coucke et al., 2018; Rastogi et al., 2019; Coope et al., 2020), we report the average F_1 scores for extracting the correct span per user utterance. If the models extract part of the span or a longer span, this is treated as an incorrect span prediction.

4 Results and Discussion

Intrinsic (Reddit) Evaluation. ConVEx reaches a precision of 84.8% and a recall of 85.3% on the held-out Reddit test set (see Table 2 again), using

[4] See the original work of Coope et al. (2020) for further technical details.

[5] Coope et al. (2020) also evaluated an approach based on the same CNN+CRF architecture as Span-{BERT, ConveRT} which does not rely on any pretrained sentence encoder, and learns task-specific subword representations from scratch. However, that approach is consistently outperformed by Span-ConveRT, and we therefore do not report it for brevity.

[6] A full description of each baseline model is beyond the scope of this work, and we refer to (Hou et al., 2020) for further details. For completeness, short summaries of each baseline model on SNIPS are provided in the Appendix B.

[7] Moreover, the methods of Hou et al. (2020) are arguably more computationally complex: at inference, their strongest models (i.e., TapNet and WPZ, see Appendix B, run BERT for every sentence in the fine-tuning set (TapNet), or run classification for every pair of test words and words from the fine-tuning set (WPZ). The computational complexity of the ConVEx approach does not scale with the fine-tuning set, only with the number of words in the query sequence.

Fraction	Span-ConveRT	Span-BERT	ConVEx -no-aux	ConVEx -full
1 (8198)	95.8	93.1	95.6	**96.0**
1/2 (4099)	**94.1**	91.4	**94.1**	**94.1**
1/4 (2049)	91.2	88.0	92.2	**92.6**
1/8 (1024)	88.5	85.3	90.0	**90.6**
1/16 (512)	81.1	76.6	86.2	**86.4**
1/32 (256)	63.8	53.6	78.4	**81.8**
1/64 (128)	57.6	42.2	73.4	**76.0**
1/128 (64)	40.5	30.6	70.9	**71.7**

Table 4: Average F_1 scores across all slots for the evaluation on RESTAURANTS-8K test data with varying training set fractions. Numbers in brackets denote the training set sizes. The peak scores in each training setup (i.e., per row) are in bold.

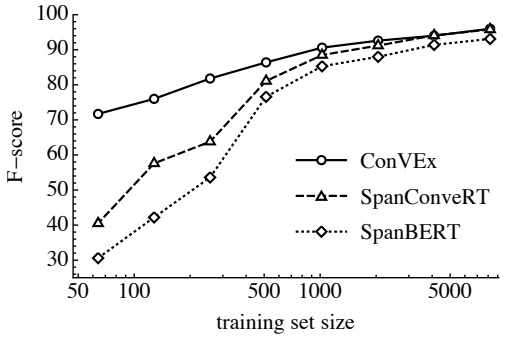


Figure 2: Plot of the average F_1 across all slots for RESTAURANTS-8K with varying training set sizes.

25% random negatives as during pretraining. The ConVEx variant without the auxiliary loss (termed *no-aux* henceforth) reaches a precision of 82.7% and a recall of 83.9%, already indicating the usefulness of the auxiliary loss.[8] These preliminary results serve mostly as a sanity check, suggesting the ability of ConVEx to generalize over unseen Reddit data, while we evaluate its downstream task efficacy in the subsequent experiments.

Evaluation on RESTAURANTS-8K and DSTC8. The main results are summarized in Table 4 and Table 5. In Figure 2 and Figure 3 we additionally plot the performance of ConVEx along with the baseline models in few-shot scenarios with varying numbers of examples. We extract several findings from the results. In full-data scenarios all models in our comparison, including the baselines from Coope et al. (2020), yield strong performance reaching $\geq 90\%$ or even $\geq 95\%$ average F_1 across the board.[9] However, it is encouraging that ConVEx is able to surpass the baseline models on average even in the full-data regimes.

Fraction	Span-ConveRT	Span-BERT	ConVEx -no-aux	ConVEx
Buses_1				
1 (1133)	93.5	93.3	95.1	**96.0**
1/2 (566)	88.9	85.3	90.4	**92.6**
1/4 (283)	84.0	77.8	**88.6**	86.7
1/8 (141)	69.1	69.6	83.6	**84.0**
1/16 (70)	58.3	44.4	74.5	**75.2**
1/32 (35)	32.7	25.5	**65.4**	59.2
Events_1				
1 (1498)	**92.7**	84.3	92.4	91.7
1/2 (749)	86.9	80.2	**88.4**	87.3
1/4 (374)	82.2	78.6	86.4	**87.2**
1/8 (187)	70.0	57.4	72.4	**82.2**
1/16 (93)	55.9	43.9	65.7	**66.6**
1/32 (47)	39.2	25.6	51.4	**54.0**
Homes_1				
1 (2064)	94.8	96.3	95.5	**98.3**
1/2 (1032)	**96.1**	**95.7**	95.6	95.6
1/4 (516)	**95.4**	**95.1**	93.0	94.5
1/8 (258)	93.4	89.5	92.2	**94.8**
1/16 (129)	86.3	76.4	**94.0**	92.3
1/32 (65)	77.1	61.2	89.4	**92.0**
RentalCars_1				
1 (874)	**94.0**	92.8	90.7	92.0
1/2 (437)	**93.1**	87.9	89.3	91.7
1/4 (218)	83.0	81.4	**87.9**	87.4
1/8 (109)	66.4	64.8	**78.8**	77.6
1/16 (54)	51.6	49.6	**62.3**	60.6
1/32 (27)	44.0	30.1	47.3	**50.3**

Table 5: F_1 scores on the DSTC8 single-domain data sets. Numbers in brackets denote the training set sizes. The peak scores in each training setup are in bold.

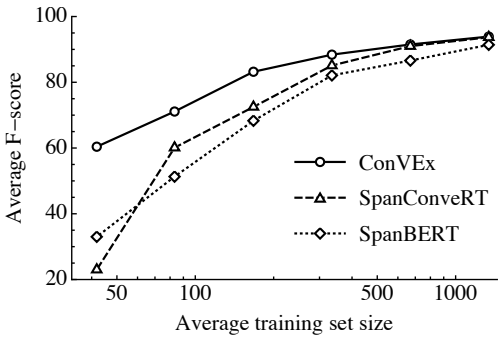


Figure 3: Plot of the average F_1 across all slots for DSTC8, against the average data set size over all slots.

True benefits of the proposed ConVEx approach, however, are revealed in Figure 2 and Figure 3: they indicate the ability of ConVEx to handle few-shot scenarios, where the gap between ConVEx and the baseline models becomes more and more pronounced as we continue to reduce the number of annotated examples for the labeling task. On RESTAURANTS-8K the gain is still small when dealing with 1,024 annotated examples (+2.1 F_1 points over the strongest baseline), but it increases to +18.4 F_1 points when 128 annotated examples are available, and further to +31.2 F_1 points when

[8] While we evaluate the two ConVEx variants also in the slot-labeling tasks later, unless noted otherwise, in all experiments we assume the use of the variant *with* the *aux* loss.

[9] The only exception is lower performance of Span-BERT on the DSTC8 *Homes_1* evaluation. In general, as shown previously by Coope et al. (2020) and revalidated in our experiments, conversational pretraining based on response selection (ConveRT) seems more useful for conversational applications that regular LM-based pretraining (BERT).

	Weather	Music	Playlist	Book	Search	Restaurant	Creative	*Average*
Hou et al. (2020)								
TransferBERT	59.4	42.0	46.1	20.7	28.2	67.8	58.6	46.1
SimBERT	53.5	54.1	42.8	75.5	57.1	55.3	32.4	52.9
WPZ+BERT	67.8	56.0	46.0	72.2	73.6	60.2	66.9	63.2
TapNet	53.0	49.8	54.9	83.4	63.1	59.8	67.0	61.6
TapNet+CDT	66.5	66.4	68.2	**85.8**	73.6	64.2	68.5	70.4
L-WPZ+CDT	**74.7**	56.7	52.2	78.8	80.6	69.6	67.5	68.6
L-TapNet+CDT	71.6	67.2	75.9	84.4	82.6	70.1	**73.4**	75.0
This work								
ConVEx (with aux)	71.5	**77.6**	**79.0**	84.5	**84.0**	**73.8**	67.4	**76.8**

Table 6: F_1 scores on SNIPS 5-shot evaluation, following the exact setup of Hou et al. (2020). For an overview of the baseline models from Hou et al. (2020), see the original work and short summaries available in Appendix B.

Fraction	ConVEx	ConVEx ensemble
1 (8198)	**96.0**	95.8
1/2 (4099)	94.1	**94.2**
1/4 (2049)	**92.6**	92.5
1/8 (1024)	90.6	**90.7**
1/16 (512)	86.4	**88.2**
1/32 (256)	81.8	**83.9**
1/64 (128)	76.0	**78.2**
1/128 (64)	71.7	**73.5**

Table 7: Average F_1 scores across all slots for RESTAURANTS-8K for ConVEx with and without ensembling. The ConVEx ensemble model fine-tunes 3 decoders per slot, and then averages their output scores.

only 64 annotated examples are available. We can trace a similar behavior on DSTC8, with gains reported for all the DSTC8 single-domain subsets in few-shot setups.

These results point to the following key conclusion. While pretrained representations are clearly useful for slot-labeling dialog tasks, and the importance of pretraining becomes increasingly important when we deal with few-shot scenarios, the chosen pretraining paradigm has a profound impact on the final performance. The pairwise cloze pretraining task, tailored for slot-labeling tasks in particular, is more robust and better adapted to few-shot slot-labeling tasks. This also verifies our hypothesis that it is possible to learn effective domain-specific slot-labeling systems by simply fine-tuning a pretrained general-purpose slot labeler relying only on a handful of domain-specific examples.

SNIPS Evaluation (5-Shot). The versatility of the proposed ConVEx approach is further verified in the 5-shot labeling task on SNIPS following Hou et al. (2020)'s setup. The results are provided in Table 6. We report the highest average F_1 scores with ConVEx. Furthermore, ConVEx also surpasses all the baselines in 4/7 domains, while the highest scores in the remaining three domains are achieved by three different models from Hou et al. (2020). This again hints at the robustness of ConVEx, especially in few-shot scenarios, and shows that the pretrained model can be adapted to a wide spectrum of slot-labeling tasks and domains.

These results also stand in contrast with the previous findings of Hou et al. (2020) where they claimed "*...that fine-tuning on extremely limited examples leads to poor generalization ability*". On the contrary, our results on SNIPS as well as on the other two data sets validate that it is possible to fine-tune a pretrained slot-labeling model directly with a limited number of annotated examples for various domains, without hurting the generalization ability of ConVEx. In other words, we demonstrate that the mainstream "pretrain then fine-tune" paradigm is a viable solution to sequence-labeling tasks in few-shot scenarios, but with the condition that the pretraining task must be structurally well-aligned with the intended downstream tasks.

Further Discussion. We also run experiments to further analyze the main properties of ConVEx. First, we compare the two ConVEx variants with and without the auxiliary loss (see Eq. (1) in §2); the results are provided in Table 4 and Table 5. In general, as already hinted by the intrinsic evaluation on Reddit, the variant with the auxiliary loss has a slight edge over the *no-aux* alternative. The gains are small but consistent on RESTAURANTS-8K, while we see more fluctuation on DSTC8 and several runs where *no-aux* slightly outperforms the full ConVEx. However, the gaps are typically low and can be partially attributed to the inherent variation expected in the low-data setups.

Next, we analyze the benefits of model ensembling, as done in the 5-shot SNIPS task, also on RESTAURANTS-8K. The results across different training data sizes are shown in Table 7. While there is no performance difference when a sufficient number of annotated examples is available,

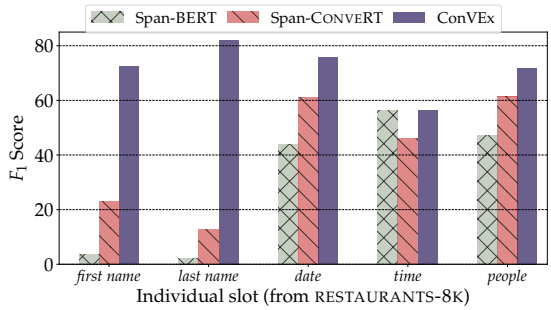


Figure 4: Average F_1 scores for each slot for the Span-ConveRT and ConVEx models trained on $1/128$ of the RESTAURANTS-8K training set, i.e., 64 examples.

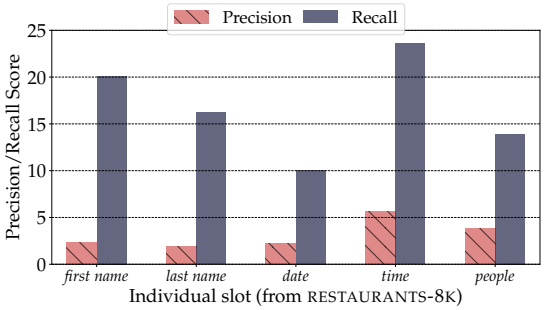


Figure 5: Performance of the ConVEx decoder across all slots in RESTAURANTS-8K without any fine-tuning.

the scores suggest that the model ensembling strategy does yield small but consistent improvements in few-shot scenarios, as it mitigates the increased variance that is typically met in these setups.

4.1 Inductive Bias of ConVEx

In sum, ConVEx outperforms current state-of-the-art slot-labeling models such as Span-ConveRT, especially in low-data settings, where the performance difference is particularly large. The model architectures of Span-{BERT, ConveRT} and ConVEx are very similar: the difference in performance thus arises mainly from the pretraining task, and the fact that ConVEx's sequence-decoding layers are pretrained, rather than learned from scratch. This section analyses the inductive biases of the ConVEx model, that is, how the pretraining regime and the main assumptions affect its behavior and performance before and after fine-tuning.

First, we analyze *per-slot performance* on RESTAURANTS-8K, comparing ConVEx (with the auxiliary loss) with Span-BERT and Span-ConveRT as the previous state-of-the-art models on this evaluation set. The scores in a few-shot scenario with 64 examples are provided in Figure 4, and we observe similar patterns in other few-shot scenarios. The results indicate the largest performance gap for the slots *first name* and *last name*. This is expected, given that by the ConVEx design the keyphrases extracted from Reddit consist of rare words, and are thus likely to cover plenty of names without sufficient coverage in small domain-specific data sets. Nonetheless, it is important to mark prominent gains over the baselines achieved also for the other slots with narrower semantic fields, where less lexical variability is expected (*date* and *people*).

We can also expose the built-in biases of the ConVEx model by applying it with no fine-tuning. Figure 5 shows the results of running ConVEx with no slot-specific fine-tuning on the RESTAURANTS-8K test set, feeding the user input as both the template and input sentence. We extract at most one value from each sentence, where the model predicted a value for 96% of all the test examples, 16% of which corresponded to an actual labeled slot, and 86% did not. The highest recalls were for the name slots, and the time slot, which correlates with the slot-level breakdown results from Figure 4.

The most frequent predictions from the non-finetuned ConVEx decoder that do not correspond to a labeled slot on RESTAURANTS-8K give further insight into its inductive biases. The top 10 extracted non-labeled values are in descending order: *booking, book, reservation, a reservation, a table, indoors, restaurant, cuisine, outside table,* and *outdoors*. Some of these may make sense to be modeled as slot values with an extended ontology, such as *indoors* or *outdoors/outside table*.

5 Conclusion and Future Work

We have introduced *ConVEx* (Conversational Value Extractor), a light-weight pretraining and fine-tuning neural approach to slot-labeling dialog tasks. We have demonstrated that it is possible to learn domain-specific slot labelers even in low-data regimes by simply fine-tuning decoder layers of the pretrained general-purpose ConVEx model.

The ConVEx framework achieves a new leap in performance by aligning the pretraining phase with the downstream fine-tuning phase for sequence labeling tasks. This is realized through the novel *pairwise cloze* pretraining objective, a sentence-pair value extraction task that is structurally aligned with the downstream slot-labeling tasks. While ex-

isting approaches typically require learning additional sequence level layers from scratch, ConVEx requires no new layers and can be fully fine-tuned. We have validated the effectiveness and usefulness of the ConVEx approach to slot labeling across a spectrum of diverse slot-labeling domains and data sets, reporting state-of-the-performance in full-data setups, as well as the strongest gains in the most challenging, few-shot setups.

In future work, we plan to investigate the limits of data-efficient slot labeling, focusing on one-shot and zero-shot setups. We will also apply ConVEx to related tasks such as named entity recognition and conversational question answering.

Acknowledgements

We would like to thank Yutai Hou for sharing the data and evaluation episodes for the SNIPS evaluation. Thanks to our colleagues at PolyAI for fruitful discussions and critical examinations of this work. We would also like to thank Sam Coope and Tyler Farghly for their help with rerunning and validating Span-BERT and Span-ConveRT.

References

Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2016. Conversational contextual cues: The case of personalization and history for response ranking. *CoRR*, abs/1606.00372.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP*, pages 5016–5026.

Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. 2020. DIET: Lightweight language understanding for dialogue systems. *CoRR*, abs/2004.09936.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of EMNLP*, pages 169–174.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL*, pages 8440–8451.

Sam Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. 2020. Span-ConveRT: Few-shot span extraction for dialog with pretrained conversational representations. In *Proceedings of ACL*, pages 107–121.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips Voice Platform: An embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, pages 12–16.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, page 993–1000.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, GP Shrivatsa Bhargav, Dinesh Garg, and Avirup Sil. 2020. Span selection pre-training for question answering. In *Proceedings of ACL*, pages 2773–2782.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Matthew Henderson, Pawel Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, and Tsung-Hsien Wen. 2019a. A repository of conversational datasets. In *Proceedings of the 1st Workshop on Natural Language Processing for Conversational AI*, pages 1–10.

Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. 2020. ConveRT: Efficient and accurate conversational representations from transformers. In *Findings of EMNLP*.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014. The Second Dialog State Tracking Challenge. In *Proceedings of SIGDIAL*, pages 263–272.

Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. 2019b. Training neural response selection for task-oriented dialogue systems. In *Proceedings of ACL*, pages 5392–5404.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of ACL*, pages 1381–1393.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *Proceedings of ICLR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR (Conference Track)*.

Jason Krone, Yi Zhang, and Mona Diab. 2020. Learning to classify intents and slot labels given a handful of examples. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 96–108.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *CoRR*, abs/2006.15020.

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

Ilya Loshchilov and Frank Hutter. 2017. SGDR: stochastic gradient descent with warm restarts. In *Proceedings of ICLR*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of NAACL-HLT: Tutorials*, pages 15–18.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. *Communications of the ACM*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NeurIPS*, pages 4077–4087.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of ACL*, pages 3645–3650.

Charles Sutton and Andrew McCallum. 2012. An introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*, 4(4):267–373.

Gökhan Tür and Renato De Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. John Wiley & Sons.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas*, pages 193–199.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*, pages 6000–6010.

Jason D. Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of SIGDIAL*, pages 282–291.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of NeurIPS*, pages 5754–5764.

Whan Sung Yoon, Jun Seo, and Jaekyun Moon. 2019. TapNet: neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of ICML*, pages 282–291.

Steve Young. 2010. Still talking to machines (cognitively speaking). In *Proceedings of INTERSPEECH*, pages 1–10.

Steve J. Young. 2002. Talking to machines (statistically speaking). In *Proceedings of INTERSPEECH*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Tiancheng Zhao, Kaige Xie, and Maxine Eskénazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. In *Proceedings of NAACL-HLT*, pages 1208–1218.

	people	*time*	*date*	*first name*	*last name*	Total
train	**2164** (547)	**2164** (547)	**1721** (601)	**887** (364)	**891** (353)	8198
test	**983** (244)	**853** (276)	**802** (300)	**413** (177)	**426** (174)	3731

Table 8: The number of examples for each slot in the RESTAURANTS-8K data set. Numbers in brackets show how many examples have the slot *is_requested*.

Sub-domain	Train Size	Test Size	Slots
Buses_1	1133	377	from_location (169/54), leaving_date (165/57), to_location (166/52)
Events_1	1498	521	city_of_event (253/82), date (151/33), subcategory (56/26)
Homes_1	2064	587	area (288/86), visit_date (237/62)
RentalCars_1	874	328	dropoff_date (112/42), pickup_city (116/48), pickup_date (120/43), pickup_time (119/43)

Table 9: Statistics of the used data sets extracted from the DSTC8 schema-guided dialog dataset. The number of examples in the train and test sets for each slot are reported in parentheses.

Domain	# of Sentences	Labels
Weather	2,100	10
Music	2,100	10
Playlist	2,042	6
Book	2,056	8
Search	2,059	8
Restaurants	2,073	15
Creative	2,054	3

Table 10: Statistics of the original SNIPS data set.

A Evaluation Data Statistics

For completeness, we provide the summary stats of the evaluation data used in our work:

Table 8 shows the statistics of the RESTAURANTS-8K data set. The data set is available at: `github.com/PolyAI-LDN/task-specific-datasets`.

Table 9 shows the statistics of the DSTC8 data set. The data set is available at: `github.com/PolyAI-LDN/task-specific-datasets`.

Table 10 provides the statistics of the original SNIPS data set (Coucke et al., 2018), For further details on how the data set has been used in the 5-shot evaluation setup we refer the reader to the work of Hou et al. (2020). The data sets are available at: `github.com/AtmaHou/FewShotTagging`

B Baseline Models in the SNIPS Evaluation

This appendix provides a brief summary of the models from Hou et al. (2020) included in the SNIPS evaluation (Table 6) alongside ConVEx.

TransferBERT is a direct application of BERT (Devlin et al., 2019) to sequence labeling. It is first trained on the source domains. As the sequence labeling layers are domain-specific, these are then removed, and new layers are fine-tuned on the in-domain training set (i.e., Hou et al. (2020) refer to it as the *support set*; this is exactly what we use for fine-tuning ConVEx).

SimBERT predicts sequence labels according to the cosine similarity between the representations from a BERT model of the input tokens with tokens in the support set, selecting the labels of the most similar labeled tokens.

WarmProtoZero (WPZ) (Fritzler et al., 2019) applies Prototypical Networks (Snell et al., 2017) to sequence labeling tasks. It treats sequence-labeling as word-level classification, and can either use randomly initialized word embeddings, or pretrained representations in the case of WPZ+BERT.

TapNet is a few-shot learning paradigm originally applied to image classification (Yoon et al., 2019). This works similarly to Prototypical Networks, but includes a task-adaptive network that projects examples into a space where words of differing labels are well separated.

Collapsed Dependency Transfer (CDT) is a technique for simplifying transition dynamics of a CRF, applied to both TapNet and WPZ. This represents the full transition matrix using shared abstract transitions, e.g. modeling transitions between any *Begin* tag to the *Begin* tag of any different slot using a shared probability.

Label Enhanced models, denoted *L-WPZ* and *L-TapNet* use the semantics of the label names themselves to enrich the word-label similarity modeling.