SoDA: On-device Conversational Slot Extraction

Sujith Ravi
SliceX AI
Menlo Park, CA
sravi@sravi.org

Zornitsa Kozareva
Google
Mountain View, CA
zornitsa@kozareva.com

Abstract

We propose a novel on-device neural sequence labeling model which uses embedding-free projections and character information to construct compact word representations to learn a sequence model using a combination of bidirectional LSTM with self-attention and CRF. Unlike typical dialog models that rely on huge, complex neural network architectures and large-scale pre-trained Transformers to achieve state-of-the-art results, our method achieves comparable results to BERT and even outperforms its smaller variant DistilBERT on conversational slot extraction tasks. Our method is faster than BERT models while achieving significant model size reduction–our model requires 135x and 81x fewer model parameters than BERT and DistilBERT, respectively. We conduct experiments on multiple conversational datasets and show significant improvements over existing methods including recent on-device models. Experimental results and ablation studies also show that our neural models preserve tiny memory footprint necessary to operate on smart devices, while still maintaining high performance.

1 Introduction

In today's world, people rely on their digital devices like mobile phones, smartwatches, home assistants like Google and Alexa to alleviate mundane tasks like play favorite songs, recommend food recipes among others. A big part of the language understanding capabilities of such assistive devices happens on cloud, where the relevant slots, entities and intents are extracted in order for the request to be fulfilled. However, is it not always safe to send data to cloud, or when we travel it is not always possible to have internet connectivity, yet we want to enjoy the same capabilities.

These challenges can be solved by building on-device neural models that can perform inference on device and extract the slot (entity) information needed for language understanding. The model will operate entirely on the device chip and will not send or request any external information. Such on-device models should have low latency, small memory and model sizes to fit on memory-constrained devices like mobile phones, watches and IoT.

Recently, there has been a lot of interest and novel research in developing on-device models. Large body of work focuses on wake word detection (Lin et al., 2018; He et al., 2017), text classification like intent recognition (Ravi and Kozareva, 2018), news and product reviews (Kozareva and Ravi, 2019; Ravi and Kozareva, 2019; Sankar et al., 2021b,a).

In this paper, we propose a novel on-device neural sequence tagging model called SoDA . Our novel approach uses embedding-free projections and character-level information to construct compact word representations and learns a sequence model on top of the projected representations using a combination of bidirectional LSTM with self-attention and CRF model. We conduct exhaustive evaluation on different conversational slot extraction datasets. The main contributions of our work are as follows:

- Introduced a novel on-device neural sequence tagging model called SoDA .

- Our novel neural network dynamically constructs embedding-free word representations from raw text using embedding-free projections with task-specific *conditioning* and CNN together with a bidirectional LSTM coupled with self-attention and CRF layer. The resulting network is compact, does not require storing any pre-trained word embedding tables or huge parameters, and is suitable for on-device applications.

- Conducted exhaustive evaluation on multiple conversational slot extraction tasks and demonstrate that our on-device model SoDA reaches state-of-the-art performance and even outperforms larger, non-on-device models like Capsule-NLU (Zhang et al., 2019), StackPropagation (Qin et al., 2019), Interrelated SF-First with CRF (E et al., 2019), joint BiLSTM (Hakkani-Tur et al., 2016), attention RNN (Liu and Lane, 2016), gated attention (Goo et al., 2018) and even BERT models (Sanh et al., 2019).

- Our on-device SoDA model also significantly outperforms state-of-the-art on-device slot extraction models of (Ahuja and Desai, 2020), which are based on convolution and are further compressed with structured pruning and distillation.

- Finally, we conduct a series of ablation studies that show SoDA 's compact size needed for conversational assistant devices like Google and Alexa, smart watches while maintaining high performance.

2 SoDa: On-device Sequence Labeling

In this section, we describe the components of our SoDA architecture as shown in Figure 1.

2.1 Input Word Embeddings

Given an input text X containing a sequence of words $(x_1, x_2, ..., x_n)$, where x_i refers to i-th word in the sentence, we first construct a sequence of vectors $\mathcal{E}(X) = (e_1, e_2, ..., e_n)$ where e_i denotes a vector representation for word x_i.

2.1.1 Word Embedding via Projection

Learning good representations for word types from the limited training data (as in slot extraction) is challenging since there are many parameters to estimate. Most neural network approaches for NLP tasks rely on word embedding matrices to overcome this issue. Almost every recent neural network model uses pre-trained word embeddings (e.g., Glove (Pennington et al., 2014), word2vec (Mikolov et al., 2013)) learned from a large corpus that are then plugged into the model and looked up to construct vector representations of individual words and optionally fine-tuned for the specific task. However, these embedding matrices are often huge and require lot of memory

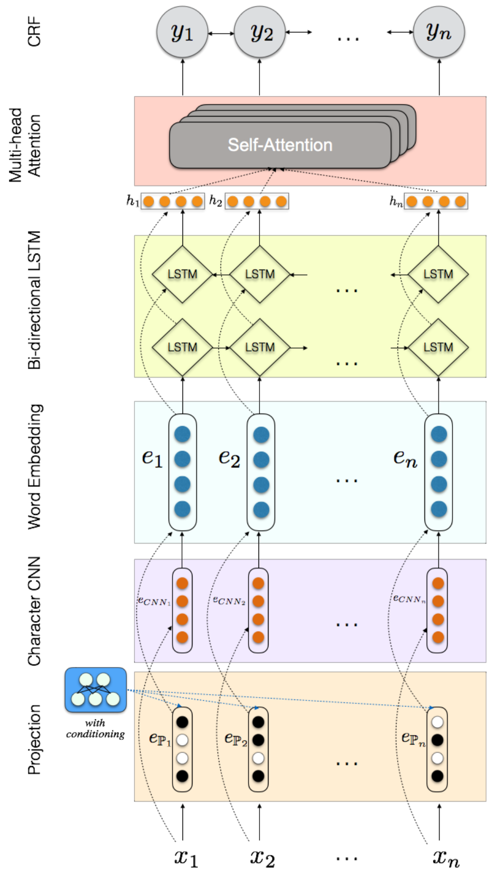

Figure 1: Model architecture for SoDA On-device Sequence Labeling Neural Network.

$O(V \cdot d)$ which is infeasible for on-device applications where storage is limited. Here, V is the vocabulary size and can be huge from 100K to millions of entries, and d is the embedding dimension. For example, using 300-dimensional Glove embeddings with 400K entries and `float32` values requires `480MB` in storage for the embedding table alone. Even without any pre-training, $O(V \cdot d)$ parameters still need to be estimated which contributes to the model size and latency. Even methods that resort to sub-word sequences and reduce vocabulary size requires explicitly storing and looking up these parameters. For English, simple character trigrams with 36 alphanumeric characters requires $V = 36^3 = 47K$ entries in the embedding matrix.

Embedding-free Projections: For generating $\mathcal{E}(X)$, we compute e_i word vector representations *dynamically* building on a locality-sensitive projection approach similar to (Ravi, 2017).

For each word x, we extract character-level information (i.e., character sequences) from the word to construct a sparse feature vector $\mathcal{F}(x_i)$.

$$\mathcal{F}(x) = \{\langle f_1, w_1 \rangle, ..., \langle f_K, w_K \rangle\} \qquad (1)$$

where, f_k represents each feature id (`Fingerprint` of the raw character skip-gram) and w_k its corresponding weight (observed count in the specific input x).

We use locality-sensitive projections (Ravi, 2017) to dynamically transform the intermediate feature vector $\mathcal{F}(x)$ to binary representation $\mathbb{P}(x)$.

$$\begin{aligned} \mathbb{P}(x) &= \mathbb{P}(\mathcal{F}(x)) && (2) \\ &= \mathbb{P}(\{\langle f_1, w_1\rangle, ..., \langle f_K, w_K\rangle\}) && (3) \end{aligned}$$

This step uses locality-sensitive hashing (LSH) (Charikar, 2002) to convert the high-dimensional sparse feature vector $\mathcal{F}(x)$ into a very compact, low-dimensional binary representation *on-the-fly*. The transformation uses a series of d binary hash functions $\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_d$ to generate a binary value (`-1` or `+1`) for each dimension j resulting in a d-dimensional binary vector. Each binary hash function is parameter-free since we only use the dimension id j and observed features ids f_k to construct a randomized vector $\mathcal{R}_j(x)$ with same number of non-zero entries r_K as $\mathcal{F}(x)$.

$$\begin{aligned} \mathcal{R}_j(x) &= \{\langle f_1, r_1\rangle, ..., \langle f_K, r_K\rangle\} && (4) \\ \mathbb{P}_j(x) &= \mathtt{sgn}(\mathcal{R}_j(x) \cdot \mathcal{F}(x)) && (5) \\ \mathbb{P}(x) &= \langle \mathbb{P}_1(x), \mathbb{P}_2(x), ..., \mathbb{P}_d(x)\rangle && (6) \end{aligned}$$

For our sequence tagging model, we use $\delta \cdot d$ projection dimensions to model character sequences occurring in the word (up to *5-grams, 0-skip* character-level features). We use the remaining $(1 - \delta) \cdot d$ dimensions to model the whole word feature. For sequence tagging experiments, we set $\delta = 0.9$. The projection operations $\mathbb{P}_j$ can be computed fast and on-the-fly during training and inference without any embedding tables or additional parameters. The locality-sensitive nature of the projections enable learning a compact representation that captures semantic similarity (at word and sub-word level) in the high-dimensional space with a small memory footprint. For more details on projection operations, refer (Ravi, 2017).

Conditioning Projections: We could use the dynamically constructed projection vector $\mathbb{P}(x)$ directly instead of embeddings to build the rest of our model. But to prevent the models from depending on static projection representations too strongly, we further *condition* or fine-tune the projections on specific sequence tagging task during training to learn better task-specific representations $\mathcal{E}(x)$.

Note that unlike prior approaches that use pre-trained embeddings and fine-tune the $O(V \cdot d)$ parameters on individual tasks, we use far fewer parameters $O(M); M \ll V \cdot d$ for the *projection conditioning* step so as to keep the resulting model size compact and not incur huge additional memory or time complexity for inference on device.

For sequence tagging, we apply two types of conditioning operators on the projection output $\mathbb{P}(.)$ to generate the final $\mathcal{E}(.)$ vector representations for words in the input sequence.

- *Hadamard product* ($\circ$):

$$\mathcal{E}(X) = \mathbb{P}(X) \circ \mathbf{W}_{c_H} + \mathbf{b}_{c_H} \quad (7)$$

  where, $\mathcal{E}(X)$ is the embedding for the input sequence of size $n \times d$. $\mathbf{W}_{c_H}$ and $\mathbf{b}_{c_H}$ are d trainable weight and bias parameters used for projection conditioning which are shared across all words. Using point-wise operations for this conditioning requires only d multiply and d add operations, keeping the number of parameters $M = 2d$ in this step very small.

- *Dense product* (D):

$$\mathcal{E}(X) = \mathbb{P}(X) \times \mathbf{W}_{c_D} + \mathbf{b}_{c_D} \quad (8)$$

  here $\mathbf{W}_{c_D}$ is a trainable shared weight matrix of size $d \times m$ and $\mathbf{W}_{c_D}$ represents bias parameters. We choose $m \leq d$, so total number of conditioning parameters $M = d \cdot (m + 1)$.

As noted, both projection conditioning operators result in a tiny number of additional model parameters $M \ll V \cdot d$ that are tuned during training.

2.1.2 Extending Character-level Representation using CNN

Earlier work (Chiu and Nichols, 2016; Ma and Hovy, 2016) showed that CNNs can be effective to model morphological information within words and encode it within neural networks using character-level embeddings. However, these approaches typically compute both word-level (from pre-trained tables) and character-level embeddings (to model long sequence contexts) and combine them to construct word vector representations in their neural network architectures.

However as we noted, word embedding lookup tables incur significant memory that are not suitable for on-device usecases. Previous results on sequence labeling (Ma and Hovy, 2016) show that character embeddings by themselves do not have the same generalizability power of word embeddings trained on large corpora, especially for names and common words appearing in regular text.

Our model SoDA uses the best of both approaches, by first constructing word embeddings using conditioned projections as described in Section 2.1.1. We further extend this with a character CNN model with shared, trainable parameters to augment the morphology information. The CNN used in our model is similar to (Chiu and Nichols, 2016; Ma and Hovy, 2016). The combined embedding layer in the SoDA model still maintains a small number of parameters ($\ll V \cdot d$), corresponding to projection conditioning and convolutions.

$$\mathcal{E}(X) = \texttt{concat}(\mathcal{E}_{\mathbb{P}}(X), \mathcal{E}_{CNN}(X)) \quad (9)$$

A dropout layer (Srivastava et al., 2014) is then applied to the joint embedding $\mathcal{E}(X)$ for regularization before being passed as input to the next layer in the SoDA neural network.

2.2 Bi-directional LSTM

Next, we apply a recurrent neural network (RNN) to operate on the sequence of projected vectors $\mathcal{E}(X) = (e_1, e_2, ..., e_n)$. We use LSTMs (Hochreiter and Schmidhuber, 1997) over the *projected* word sequences to model the temporal dynamics across the sequence to produce a state sequence $\mathcal{H}(X) = (h_1, h_2, ..., h_n)$, where h_i captures higher-level information about the sequence at time step i. LSTM is a variant of RNN with memory cells that enable capturing long-distance dependencies. LSTMs are composed of multiple gates to control the proportion of information to *forget* and *pass* through to the next time step. We use the following implementation in SoDA

For an input sentence $X = (x_1, x_2, ..., x_n)$ and corresponding sequence of projected embeddings $\mathcal{E}(X)$, where each $e_t = [e_{\mathbb{P}_t} \cdot e_{CNN_t}]$ is a d-dimensional vector, the LSTM layer in SoDA uses *input*, *forget* and *output* gates to compute a new state h_t at time step t. For sequence tagging tasks, both left and right contexts are useful to represent information at any time step. Standard LSTM as well as other sequence models only account for previous history and know nothing about the future. We use a bi-directional LSTM (Dyer et al., 2015) to efficiently model both past and future information in our SoDA model. The only change required is that model a separate forward and backward hidden state, which are updated in the same manner and concatenated to form the final output state. We also create deeper SoDA sequence models by stacking multiple bi-LSTM layers to get the projected sequence output $\mathbb{P}_{bi-LSTM}(X)$.

2.3 Self-Attention for Sequence

Attention mechanisms have become a core component of powerful neural networks used for various sequence labeling tasks (Bahdanau et al., 2014; Kim et al., 2017). Adding this to a neural sequence network allows modeling of positional dependencies without regard to their distance in the input or output sequences. This has proven particularly useful for modeling complex sequence tasks such as machine translation and led to powerful deep, attention-based neural network architectures (Vaswani et al., 2017) in recent years.

We add self-attention on top of the bi-LSTM output $\mathbb{P}_{bi-LSTM}(X)$ in SoDA to model positional dependencies in the sequence. Self-attention relates different positions of an input sequence to compute a representation of the sequence and has been successfully applied to tasks such as reading comprehension, abstractive summarization, and learning task-independent sentence representations (Cheng et al., 2016; Paulus et al., 2018; Lin et al., 2017). We use a multi-head attention (Vaswani et al., 2017) with H heads that allows SoDA sequence model to jointly attend to information from multiple representation sub-spaces at different positions. The output from the projected bi-LSTM network followed by self-attention layer in SoDA is a sequence representation denoted by $\mathcal{S}_{\mathbb{P}_{bi-LSTM}}(X)$.

2.4 CRF Tagging Model

For structured prediction tasks like sequence tagging, it is useful to model the dependencies between neighboring labels (Ling et al., 2015) and perform joint decoding of the label sequence for a given input sentence. For example, in sequence labeling tasks with `BIO` tagging scheme `I-LOC` label cannot follow `B-PER`. So, instead of decoding labels at every position separately, similarly to prior work, we perform joint decoding in our model using a condition random field (CRF) (Lafferty et al., 2001).

For an input sentence $X = (x_1, x_2, ..., x_n)$, the intermediate output vector from the projected bi-LSTM network is denoted by $\mathcal{S}_{\mathbb{P}_{bi-LSTM}} = (s_1, s_2, ..., s_n)$, where s_i represents the concatenated vector combining the forward and backward states of the projected bi-LSTM at position i. $Y = (y_1, y_2, ..., y_n)$ represents the final output tag sequence for the sentence given $\mathcal{S}$, output from the previous layer. $Y \in \mathcal{Y}(\mathcal{S})$, where $\mathcal{Y}(\mathcal{S})$ denotes the set of all possible tag sequences for $\mathcal{S}$. We define the probabilistic CRF sequence model as a conditional probability $p(Y|\mathcal{S};\theta)$ over all possible label sequences Y given $\mathcal{S}$ as follows:

$$p(Y|\mathcal{S};\theta) = \frac{\prod_{i=1}^{n} \phi_i(y_{i-1}, y_i, \mathcal{S})}{\sum_{y' \in \mathcal{Y}(\mathcal{S})} \prod_{i=1}^{n} \phi_i(y'_{i-1}, y'_i, \mathcal{S})} \quad (10)$$

where, $\phi_i(y_j, y_k, \mathcal{S}) = \exp(\mathbf{W}_\theta^T s_i + \mathbf{b}_\theta)$ is a parameterized transition matrix with weights $\mathbf{W}_\theta$ and bias $\mathbf{b}_\theta$ that scores transition from tag y_j to y_k for each position i in the sentence. The transition matrix is a square matrix of size L, where L represents the number of distinct tag labels that includes special begin and end tags for a sentence.

We use maximum-likelihood estimation to jointly optimize the CRF parameters θ along with other network parameters during training $L_\theta(.) = \sum_i \log p(Y|\mathcal{S};\theta)$. Since we only use first-order transition dependencies between labels, the partition functions can be computed efficiently using the Viterbi algorithm for both training and inference. Once trained, we perform sequence decoding as follows $\mathbf{y}^* = \text{argmax}_{Y \in \mathcal{Y}(\mathcal{S})}\, p(Y|\mathcal{S};\theta_{trained})$.

2.5 Putting it all together: SoDA Network

Finally, we construct our end-to-end on-device neural network SoDA by combining all components progressively: *word representation* (using conditioned projections + CNN), *projected bi-LSTM sequence model* with *self-attention layer* and *CRF* layer. The input sequence X is passed through the on-device SoDA network and final layer to get decoded output tag sequence Y.

3 SoDA Training and Parameters

We now describe details for training the on-device SoDA neural network. We implement the model using TensorFlow. For each sequence labeling task, we train the parameters of the model on the corresponding dataset, then apply the same steps in order for inference and evaluate the decoded tag sequence output against the gold label sequence.

3.1 Optimization

During training, we estimate the SoDA parameters with Adam optimizer (Kingma and Ba, 2014) that is applied over shuffled mini-batches of size 20. We choose an initial learning rate of `1e-3` with gradient clipping.

Early Stopping: We use early stopping (Caruana et al., 2000) based on performance on held-out dev sets. In our experiments, we typically observe good validation performance within 10-20 epochs.

Conditioning Projections: As described in Section 2.1.1, we condition the dynamically constructed projected word representations to learn task-specific projection parameters. We use two different types of conditioning operators: *Hadamard* ($\circ$), and *Dense* (D). We choose $m = d$ for the dense version, yielding $M = d^2 + d$ parameters and $M = 2d$ for the former. We observed that the *Dense* version with slightly more parameters performed better overall on sequence tasks and hence use this as the default version for SoDA in our experiments. We did not do any data or task-specific tuning or processing.

Dropout: During training, we apply dropout (Srivastava et al., 2014) for regularization in our model with a fixed rate `0.3`.

3.2 Hyper-Parameters

Word Representations: We use $d = 300$ projection size for $\mathcal{E}_\mathbb{P}(.)$. Unlike other neural models, our on-device network does **not** require storing and loading any pre-trained word embedding matrices and does **not** need any $O(V \cdot d)$ parameters for modeling the vocabulary. Hence, we do not have to apply any pruning techniques to keep vocabularies small.

Projected Sequence Layer: For the sequence layer we use 2-layer bi-LSTM with `100` state size.

Self-Attention Layer: We set $H = 4$ heads for the multi-head attention model and attention size = bi-LSTM state size.

CRF Tagging: We use CRF model as the default output model for all SoDA networks.

4 Datasets and Experimental Setup

4.1 Dataset Description

We evaluate our on-device SoDA model on widely used and popular conversational slot extraction datasets.

• **ATIS: Slot Extraction** The Airline Travel Information Systems dataset (Tür et al., 2010) is

Model	ATIS F1	ATIS Sent. Acc.	SNIPS F1	SNIPS Sent. Acc.
SoDA (our on-device model)	**95.8**	**88.1**	**93.6**	**85.1**
DistillBERT (66M) (Ahuja and Desai, 2020; Sanh et al., 2019)	95.4↑	-	94.6	-
BERT (110M) (Ahuja and Desai, 2020; Devlin et al., 2019)	96.0	-	95.1	-
Capsule-NLU (Zhang et al., 2019)	95.2 ↑	83.4 ↑	91.8 ↑	80.9 ↑
StackPropagation (Qin et al., 2019)	95.9	86.5↑	94.2	86.9
Interrelated SF-First with CRF (E et al., 2019)	95.7 ↑	86.8 ↑	91.4↑	80.6↑
GatedFullAtten. (Goo et al., 2018)	94.8 ↑	82.2 ↑	88.8 ↑	75.5 ↑
GatedIntentAtten. (Goo et al., 2018)	95.2 ↑	82.6 ↑	88.3 ↑	74.6 ↑
JointBiLSTM (Hakkani-Tur et al., 2016)	94.3 ↑	80.7 ↑	87.3 ↑	73.2 ↑
Atten.RNN (Liu and Lane, 2016)	94.2 ↑	78.9 ↑	87.8 ↑	74.1 ↑

Table 1: Comparison of **SoDA** against other **Non-On-Device** Conversational Slot Extraction Methods. All methods are significantly larger in model size than SoDA ; ↑ indicates SoDA improvement

Model	ATIS (F1)	SNIPS(F1)
SoDA (our on-device model)	**95.83**	**93.6**
Convolution (Ahuja and Desai, 2020)		
Single-task	94.01↑	85.06 ↑
Multi-task	94.30↑	84.38↑
Convolution-Compressed (Ahuja and Desai, 2020)		
Structured Pruning Single-task	94.61↑	85.11↑
Structured Pruning Multi-task	94.42↑	83.81 ↑

Table 2: Comparison of **SoDA** against other **On-Device** Conversational Slot Extraction Methods; ↑ indicates SoDA improvement

widely used in spoken language understanding research. The dataset contains audio recordings of people making flight reservations. We used the same data as (Tür et al., 2010; Goo et al., 2018).

• **SNIPS: Slot Extraction** To verify the generalization of the proposed model for slot extraction, we use another natural language understanding dataset with custom intent-engines collected by the Snips personal voice assistant. We used the data from (Goo et al., 2018). Compared to the single-domain ATIS dataset, Snips has multiple domains resulting in larger vocabulary.

Table 3 shows the characteristics of the two conversational slot extraction datasets such as number of entity/slot types, number of sentences in train and test data.

Dataset	#Slot Types	Train	Test
ATIS	120	4,478	893
SNIPS	72	13,084	700

Table 3: Conversational Slot Extraction Dataset Characteristics

4.2 Experimental Setup & Metrics

We setup our experiments as given a sequence labeling task and a dataset, we train an on-device SoDA model. Similarly to prior work, for each ATIS and SNIPS datasets, we report F_1 score on the test set and the overall sentence accuracy (Hakkani-Tur et al., 2016; Goo et al., 2018).

5 Results for Conversational Slot Extraction

This section presents results from the conversational slot extraction task on the ATIS and SNIPS datasets. Tables 1 and 2 show the obtained results from our on-device SoDA approach, which outperformed prior state-of-the-art on-device slot extractors based on single and multi-task convolution including the compressed convolution models (Ahuja and Desai, 2020). Our on-device SoDA even outperformed prior non-on-device state-of-the-art neural models like Capsule-NLU, StackPropagation, RNN, CNN, Gated full attention, joint intent-slot modeling and even BERT models on ATIS and SNIPS datasets.

5.1 Comparison with On-Device State-of-the-art Slot Extractors

An important study in this work is a comparison between our on-device model against prior state-of-the-art on-device slot extraction models (Ahuja and Desai, 2020). The models of (Ahuja and Desai, 2020) are based on simple convolution model compressed with structured pruning. Two variations of this model are developed: single task where only one task is performed like slot extraction and multi-task model where two conversational tasks (slot extraction and intent detection) are jointly optimized. The multi-task approach was commonly used in earlier works (Hakkani-Tur et al., 2016) to improve the performance of the individual tasks. (Ahuja and Desai, 2020) further compressed these models with structured pruning and distillation. As shown in Table 2, SoDA outperforms the convolution single and multi-task approaches by 1.82% for ATIS and 8.54% for SNIPS datasets. Similarly, SoDA outperforms even the compressed single and multi-task model variants by 1.22% ATIS and 9.76% for SNIPS without relying on pruning or distillation. The significant performance improvements for SoDA model stem from the memory-efficient and robust projection representations which better capture word and semantic similarity.

5.2 Comparison with Non-On-Device Slot Extractors

The main objective of on-device work is to develop small and efficient models that fit on devices with limited memory and capacity. In contrast, non-on-device models do not have any memory and capacity constraints, as they use all resources available on the server side. Therefore, a direct comparison between on-device and non-on-device models is not fair. Taking into consideration these major differences, we show in Table 1 results from SoDA and state-of-the-art non-on-device models with the objective to highlight the power of our on-device work in achieving competitive results and even outperforming widely used approaches such as Capsule-NLU, StackPropagation, RNN, CNN, Gated full attention, joint intent-slot modeling and even BERT models on ATIS and SNIPS datasets.

SoDA on-device model significantly improves over Capsule-NLU (Zhang et al., 2019) which uses capsule networks to model semantic hierarchy between words, slots and intent using dynamic routing by agreement schema. SoDA also improves over the Interrelated SF-First with CRF approach (E et al., 2019), which uses BiLSTM with attentive sub-networks for slot and intent modeling. Similarly, improvements are seen over the attention RNN model (Liu and Lane, 2016) on ATIS and SNIPS. SoDA also achieves better performance than the joint BiLSTM model of (Hakkani-Tur et al., 2016), which uses intents to guide the possible slot types associated with the intent. Unlike those approaches, SoDA does not use any additional information such as the intent classes to further constraint the slot types nor it uses any pre-trained embeddings, yet SoDA achieves better performance than the joint BiLSTM models and capsule networks on both datasets.

Finally, we also compare results against the most recent state-of-the-art neural models of (Goo et al., 2018). Both models are non-on-device. One uses full attention, while the other uses gated intent attention for the slot extractor. Overall, SoDA significantly improves over both gated attention neural models (Goo et al., 2018) with +0.6% to +1% accuracy on ATIS and +4.8% to +5.3% accuracy on SNIPS. This is pretty impressive given that SoDA does not rely on any intent information to constraint the slot type during extraction and also SoDA is an embedding free method that learns the representations on the fly resulting in producing magnitudes smaller models, which remain highly accurate.

We also compare our approach SoDA against much larger, contemporary BERT models (Devlin et al., 2019; Sanh et al., 2019) that rely on large-scale, pre-trained Transformer networks. Surprisingly, SoDA achieves comparable results to BERT and even outperforms its memory-optimized variant DistilBERT (Sanh et al., 2019) while achieving 135x and 81x compression rates, respectively.

6 SoDA Performance Analysis

Next, we show various ablation studies that evaluate the performance of different SoDA components.

6.1 Parameters vs F_1

We study the impact of the number of parameters on SoDA F_1 performance. We control the model size by varying the parameters corresponding to the projection and BiLSTM state sizes. For instance, on ATIS SoDA achieves 95.83% F_1 with 814556 parameters; 94.75% with 212540 parameters; 93.85% with 73290 parameters; 92.69% with as few as 59365 parameters. This study shows that even with less parameters, SoDA achieves high performance.

6.2 Model Size vs F_1

We study how the model size affects SoDA 's performance. Figure 2 shows results of the model size with the corresponding F_1 of SoDA on ATIS slot extraction. Even with very small memory size of 286KB SoDA still achieves high performance of 93.85 F_1. Moreover, SoDA achieves results comparable to BERT Transformer models but at a tiny fraction of the model size.

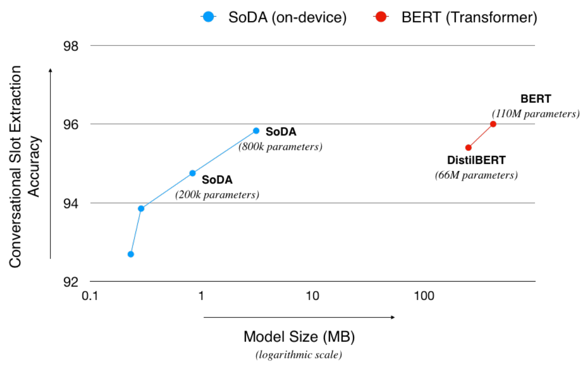

Figure 2: Effect of SoDA and BERT Model Sizes on Slot Extraction Accuracy for ATIS.

6.3 Impact of Projection Conditioning on F_1

We compare the projection conditioning mechanisms we introduced. On ATIS, the Hadamard ($\circ$) conditioning reaches 94.8% F_1 vs Dense (D) conditioning reaches 95.8% F_1. This comparison shows that Dense conditioning is better.

6.4 Impact of CNN on F_1

We evaluate the impact of CNN model on SoDA for ATIS. SoDA without CNN reaches 88.85% F_1 compared to 95.8% F_1 for SoDA with CNN. This shows that adding character information to embedding-free projections further boosts performance for on-device sequence tagging.

6.5 Impact of CRF on F_1

We evaluate the impact of CRF model on SoDA for ATIS. Adding CRF to the SoDA model yields +1.07% going from 94.73% to 95.80% F_1, which shows the benefit of CRF also for on-device.

6.6 Efficiency/Speed of Training Time on Single CPU

Training SoDA on a single machine with CPU $1.3GHz$ Intel core and 8GB memory for ATIS takes 9.6 min to converge with 0.8 min per epoch with 56K tokens. Inference takes $<< 10ms$ on Nexus 5 smartphone device which is an order magnitude faster than DistilBERT and BERT models running on CPU.

7 Conclusion

We introduced a novel on-device conversational slot extraction model called SoDA which uses embedding-free projections and character information to construct compact word representations, and then learn a sequence model using a combination of bidirectional LSTM with self-attention and CRF. We evaluate our approach on multiple slot extraction datasets. Our on-device model SoDA achieves state-of-the-art results and also improved over non-on-device models like Capsule-NLU (Zhang et al., 2019), StackPropagation (Qin et al., 2019), Interrelated SF-First with CRF (E et al., 2019), joint BiLSTM (Hakkani-Tur et al., 2016), attention RNN (Liu and Lane, 2016), gated attention (Goo et al., 2018) and even BERT models (Sanh et al., 2019).

Our on-device SoDA model also significantly outperforms state-of-the-art on-device slot extraction models of (Ahuja and Desai, 2020), which are based on convolution and are further compressed with structured pruning and distillation.

As shown in the evaluation and ablation studies, unlike existing large neural networks that rely on additional information such as pre-trained embeddings, intent information and knowledge bases, SoDA does not use any external resources, and yet it achieves good performance, while maintaining compact size.

References

Ojas Ahuja and Shrey Desai. 2020. Accelerating natural language understanding in task-oriented dialog. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 46–53, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Rich Caruana, Steve Lawrence, and C Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. volume 13, pages 402–408.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings* *of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA. ACM.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561. Association for Computational Linguistics.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. 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)*, pages 334–343. Association for Computational Linguistics.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471, Florence, Italy. Association for Computational Linguistics.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. 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)*, pages 753–757. Association for Computational Linguistics.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw. 2017. Streaming small-footprint keyword spotting using sequence-to-sequence models. *CoRR*, abs/1710.09617.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. *CoRR*, abs/1702.00887.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zornitsa Kozareva and Sujith Ravi. 2019. ProSeqo: Projection sequence networks for on-device text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3894–3903, Hong Kong, China. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zhong Qiu Lin, Audrey G. Chung, and Alexander Wong. 2018. Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge. *CoRR*, abs/1810.08559.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR*, abs/1508.02096.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling,

Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2078–2087, Hong Kong, China. Association for Computational Linguistics.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 804–810.

Sujith Ravi and Zornitsa Kozareva. 2019. On-device structured and context partitioned projection networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3784–3793, Florence, Italy. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2021a. On-device text representations robust to misspellings via projections. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2871–2876.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2021b. ProFormer: Towards on-device LSH projection based transformers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2823–2828. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2010. What is left to be understood in atis? In *Proceedings of 2010 IEEE Spoken Language Technology Workshop (SLT)*, pages 19–24.

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

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