Domain Attention with an Ensemble of Experts

Young-Bum Kim[†] **Karl Stratos**[‡] **Dongchan Kim**[†]

[†]Microsoft AI and Research
[‡]Bloomberg L. P.
{ybkim, dongchan.kim}@microsoft.com
me@karlstratos.com

Abstract

An important problem in domain adaptation is to quickly generalize to a new domain with limited supervision given K existing domains. One approach is to retrain a global model across all $K + 1$ domains using standard techniques, for instance Daumé III (2009). However, it is desirable to adapt without having to reestimate a global model from scratch each time a new domain with potentially new intents and slots is added. We describe a solution based on attending an ensemble of domain experts. We assume K domain-specific intent and slot models trained on respective domains. When given domain $K + 1$, our model uses a weighted combination of the K domain experts' feedback along with its own opinion to make predictions on the new domain. In experiments, the model significantly outperforms baselines that do not use domain adaptation and also performs better than the full retraining approach.

1 Introduction

An important problem in domain adaptation is to quickly generalize to a new domain with limited supervision given K existing domains. In spoken language understanding, new domains of interest for categorizing user utterances are added on a regular basis[1]. For instance, we may add `ORDERPIZZA` domain and desire a domain-specific intent and semantic slot tagger with a limited amount of training data. Training only on the target domain fails to utilize the existing resources in other domains that are relevant (e.g., labeled data for `PLACES` domain with `place_name`, `location` as the slot types), but naively training on the union of all domains does not work well since different domains can have widely varying distributions.

Domain adaptation offers a balance between these extremes by using all data but simultaneously distinguishing domain types. A common approach for adapting to a new domain is to retrain a global model across all $K + 1$ domains using well-known techniques, for example the feature augmentation method of Daumé III (2009) which trains a single model that has one domain-invariant component along with $K + 1$ domain-specific components each of which is specialized in a particular domain. While such a global model is effective, it requires re-estimating a model from scratch on all $K + 1$ domains each time a new domain is added. This is burdensome particularly in our scenario in which new domains can arise frequently.

In this paper, we present an alternative solution based on attending an ensemble of domain experts. We assume that we have already trained K domain-specific models on respective domains. Given a new domain $K+1$ with a small amount of training data, we train a model on that data alone but queries the K experts as part of the training procedure. We compute an attention weight for each of these experts and use their combined feedback along with the model's own opinion to make predictions. This way, the model is able to selectively capitalize on relevant domains much like in

MoE那篇有没有考虑目标领域的意见？

[1]A scenario frequently arising in practice is having a request for creating a new virtual domain targeting a specific application. One typical use case is that of building natural language capability through intent and slot modeling (without actually building a domain classifier) targeting a specific application.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 643–653
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1060

standard domain adaptation but without explicitly re-training on all domains together.

In experiments, we show clear gains in a domain adaptation scenario across 7 test domains, yielding average error reductions of 44.97% for intent classification and 32.30% for slot tagging compared to baselines that do not use domain adaptation. Moreover we have higher accuracy than the full re-training approach of Kim et al. (2016c), a neural analog of Daumé III (2009).

2 Related Work

2.1 Domain Adaptation

There is a venerable history of research on domain adaptation (Daume III and Marcu, 2006; Daumé III, 2009; Blitzer et al., 2006, 2007; Pan et al., 2011) which is concerned with the shift in data distribution from one domain to another. In the context of NLP, a particularly successful approach is the feature augmentation method of Daumé III (2009) whose key insight is that if we partition the model parameters to those that handle *common patterns* and those that handle *domain-specific patterns*, the model is forced to learn from all domains yet preserve domain-specific knowledge. The method is generalized to the neural paradigm by Kim et al. (2016c) who jointly use a domain-specific LSTM and also a global LSTM shared across all domains. In the context of SLU, Jaech et al. (2016) proposed K domain-specific feedforward layers with a shared word-level LSTM layer across domains; Kim et al. (2016c) instead employed $K + 1$ LSTMs. Hakkani-Tür et al. (2016) proposed to employ a sequence-to-sequence model by introducing a fictitious symbol at the end of an utterance of which tag represents the corresponding domain and intent.

All these methods require one to re-train a model from scratch to make it learn the correlation and invariance between domains. This becomes difficult to scale when there is a new domain coming in at high frequency. We address this problem by proposing a method that only calls K trained domain experts; we do not have to re-train these domain experts. This gives a clear computational advantage over the feature augmentation method.

2.2 Spoken Language Understanding

Recently, there has been much investment on the personal digital assistant (PDA) technology in industry (Sarikaya, 2015; Sarikaya et al., 2016). Apples Siri, Google Now, Microsofts Cortana, and Amazons Alexa are some examples of personal digital assistants. Spoken language understanding (SLU) is an important component of these examples that allows natural communication between the user and the agent (Tur, 2006; El-Kahky et al., 2014). PDAs support a number of scenarios including creating reminders, setting up alarms, note taking, scheduling meetings, finding and consuming entertainment (i.e. movie, music, games), finding places of interest and getting driving directions to them (Kim et al., 2016a).

Naturally, there has been an extensive line of prior studies for domain scaling problems to easily scale to a larger number of domains: pre-training (Kim et al., 2015c), transfer learning (Kim et al., 2015d), constrained decoding with a single model (Kim et al., 2016a), multi-task learning (Jaech et al., 2016), neural domain adaptation (Kim et al., 2016c), domainless adaptation (Kim et al., 2016b), a sequence-to-sequence model (Hakkani-Tür et al., 2016), adversary domain training (Kim et al., 2017) and zero-shot learning(Chen et al., 2016; Ferreira et al., 2015).

There are also a line of prior works on enhancing model capability and features: jointly modeling intent and slot predictions (Jeong and Lee, 2008; Xu and Sarikaya, 2013; Guo et al., 2014; Zhang and Wang, 2016; Liu and Lane, 2016a,b), modeling SLU models with web search click logs (Li et al., 2009; Kim et al., 2015a) and enhancing features, including representations (Anastasakos et al., 2014; Sarikaya et al., 2014; Celikyilmaz et al., 2016, 2010; Kim et al., 2016d) and lexicon (Liu and Sarikaya, 2014; Kim et al., 2015b).

3 Method

We use an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes an input vector x and a state vector h to output a new state vector $h' = \phi(x, h)$. See Hochreiter and Schmidhuber (1997) for a detailed description. At a high level, the individual model consists of builds on several ingredients shown in Figure 1: character and word embedding, a bidirectional LSTM (BiLSTM) at a character layer, a BiLSTM at word level, and feedfoward network at the output.

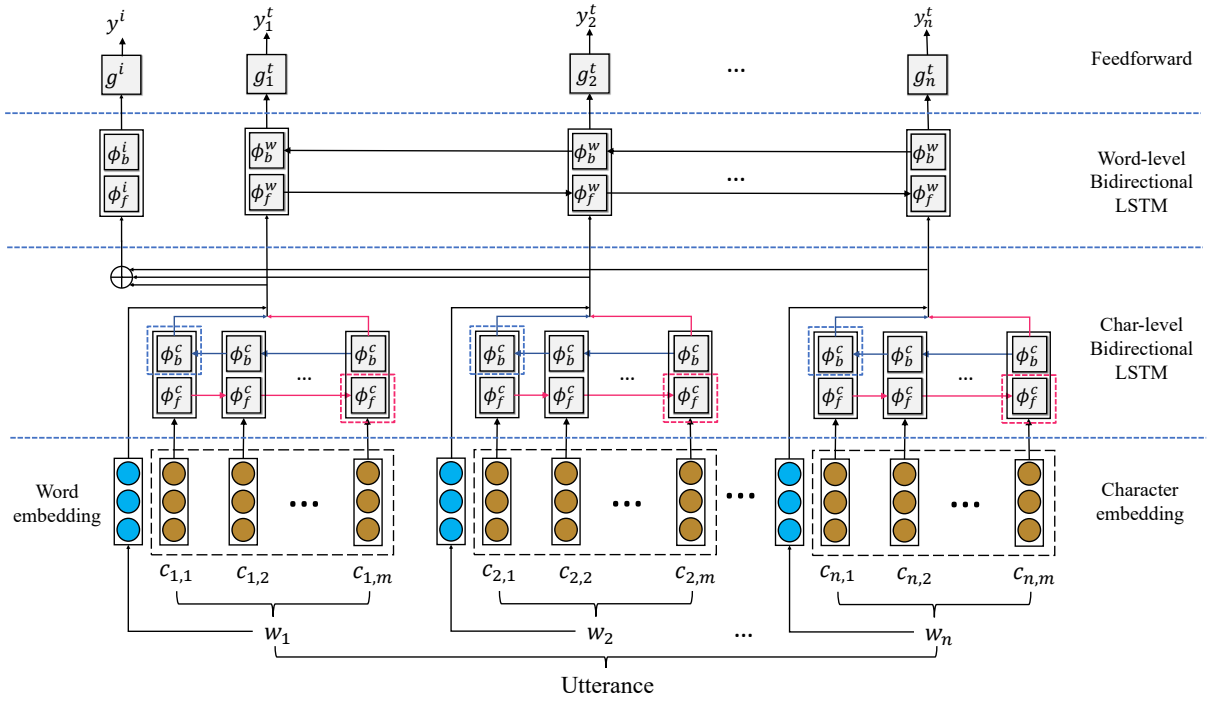

Figure 1: The overall network architecture of the individual model.

3.1 Individual Model Architecture

Let $\mathcal{C}$ denote the set of character types and $\mathcal{W}$ the set of word types. Let $\oplus$ denote the vector concatenation operation. A wildly successful architecture for encoding a sentence $(w_1 \dots w_n) \in \mathcal{W}^n$ is given by bidirectional LSTMs (BiLSTMs) (Schuster and Paliwal, 1997; Graves, 2012). Our model first constructs a network over an utterance closely following Lample et al. (2016). The model parameters Θ associated with this BiLSTM layer are

- Character embedding $e_c \in \mathbb{R}^{25}$ for each $c \in \mathcal{C}$
- Character LSTMs $\phi^{\mathcal{C}}_f, \phi^{\mathcal{C}}_b : \mathbb{R}^{25} \times \mathbb{R}^{25} \to \mathbb{R}^{25}$
- Word embedding $e_w \in \mathbb{R}^{100}$ for each $w \in \mathcal{W}$
- Word LSTMs $\phi^{\mathcal{W}}_f, \phi^{\mathcal{W}}_b : \mathbb{R}^{150} \times \mathbb{R}^{100} \to \mathbb{R}^{100}$

Let $w_1 \dots w_n \in \mathcal{W}$ denote a word sequence where word w_i has character $w_i(j) \in \mathcal{C}$ at position j. First, the model computes a character-sensitive word representation $v_i \in \mathbb{R}^{150}$ as

$$
\begin{aligned}
f^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_f\left(e_{w_i(j)}, f^{\mathcal{C}}_{j-1}\right) && \forall j = 1 \dots |w_i| \\
b^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_b\left(e_{w_i(j)}, b^{\mathcal{C}}_{j+1}\right) && \forall j = |w_i| \dots 1 \\
v_i &= f^{\mathcal{C}}_{|w_i|} \oplus b^{\mathcal{C}}_1 \oplus e_{w_i}
\end{aligned}
$$

for each $i = 1 \dots n$.[2] Next, the model computes

$$
\begin{aligned}
f^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_f\left(v_i, f^{\mathcal{W}}_{i-1}\right) && \forall i = 1 \dots n \\
b^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_b\left(v_i, b^{\mathcal{W}}_{i+1}\right) && \forall i = n \dots 1
\end{aligned}
$$

and induces a character- and context-sensitive word representation $h_i \in \mathbb{R}^{200}$ as

$$h_i = f^{\mathcal{W}}_i \oplus b^{\mathcal{W}}_i \tag{1}$$

for each $i = 1 \dots n$. These vectors can be used to perform intent classification or slot tagging on the utterance.

Intent Classification We can predict the intent of the utterance using $(h_1 \dots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{I}$ denote the set of intent types. We introduce a single-layer feedforward network $g^i : \mathbb{R}^{200} \to \mathbb{R}^{|\mathcal{I}|}$ whose parameters are denoted by Θ^i. We compute a $|\mathcal{I}|$-dimensional vector

$$\mu^i = g^i\left(\sum_{i=1}^{n} h_i\right)$$

and define the conditional probability of the correct intent τ as

$$p(\tau | h_1 \dots h_n) \propto \exp\left(\mu^i_\tau\right) \tag{2}$$

[2] For simplicity, we assume some random initial state vectors such as $f^{\mathcal{C}}_0$ and $b^{\mathcal{C}}_{|w_i|+1}$ when we describe LSTMs.

The intent classification loss is given by the negative log likelihood:

$$L^i\left(\Theta, \Theta^i\right) = -\sum_l \log p\left(\tau^{(l)}|h^{(l)}\right) \quad (3)$$

where l iterates over intent-annotated utterances.

Slot Tagging We predict the semantic slots of the utterance using $(h_1 \ldots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{S}$ denote the set of semantic types and $\mathcal{L}$ the set of corresponding BIO label types [3] that is, $\mathcal{L} = \{\texttt{B}\text{-}e : e \in \mathcal{E}\} \cup \{\texttt{I}\text{-}e : e \in \mathcal{E}\} \cup \{\texttt{O}\}$. We add a transition matrix $T \in \mathbb{R}^{|\mathcal{L}|\times|\mathcal{L}|}$ and a single-layer feedforward network $g^t : \mathbb{R}^{200} \rightarrow \mathbb{R}^{|\mathcal{L}|}$ to the network; denote these additional parameters by Θ^t. The conditional random field (CRF) tagging layer defines a joint distribution over label sequences of $y_1 \ldots y_n \in \mathcal{L}$ of $w_1 \ldots w_n$ as

$$p(y_1 \ldots y_n|h_1 \ldots h_n) \propto \exp\left(\sum_{i=1}^{n} T_{y_{i-1},y_i} \times g^t_{y_i}(h_i)\right) \quad (4)$$

The tagging loss is given by the negative log likelihood:

$$L^t\left(\Theta, \Theta^t\right) = -\sum_l \log p\left(y^{(l)}|h^{(l)}\right) \quad (5)$$

where l iterates over tagged sentences in the data. Alternatively, we can optimize the local loss:

$$L^{t-loc}\left(\Theta, \Theta^t\right) = -\sum_l \sum_i \log p\left(y_i^{(l)}|h_i^{(l)}\right) \quad (6)$$

where $p(y_i|h_i) \propto \exp\left(g^t_{y_i}(h_i)\right)$.

4 Method

4.1 Domain Attention Architecture

Now we assume that for each of the K domains we have an individual model described in Section 3.1. Denote these domain experts by $\Theta^{(1)} \ldots \Theta^{(K)}$. We now describe our model for a new domain $K+1$. Given an utterance $w_1 \ldots w_n$, it uses a BiLSTM layer to induce a feature representation $h_1 \ldots h_n$ as specified in (1). It further invokes K domain experts $\Theta^{(1)} \ldots \Theta^{(K)}$ on this utterance to obtain the feature representations $h_1^{(k)} \ldots h_n^{(k)}$ for

[3] For example, to/O San/B-Source Francisco/I-Source airport/O.

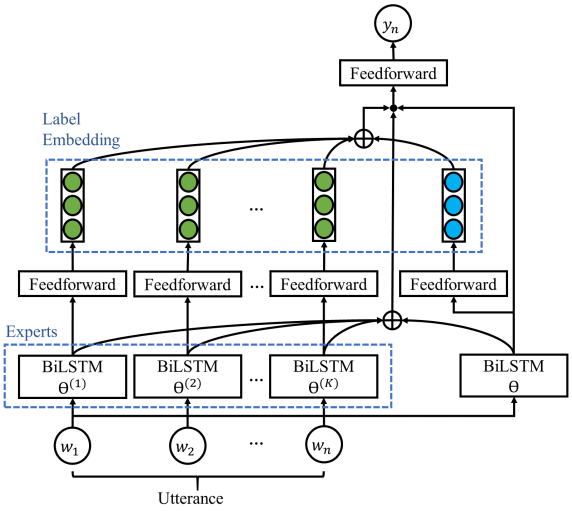

Figure 2: The overall network architecture of the domain attention, which consists of three components: (1) K domain experts + 1 target BiLSTM layer to induce a feature representation, (2) K domain experts + 1 target feedfoward layer to output pre-trained label embedding (3) a final feedforward layer to output an intent or slot. We have two separate attention mechanisms to combine feedback from domain experts.

$k = 1 \ldots K$. For each word w_i, the model computes an attention weight for each domain $k = 1 \ldots K$ domains as

$$q^{\text{dot}}_{i,k} = h_i^\top h^{(k)} \quad (7)$$

in the simplest case. We also experiment with the bilinear function

$$q^{\text{bi}}_{i,k} = h_i^\top B h^{(k)} \quad (8)$$

where B is an additional model parameter, and also the feedforward function

$$q^{\text{feed}}_{i,k} = W \tanh\left(U h_i^\top + V h^{(k)} + b^1\right) + b^2 \quad (9)$$

where U, V, W, b^1, b^2 are additional model parameters. The final attention weights $a_i^{(1)} \ldots a_i^{(1)}$ are obtained by using a softmax layer

$$a_{i,k} = \frac{\exp(q_{i,k})}{\sum_{k=1}^{K} \exp(q_{i,k})} \quad (10)$$

The weighted combination of the experts' feedback is given by

$$h_i^{\text{experts}} = \sum_{k=1}^{K} a_{i,k} h_i^{(k)} \quad (11)$$

and the model makes predictions by using $\bar{h}_1 \ldots \bar{h}_n$ where

$$\bar{h}_i = h_i \oplus h_i^{\text{experts}} \tag{12}$$

These vectors replace the original feature vectors h_i in defining the intent or tagging losses.

4.2 Domain Attention Variants

We also consider two variants of the domain attention architecture in Section 4.1.

Label Embedding In addition to the state vectors $h^{(1)} \ldots h^{(K)}$ produced by K experts, we further incorporate their final (discrete) label predictions using pre-trained label embeddings. We induce embeddings e^y for labels y from all domains using the method of Kim et al. (2015d). At the i-th word, we predict the most likely label $y^{(k)}$ under the k-th expert and compute an attention weight as

$$\bar{q}_{i,k}^{\text{dot}} = h_i^\top e^{y^{(k)}} \tag{13}$$

Then we compute an expectation over the experts' predictions

$$\bar{a}_{i,k} = \frac{\exp(\bar{q}_{i,k})}{\sum_{k=1}^{K} \exp(\bar{q}_{i,k})} \tag{14}$$

$$h_i^{\text{label}} = \sum_{k=1}^{K} \bar{a}_{i,k} e_i^{y^{(k)}} \tag{15}$$

and use it in conjunction with $\bar{h}_i$. Note that this makes the objective a function of discrete decision and thus non-differentiable, but we can still optimize it in a standard way treating it as learning a stochastic policy.

Selective Attention Instead of computing attention over all K experts, we only consider the top $K' \leq K$ that predict the highest label scores. We only compute attention over these K' vectors. We experiment with various values of K'

5 Experiments

In this section, we describe the set of experiments conducted to evaluate the performance of our model. In order to fully assess the contribution of our approach, we also consider several baselines and variants besides our primary expert model.

Domain	$\lvert I\rvert$	$\lvert S\rvert$	Description
EVENTS	10	12	Buy event tickets
FITNESS	10	9	Track health
M-TICKET	8	15	Buy movie tickets
ORDERPIZZA	19	27	Order pizza
REMINDER	19	20	Remind task
TAXI	8	13	Find/book an cab
TV	7	5	Control TV

Table 1: The number of intent types ($|I|$), the number of slot types ($|S|$), and a short description of the test domains.

	Overlapping	
Domain	Intents	Slots
EVENTS	70.00%	75.00%
FITNESS	30.00%	77.78%
M-TICKET	37.50%	100.00%
ORDERPIZZA	47.37%	74.07%
REMINDER	68.42%	85.00%
TAXI	50.00%	100.00%
TV	57.14%	60.00%
AVG	51.49%	81.69%

Table 2: The overlapping percentage of intent types and slot types with experts or source domains.

5.1 Test domains and Tasks

To test the effectiveness of our proposed approach, we apply it to a suite of 7 Microsoft Cortana domains with 2 separate tasks in spoken language understanding: (1) intent classification and (2) slot (label) tagging. The intent classification task is a multi-class classification problem with the goal of determining to which one of the $|I|$ intents a user utterance belongs within a given domain. The slot tagging task is a sequence labeling problem with the goal of identifying entities and chunking of useful information snippets in a user utterance. For example, a user could say "*reserve a table at joeys grill for thursday at seven pm for five people*". Then the goal of the first task would be to classify this utterance as "*make_reservation*" intent given the places domain, and the goal of the second task would be to tag "*joeys grill*" as `restaurant`, "*thursday*" as `date`, "*seven pm*" as `time`, and "*five*" as `number_people`.

The short descriptions on the 7 test domains are shown in Table 1. As the table shows, the test domains have different granularity and diverse semantics. For each personal assistant test domain,

we only used 1000 training utterances to simulate scarcity of newly labeled data. The amount of development and test utterance was 100 and 10k respectively.

The similarities of test domains, represented by overlapping percentage, with experts or source domains are represented in Table 2. The intent overlapping percentage ranges from 30% on FITNESS domain to 70% on EVENTS, which averages out at 51.49%. And the slots for test domains overlaps more with those of source domains ranging from 60% on TV domain to 100% on both M-TICKET and TAXI domains, which averages out at 81.69%.

5.2 Experimental Setup

Category	$\|D\|$	Example
Trans.	4	BUS, FLIGHT
Time	4	ALARM, CALENDAR
Media	5	MOVIE, MUSIC
Action	5	HOMEAUTO, PHONE
Loc.	3	HOTEL, BUSINESS
Info	4	WEATHER, HEALTH
TOTAL	25	

Table 3: Overview of experts or source domains: Domain categories which have been created based on the label embeddings. These categorizations are solely for the purpose of describing domains because of the limited space and they are completely unrelated to the model. The number of sentences in each domain is in the range of 50k to 660k and the number of unique intents and slots are 200 and 500 respectively. In total, we have 25 domain-specific expert models. For the average performance, intent accuracy is 98% and slot F1 score is 96%.

In testing our approach, we consider a domain adaptation (DA) scenario, where a target domain has a limited training data and the source domain has a sufficient amount of labeled data. We further consider a scenario, creating a new virtual domain targeting a specific scenario given a large inventory of intent and slot types and underlying models build for many different applications and scenarios. One typical use case is that of building natural language capability through intent and slot modeling (without actually building a domain classifier) targeting a specific application. Therefore, our experimental settings are rather different from previously considered settings for domain adaptation in two aspects:

- *Multiple source domains:* In most previous works, only a pair of domains (source vs. target) have been considered, although they can be easily generalized to $K > 2$. Here, we experiment with $K = 25$ domains shown in Table 3.

- *Variant output:* In a typical setting for domain adaptation, the label space is invariant across all domains. Here, the label space can be different in different domains, which is a more challenging setting. See Kim et al. (2015d) for details of this setting.

For this DA scenario, we test whether our approach can effectively make a system to quickly generalize to a new domain with limited supervision given K existing domain experts shown in 3 .

In summary, our approach is tested with 7 Microsoft Cortana personal assistant domains across 2 tasks of intent classification and slot tagging. Below shows more detail of our baselines and variants used in our experiments.

Baselines: All models below use same underlying architecture described in Section 3.1

- TARGET: a model trained on a targeted domain without DA techniques.

- UNION: a model trained on the union of a targeted domain and 25 domain experts.

- DA: a neural domain adaptation method of Kim et al. (2016c) which trains domain specific K LSTMs with a generic LSTM on all domain training data.

Domain Experts (DE) variants: All models below are based on attending on an ensemble of 25 domain experts (DE) described in Section 4.1, where a specific set of intent and slots models are trained for each domain. We have two feedback from domain experts: (1) feature representation from LSTM, and (2) label embedding from feedfoward described in Section 4.1 and Section 4.2, respectively.

- DE^B: DE without domain attention mechanism. It uses the unweighted combination of first feedback from experts like bag-of-word model.

- DE^1: DE with domain attention with the weighted combination of the first feedbacks from experts.
- DE^2: DE^1 with additional weighted combination of second feedbacks.
- DE^{S2}: DE^2 with selected attention mechanism, described in Section 4.2.

In our experiments, all the models were implemented using Dynet (Neubig et al., 2017) and were trained using Stochastic Gradient Descent (SGD) with Adam (Kingma and Ba, 2015)—an adaptive learning rate algorithm. We used the initial learning rate of 4×10^{-4} and left all the other hyper parameters as suggested in Kingma and Ba (2015). Each SGD update was computed without a minibatch with Intel MKL (Math Kernel Library)[4]. We used the dropout regularization (Srivastava et al., 2014) with the keep probability of 0.4 at each LSTM layer.

To encode user utterances, we used bidirectional LSTMs (BiLSTMs) at the character level and the word level, along with 25 dimensional character embedding and 100 dimensional word embedding. The dimension of both the input and output of the character LSTMs were 25, and the dimensions of the input and output of the word LSTMs were 150[5] and 100, respectively. The dimension of the input and output of the final feedforward network for intent, and slot were 200 and the number of their corresponding task. Its activation was rectified linear unit (ReLU).

To initialize word embedding, we used word embedding trained from (Lample et al., 2016). In the following sections, we report intent classification results in accuracy percentage and slot results in F1-score. To compute slot F1-score, we used the standard CoNLL evaluation script[6]

5.3 Results

We show our results in the DA setting where we had a sufficient labeled dataset in the 25 source domains shown in Table 3, but only 1000 labeled data in the target domain. The performance of the baselines and our domain experts DE variants are shown in Table 4. The top half of the table shows the results of intent classification and the results of slot tagging is in the bottom half.

The baseline which trained only on the target domain (TARGET) shows a reasonably good performance, yielding on average 87.7% on the intent classification and 83.9% F1-score on the slot tagging. Simply training a single model with aggregated utterance across all domains (UNION) brings the performance down to 77.4% and 75.3%. Using DA approach of Kim et al. (2016c) shows a significant increase in performance in all 7 domains, yielding on average 90.3% intent accuracy and 86.2%.

The DE without domain attention (DE^B) shows similar performance compared to DA. Using DE model with domain attention (DE^1) shows another increase in performance, yielding on average 90.9% intent accuracy and 86.9%. The performance increases again when we use both feature representation and label embedding (DE^2), yielding on average 91.4% and 88.2% and observe nearly 93.6% and 89.1% when using selective attention (DE^{S2}). Note that DE^{S2} selects the appropriate number of experts per layer by evaluation on a development set.

The results show that our expert variant approach (DE^{S2}) achieves a significant performance gain in all 7 test domains, yielding average error reductions of 47.97% for intent classification and 32.30% for slot tagging. The results suggest that our expert approach can quickly generalize to a new domain with limited supervision given K existing domains by having only a handful more data of 1k newly labeled data points. The poor performance of using the union of both source and target domain data might be due to the relatively very small size of the target domain data, overwhelmed by the data in the source domain. For example, a word such as "*home*" can be labeled as place_type under the TAXI domain, but in the source domains can be labeled as either home_screen under the PHONE domain or contact_name under the CALENDAR domain.

5.4 Training Time

The Figure 3 shows the time required for training DE^{S2} and DA of Kim et al. (2016c). The training time for DE^{S2} stays almost constant as the number of source domains increases. However, the training time for DA grows exponentially in the number of source domains. Specifically, when trained

[4]https://software.intel.com/en-us/articles/intelr-mkl-and-c-template-libraries

[5]We concatenated last two outputs from the character LSTM and word embedding, resulting in 150 (25+25+100)

[6]http://www.cnts.ua.ac.be/conll2000/chunking/output.html

Task	Domain	TARGET	UNION	DA	DE^{B}	DE^{1}	DE^{2}	DE^{S2}
Intent	EVENTS	88.3	78.5	89.9	93.1	92.5	92.7	**94.5**
	FITNESS	88.0	77.7	92.0	92.0	91.2	91.8	**94.0**
	M-TICKET	88.2	79.2	91.9	94.4	91.5	92.7	**93.4**
	ORDERPIZZA	85.8	76.6	87.8	89.3	89.4	90.8	**92.8**
	REMINDER	87.2	76.3	91.2	90.0	90.5	90.2	**93.1**
	TAXI	87.3	76.8	89.3	89.9	89.6	89.2	**93.7**
	TV	88.9	76.4	90.3	81.5	91.5	92.0	**94.0**
	AVG	87.7	77.4	90.3	90.5	90.9	91.4	**93.6**
Slot	EVENTS	84.8	76.1	87.1	87.4	88.1	89.4	**90.2**
	FITNESS	84.0	75.6	86.4	86.3	87.0	88.1	**88.9**
	M-TICKET	84.2	75.6	86.4	86.1	86.8	88.4	**89.7**
	ORDERPIZZA	82.3	73.6	84.2	84.4	85.0	86.3	**87.1**
	REMINDER	83.5	75.0	85.9	86.3	87.0	88.3	**89.2**
	TAXI	83.0	74.6	85.6	85.5	86.3	87.5	**88.6**
	TV	85.4	76.7	87.7	87.6	88.3	89.3	**90.1**
	AVG	83.9	75.3	86.2	86.2	86.9	88.2	**89.1**

Table 4: Intent classification accuracy (%) and slot tagging F1-score (%) of our baselines and variants of DE. The numbers in boldface indicate the best performing methods.

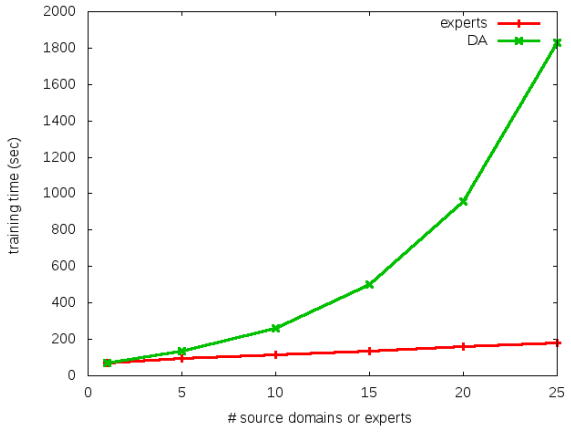

Figure 3: Comparison of training time between our DE^{S2} model and DA model of Kim et al. (2016c) as the number of domains increases. The horizontal axis means the number of domains, the vertical axis is training time per epoch in seconds. Here we use CALENDAR as the target domain, which has 1k training data.

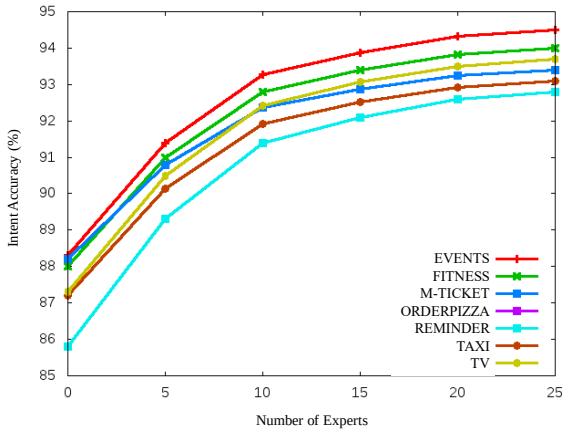

Figure 4: Learning curves in accuracy across all seven test domains as the number of expert domains increases.

with 1 source or expert domain, both took around a minute per epoch on average. When training with full 25 source domains, DE^{S2} took 3 minutes per epoch while DA took 30 minutes per epoch. Since we need to iterate over all 25+1 domains to re-train the global model, the net training time ratio could be over 250.

5.5 Learning Curve

We also measured the performance of our methods as a function of the number of domain experts. For each test domain, we consider all possible sizes of experts ranging from 1 to 25 and we then take the average of the resulting performances obtained from the expert sets of all different sizes. Figure 4 shows the resulting learning curves for each test domain. The overall trend is clear: as the more expert domains are added, the more the test performance improves. With ten or more expert domains added, our method starts to get saturated achiev-

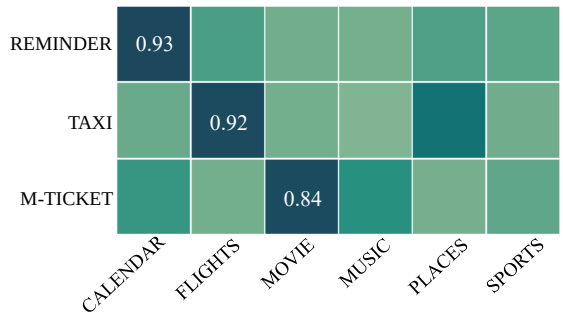

Figure 5: Heatmap visualizing attention weights.

ing more than 90% in accuracy across all seven domains.

5.6 Attention weights

From the heatmap shown in Figure 5, we can see that the attention strength generally agrees with common sense. For example, the M-TICKET and TAXI domain selected MOVIE and PLACES as their top experts, respectively.

5.7 Oracle Expert

Domain	TARGET	DE^2	Top 1
ALARM	70.1	**98.2**	ALARM (.99)
HOTEL	65.2	**96.9**	HOTEL (.99)

Table 5: Intent classification accuracy with an oracle expert in the expert pool.

The results in Table 5 show the intent classification accuracy of DE^2 when we already have the same domain expert in the expert pool. To simulate such a situation, we randomly sampled 1,000, 100, and 100 utterances from each domain as training, development and test data, respectively. In both ALARM and HOTEL domains, the trained models only on the 1,000 training utterances (TARGET) achieved only 70.1%and 65.2% in accuracy, respectively. Whereas, with our method (DE^2) applied, we reached almost the full training performance by selectively paying attention to the oracle expert, yielding 98.2% and 96.9%, respectively. This result again confirms that the behavior of the trained attention network indeed matches the semantic closeness between different domains.

5.8 Selective attention

The results in Table 6 examines how the intent prediction accuracy of DE^{S2} varies with respect to the number of experts in the pool. The rationale behind DE^{S2} is to alleviate the downside of soft attention, namely distributing probability mass over all items even if some are bad items. To deal with such issues, we apply a hard cut-off at top k domains. From the result, a threshold at top 3 or 5 yielded better results than that of either 1 or 25 experts. This matches our common sense that their are only a few of domains that are close enough to be of help to a test domain. Thus it is advisable to find the optimal k value through several rounds of experiments on a development dataset.

Domain	Top 1	Top 3	Top 5	Top 25
EVENTS	98.1	98.8	**99.2**	96.4
TV	81.4	**82.0**	81.7	80.9
AVG	89.8	90.4	**90.5**	88.7

Table 6: Accuracies of DE^{S2} using different number of experts.

6 Conclusion

In this paper, we proposed a solution for scaling domains and experiences potentially to a large number of use cases by reusing existing data labeled for different domains and applications. Our solution is based on attending an ensemble of domain experts. When given a new domain, our model uses a weighted combination of domain experts' feedback along with its own opinion to make prediction on the new domain. In both intent classification and slot tagging tasks, the model significantly outperformed baselines that do not use domain adaptation and also performed better than the full re-training approach. This approach enables creation of new virtual domains through a weighted combination of domain experts' feedback reducing the need to collect and annotate the similar intent and slot types multiple times for different domains. Future work can include an extension of domain experts to take into account dialog history aiming for a holistic framework that can handle contextual interpretation as well.

References

Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pages 3246–3250.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. volume 7, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 120–128.

Asli Celikyilmaz, Ruhi Sarikaya, Minwoo Jeong, and Anoop Deoras. 2016. An empirical investigation of word class-based features for natural language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(6):994–1005.

Asli Celikyilmaz, Silicon Valley, and Dilek Hakkani-Tur. 2010. Convolutional neural network based semantic tagging with entity embeddings. *genre* .

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 6045–6049.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26:101–126.

Ali El-Kahky, Derek Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. IEEE, Proceedings of the ICASSP.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Alex Graves. 2012. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, pages 15–35.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 554–559.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung 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*.

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

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117* .

Minwoo Jeong and Gary Geunbae Lee. 2008. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing* 16(7):1287–1302.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the NAACL*. Association for Computational Linguistics.

Young-Bum Kim, Alexandre Rochette, and Ruhi Sarikaya. 2016a. Natural language model reusability for scaling to different domains. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics* .

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Annual Meeting of the Association for Computational Linguistics*.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 192–198.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016b. Domainless adaptation by constrained decoding on a schema lattice. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)* .

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016c. Frustratingly easy neural domain adaptation. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)* .

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016d. Scalable semi-supervised query classification using matrix sketching. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 8.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. *ACL. Association for Computational Linguistics* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*. .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*.

Bing Liu and Ian Lane. 2016a. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*. pages 685–689.

Bing Liu and Ian Lane. 2016b. Joint online spoken language understanding and language modeling with recurrent neural networks. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Los Angeles.

Xiaohu Liu and Ruhi Sarikaya. 2014. A discriminative model based entity dictionary weighting approach for spoken language understanding. In *Spoken Language Technology Workshop (SLT)*. IEEE, pages 195–199.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2):199–210.

Ruhi Sarikaya. 2015. *The technology powering personal digital assistants*. Keynote at Interspeech, Dresden, Germany.

Ruhi Sarikaya, Asli Celikyilmaz, Anoop Deoras, and Minwoo Jeong. 2014. Shrinkage based features for slot tagging with conditional random fields. In *INTERSPEECH*. pages 268–272.

Ruhi Sarikaya, Paul Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiuahu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *IEEE Workshop on Spoken Language Technology*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Nitish Srivastava, Geoffrey E 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(1):1929–1958.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In *In Proceedings of the ICASSP*. Toulouse, France.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pages 78–83.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. IJCAI.