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. 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
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.
3.1 Individual Model Architecture

Let $C$ 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 $(w_1 \ldots w_n) \in \mathcal{W}^n$ is given by bidirectional LSTMs (BiLSTMs) (Schuster and Paliwal, 1997; Graves, 2012). Our model first constructs a network over an utterance. Our architecture for encoding a sentence is as follows. Let $\Theta$ denote a word sequence where $\forall i = 1 \ldots n$. Next, the model computes

$$f_i^W = \phi_f^W (v_i, f_{i-1}^W) \quad \forall i = 1 \ldots n$$

$$b_i^W = \phi_b^W (v_i, b_{i+1}^W) \quad \forall i = n \ldots 1$$

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

$$h_i = f_i^W \oplus b_i^W \quad (1)$$

for each $i = 1 \ldots n$. 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 $(h_1 \ldots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{I}$ denote the set of intent types. We introduce a single-layer feedforward network $g^i : \mathbb{R}^{200} \rightarrow \mathbb{R}^{\vert \mathcal{I} \vert}$ whose parameters are denoted by $\Theta^i$. We compute a $\vert \mathcal{I} \vert$-dimensional vector

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

and define the conditional probability of the correct intent $\tau$ as

$$p(\tau \vert h_1 \ldots h_n) \propto \exp (\mu^\tau) \quad (2)$$

for simplicity, we assume some random initial state vectors such as $f_0^C$ and $b_1^W$ when we describe LSTMs.

---

Figure 1: The overall network architecture of the individual model.
The intent classification loss is given by the negative log likelihood:

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

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 \( S \) denote the set of semantic types and \( \mathcal{L} \) the set of corresponding BIO label types \(^3\) that is, \( \mathcal{L} = \{B-e : e \in \mathcal{E}\} \cup \{I-e : e \in \mathcal{E}\} \cup \{O\} \). We add a transition matrix \( T \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}|} \) and a single-layer feedforward network \( g^l : \mathbb{R}^{200} \rightarrow \mathbb{R}^{|\mathcal{L}|} \) to the network; denote these additional parameters by \( \Theta^i \). 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

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

The tagging loss is given by the negative log likelihood:

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

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

\[
L^t-loc (\Theta, \Theta^t) = - \sum_l \sum_i \log p \left( y^{(l)}_i | h^{(l)}_i \right) \tag{6}
\]

where \( p(y_i|h_i) \propto \exp \left( g^l_i(h_i) \right) \).

### 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

\[^3\text{For example, to/O San/B-Source Francisco/I-Source airport/O.}\]

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 feedforward 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

\[
q_{i,k}^{\text{det}} = h_i^\top h^{(k)} \tag{7}
\]

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

\[
q_{i,k}^{\text{bile}} = h_i^\top B h^{(k)} \tag{8}
\]

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

\[
q_{i,k}^{\text{feed}} = W \tanh \left( U h_i^\top + V h^{(k)} + b^1 \right) + b^2 \tag{9}
\]

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

\[
a_{i,k} = \frac{\exp(q_{i,k})}{\sum_{k=1}^K \exp(q_{i,k})} \tag{10}
\]

The weighted combination of the experts’ feedback is given by

\[
h_i^{\text{experts}} = \sum_{k=1}^K a_{i,k} h_i^{(k)} \tag{11}
\]
and the model makes predictions by using $\bar{h}_1 \ldots \bar{h}_n$ where

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

(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 $h_1^{(1)} \ldots h_K^{(K)}$ 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

$$\bar{q}_{i,k} = h_i^T e^{y^{(k)}}$$

(13)

Then we compute an expectation over the experts’ predictions

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

(14)

$$\bar{h}_i^{\text{label}} = \sum_{k=1}^{K} \bar{a}_{i,k} e^{y^{(k)}}$$

(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       | $|I|$ | $|S|$ | 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.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Intents</th>
<th>Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENTS</td>
<td>70.00%</td>
<td>75.00%</td>
</tr>
<tr>
<td>FITNESS</td>
<td>30.00%</td>
<td>77.78%</td>
</tr>
<tr>
<td>M-TICKET</td>
<td>37.50%</td>
<td>100.00%</td>
</tr>
<tr>
<td>ORDERPIZZA</td>
<td>47.37%</td>
<td>74.07%</td>
</tr>
<tr>
<td>REMINDER</td>
<td>68.42%</td>
<td>85.00%</td>
</tr>
<tr>
<td>TAXI</td>
<td>50.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>TV</td>
<td>57.14%</td>
<td>60.00%</td>
</tr>
</tbody>
</table>

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 Table 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 feedforward described in Section 4.1 and Section 4.2, respectively.

- **DE$^R$**: DE without domain attention mechanism. It uses the unweighted combination of first feedback from experts like bag-of-word model.
• **DE**: DE with domain attention with the weighted combination of the first feedbacks from experts.

• **DE**: DE with additional weighted combination of second feedbacks.

• **DE**: DE 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 $4 \times 10^{-4}$ 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). 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 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.

### 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) shows similar performance compared to **DA**. Using **DE** model with domain attention (DE) 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), yielding on average 91.4% and 88.2% and observe nearly 93.6% and 89.1% when using selective attention (DE). Note that DE selects the appropriate number of experts per layer by evaluation on a development set.

The results show that our expert variant approach (DE) 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** and DA of Kim et al. (2016c). The training time for **DE** 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...
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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Domain</th>
<th>TARGET</th>
<th>UNION</th>
<th>DA</th>
<th>DE^S1</th>
<th>DE</th>
<th>DE^S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent</td>
<td>EVENTS</td>
<td>88.3</td>
<td>78.5</td>
<td>89.9</td>
<td>93.1</td>
<td>92.5</td>
<td>92.7</td>
</tr>
<tr>
<td></td>
<td>FITNESS</td>
<td>88.0</td>
<td>77.7</td>
<td>92.0</td>
<td>92.0</td>
<td>91.2</td>
<td>91.8</td>
</tr>
<tr>
<td></td>
<td>M-TICKET</td>
<td>88.2</td>
<td>79.2</td>
<td>91.9</td>
<td>94.4</td>
<td>91.5</td>
<td>92.7</td>
</tr>
<tr>
<td></td>
<td>ORDERPIZZA</td>
<td>85.8</td>
<td>76.6</td>
<td>87.8</td>
<td>89.3</td>
<td>89.4</td>
<td>90.8</td>
</tr>
<tr>
<td></td>
<td>REMINDER</td>
<td>87.2</td>
<td>76.3</td>
<td>91.2</td>
<td>90.0</td>
<td>90.5</td>
<td>90.2</td>
</tr>
<tr>
<td></td>
<td>TAXI</td>
<td>87.3</td>
<td>76.8</td>
<td>89.3</td>
<td>89.9</td>
<td>89.6</td>
<td>89.2</td>
</tr>
<tr>
<td></td>
<td>TV</td>
<td>88.9</td>
<td>76.4</td>
<td>90.3</td>
<td>81.5</td>
<td>91.5</td>
<td>92.0</td>
</tr>
<tr>
<td></td>
<td>AVG</td>
<td>87.7</td>
<td>77.4</td>
<td>90.3</td>
<td>90.5</td>
<td>90.9</td>
<td>91.4</td>
</tr>
<tr>
<td>Slot</td>
<td>EVENTS</td>
<td>84.8</td>
<td>76.1</td>
<td>87.1</td>
<td>87.4</td>
<td>88.1</td>
<td>89.4</td>
</tr>
<tr>
<td></td>
<td>FITNESS</td>
<td>84.0</td>
<td>75.6</td>
<td>86.4</td>
<td>86.3</td>
<td>87.0</td>
<td>88.1</td>
</tr>
<tr>
<td></td>
<td>M-TICKET</td>
<td>84.2</td>
<td>75.6</td>
<td>86.4</td>
<td>86.1</td>
<td>86.8</td>
<td>88.4</td>
</tr>
<tr>
<td></td>
<td>ORDERPIZZA</td>
<td>82.3</td>
<td>73.6</td>
<td>84.2</td>
<td>84.4</td>
<td>85.0</td>
<td>86.3</td>
</tr>
<tr>
<td></td>
<td>REMINDER</td>
<td>83.5</td>
<td>75.0</td>
<td>85.9</td>
<td>86.3</td>
<td>87.0</td>
<td>88.3</td>
</tr>
<tr>
<td></td>
<td>TAXI</td>
<td>83.0</td>
<td>74.6</td>
<td>85.6</td>
<td>85.5</td>
<td>86.3</td>
<td>87.5</td>
</tr>
<tr>
<td></td>
<td>TV</td>
<td>85.4</td>
<td>76.7</td>
<td>87.7</td>
<td>87.6</td>
<td>88.3</td>
<td>89.3</td>
</tr>
<tr>
<td></td>
<td>AVG</td>
<td>83.9</td>
<td>75.3</td>
<td>86.2</td>
<td>86.2</td>
<td>86.9</td>
<td>88.2</td>
</tr>
</tbody>
</table>

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.

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-
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

<table>
<thead>
<tr>
<th>Domain</th>
<th>TARGET</th>
<th>DE²</th>
<th>Top 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALARM</td>
<td>70.1</td>
<td><strong>98.2</strong></td>
<td>ALARM (.99)</td>
</tr>
<tr>
<td>HOTEL</td>
<td>65.2</td>
<td><strong>96.9</strong></td>
<td>HOTEL (.99)</td>
</tr>
</tbody>
</table>

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

The results in Table 5 show the intent classification accuracy of DE² 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 (DE²) 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 DE² varies with respect to the number of experts in the pool. The rationale behind DE² 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 there 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.

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


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.


In Spoken Language Technology Workshop (SLT), 2014 IEEE. IEEE, pages 554–559.


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


