Frustratingly Easy Neural Domain Adaptation

Young-Bum Kim
Microsoft
Redmond, WA
ybkim@microsoft.com

Karl Stratos*
Bloomberg L. P.
New York, NY
me@karlstratos.com

Ruhi Sarikaya†
Amazon
Seattle, WA
rsarikay@amazon.com

Abstract

Popular techniques for domain adaptation such as the feature augmentation method of Daumé III (2009) have mostly been considered for sparse binary-valued features, but not for dense real-valued features such as those used in neural networks. In this paper, we describe simple neural extensions of these techniques. First, we propose a natural generalization of the feature augmentation method that uses $K + 1$ LSTMs where one model captures global patterns across all K domains and the remaining K models capture domain-specific information. Second, we propose a novel application of the framework for learning shared structures by Ando and Zhang (2005) to domain adaptation, and also provide a neural extension of their approach. In experiments on slot tagging over 17 domains, our methods give clear performance improvement over Daumé III (2009) applied on feature-rich CRFs.

1 Introduction

There are often multiple types of data that share common characteristics. For example, in this work we are interested in slot tagging of user queries under as many as 17 different domains. Correctly labeling a given query requires the knowledge of the domain that the query is in. A word such as "home" can be labeled as either `contact_name` under the COMMUNICATION domain, `place_type` under the PLACES domain, or `home_screen` under the XBOXENTERTAINMENTSEARCH domain. On the other hand, a function word such as "the" is labeled as `others` across all domains.

Ideally, we would like to train on all sources of data in order to learn about words that are shared across domains, but naively combining these sources for training does not work well since this ignores label inconsistencies across domains. The other extreme is to only use data only from the domain of interest, but it misses out opportunities to learn global patterns shared by multiple domains. The goal of domain adaptation is precisely to address this conundrum: how can we learn from multiple sources of data but retain the integrity of individual domains?

There has been much progress in domain adaptation. A notable example 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 of Daumé III (2009) is usually considered for sparse binary-valued features which underlie conventional NLP systems. With such features, the parameter partitioning can be achieved with trivial data preprocessing: by conjoining feature types with domain indicators and using them alongside the original feature types. But it is not clear how this approach can be extended to dense real-valued feature values, which are used in many recent NLP systems based on neural networks.

In this paper, we describe natural generalizations of such domain adaptation techniques to neural networks. First, we propose a neural extension of the feature augmentation method of Daumé III (2009)

* Work done while at Columbia University.

† Work done while at Microsoft.

in which we achieve the effect of model partitioning by having a global LSTM used across all domains and independent LSTMs used within individual domains, and then combining their outputs in the top layer. Second, we propose using the framework for learning predictive structures by Ando and Zhang (2005) for domain adaptation which has not previously been considered for this task (the original work only considers multi-tasking in the context of semi-supervised learning): we likewise consider a neural extension of this framework.

We perform slot tagging experiments on 17 different personal digital assistant domains that Cortana handles (Tur, 2006; Anastasakos et al., 2014; Kim et al., 2015a; Kim et al., 2015c; Kim et al., 2015b; Kim et al., 2016a; Kim et al., 2016b). Our methods give clear performance improvement over naive baselines such as training K independent models on individual domains or training one model on the union of all domains. Our methods also significantly outperform the feature augmentation method of Daumé III (2009) with standard sparse binary features implemented with conditional random fields (CRFs).

The rest of the paper is organized as follows. In Section 2, we discuss related works. In Section 3, we provide background information on domain adaptation and sequence modeling. First, we review the feature augmentation method of Daumé III (2009) (Section 3.1). Then we review the framework for learning predictive structures by Ando and Zhang (2005) and observe how it can be naturally considered for domain adaptation (Section 3.2). We also give a brief introduction to neural networks for sequence modeling (Section 3.3). In Section 4, we present a neural extension of the feature augmentation method. In Section 5, we present a neural extension of the framework of Ando and Zhang (2005). In Section 6, we give experimental results.

2 Related Works

Domain adaptation and multi-taking with neural networks have been an active research area. We discuss some examples of previous works and how our work differs.

Many past approaches to domain adaptation simply augment the network with a parameter that activates on the current domain. For instance, Alumäe (2013) and Tilk and Alumäe (2014) tackle multi-domain neural language modeling, with both feedforward and recurrent networks, by introducing a domain-indicator parameter. Similarly, Ammar et al. (2016) address multi-lingual dependency parsing with the stack LSTM by introducing a language-indicator parameter that exploits various resources for the given language such as the language identity and WALS typological properties.

Our work is distinguished from these works in that we use a *global* $(K+1)$-th model that captures general patterns across K domains, rather than just K models augmented with domain indicators. This explicit modeling of domain-independent patterns is more in the spirit of the original feature augmentation method of Daumé III (2009). The problem of adapting a general network to particular task has been studied in the context of speech recognition acoustic models (Yao et al., 2012; Mirsamadi and Hansen, 2015).

A task closely related to domain adaptation is multi-tasking: training a single model to perform different tasks (not just different domains) in hope to exploit common properties across tasks. Multi-tasking has also been investigated extensively in the NLP neural network community, notably the "NLP from scratch" work by Collobert et al. (2011) in which various sequence labeling tasks are tackled by a single network with convolution and CRF layers, and its numerous followup studies such as Yang et al. (2016).

3 Background

3.1 Domain Adaptation by Feature Augmentation

Daumé III (2009) propose the following simple approach to domain adaptation. Suppose there are K distinct domains, and let d denote the total number of feature types. The usual setting in NLP is that these features are high-dimensional, sparse, and binary-valued. For example in a CRF, a feature type may ask if the current word is "apple" and the previous word is "company": it takes value 1 if the answer is yes and 0 otherwise. Thus without distinguishing domains, a feature vector takes the form $x \in \{0,1\}^d$ for any domain $k \in \{1 \dots K\}$.

Now, consider learning a naive model $f : \{0,1\}^d \rightarrow \Omega$ over all K domains where Ω denotes an output space. Without domain adaptation, the model f receives a feature vector $x \in \{0,1\}^d$ and simply outputs the value $f(x) \in \Omega$ no matter which domain x corresponds to. The proposal of Daumé III (2009) is that we train a model $f^{aug} : \{0,1\}^{(K+1)d} \rightarrow \Omega$ that instead uses an *augmented* feature vector $x^{aug} \in \{0,1\}^{(K+1)d}$ defined as

$$x^{aug} = (x, 0, \ldots, x^k, \ldots, 0)$$

where $x^k \in \{0,1\}^d$ is a feature representation of x under the k-th domain. In other words, we expand the feature space by $K+1$ times by duplicating the original feature x for K times: then given an input from the k-th domain, we only use the original x and the corresponding domain representation x^k.

Note that this essentially partitions the model parameters to those that handle the representation x used across all domains and those that handle x^k for individual domains $k \in \{1 \ldots K\}$. In the sparse binary feature regime, this can be achieved by conjoining the original feature types with a domain indicator. For example, the i-th feature value of x^k can look like

$$x_i^k = \begin{cases} 1 & \text{if the current word is "apple"} \\ & \text{AND the previous word is "company"} \\ & \text{AND the domain is } k \\ 0 & \text{otherwise} \end{cases}$$

This can be also thought of as data preprocessing in which we duplicate the data in each domain and mark one copy with the domain identity.

3.2 Shared Structure Learning Framework

Ando and Zhang (2005) propose learning a shared structure across multiple related classification tasks. Specifically, they consider K binary classification tasks each of which has its own linear classifier $f_k : \mathbb{R}^d \rightarrow \mathbb{R}$ mapping a d-dimensional feature vector $x \in \mathbb{R}^d$ to a classification score

$$f_k(x) := (u_k + \Theta v_k)^\top x = u_k^\top x + v_k^\top \Theta^\top x$$

Here, $u_k \in \mathbb{R}^d$ and $v_k \in \mathbb{R}^m$ are task-specific parameters but $\Theta \in \mathbb{R}^{d \times m}$ is a global parameter shared by all classifiers $f_1 \ldots f_K$. In particular, if Θ is zero then each classifier is an independent linear function $u_k^\top x$. The predicted label is the sign of the classification score $f_k(x)$.

The parameter sharing makes the estimation problem challenging, but Ando and Zhang (2005) develop an effective alternating loss minimization algorithm using a variational property of singular value decomposition (SVD). The original work applied this framework to semi-supervised learning and achieved strong empirical results. But note that it can be viewed as domain adaptation where the K classification tasks are different "domains" and we aim to learn a global parameter Θ shared across the domains.

3.3 Neural Networks for Sequence Modeling

Recently, there has been much success in tackling sequence modeling problems using neural networks. By far the most popular network for sequence modeling is long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), which is a special case of more general recurrent neural network (RNN) architecture. An RNN extends a traditional feedforward network by recursively receiving inputs generated by the model itself. This essentially defines a feedforward network in which the same set of parameters are used multiple times and is well-suited for sequential data. But a simple RNN suffers from the vanishing gradient problem and does not model long-term dependency very well (Pascanu et al., 2013). An LSTM addresses this issue by introducing a memory cell in RNN architecture that can control how much past information to retain or to forget. This modification has been shown to be crucial in practice. Many recent works in NLP have achieved state-of-the-art results using variants of LSTM, for example in dependency parsing (Dyer et al., 2015) and machine translation (Bahdanau et al., 2014).

4 A Neural Extension of the Feature Augmentation Method

We now describe a neural extension of the feature augmentation method for domain adaptation (Daumé III, 2009). Our model consists of $K + 1$ LSTMs: one LSTM θ is used on all domains, and the remaining K LSTMs $\theta^1 \dots \theta^K$ are used only for the corresponding domains. More specifically, we predict one of L labels at the t-th time step in a given user query in domain $k \in \{1 \dots K\}$ as follows. The common LSTM θ produces an output vector $h_t \in \mathbb{R}^d$ and the domain-specific LSTM θ^k produces an output vector $h_t^k \in \mathbb{R}^d$. The output dimension d of these LSTMs is empirically chosen. The model has parameters $W \in \mathbb{R}^{L \times d}$ and $W^k \in \mathbb{R}^{L \times d}$ at the top layer and combines the LSTM outputs as $z_t^k \in \mathbb{R}^L$ where

$$z_t^k = [W\ W^k] \begin{bmatrix} h_t \\ h_t^k \end{bmatrix} = W h_t + W^k h_t^k \tag{1}$$

which is a direct analogue of the feature augmentation method with sparse binary-valued features:

$$x^{aug} = (x, 0, \dots, x^k, \dots, 0)$$

This combined representation (1) is put through a softmax function to produce a distribution over L labels. Figure 1 (a) provides an illustration of the proposed architecture The model can be trained by maximizing the log likelihood of correct predictions in the training data:

$$J(W, W^1 \dots W^k, \theta, \theta^1 \dots \theta^k) = \sum_t \log \left(\frac{\exp [z_t]_{ans(t)}}{\sum_{l=1}^{L} \exp [z_t]_l} \right) \tag{2}$$

where $ans(t) \in \{1 \dots L\}$ is the ground-truth label for the t-th training example. Note that we do not consider the sentence-level log likelihood (which requires an additional CRF layer for global normalization) for simplicity.

5 A Neural Extension of the Shared Structure Learning Framework

We describe a neural extension of the shared structure learning framework of Ando and Zhang (2005). Recall that Ando and Zhang (2005) consider K binary classifiers $f_k : \mathbb{R}^d \to \mathbb{R}$ with a global parameter $\Theta \in \mathbb{R}^{d \times m}$:

$$f_k(x) = u_k^\top x + v_k^\top \Theta^\top x \tag{3}$$

The model can be seen as combining the given input x (the first term) and a transformed input $\Theta^\top x$ (the second term) where the transformation is learned across K tasks.

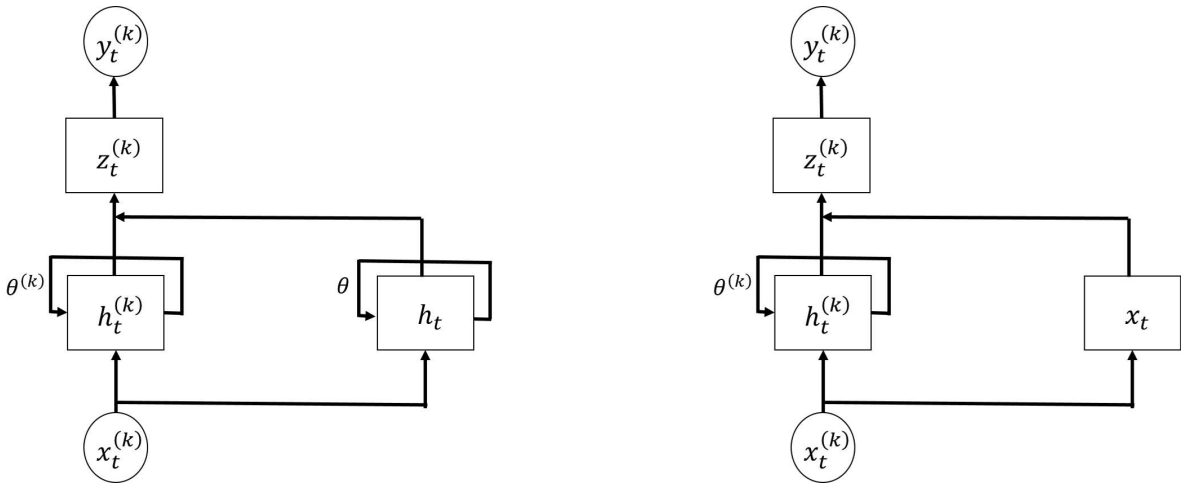
We extend this framework with the following model. The model consists of K LSTMs $\theta^1 \dots \theta^K$ corresponding to K domains. At the t-th time step in a given user query in domain $k \in \{1 \dots K\}$, we compute a direct analogue of (3) by adding the output vector of the k-th LSTM θ^k on the input vector (word embedding) x_t to the original input vector to create

$$z_t^k = W x_t + W^k \theta^k(x_t)$$

where $\theta^k(x_t) \in \mathbb{R}^d$ denotes the output vector of the k-th LSTM on input x_t.

Figure 1 (b) provides an illustration of the proposed architecture This combined representation is put through a softmax function to produce a distribution over L labels. The model can be trained by maximizing the log likelihood of correct predictions in the training data similarly in (2).

Note that the proposed extensions of Daumé III (2009) and Ando and Zhang (2005) only differ in how the domain-independent information is derived. In particular, the extension of Ando and Zhang (2005) can be seen as a weaker version of the extension of Daumé III (2009) since the domain-independent information is used as is (i.e., x_t) in the former while first transformed by a domain specific LSTM in the latter (i.e., h_t).



(a) Feature augmentation method

(b) Shared structure learning framework

Figure 1: Illustration of the proposed neural extensions of Daumé III (2009) and Ando and Zhang (2005).

6 Experiments

In this section, we turn to experimental findings to provide empirical support for our proposed methods. First, we note that our experimental settings are rather different from previously considered settings for domain adaptation in many aspects:

- *Sufficient target data:* In a typical setting for domain adaptation, one generally assumes that the source domain has a sufficient amount of labeled data but the target domain has an insufficient amount of labeled data. However, we have sufficient amounts of labeled data for all domains: our goal is to effectively utilize all of them.
- *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.
- *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 $K = 17$ domains.

To test the effectiveness of our approach, we apply it to the slot sequence tagging task in a suite of 17 personal assistant domains. The goal is to find the correct semantic tagging of the words in a given user utterance. For example, a user could say *"reserve a table at joeys grill for thursday at seven pm for five people"*; then the model needs to tag *"joeys grill"* with `restaurants`, *"thursday"* with `date`, *"seven pm"* with `time`, and "five" with `numberpeople`. The data statistics and the short descriptions of the domains are shown in Table 1. As the table indicates, the domains have very different attributes. We have 131 unique labels across these domains (a different subset of these labels is used in each domain). The total numbers of training, development and test queries across domains are 2640K, 129K and 64K, respectively.

6.1 Setting

In all our experiments, we train our models using stochastic gradient descent (SGD) with Adam (Kingma and Ba, 2015)—an adaptive learning rate algorithm. We use the initial learning rate of 2×10^{-4} and leave all the other hyperparameters as suggested in Kingma and Ba (2015). Each SGD update is computed using a minibatch of size 128. We use the dropout regularization (Srivastava et al., 2014) with the keep probability of 0.5. The dimension of the hidden layer is $d = 100$. For simplicity, we use simple forward-directional LSTMs rather than bi-direction LSTMs, as we observed that they yielded similar results on

	# of labels	#train	#test	#dev	#vocab	Description
Alarm	8	157K	13K	7K	3504	Set alarms
Calendar	20	130K	12K	7K	11077	Set appointments and meeting in calendar
Comm.	21	750K	60K	26K	69414	Make a call and sent messages
Entertain.	15	150K	8K	3.7K	17239	Find songs and movies
Events	6	7K	1.5K	1K	691	Find events and book a ticket
Hotel	17	4.5K	2.6K	1.8K	8022	Book hotel
Mediactrl	10	126K	11K	6K	12589	Set up a music player
Mvtickets	7	4K	1.2K	1K	2235	Find movie theater and book a ticket
Mystuff	18	3.9K	1.8K	1K	8711	Find and open a document
Note	3	6.7K	1.2K	0.6K	4269	Edit and create note
Ondevice	6	228K	2.5K	1.5K	5332	Set up a phone
Orderfood	11	7K	1.4k	0.6K	3686	Order food using app
Places	32	479K	4.5K	2.5K	51424	Find location and direction
Reminder	16	307K	2.7K	1.2K	27510	Remind appointment and to-do list
Reservations	12	2.4K	1K	0.6K	1269	Make a restaurant reservations
Taxi	10	6.3K	2K	1.2K	4391	Find and book a cab
Weather	9	281K	2.5K	1.5K	11878	Ask weather
Overall	131	2640K	129K	64K	139310	

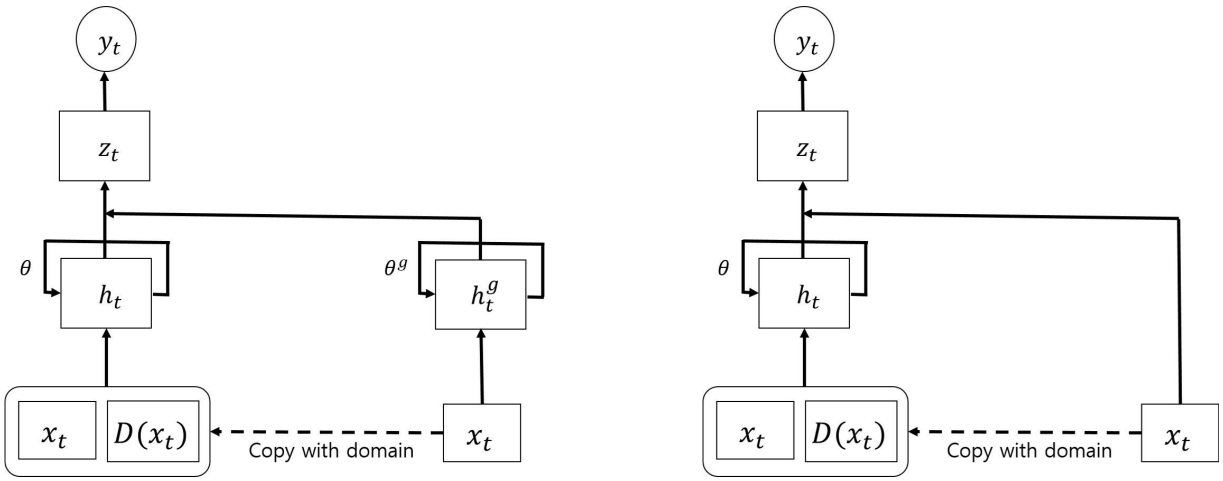
Table 1: Data sets used in the experiments. For each domain, the number of unique labels, the number of quires in the training, development, and test sets, input vocabulary size of the training set, and short description about domain.

the development set in preliminary experiments. To initialize word embedding, we used word embedding trained on 6 billion tokens (6B-200d) from Wikipedia 2014 plus Gigaword 5 (Pennington et al., 2014). These configurations were selected by observing the models performance on held-out development set.

We compare the following methods for the slot tagging tasks:

- *NoAdapt*: train a feature-rich CRF only on target training data.
- *Union*: train a feature-rich CRF on the union of source and target training data.
- *Daume*: train a feature-rich CRF with the discrete feature duplication method of Daumé III (2009).
- *1D&E*: train a domain specific LSTM with a generic embedding on all domain training data, shown on the right in Figure 2.
- *1D&L*: train a domain specific LSTM with a generic LSTM on all domain training data, shown on the left in Figure 2.
- *KD&E*: train domain specific K LSTMs with a generic embedding on all domain training data, shown on the right in Figure 1.
- *KD&L*: train domain specific K LSTMs with a generic LSTM on all domain training data, shown on the left in Figure 1.

The methods *1D&E* and *1D&L* are slight variants of the main proposed neural networks *KD&E* and *KD&L* in Figure 1. These only use a single LSTM for K domains in addition to a common LSTM: it is mostly presented for comparison purposes. Note that we have a single model to tag all domains (except for *NoAdapt* with which we have K models).



(a) 1 domain specific LSTM + generic LSTM

(b) 1 domain specific LSTM + generic embedding

Figure 2: Illustration of slight variants of the proposed neural extensions of Daumé III (2009) and Ando and Zhang (2005) in which we have 2 instead of $K + 1$ LSTMs: one is used for individual domains, one is shared across all domains.

Domain	Noadapt	Union	Daumé III (2009)
Alarm	95.89	89.24	**97.36**
Calendar	91.03	82.09	**94.4**
Comm.	94.94	81.79	**96.6**
Entertain.	**93.83**	89.91	92.61
Events	87.84	58.74	**89.34**
Hotel	91.75	84.54	**92.76**
Mediactrl	90.39	77.76	**92.3**
Mvtickets	92.75	79.21	**94.43**
Mystuff	**90.92**	79.87	89.95
Note	87.9	63.21	**90.86**
Ondevice	93.59	89.48	**96.44**
Orderfood	93.52	85.71	**95.78**
Places	92.75	87.46	**94.13**
Reminder	91.81	84.25	**94.02**
Reservations	92.68	74.45	**94.43**
Taxi	90.27	79.22	**94.75**
Weather	97.27	91.27	**98.44**
Average	92.30	81.07	**94.04**

Table 2: F1 scores across seventeen personal assistant domains for *Noadapt*, *Union* and Daumé III (2009) in a setting with sparse binary-valued features.

6.2 Results

For non-neural models, we use conditional random fields (CRFs) (Lafferty et al., 2001) with a rich set of binary-valued features such as lexical features, gazetteers, Brown clusters (Brown et al., 1992) and context words. For parameter estimation, we used L-BFGS (Liu and Nocedal, 1989) with 100 as the maximum iteration count and 1.0 for the L2 regularization parameter.

Their results are shown in Table 2. A few observations: the model without domain adaptation (*Noadapt*) is already very competitive because we have sufficient training data. However, simply training a single model with aggregated queries across all domains significantly degrades performance (*Union*). This is because in many cases the same query is labeled differently depending on the domain and the

context. This is also because the amounts of training data are widely different across different domains. For example, slots (e.g. `app name`) in other domains are overwhelmed by slots in PLACES domain such as `place name` since our PLACES domain is the largest in terms of number of slot types and 2nd largest in terms of dataset size. Finally, the feature augmentation method of Daumé III (2009) dramatically improves performance across all domains, achieving the average F1 score of 94.04.

Daumé III (2009)	Domain	NoAdapt	1D&E	1D&L	KD&E	KD&L
97.36	Alarm	94.64	97.23	97.26	97.46	**97.5**
94.4	Calendar	93.49	95.03	95.06	95.14	**96.67**
96.6	Comm.	95.36	96.82	96.83	96.96	**97.08**
92.61	Entertain.	94.73	94.34	94.28	**94.61**	94.59
89.34	Events	87.12	90.33	**92.82**	90.92	92.65
92.76	Hotel	92.28	94.78	96.11	96.4	**97.71**
92.3	Mediactrl	91.91	90.85	92.23	92.22	**93.49**
94.43	Mvtickets	92.75	94.98	96.76	96.41	**97.55**
89.95	Mystuff	89.34	88.42	88.23	88.68	**90.71**
90.86	Note	89.89	91.77	94.29	91.38	**94.61**
96.44	Ondevice	96.65	97.29	97.31	97.52	**97.64**
95.78	Orderfood	93.28	95.23	**96.65**	96.45	96.26
94.13	Places	94.47	95.85	96.52	96.52	**96.64**
94.02	Reminder	91.53	92.96	93.14	93.27	**93.37**
94.43	Reservations	92.82	94.43	95.45	95.76	**96.09**
94.75	Taxi	88.32	91.7	91.94	91.51	**92.07**
98.44	Weather	98.07	98.69	98.46	98.45	**98.8**
94.04	Average	92.74	94.16	94.90	94.69	**95.50**

Table 3: F1 scores for Daumé III (2009) and LSTM model variants across seventeen personal assistant domains.

The results with our proposed LSTM domain adaptation models are also shown in Table 3. For easy comparison, the results with Daumé III (2009) applied on feature-rich CRFs are duplicated at the leftmost column.

The *NoAdapt* yields 92.74% average F1-measure which is higher than the performance of CRFs without the rich hand-crafted features. We consistently and significantly improve performance by using adaptation techniques. First, the *1D&E* improves F1 performance to 94.16%. Using not generic embedding, but generic LSTM *(1D&L)* boosts the performance up to 94.9%. Also, having K domain specific LSTMs with generic embeddings (*KD&E*) which is very close to Ando and Zhang (2005) achieves performance gain from *1D&E*, but fail to provide any improvement from 1D&L. Finally, KD&L, which is directly inspired by Daumé III (2009), achieves the best performance of 95.5%, indicating that having K different domain specific LSTMs and a generic LSTM yields significant gains on most of the domains.

7 Conclusion

In this work, we described novel neural approaches to domain adaptation. Adding to the rich history of works in domain adaptation and multi-tasking with neural networks, we proposed a neural analogue of the well-established feature augmentation method of Daumé III (2009) and the shared structure learning framework of Ando and Zhang (2005). Our extensions are natural and simple. In experiments on slot tagging over 17 domains, we demonstrated the effectiveness of our methods by delivering clear performance gains over both naive and strong baselines.

Acknowledgements

We thank Minjoon Seo and anonymous reviewers for their constructive feedback.

References

Tanel Alumäe. 2013. Multi-domain neural network language model. In *INTERSPEECH*, pages 2182–2186. Citeseer.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *CoRR, abs/1602.01595*.

Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *ICASSP*, pages 3246–3250. IEEE.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

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

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

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

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, 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, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. *ACL. Association for Computational Linguistics*.

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

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016b. Scalable semi-supervised query classification using matrix sketching. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

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

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Seyedmahdad Mirsamadi and John HL Hansen. 2015. A study on deep neural network acoustic model adaptation for robust far-field speech recognition. In *Proceedings of Interspeech*, pages 2430–2434.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

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.

Ottokar Tilk and Tanel Alumäe. 2014. Multi-domain recurrent neural network language model for medical speech recognition. In *Baltic HLT*, pages 149–152.

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

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. 2012. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 366–369. IEEE.