Domainless Adaptation by Constrained Decoding on a Schema Lattice

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

In many applications such as personal digital assistants, there is a constant need for new domains to increase the system's coverage of user queries. A conventional approach is to learn a separate model every time a new domain is introduced. This approach is slow, inefficient, and a bottleneck for scaling to a large number of domains. In this paper, we introduce a framework that allows us to have a single model that can handle all domains: including unknown domains that may be created in the future as long as they are covered in the master schema. The key idea is to remove the need for distinguishing domains by explicitly predicting the schema of queries. Given permitted schema of a query, we perform constrained decoding on a lattice of slot sequences allowed under the schema. The proposed model achieves competitive and often superior performance over the conventional model trained separately per domain.

1 Introduction

Recently, there has been much investment on the personal digital assistant (PDA) technology in industry (Sarikaya, 2015). Apple's Siri, Google Now, Microsoft's Cortana, and Amazon's Alexa are some examples of personal digital assistants. Spoken language understanding is an important component of these examples that allows natural communication between the user and the agent (Tur, 2006; El-Kahky et al., 2014; Kim et al., 2015a; Kim et al., 2016b). 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. The number of domains supported by these systems constantly increases, and whether there is a method that allows us to easily scale to a larger number of domains is an unsolved problem (Kim et al., 2015d; Kim et al., 2016a).

The main reason behind the need for additional domains is that we require a new set of schema (i.e., query topics), composed of intents, and slots for processing user queries in a new category. For example, a query in the TAXI domain is processed according to domain-specific schema that is different from those in the HOTEL domain. This in turn requires collecting and annotating new data, which is time consuming and expensive. Once the data is prepared, we also need to build a new system (i.e., models) for this specific domain. In particular, slot modeling is one of the most demanding components of the system in terms of costs in annotation and computation.

In this paper, we introduce a new approach that entirely removes the costs traditionally associated with enlarging the set of supported domains while significantly improving performance. This approach uses a single model to handle all domains: including unknown domains that may be created in the future using a combination of intents and slots in the master schema. The key idea is to remove the need for distinguishing domains by explicitly predicting topics/schema of queries. Thus we obviate the need and directly predict the schema from queries by multi-label classification (either with an RNN or with binary

* Work done while at Columbia University.

† Work done while at Microsoft.

logistic regressions). Given permitted schema of a query, we perform constrained decoding on a lattice of slot sequences allowed under the schema.

In experiments on slot tagging 17 Cortana personal digital assistant domains, we observe that our single model outperforms each of the 17 models trained separately on different domains. This is because our model is able to leverage the data in all domains by reusing the same slots. It can be viewed as a form of domain adaptation, although "domainless adaptation" may be a more accurate description since we remove the need for distinguishing domains!

2 Background

2.1 Domain Adaptation

The goal of domain adaptation is to jointly leverage multiple sources of data (i.e., domains) in attempt to improve performance on any particular domain. There is a rich body of work in domain adaptation for natural language processing. A notable example is the feature augmentation method of Daumé III (2009), who propose partitioning the model parameters to those that handle common patterns and those that handle domain-specific patterns. This way, the model is forced to learn from all domains yet preserve domain-specific knowledge.

Another domain adaptation technique used in natural language processing utilizes unlabeled data in source and target distributions to find shared patterns (Blitzer et al., 2006; Blitzer et al., 2011). This is achieved by finding a shared subspace between the two domains through singular value decomposition (SVD). Unlike the feature augmentation method of Daumé III (2009), however, it does not leverage labeled data in the target domain.

This work is rather different from the conventional works in domain adaptation in that we remove the need to distinguish domains: we have a single model that can handle arbitrary (including unknown) domains. Among other benefits, this approach removes the error propagation due to domain misclassification. Most domain adaptation methods require that we know the data's domain at test time (e.g., the feature augmentation method). But in practice, the domain needs to be predicted separately by a domain classifier whose error propagates to later stages of processing such as intent detection and slot tagging.

2.2 Constrained Decoding

In a later section, we perform constrained decoding on a lattice of possible label sequences. This technique was originally proposed for transfer learning by Täckström et al. (2013). Suppose we have sequences that are only partially labeled. That is, for each token x_j in sequence $x_1 \dots x_n$ we have a set of allowed label types $\mathcal{Y}(x_j)$. Täckström et al. (2013) define a constrained lattice $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}_1) \times \dots \times \mathcal{Y}(x_n, \tilde{y}_n)$ where at each position j a set of allowed label types is given as:

$$\mathcal{Y}(x_j, \tilde{y}_j) = \begin{cases} \{\tilde{y}_j\} & \text{if } \tilde{y}_j \text{ is given} \\ \mathcal{Y}(x_j) & \text{otherwise} \end{cases}$$

We compute the most likely sequence in the lattice for a given observation sequence x under model θ as:

$$y^* = \underset{y \in \mathcal{Y}(x, \tilde{y})}{\arg\max}\, p_\theta(y|x)$$

3 Methods

In this section, we describe our domainless prediction framework. It consists of two stages:

1. Given a query, we perform *multi-label classification* to predict a set of allowed schema for the query.
2. Given the predicted schema, we perform *constrained decoding* on the lattice of valid slot sequences.

Since this framework does not involve domain prediction at all, given a query in an unknown domain we can still use the same model to infer its slot sequence, as long as the new domain is composed of existing slots and intents. In cases where the new domain needs an a new intent or slot, the underlying generic models have to updated with the updated schema.

3.1 Schema Prediction

The first stage produces a set of label types that serve as constraints in the second stage. To this end, we use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) (Figure 1). The LSTM processes the given query to produce a fixed-size vector where the input at each time step is the word embedding corresponding to the word used at the time. We initialize these word embeddings with GloVe vectors (Pennington et al., 2014). Then the network maps the query vector to a distribution over schema types.

In more detail, we first map each word of the utterance into d-dimensional vector space using an embedding matrix of size V by d (which is trained along with other parameters in the network), where V is the size of the vocabulary. Then we map the sequence of the word vectors, $\{x_1, \dots x_T\}$, to LSTM outputs $\{h_1, \dots h_T\}$ where we take the last output to be a d-dimensional summary vector of the utterance $s = h_T$. We then use parameters $W \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^k$ where k is the number of slot types and compute

$$\hat{y} = \text{softmax}\left(Ws + b\right)$$

Thus $\hat{y}_i \in [0, 1]$ is the probability of slot i for the given utterance. To train the model, we minimize the sum of squared errors $||\hat{y} - y||$ (we could certainly use other metrics such as the KL divergence, but we did not pursue this direction).

At test time, we need to perform multi-label classification with the predicted probabilities of slot types $\hat{y} \in [0, 1]^k$. We achieve this by thresholding. But rather than using 0.5 as the threshold, we use the *minimum* probability of a ground-truth schema type from the training data. This results in predictions that are very high in recall at the expense of some precision. This trade-off is suitable in our setting, since these labels are only constraints in the second stage: while missing true labels causes the second stage to fail, over-predicting labels does not.

Since the minimum probabilities of ground-truth schema types are observed in the training data, we can train an separate model (SVM) to predict the threshold value for unseen inputs. In summary, given a test utterance, we first use the LSTM network to compute a distribution of slot types $\hat{y}$, next use the trained regressor to predict a suitable value of threshold, and take labels that have probabilities higher than the threshold. Figure 1 illustrates the process.

3.2 Constrained Decoding

In sequence learning, given a sample query $x_1 \dots x_n$, the decoding problem is to find the most likely tag sequence among all the possible sequences, $y_1 \dots y_n$:

$$f(x_1 \dots x_n) = \arg\max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

Here, given constraints $\tilde{y}$ from the first stage, we can simply define a constrained lattice *lattice* $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}) \times \dots \times \mathcal{Y}(x_n, \tilde{y})$ by pruning all tags not licensed by the constraints, as shown in Figure 2. Then, to find the most likely tag sequence which does not violate the given constrained lattice, we perform the decoding in the constrained lattice:

$$f(x_1 \dots x_n, \tilde{y}) = \arg\max_{\mathcal{Y}(x, \tilde{y})} p(x_1 \dots x_n, y_1 \dots y_n)$$

In experiments, we train a single sequence labeling model (CRF) on all domains, but at test time apply this constrained decoding with slot types predicted by the model in Section 3.1.

3.3 Relation to the Union Method

One of the most naive baselines in domain adaptation is to simply train a single model on the union of all data in different domains; at test time, the model predicts labels for any input regardless of which domain it comes from. Since our approach uses all data as well, it can be seen as a variation on this naive method.

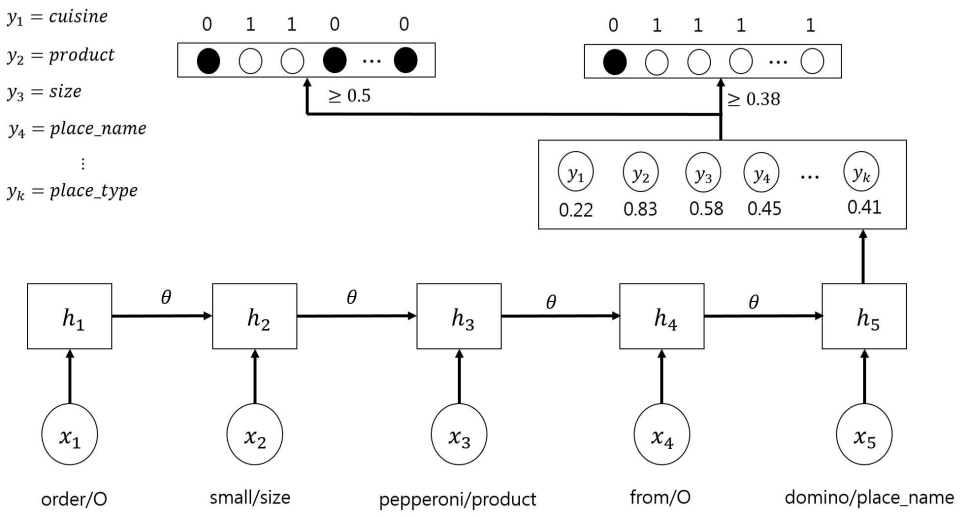

Figure 1: Illustration of schema prediction. In given a query, "order small pepperoni from domino", the word "small", "pepperoni" and "domino" is tagged as *size*, *product* and *place_name*, respectively. Therefore, LSTM multi-label classification model should predict a set of slots a query would be tagged with. When we fix a threshold for final result to 0.5, we can get two permitted labels, *product* and *size*. Whereas, we select different threshold corresponding to each query such as 0.38 in this example, we can obtain four permitted labels, *product*, *size*, *place_name* and *place_type*.

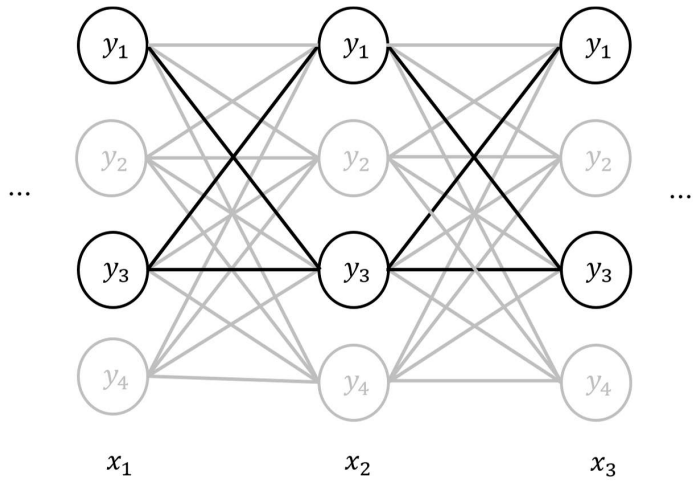

Figure 2: Constrained Lattice: Disabling nodes and transition while decoding the lattice to honor given constraints of domain schema.

The naive method also implicitly makes a decision on the domain of a query when the model predicts domain-specific labels. But it is well-known that this approach typically, unlike ours, yields poor performance. We conjecture that the reason for poor performance is the following. In the union method, the model must perform the *segmentation* as well as labeling of slots, which involves predicting labels in the BIO format (B: begin, I: inside, O: outside) (Ramshaw and Marcus, 1999). In comparison, our method only predicts possible labels and delegates inference to constrained decoding. Thus it can potentially

make more efficient use of labeled data.

4 Experiments

In this section, we turn to experimental findings to provide empirical support for our proposed methods.

4.1 Setting

	# of labels	# of shared label	#train	#test	#dev	#vocab	Description
Alarm	8	6	178K	14K	13K	3628	Set alarms
Calendar	20	16	220K	17K	15.6K	11574	Set events in calendar
Comm.	21	13	780K	72K	29K	69817	Make a call and sent text
Entertain.	15	5	173K	11K	9.3K	17521	Search movies and music
Events	6	4	12K	6K	5K	835	Purchase tickets to events
Hotel	17	9	7.3K	5.7K	4.9K	8172	Book hotel
Mediactrl	10	8	132K	19K	16K	12802	Set up a music player
Mvtickets	7	7	13K	8.2K	7.8K	2298	Buy movie tickets and find showtime
Mystuff	18	12	6.4K	3.6K	3.2K	8824	Find files and attachments
Note	3	1	7.8K	2.9K	2.5K	4756	Find, edit and create a note
Ondevice	6	5	259K	9.4K	6.4K	5386	Control the device
Orderfood	11	10	20K	2.7k	2.6K	3745	Order food using app
Places	32	19	488K	9.4K	8.7K	51611	Find location and direction
Reminder	16	12	338K	21.8K	18K	27823	Find, edit and create reminders
Reservations	12	11	17K	4K	3K	2920	Make a restaurant reservations
Taxi	10	10	10.7K	4.9K	3.1K	451	Find and book a cab
Weather	9	2	302K	5.5K	5.2K	12344	Ask weather
Overall	131	62	2964K	217K	153K	245K	

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

To test the effectiveness of the proposed approach, we apply it to a suite of 17 Cortana personal assistant domains for slot (label) tagging tasks, where the goal is to find the correct semantic tags 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 goal is to tag "joeys grill" with `restaurant`, "thursday" with `date`, "seven pm" with `time`, and "five" with `number_people`. The data statistics and short descriptions about the 17 domains are shown in Table 1. As the table indicates, the domains have very different granularity and diverse semantics. The total numbers of training, test and development queries across domains are 2964K, 217K and 153K, respectively. Note that we keep domain-specific slots such as `alarm_state`, but there are enough shared labels across domains. To be specific, we have shared 62 labels among 131 labels. In ALARM domain, there are 6 shared slots among 8 slots.

4.2 Results

In all our experiments, we follow same setting as in (Kim et al., 2015b; Kim et al., 2015c). We trained Conditional Random Fields (CRFs)(Lafferty et al., 2001) and used n-gram features up to $n = 3$, regular expression, lexicon features, and Brown Clusters (Brown et al., 1992). With these features, we compare the following methods for slot tagging[1]:

- *In-domain*: Train a domain-specific model using the domain-specific data covering the slots supported in that domain.
- *Binary*: Train a binary classifier for each slot type, assuming prediction for each slot type is independent of one another. Then combine the classification result with the slots needed for a given schema. For each binary slot tagger targeting a specific slot type, the labeled data is programatically

[1]For parameter estimation, we used L-BFGS (Liu and Nocedal, 1989) with 100 as the maximum number of iterations and 1.0 for the L2 regularization parameter.

Domain	In-domain	Binary	Post	Const(CRFs)	Const(LSTM)
Alarm	92.89	74.49	89.81	93.56	94.23
Calendar	90.03	75.62	82.14	88.57	88.16
Communication	92.94	84.17	86.93	92.14	90.39
Entertainment	93.83	83.28	91.26	93.17	94.37
Events	85.84	69.84	78.30	85.00	85.80
Hotel	91.25	73.86	77.45	91.12	90.81
Mediacontrol	86.39	83.70	86.22	87.07	86.43
Movietickets	91.75	85.39	87.03	91.06	91.35
Mystuff	87.92	51.30	80.48	84.88	82.46
Note	87.60	51.25	71.67	84.32	83.87
Ondevice	93.59	70.13	88.26	94.27	94.08
Orderfood	93.52	83.34	90.74	92.84	91.84
Places	91.75	75.27	87.69	89.55	90.96
Reminder	89.31	72.67	81.38	88.57	88.27
Reservations	92.68	86.10	91.07	93.56	94.32
Taxi	88.27	76.91	85.50	90.32	89.65
Weather	96.27	89.12	94.38	96.44	96.50
Average	90.93	75.67	85.31	90.38	90.21

Table 2: F1 scores for models which can handle all domains.

mapped to create a new labeled data set, where only the target label is kept while all the other labels are mapped `other` label.

- *Post*: Train a single model with all domain data, take the one-best parse of the tagger and filter-out slots outside the a given schema.

- *Const*: Train a single model with all domain data and then perform constrained decoding using a given schema.

To evaluate performance of the constrained decoding approach without schema prediction, we compare the performance among possible models, which can handle all domains in Table 2. Here, a schema is given from a pre-trained domain classifier with an average accuracy of 97%.

We consider *In-domain* as a plausible upper bound of the performance, yielding 90.93% of F1 on average. Second, *Binary* has the lowest performance of 75.67%. When we train a binary classifier for each slot type, the other slots that provide valuable contextual information are ignored. This leads to the degradation in tagging accuracy. Third, *Post* improves F1 scores across domains, resulting in 85.31% F1 on average. Note that this technique does not handle ambiguities and data distribution mismatches due to combining multiple domain specific data with different data sizes. Finally, *Const(CRF)* leads to consistent gains across all domains, achieving 90.38%, which almost matches the *In-domain* performance. *Const(CRF)* performs better than *Binary* because *Const(CRF)* constrains the best path search to the target domain schema. It does not consider the schema elements that are outside of the target domain schema. By doing so, it addresses the training data distribution issue as well as overlap on various schema elements.

Also, we performed experiments with LSTM for slot tagging by masking scores of predicted class labels from predicted schema. LSTM is one of the most popular deep learning techniques for sequence tagging (Bahdanau et al., 2014; Dyer et al., 2015), but we observe that the LSTM results (*Const(LSTM)*) on our dataset are very similar to that of CRFs (*Const(CRF)*, as shown in Table 2. In the following experiments, we mostly focus on the *Const* version of CRFs for simplicity.

The main results of constrained decoding with different schema prediction methods are shown in Table 3. *Bin* approach trains k binary logistic regression classifier for each slot type. Each binary classifier

Constrained by		Query					Domain
	In-domain	Bin_{Fix}	Bin_{Min}	Mult_{Fix}	Mult_{Min}	GoldQ	PredD
Alarm	92.89	90.56	84.89	91.37	96.29	97.74	93.56
Calendar	90.03	87.6	80.03	89.17	91.86	92.08	88.57
Comm.	92.94	91.28	77.94	91.89	93.29	95.97	92.14
Entertain.	93.83	92.41	81.83	91.9	95.54	96.5	93.17
Events	85.84	82.71	74.84	84.23	88.26	89.57	85
Hotel	91.25	89.25	82.25	90.75	92.23	93.21	91.12
Media.	86.39	82.94	85.39	84.58	92.99	93.99	87.07
Mvtickets	91.75	87.86	72.75	86.16	91.67	92.8	91.06
Mystuff	87.92	83.09	83.92	85.12	90.36	91.86	84.88
Note	87.6	81.84	78.6	83.38	87.58	88.42	84.32
Ondevice	93.59	91.87	69.59	92.22	97.79	98.6	94.27
Orderfood	93.52	91.25	76.52	93.62	95.92	96.24	92.84
Places	91.75	89.82	79.75	87.59	94.27	96.8	89.55
Reminder	89.31	86.1	71.31	87.18	93.89	94.07	88.57
Reservations	92.68	89.57	85.68	91.29	94.28	96.28	93.56
Taxi	88.27	86.63	72.27	89.07	95.42	97.11	90.32
Weather	96.27	94.65	89.27	95.8	98.5	99.11	96.44
Average	90.93	88.20	79.23	89.14	93.55	94.73	90.38

Table 3: F1 scores for *Const* with various schema prediction methods across 17 personal assistant domains.

determines if a query has a specific slot or not, while *Mult* approaches use a single LSTM to predict a set of allowed schema for a query. Subscript Fix denotes that a fixed threshold (0.5) is used to make a decision of positive versus negative label, and Min denotes that the threshold is set to be the minimum of positive label thresholds, which hence gives the maximum recall rate. *GoldQ* denotes the decoding was constrained by true schema for a query and and *PredD* denotes the decoding was constrained by a predicted domain. Here *In-domain* also uses domain classifier.

In the preliminary experiments, we observed that there are significant performance improvements when performing constrained decoding given a gold standard schema of a query *(GoldQ)*. However, it is very difficult to get similar performance by the predicted schema. The main reason is because it does not guarantee recall. As you can see, all methods based on schema prediction except for $\textit{Mult}_{Min}$, fail to achieve any improvement compared to *In-domain* and *PredD*. So, we use the minimum probability of a ground-truth schema type per query to increase recall. Using predicted minimum boundary $\textit{Mult}_{Min}$ finally boost up performance up to 93.55%, huge relative error reduction of 33% over *In-domain* approach.

Unlike previous experiments, the experiments shown in Table 4 assume that the true domain and its schema are given. So, there are no domain classification error. $\textit{MULT}_{Min}$ removes the predicted schema elements that are outside of the true domain schema. *In-domain* yields 92.53% F1 score. $\textit{MULT}_{Min}$ boosts the performance to 93.99%.

To further compare multi-classification approach to binary approach, we show performance for multi-class labeling task in Table 5. Unlike slot tagging performance, $\textit{Mult}_{Fix}$ has the highest F1 score because of its precision. $\textit{Mult}_{Min}$ has high recall at the slight expense of precision. However, binary logistic regression (Bin_{Min}) fails to keep reasonable precision. This is because logistic regression models are over-fitted to each label, minimum boundary is very low and thus it causes a lot false positives.

For the last scenario shown in Table 6, we assume that we do not have training data for the test domains. The amount of test data is about 2k. The $\textit{Mult}_{Min}$ performs reasonably well, yielding 96.48% on average. Interestingly, for the *Bus* domain, we can get almost perfect tagging performance of 99.5%. Note that all tags in *Bus* and *Ferry* domains are fully covered by our single model, but the *ShopElectric*

	In-domain	MULT$_{Min}$
Alarm	95.53	96.23
Calendar	90.37	92.46
Comm.	93.08	94.29
Entertain.	94.48	95.84
Events	89.47	90.02
Hotel	93.16	93.68
Mediactrl	90.87	92.7
Mvtickets	92.5	92.98
Mystuff	88.76	89.82
Note	89.48	90.58
Ondevice	95.27	97.26
Orderfood	94.35	96
Places	93.18	95.19
Reminder	90.22	92.77
Reservations	93.34	95.01
Taxi	91.37	94.2
Weather	97.64	98.83
Average	92.53	93.99

Table 4: F1 score for *In-domain* and *MULT$_{Min}$* across domains for constrained with predicted multi labels given true domain schema.

	Bin$_{Fix}$			Bin$_{Min}$			Mult$_{Fix}$			Mult$_{Min}$		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Alarm	87.7	67.6	76.4	65.2	99.2	78.7	88.2	86.5	87.3	69.3	99.1	81.6
Calendar	83.7	83.4	83.5	72.8	99.2	86.1	90.1	80.7	85.1	70.5	99.7	82.4
Comm.	78.7	86.1	82.2	66.1	98.8	79.2	86.5	85	85.7	74.8	98.9	85.2
Entertain.	87.4	72.6	79.3	58.4	99.7	73.5	88.3	84.6	86.4	63.3	99.1	77.4
Events	74.8	75.6	75.2	62.2	99.1	83.5	84.3	85.9	85.1	70.8	98.8	82.6
Hotel	87.3	85.9	86.6	58.2	98.4	73.1	84.9	89.2	87	82.2	98.2	89.6
Mediactrl	91.9	68.6	78.5	69.9	99.8	82.1	86	92.4	89.1	79.8	99.5	88.6
Mvtickets	81.3	77	79.1	72.4	99.3	83.8	86.4	87.7	87.1	73.2	99.3	84.3
Mystuff	87.6	82.4	84.9	61.7	98.9	79.5	78.8	82.4	80.5	78.3	98.5	87.4
Note	84.7	77.7	81.1	64.5	98.7	77.8	92.3	87.5	89.8	67.3	98	80
Ondevice	87.2	84.7	85.9	72.3	99.3	83.4	89.2	85.5	87.3	75.7	98.6	85.9
Orderfood	85	70.2	76.9	69.1	99.4	81.4	92.4	82.5	87.2	82.5	99.1	90.2
Places	89.9	83.5	85.9	73.1	99.6	85.9	87.2	70.7	75.8	68.9	99	75.7
Reminder	86.5	75.9	80.9	71.9	98.5	84.7	91.4	84.8	88	83.5	99.2	90.4
Reservations	90.7	85.2	87.9	51.1	99.3	67.5	90.7	80.5	85.3	79.9	99.5	88.6
Taxi	87.7	82	84.8	71.9	99.3	84.6	94.4	88.5	91.3	78.5	98.9	87.7
Weather	85	72.2	78.1	52.6	99.4	68.8	91.6	88.2	89.9	65.9	99.5	79.3
	85.7	78.3	81.6	65.5	99.2	79.6	88.4	84.9	86.4	74.4	99	84.5

Table 5: Multi-labeling task performance for schema prediction methods.

	Ferry	Bus	ShopElectric.	AVG.
Mult$_{Min}$	96.86	99.5	93.08	96.48

Table 6: F1 scores for *Mult$_{Min}$* across new domains which do not have domain specific training data.

domain is partially covered.

5 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. The single slot tagging coupled with schema prediction and constrained decoding achieves competitive and often superior performance over the conventional model trained in per domain fashion. This approach enables creation of new virtual domains through any combination of slot types covered in the single slot tagger schema, reducing the need to collect and annotate the same slot types multiple times for different domains.

Acknowledgements

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

References

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

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*, pages 120–128. Association for Computational Linguistics.

John Blitzer, Sham Kakade, and Dean P Foster. 2011. Domain adaptation with coupled subspaces. In *AISTATS*, pages 173–181.

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.

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*.

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.

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.

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.

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.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

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

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

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