New Transfer Learning Techniques for Disparate Label Sets

Young-Bum Kim[†] **Karl Stratos**[‡] **Ruhi Sarikaya**[†] **Minwoo Jeong**[†]

[†]Microsoft Corporation, Redmond, WA
[‡]Columbia University, New York, NY
{ybkim, ruhi.sarikaya, minwoo.jeong}@microsoft.com
stratos@cs.columbia.edu

Abstract

In natural language understanding (NLU), a user utterance can be labeled differently depending on the domain or application (e.g., weather vs. calendar). Standard domain adaptation techniques are not directly applicable to take advantage of the existing annotations because they assume that the label set is invariant. We propose a solution based on label embeddings induced from canonical correlation analysis (CCA) that reduces the problem to a standard domain adaptation task and allows use of a number of transfer learning techniques. We also introduce a new transfer learning technique based on pretraining of hidden-unit CRFs (HUCRFs). We perform extensive experiments on slot tagging on eight personal digital assistant domains and demonstrate that the proposed methods are superior to strong baselines.

1 Introduction

The main goal of NLU is to automatically extract the meaning of spoken or typed queries. In recent years, this task has become increasingly important as more and more speech-based applications have emerged. Recent releases of personal digital assistants such as Siri, Google Now, Dragon Go and Cortana in smart phones provide natural language based interface for a variety of domains (e.g. places, weather, communications, reminders). The NLU in these domains are based on statistical machine learned models which require annotated training data. Typically each domain has its own schema to annotate the words and queries. However the meaning of words and utterances could be different in each domain. For example, "sunny" is considered a weather condition in the weather domain but it may be a song title in a music domain. Thus every time a new application is developed or a new domain is built, a significant amount of resources is invested in creating annotations specific to that application or domain.

One might attempt to apply existing techniques (Blitzer et al., 2006; Daumé III, 2007) in domain adaption to this problem, but a straightforward application is not possible because these techniques assume that the label set is invariant.

In this work, we provide a simple and effective solution to this problem by abstracting the label types using the canonical correlation analysis (CCA) by Hotelling (Hotelling, 1936) a powerful and flexible statistical technique for dimensionality reduction. We derive a low dimensional representation for each label type that is maximally correlated to the average context of that label via CCA. These shared label representations, or label embeddings, allow us to map label types across different domains and reduce the setting to a standard domain adaptation problem. After the mapping, we can apply the standard transfer learning techniques to solve the problem.

Additionally, we introduce a novel pretraining technique for hidden-unit CRFs (HUCRFs) to effectively transfer knowledge from one domain to another. In our experiments, we find that our pretraining method is almost always superior to strong baselines such as the popular domain adaptation method of Daumé III (2007).

2 Problem description and related work

Let $\mathcal{D}$ be the number of distinct domains. Let X_i be the space of observed samples for the i-th domain. Let Y_i be the space of possible labels for the i-th domain. In most previous works in domain adaptation (Blitzer et al., 2006; Daumé III, 2007), observed data samples may vary but label space is

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 473–482, Beijing, China, July 26-31, 2015.

invariant[1]. That is,

$$Y_i = Y_j \qquad \forall i, j \in \{1 \dots \mathcal{D}\}$$

but $X_i \neq X_j$ for some domains i and j. For example, in part-of-speech (POS) tagging on newswire and biomedical domains, the observed data sample may be radically different but the POS tag set remains the same.

In practice, there are cases, where the same query is labeled differently depending on the domain or application and the context. For example, *Fred Myer* can be tagged differently; "send a text message to *Fred Myer*" and "get me driving direction to *Fred Myer* ". In the first case, *Fred Myer* is person in user's contact list but it is a grocery store in the second one.

So, we relax the constraint that label spaces must be the same. Instead, we assume that surface forms (i.e words) are similar. This is a natural setting in developing multiple applications on speech utterances; input spaces (service request utterances) do not change drastically but output spaces (slot tags) might.

Multi-task learning differs from our task. In general multi-task learning aims to improve performance across all domains while our domain adaptation objective is to optimize the performance of semantic slot tagger on the target domain.

Below, we review related work in domain adaption and natural language understanding (NLU).

2.1 Related Work

Domain adaptation has been widely used in many natural language processing (NLP) applications including part-of-speech tagging (Schnabel and Schütze, 2014), parsing (McClosky et al., 2010), and machine translation (Foster et al., 2010). Most of the work can be classified either supervised domain adaptation (Chelba and Acero, 2006; Blitzer et al., 2006; Daume III and Marcu, 2006; Daumé III, 2007; Finkel and Manning, 2009; Chen et al., 2011) or semi-supervised adaptation (Ando and Zhang, 2005; Jiang and Zhai, 2007; Kumar et al., 2010; Huang and Yates, 2010). Our problem setting falls into the former.

Multi-task learning has become popular in NLP. Sutton and McCallum (2005) showed that joint learning and/or decoding of sub-tasks helps to improve performance. Collobert and Weston (2008) proved the similar claim in a deep learning architecture. While our problem resembles their settings, there are two clear distinctions. First, we aim to optimize performance on the target domain by minimizing the gap between source and target domain while multi-task learning jointly learns the shared tasks. Second, in our problem the domains are different, but they are closely related. On the other hand, prior work focuses on multiple subtasks of the same data.

Despite the increasing interest in NLU (De Mori et al., 2008; Xu and Sarikaya, 2013; Sarikaya et al., 2014; Xu and Sarikaya, 2014; Anastasakos et al., 2014; El-Kahky et al., 2014; Liu and Sarikaya, 2014; Marin et al., 2014; Celikyilmaz et al., 2015; Ma et al., 2015; Kim et al., 2015), transfer learning in the context of NLU has not been much explored. The most relevant previous work is Tur (2006) and Li et al. (2011), which described both the effectiveness of multi-task learning in the context of NLU. For multi-task learning, they used shared slots by associating each slot type with aggregate active feature weight vector based on an existing domain specific slot tagger. Our empirical results shows that these vector representation might be helpful to find shared slots across domain, but cannot find bijective mapping between domains.

Also, Jeong and Lee (2009) presented a transfer learning approach in multi-domain NLU, where the model jointly learns slot taggers in multiple domains and simultaneously predicts domain detection and slot tagging results.[2] To share parameters across domains, they added an additional node for domain prediction on top of the slot sequence. However, this framework also limited to a setting in which the label set remains invariant. In contrast, our method is restricted to this setting without any modification of models.

3 Sequence Modeling Technique

The proposed techniques in Section 4 and 5 are generic methodologies and not tied to any particular models such as any sequence models and instanced based models. However, because of superior performance over CRF, we use a hidden unit CRF (HUCRF) of Maaten et al. (2011).

[1] Multilingual learning (Kim et al., 2011; Kim and Snyder, 2012; Kim and Snyder, 2013) has same setting.

[2] Jeong and Lee (2009) pointed out that if the domain is given, their method is the same as that of Daumé III (2007).

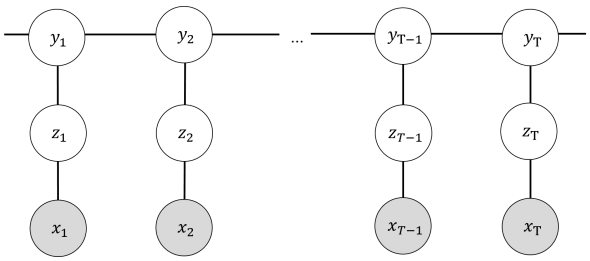

Figure 1: Graphical representation of hidden unit CRFs.

While popular and effective, a CRF is still a linear model. In contrast, a HUCRF benefits from nonlinearity, leading to superior performance over CRF (Maaten et al., 2011). Thus we will focus on HUCRFs to demonstrate our techniques in experiments.

3.1 Hidden Unit CRF (HUCRF)

A HUCRF introduces a layer of binary-valued hidden units $z = z_1 \dots z_n \in \{0, 1\}$ for each pair of label sequence $y = y_1 \dots y_n$ and observation sequence $x = x_1 \dots x_n$. A HUCRF parametrized by $\theta \in \mathbb{R}^d$ and $\gamma \in \mathbb{R}^{d'}$ defines a joint probability of y and z conditioned on x as follows:

$$p_{\theta,\gamma}(y, z|x) = \frac{\exp(\theta^\top \Phi(x, z) + \gamma^\top \Psi(z, y))}{\sum_{\substack{z' \in \{0,1\}^n \\ y' \in \mathcal{Y}(x,z')}} \exp(\theta^\top \Phi(x, z') + \gamma^\top \Psi(z', y'))} \quad (1)$$

where $\mathcal{Y}(x, z)$ is the set of all possible label sequences for x and z, and $\Phi(x, z) \in \mathbb{R}^d$ and $\Psi(z, y) \in \mathbb{R}^{d'}$ are global feature functions that decompose into local feature functions:

$$\Phi(x, z) = \sum_{j=1}^{n} \phi(x, j, z_j)$$

$$\Psi(z, y) = \sum_{j=1}^{n} \psi(z_j, y_{j-1}, y_j)$$

HUCRF forces the interaction between the observations and the labels at each position j to go through a latent variable z_j: see Figure 1 for illustration. Then the probability of labels y is given by marginalizing over the hidden units,

$$p_{\theta,\gamma}(y|x) = \sum_{z \in \{0,1\}^n} p_{\theta,\gamma}(y, z|x)$$

As in restricted Boltzmann machines (Larochelle and Bengio, 2008), hidden units are conditionally independent given observations and labels. This allows for efficient inference with HUCRFs despite their richness (see Maaten et al. (2011) for details). We use a perceptron-style algorithm of Maaten et al. (2011) for training HUCRFs.

4 Transfer learning between domains with different label sets

In this section, we describe three methods for utilizing annotations in domains with different label types. First two methods are about transferring features and last method is about transferring model parameters. Each of these methods requires some sort of *mapping* for label types. A fine-grained label type needs to be mapped to a coarse one; a label type in one domain needs to be mapped to the corresponding label type in another domain. We will provide a solution to obtaining these label mappings automatically in Section 5.

4.1 Coarse-to-fine prediction

This approach has some similarities to the method of Li et al. (2011) in that shared slots are used to transfer information between domains. In this two-stage approach, we train a model on the source domain, make predictions on the target domain, and then use the predicted labels as additional features to train a final model on the target domain. This can be helpful if there is some correlation between the label types in the source domain and the label types in the target domain.

However, it is not desirable to directly use the label types in the source domain since they can be highly specific to that particular domain. An effective way to combat this problem is to reduce the original label types such `start-time`, `contract-info`, and `restaurant` as to a set of coarse label types such as `name`, `date`, `time`, and `location` that are universally shared across all domains. By doing so, we can use the first model to predict generic labels such as `time` and then use the second model to use this information to predict fine-grained labels such as `start-time` and `end-time`.

4.2 Method of Daumé III (2007)

In this popular technique for domain adaptation, we train a model on the union of the source domain data and the target domain data

but with the following preprocessing step: each feature is duplicated and the copy is conjoined with a domain indicator. For example, in a `WEATHER` domain dataset, a feature that indicates the identity of the string "Sunny" will generate both $w(0) = Sunny$ and $(w(0) = Sunny) \wedge (domain = WEATHER)$ as feature types. This preprocessing allows the model to utilize all data through the common features and at the same time specialize to specific domains through the domain specific features. This is especially helpful when there is label ambiguity on particular features (e.g., "Sunny" might be a `weather-condition` in a `WEATHER` domain dataset but a `music-song-name` in a `MUSIC` domain dataset).

Note that a straightforward application of this technique is in general not feasible in our situation. This is because we have features conjoined with label types and our domains do *not* share label types. This breaks the sharing of features across domains: many feature types in the source domain are disjoint from those in the target domain due to different labeling.

Thus it is necessary to first map source domain label types to target domain label type. After the mapping, features are shared across domains and we can apply this technique.

4.3 Transferring model parameter

In this approach, we train HUCRF on the source domain and transfer the learned parameters to initialize the training process on the target domain. This can be helpful for at least two reasons:

1. The resulting model will have parameters for feature types observed in the source domain as well as the target domain. Thus it has better feature coverage.

2. If the training objective is non-convex, this initialization can be helpful in avoiding bad local optima.

Since the training objective of HUCRFs is non-convex, both benefits can apply. We show in our experiments that this is indeed the case: the model benefits from both better feature coverage and better initialization.

Note that in order to use this approach, we need to map source domain label types to target domain label type so that we know which parameter in the source domain corresponds to which parameter in the target domain. This can be a many-to-one, one-to-many, one-to-one mapping depending on the label sets.

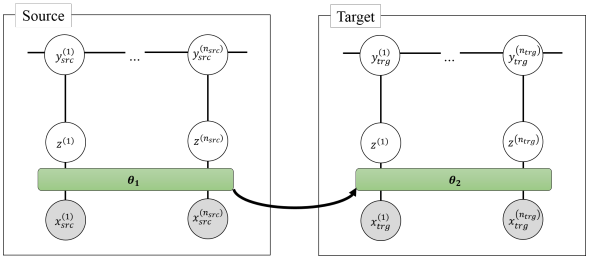

Figure 2: Illustration of a pretraining scheme for HUCRFs.

4.3.1 Pretraining with HUCRFs

In fact, pretraining HUCRFs in the source domain can be done in various ways. Recall that there are two parameter types: $\theta \in \mathbb{R}^d$ for scoring observations and hidden states and $\gamma \in \mathbb{R}^{d'}$ for scoring hidden states and labels (Eq. (1)). In pretraining, we first train a model (θ_1, γ_1) on the source data $\{(x^{(i)}_{src}, y^{(i)}_{src})\}_{i=1}^{n_{src}}$:

$$(\theta_1, \gamma_1) \approx \arg\max_{\theta,\gamma} \sum_{i=1}^{n_{src}} \log p_{\theta,\gamma}(y^{(i)}_{src}|x^{(i)}_{src})$$

Then we train a model (θ_2, γ_2) on the target data $\{(x^{(i)}_{trg}, y^{(i)}_{trg})\}_{i=1}^{n_{trg}}$ by initializing $(\theta_2, \gamma_2) \leftarrow (\theta_1, \gamma_1)$:

$$(\theta_2, \gamma_2) \approx \arg\max_{\theta,\gamma} \sum_{i=1}^{n_{trg}} \log p_{\theta,\gamma}(y^{(i)}_{trg}|x^{(i)}_{trg})$$

Here, we can choose to initialize only $\theta_2 \leftarrow \theta_1$ and discard the parameters for hidden states and labels since they may not be the same. The θ_1 parameters model the hidden structures in the source domain data and serve as a good initialization point for learning the θ_2 parameters in the target domain. This can be helpful if the mapping between the label types in the source data and the label types in the target data is unreliable. This process is illustrated in Figure 2.

5 Automatic generation of label mappings

All methods described in Section 4 require a way to propagate the information in label types across different domains. A straightforward solution would be to manually construct

such mappings by inspection. For instance, we can specify that `start-time` and `end-time` are grouped as the same label `time`, or that the label `public-transportation-route` in the `PLACES` domain maps to the label `implicit-location` in the `CALENDAR` domain.

Instead, we propose a technique that automatically generates the label mappings. We induce vector representations for all label types through canonical correlation analysis (CCA) — a powerful and flexible technique for deriving low-dimensional representation. We give a review of CCA in Section 5.1 and describe how we use the technique to construct label mappings in Section 5.2.

5.1 Canonical Correlation Analysis (CCA)

CCA is a general technique that operates on a pair of multi-dimensional variables. CCA finds k dimensions (k is a parameter to be specified) in which these variables are maximally correlated.

Let $x_1 \dots x_n \in \mathbb{R}^d$ and $y_1 \dots y_n \in \mathbb{R}^{d'}$ be n samples of the two variables. For simplicity, assume that these variables have zero mean. Then CCA computes the following for $i = 1 \dots k$:

$$\underset{\substack{u_i \in \mathbb{R}^d,\, v_i \in \mathbb{R}^{d'}: \\ u_i^\top u_{i'} = 0 \;\; \forall i' < i \\ v_i^\top v_{i'} = 0 \;\; \forall i' < i}}{\arg\max} \frac{\sum_{l=1}^{n} (u_i^\top x_l)(v_i^\top y_l)}{\sqrt{\sum_{l=1}^{n} (u_i^\top x_l)^2} \sqrt{\sum_{l=1}^{n} (v_i^\top y_l)^2}}$$

In other words, each (u_i, v_i) is a pair of projection vectors such that the *correlation* between the projected variables $u_i^\top x_l$ and $v_i^\top y_l$ (now scalars) is maximized, under the constraint that this projection is *uncorrelated* with the previous $i - 1$ projections.

This is a non-convex problem due to the interaction between u_i and v_i. Fortunately, a method based on singular value decomposition (SVD) provides an efficient and exact solution to this problem (Hotelling, 1936). The resulting solution $u_1 \dots u_k \in \mathbb{R}^d$ and $v_1 \dots v_k \in \mathbb{R}^{d'}$ can be used to project the variables from the original d- and d'-dimensional spaces to a k-dimensional space:

$$x \in \mathbb{R}^d \longrightarrow \bar{x} \in \mathbb{R}^k : \bar{x}_i = u_i^\top x$$
$$y \in \mathbb{R}^{d'} \longrightarrow \bar{y} \in \mathbb{R}^k : \bar{y}_i = v_i^\top y$$

The new k-dimensional representation of each variable now contains information about the other variable. The value of k is usually selected to be much smaller than d or d', so the representation is typically also low-dimensional.

5.2 Inducing label embeddings

We now describe how to use CCA to induce vector representations for label types. Using the same notation, let n be the number of instances of labels in the entire data. Let $x_1 \dots x_n$ be the original representations of the label samples and $y_1 \dots y_n$ be the original representations of the associated words set contained in the labels.

We employ the following definition for the original representations for reasons we explain below. Let d be the number of distinct label types and d' be the number of distinct word types.

- $x_l \in \mathbb{R}^d$ is a zero vector in which the entry corresponding to the label type of the l-th instance is set to 1.
- $y_l \in \mathbb{R}^{d'}$ is a zero vector in which the entries corresponding to words spanned by the label are set to 1.

The motivation for this definition is that similar label types often have similar or same word.

For instance, consider two label types `start-time`, (start time of a calendar event) and `end-time`, meaning (the end time of a calendar event). Each type is frequently associated with phrases about time. The phrases {"9 pm", "7", "8 am"} might be labeled as `start-time`; the phrases {"9 am", "7 pm"} might be labeled as `end-time`. In these examples, both label types share words "am", "pm", "9", and "7" even though phrases may not match exactly.

Figure 3 gives the CCA algorithm for inducing label embeddings. It produces a k-dimensional vector for each label type corresponding to the CCA projection of the one-hot encoding of that label.

5.3 Discussion on alternative label representations

We point out that there are other options for inducing label representations besides CCA. For instance, one could simply use the sparse feature vector representation of each label. However, CCA's low-dimensional projection is computationally more convenient and arguably more generalizable. One can also consider training a predictive model similar to `word2vec` (Mikolov

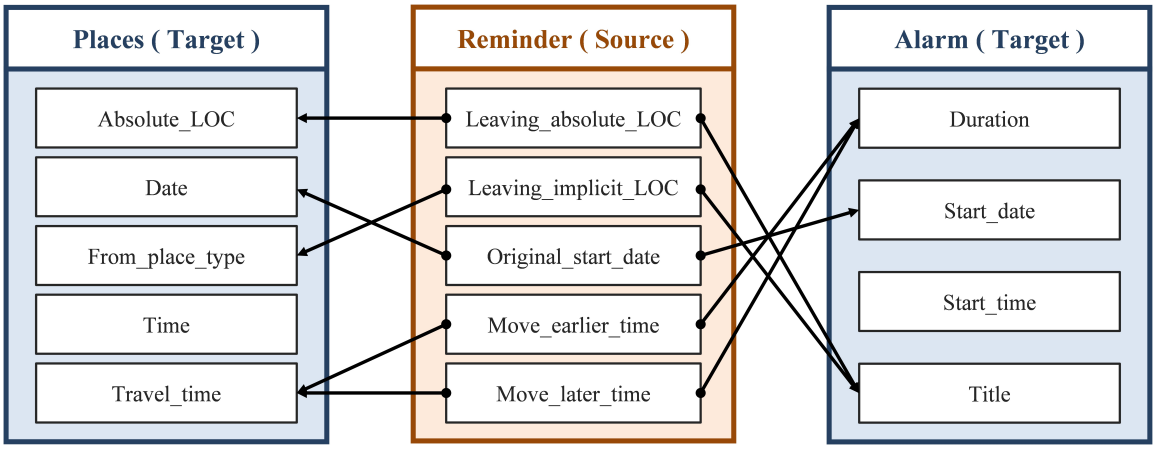

Figure 4: Bijective mapping: labels in REMINDER domain (orange box) are mapped into those in PLACES and ALARM domains.

CCA-LABEL
Input: labeled sequences $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, dimension k
Output: label vector $v \in \mathbb{R}^k$ for each label type

1. For each label type $l \in \{1 \dots d\}$ and word type $w \in \{1 \dots d\}$ present in the sequences, calculate
   - **count**(l) = number of times label l occurs
   - **count**(w) = number of times word w occurs
   - **count**(l, w) = number of times word w occurs under label l
2. Define a matrix $\Omega \in \mathbb{R}^{d \times d'}$ where:
$$\Omega_{l,w} = \frac{\textbf{count}(l, w)}{\sqrt{\textbf{count}(l)\textbf{count}(w)}}$$
3. Perform rank-k SVD on Ω. Let $U \in \mathbb{R}^{d \times k}$ be a matrix where the i-th column is the left singular vector of Ω corresponding to the i-th largest singular value.
4. For each label l, set the l-th normalized row of U to be its vector representation.

Figure 3: CCA algorithm for inducing label embeddings.

et al., 2013). But this requires significant efforts in implementation and also very long training time. In contrast, CCA is simple, efficient, and effective and can be readily implemented. Also, CCA is theoretically well understood while methods inspired by neural networks are not.

5.4 Constructing label mappings

Vector representations of label types allow for natural solutions to the task of constructing label mappings.

5.4.1 Mapping to a coarse label set

Given a domain and the label types that occur in the domain, we can reduce the number of label types by simply clustering their vector representations. For instance, if the embeddings for start-time and end-time are close together, they will be grouped as a single label type. We run the k-means algorithm on the label embeddings to obtain this coarse label set.

Table 1 shows examples of this clustering. It demonstrates that the CCA representations obtained by the procedure described in Section 5.2 are indeed informative of the labels' properties.

Cluster	Labels	Cluster	Labels
Time	start_time end_time original_start_time travel_time	Person	contact_info artist from_contact_name relationship_name
Loc	absolute_loc leaving_loc from_loc position_ref	Loc_ATTR	prefer_route public_trans_route nearby distance

Table 1: Some of cluster examples

5.4.2 Bijective mapping between label sets

Given a pair of domains and their label sets, we can create a bijective label mapping by finding the nearest neighbor of each label type. Figure 4 shows some actual examples of CCA-based bijective maps, where the label set in the REMINDER domain is mapped to the PLACES and ALARM domains. One particularly interesting example is that move_earlier_time in REMINDER domain is mapped to Travel_time in PLACES and Duration in ALARM domain. This is a tag used in a user utterance requesting to move an

Domains	# of label	Source Training	Test	Description
Alarm	7	27865	3334	Set alarms
Calendar	20	50255	7017	Set appointments & meetings in the calendar
Communication	18	104881	14484	Make calls, send texts, and communication related user request
Note	4	17445	2342	Note taking
Ondevice	7	60847	9704	Phone settings
Places	32	150348	20798	Find places & get direction
Reminder	16	62664	8235	Setting time, person & place based reminder
Weather	9	53096	9114	Weather forecasts & historical information about weather patterns

Table 2: Size of number of label, labeled data set size and description for Alarm, Calendar, Communication, Note, Ondevice, Places, Reminder and Weather domains partitioned into training and test set.

appointment to an earlier time. For example, in the query "move the dentist's appointment up by 30 minutes.", the phrase "30 minutes" is tagged with `move_earlier_time`. The role of this tag is very similar to the role of `Travel_time` in `PLACES` (not `Time`) and `Duration` in `ALARMS` (not `Start_date`), and CCA is able to recover this relation.

6 Experiments

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

6.1 Setup

To test the effectiveness of our approach, we apply it to a suite of eight Cortana personal assistant domains for slot sequence tagging tasks, where the goal is to find the correct semantic tagging of the words in a given user utterance.

The data statistics and short descriptions are shown in Table 2. As the table indicates, the domains have very different granularity and diverse semantics.

6.2 Baselines

In all our experiments, we trained HUCRF and only used n-gram features, including unigram, bigram, and trigram within a window of five words ($\pm$2 words) around the current word as binary feature functions. With these features, we compare the following methods for slot tagging:

- *NoAdapt*: train only on target training data.
- *Union*: train on the union of source and target training data.
- *Daume*: train with the feature duplication method described in 4.2.
- *C2F*: train with the coarse-to-fine prediction method described in 4.1.
- *Pretrain*: train with the pretraining method described in 4.3.1.

To apply these methods except for *Target*, we treat each of the eight domains in turn as the test domain, with one of remaining seven domain as the source domain. As in general domain adaptation setting, we assume that the source domain has a sufficient amount of labeled data but the target domain has an insufficient amount of labeled data. Specifically, For each test or target domain, we only use 10% of the training examples to simulate data scarcity. In the following experiments, we report the slot F-measure, using the standard CoNLL evaluation script [3]

6.3 Results on mappings

	Mapping technique		
Adaptation technique	Manual	Li et al. (2011)	CCA
Union	68.16	64.7	70.51
Daume	73.42	67.32	75.85
C2F	75.47	75.69	76.29
Pretrain	77.72	76.99	78.76
NoAdapt		75.13	

Table 3: Comparison of slot F1 scores using the proposed CCA-derived mapping versus other mapping methods combined with different adaptation techniques.

To assess the quality of our automatic mapping methods via CCA described in Section 5, we compared against manually established mappings and also the mapping method of Li et al. (2011). The method of Li et al. (2011) is to associate each slot type with the aggregate active feature weight vectors based on an existing domain specific slot tagger (a CRF). Manual mapping were performed

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

Target	Source		Minimum distance domain performance			
Domain	Nearest Domain	NoAdapt	Union	Daume	C2F	Pretrain
Alarm	Calendar	74.82	84.46	**84.97**	81.54	84.88
Calendar	Reminder	70.51	73.94	73.07	72.82	**77.08**
Note	Reminder	65.38	56.39	**69.89**	66.6	69.55
Ondevice	Weather	70.86	66.66	71.17	71.49	**73.5**
Reminder	Calendar	77.3	**83.38**	82.19	81.29	83.22
Communication	Reminder	79.31	74.28	80.33	79.66	**82.96**
Places	Weather	73.93	73.74	75.86	73.73	**80.11**
Weather	Places	92.78	92.88	94.43	93.75	**97.18**
Average	-	75.61	75.72	78.99	77.61	**81.06**

Table 4: Slot F1 scores on each target domain using adapted models from the nearest source domain.

Source \ Target		Alarm	Calendar	Note	Ondevice	Reminder	Communication	Places	Weather	Average
NoAdapt		74.82	70.51	65.38	70.86	77.3	79.31	73.93	92.78	75.61
Alarm	Union	-	72.26	59.92	67.32	79.45	77.91	73.78	92.67	74.76
	Daume	-	72.77	66.28	70.94	81.12	80.38	75.62	93.12	77.18
	C2F	-	70.59	64.06	71	78.8	79.5	74.29	92.75	75.86
	Pretrain	-	**76.68**	**68.12**	**71.8**	**81.25**	**81.5**	**77.1**	**95.03**	**78.78**
Calendar	Union	84.46	-	50.64	64.7	**83.38**	75.02	71.13	93.2	74.65
	Daume	**84.97**	-	65.43	70.12	82.19	79.78	75.21	93.1	78.69
	C2F	81.54	-	66.08	71.22	81.29	80.11	73.75	93.18	78.17
	Pretrain	84.88	-	**69.21**	**72.3**	83.22	**82.75**	**77.89**	**95.8**	**80.86**
Note	Union	60.26	60.42	-	65.79	69.81	76.85	70.56	90.02	70.53
	Daume	66.03	67.38	-	69.54	76.65	77.83	73.49	92.09	74.72
	C2F	74.68	70.51	-	71.34	77.49	79.48	74.17	92.89	77.22
	Pretrain	**75.52**	**72.4**	-	**71.4**	**80.1**	**82.06**	**76.53**	**94.22**	**78.89**
Ondevice	Union	63.72	66.28	55.67	-	75.16	74.85	70.59	90.7	71.00
	Daume	71.01	69.39	64.02	-	75.75	77.92	74.41	92.62	75.02
	C2F	74.02	70.33	64.99	-	77.43	79.53	73.84	92.71	76.12
	Pretrain	**76.27**	**71.59**	**67.21**	-	**78.67**	**82.34**	**77.45**	**95.04**	**78.37**
Reminder	Union	84.74	73.94	56.39	61.27	-	74.28	68.14	92.22	73.00
	Daume	84.66	73.07	**69.89**	67.94	-	80.33	73.36	93.19	77.49
	C2F	80.42	72.82	66.6	71.36	-	79.66	74.35	92.38	76.80
	Pretrain	**84.75**	**77.08**	69.55	**71.9**	-	**82.96**	**78.57**	**95.37**	**80.03**
Communication	Union	58.25	54.69	65.28	62.95	63.98	-	68.16	87.13	65.78
	Daume	70.4	67.41	69.14	69.26	77.67	-	73.33	92.82	74.29
	C2F	74.54	70.84	65.48	70.81	77.68	-	74.15	92.79	75.18
	Pretrain	**76.04**	**74.01**	**68.76**	**73.2**	**80.74**	-	**76.83**	**94.58**	**77.74**
Places	Union	71.7	67.56	45.37	53.93	67.78	63.67	-	92.88	66.13
	Daume	75.69	69.01	66.11	65.46	79.01	78.42	-	94.43	75.45
	C2F	**78.9**	71.64	66.93	71.26	79.2	79.19	-	93.75	77.27
	Pretrain	76.8	**74.12**	**67.5**	**72.7**	**81**	**81.89**	-	**97.18**	**78.74**
Weather	Union	69.43	58.53	56.76	66.66	74.98	77.53	73.74	-	68.23
	Daume	75	71.73	66.54	71.17	**79.36**	80.57	75.86	-	74.32
	C2F	**77.61**	71.47	63.24	71.49	78.44	79.43	73.73	-	73.63
	Pretrain	77.37	**74.5**	**68.23**	**73.5**	80.96	**82.05**	**80.11**	-	**76.67**
Average	Union	70.37	64.81	55.72	63.23	73.51	74.3	70.87	91.26	70.51
	Daume	75.4	70.23	66.77	69.2	78.32	79.32	74.47	93.05	75.85
	C2F	77.39	71.17	65.4	71.21	78.62	79.56	74.04	92.92	76.29
	Pretrain	**78.80**	**74.34**	**68.37**	**72.40**	**80.85**	**82.22**	**77.78**	**95.32**	**78.76**

Table 5: Slot F1 scores of using Union, Daume, Coarse-to-Fine and pretraining on all pairs of source and target data. The numbers in boldface are the best performing adaptation technique in each pair.

by two experienced annotators who have PhD in linguistics and machine learning. Each annotator first assigned mapping slot labels independently and then both annotators collaborated to reduce disagreement of their mapping results. Initially, the disagreement of their mapping rate between two annotators was about 30% because labels of slot tagging are very diverse; furthermore, in some cases it is not clear for human annotators if there exists a valid mapping.

The results are shown at Table 3. Vector representation of Li et al. (2011) increases the F1 score slightly from 75.13 to 75.69 in *C2F*, but it does not help as much in cases that require bijective mapping: *Daume*, *Union* and *Pretrain*.

In contrast, the proposed CCA based technique consistently outperforms the *NoAdapt* baselines by significant margins. More importantly, it also outperforms manual results under all conditions. It is perhaps not so surprising – the CCA derived mapping is completely data driven, while human annotators have nothing but the prior linguistic

knowledge about the slot tags and the domain.

6.4 Main Results

The full results are shown in Table 5, where all pairs of source and target languages are considered for domain adaptation. It is clear from the table that we can always achieve better results using adaptation techniques than the non-adapted models trained only on the target data. Also, our proposed pretraining method outperforms other types of adaptation in most cases.

The overall result of our experiments are shown in Table 4. In this experiment, we compare different adaptation techniques using our suggested CCA-based mapping. Here, except for *NoAdapt*, we use both the target and the nearest source domain data. To find the nearest domain, we first map fine grained label set to coarse label set by using the method described in Section 5.4.1 and then count how many coarse labels are used in a domain. And then we can find the nearest source domain by calculating the l_2 distance between the multinomial distributions of the source domain and the target domain over the set of coarse labels.

For example, for `CALENDAR`, we identify `REMINDER` as the nearest domain and vice versa because most of their labels are attributes related to `time`. In all experiments, the domain adapted models perform better than using only target domain data which achieves 75.1% F1 score. Simply combining source and target domain using our automatically mapped slot labels performs slightly better than baseline. *C2F* boosts the performance up to 77.61% and *Daume* is able to reach 78.99%.[4] Finally, our proposed method, *pretrain* achieves nearly 81.02% F1 score.

7 Conclusion

We presented an approach to take advantage of existing annotations when the data are similar but the label sets are different. This approach was based on label embeddings from CCA, which reduces the setting to a standard domain adaptation problem. Combined with a novel pretraining scheme applied to hidden-unit CRFs, our approach is shown to be superior to strong baselines in extensive experiments for slot tagging on eight distinct personal assistant domains.

[4]It is known that *Daume* is less beneficial when the source and target domains are similar due to the increased number of features.

References

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

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

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the EMNLP*, pages 120–128. Association for Computational Linguistics.

Asli Celikyilmaz, Dilek Hakkani-Tur, Panupong Pasupat, and Ruhi Sarikaya. 2015. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. AAAI - Association for the Advancement of Artificial Intelligence.

Ciprian Chelba and Alex Acero. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *Advances in neural information processing systems*, pages 2456–2464.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the ICML*, pages 160–167. ACM.

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

Hal Daumé III. 2007. Frustratingly easy domain adaptation. *proceedings of the ACL*, page 256.

Renato De Mori, Frédéric Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken language understanding. *Signal Processing Magazine, IEEE*, 25(3):50–58.

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.

Jenny Rose Finkel and Christopher D Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of the ACL*, pages 602–610. Association for Computational Linguistics.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the EMNLP*, pages 451–459. Association for Computational Linguistics.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Fei Huang and Alexander Yates. 2010. Exploring representation-learning approaches to domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 23–30. Association for Computational Linguistics.

Minwoo Jeong and Gary Geunbae Lee. 2009. Multi-domain spoken language understanding with transfer learning. *Speech Communication*, 51(5):412–424.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the ACL*, volume 7, pages 264–271. Association for Computational Linguistics.

Young-Bum Kim and Benjamin Snyder. 2012. Universal grapheme-to-phoneme prediction over latin alphabets. In *Proceedings of the EMNLP*, pages 332–343, Jeju Island, South Korea, July. Association for Computational Linguistics.

Young-Bum Kim and Benjamin Snyder. 2013. Unsupervised consonant-vowel prediction over hundreds of languages. In *Proceedings of the ACL*, pages 1527–1536. Association for Computational Linguistics.

Young-Bum Kim, João V Graça, and Benjamin Snyder. 2011. Universal morphological analysis using structured nearest neighbor prediction. In *Proceedings of the EMNLP*, pages 322–332. Association for Computational Linguistics.

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

Abhishek Kumar, Avishek Saha, and Hal Daume. 2010. Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 478–486.

Hugo Larochelle and Yoshua Bengio. 2008. Classification using discriminative restricted boltzmann machines. In *Proceedings of the ICML*.

Xiao Li, Ye-Yi Wang, and Gökhan Tür. 2011. Multi-task learning for spoken language understanding with shared slots. In *Proceeding of the INTERSPEECH*, pages 701–704. IEEE.

Xiaohu Liu and Ruhi Sarikaya. 2014. A discriminative model based entity dictionary weighting approach for spoken language understanding. IEEE Institute of Electrical and Electronics Engineers.

Yi Ma, Paul A. Crook, Ruhi Sarikaya, and Eric Fosler-Lussier. 2015. Knowledge graph inference for spoken dialog systems. In *Proceedings of the ICASSP*. IEEE.

Laurens Maaten, Max Welling, and Lawrence K Saul. 2011. Hidden-unit conditional random fields. In *International Conference on Artificial Intelligence and Statistics*.

Alex Marin, Roman Holenstein, Ruhi Sarikaya, and Mari Ostendorf. 2014. Learning phrase patterns for text classification using a knowledge graph and unlabeled data. ISCA - International Speech Communication Association.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proceedings of the NAACL*, pages 28–36. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ruhi Sarikaya, Asli C, Anoop Deoras, and Minwoo Jeong. 2014. Shrinkage based features for slot tagging with conditional random fields. Proceeding of ISCA - International Speech Communication Association, September.

Tobias Schnabel and Hinrich Schütze. 2014. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26.

Charles Sutton and Andrew McCallum. 2005. Composition of conditional random fields for transfer learning. In *Proceedings of the EMNLP*, pages 748–754. Association for Computational Linguistics.

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

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

Puyang Xu and Ruhi Sarikaya. 2014. Targeted feature dropout for robust slot filling in natural language understanding. ISCA - International Speech Communication Association.