Adversarial Adaptation of Synthetic or Stale Data

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

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

Abstract

Two types of data shift common in practice are 1. transferring from synthetic data to live user data (a deployment shift), and 2. transferring from stale data to current data (a temporal shift). Both cause a distribution mismatch between training and evaluation, leading to a model that overfits the flawed training data and performs poorly on the test data. We propose a solution to this mismatch problem by framing it as domain adaptation, treating the flawed training dataset as a source domain and the evaluation dataset as a target domain. To this end, we use and build on several recent advances in neural domain adaptation such as adversarial training (Ganin et al., 2016) and domain separation network (Bousmalis et al., 2016), proposing a new effective adversarial training scheme. In both supervised and unsupervised adaptation scenarios, our approach yields clear improvement over strong baselines.

1 Introduction

Spoken language understanding (SLU) systems analyze various aspects of a user query by classifying its domain, intent, and semantic slots. For instance, the query `how is traffic to target in bellevue` has domain `PLACES`, intent `CHECK_ROUTE_TRAFFIC`, and slots `PLACE_NAME: target` and `ABSOLUTE_LOCATION: bellevue`.

We are interested in addressing two types of data shift common in SLU applications. The first data shift problem happens when we transfer from synthetic data to live user data (a deployment shift). This is also known as the "cold-start" problem; a model cannot be trained on the real usage data prior to deployment simply because it does not exist. A common practice is to generate a large quantity of synthetic training data that mimics the expected user behavior. Such synthetic data is crafted using domain-specific knowledge and can be time-consuming. It is also flawed in that it typically does not match the live user data generated by actual users; the real queries submitted to these systems are different from what the model designers expect to see.

The second data shift problem happens when we transfer from stale data to current data (a temporal shift). In our use case, we have one set of training data from 2013 and wish to handle data from 2014–2016. This is problematic since the content of the user queries changes over time (e.g., new restaurant or movie names may be added). Consequently, the model performance degrades over time.

Both shifts cause a distribution mismatch between training and evaluation, leading to a model that overfits the flawed training data and performs poorly on the test data. We propose a solution to this mismatch problem by framing it as domain adaptation, treating the flawed training dataset as a source domain and the evaluation dataset as a target domain. To this end, we use and build on several recent advances in neural domain adaptation such as adversarial training (Ganin et al., 2016) and domain separation network (Bousmalis et al., 2016), proposing a new adversarial training scheme based on randomized predictions.

We consider both supervised and unsupervised adaptation scenarios (i.e., absence/presence of labeled data in the target domain). We find that unsupervised DA can greatly improve performance without requiring additional annotation. Super-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1297–1307
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1119

vised DA with a small amount of labeled data gives further improvement on top of unsupervised DA. In experiments, we show clear gains in both deployment and temporal shifts across 5 test domains, yielding average error reductions of 74.04% and 41.46% for intent classification and 70.33% and 32.0% for slot tagging compared to baselines without adaptation.

2 Related Work

2.1 Domain Adaptation

Our work builds on the recent success of DA in the neural network framework. Notably, Ganin et al. (2016) propose an adversarial training method for unsupervised DA. They partition the model parameters into two parts: one inducing domain-specific (or private) features and the other domain-invariant (or shared) features. The domain-invariant parameters are adversarially trained using a gradient reversal layer to be poor at domain classification; as a consequence, they produce representations that are domain agnostic. This approach is motivated by a rich literature on the theory of DA pioneered by Ben-David et al. (2007). We describe our use of adversarial training in Section 3.2.3. A special case of Ganin et al. (2016) is developed independently by Kim et al. (2016c) who motivate the method as a generalization of the feature augmentation method of Daumé III (2009).

Bousmalis et al. (2016) extend the framework of Ganin et al. (2016) by additionally encouraging the private and shared features to be mutually exclusive. This is achieved by minimizing the dot product between the two sets of parameters and simultaneously reconstructing the input (for all domains) from the features induced by these parameters.

Both Ganin et al. (2016) and Bousmalis et al. (2016) discuss applications in computer vision. Zhang et al. (2017) apply the method of Bousmalis et al. (2016) to tackle transfer learning in NLP. They focus on transfer learning between classification tasks over the same domain ("aspect transfer"). They assume a set of keywords associated with each aspect and use these keywords to inform the learner of the relevance of each sentence for that aspect.

2.2 Spoken Language Understanding

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

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

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

All the above works assume that there are no any data shift issues which our work try to solve.

3 Method

3.1 BiLSTM Encoder

We use an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes an input vector x and a state vector h to output a new state vector $h' = \phi(x, h)$. See Hochreiter and Schmidhuber (1997) for a detailed description.

Let $\mathcal{C}$ denote the set of character types and $\mathcal{W}$ the set of word types. Let $\oplus$ denote the vector concatenation operation. We encode an utterance using the wildly successful architecture given by bidirectional LSTMs (BiLSTMs) (Schuster and

Paliwal, 1997; Graves, 2012). The model parameters Θ associated with this BiLSTM layer are

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

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

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

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

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

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

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

for each $i = 1 \dots n$. For convenience, we write the entire operation as a mapping BiLSTM_Θ:

$$(h_1 \dots h_n) \leftarrow \text{BiLSTM}_\Theta(w_1 \dots w_n)$$

3.2 Unsupervised DA

In unsupervised domain adaptation, we assume labeled data for the source domain but not the target domain. Our approach closely follows the previous work on unsupervised neural domain adaptation by Ganin et al. (2016) and Bousmalis et al. (2016). We have three BiLSTM encoders described in Section 3.1:

1. Θ^{src}: induces source-specific features
2. Θ^{tgt}: induces target-specific features
3. Θ^{shd}: induces domain-invariant features

We now define a series of loss functions defined by these encoders.

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

3.2.1 Source Side Tagging Loss

The most obvious objective is to minimize the model's error on labeled training data for the source domain. Let $w_1 \dots w_n \in \mathcal{W}$ be an utterance in the source domain annotated with labels $y_1 \dots y_n \in \mathcal{L}$. We induce

$$\begin{aligned} (h^{\text{src}}_1 \dots h^{\text{src}}_n) &\leftarrow \text{BiLSTM}_{\Theta^{\text{src}}}(w_1 \dots w_n) \\ (h^{\text{shd}}_1 \dots h^{\text{shd}}_n) &\leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n) \end{aligned}$$

Then we define the probability of tag $y \in \mathcal{L}$ for the i-th word as

$$\begin{aligned} z_i &= W^2_{\text{tag}} \tanh\left(W^1_{\text{tag}} \bar{h}_i + b^1_{\text{tag}}\right) + b^2_{\text{tag}} \\ p(y|h_i) &\propto \exp\left([z_i]_y\right) \end{aligned}$$

where $\bar{h}_i = h^{src}_i \oplus h^{\text{shd}}_i$ and $\Theta^{\text{tag}} = \{W^1_{\text{tag}}, W^2_{\text{tag}}, b^1_{\text{tag}}, b^2_{\text{tag}}\}$ denotes additional feed-foward parameters. The tagging loss is given by the negative log likelihood

$$L^{\text{tag}}\left(\Theta^{\text{src}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) = -\sum_i \log p\left(y_i|\bar{h}_i\right)$$

where we iterate over annotated words (w_i, y_i) on the source side.

3.2.2 Reconstruction Loss

Following previous works, we ground feature learning by reconstructing encoded utterances. Both Bousmalis et al. (2016) and Zhang et al. (2017) use mean squared errors for reconstruction, the former of image pixels and the latter of words in a context window. In contrast, we use an attention-based LSTM that fully re-generates the input utterance and use its log loss.

More specifically, let $w_1 \dots w_n \in \mathcal{W}$ be an utterance in domain $d \in \{\text{src}, \text{tgt}\}$. We first use the relevant encoders as before

$$\begin{aligned} (h^d_1 \dots h^d_n) &\leftarrow \text{BiLSTM}_{\Theta^d}(w_1 \dots w_n) \\ (h^{\text{shd}}_1 \dots h^{\text{shd}}_n) &\leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n) \end{aligned}$$

The concatenated vectors $\bar{h}_i = h^d_i \oplus h^{\text{shd}}_i$ are fed into the standard attention-based decoder (Bahdanau et al., 2014) to define the probability of word w at each position i with state vector μ_{i-1} (where $\mu_0 = \bar{h}_n$):

$$\begin{aligned} \alpha_j &\propto \exp\left(\mu^\top_{i-1} \bar{h}_j\right) \quad \forall j \in \{1 \dots n\} \\ \tilde{h}_i &= \sum_{j=1}^{n} \alpha_j \bar{h}_j \\ \mu_i &= \phi^{\mathcal{R}}(\mu_{i-1} \oplus \tilde{h}_i, \mu_{i-1}) \\ p(w|\mu_i) &\propto \exp\left([W^1_{\text{rec}} \mu_i + b^1_{\text{rec}}]_w\right) \end{aligned}$$

where $\Theta^{\text{rec}} = \{\phi^{\mathcal{R}}, W^1_{\text{rec}}, b^1_{\text{rec}}\}$ denotes additional parameters. The reconstruction loss is given by the negative log likelihood

$$L^{\text{rec}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{rec}}\right) = -\sum_i \log p\left(w_i|\mu_i\right)$$

where we iterate over words w_i in both the source and target utterances.

3.2.3 Adversarial Domain Classification Loss

Ganin et al. (2016) propose introducing an adversarial loss to make shared features domain-invariant. This is motivated by a theoretical result of Ben-David et al. (2007) who show that the generalization error on the target domain depends on how "different" the source and the target domains are. This difference is approximately measured by

$$2\left(1 - 2\inf_{\Theta} \text{error}(\Theta)\right) \tag{2}$$

where $\text{error}(\Theta)$ is the domain classification error using model Θ. It is assumed that the source and target domains are balanced so that $\inf_\Theta \text{error}(\Theta) \leq 1/2$ and the difference lies in $[0, 2]$. In other words, we want to make $\text{error}(\Theta)$ as large as possible in order to generalize well to the target domain. The intuition is that the more domain-invariant our features are, the easier it is to benefit from the source side training when testing on the target side. It can also be motivated as a regularization term (Ganin et al., 2016).

Let $w_1 \dots w_n \in \mathcal{W}$ be an utterance in domain $d \in \{\text{src}, \text{tgt}\}$. We first use the shared encoder

$$(h^{\text{shd}}_1 \dots h^{\text{shd}}_n) \leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n)$$

It is important that we only use the shared encoder for this loss. Then we define the probability of domain d for the utterance as

$$z_i = W^2_{\text{adv}} \tanh\left(W^1_{\text{adv}} \sum_{i=1}^{n} h^{\text{shd}}_i + b^1_{\text{adv}}\right) + b^2_{\text{adv}}$$

$$p(d|h_i) \propto \exp\left([z_i]_d\right)$$

where $\Theta^{\text{adv}} = \{W^1_{\text{adv}}, W^2_{\text{adv}}, b^1_{\text{adv}}, b^2_{\text{adv}}\}$ denotes additional feedfoward parameters. The adversarial domain classification loss is given by the *positive* log likelihood

$$L^{\text{adv}}\left(\Theta^{\text{shd}}, \Theta^{\text{adv}}\right) = \sum_i \log p\left(d^{(i)}|w^{(i)}\right)$$

where we iterate over domain-annotated utterances $(w^{(i)}, d^{(i)})$.

Random prediction training While past work only consider using a negative gradient (Ganin et al., 2016; Bousmalis et al., 2016) or positive log likelihood (Zhang et al., 2017) to perform adversarial training, it is unclear whether these approaches are optimal for the purpose of "confusing" the domain predictor. For instance, minimizing log likelihood can lead to a model accurately predicting the *opposite* domain, compromising the goal of inducing domain-invariant representations. Thus we propose to instead optimize the shared parameters for *random domain predictions*. Specifically, the above loss is replaced with

$$L^{\text{adv}}\left(\Theta^{\text{shd}}, \Theta^{\text{adv}}\right) = -\sum_i \log p\left(d^{(i)}|w^{(i)}\right)$$

where $d^{(i)}$ is set to be `src` with probability 0.5 and `tgt` with probability 0.5. By optimizing for random predictions, we achieve the desired effect: the shared parameters are trained to induce features that cannot discriminate between the source and the target domains.

3.2.4 Non-Adversarial Domain Classification Loss

In addition to the adversarial loss for domain-invariant parameters, we also introduce a non-adversarial loss for domain-specific parameters. Given $w_1 \dots w_n \in \mathcal{W}$ in domain $d \in \{\text{src}, \text{tgt}\}$, we use the private encoder

$$(h^{\text{d}}_1 \dots h^{\text{d}}_n) \leftarrow \text{BiLSTM}_{\Theta^{\text{d}}}(w_1 \dots w_n)$$

It is important that we only use the private encoder for this loss. Then we define the probability of domain d for the utterance as

$$z_i = W^2_{\text{nadv}} \tanh\left(W^1_{\text{nadv}} \sum_{i=1}^{n} h^{\text{d}}_i + b^1_{\text{nadv}}\right) + b^2_{\text{nadv}}$$

$$p(d|h_i) \propto \exp\left([z_i]_d\right)$$

where $\Theta^{\text{nadv}} = \{W^1_{\text{nadv}}, W^2_{\text{nadv}}, b^1_{\text{nadv}}, b^2_{\text{nadv}}\}$ denotes additional feedfoward parameters. The non-adversarial domain classification loss is given by the negative log likelihood

$$L^{\text{nadv}}\left(\Theta^{\text{d}}, \Theta^{\text{nadv}}\right) = \sum_i \log p\left(d^{(i)}|w^{(i)}\right)$$

where we iterate over domain-annotated utterances $(w^{(i)}, d^{(i)})$.

3.2.5 Orthogonality Loss

Finally, following Bousmalis et al. (2016), we further encourage the domain-specific features to be mutually exclusive with the shared features by imposing soft orthogonality constraints. This is achieved as follows. Given an utterance $w_1 \dots w_n \in \mathcal{W}$ in domain $d \in \{\text{src}, \text{tgt}\}$. We compute

$$(h_1^d \dots h_n^d) \leftarrow \text{BiLSTM}_{\Theta^d}(w_1 \dots w_n)$$
$$(h_1^{\text{shd}} \dots h_n^{\text{shd}}) \leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n)$$

The orthogonality loss for this utterance is given by

$$L^{\text{orth}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}\right) = \sum_i (h_i^d)^\top h_i^{\text{shd}}$$

where we iterate over words i in both the source and target utterances.

3.2.6 Joint Objective

For unsupervised DA, we optimize

$$\begin{aligned} L^{\text{unsup}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}, \Theta^{\text{rec}}, \Theta^{\text{adv}}\right) = \\ L^{\text{tag}}\left(\Theta^{\text{src}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) + \\ L^{\text{rec}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{rec}}\right) + \\ L^{\text{adv}}\left(\Theta^{\text{shd}}, \Theta^{\text{adv}}\right) + \\ L^{\text{nadv}}\left(\Theta^{\text{src}}, \Theta^{\text{nadv}}\right) + \\ L^{\text{nadv}}\left(\Theta^{\text{tgt}}, \Theta^{\text{nadv}}\right) + \\ L^{\text{orth}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}\right) \end{aligned}$$

with respect to all model parameters. In an online setting, given an utterance we compute its reconstruction, adversarial, orthogonality, and tagging loss if in the source domain, and take a gradient step on the sum of these losses.

3.3 Supervised DA

In supervised domain adaptation, we assume labeled data for both the source domain and the target domain. We can easily incorporate supervision in the target domain by adding $L^{\text{tag}}\left(\Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right)$ to the unsupervised DA objective:

$$\begin{aligned} L^{\text{sup}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}, \Theta^{\text{rec}}, \Theta^{\text{adv}}\right) = \\ L^{\text{unsup}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}, \Theta^{\text{rec}}, \Theta^{\text{adv}}\right) + \\ L^{\text{tag}}\left(\Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) \end{aligned} \quad (3)$$

We mention that the approach by Kim et al. (2016c) is a special case of this objective; they optimize

$$\begin{aligned} L^{\text{sup2}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) = L^{\text{tag}}\left(\Theta^{\text{src}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) + \\ L^{\text{tag}}\left(\Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) \end{aligned} \quad (4)$$

which is motivated as a neural extension of the feature augmentation method of Daumé III (2009).

4 Experiments

In this section, we conducted a series of experiments to evaluate the proposed techniques on datasets obtained from real usage.

4.1 Test Domains and Tasks

We test our approach on a suite of 5 Microsoft Cortana domains with 2 separate tasks in spoken language understanding: (1) intent classification and (2) slot (label) tagging. The intent classification task is a multi-class classification problem with the goal of determining to which one of the n intents a user utterance belongs conditioning on the given domain. The slot tagging task is a sequence labeling problem with the goal of identifying entities and chunking of useful information snippets in a user utterance. For example, a user could say `reserve a table at joeys grill for thursday at seven pm for five people`. Then the goal of the first task would be to classify this utterance as `MAKE_RESERVATION` intent given the domain `PLACES`, and the goal of the second task would be to tag `joeys grill` as `RESTAURANT`, `thursday` as `DATE`, `seven pm` as `TIEM`, and `five` as `NUMBER_PEOPLE`.

Table 1 gives a summary of the 5 test domains. We note that the domains have various levels of label granularity.

Domain	Intent	Slot	Description
calendar	23	43	Set appointments in calendar
comm.	38	45	Make calls & send messages
places	35	64	Find locations & directions
reminder	14	35	Remind tasks in a to-do list
weather	13	19	Get weather information

Table 1: The number of intents, the number of slots and a short description of the test domains.

4.2 Experimental Setup

We consider 2 possible domain adaptation (DA) scenarios: (1) adaptation of an engineered dataset to a live user dataset and (2) adaptation of an old

dataset to a new dataset. For the first DA scenario, we test whether our approach can effectively make a system adapt from experimental, engineered data to real-world, live data. We use synthetic data which domain experts manually create based on a given domain schema[2] before the system goes live as the engineered data. We use transcribed dataset from users' speech input as the live user data. For the second scenario, we test whether our approach can effectively make a system adapt over time. A large number of users will quickly generate a large amount of data, and the usage pattern could also change. We use annotation data over 1 month in 2013 (more precisely August of 2013) as our old dataset, and use the whole data between 2014 and 2016 as our new dataset regardless of whether the data type is engineered or live user.

As we describe in the earlier sections, we consider both supervised and unsupervised DA. We apply our DA approach with labeled data in the target domain for the supervised setting and with unlabeled data for the unsupervised one. We give details of the baselines and variants of our approach below.

Unsupervised DA baselines and variants:

- *SRC*: a single LSTM model trained on a source domain without DA techniques
- DA^W: an unsupervised DA model with a word-level decoder (i.e., re-generate each word independently)
- DA^S: an unsupervised DA model with a sentence-level decoder described in Section 3.2

Supervised DA baselines and variants:

- *SRC*: a single LSTM model trained only on a source domain
- *TGT*: a single LSTM model trained only on a target domain
- *Union*: a single LSTM model trained on the union of source and target domains.
- DA: a supervised DA model described in Section 3.3
- DA^A: *DA* with adversary domain training
- DA^U: *DA* with reasonably sufficient unlabeled data

In our experiments, all the models were implemented using Dynet (Neubig et al., 2017) and were trained using Stochastic Gradient Descent (SGD) with Adam (Kingma and Ba, 2015)—an adaptive learning rate algorithm. We used the initial learning rate of 4×10^{-4} and left all the other hyper parameters as suggested in Kingma and Ba (2015). Each SGD update was computed without a minibatch with Intel MKL (Math Kernel Library)[3]. We used the dropout regularization (Srivastava et al., 2014) with the keep probability of 0.4.

We encode user utterances with BiLSTMs as described in Section 3.1. We initialize word embeddings with pre-trained embeddings used by Lample et al. (2016). In the following sections, we report intent classification results in accuracy percentage and slot results in F1-score. To compute slot F1-score, we used the standard CoNLL evaluation script[4]

4.3 Results: Unsupervised DA

We first show our results in the unsupervised DA setting where we have a labeled dataset in the source domain, but only unlabeled data in the target domain. We assume that the amount of data in both datasets is sufficient. Dataset statistics are shown in Table 2.

The performance of the baselines and our model variants are shown in Table 3. The left side of the table shows the results of the DA scenario of adapting from engineered data to live user data, and the baseline which trained only on the source domain (SRC) show a poor performance, yielding on average 48.5% on the intent classification and 42.7% F1-score on the slot tagging. Using our DA approach with a word-level decoder (DA^W) shows a significant increase in performance in all 5 test domains, yielding on average 82.2% intent accuracy and 80.5% slot F1-score. The performance increases further using the DA approach with a sentence-level decoder DA^S, yielding on average 85.6% intent accuracy and 83.0% slot F1-score.

The right side of the table shows the results of the DA scenario of adapting from old to new data, and the baseline trained only on SRC also show

[2]This is a semantic template that defines a set of intents and slots for each domain according to the intended functionality of the system.

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

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

	Engineered → Live User				Old → New			
Domain	Train	Train*	Dev	Test	Train	Train*	Dev	Test
calendar	16904	50000	1878	10k	13165	13165	1463	10k
communication	32072	50000	3564	10k	12631	12631	1403	10k
places	23410	50000	2341	10k	21901	21901	2433	10k
reminder	19328	50000	1933	10k	16245	16245	1805	10k
weather	20794	50000	2079	10k	15575	15575	1731	10k
AVG	23590	50000	2359	10k	15903	15903	1767	10k

Table 2: Data statistics for unsupervised domain adaptation; In the first row, the columns are adaptation of engineered dataset to live user dataset, and and adaptation of old dataset to new dataset. In the second row, columns are domain, size of labeled training, unlabeled training, development and test sets. * denotes unlabeled data

		Engineered → User Live			Old → New		
Task	Domain	SRC	DA^W	DA^S	SRC	DA^W	DA^S
Intent	calendar	47.5	82.0	**84.6**	50.7	85.7	**88.8**
	communication	45.8	75.3	**81.2**	49.4	83.2	**86.2**
	places	48.5	83.7	**86.3**	51.7	88.1	**91.1**
	reminder	50.7	83.9	**88.7**	53.3	88.8	**92.8**
	weather	50.3	86.3	**87.1**	53.4	89.1	**92.2**
	AVG	48.5	82.2	**85.6**	51.7	86.9	**90.2**
Slot	calendar	42.4	79.4	**81.7**	42.2	84.7	**87.9**
	communication	41.1	75.3	**79.1**	41.5	85.3	**89.1**
	places	40.2	81.6	**83.8**	44.1	85.4	**88.7**
	reminder	42.6	83.5	**85.7**	47.4	87.6	**91.2**
	weather	47.2	82.8	**84.7**	43.2	85.6	**89.5**
	AVG	42.7	80.5	**83.0**	43.7	85.7	**89.3**

Table 3: Intent classification accuracy (%) and slot tagging F1-score (%) for the unsupervised domain adaptation. The results that perform in each domain are in bold font.

	Engineered → Live User				Old → New			
Domain	Train	Train*	Dev	Test	Train	Train*	Dev	Test
calendar	16904	1000	100	10k	13165	1000	100	10k
communication	32072	1000	100	10k	12631	1000	100	10k
places	23410	1000	100	10k	21901	1000	100	10k
reminder	19328	1000	100	10k	16245	1000	100	10k
weather	20794	1000	100	10k	15575	1000	100	10k
AVG	23590	1000	100	10k	15903	1000	100	10k

Table 4: Data statistics for supervised domain adaptation

		Engineered → User Live						Old → New					
	Domain	SRC	TGT	Union	DA	DA^A	DA^U	SRC	TGT	Union	DA	DA^A	DA^U
I	calendar	47.5	69.2	48.3	80.7	80.5	**82.4**	50.7	69.2	49.9	74.4	75.4	**75.8**
	comm.	45.8	67.4	47.0	77.5	78.0	**79.7**	49.4	65.8	50.0	70.2	70.7	**71.9**
	places	48.5	71.2	48.5	82.0	82.4	**83.2**	51.7	69.6	52.2	75.8	76.4	**77.3**
	reminder	50.7	75.0	49.9	83.9	84.1	**87.3**	53.3	72.3	53.9	77.2	78.0	**78.5**
	weather	50.3	73.8	49.6	84.3	84.7	**85.6**	53.4	71.4	52.7	76.9	78.1	**79.2**
	AVG	48.5	71.3	48.7	81.7	81.9	**83.6**	51.7	69.7	51.7	74.9	75.7	**76.5**
S	calendar	42.4	64.9	43.0	76.1	76.7	**77.1**	42.2	61.8	41.6	68.0	66.9	**69.3**
	comm.	41.1	62.0	40.4	73.3	72.1	**73.8**	41.5	61.1	44.9	67.2	66.3	**68.4**
	places	40.2	61.8	39.0	72.1	72.0	**72.9**	44.1	64.6	47.7	70.1	68.7	**72.5**
	reminder	42.6	65.1	42.6	76.8	75.7	**80.0**	47.4	70.9	44.2	78.4	76.2	**78.9**
	weather	47.2	71.2	46.4	82.6	83.0	**84.4**	43.2	64.1	44.7	71.0	69.0	**70.2**
	AVG	42.7	65.0	42.3	76.2	75.9	**77.6**	43.7	64.5	44.6	71.0	69.4	**71.9**

Table 5: Intent classification accuracy (%) and slot tagging F1-score (%) for the supervised domain adaptation.

a similar poor performance, yielding on average 51.7% accuracy and 43.7% F1-score. DA^W approach shows a significant performance increase in all 5 test domains, yielding on average 86.9% intent accuracy and 85.7% slot F1-score. Similarly, the performance increases further with the DA^S with 90.2% intent accuracy and 89.3% F1-score.

Our DA approach variants yield average error reductions of 72.04% and 79.71% for intent classification and 70.33% and 80.99% for slot tagging. The results suggest that our DA approach can quickly make a model adapt from synthetic data to real-world data and from old data to new data with the additional use of only 2 to 2.5 more data from the target domain. Aside from the performance boost itself, the approach shows even more power since the new data from the target down do not need to be labeled and it only requires collecting a little more data from the target domain. We note that the model development sets were created only from the source domain for a fully unsupervised setting. But having the development set from the target domain shows even more boost in performance although not shown in the results, and labeling only the development set from the target domain is relatively less expensive than labeling the whole dataset.

4.4 Results: Supervised DA

Second, we show our results in the supervised DA setting where we have a sufficient amount of labeled data in the source domain but relatively insufficient amount of labeled data in the target domain. Having more labeled data in the target domain would most likely help with the performance, but we intentionally made the setting more disadvantageous for our DA approach to better simulate real-world scenarios where there is usually lack of resources and time to label a large amount of new data. For each personal assistant test domain, we only used 1000 training utterances to simulate scarcity of newly labeled data, and dataset statistics are shown in Table 2. Unlike the unsupervised DA scenario, here we used the development sets created from the target domain shown in Table 4.

The left side of Table 5 shows the results of the supervised DA approach of adapting from engineered data to live user data. The baseline trained only on the source (SRC) shows on average 48.5% intent accuracy and 42.7% slot F1-score. Training only on the target domain (TGT) increases the performance to 71.3% and 65.0%, but training on the union of the source and target domains (Union) again brings the performance down to 48.7% and 42.3%. As shown in the unsupervised setting, using our DA approach (DA) shows significant performance increase in all 5 test domains, yielding on average 81.7% intent accuracy and 76.2% slot tagging. The DA approach with adversary domain training (DA^A) shows similar performance compared to that of DA, and performance shows more increase when using our DA approach with sufficient unlabeled data[5] (DA^U), yielding on average 83.6% and 77.6%. For the second scenario of adapting from old to new dataset, the results show a very similar trend in performance.

The results show that our supervised DA (DA) approach also achieves a significant performance gain in all 5 test domains, yielding average error reductions of 68.18% and 51.35% for intent classification and 60.90% and 50.09% for slot tagging. The results suggest that an effective domain adaptation can be done using the supervised DA by having only a handful more data of 1k newly labeled data points. In addition, having both a small amount of newly labeled data combined with sufficient unlabeled data can help the models perform even better. The poor performance of using the union of both source and target domain data might be due to the relatively very small size of the target domain data, overwhelmed by the data in the source domain.

4.5 Results: Adversarial Domain Classification Loss

Task	Domain	Eng. → User RAND	Eng. → User ADV
Intent	calendar	**84.6**	81.1
	communication	**81.2**	77.9
	places	**86.3**	83.5
	reminder	**88.7**	85.8
	weather	**87.1**	84.2
	AVG	**85.6**	82.5
Slot	calendar	**81.7**	78.7
	communication	**79.1**	75.7
	places	**83.8**	80.6
	reminder	**85.7**	82.4
	weather	**84.7**	81.7
	AVG	**83.0**	79.8

Table 6: Intent classification accuracy (%) and slot tagging F1-score (%) for the unsupervised domain adaptation with two different adversarial classification losses – our claimed random domain predictions (RAND) and adversarial loss (ADVR) of Ganin et al. (2016) as explained in 3.2.3.

[5]This data is used for unsupervised DA experiments (Table 2).

The impact on the performance of two different adversarial classification losses are shown in Table 6. RAND represents the unsupervised DA model with sentence-level decoder (DA^S) using random prediction loss. The ADV shows the performance of same model using the adversarial loss of Ganin et al. (2016) as described in 3.2.3. Unfortunately, in the deployment shift scenario, using the adversarial loss fails to provide any improvement on intent classification accuracy and slot tagging F1 score, achieving 82.5% intent accuracy and 79.8% slot F1 score. These results align with our hypothesis that the adversarial loss using does not confuse the classifier sufficiently.

4.6 Proxy $\mathcal{A}$-distance

	Eng. → User	Old → New
Domain	$d_{\mathcal{A}}$	$d_{\mathcal{A}}$
calendar	0.58	0.43
comm.	0.54	0.44
places	0.68	0.62
reminder	0.54	0.57
weather	0.57	0.54
AVG	0.58	0.52

Table 7: Proxy $\mathcal{A}$-distance of resulting models: (1) engineered and live user dataset and (2) old and new dataset.

The results in shown in Table 7 show Proxy $\mathcal{A}$-distance(Ganin et al., 2016) to check if our adversary domain training generalize well to the target domain. The distance between two datasets is computed by

$$\hat{d}_A = 2(1 - 2\min\{\varepsilon, 1-\varepsilon\}) \quad (5)$$

where ε is a generalization error in discriminating between the source and target datasets.

The range of $\hat{d}_A$ distance is between 0 and 2.0. 0 is the best case where adversary training successfully fake shared encoder to predict domains. In other words, thanks to adversary training our model make the domain-invariant features in shared encoder in order to generalize well to the target domain.

4.7 Vocabulary distance between engineered data and live user data

The results in shown in Table 8 show the discrepancy between two datasets. We measure the degree of overlap between vocabulary V employed by the two datasets. We simply take the Jaccard coefficient between the two sets of such vocabulary:

$$d_V(s,t) = 1 - JC(V_s, V_t),$$

where V_s is the set of vocabulary in source s domain, and V_t is the corresponding set for target t domain and $JC(A,B) = \frac{|A \cap B|}{|A \cup B|}$ is the Jaccard coefficient, measuring the similarity of two sets. The distance d_V is the high it means that they are not shared with many words. Overall, the distance between old and new dataset are still far and the number of overlapped are small, but better than live user case.

	Eng. → User	Old → New
Domain	d_V	d_V
calendar	0.80	0.72
comm.	0.80	0.93
places	0.82	0.72
reminder	0.89	0.71
weather	0.72	0.73
AVG	0.80	0.76

Table 8: Distance between different datasets: (1) engineered and live user dataset and (2) old and new dataset.

5 Conclusion

In this paper, we have addressed two types of data shift common in SLU applications: 1. transferring from synthetic data to live user data (a deployment shift), and 2. transferring from stale data to current data (a temporal shift). Our method is based on domain adaptation, treating the flawed training dataset as a source domain and the evaluation dataset as a target domain. We use and build on several recent advances in neural domain adaptation such as adversarial training and domain separation network, proposing a new effective adversarial training scheme based on randomized predictions. In both supervised and unsupervised adaptation scenarios, our approach yields clear improvement over strong baselines.

References

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

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

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19:137.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

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

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

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

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

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

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

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35.

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

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

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

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

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

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

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

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

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Annual Meeting of the Association for Computational Linguistics*.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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).

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

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

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

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188* .