Efficient Large-Scale Neural Domain Classification with Personalized Attention

Young-Bum Kim **Dongchan Kim** **Anjishnu Kumar** **Ruhi Sarikaya**
Amazon Alexa
{youngbum,dongchan,anjikum,rsarikaya}@amazon.com

Abstract

In this paper, we explore the task of mapping spoken language utterances to one of thousands of natural language understanding domains in intelligent personal digital assistants (IPDAs). This scenario is observed in mainstream IPDAs in industry that allow third parties to develop thousands of new domains to augment built-in first party domains to rapidly increase domain coverage and overall IPDA capabilities. We propose a scalable neural model architecture with a shared encoder, a novel attention mechanism that incorporates personalization information and domain-specific classifiers that solves the problem efficiently. Our architecture is designed to efficiently accommodate incremental domain additions achieving two orders of magnitude speed up compared to full model retraining. We consider the practical constraints of real-time production systems, and design to minimize memory footprint and runtime latency. We demonstrate that incorporating personalization significantly improves domain classification accuracy in a setting with thousands of overlapping domains.

1 Introduction

Intelligent personal digital assistants (IPDAs) are one of the most advanced and successful artificial intelligence applications that have spoken language understanding (SLU). Many IPDAs have recently emerged in industry including Amazon Alexa, Google Assistant, Apple Siri, and Microsoft Cortana (Sarikaya, 2017). IPDAs have traditionally supported only dozens of well-separated domains, each defined in terms of a specific application or functionality such as calendar and local search (Tur and de Mori, 2011; Sarikaya et al., 2016). To rapidly increase domain coverage and extend capabilities, some IPDAs have released Software Development Toolkits (SDKs) to allow third-party developers to quickly build and integrate new domains, which we refer to as *skills* henceforth. Amazon's *Alexa Skills Kit* (Kumar et al., 2017a), Google's *Actions* and Microsoft's *Cortana Skills Kit* are all examples of such SDKs. Alexa Skills Kit is the largest of these services with over 40,000 skills.

For IPDAs, finding the most relevant skill to handle an utterance is an open problem for three reasons. First, the sheer number of skills makes the task difficult. Unlike traditional systems that have on the order of 10-20 built-in domains, large-scale IPDAs can have up to 40,000 skills. Second, the number of skills is rapidly expanding with 100+ new skills added per week. Large-scale IPDAs should be able to accommodate new skills efficiently without compromising performance. Third, unlike traditional built-in domains that are carefully designed to be disjoint by a central team, skills are built independently by different developers and can cover overlapping functionalities. For instance, there are over 50 recipe skills in Alexa that can handle recipe-related utterances.

One simple solution to this problem has been to require the user to explicitly mention a skill name and follow a strict invocation pattern as in *"Ask {Uber} to {get me a ride}."* However, this significantly limits the natural interaction with IPDAs. Users have to remember skill names and invocation patterns, and it places a cognitive burden on users who tend to forget both. Skill discovery is difficult with a pure voice user interface, it is hard for users to know the capabilities of thousands of skills a priori, which may leads to limited user en-

gagement with skills and potentially with IPDAs.

In this paper, we propose a solution that addresses all three practical challenges without requiring skill names or invocation patterns. Our approach is based on a scalable neural model architecture with a shared encoder, a skill attention mechanism and skill-specific classification networks that can efficiently perform large-scale skill classification in IPDAs using a weakly supervised training dataset. We demonstrate that our model achieves a high accuracy on a manually transcribed test set after being trained with weak supervision. Moreover, our architecture is designed to efficiently integrate new skills that appear in-between full model retraining cycles into the model. Besides accuracy, we also keep practical constraints in mind and focus on minimizing memory footprint and runtime latency, while ensuring architecture is scalable to thousands of skills, all of which are important for real-time production systems. Furthermore, we investigate two different ways of incorporating user personalization information into the model, our naive baseline method adds the information as a 1-bit flag in the feature space of the skill-specific networks, the *personalized attention* technique computes a convex combination of skill embeddings for the user's enabled skills and significantly outperforms the naive personalization baseline. We show the effectiveness of our approach with extensive experiments using 1,500 skills from a deployed IPDA system.

2 Related Work

Traditional multi-domain SLU/NLU systems are designed hierarchically, starting with domain classification to classify an incoming utterance into one of many possible domains, followed by further semantic analysis with domain-specific intent classification and slot tagging (Tur and de Mori, 2011). Traditional systems have typically been limited to a small number of domains, designed by specialists to be well-separable. Therefore, domain classification has been considered a less complex task compared to other semantic analysis such as intent and slot predictions. Traditional domain classifiers are built using simple linear models such as Multinomial Logistic Regression or Support Vector Machines in a one-versus-all setting for multi-class prediction. The models typically use word n-gram features and also those based on static lexicon match, and there have been several recent studies applying deep learning techniques (Xu and Sarikaya, 2014).

There is also a line of prior work on enhancing sequential text classification or tagging. Hierarchical character-to-word level LSTM (Hochreiter and Schmidhuber, 1997) architectures similar to our models have been explored for the Named Entity Recognition task by Lample et al. (2016). Character-informed sequence models have also been explored for simple text classification with small sets of classes by Xiao and Cho (2016). Joulin et al. (2016) explored highly scalable text classification using a shared hierarchical encoder, but their hierarchical softmax-based output formulation is unsuitable for incremental model updates. Work on zero-shot domain classifier expansion by Kumar et al. (2017b) struggled to rank incoming domains higher than training domains. The attention-based approach of Kim et al. (2017d) does not require retraining from scratch, but it requires keeping all models stored in memory which is computationally expensive. Multi-Task learning was used in the context of SLU by Tur (2006) and has been further explored using neural networks for phoneme recognition (Seltzer and Droppo, 2013) and semantic parsing (Fan et al., 2017; Bapna et al., 2017). There have been many other pieces of prior work on improving NLU systems with pre-training (Kim et al., 2015b; Celikyilmaz et al., 2016; Kim et al., 2017e), multi-task learning (Zhang and Wang, 2016; Liu and Lane, 2016; Kim et al., 2017b), transfer learning (El-Kahky et al., 2014; Kim et al., 2015a,c; Chen et al., 2016a; Yang et al., 2017), domain adaptation (Kim et al., 2016; Jaech et al., 2016; Liu and Lane, 2017; Kim et al., 2017d,c) and contextual signals (Bhargava et al., 2013; Chen et al., 2016b; Hori et al., 2016; Kim et al., 2017a).

3 Weakly Supervised Training Data Generation

Our model addresses the domain classification task in SLU systems. In traditional IPDA systems, these domains are hand-crafted by experts to be well separable and can easily be annotated by humans because they are small in number. The emergence of self-service SLU results in a large number of potentially mutually overlapping SLU domains. This means that eliciting large volumes of high quality human annotations to train our model

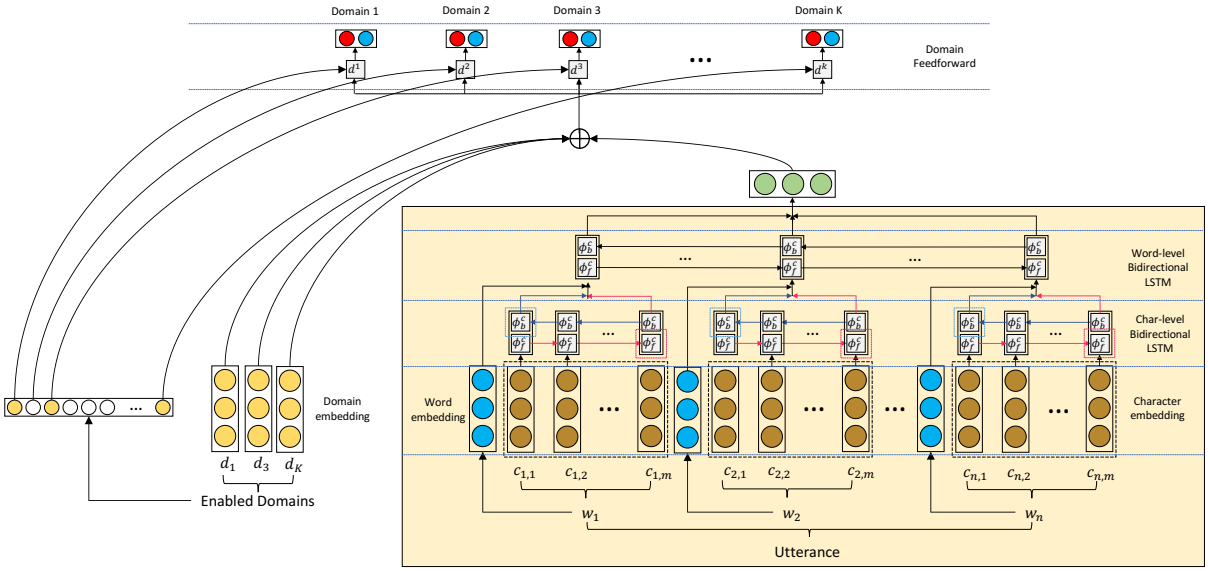

Figure 1: The overall architecture of the personalized dynamic domain classifier.

is no longer feasible, and we cannot assume that domains are designed to be well separable.

Instead we can generate training data by adopting the weak supervision paradigm introduced by (Hoffmann et al., 2011), which proposes using heuristic labeling functions generate large numbers of noisy data samples. Clean data generation with weak supervision is a challenging problem, so we address it by decomposing it into two simpler problems, of candidate generation and noise suppression, however it remains important for our model to be noise robust.

3.1 Data Programming

The key insight of the *Data Programming* approach is that $O(1)$ simple labeling functions can be used to approximate $O(n)$ human annotated data points with much less effort. We adopt the formalism used by (Ratner et al., 2016) to treat each of instance data generation rule as a rich generative model, defined by a labeling function λ and describe different families of labeling functions. Our data programming pipeline is analogous to the noisy channel model proposed for spelling correction by (Kernighan et al., 1990), and consists of a set of candidate generation and noise detection functions.

$$\arg\max_{\mu} P(\mu|s_i) = \arg\max_{\mu} P(s_i|\mu).P(\mu)$$

where μ and s_i represent utterances and the ith skill respectively. $P(s_i|\mu)$ the probability of a skill being valid for an utterance is approximated by simple functions that act as candidate data *generators* $\lambda_g \in \Lambda_g$ based on recognitions produced by a family of *query patterns* $\lambda_q \in \Lambda_q$. $P(\mu)$ is represented by a family of simple functions that act as *noise detectors* $\lambda_n \in \Lambda_n$, which mark utterances as likely being noise.

We apply the technique to the query logs of a popular IPDA, which has support for personalized third party domains. Looking at the structure of utterances that match query pattern λ_q, each utterance of form *"Ask* $\{Uber\}$ *to* $\{get\ me\ a\ car\}$*"* can be considered as being parametrized by the underlying latent command μ_z, that is *"Get me a car"*, a target domain corresponding to service s_t, which in this case is *Uber* and the query recognition pattern λ_q, in this case *"Ask* $\{s_t\}$ *to* $\{\mu_z\}$*"*. Next we assume that the distribution of latent commands over domains are independent of the query pattern.

$$P(\mu_z, s_t) \approx P(\mu, s_t, \lambda_q)$$

Making this simple distributional approximation allows us to generate a large number of noisy training samples. The family of generator functions $\lambda_g \in \Lambda_g$ is thus defined such that $u_z = \lambda_g^i(\mu, \lambda_q^i)$

3.2 Noise Reduction

The distribution defined above contains a large number of noisy positive samples. Related to $P(\mu)$ in the noisy channel in the spell correction context, we defined a small family of heuristic noise detection functions $\lambda_n \in \Lambda_n$ that discards

training data instances that are not likely to be well formed. For instance,

- λ_n^1 requires u to contain a minimum threshold of information by removing those with μ_z that has token length fewer than 3. Utterances shorter than this mostly consist of non-actionable commands.
- λ_n^2 discards all data samples below a certain threshold of occurrences in live traffic, since utterances that are rarely observed are more likely to be ASR errors or unnatural.
- λ_n^3 discards the data samples for a domain if they come from an overly broad pattern with a catch-all behavior.
- λ_n^4 discards utterances that belong to shared intents provided by the SLU SDK.

The end result of this stage is to retain utterances such as 'call me a cab' from 'Ask Uber to call me a cab' but discard 'Boston' from 'Ask Accuweather for Boston'. One can easily imagine extending this framework with other high recall noise detectors, for example, using language models to discard candidates that are unlikely to be spoken sentences.

4 Model Architecture

Our model consists of a *shared encoder* network consisting of an orthography-sensitive hierarchical LSTM encoder that feeds into a set of domain specific classification layers trained to make a binary decision for each output label.

Our main novel contribution is the extension of this architecture with a *personalized attention* mechanism which uses the attention mechanism of (Bahdanau et al., 2014) to attend to memory locations corresponding to the specific domains enabled by a user, and allows the system to learn semantic representations of each domain via *domain embeddings*. As we will show, incorporating personalization features is key to disambiguating between multiple overlapping domains[1], and the personalized attention mechanism outperforms more naive forms of personalization. The personalized attention mechanism first computes an attention weight for each of enabled domains, performs a convex combination to compute a context vector and then concatenates this vector to the encoded utterance before the final domain classification. Figure 1 depicts the model in detail.

Our model can efficiently accommodate new domains not seen during initial training by keeping the shared encoder frozen, bootstrapping a domain embedding based on existing ones, then optimizing a small number of network parameters corresponding to domain-specific classifier, which is orders of magnitude faster and more data efficient than retraining the full classifier.

We make design decisions to ensure that our model has a low memory and latency footprint. We avoid expensive large vocabulary matrix multiplications on both the input and output stages, and instead use a combination of character embeddings and word embeddings in the input stage.[2] The output matrix is lightweight because each domain-specific classifier is a matrix of only 201×2 parameters. The inference task can be trivially parallelized across cores since there's no requirement to compute a partition function across a high-dimensional softmax layer, which is the slowest component of large label multiclass neural networks. Instead, we achieve comparability between the probability scores generated by individual models by using a customized loss formulation.[3]

4.1 Shared Encoder

First we describe our shared hierarchical utterance encoder, which is marked by the almond colored box in Figure 1. Our hierarchical character to word to utterance design is motivated by the need to make the model operate on an open vocabulary in terms of words and to make it robust to small changes in orthography resulting from fluctuations in the upstream ASR system, all while avoiding expensive large matrix multiplications associated with one-hot word encoding in large vocabulary systems.

We denote an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes a d dimensional input vector x and a d' dimensional state vector h to output a new d' dimensional state vector $h' =$

[1] We assume that users can customize their IPDA settings to enable certain domains.

[2] Using a one-hot representation of word vocabulary size 60,000 and hidden dimension 100 would require learning a matrix of size 60000 x 100 - using 100-dim word embeddings requires only a $\mathcal{O}(1)$ lookup followed by a 100 x 100 matrix, thus allowing our model to be significantly smaller and faster despite having what is effectively an open vocabulary

[3] Current inference consumes 50MB memory and the p99 latency is 15ms.

$\phi(x, h)$. Let $\mathcal{C}$ denote the set of characters and $\mathcal{W}$ the set of words in a given utterance. Let $\oplus$ denote the vector concatenation operation. We encode an utterance using BiLSTMs, and the model parameters Θ associated with this BiLSTM layer are

- Char embeddings $e_c \in \mathbb{R}^{25}$ for each $c \in \mathcal{C}$
- Char LSTMs $\phi_f^{\mathcal{C}}, \phi_b^{\mathcal{C}} : \mathbb{R}^{25} \times \mathbb{R}^{25} \rightarrow \mathbb{R}^{25}$
- Word embeddings $e_w \in \mathbb{R}^{50}$ for each $w \in \mathcal{W}$
- Word LSTMs $\phi_f^{\mathcal{W}}, \phi_b^{\mathcal{W}} : \mathbb{R}^{100} \times \mathbb{R}^{50} \rightarrow \mathbb{R}^{50}$

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 $v_i \in \mathbb{R}^{100}$ as

$$\begin{aligned} f_j^{\mathcal{C}} &= \phi_f^{\mathcal{C}}\left(e_{w_i(j)}, f_{j-1}^{\mathcal{C}}\right) && \forall j = 1 \dots |w_i| \\ b_j^{\mathcal{C}} &= \phi_b^{\mathcal{C}}\left(e_{w_i(j)}, b_{j+1}^{\mathcal{C}}\right) && \forall j = |w_i| \dots 1 \\ v_i &= f_{|w_i|}^{\mathcal{C}} \oplus b_1^{\mathcal{C}} \oplus e_{w_i} \end{aligned}$$

for each $i = 1 \dots n$.[4] These word representation vectors are encoded by forward and backward LSTMs for word $\phi_f^{\mathcal{W}}, \phi_b^{\mathcal{W}}$ as

$$\begin{aligned} f_i^{\mathcal{W}} &= \phi_f^{\mathcal{W}}\left(v_i, f_{i-1}^{\mathcal{W}}\right) && \forall i = 1 \dots n \\ b_i^{\mathcal{W}} &= \phi_b^{\mathcal{W}}\left(v_i, b_{i+1}^{\mathcal{W}}\right) && \forall i = n \dots 1 \end{aligned}$$

and induces a character and context-sensitive word representation $h_i \in \mathbb{R}^{100}$ as

$$h_i = f_i^{\mathcal{W}} \oplus b_i^{\mathcal{W}}$$

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

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

$$\bar{h} = \sum_{i=1}^{n} h_i \tag{1}$$

4.2 Domain Classification

Our Multitask domain classification formulation is motivated by a desire to avoid computing the full partition function during test time, which tends to be the slowest component of a multiclass neural network classifer, as has been documented before by (Jozefowicz et al., 2016) and (Mikolov et al., 2013), amongst others.

[4]For simplicity, we assume some random initial state vectors such as $f_0^{\mathcal{C}}$ and $b_{|w_i|+1}^{\mathcal{C}}$ when we describe LSTMs.

However, we also want access to reliable probability estimates instead of raw scores - we accomplish this by constructing a custom loss function. During training, each domain classifier receives in-domain (IND) and out-of-domain (OOD) utterances, and we adapt the one-sided selection mechanism of (Kubat et al., 1997) to prevent OOD utterances from overpowering IND utterances, thus an utterance in a domain $d \in \mathcal{D}$ is considered as an IND utterance in the viewpoint of domain d and OOD for all other domains.

We first use the shared encoder to compute the utterance representation $\bar{h}$ as previously described. Then we define the probability of domain $\tilde{d}$ for the utterance by mapping $\bar{h}$ to a 2-dimensional output vector with a linear transformation for each domain $\tilde{d}$ as

$$z^{\tilde{d}} = \sigma(W^{\tilde{d}} \cdot \bar{h} + b^{\tilde{d}})$$

$$p(\tilde{d}|\bar{h}) \propto \begin{cases} \exp\left([z^{\tilde{d}}]_{IND}\right), & \text{if } \tilde{d} = d \\ \exp\left([z^{\tilde{d}}]_{OOD}\right), & \text{otherwise} \end{cases}$$

where σ is scaled exponential linear unit (SeLU) for normalized activation outputs (Klambauer et al., 2017) and $[z^{\tilde{d}}]_{IND}$ and $[z^{\tilde{d}}]_{OOD}$ denote the values in the IND and OOD position of vector $z^{\tilde{d}}$.

We define the joint domain classification loss $\mathcal{L}^{\mathcal{D}}$ as the summation of positive ($\mathcal{L}^P$) and negative ($\mathcal{L}^N$) class loss functions [5]:

$$\mathcal{L}^P\left(\Theta, \Theta^{\tilde{d}}\right) = -\log p\left(\tilde{d}|\bar{h}\right)$$

$$\mathcal{L}^N\left(\Theta, \Theta^{\tilde{d}}\right) = -\frac{1}{k-1}\left(\sum_{\bar{d} \in \mathcal{D}, \bar{d} \neq \tilde{d}} \log p\left(\bar{d}|\bar{h}\right)\right)$$

$$\mathcal{L}^{\mathcal{D}}\left(\Theta, \Theta^{\tilde{d}}\right) = L^P\left(\Theta, \Theta^{\tilde{d}}\right) + L^N\left(\Theta, \Theta^{\tilde{d}}\right)$$

where k is the total number of domains. We divide the second term by $k-1$ so that $\mathcal{L}^P$ and $\mathcal{L}^N$ are balanced in terms of the ratio of the training examples for a domain to those for other domains. While a softmax over the entire domains tends to highlight only the ground-truth domain while suppressing all the rest, the our joint domain classification with a softmax over two classes is designed to produce a more balanced confidence score per domain independent of other domains.

[5]$\Theta^{\tilde{d}}$ denotes the additional parameters in the classification layer for domain $\tilde{d}$.

4.3 Personalized Attention

We explore encoding a user's domain preferences in two ways. Our baseline method is a *1-bit flag* that is appended to the input features of each domain-specific classifier. Our novel *personalized attention* method induces domain embeddings by supervising an attention mechanism to attend to a user's enabled domains with different weights depending on their relevance. The domain embedding matrix in Figure 1 represents the embeddings of a user's enabled domains. We hypothesize that attention enables the network learn richer representations of user preferences and domain co-occurrence features.

Let $e_{\mathcal{D}}(\tilde{d}) \in R^{100}$ and $\bar{h} \in R^{100}$ denote the domain embeddings for domain $\tilde{d}$ and the utterance representation calculated by Eq. (1), respectively. The domain attention weights for a given user u who has a preferred domain list $d^{(u)} = \left(\tilde{d}_1^{(u)}, \ldots, \tilde{d}_k^{(u)}\right)$ are calculated by the dot-product operation,

$$a_i = \bar{h} \cdot e_{\mathcal{D}}\left(\tilde{d}_i^{(u)}\right) \qquad \forall i = 1 \ldots k$$

The final, normalized attention weights $\bar{a}$ are obtained after normalization via a softmax layer,

$$\bar{a}_i = \frac{\exp(a_i)}{\sum_{j=1}^{k} \exp(a_j)} \qquad \forall i = 1 \ldots k$$

The weighted combination of domain embeddings is

$$\bar{S}^{attended} = \sum_{i=1}^{k} \left(\bar{a}_i \cdot e_{\mathcal{D}}\left(\tilde{d}_i^{(u)}\right)\right)$$

Finally the two representations of enabled domains, namely the *attention* model and *1-bit flag* are then concatenated with the utterance representation and used to make per-domain predictions via domain-specific affine transformations:

$$\bar{z}_{att} = \bar{h} \oplus \bar{S}^{attended}$$
$$\bar{z}_{1bit} = \bar{h} \oplus \mathbb{I}(\tilde{d} \in enabled)$$

Here $\mathbb{I}(\tilde{d} \in enabled)$ is a 1-bit indicator for whether the domain is enabled by the user or not. $\bar{z}_{att}$ and $\bar{z}_{1bit}$ represent the encoded hidden state of the `Attention` and `1-Bit Flag` configurations of the model from the experiment section. In our experiments we will compare these two ways of encoding personalization information, as well as evaluate a variant that combines the two. In this way we can ascertain whether the two personalization signals are complementary via an ablation study on the full model.

4.4 Domain Bootstrapping

Our model separates the responsibilities for utterance representation and domain classification between the shared encoder and the domain-specific classifiers. That is, the shared encoder needs to be retrained only if it cannot encode an utterance well (e.g., a new domain introduces completely new words) and the existing domain classifiers need to be retrained only when the shared encoder changes. For adding new domains efficiently without full retraining, the only two components in the architecture need to be updated for each new domain $\tilde{d}_{new}$, are the domain embeddings for the new domain and its domain-specific classifier.[6] We keep the weights of the encoder network frozen and use the hidden state vector $\bar{h}$, calculated by Eq. 1, as a feature vector to feed into the downstream classifiers. To initialize the m-dimensional domain embeddings $e_{\tilde{d}_{new}}$, we use the column-wise average of all utterance vectors in the training data $\bar{h}^{avg}$, and project it to the domain embedding space using a matrix $U \in R^{m \times m}$. Thus,

$$e_{\tilde{d}_{new}} = U^* \cdot \bar{h}^{avg}$$

The parameters of U^* are learned using the column-wise average utterance vectors $\bar{h}_j^{avg}$ and learned domain vectors for all existing domains d_j

$$U^* = \arg\min_{U} ||U \cdot \bar{h}_j^{avg} - e_{d_j}|| \quad \forall d_j \in \mathcal{D}$$

This is a write-to-memory operation that creates a new domain representation after attending to all existing domain representations. We then train the parameters of the domain-specific classifier with the new domain's data while keeping the encoder fixed. This mechanism allows us to efficiently support new domains that appear in-between full model deployment cycles without compromising performance on existing domains. A full model refresh would require us to fully retrain with the domains that have appeared in the intermediate period.

[6] We have assumed that the shared encoder covers most of the vocabulary of new domains; otherwise, the entire network may need to be retrained. Based on our observation of live usage data, this assumption is reasonable since the shared encoder after initial training is still shown to cover 95% of the vocabulary of new domains added in the subsequent week.

	WEAK			Mturk		
	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
Binary	78.29	87.90	88.92	73.79	85.35	86.45
MultiClass	78.58	87.12	88.11	73.78	84.54	85.55
MultiTask	80.46	89.27	90.16	75.66	86.48	87.66
1-Bit Flag	91.97	95.89	96.68	86.50	92.47	93.09
Attention*	94.83	97.11	98.35	89.64	95.39	96.70
1-Bit + Att	**95.19**	**97.32**	**98.64**	**89.65**	**95.79**	**96.98**

Table 1: The performance of different variants of our neural model in terms of top-N accuracy. `Binary` trains a separate binary classifier for each skill. `MultiClass` has a shared encoder followed by a softmax. `MultiTask` replaces the softmax with per-skill classifiers. `1-Bit Flag` adds a single bit for personalization to each skill classifier in MultiTask. `Attention` extends MultiTask with personalized attention. The last 3 models are personalized. **Best single encoding.*

5 Experiments

In this section we aim to demonstrate the effectiveness of our model architecture in two settings. First, we will demonstrate that attention based personalization significantly outperforms the baseline approach. Secondly, we will show that our model new domain bootstrapping procedure results in accuracies comparable to full retraining while requiring less than 1% of the orignal training time.

5.1 Experimental Setup

Weak: This is a weakly supervised dataset was generated by preprocessing utterances with strict invocation patterns according to the setup mentioned in Section 3. The dataset consists of 5.34M utterances from 637,975 users across 1,500 different skills. Since we are interested in capturing the temporal effects of the dataset as well as personalization effects, we partitioned the data based both on user and time. Our core training data for the experiments in this paper was drawn from one month of live usage, the validation data came from the next 15 days of usage, and the test data came from the subsequent 15 days. The training, validation and test sets are user-independent, and each user belongs to only one of the three sets to ensure no leakage of information.

MTurk: Since the Weak dataset is generated by weak supervision, we verified the performance of our approach with human generated utterances. A random sample of 12,428 utterances from the test partition of users were presented to 300 human judges, who were asked to produce two natural ways to issue the same command. This dataset is treated as a representative clean held out test set on which we can observe the generalization of our weakly supervised training and validation data to natural language.

New Skills: In order to simulate the scenario in which new skills appear within a week between model updates, we selected 250 new skills which do not overlap with the skills in the Weak dataset. The vocabulary size of 1,500 skills is 200K words, and on average, 5% of the vocabulary for new skills is not covered. We randomly sampled 4,000 unique utterances for each skill using the same weak supervision method, and split them into 3,000 utterances for training and 1,000 for testing.

5.2 Results and Discussion

Generalization from Weakly Supervised to Natural Utterances We first show the progression of model performance as we add more components to show their individual contribution. Secondly, we show that training our models on a weakly supervised dataset can generalize to natural speech by showing their test performance on the human-annotated test data. Finally, we compare two personalization strategies.

The full results are summarized in Table 1, which shows the top-N test results separately for the Weak dataset (weakly supervised) and MTurk dataset (human-annotated). We report top-N accuracy to show the potential for further re-ranking or disambiguation downstream. For top-1 results on the Weak dataset, using a separate binary classifier for each domain (Binary) shows a prediction accuracy at 78.29% and using a softmax layer on top of the shared encoder (MultiClass) shows a comparable accuracy at 78.58%. The performance shows a slight improvement when using the Multitask neural loss structure, but adding personalization signals to the Multitask structure showed a significant boost in performance. We noted the large difference between the 1-bit and attention architecture. At 94.83% accuracy, attention resulted in 35.6% relative error reduction over the 1-bit baseline 91.97% on the Weak validation set and 23.25% relative on the MTurk test set. We hypothesize that this may be because the attention mechanism allows the model to focus on complementary features in case of overlapping domains as well as learning domain co-occurrence statistics, both of which are not possible with the simple 1-bit flag.

When both personalization representations were combined, the performance peaked at 95.19% for the Weak dataset and a more modest

	Time	Accuracy
Binary	34.81	78.13
Expand	30.34	94.03
Refresh	5300.18	94.58

Table 2: Comparison of per-epoch training time (seconds) and top-1 accuracy (%) on an NVIDIA Tesla M40 GPU.

89.65% for the MTurk dataset. The improvement trend is extremely consistent across all top-N results for both of the Weak and MTurk datasets across all experiments. The disambiguation task is complex due to similar and overlapping skills, but the results suggest that incorporating personalization signals equip the models with much better discriminative power. The results also suggest that the two mechanisms for encoding personalization provide a small amount of complementary information since combining them together is better than using them individually. Although the performance on the Weak dataset tends to be more optimistic, the best performance on the human-annotated test data is still close to 90% for top-1 accuracy, which suggests that training our model with the samples derived from the invocation patterns can generalize well to natural utterances.

Rapid Bootstrapping of New Skills We show the results when new domains are added to an IPDA and the model needs to efficiently accommodate them with a limited number of data samples. We show the classification performance on the skills in the New Skills dataset while assuming we have access to the WEAK dataset to pre-train our model for transfer learning. In the `Binary` setting, a domain-specific binary classifier is trained for each domain. `Expand` describes the case in which we incrementally train on top of an existing model. `Refresh` is the setting in which the model is fully retrained from scratch with the new data - which would be ideal in case there were no time constraints.

We record the average training time for each epoch and the performance is measured with top-1 classification accuracy over new skills. The experiment results can be found in Table 2. Adapting a new skill is two orders of magnitude faster (30.34 seconds) than retraining the model (5300.18 seconds) while achieving 94.03% accuracy which is comparable to 94.58% accuracy of full retraining. The first two techniques can also be easily parallelized unlike the `Refresh` configuration.

	Top-1	Top-3	Top-5
Full	6.17	14.30	20.41
Enabled	85.62	96.15	98.06

Table 3: Top-N prediction accuracy (%) on the full skill set (Full) and only enabled skills (Enabled).

Behavior of Attention Mechanism Our expectation is that the model is able to learn to attend the relevant skills during the inference process. To study the behavior of the attention layer, we compute the top-N prediction accuracy based on the most relevant skills defined by the attention weights. In this experiment, we considered the subset of users who had enabled more than 20 domains to exclude trivial cases[7]. The results are shown in Table 3. When the model attends to the entire set of 1500 (*Full*), the top-5 prediction accuracy is 20.41%, which indicates that a large number of skills can process the utterance, and thus it is highly likely to miss the correct one in the top-5 predictions. This ambiguity issue can be significantly improved by users' enabled domain lists as proved by the accuracies (*Enabled*): 85.62% for top-1, 96.15% for top-3, and 98.06% for top-5.[8] Thus the attention mechanism can thus be viewed as an initial soft selection which is then followed by a fine-grained selection at the classification stage.

End-to-End User Evaluation All intermediate metrics on this task are proxies to a human customer's eventual evaluation. In order to assess the user experience, we need to measure its end-to-end performance. For a brief end-to-end system evaluation, 983 utterances from 283 domains were randomly sampled from the test set in the large-scale IPDA setting. 15 human judges (male=12, female=3) rated the system responses, 1 judge per utterance, on a 5-point Likert scale with 1 being *Terrible* and 5 being *Perfect*. The judgment score of 3 or above was taken as *SUCCESS* and 2 or below as *DEFECT*. The end-to-end *SUCCESS* rate,

[7]Thus, the random prediction accuracy on enabled domains is less than 5% and across the Full domain list is 0.066%

[8]Visual inspection of the embeddings confirms that meaningful clusters are learned. We see clusters related to home automation, commerce, cooking, trivia etc, we show some examples in Figure 2, 3 and 4. However there are still other clusters where the the relationships cannot be established as easily. An example of these is show in Figure 5. The personalized attention mechanism is learned using the semantic content as well as personalization signals, so we hypothesize clusters like this may be capturing user tendencies to enable these domains in a correlated manner.

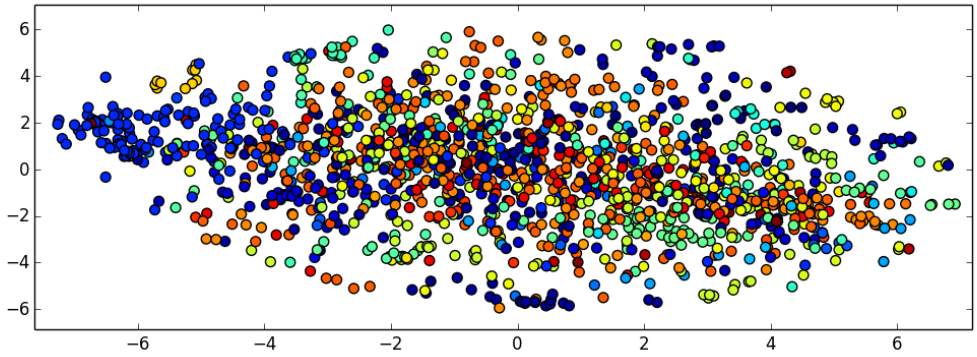

Figure 2: Embeddings of different domain categories visualized in 2D using TSNE (van der Maaten and Hinton, 2008). Different colors represent different categories, for e.g. the large blue cluster on the left is Home Automation.

thus user satisfaction, was shown to be 95.52%. The discrepancy between this score and the score produced on MTurk dataset indicates that even in cases in which the model makes classification mistakes, some of these interpretations remain perceptually meaningful to humans.

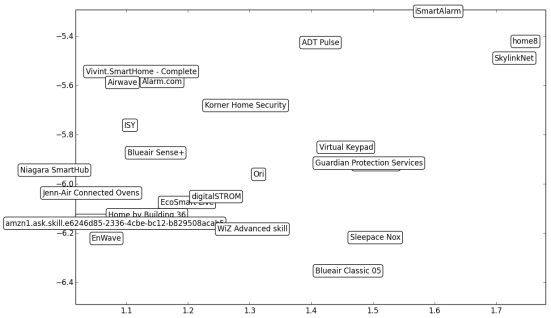

Figure 3: A large cluster of home automation domains.

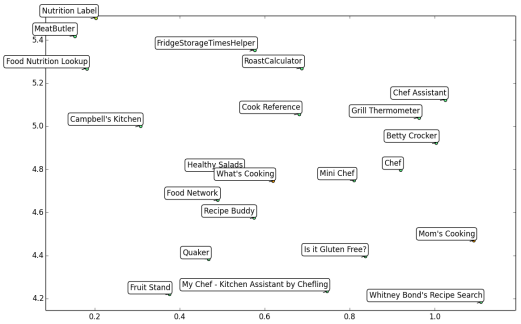

Figure 4: A cluster of domains related to cooking.

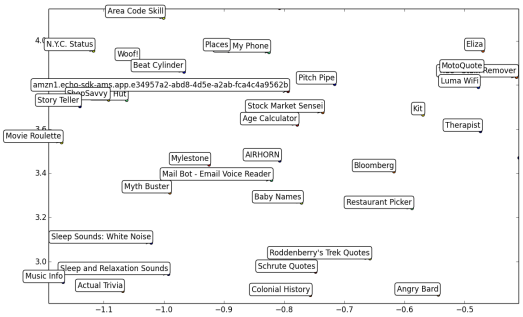

Figure 5: A mixed cluster with several different domain categories represented.

6 Conclusions

We have described a neural model architecture to address large-scale skill classification in an IPDA used by tens of millions of users every day. We have described how personalization features and an attention mechanism can be used for handling ambiguity between domains. We have also shown that the model can be extended efficiently and incrementally for new domains, saving multiple orders of magnitude in terms of training time. The model also addresses practical constraints of having a low memory footprint, low latency and being easily parallelized, all of which are important characteristics for real-time production systems. Significant challenges still remain in terms of incorporating contextual and dialogue cues into a large scale NLU system which is trained without manual labels.

References

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

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero shot frame semantic parsing for domain scaling. In Interspeech 2017.

A. Bhargava, Asli Celikyilmaz, Dilek Z. Hakkani-Tur, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. IEEE International Conference on Acoustics, Speech and Signal Processing, pages 8337–8341.

Asli Celikyilmaz, Ruhi Sarikaya, Dilek Hakkani-Tür, Xiaohu Liu, Nikhil Ramesh, and Gökhan Tür. 2016. A new pre-training method for training deep learning models with application to spoken language understanding. In Interspeech, pages 3255–3259.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016a. 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, pages 6045–6049.

Yun-Nung Chen, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016b. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In Interspeech.

Ali El-Kahky, Xiaohu 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. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4067–4071. IEEE.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. CoRR, abs/1706.04326.

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

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 541–550. Association for Computational Linguistics.

Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R Hershey. 2016. Context-sensitive and role-dependent spoken language understanding using bidirectional and attention lstms. Interspeech, pages 3236–3240.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. In Interspeech.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.

Mark D Kernighan, Kenneth W Church, and William A Gale. 1990. A spelling correction program based on a noisy channel model. In Proceedings of the 13th conference on Computational linguistics-Volume 2, pages 205–210. Association for Computational Linguistics.

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 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 84–92.

Young-Bum Kim, Sungjin Lee, and Ruhi Sarikaya. 2017a. Speaker-sensitive dual memory networks for multi-turn slot tagging. In Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE, pages 547–553. IEEE.

Young-Bum Kim, Sungjin Lee, and Karl Stratos. 2017b. Onenet: Joint domain, intent, slot prediction for spoken language understanding. In Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE, pages 547–553. IEEE.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017c. Adversarial adaptation of synthetic or stale data. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1297–1307. Association for Computational Linguistics.

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

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015b. Pre-training of hidden-unit crfs. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, volume 2, pages 192–198.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 387–396.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2017e. A framework for pre-training hidden-unit conditional random fields and its extension to long short term memory networks. Computer Speech & Language, 46:311–326.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015c. New transfer learning techniques for disparate label sets. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, volume 1, pages 473–482.

Gunter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. CoRR, abs/1706.02515.

Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In ICML, volume 97, pages 179–186. Nashville, USA.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, and Markus Dreyer. 2017a. Just ask: Building an architecture for extensible self-service spoken language understanding. arXiv preprint arXiv:1711.00549.

Anjishnu Kumar, Pavankumar Reddy Muddireddy, Markus Dreyer, and Björn Hoffmeister. 2017b. Zero-shot learning across heterogeneous overlapping domains. Proc. Interspeech 2017, pages 2914–2918.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In Proceedings of NAACL-HLT, pages 260–270.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In Interspeech, pages 685–689.

Bing Liu and Ian Lane. 2017. Multi-domain adversarial learning for slot filling in spoken language understanding. In NIPS Workshop on Conversational AI.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research, 9:2579–2605.

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

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In Advances in Neural Information Processing Systems, pages 3567–3575.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. IEEE Signal Processing Magazine, 34(1):67–81.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In Spoken Language Technology Workshop (SLT), 2016 IEEE, pages 391–397. IEEE.

Michael L Seltzer and Jasha Droppo. 2013. Multitask learning in deep neural networks for improved phoneme recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6965–6969. IEEE.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 1, pages I–I. IEEE.

Gokhan Tur and Renato de Mori. 2011. Spoken Language Understanding: Systems for Extracting Semantic Information from Speech. New York, NY: John Wiley and Sons.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. arXiv preprint arXiv:1602.00367.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 136–140. IEEE.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. International Conference on Learning Representation (ICLR).

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In International Joint Conference on Artificial Intelligence (IJCAI), pages 2993–2999.