Multi-domain Neural Network Language Generation for Spoken Dialogue Systems

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Steve Young
Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK
{thw28,mg436,nm480,lmr46,phs26,djv27,sjy}@cam.ac.uk

Abstract

Moving from limited-domain natural language generation (NLG) to open domain is difficult because the number of semantic input combinations grows exponentially with the number of domains. Therefore, it is important to leverage existing resources and exploit similarities between domains to facilitate domain adaptation. In this paper, we propose a procedure to train multi-domain, Recurrent Neural Network-based (RNN) language generators via multiple adaptation steps. In this procedure, a model is first trained on counterfeited data synthesised from an out-of-domain dataset, and then fine tuned on a small set of in-domain utterances with a discriminative objective function. Corpus-based evaluation results show that the proposed procedure can achieve competitive performance in terms of BLEU score and slot error rate while significantly reducing the data needed to train generators in new, unseen domains. In subjective testing, human judges confirm that the procedure greatly improves generator performance when only a small amount of data is available in the domain.

1 Introduction

Modern Spoken Dialogue Systems (SDS) are typically developed according to a well-defined *ontology*, which provides a structured representation of the *domain* data that the dialogue system can talk about, such as searching for a restaurant or shopping for a laptop. Unlike conventional approaches employing a substantial amount of handcrafting for each individual processing component (Ward and Issar, 1994; Bohus and Rudnicky, 2009), statistical approaches to SDS promise a domain-scalable framework which requires a minimal amount of human intervention (Young et al., 2013). Mrkšić et al. (2015) showed improved performance in belief tracking by training a general model and adapting it to specific domains. Similar benefit can be observed in Gašić et al. (2015), in which a Bayesian committee machine (Tresp, 2000) was used to model policy learning in a multi-domain SDS regime.

In past decades, adaptive NLG has been studied from linguistic perspectives, such as systems that learn to tailor user preferences (Walker et al., 2007), convey a specific personality trait (Mairesse and Walker, 2008; Mairesse and Walker, 2011), or align with their conversational partner (Isard et al., 2006). Domain adaptation was first addressed by Hogan et al. (2008) using a generator based on the Lexical Functional Grammar (LFG) f-structures (Kaplan and Bresnan, 1982). Although these approaches can model rich linguistic phenomenon, they are not readily adaptable to data since they still require many handcrafted rules to define the search space. Recently, RNN-based language generation has been introduced (Wen et al., 2015a; Wen et al., 2015b). This class of statistical generators can learn generation decisions directly from dialogue act (DA)-utterance pairs without any semantic annotations (Mairesse and Young, 2014) or hand-coded grammars (Langkilde and Knight, 1998; Walker et al., 2002). Many existing adaptation approaches (Wen et al., 2013; Shi et al., 2015; Chen et al., 2015) can be directly applied due to the

Proceedings of NAACL-HLT 2016, pages 120–129,
San Diego, California, June 12-17, 2016.

flexibility of the underlying RNN language model (RNNLM) architecture (Mikolov et al., 2010).

Discriminative training (DT) has been successfully used to train RNNs for various tasks. By optimising directly against the desired objective function such as BLEU score (Auli and Gao, 2014) or Word Error Rate (Kuo et al., 2002), the model can explore its output space and learn to discriminate between good and bad hypotheses. In this paper we show that DT can enable a generator to learn more efficiently when in-domain data is scarce.

The paper presents an incremental recipe for training multi-domain language generators based on a purely data-driven, RNN-based generation model. Following a review of related work in section 2, section 3 describes the detailed RNN generator architecture. The data counterfeiting approach for synthesising an in-domain dataset is introduced in section 4, where it is compared to the simple model fine-tuning approach. In section 5, we describe our proposed DT procedure for training natural language generators. Following a brief review of the data sets used in section 6, corpus-based evaluation results are presented in section 7. In order to assess the subjective performance of our system, a quality test and a pairwise preference test are presented in section 8. The results show that the proposed adaptation recipe improves not only the objective scores but also the user's perceived quality of the system. We conclude with a brief summary in section 9.

2 Related Work

Domain adaptation problems arise when we have a sufficient amount of labeled data in one domain (the *source* domain), but have little or no labeled data in another related domain (the *target* domain). Domain adaptability for real world speech and language applications is especially important because both language usage and the topics of interest are constantly evolving. Historically, domain adaptation has been less well studied in the NLG community. The most relevant work was done by Hogan et al. (2008). They showed that an LFG f-structure based generator could yield better performance when trained on in-domain sentences paired with pseudo parse tree inputs generated from a state-of-the-art, but out-of-domain parser. The SPoT-based generator proposed by Walker et al. (2002) has the potential to address domain adaptation problems. However, their published work has focused on tailoring user preferences (Walker et al., 2007) and mimicking personality traits (Mairesse and Walker, 2011). Lemon (2008) proposed a Reinforcement Learning (RL) framework in which policy and NLG components can be jointly optimised and adapted based on online user feedback. In contrast, Mairesse et al. (2010) has proposed using active learning to mitigate the data sparsity problem when training data-driven NLG systems. Furthermore, Cuayhuitl et al. (2014) trained statistical surface realisers from unlabelled data by an automatic slot labelling technique.

In general, feature-based adaptation is perhaps the most widely used technique (Blitzer et al., 2007; Pan and Yang, 2010; Duan et al., 2012). By exploiting correlations and similarities between data points, it has been successfully applied to problems like speaker adaptation (Gauvain and Lee, 1994; Leggetter and Woodland, 1995) and various tasks in natural language processing (Daumé III, 2009). In contrast, model-based adaptation is particularly useful for language modeling (LM) (Bellegarda, 2004). Mixture-based topic LMs (Gildea and Hofmann, 1999) are widely used in N-gram LMs for domain adaptation. Similar ideas have been applied to applications that require adapting LMs, such as machine translation (MT) (Koehn and Schroeder, 2007) and personalised speech recognition (Wen et al., 2012).

Domain adaptation for Neural Network (NN)-based LMs has also been studied in the past. A feature augmented RNNLM was first proposed by Mikolov and Zweig (2012), but later applied to multi-genre broadcast speech recognition (Chen et al., 2015) and personalised language modeling (Wen et al., 2013). These methods are based on fine-tuning existing network parameters on adaptation data. However, careful regularisation is often necessary (Yu et al., 2013). In a slightly different area, Shi et al. (2015) applied curriculum learning to RNNLM adaptation.

Discriminative training (DT) (Collins, 2002) is an alternative to the maximum likelihood (ML) criterion. For classification, DT can be split into two phases: (1) decoding training examples using the current model and scoring them, and (2) adjusting the model parameters to maximise the separation

between the correct target annotation and the competing incorrect annotations. It has been successfully applied to many research problems, such as speech recognition (Kuo et al., 2002; Voigtlaender et al., 2015) and MT (He and Deng, 2012; Auli et al., 2014). Recently, Auli and Gao (2014) trained an RNNLM with a DT objective and showed improved performance on an MT task. However, their RNN probabilities only served as input features to a phrase-based MT system.

3 The Neural Language Generator

The neural language generation model (Wen et al., 2015a; Wen et al., 2015b) is a RNNLM (Mikolov et al., 2010) augmented with semantic input features such as a dialogue act[1] (DA) denoting the required semantics of the generated output. At every time step t, the model consumes the 1-hot representation of both the DA $\mathbf{d}_t$ and a token $\mathbf{w}_t$[2] to update its internal state $\mathbf{h}_t$. Based on this new state, the output distribution over the next output token is calculated. The model can thus generate a sequence of tokens by repeatedly sampling the current output distribution to obtain the next input token until an end-of-sentence sign is generated. Finally, the generated sequence is lexicalised[3] to form the target utterance.

The Semantically Conditioned Long Short-term Memory Network (SC-LSTM) (Wen et al., 2015b) is a specialised extension of the LSTM network (Hochreiter and Schmidhuber, 1997) for language generation which has previously been shown capable of learning generation decisions from paired DA-utterances end-to-end without a modular pipeline (Walker et al., 2002; Stent et al., 2004). Like LSTM, SC-LSTM relies on a vector of memory cells $\mathbf{c}_t \in \mathbb{R}^n$ and a set of elementwise multiplication gates to control how information is stored, forgotten, and exploited inside the network. The SC-LSTM architecture used in this paper is defined by the following equations,

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{r}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \tanh \end{pmatrix} \mathbf{W}_{5n,2n} \begin{pmatrix} \mathbf{w}_t \\ \mathbf{h}_{t-1} \end{pmatrix}$$

$$\mathbf{d}_t = \mathbf{r}_t \odot \mathbf{d}_{t-1}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t + tanh(\mathbf{W}_{dc}\mathbf{d}_t)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

where n is the hidden layer size, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t, \mathbf{r}_t \in [0,1]^n$ are input, forget, output, and reading gates respectively, $\hat{\mathbf{c}}_t$ and $\mathbf{c}_t$ are proposed cell value and true cell value at time t, $\mathbf{W}_{5n,2n}$ and $\mathbf{W}_{dc}$ are the model parameters to be learned. The major difference of the SC-LSTM compared to the vanilla LSTM is the introduction of the reading gates for controlling the semantic input features presented to the network. It was shown in Wen et al. (2015b) that these reading gates act like keyword and key phrase detectors that learn the alignments between individual semantic input features and their corresponding realisations without additional supervision.

After the hidden layer state is obtained, the computation of the next word distribution and sampling from it is straightforward,

$$p(w_{t+1}|w_t, w_{t-1}, ...w_0, \mathbf{d}_t) = softmax(\mathbf{W}_{ho}\mathbf{h}_t)$$

$$w_{t+1} \sim p(w_{t+1}|w_t, w_{t-1}, ...w_0, \mathbf{d}_t).$$

where $\mathbf{W}_{ho}$ is another weight matrix to learn. The entire network is trained end-to-end using a cross entropy cost function, between the predicted word distribution $\mathbf{p}_t$ and the actual word distribution $\mathbf{y}_t$, with regularisations on DA transition dynamics,

$$F(\theta) = \sum_t \mathbf{p}_t^\intercal log(\mathbf{y}_t) + \|\mathbf{d}_\mathrm{T}\| + \sum_{t=0}^{\mathrm{T}-1} \eta\xi^{\|\mathbf{d}_{t+1}-\mathbf{d}_t\|} \quad (1)$$

where $\theta = \{\mathbf{W}_{5n,2n}, \mathbf{W}_{dc}, \mathbf{W}_{ho}\}$, $\mathbf{d}_\mathrm{T}$ is the DA vector at the last index T, and η and ξ are constants set to 10^{-4} and 100, respectively.

4 Training Multi-domain Models

Given training instances (represented by DA and sentence tuples $\{d_i, \Omega_i\}$) from the source domain $\mathbb{S}$ (rich) and the target domain $\mathbb{T}$ (limited), the goal is to find a set of SC-LSTM parameters $\theta_\mathbb{T}$ that can perform acceptably well in the target domain.

[1] A combination of an action type and a set of slot-value pairs. e.g. *inform(name="Seven days",food="chinese")*

[2] We use *token* instead of *word* because our model operates on text for which slot values are replaced by their corresponding slot tokens. We call this procedure delexicalisation.

[3] The process of replacing slot token by its value.

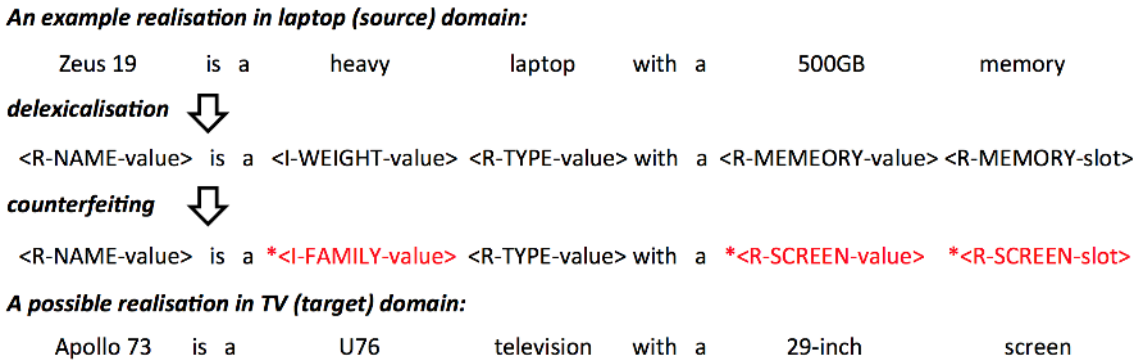


Figure 1: An example of data counterfeiting algorithm. Both slots and values are delexicalised. Slots and values that are not in the target domain are replaced during data counterfeiting (shown in *red* with * sign). The prefix inside bracket <> indicates the slot's functional class (I for *informable* and R for *requestable*).

4.1 Model Fine-Tuning

A straightforward way to adapt NN-based models to a target domain is to continue training or fine-tuning a well-trained generator on whatever new target domain data is available. This training procedure is as follows:

1. Train a source domain generator $\theta_{\mathbb{S}}$ on source domain data $\{d_i, \Omega_i\} \in \mathbb{S}$ with all values delexicalised[4].

2. Divide the adaptation data into training and validation sets. Refine parameters by training on adaptation data $\{d_i, \Omega_i\} \in \mathbb{T}$ with early stopping and a smaller starting learning rate. This yields the target domain generator $\theta_{\mathbb{T}}$.

Although this method can benefit from parameter sharing of the LM part of the network, the parameters of similar input slot-value pairs are not shared[4]. In other words, realisation of any unseen slot-value pair in the target domain can only be learned from scratch. Adaptation offers no benefit in this case.

4.2 Data Counterfeiting

In order to maximise the effect of domain adaptation, the model should be able to (1) generate acceptable realisations for unseen slot-value pairs based on similar slot-value pairs seen in the training data, and (2) continue to distinguish slot-value pairs that are similar but nevertheless distinct. Instead of exploring weight tying strategies in different training stages (which is complex to implement and typically relies on ad hoc tying rules), we propose instead a data counterfeiting approach to *synthesise* target domain data from source domain data. The procedure is shown in Figure 1 and described as following:

1. Categorise slots in both source and target domain into classes, according to some similarity measure. In our case, we categorise them based on their functional type to yield three classes: *informable*, *requestable*, and *binary*[5].

2. Delexicalise all slots and values.

3. For each slot s in a source instance $(d_i, \Omega_i) \in \mathbb{S}$, randomly select a new slot s' that belongs to both the target ontology and the class of s to replace s. Repeat this process for every slot in the instance and yield a new pseudo instance $(\hat{d}_i, \hat{\Omega}_i) \in \mathbb{T}$ in the target domain.

4. Train a generator $\hat{\theta}_{\mathbb{T}}$ on the counterfeited dataset $\{\hat{d}_i, \hat{\Omega}_i\} \in \mathbb{T}$.

5. Refine parameters on real in-domain data. This yields final model parameters $\theta_{\mathbb{T}}$.

This approach allows the generator to share realisations among slot-value pairs that have similar functionalities, therefore facilitates the transfer learning

[4]We have tried training with both slots and values delexicalised and then using the weights to initialise unseen slot-value pairs in the target domain. However, this yielded even worse results since the learned semantic alignment stuck at local minima. Pre-training only the LM parameters did not produce better results either.

[5]*Informable* class include all non-binary informable slots while *binary* class includes all binary informable slots.

	Laptop	Television
informable slots	family, *pricerange, batteryrating, driverange, weightrange, **isforbusinesscomputing**	family, *pricerange, screensizerange, ecorating, hdmiport, **hasusbport**
requestable slots	*name, *type, *price, warranty, battery, design, dimension, utility, weight, platform, memory, drive, processor	*name, *type, *price, resolution, powerconsumption, accessories, color, screensize, audio
act type	*inform, *inform_only_match, *inform_on_match, inform_all, *inform_count, inform_no_info, *recommend, compare, *select, suggest, *confirm, *request, *request_more, *goodbye	

bold=binary slots, *=overlap with SF Restaurant and Hotel domains, all *informable slots* can take "dontcare" value

Table 1: Ontologies for Laptop and TV domains

of rare slot-value pairs in the target domain. Furthermore, the approach also preserves the co-occurrence statistics of slot-value pairs and their realisations. This allows the model to learn the gating mechanism even before adaptation data is introduced.

5 Discriminative Training

In contrast to the traditional ML criteria (Equation 1) whose goal is to maximise the log-likelihood of correct examples, DT aims at separating correct examples from competing incorrect examples. Given a training instance (d_i, Ω_i), the training process starts by generating a set of candidate sentences $Gen(d_i)$ using the current model parameter θ and DA d_i. The discriminative cost function can therefore be written as

$$\begin{aligned} F(\theta) &= -\mathbb{E}[L(\theta)] \\ &= -\sum_{\Omega \in Gen(d_i)} p_\theta(\Omega|d_i) L(\Omega, \Omega_i) \end{aligned} \quad (2)$$

where $L(\Omega, \Omega_i)$ is the scoring function evaluating candidate Ω by taking ground truth Ω_i as reference. $p_\theta(\Omega|d_i)$ is the normalised probability of the candidate and is calculated by

$$p_\theta(\Omega|d_i) = \frac{\exp[\gamma \log p(\Omega|d_i,\theta)]}{\sum_{\Omega' \in Gen(d_i)} \exp[\gamma \log p(\Omega'|d_i,\theta)]} \quad (3)$$

$\gamma \in [0, \infty]$ is a tuned scaling factor that flattens the distribution for $\gamma < 1$ and sharpens it for $\gamma > 1$. The unnormalised candidate likelihood $\log p(\Omega|d_i, \theta)$ is produced by summing token likelihoods from the RNN generator output,

$$\log p(\Omega|d_i, \theta) = \sum_{w_t \in \Omega} \log p(w_t|d_i, \theta) \quad (4)$$

The scoring function $L(\Omega, \Omega_i)$ can be further generalised to take several scoring functions into account

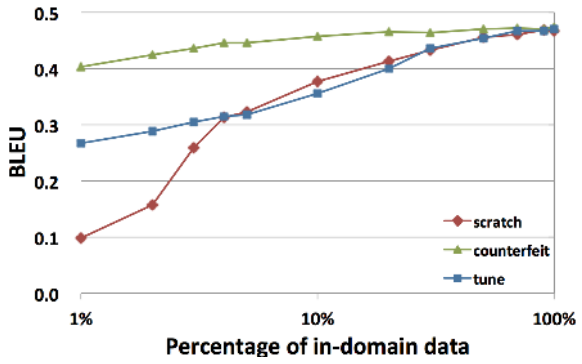
$$L(\Omega, \Omega_i) = \sum_j L_j(\Omega, \Omega_i)\beta_j \quad (5)$$

where β_j is the weight for j-th scoring function. Since the cost function presented here (Equation 2) is differentiable everywhere, back propagation can be applied to calculate the gradients and update parameters directly.

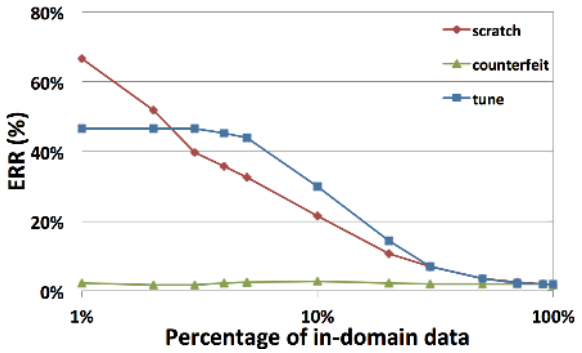
6 Datasets

In order to test our proposed recipe for training multi-domain language generators, we conducted experiments using four different domains: finding a restaurant, finding a hotel, buying a laptop, and buying a television. Datasets for the restaurant and hotel domains have been previously released by Wen et al. (2015b). These were created by workers recruited by Amazon Mechanical Turk (AMT) by asking them to propose an appropriate natural language realisation corresponding to each system dialogue act actually generated by a dialogue system. However, the number of actually occurring DA combinations in the restaurant and hotel domains were rather limited ($\sim$200) and since multiple references were collected for each DA, the resulting datasets are not sufficiently diverse to enable the assessment of the generalisation capability of the different training methods over unseen semantic inputs.

In order to create more diverse datasets for the laptop and TV domains, we enumerated all possible combinations of dialogue act types and slots based on the ontology shown in Table 1. This yielded

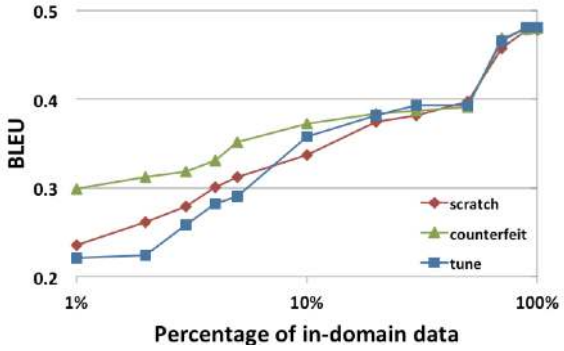


(a) BLEU score curve

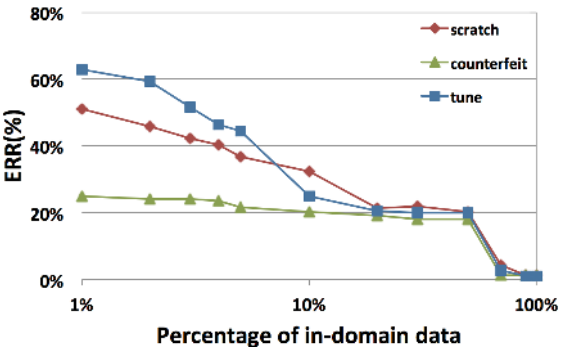


(b) Slot error rate curve

Figure 2: Results evaluated on TV domain by adapting models from laptop domain. Comparing train-from-scratch model (*scratch*) with model fine-tuning approach (*tune*) and data counterfeiting method (*counterfeit*). $10\% \approx 700$ examples.



(a) BLEU score curve



(b) Slot error rate curve

Figure 3: The same set of comparison as in Figure 2, but the results were evaluated by adapting from SF restaurant and hotel joint dataset to laptop and TV joint dataset. $10\% \approx 2K$ examples.

about 13K distinct DAs in the laptop domain and 7K distinct DAs in the TV domain. We then used AMT workers to collect just one realisation for each DA. Since the resulting datasets have a much larger input space but only one training example for each DA, the system must learn partial realisations of concepts and be able to recombine and apply them to unseen DAs. Also note that the number of act types and slots of the new ontology is larger, which makes NLG in both laptop and TV domains much harder.

7 Corpus-based Evaluation

We first assess generator performance using two objective evaluation metrics, the BLEU-4 score (Papineni et al., 2002) and slot error rate ERR (Wen et al., 2015b). Slot error rates were calculated by averaging slot errors over each of the top 5 realisations in the entire corpus. We used multiple references to compute the BLEU scores when available (i.e. for the restaurant and hotel domains). In order to better compare results across different methods, we plotted the BLEU and slot error rate curves against different amounts of adaptation data. Note that in the graphs the x-axis is presented on a log-scale.

7.1 Experimental Setup

The generators were implemented using the Theano library (Bergstra et al., 2010; Bastien et al., 2012), and trained by partitioning each of the collected corpora into a training, validation, and testing set in the ratio 3:1:1. All the generators were trained by treating each sentence as a mini-batch. An l_2 regularisation term was added to the objective function for every 10 training examples. The hidden layer size was set to be 100 for all cases. Stochastic gradient descent and back propagation through time (Werbos, 1990) were used to optimise the parameters. In order to prevent overfitting, early stopping was implemented using the validation set.

During decoding, we over-generated 20 utterances and selected the top 5 realisations for each DA

according to the following reranking criteria,

$$R = -(F(\theta) + \lambda \mathrm{ERR}) \quad (6)$$

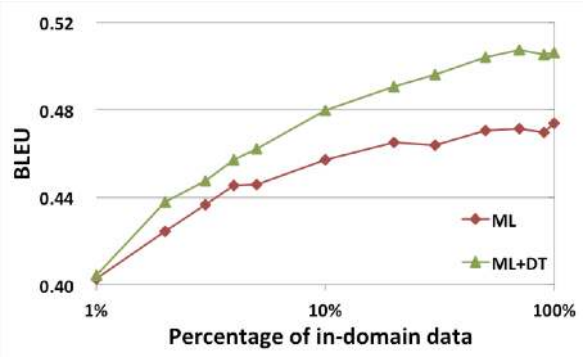
where λ is a tradeoff constant, $F(\theta)$ is the cost generated by network parameters θ, and the slot error rate ERR is computed by exact matching of the slot tokens in the candidate utterances. λ is set to a large value (10) in order to severely penalise nonsensical outputs. Since our generator works stochastically and the trained networks can differ depending on the initialisation, all the results shown below were averaged over 5 randomly initialised networks.

7.2 Data Counterfeiting

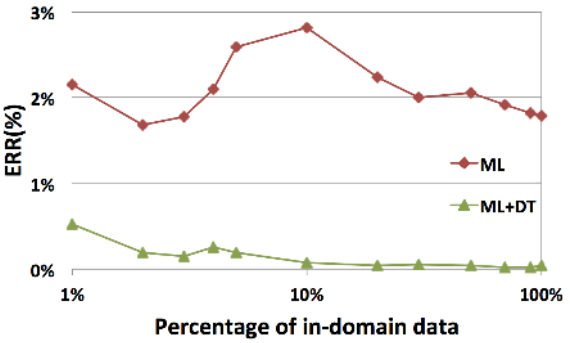
We first compared the data counterfeiting (*counterfeit*) approach with the model fine-tuning (*tune*) method and models trained from scratch (*scratch*). Figure 2 shows the result of adapting models between similar domains, from laptop to TV. Because of the parameter sharing in the LM part of the network, model fine-tuning (*tune*) achieves a better BLEU score than training from scratch (*scratch*) when target domain data is limited. However, if we apply the data counterfeiting (*counterfeit*) method, we obtain an even greater BLEU score gain. This is mainly due to the better realisation of unseen slot-value pairs. On the other hand, data counterfeiting (*counterfeit*) also brings a substantial reduction in slot error rate. This is because it preserves the co-occurrence statistics between slot-value pairs and realisations, which allows the model to learn good semantic alignments even before adaptation data is introduced. Similar results can be seen in Figure 3, in which adaptation was performed on more disjoint domains: restaurant and hotel joint domain to laptop and TV joint domain. The data counterfeiting (*counterfeit*) method is still superior to the other methods.

7.3 Discriminative Training

The generator parameters obtained from data counterfeiting and ML adaptation were further tuned by applying DT. In each case, the models were optimised using two objective functions: BLEU-4 score and slot error rate. However, we used a soft version of BLEU called sentence BLEU as described in Auli and Gao (2014), to mitigate the sparse n-gram match problem of BLEU at the sentence level. In our experiments, we set γ to 5.0 and β_j to 1.0 and -1.0 for



(a) Effect of DT on BLEU



(b) Effect of DT on slot error rate

Figure 4: Effect of applying DT training after ML adaptation. The results were evaluated on laptop to TV adaptation. $10\% \approx 700$ examples.

BLEU and ERR, respectively. For each DA, we applied our generator 50 times to generate candidate sentences. Repeated candidates were removed. We treated the remaining candidates as a single batch and updated the model parameters by the procedure described in section 5. We evaluated performance of the algorithm on the laptop to TV adaptation scenario, and compared models with and without discriminative training (*ML+DT* & *ML*). The results are shown in Figure 4 where it can be seen that DT consistently improves generator performance on both metrics. Another interesting point to note is that slot error rate is easier to optimise compared to BLEU (ERR$\rightarrow$ 0 after DT). This is probably because the sentence BLEU optimisation criterion is only an approximation of the corpus BLEU score used for evaluation.

8 Human Evaluation

Since automatic metrics may not consistently agree with human perception (Stent et al., 2005), human testing is needed to assess subjective quality. To do

Method	TV to Laptop		laptop to TV	
	Info.	Nat.	Info.	Nat.
scrALL	2.64	2.37	2.54	2.36
DT-10%	**2.52****	**2.25****	**2.51**	2.19**
ML-10%	2.51**	2.22**	2.45**	**2.22****
scr-10%	2.24**	2.03**	2.00**	1.92**

* p <0.05, ** p <0.005

Table 2: Human evaluation for utterance quality in two domains. Results are shown in two metrics (rating out of 3). Statistical significance was computed using a two-tailed Student's t-test, between the model trained with full data (*scrALL*) and all others.

this, a set of judges were recruited using AMT. We tested our models on two adaptation scenarios: laptop to TV and TV to laptop. For each task, two systems among the four were compared: training from scratch using full dataset (*scrALL*), adapting with DT training but only 10% of target domain data (*DT-10%*), adapting with ML training but only 10% of target domain data (*ML-10%*), and training from scratch using only 10% of target domain data (*scr-10%*). In order to evaluate system performance in the presence of language variation, each system generated 5 different surface realisations for each input DA and the human judges were asked to score each of them in terms of informativeness and naturalness (rating out of 3), and also asked to state a preference between the two. Here *informativeness* is defined as whether the utterance contains all the information specified in the DA, and *naturalness* is defined as whether the utterance could plausibly have been produced by a human. In order to decrease the amount of information presented to the judges, utterances that appeared identically in both systems were filtered out. We tested about 2000 DAs for each scenario distributed uniformly between contrasts except that allowed 50% more comparisons between *ML-10%* and *DT-10%* because they were close.

Table 2 shows the subjective quality assessments which exhibit the same general trend as the objective results. If a large amount of target domain data is available, training everything from scratch (*scrALL*) achieves a very good performance and adaptation is not necessary. However, if only a limited amount of in-domain data is available, efficient adaptation is critical (*DT-10% & ML-10% > scr-10%*). Moreover, judges also preferred the DT trained generator (*DT-10%*) compared to the ML trained generator (*ML-10%*), especially for *informativeness*. In the laptop to TV scenario, the *informativeness* score of DT method (*DT-10%*) was considered indistinguishable when comparing to the method trained with full training set (*scrALL*). The preference test results are shown in Table 3. Again, adaptation methods (*DT-10% & ML-10%*) are crucial to bridge the gap between domains when the target domain data is scarce (*DT-10% & ML-10% > scr-10%*). The results also suggest that the DT training approach (*DT-10%*) was preferred compared to ML training (*ML-10%*), even though the preference in this case was not statistically significant.

Pref.%	scr-10%	ML-10%	DT-10%	scrALL
scr-10%	-	34.5**	33.9**	22.4**
ML-10%	65.5**	-	44.9	36.8**
DT-10%	66.1**	55.1	-	35.9**
scrALL	77.6**	63.2**	64.1**	-

* p <0.05, ** p <0.005

(a) Preference test on TV to laptop adaptation scenario

Pref.%	scr-10%	ML-10%	DT-10%	scrALL
scr-10%	-	17.4**	14.2**	14.8**
ML-10%	82.6**	-	48.1	37.1**
DT-10%	85.8**	51.9	-	41.6*
scrALL	85.2**	62.9**	58.4*	-

* p <0.05, ** p <0.005

(b) Preference test on laptop to TV adaptation scenario

Table 3: Pairwise preference test among four approaches in two domains. Statistical significance was computed using two-tailed binomial test.

9 Conclusion and Future Work

In this paper we have proposed a procedure for training multi-domain, RNN-based language generators, by data counterfeiting and discriminative training. The procedure is general and applicable to any data-driven language generator. Both corpus-based evaluation and human assessment were performed. Objective measures on corpus data have demonstrated that by applying this procedure to adapt models between four different dialogue domains, good performance can be achieved with much less training data. Subjective assessment by human judges confirm the effectiveness of the approach.

The proposed domain adaptation method requires a small amount of annotated data to be collected offline. In our future work, we intend to focus on training the generator on the fly with real user feedback during conversation.

Acknowledgments

Tsung-Hsien Wen and David Vandyke are supported by Toshiba Research Europe Ltd, Cambridge Research Laboratory.

References

Michael Auli and Jianfeng Gao. 2014. Decoder integration and expected bleu training for recurrent neural network language models. In *Proceedings of ACL*. Association for Computational Linguistics.

Michael Auli, Michel Galley, and Jianfeng Gao. 2014. Large-scale expected bleu training of phrase-based reordering models. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*. Association for Computational Linguistics.

Dan Bohus and Alexander I. Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech and Language*.

Xie Chen, Tan Tian, Liu Xunying, Lanchantin Pierre, Wan Moquan, Mark Gales, and Woodland Phil. 2015. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In *Proceedings of InterSpeech*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Heriberto Cuayhuitl, Nina Dethlefs, Helen Hastie, and Xingkun Liu. 2014. Training a statistical surface realiser from automatic slot labelling. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *CoRR*, abs/0907.1815.

Lixin Duan, Dong Xu, and Ivor W. Tsang. 2012. Learning with augmented features for heterogeneous domain adaptation. *CoRR*, abs/1206.4660.

Milica Gašić, Nikola Mrkšić, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Policy committee for adaptation in multi-domain spoken dialogue systems. In *Proceedings of ASRU*.

Jean-Luc Gauvain and Chin-Hui Lee. 1994. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *Speech and Audio Processing, IEEE Transactions on*.

Daniel Gildea and Thomas Hofmann. 1999. Topic-based language models using em. In *Proceedings of EuroSpeech*.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of ACL*. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Deirdre Hogan, Jennifer Foster, Joachim Wagner, and Josef van Genabith. 2008. Parser-based retraining for domain adaptation of probabilistic generators. In *Proceedings of INLG*. Association for Computational Linguistics.

Amy Isard, Carsten Brockmann, and Jon Oberlander. 2006. Individuality and alignment in generated dialogues. In *Proceedings of INLG*. Association for Computational Linguistics.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-Functional Grammar: a formal system for grammatical representation. In Joan Bresnan, editor, *The mental representation of grammatical relations*. MIT Press.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of StatMT*. Association for Computational Linguistics.

Hong-kwang Kuo, Eric Fosler-lussier, Hui Jiang, and Chin-hui Lee. 2002. Discriminative training of language models for speech recognition. In *Proceedings of ICASSP*.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of ACL*. Association for Computational Linguistics.

Chris Leggetter and Philip Woodland. 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*.

Oliver Lemon. 2008. Adaptive natural language generation in dialogue using reinforcement learning. In *Proceedings of SemDial*.

Franois Mairesse and Marilyn Walker. 2008. Trainable generation of big-five personality styles through data-driven parameter estimation. In *Proceedings of ACL*. Association for Computational Linguistics.

François Mairesse and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computer Linguistics*.

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computer Linguistics*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of ACL*. Association for Computational Linguistics.

Tomáš Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proceedings of SLT*.

Tomáš Mikolov, Martin Karafit, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of InterSpeech*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *Proceedings of ACL*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. on Knowledge and Data Engineering*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. Association for Computational Linguistics.

Yangyang Shi, Martha Larson, and Catholijn M. Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer, Speech and Language*.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of ACL*. Association for Computational Linguistics.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of CICLing 2005*.

Volker Tresp. 2000. A bayesian committee machine. *Neural Computation*.

Paul Voigtlaender, Patrick Doetsch, Simon Wiesler, Ralf Schluter, and Hermann Ney. 2015. Sequence-discriminative training of recurrent neural networks. In *Proceedings of ICASSP*.

Marilyn A Walker, Owen C Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*.

Marilyn Walker, Amanda Stent, Franois Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)*.

Wayne Ward and Sunil Issar. 1994. Recent improvements in the cmu spoken language understanding system. In *Proceedings of Workshop on HLT*. Association for Computational Linguistics.

Tsung-Hsien Wen, Hung-Yi Lee, Tai-Yuan Chen, and Lin-Shan Lee. 2012. Personalized language modeling by crowd sourcing with social network data for voice access of cloud applications. In *Proceedings of SLT*.

Tsung-Hsien Wen, Aaron Heidel, Hung yi Lee, Yu Tsao, and Lin-Shan Lee. 2013. Recurrent neural network based language model personalization by social network crowdsourcing. In *Proceedings of InterSpeech*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of SIGdial*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. 2013. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *Proceedings of ICASSP*.