A Deep Fusion Model for Domain Adaptation in Phrase-based MT

Nadir Durrani
ndurrani@qf.org.qa

Hassan Sajjad
hsajjad@qf.org.qa

Shafiq Joty
sjoty@qf.org.qa

Ahmed Abdelali
aabdelali@qf.org.qa

Qatar Computing Research Institute – HBKU

Abstract

We present a novel fusion model for domain adaptation in Statistical Machine Translation. Our model is based on the joint source-target neural network (Devlin et al., 2014), and is learned by fusing in- and out-domain models. The adaptation is performed by backpropagating errors from the output layer to the word embedding layer of each model, subsequently adjusting parameters of the composite model towards the in-domain data. On the standard tasks of translating English-to-German and Arabic-to-English TED talks, we observed average improvements of +0.9 and +0.7 BLEU points, respectively over a competition grade phrase-based system. We also demonstrate improvements over existing adaptation methods.

1 Introduction

The quality of machine translation systems is sensitive to the domain of the training data. More data, but out-of-domain data, may not be best suited and could even be harmful for specific translation tasks such as translating TED talks (Cettolo et al., 2014), patents (Fujii et al., 2010) and educational content (Guzmán et al., 2013). This is because of the difference in stylistic variations, vocabulary choices and word sense ambiguities across genres. *Domain adaptation* aims at finding the optimal point, that maximizes on the useful information available in the out-domain data, in favor of the in-domain data, while preventing it from degrading the performance of the system. This is either done by selecting a subset from the out-domain data, which is closer to the in-domain data (Matsoukas et al., 2009; Moore and Lewis, 2010), or by re-weighting the probability distribution in favor of the in-domain data (Foster and Kuhn, 2007; Sennrich, 2012).

Recently, there has been a growing interest in deep neural networks (DNNs) and word embeddings with application to numerous NLP problems. The ability to generalize and to better capture non-local dependencies gives edge to the neural models over their traditional counter-part. Several researchers have also attempted to employ DNNs for domain adaptation in SMT. Duh et al. (2013) and Durrani et al. (2015a) used DNNs for data selection. Joty et al. (2015) proposed a DNN-based adaptation model for SMT that regularizes the loss function with respect to the in-domain model.

In this paper, we propose a deep fusion approach to domain adaptation for SMT. We use the Neural Network Joint Model (NNJM) proposed by Devlin et al. (2014) as our base model. However, rather than training the model on a plain concatenation of in- and out-domain data or a weighted concatenation (Joty et al., 2015), we first train in- and out-domain NNJM models, and then learn a composite model by readjusting their parameters through backpropagating errors from the output layer to the word embedding layer of each model. The intuition behind learning the models separately, is to learn in-domain model parameters without contaminating them with the out-domain data. In a variant of our model, we restrict backpropagation to only the outermost hidden layer and adjust only the final layer combination weights. The composite model is used as a feature during decoding, where it can help assigning higher scores to the hypotheses that represent lexical choices and patterns preferred by the in-domain data.

We evaluated our model on a standard task of translating TED talks for English-to-German and Arabic-to-English language pairs. Compared to baseline NNJM models trained on a concatenation of in- and out-domain data, our model achieves an average improvement of up to 0.9 BLEU points. The most relevant to our work are the NDAM model of Joty et al. (2015) and the *fine-tuning* method of Luong and Manning (2015). The NDAM model uses data dependent regularization in the NNJM model to weight training instances, while training the model on the concatenated data. The fine-tuning method first trains the NNJM on the concatenated data, then runs a few additional epochs on the in-domain data to tune the model towards in-domain. We found our method to outperform both the NDAM and fine-tuning methods. We also carried experiments against phrase-table weighting methods such as instance weighting (Sennrich, 2012), and phrase-table fill-up combination (Bisazza et al., 2011) and found our approach to outperform these. Our approach is complementary and the gains obtained were found to be additive on top of phrase-table adaptation and MML-based data-selection (Axelrod et al., 2011).

The remainder of this paper is organized as follows. Section 2 briefly describes neural network joint model. Section 3 describes our fusion model for domain adaptation. Section 4 presents results and analysis. Section 5 gives an account on the related work and Section 6 concludes the paper.

2 Neural Network Joint Model

Neural models are quickly becoming the state-of-the-art in machine translation. The ability to generalize and better capture non-local dependencies gives them edge over traditional models. The two most prevalent approaches are to use NNs as a feature inside SMT decoder (Vaswani et al., 2013; Devlin et al., 2014), or as an end-to-end translation system (Luong et al., 2015; Bahdanau et al., 2015; Sennrich et al., 2016) designed as fully trainable model of which every component is tuned based on training corpora to maximize its translation performance. Our work falls in the former category and extends NNJM.

The NNJM model learns a feed-forward neural network from augmented streams of source and target sequences. For a bilingual sentence pair (S, T), NNJM defines a conditional probability distribution:

$$P(T|S) \approx \prod_{i=1}^{|T|} P(t_i|t_{i-1} \dots t_{i-n+1}, \mathbf{s_i}) \tag{1}$$

where, $\mathbf{s_i}$ is an m-word source window for a target word t_i based on the one-to-one alignment between T and S. This is essentially an $(m+n)$-gram bilingual language model originally proposed by Bengio et al. (2003). As shown in Figure 1, each input word in the context has a D dimensional vector representation in the shared embedding layer $E \in \mathbb{R}^{|V_i| \times D}$, where V_i is the input vocabulary; E is considered as a model parameter to be learned. The context of the sequence is represented by a concatenated vector $\mathbf{x_n} \in \mathbb{R}^{(m+n-1)D}$, which is then passed through non-linear hidden layers to learn high-level abstract representations. The output layer defines a `softmax` over the output vocabulary V_o:

$$P(y_n = k|\mathbf{x_n}, \theta) = \frac{\exp\ (\mathbf{w}_k^T \mathbf{z_n})}{\sum_{m=1}^{|V_o|} \exp\ (\mathbf{w}_m^T \mathbf{z_n})} \tag{2}$$

where $\mathbf{z}_n = \phi(\mathbf{x_n})$ defines the non-linear transformations of $\mathbf{x_n}$ through one or more hidden layers, and $\mathbf{w}_k$ are the weights from the outermost hidden layer to the output layer. By setting m and n to be sufficiently large, NNJM can capture long-range cross-lingual dependencies between words. The maximum (log) likelihood objective of the model can be written as:

$$J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{|V_o|} y_{nk}\ log\ P(y_n = k|\mathbf{x_n}, \theta) \tag{3}$$

where, $y_{nk} = I(y_n = k)$ is an indicator variable (i.e., y_{nk}=1 when y_n=k, otherwise 0).

A major efficiency bottleneck of NNJM is to surmount the computational cost involved in training the model and applying it for MT decoding. Devlin et al. (2014) proposed two tricks to speed up computation in decoding. The first one is to pre-compute the hidden layer computations and fetch them directly as needed during decoding. The second technique is to train a *self-normalized* NNJM to avoid computation of the softmax normalization factor (i.e., the denominator in Equation 2) in decoding. However, self-normalization does not solve the computational cost of training the model. In the following, we describe a method that addresses this issue.

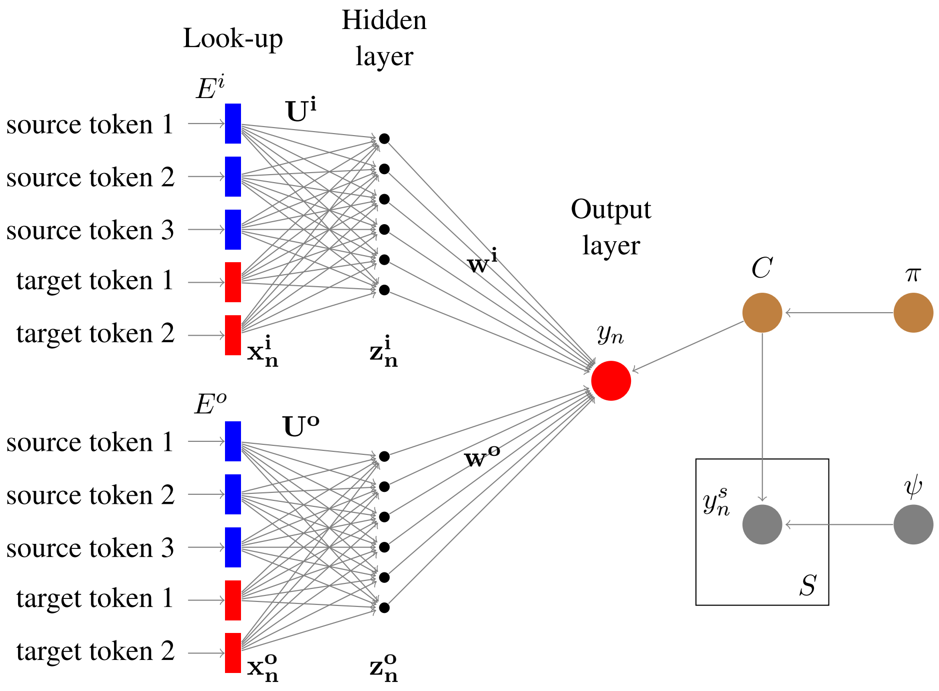

Figure 1: Fusion of two simplified neural network joint models with noise contrastive loss. For illustration, we use 3-gram target words (i.e., 2-words history) and a source context of size 3. The output y_n is shown as a single categorical variable (scalar) as opposed to the traditional one-hot vector representation.

Noise Contrastive Estimation: Like other deep neural models, optimization in NNJM is performed using first-order online methods, such as stochastic gradient ascent (SGA) with standard *backpropagation* algorithm. Unfortunately, training NNJM could be impractically slow using the standard maximum likelihood objective (Eq. 3), because for each training instance $(\mathbf{x_n}, y_n)$, the softmax output layer (Eq. 2) needs to compute a summation over all words in the output vocabulary.

One way to avoid this repetitive computation is to use Noise contrastive estimation or NCE (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012), which adds S noise samples (see Figure 1) for each training instance by sampling from a known noise distribution (e.g., `unigram`). NCE is then defined as such to discriminate a *true* instance from a *noisy* one. Let $C \in \{0, 1\}$ denote the class of an instance with $C = 1$ indicating *true* and $C = 0$ indicating *noise*. NCE maximizes the following conditional log likelihood:

$$J(\theta) = \sum_{n=1}^{N} \Big[log[P(C = 1|y_n, \mathbf{x}_n, \theta)] + \sum_{m=1}^{M} log[P(C = 0|y_n^m, \mathbf{x}_n, \psi)] \Big], \tag{4}$$

which can be simplified as $J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{|V_o|} \Big[\; y_{nk} \; log \; \sigma_{nk} + \sum_{s=1}^{S} y_{nk}^s \; log \; \psi_{nk} \; \Big]$, where σ_{nk}=exp$(\mathbf{w}_k^T \mathbf{z_n})$ is the unnormalized score and ψ_{nk}=$P(y_n^s = k|\mathbf{x}_n, \psi)$ is the noise distribution; y_{nk} and y_{nk}^s are indicator variables indicating the true output and the noise sample, respectively. NCE reduces the number of computations needed at the output layer from $|V_o|$ to $S + 1$, where $S << |V_o|$.

3 Neural Fusion Models

The NNJM model trained from a simple concatenation of large and diverse multi-domain data with in-domain data could be sub-optimal, because the resulting probability distribution can diverge from the target domain hurting the system performance. The goal in domain adaptation is to restrict this drift while still making the best use of the available data. Joty et al. (2015) recently proposed a weighted concatenation approach using data dependent regularizations in the loss function. While it helps, their approach does not fully prevent out-domain data from contaminating the model parameters.

3.1 Fusion Models

We take a different approach of learning in- and out-domain models separately, then fusing them together by readjusting their parameters towards in-domain data.

Fusion Model-I: Let θ^i and θ^o be the parameter sets of the trained in- and out-domain NNJMs, respectively. We combine the two models by redefining the `softmax` output layer (Eq. 2) as follows:

$$P(y_n = k|\mathbf{x_n^i}, \mathbf{x_n^o}, \theta^i, \theta^o) = \frac{\exp\ ([\mathbf{w}_k^i, \mathbf{w}_k^o]^T[\mathbf{z}_n^i, \mathbf{z}_n^o])}{\sum_{m=1}^{|V_o|} \exp\ ([\mathbf{w}_m^i, \mathbf{w}_m^o]^T[\mathbf{z}_n^i, \mathbf{z}_n^o])}$$

where $[\mathbf{w}_k^i, \mathbf{w}_k^o]$ is the concatenation of the output layer weights of in-domain and out-domain models; similarly, $[\mathbf{z}_n^i, \mathbf{z}_n^o]^T$ is the concatenation of the activations at the outermost hidden layer of the two models. Figure 1 demonstrates the fusion process with two simplified NNJMs: (*i*) the in-domain NNJM parameterized by $\theta^i = [E^i, U^i, W^i]$, and (*ii*) the out-domain NNJM parameterized by $\theta^o = [E^o, U^o, W^o]$.[1]

We train this model on the in-domain data using backpropagation on the NCE objective, where each participating model uses its own noise distribution. The errors (i.e., gradients) of the objective with respect to the final layer weight vectors $\mathbf{w}_j^d$ are:

$$\nabla_{\mathbf{w}_j^d} J(\theta) = \sum_{n=1}^{N} \Big[(y_{nj} - \sigma_{nj})\mathbf{z}_n^d \Big] \tag{5}$$

where $d \in \{i, o\}$ in the superscript denotes the domain, $y_{nj} = I(y_n = j)$ is an indicator variable, and $\sigma_{nj} = \exp(\mathbf{w}_j^T \mathbf{z_n^d})$ is the unnormalized score in the output. We train the model by backpropagating these errors from the output layer of the neural network to the word embedding layer (i.e., look-up layer) of each model. Therefore, all the parameters of the participating models (i.e., E, U and W) are fine-tuned on the in-domain data. Such training yields adjusted models that are collectively optimized for the target in-domain data.

Fusion Model-II: A variation of the above model is to tune only the final layer combination weights $[\mathbf{w}_k^i, \mathbf{w}_k^o]$, which can be achieved by restricting the backpropagation only to the outermost hidden layer of the neural network models. This model is faster to train than the above model but could suffer from limited representation power.

3.2 Interpolation

Another approach we explored is to combine in- and out-domain NNJM models through linear interpolation. This approach has been extensively tried out in the literature to interpolate phrase-translation models (Foster and Kuhn, 2007). Several metrics such as tf/idf, LSA or perplexity have been employed for weighting. Here we interpolate multiple NNJM models instead. The mixture weights are computed by optimizing perplexity on the in-domain tuning set[2] using a standard EM-based algorithm as described below:

Model Weighting by EM: Let $\theta_d \in \{\theta_1, \ldots \theta_D\}$ represent an NNJM model trained on domain d, where D is the total number of domains. The probability of a sequence $\mathbf{x}_n$ can be written as a mixture of D probability densities, each coming from a different model:

[1]Although we define the fusion for two NNJM models, this can be easily generalized to multiple models.

[2]The tuning-set is required to be word-aligned and then converted into an augmented streams of source and target strings (for NNJM) to compute model-wise perplexities.

$$P(\mathbf{x}_n|\theta,\lambda) = \sum_{d=1}^{D} P(\mathbf{x}_n|z_n = d, \theta_d)\ \lambda_d$$

where $P(\mathbf{x}_n|z_n = d, \theta_d)$ represents the probability of $\mathbf{x}_n$ assigned by model θ_d, and the mixture weights λ_d satisfy $0 \leq \lambda_d \leq 1$ and $\sum_{d=1}^{D} \lambda_d = 1$. In our setting, $\theta = \{\theta_1, \ldots \theta_D\}$ is known, and we can use EM to learn the mixture weights. The expected complete data log likelihood is given by:

$$E[L(\lambda)] = \sum_{n=1}^{N} \sum_{d=1}^{D} r_{nd}\ log\left[P(\mathbf{x}_n|z_n = d, \theta_d)\lambda_d\right]$$

where $r_{nd} = P(z_n = d|\mathbf{x}_n, \theta_d, \lambda_d^{t-1})$ is the responsibility that domain d takes for data point n given the mixing weight in the previous step λ_d^{t-1}. In the E-step, we compute r_{nd} and we update λ in the M-step. More specifically:

E-step: Compute $r_{nd}^t = \frac{\lambda_d^{t-1} P(\mathbf{x}_n|z_n=d,\theta_d)}{\sum_{d'=1}^{D} \lambda_{d'}^{t-1} P(\mathbf{x}_n|z_n=d,\theta_{d'})}$

M-step: Update $\lambda_d^t = \frac{1}{N} \sum_{n=1}^{N} r_{nd}^t$

Once we have learned the relative weights of the models based on the in-domain tuning data, we can linearly interpolate the models as:

$$P_{nnjm}(T|S) \approx \prod_{i=1}^{|T|} \sum_{d} \lambda_d P(t_i|t_{i-1} \ldots t_{i-n+1}, \mathbf{s_i}, \theta_d)$$

An alternative way to combine the models is through log-linear interpolation by optimizing weights, directly on BLEU, along with other features inside of the SMT pipeline.

Note that both fusion and linear interpolation have the same number of parameters, which is the sum of the size of the base models. In fusion, we readjust all the parameters of the base models (or just the output layer weights in fusion-II), where in linear interpolation, we only learn their mixing weight.

4 Experiments

Data: We experimented with the data made available for the translation task of the International Workshop on Spoken Language Translation IWSLT (Cettolo et al., 2014). We used TED talks as our in-domain ($\approx$ 177K sentences) corpus. For Arabic-to-English, we used the multiUN ($\approx$ 3.7M sentences) (Eisele and Chen, 2010) as our out-domain corpora. For English-to-German, we used data made available ($\approx$ 4.4M sentences) for the 9th Workshop on Machine Translation[3] as our out-domain data. Language models were trained on all the available monolingual data (English: $\approx$ 287.3M and German: $\approx$ 59.5M sentences). Machine translation systems were tuned on concatenation of the dev- and test2010 and evaluated on test2011-2013 datasets. We used *Farasa* (Abdelali et al., 2016) to tokenize Arabic and the default *Moses* tokenizer for English-and German. All data was truecased. See Table 1 for data sizes.

NN Training: The NNJM models were trained using the NPLM[4] toolkit (Vaswani et al., 2013) with the following settings: a target context of 5 words and an aligned source window of 9 words. We restricted source and target side vocabularies to the 20K and 40K most frequent words in the in-domain data.[5] The word vector size D and the hidden layer size were set to 150 and 750, respectively. Training was done using SGD with NCE using 100 noise samples and a mini-batch size of 1000. All models were trained for 15 epochs. Training NN models is expensive.[6] In the interest of time, we therefore reduced the NN

[3]http://www.statmt.org/wmt14/translation-task.html

[4]http://nlg.isi.edu/software/nplm/

[5]Less frequent words are mapped to `unk` class if they were found in the in-domain data or `unk`$_o$ otherwise. Please refer to (Joty et al., 2015) for how the vocabularies from in-domain and out-of-domain are mapped.

[6]Training model with the whole corpus requires roughly 12 days of wall-clock time (18 hours/epoch) to train NNJM models on our machines (on a Linux Ubuntu 12.04.5 LTS running on a 16 Core Intel Xeon E5-2650 2.00Ghz and 64Gb RAM). We ran a baseline experiment with all the data and did not find it better than the system trained on randomly selected subset.

English-German				**Arabic-English**			
Corpus	Sentences	Tok_{EN}	Tok_{DE}	Corpus	Sentences	Tok_{AR}	Tok_{EN}
iwslt	177K	3.5M	3.3M	iwslt	186K	2.7M	1.8M
news	200K	5.0M	5.1M	un	3.7M	12.4M	12.3M
ep	1.9M	51.0M	48.7M	-	-	-	-
cc	2.3M	57.5M	53.9M	-	-	-	-
Test Set	Sent.	Tok_{EN}	Tok_{DE}	Corpus	Sent.	Tok_{AR}	Tok_{EN}
tune	2437	51K	48K	tune	2456	48K	52K
test-11	1433	4K	23K	test-11	1199	21K	24K
test-12	1700	28K	26K	test-12	1702	30K	32K
test-13	993	18K	17K	test-13	1169	26K	28K

Table 1: Statistics of the English-German and Arabic-English training corpora in terms of Sentences and Tokens (represented in millions). ep = Europarl, cc = Common Crawl, un = United Nations

training to a subset of 1 million sentences containing all the in-domain data and a random selection of sentences from the out-domain data.

Machine Translation Settings: We trained a Moses system (Koehn et al., 2007), with the settings described in (Birch et al., 2014): a maximum sentence length of 80, Fast-Aligner for word-alignments (Dyer et al., 2013), an interpolated Kneser-Ney smoothed 5-gram language model (Heafield, 2011), lexicalized reordering model (Galley and Manning, 2008), a 5-gram OSM (Durrani et al., 2015b) and other defaults. We used k-best batch MIRA (Cherry and Foster, 2012) for tuning. Arabic OOVs were transliterated using unsupervised transliteration module (Durrani et al., 2014) in Moses.

Baselines: Baseline MT systems were trained by simply concatenating all the data. We included NNJM model trained on a plain concatenation of the data as a feature in our baseline system. In the adapted systems, we either replaced it with the NDAM models trained on weighted concatenation, or with our fusion models (NFM*), where models are trained independently and adjusted towards in-domain data or by interpolating them linearly (EM-weighting) or log-linearly. We also tried the approach of Luong and Manning (2015) by *Fine Tuning* baseline model towards in-domain data (i.e., by training the neural network model for additional epochs on in-domain data). We trained for 10 more epochs.

Phrase-table Adaptation: We also compared performance of our models against state-of-the-art model adaptation techniques available in Moses. Rather than training the MT systems on concatenated data, we train phrase-tables from in- and out-domain data separately and combine them through *Linear Phrase-table Interpolation*, *Instance Weighting* and *Fill-up* methods.[7]

Data Selection: Finally, we compared our system against MML-filtering (Axelrod et al., 2011), although this technique falls in the array of data-selection methods unlike our method which is a model weighting technique. We selected 0%, 2.5%, 5%, 10%, 20%, 40% and 100% out-domain data, and trained MT systems by concatenating the selected data with the in-domain data. The optimal thresholds were found to be 20% and only 5% in English-German and Arabic-English[8] respectively.

Comparing with Neural Model Weighting: The first row in Table 2 shows results for the baseline system, which includes NNJM trained on plain concatenation. The next set of rows show results for systems, where the NNJM model is adapted using weighted concatenation (NDAM), by interpolating (linearly or log-linearly) in- and out-domain models, by fine tuning the baseline model by running epochs on in-domain data only. This method gave significantly better results on the English-German task, but was not found effective in the Arabic-English direction. Next rows show results of our neural fusion

[7]Word alignment is still carried on the concatenated data.

[8]Sajjad et al. (2013) selected 3% of the UN data in their best competition grade system.

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Baseline (NNJM)	27.3	22.9	24.5	24.9		26.1	29.4	30.5	28.7	
NDAM	27.5	23.4	25.1	25.3	+0.4	26.1	29.6	30.9	28.9	+0.2
Linear	27.2	23.5	25.0	25.3	+0.4	26.7	30.2	30.3	29.1	+0.4
Log-Linear	27.0	23.8	25.2	25.3	+0.4	26.4	30.0	30.5	29.0	+0.3
Fine Tuning	27.7	23.9	25.3	25.6	+0.7	26.1	29.6	30.9	28.9	+0.2
NFM-I	27.8	24.1	25.6	25.8	+0.9	26.9	30.2	31.1	29.4	+0.7
NFM-II	27.5	23.9	25.4	25.6	+0.7	26.7	30.0	31.0	29.2	+0.5

Table 2: Comparing with Neural Model Weighting Methods – NDAM (Joty et al. 2015), Linear (Durrani et al. 2015), Fine Tuning (Luong and Manning 2015), NFM* = Neural Fusion Models (this work)

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Baseline (NNJM)	27.3	22.9	24.5	24.9		26.1	29.4	30.5	28.7	
PT Interpolation	27.5	23.2	24.8	25.2	+ 0.3	26.4	29.9	30.3	28.9	+0.2
Instance Wt.	27.3	23.4	25.1	25.3	+ 0.4	26.9	30.3	30.3	29.2	+0.5
Fill Up	27.3	23.4	24.6	25.1	+ 0.2	26.4	29.7	30.4	28.9	+0.2
NFM-I	27.8	24.1	25.6	25.8	+0.9	26.9	30.2	31.1	29.4	+0.7
NFM-I + Instance Wt.	28.0	23.8	25.3	25.7	+0.8	27.5	30.7	30.8	29.7	+1.0

Table 3: Comparing with Translation Model Adaptation Techniques – Linear Interpolation and Instance Weighting (Senrich 2012), Fill-up Method (Bisazza et al. 2011), NFM = Neural Fusion Model

models (NFM*). Our models significantly[9] outperformed the baseline system and were also found better than other neural adapted models. We found that our model gets higher weights than the baseline NNJM model (0.054 compared to 0.047 in En-De and 0.043 compared to 0.040 in Ar-En) in MERT training.

Fusion Model-I (NFM-I), which performs deeper fusion (i.e., till the embedding layer) worked better than Fusion Model-II (NFM-II), where backpropagation is restricted only to the out-most hidden layer, showing that there is an additional value in doing a deeper fusion.

Comparing with Phrase-table Weighting: Next we compare our model with the domain adaptation methods available in Moses (See Table 3). Here instead of adapting the NNJM model, we perform adaptation on translation-tables, by interpolating in- and out-domain phrase-tables, through instance weighting or through fill-up method. Instance weighting method gave best improvements among the lot, however, our fusion model outperform these methods in both language pairs and in most of the test-sets. Additional experiments combining phrase-table adaptation and fusion model found gains to be additive in Arabic-to-English language pair. But no further improvements were observed in English-to-German, except for test2011.

Comparing with Data Selection: In Table 4 we experiment with MML-based filtering and probe whether our model can also improve on top of data selection. Firstly, selecting no out-domain data degrades the English-to-German system. On the contrary, the Arabic-to-English system substantially improves. This shows that general domain data is useful for English-to-German and much of the out-domain data (UN corpus) used in these experiments is harmful in the case of Arabic-to-English. In comparison, data selection was found to be less useful in the case of English-to-German. But we found

[9] $p < 0.05$ using bootstrap resampling (Koehn, 2004), with 1000 samples.

	English-to-German				Arabic-to-English			
System	tst11	tst12	tst13	Avg	tst11	tst12	tst13	Avg
Baseline$_{cat}$	27.3	22.9	24.5	24.9	26.1	29.4	30.5	28.7
Baseline$_{ID}$	26.7	22.5	23.6	24.3	27.2	30.0	30.2	29.1
MML	26.9	22.9	24.4	24.7	27.4	30.8	30.9	29.7
+NFM-I	27.6	23.1	25.0	25.2	27.6	31.2	31.1	30.0

Table 4: Comparing with MML (Axelrod et al 2011)

that using our fusion model instead of baseline NNJM in either system still gave improvements (+0.5 and +0.3 in English-German and Arabic-English, respectively).

5 Related Work

Previous work on domain adaptation in MT can be broken down broadly into two main categories namely *data selection* and *model adaptation*.

Data selection has shown to be an effective way to discard poor quality or irrelevant training instances, which when included in an MT system, hurts its performance. Selection based methods can be helpful to reduce computational cost when training is expensive and also when memory is constrained. Data selection was done earlier for language modeling using information retrieval techniques (Hildebrand et al., 2005) and perplexity measures (Moore and Lewis, 2010). Axelrod et al. (2011) further extended the work of Moore and Lewis (2010) to translation model adaptation by using both source- and target-side language models. Duh et al. (2013) used a recurrent neural language model instead of an ngram-based language model to do the same. Translation model features were used recently by (Liu et al., 2014; Hoang and Sima'an, 2014) for data selection. Durrani et al. (2015a) performed data selection using operation sequence model and NNJM models.

An alternative to completely filtering out less useful data is to minimize its effect by down-weighting it. It is more robust than selection since it takes advantage of the complete out-domain data with intelligent weighting towards the in-domain. Our work falls in this line of research. Matsoukas et al. (2009) proposed a classification-based sentence weighting method for adaptation. Foster et al. (2010) extended this by weighting phrases rather than sentence pairs. Other researchers have carried out weighting by merging phrase-tables through linear interpolation (Finch and Sumita, 2008; Nakov and Ng, 2009) or log-linear combination (Foster and Kuhn, 2009; Bisazza et al., 2011; Sennrich, 2012) and through phrase training based adaptation (Mansour and Ney, 2013). Chen et al. (2013) used a vector space model for adaptation at the phrase level. Every phrase pair is represented as a vector, where every entry in the vector reflects its relatedness with each domain.

Other work on domain adaptation includes but not limited to studies focusing on topic models (Eidelman et al., 2012; Hasler et al., 2014), dynamic adaptation without in-domain data (Sennrich et al., 2013; Mathur et al., 2014) and sense disambiguation (Carpuat et al., 2013).

6 Conclusion and Future Work

We presented a deep fusion model based on the neural network joint model (NNJM) of Devlin et al. (2014). The model is learned by fusing in- and out-domain NNJM models into a composite model by adjusting their parameters in favor of the in-domain data. When used as a feature during decoding, our model obtains statistically significant improvements on top of a competition grade phrase-based baseline system. We also showed improvements compared to previous adaptation methods. Further gains were obtained when our models were combined with existing methods. Although this study focused on fusing multiple neural models for domain adaptation in phrase-based SMT, the central idea can be adopted in the end-to-end NMT systems (Bahdanau et al., 2015), where the goal will be to fuse multiple NMT systems. We intend to explore this idea in our future work.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California, June. Association for Computational Linguistics.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, Edinburgh, United Kingdom.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Alexandra Birch, Matthias Huck, Nadir Durrani, Nikolay Bogoychev, and Philipp Koehn. 2014. Edinburgh SLT and MT system description for the IWSLT 2014 evaluation. In *Proceedings of the 11th International Workshop on Spoken Language Translation*, IWSLT '14, Lake Tahoe, CA, USA.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143.

Marine Carpuat, Hal Daume III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign. *Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US.*

Boxing Chen, Roland Kuhn, and George Foster. 2013. Vector space model for adaptation in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '12, Montréal, Canada.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, August.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an Unsupervised Transliteration Model into Statistical Machine Translation. In *Proceedings of the 15th Conference of the European Chapter of the ACL (EACL 2014)*, Gothenburg, Sweden, April.

Nadir Durrani, Hassan Sajjad, Shafiq Joty, Ahmed Abdelali, and Stephan Vogel. 2015a. Using joint models for domain adaptation in statistical machine translation. In *Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV)*, Florida, USA, To Appear. AMTA.

Nadir Durrani, Helmut Schmid, Alexander Fraser, Philipp Koehn, and Hinrich Schütze. 2015b. The Operation Sequence Model – Combining N-Gram-based and Phrase-based Statistical Machine Translation. *Computational Linguistics*, 41(2):157–186.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL'13.*

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Jeju Island, Korea, July.

Andreas Eisele and Yu Chen. 2010. MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, Valleta, Malta, May.

Andrew Finch and Eiichiro Sumita. 2008. Dynamic model interpolation for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio, June.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07.

George Foster and Roland Kuhn. 2009. Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, Athens, Greece.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2010. Overview of the patent translation task at the ntcir-8 workshop. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pages 293–302.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y.W. Teh and M. Titterington, editors, *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 297–304.

Francisco Guzmán, Hassan Sajjad, Stephan Vogel, and Ahmed Abdelali. 2013. The AMARA corpus: Building resources for translating the web's educational content. In *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based mt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden, April.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, Budapest, May.

Cuong Hoang and Khalil Sima'an. 2014. Latent domain translation models in mix-of-domains haystack. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland.*

Shafiq Joty, Hassan Sajjad, Nadir Durrani, Kamla Al-Mannai, Ahmed Abdelali, and Stephan Vogel. 2015. How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, September.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, Barcelona, Spain.

Le Liu, Yu Hong, Hao Liu, Xing Wang, and Jianmin Yao. 2014. Effective selection of translation model training data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland, June.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Saab Mansour and Hermann Ney. 2013. Phrase training based adaptation for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, June.

Prashant Mathur, Sriram Venkatapathy, and Nicola Cancedda. 2014. Fast domain adaptation of smt models without in-domain parallel data. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland, August.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, EMNLP '09.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the Association for Computational Linguistics (ACL'10)*, Uppsala, Sweden.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*, Singapore.

Hassan Sajjad, Francisco Guzmn, Preslav Nakov, Ahmed Abdelali, Kenton Murray, Fahad Al Obaidli, and Stephan Vogel. 2013. QCRI at IWSLT 2013: Experiments in Arabic-English and English-Arabic spoken language translation. In *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, August.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, April.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.