Available online at www.sciencedirect.com

ScienceDirect

Computer Speech & Language 45 (2017) 161–179

www.elsevier.com/locate/csl

Domain adaptation using neural network joint model

Shafiq Joty, Nadir Durrani*, Hassan Sajjad, Ahmed Abdelali

Arabic Language Technologies, Qatar Computing Research Institute, HBKU, Qatar

Received 6 May 2016; received in revised form 10 November 2016; accepted 7 December 2016

Abstract

We explore neural joint models for the task of domain adaptation in machine translation in two ways: (*i*) we apply state-of-the-art domain adaptation techniques, such as mixture modelling and data selection using the recently proposed Neural Network Joint Model (NNJM) (Devlin et al., 2014); (*ii*) we propose two novel approaches to perform adaptation through instance weighting and weight readjustment in the NNJM framework. In our first approach, we propose a pair of models called Neural Domain Adaptation Models (NDAM) that minimizes the cross entropy by regularizing the loss function with respect to in-domain (and optionally to out-domain) model. In the second approach, we present a set of Neural Fusion Models (NFM) that combines the in- and the out-domain models by readjusting their parameters based on the in-domain data.

We evaluated our models on the standard task of translating English-to-German and Arabic-to-English TED talks. The NDAM models achieved better perplexities and modest BLEU improvements compared to the baseline NNJM, trained either on in-domain or on a concatenation of in- and out-domain data. On the other hand, the NFM models obtained significant improvements of up to +0.9 and +0.7 BLEU points, respectively. We also demonstrate improvements over existing adaptation methods such as instance weighting, phrasetable fill-up, linear and log-linear interpolations.

Keywords: Machine translation; Domain adaptation; Neural network joint model; Distributed representation of texts; Noise contrastive estimation

1. Introduction

Parallel data required to train Statistical Machine Translation (SMT) systems is often inadequate as it is typically collected opportunistically from wherever available (Koehn and Schroeder, 2007). The conventional wisdom is that more data improves the translation quality. Additional data however, may not be best suited for tasks such as translating TED talks (Cettolo et al., 2014), patents (Fujii et al., 2010) and educational content (Abdelali et al., 2014), that posit the challenges of dealing with word-sense ambiguities and stylistic variance of other genres. When additional data, later referred as *out-domain* data, is much larger than *in-domain* data, the resultant distribution can get biased towards out-domain, yielding a sub-optimal system. For example, an Arabic-to-English SMT system trained by simply concatenating in- and out-domain data translates the Arabic phrase "عن مشكلة الحمل الزائد للاختيار", to "about the problem of unwanted pregnancy". This translation is inaccurate in the context of the in-domain data,

* Corresponding author.
E-mail address: sjoty@qf.org.qa (S. Joty), ndurrani@qf.org.qa (N. Durrani), hsajjad@qf.org.qa (H. Sajjad), aabdelali@qf.org.qa (A. Abdelali).

http://dx.doi.org/10.1016/j.csl.2016.12.006

where it should be translated to "about the problem of choice overload". The sense of the Arabic phrase taken from out-domain data completely changes the meaning of the sentence.

Domain adaptation aims to preserve the identity of the in-domain data while exploiting the out-domain data in favor of the in-domain data and avoid possible drift towards out-domain jargon and style. This is typically done either by selecting a subset from the out-domain data, which is closer to the in-domain (Matsoukas et al., 2009; Moore and Lewis, 2010), or by reweighting the probability distribution in favor of the in-domain data (Foster and Kuhn, 2007; Sennrich, 2012).

Joint sequence ngram-based models (Mariño et al., 2006; Durrani et al., 2013) have shown to be effective in improving the quality of machine translation and have achieved state-of-the-art performance recently. Their ability to capture non-local dependencies makes them superior to the traditional models, which do not consider contextual information across phrasal boundaries. Such models however suffer from data sparsity. As the length of the sequence increases, the test sequences are likely to be different from the ones used for training the models. To overcome this problem, a transition towards continuous space modeling using neural networks has been proposed (Devlin et al., 2014; Bengio et al., 2003; Le et al., 2012). In this framework, a distributed representation is learned for each word in the process of modeling the word sequences.

We hypothesize that the distributed vector representation of neural models helps to bridge the lexical differences between the in-domain and out-domain data, and adaptation is necessary to avoid deviation and drift of the model from the in-domain, which otherwise happens because of the large out-domain data.

In this paper, we explore Neural Network Joint Model (NNJM) proposed by Devlin et al. (2014) for the task of domain adaptation in Statistical Machine Translation (SMT). Preliminarily, we customize state-of-the-art methods in domain adaptation, such as mixture modelling (Foster and Kuhn, 2007) and MML (Axelrod et al., 2011) to be used with NNJM. We train NNJM models from in- and out-domain data individually and interpolate them linearly or log-linearly to perform adaptation. We also tried NNJM-based data selection similar to MML filtering. Later, we propose two sets of novel models based on the NNJM framework: (*i*) Neural Domain Adaptation Models (NDAM), where we minimize the cross entropy by regularizing the loss function with respect to in-domain (and optionally to out-domain) model(s); (*ii*) Neural Fusion Models (NFM), where we combine in- and out-domain models and readjust their parameters to minimize the loss on the in-domain sequences.

The NDAM models use data dependent regularizations in their loss functions to perform instance weighting. In our first NDAM model (NDAM-v1), we use a regularizer based on an in-domain model to bias the resultant model towards the in-domain data. In the second model (NDAM-v2), we additionally use an out-domain model to penalize out-domain sequences that are similar to the out-domain data. The regularizers in our loss functions are inspired from the data selection methods proposed in Moore and Lewis (2010); Axelrod et al. (2011).

In the NFM models we train in- and out-domain NNJM models and fuse them by readjusting their parameters towards in-domain data. This is achieved by backpropogating errors from the output layer to the word embedding layer of each model. In a variant of the NFM model, we restrict backpropogation to only the outermost hidden layer and adjust only the final layer combination weights.

We evaluated our models against strong baselines on a standard task of translating IWSLT TED talks for English-to-German (EN-DE) and Arabic-to-English (AR-EN) language pairs using BLEU (Papineni et al., 2002). The baseline MT system uses an NNJM trained on a concatenation of in- and out-domain data ($NNJM_c$). In the adapted models we simply replace the baseline NNJM model with our adapted versions. The most relevant baseline to our work is the fine-tuning method proposed in Luong and Manning (2015). This method first learns a neural model on the concatenated data, then trains it further on the in-domain data to tune the model towards in-domain. We applied fine-tuning to the NNJM model and report it as an additional baseline.

We also compared our models against state-of-the-art model adaptation techniques including phrase-table fill-up (Bisazza et al., 2011), phrasetable interpolation and instance (phrase-level) weighting (Foster and Kuhn, 2007; Sennrich, 2012), and also against existing data selection methods like Modified-Moore-Lewis (MML) (Axelrod et al., 2011). Below is a summary of our main findings:

- The NDAM models gave an average improvement of +0.4 BLEU points over the best baseline. NDAM-v1 performed better on English-to-German and NDAM-v2 performs better on Arabic-to-English.
- The NNJM mixture models also gave average improvements of up to +0.4 BLEU points. Log-linear interpolation performed slightly worse than linear interpolation on Arabic-to-English.

- Fine-tuning NNJM using in-domain data, was found to be quite effective for English-to-German pair but not for the Arabic−English direction.
- The neural fusion models (NFM) yielded the best improvements of up to +0.9 BLEU points in English-to-German and +0.7 BLEU points in Arabic-to-English. The NFM variation which tunes only the final-layer weights, performed slightly worse than its deeper variant.
- Our fusion models also outperformed state-of-the-art translation model adaptation techniques such as linear interpolation of phrase-tables, instance weighting, and phrase-table fill-up methods. The gains were found to be additive in the case of Arabic-to-English.
- Data selection based on NNJM model performed on par with MML based selection, demonstrating occasional gains.
- We further demonstrated that the performance of our models is additive to data selection.

This article builds upon two previous conference papers: (*i*) Joty et al. (2015) which presented the NDAM models, and (*ii*) Durrani et al. (2015a), which presented interpolation of NNJMs. Compared to these papers, the main novelties introduced in this paper are: (*i*) the fusion models (the best performing models) to combine multiple NNJMs, and (*ii*) more comparisons and analysis of the results, which gives us more insights of the models.

The rest of the paper is organized as follows. Section 2 revisits the NNJM model. Section 3 gives an account on existing de facto domain adaptation methods and our customization to enable them with the NNJM model. Sections 3.2 and 3.3 present novel neural models for domain adaptation. Section 4 provide implementation details of our models. Section 5 gives details on experimental setup and results. Section 6 gives a brief account on the related work and Section 7 concludes the paper with future directions.

2. Neural network joint model

There has been a great deal of effort dedicated to neural networks and word embeddings in recent years with applications to SMT and other natural language processing tasks (Bengio et al., 2003; Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Gao et al., 2014; Schwenk, 2012; Collobert et al., 2011; Mikolov et al., 2013a; Socher et al., 2013; Hinton et al., 2012).

Neural models are fast 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: (*i*) to use the neural model as a feature inside the SMT decoder (Devlin et al., 2014; Vaswani et al., 2013), and (*ii*) to build an end-to-end translation system (Luong and Manning, 2015; Bahdanau et al., 2015; Sennrich et al., 2016) designed as a 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.

Devlin et al. (2014) recently proposed a neural bilingual model called Neural Network Joint Model (NNJM) and integrated it into the SMT decoder as an additional feature. We use the NNJM as the base model for our domain adaptation work. In the following, we revisit the NNJM briefly.

2.1. Model

Given a source sentence S and its corresponding target sentence T, the NNJM model computes the conditional probability $P(T|S)$ as follows:

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

where, $\mathbf{s_i}$ is a q-word source window for the target word t_i based on the one-to-one (non-NULL) alignment of T to S. Notice that this is essentially a $(p+q)$-gram neural network LM (NNLM) originally proposed by Bengio et al. (2003). Fig. 1 shows a simplified NNJM with a target context window of 2-words and a source context window of 3-words. In this example, the target word being modeled is *choice*, and the source context contains its aligned word للاختيار and two of its neighboring words, الزائد and الحمل. The target-side context contains *problem* and *of*, the two previous words of the currently modeled *choice*.

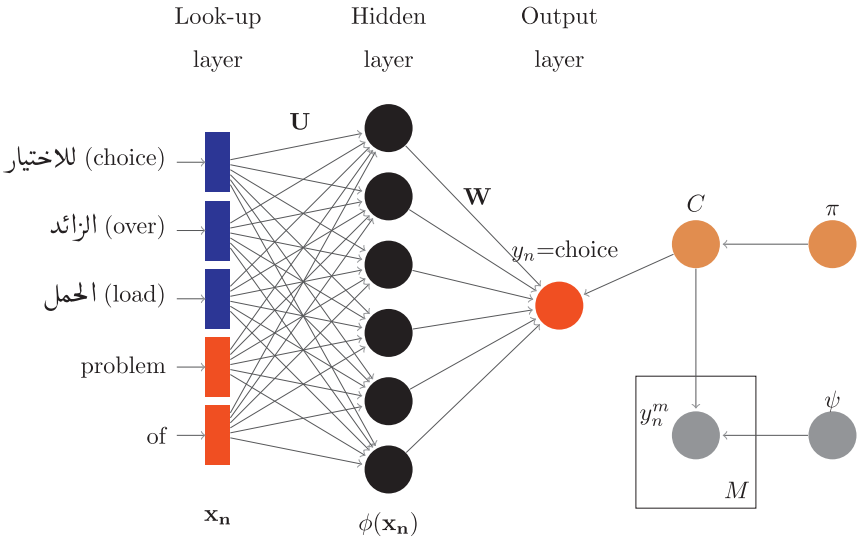

Fig. 1. A simplified neural network joint model with noise contrastive loss. We use a target history of 2 words and a source context of 3 words. For illustration, the output y_n is shown as a single categorical variable (scalar) as opposed to the traditional one-hot vector representation.

In the model each input word (source or target side) in the context is represented by a D dimensional vector in the shared look-up layer $L \in \mathbb{R}^{|V_i| \times D}$, where V_i is the input vocabulary; L is considered as a model parameter to be learned. The look-up layer then creates a context vector $\mathbf{x_n}$ representing the context words of the $(p+q)$-gram sequence by concatenating their respective vectors in L. The concatenated vector is then passed through the non-linear hidden layers to learn a high-level representation, which is in turn fed to the output layer. The output layer has a softmax activation defined over the output vocabulary V_o of target words. Formally, the probability of getting kth

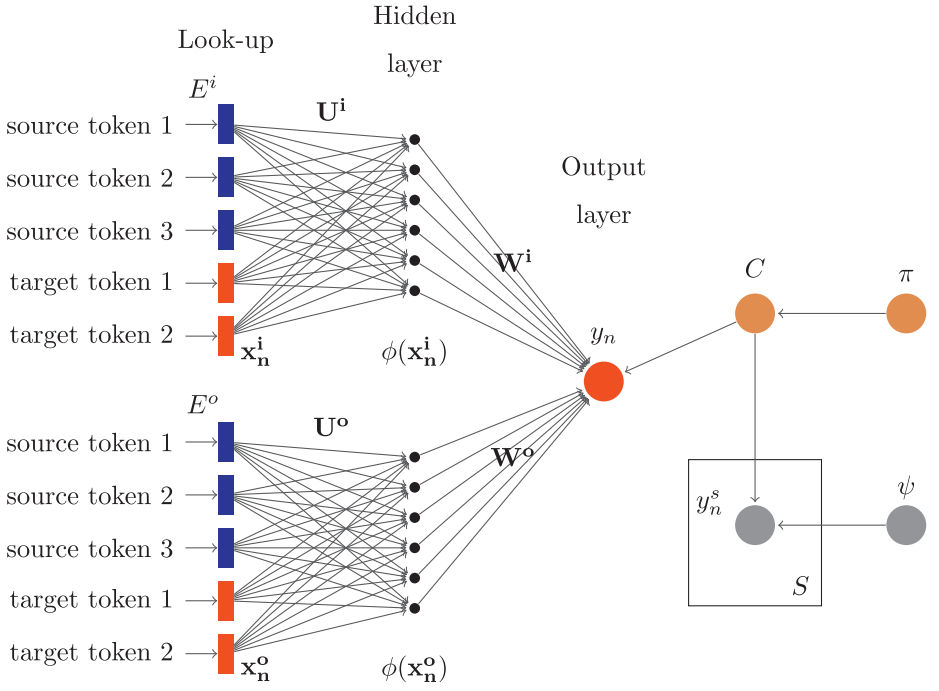

Fig. 2. Fusion of two simplified neural network joint models with noise contrastive loss. We use 3-gram target words (i.e., 2-words history) and a source context of size 3. For illustration, the output y_n is shown as a single categorical variable (scalar) as opposed to the traditional one-hot vector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

word in the output given the context $\mathbf{x_n}$ can be written as:

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

where $\mathbf{z}_n = \boldsymbol{\varphi}(\mathbf{x_n})$ defines the transformations of $\mathbf{x_n}$ through the hidden layers, and $\mathbf{w}_k$ are the weights from the last hidden layer to the output layer. For notational simplicity, henceforth we will use $(\mathbf{x_n}, y_n)$ to represent a training sequence. By setting p and q to be sufficiently large, NNJM can capture long-range cross-lingual dependencies between words, while still overcoming the data sparseness issue by virtue of its distributed representations (i.e., word vectors). A similar model was earlier proposed by Le et al. (2012), although they reorder the source to be in the target order, and use the model only for re-ranking.

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

2.2. Noise contrastive estimation

The standard way to train neural network language models (NNLMs) is to minimize the negative log likelihood (or cross entropy) of the training data:

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

where, $y_{nk} = I(y_n = k)$ is an indicator variable (i.e., $y_{nk} = 1$ when $y_n = k$, otherwise 0). Optimization is typically performed using first-order (gradient-based) online methods, such as stochastic gradient descend with the standard *backpropagation* algorithm. Unfortunately, training NNLMs are impractically slow because for each training instance $(\mathbf{x_n}, y_n)$, the softmax output layer (see Eq. (2)) needs to compute a summation over all words in the output vocabulary.[1] Noise contrastive estimation or NCE (Gutmann and Hyvärinen, 2010) provides an efficient and stable way to avoid this repetitive computation, which has recently been applied to different NNLMs (Vaswani et al., 2013; Mnih and Teh, 2012). We can re-write Eq. (2) as follows:

$$P(y_n = k|\mathbf{x_n}, \theta) = \frac{\sigma(y_n = k|\mathbf{x_n}, \theta)}{Z\Big(\boldsymbol{\varphi}(\mathbf{x_n}), \mathbf{W}\Big)} \tag{4}$$

where $\sigma(.)$ is the unnormalized score and $Z(.)$ is the normalization factor. In NCE, we consider $Z(.)$ as an additional model parameter along with the regular parameters, i.e., weights, look-up vectors. Notice that $Z(.)$ depends on the input context, $\mathbf{x_n}$, which means one needs to compute it for every possible input context in our training data, making it difficult to scale to large number of contexts encountered with large context sizes. Fortunately, it has been found that fixing $Z(.)$ to 1 instead of learning it during training does not affect the model performance (Mnih and Teh, 2012). Hence, we set $Z(.)$ to 1 in training our models, which has now become a common practice (Vaswani et al., 2013; Mikolov et al., 2013a).

In training with NCE, for each training instance $(\mathbf{x_n}, y_n)$, we add M noise samples $\{(\mathbf{x_n}, y_n^m)\}_{m=1}^{M}$ by sampling y_n^m from a known noise distribution ψ (e.g., unigram, uniform); see the plate notation in Fig. 1. NCE loss 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 minimizes the following conditional negative log likelihood:

$$J(\theta) \quad = -\sum_{n=1}^{N} \left[\log[P(C = 1|y_n, \mathbf{x}_n, \theta)] + \sum_{m=1}^{M} \log[P(C = 0|y_n^m, \mathbf{x}_n, \psi)] \right] \tag{5}$$

[1] This would take few weeks for a modern CPU machine to train a single NNJM model on a moderately sized parallel training corpus.

$$= \sum_{n=1}^{N} [\log[P(y_n|C=1,\mathbf{x}_n,\theta)P(C=1|\pi)] + \sum_{m=1}^{M} \log\big[\big(P(y_n^m|C=0,\mathbf{x}_n,\psi)\big)P(C=0|\pi)\big] - (M+1)\log Q] \quad (6)$$

where $Q = P(y_n, C=1|\mathbf{x}_n,\theta,\pi) + P(y_n^m, C=0|\mathbf{x}_n,\psi,\pi)$ is a normalization constant. After removing the constant terms, Eq. 6 can be simplified as:

$$J(\theta) = -\sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\left[\ y_{nk}\ \log\sigma_{nk} + \sum_{m=1}^{M} y_{nk}^m\ \log\psi_{nk}\ \right] \quad (7)$$

where $\psi_{nk} = P(y_n^m = k|\mathbf{x}_n,\psi)$ is the noise distribution, $\sigma_{nk} = \sigma(y_n = k|\mathbf{x}_n,\theta)$ is the unnormalized score at the output layer (Eq. (4)), and y_{nk} and y_{nk}^m are indicator variables as defined before. NCE reduces the number of computations needed at the output layer from $|V_o|$ to $M+1$, where M is a small number in compared to the output vocabulary size $|V_o|$. In all our experiments we use NCE loss with $M = 100$ samples in accordance to Mnih and Teh (2012).

3. Domain adaptation with neural network joint model

While additional data is often beneficial for a general purpose SMT system, an MT system trained on such corpora may not be optimal for translating texts in new domains such as medical, legal or lecture. MT systems trained from a simple concatenation of small in-domain and large out-domain data often perform below par because the out-domain data is often distant or overwhelmingly larger than the in-domain data. Additional data increases lexical ambiguity by introducing new senses to the existing in-domain vocabulary. In domain adaptation, our goal is to best exploit the out-domain data, while preserving the specificity of the in-domain data, thus avoiding possible drifts toward out-domain jargon and style.

The ability to generalize and learn complex semantic relationships (Mikolov et al., 2013b) gives a strong motivation to use the NNJM for the task of domain adaptation in machine translation. Initially we explore the existing methods for domain adaptation with NNJM, namely data selection and mixture modelling. Then we propose novel neural models for domain adaptation in SMT. In the interest of coherence and similarity of methods, we describe the methods in the following order: data selection using NNJM in Section 3.1, regularized neural adaptation models (NDAMs) in Section 3.2, and finally mixture modeling with linear interpolation and neural fusion models (NFM) in Section 3.3.

3.1. Data selection

The data selection method in domain adaptation attempts to filter out harmful data from the training corpora. A number of data selection methods have been proposed for SMT (Moore and Lewis, 2010; Axelrod et al., 2011). We use difference in cross entropies proposed in Moore and Lewis (2010). More specifically, we first train an NNJM model on the in-domain corpus, and then train another NNJM model on the out-domain data. Then we score the out-domain (parallel) sentences as follows:

$$score(s,t) = H_I(s,t) - H_O(s,t) \quad (8)$$

where $(s,\ t)$ constitutes the bilingual sequences achieved by augmenting stream of source and target sequences extracted from the bilingual sentence pairs and their alignments. H_D is the cross-entropy between the NNJM model and an empirical n-gram distribution in domain $D \in \{I,\ O\}$.

We select the top $P\%$ lowest-scoring sentences to train our models. The selected data can be used to train a complete SMT system and/or to train a new NNJM model to be used as an additional feature in the decoder.

The bilingual characteristic of the NNJM model makes our method comparable to the modified Moore-Lewis (MML) selection method of Axelrod et al. (2011), which trains source- and target-side *n-gram* language models separately from the in- and out-domains and takes a sum of cross-entropy differences over each side of the corpus. The advantages of using NNJM over MML for data selection are: (*i*) NNJM captures semantic similarity by virtue of its distributed continuous representation as opposed to the discrete (n-gram) representation of MML, and (*ii*) NNJM scores source and target

sequences by modelling their dependencies jointly, where MML does this disjointly, and therefore, may not be that effective. The disadvantage, however, is that the time require to train a neural model is significantly more.

Data selection could be useful in training scenarios with memory constraints. However, a down-side of this approach is that it requires extensive amount of experimentation to find an optimal cut-off point P. An alternative to data selection is model weighting, where we use all the data, but skew the probability distribution towards in-domain data. In the following subsections, we present three novel adaptation techniques based on the NNJM model.

3.2. Regularized Neural Adaptation Models

Our first approach is to learn an adapted model from the complete data but to use a data dependent regularization to restrict it from deviating towards the out-domain data. Based upon this idea, we propose two novel extensions of the NNJM model for domain adaptation, which we call Neural Domain Adaptation Models (NDAM). Our models add regularization to its loss function either with respect to the in-domain only or both in- and out-domains. For both models, we first present the regularized loss function using the standard softmax (normalized) output, then the corresponding unnormalized NCE output.

3.2.1. Adaptation by Regularizing with respect to In-domain

In our first neural adaptation model NDAM-v1 (NDAM version 1), we use a regularizer (or prior) based on the in-domain model to bias the resultant model towards the in-domain data. Let θ_i be an NNJM model already trained on the in-domain data. We train an adapted model θ_a on the complete in- and out-domain data, but regularizing it with respect to the in-domain model θ_i. Formally, we redefine the normalized loss function of Eq. (3) as follows:

$$J(\theta_a) = -\sum_{n=1}^{N}\sum_{k=1}^{|V_o|}[\lambda\ y_{nk}\log P(y_n=k|\mathbf{x_n},\theta_a) + (1-\lambda)y_{nk} \quad P(y_n=k|\mathbf{x_n},\theta_i)\ \log P(y_n=k|\mathbf{x_n},\theta_a)] \tag{9}$$

$$= -\sum_{n=1}^{N}\sum_{k=1}^{|V_o|}[\lambda\ y_{nk}\log \widehat{y}_{nk}(\theta_a) + (1-\lambda)\ y_{nk}\ p_{nk}(\theta_i)\log \widehat{y}_{nk}(\theta_a)] \tag{10}$$

where $\widehat{y}_{nk}(\theta_a)$ is the softmax output and $p_{nk}(\theta_i)$ is the probability of the training instance according to the in-domain model θ_i. Notice that the loss function minimizes the cross entropy or Kullback–Leibler (KL) divergence of the current model θ_a with respect to the gold labels y_n and to the in-domain model θ_i. The mixing parameter $\lambda \in [0, 1]$ determines the relative strength of the two components.[2] Similarly, we can re-define the NCE loss of Eq. (7) as:

$$J(\theta_a) = -\sum_{n=1}^{N}\sum_{k=1}^{|V_o|}[\lambda\ y_{nk}\log\sigma_{nk} + (1-\lambda)\ y_{nk}p_{nk}(\theta_i)\log\sigma_{nk} + \sum_{m=1}^{M} y_{nk}^{m}\log\psi_{nk}] \tag{11}$$

We use stochastic gradient descend with backpropagation to train this model. The derivatives of $J(\theta_a)$ with respect to the final layer weight vectors $\mathbf{w}_j$ are:

$$\nabla_{\mathbf{w}_j} J(\theta_a) = -\sum_{n=1}^{N}\left[\lambda\ (y_{nj}-\sigma_{nj}) + (1-\lambda)[p_{nj}(\theta_i) - \sum_{k} y_{nk}\ p_{nk}(\theta_i)\ \sigma_{nj}]\right] \tag{12}$$

3.2.2. Adaptation by regularizing with respect to in- and out-domains

The regularizer in NDAM-v1 is based on an in-domain model θ_i, which puts higher weights to the training instances (i.e., n-gram sequences) that are similar to the in-domain ones. This might work better when the out-domain data is similar to the in-domain data. In cases where the out-domain data is different, we might want to build a more conservative model that penalizes training instances for being similar to the out-domain ones. Our second adaptation model NDAM-v2 (NDAM version 2) is based upon this hypothesis.

[2] We used a balanced value $\lambda = 0.5$ for our experiments.

Let θ_i and θ_o be the two NNJM models already trained from the in- and out-domain data, respectively, and θ_o is trained using the same vocabulary as that of θ_i. We define the new normalized loss function as follows:

$$J(\theta_a) = -\sum_{n=1}^{N}\sum_{k=1}^{|V_o|}[\lambda\ y_{nk}\log\widehat{y}_{nk}(\theta_a) + (1-\lambda)\ y_{nk}[p_{nk}(\theta_i)-p_{nk}(\theta_o)]\log\widehat{y}_{nk}(\theta_a)] \tag{13}$$

where y_{nk}, $\widehat{y}_{nk}(\theta_a)$, $p_{nk}(\theta_i)$ and $p_{nk}(\theta_o)$ are similarly defined as before. This loss function minimizes the cross entropy of the current model θ_a with respect to the gold labels y_n and the difference between the in-domain model θ_i and the out-domain model θ_o. Intuitively, the regularizer assigns higher weights to training instances that are not only similar to the in-domain but also dissimilar to the out-domain. The parameter $\lambda \in [0, 1]$ determines the strength of the regularization. The corresponding NCE loss can be defined as follows:

$$J(\theta_a) \quad = -\sum_{n=1}^{N}\sum_{k=1}^{|V_o|}[\lambda\ y_{nk}\log\sigma_{nk} + (1-\lambda)\ y_{nk}\log\sigma_{nk}\Big(p_{nk}(\theta_i)-p_{nk}(\theta_o)\Big) + \sum_{m=1}^{M} y_{nk}^{m}\log\psi_{nk}] \tag{14}$$

The derivatives of the above cost function with respect to the final layer weight vectors $\mathbf{w}_j$ turn out to be:

$$\nabla_{\mathbf{w}_j} J(\theta_a) = -\sum_{n=1}^{N}[\lambda\ (y_{nj}-\sigma_{nj}) + (1-\lambda)[p_{nj}(\theta_i)-p_{nj}(\theta_o)-\sum_{k} y_{nk}\ \sigma_{nj}\ (p_{nk}(\theta_i)-p_{nk}(\theta_o))]] \tag{15}$$

In a way, the regularizers in our loss functions are inspired from the data selection methods of Axelrod et al. (2011), where they use cross entropy differences between the in-domain and the out-domain language models to score out-domain sentences (see Section 3.1). However, our approach is quite different from theirs in several aspects. First and most importantly, we take the scoring inside model training and use it to bias the training towards the in-domain model. Both the scoring and the training are performed at the (bilingual) *n*-gram level rather than at the sentence level. Integrating scoring inside the model allows us to learn a robust model by training/tuning the relevant parameters, while still using the complete data. Secondly, our models are based on NNJMs, while theirs utilize the traditional Markov-based generative models.

From another perspective, the regularizers in our models are weighting the training instances (i.e., bilingual n-grams) with respect to only in-domain (for NDAM-v1) or both in- and out-domains (NDAM-v2). In that way, our models are related to the phrase pair weighting method of Foster et al. (2010). They first train an instance-weighted model from the out-domain data, and then they combine it with the relative-frequency counts (i.e., n-gram estimates) in the in-domain data. The instance-weighted model uses a set of usefulness features in a discriminative model to weight the out-domain phrase counts. The features are intended to reflect the degree to which a phrase pair belongs to general language, and its degree of similarity to the in-domain. Our models are thus fundamentally very different from their approach.

3.3. Mixture models

While regularized training on the complete data based on the in- and out-domain models (as done by the above NDAM models) helps to build better adapted models, it does not fully prevent out-domain data from contaminating the resultant model parameters, especially when the out-domain data is very different from the in-domain data. Another approach to adaptation is to combine the in- and out-domain models and make a composite model. In the following we present interpolation and fusion approaches to achieve this.

3.3.1. Interpolation

Our first approach learns the combination weights of the 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[3] using a standard Expectation Maximization (EM) algorithm. Let $\theta_d \in \{\theta_1, \ldots, \theta_D\}$ represent

[3] 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.

an NNJM model trained on domain d, where D is the total number of domains. The probability of a sequence $(\mathbf{x}_n, y_n)$ can be written as a mixture of D probability densities, each coming from a different model:

$$P(y_n|\mathbf{x}_n, \theta, \lambda) = \sum_{d=1}^{D} P(y_n|\mathbf{x}_n, z_n = d, \theta_d)\ \lambda_d \tag{16}$$

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

$$P(T|S) \approx \prod_{i=1}^{|T|} \sum_{d} \lambda_d P(t_i|t_{i-1} \ldots t_{i-n+1}, \mathbf{s_i}, \theta_d) \tag{17}$$

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.

3.3.2. Neural Fusion Models

In mixture modelling we only learn the mixture weights (or the relative weights) of the participating models. Learning only the coarse-grained mixture weights can limit the composite model to be expressive enough to learn the variability in data patterns. To cope with this limitation, in the fusion models we readjust a large number of parameters of the participating models to effectively capture the variability in data patterns. In the following we present two variations of our fusion model.

Neural Fusion Model-I (NFM-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:

$$P(y_n = k|\mathbf{x_n^i}, \mathbf{x_n^o}, \theta^i, \theta^o) = \frac{\exp\ ([\mathbf{w}_k^i, \mathbf{w}_k^o]^T [\mathbf{z}_n^i, \mathbf{z}_n^o])}{\sum_{m=1}^{|V_o|} \exp\ ([\mathbf{w}_m^i, \mathbf{w}_m^o]^T [\mathbf{z}_n^i, \mathbf{z}_n^o])} \tag{18}$$

where $[\mathbf{w}_k^i, \mathbf{w}_k^o]$ is the concatenation of the output layer weights of in- and out-domain models, and $[\mathbf{z}_n^i, \mathbf{z}_n^o]^T$ is the corresponding concatenated activations at the outermost hidden layer. Fig. 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 $\theta^o = [E^o, U^o, W^o]$.[4]

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

$$\nabla_{\mathbf{w}_j^d} J(\theta) = \sum_{n=1}^{N} \left[(y_{nj} - \sigma_{nj})\mathbf{z}_n^d\right] \tag{19}$$

where y_{nj} and σ_{nj} are similarly defined as in Eq. (3); the superscript $d \in \{i, o\}$ denotes the domain. 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.

Neural Fusion Model-II (NFM-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.

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

Both fusion and linear interpolation have the same number of parameters, which is the sum of the size of the base models.[5] In fusion, we readjust all the parameters of the base models (NFM-I), or just the output layer weights (NFM-II), where in linear interpolation, we only learn their mixing weights.

4. Technical details

In this section, we describe implementation details of our adaptation models, that we found to be crucial. This includes: (*i*) using *gradient clipping* to handle vanishing/exploding gradient problem in SGD training with backpropagation; (*ii*) selecting appropriate *noise distribution* in NCE, and (*iii*) special handling of out-domain words that are unknown to the in-domain.

4.1. Gradient clipping

Two common issues with training neural networks on large data are the *vanishing* and the *exploding* gradients problems (Pascanu et al., 2013). The error gradients propagated by the backpropagation may sometimes become very small or very large which can lead to undesired (nan) values in weight matrices, causing the training to fail. We also experienced the same problem in training our models with NCE. Since our network is not so deep (only two hidden layers), the issue in our case was more of gradient exploding than gradient vanishing. Lowering the learning rate or mini-batch size worked for some datasets, but it was not a robust solution for many datasets. We adopted the simple solution called *gradient clipping* (Mikolov, 2012), where we truncate the gradient if it becomes too large or too small. In our experiments, we limit the gradients to be in the range $[-5; +5]$.

4.2. Noise distribution in noise contrastive estimation

Training with noise contrastive estimation (NCE) relies on sampling from a noise distribution (i.e., ψ in Eq. (5)), and the performance of the models varies considerably with the choice of the distribution. We explored *uniform* and *unigram* noise distributions in this work. With uniform distribution, every word in the output vocabulary has the same probability to be sampled as a noise sample. The *unigram* noise distribution is a multinomial distribution over words constructed by counting their occurrences in the output (i.e., *n*th word in the *n*-gram sequence). In our experiments, unigram distribution delivered much lower perplexity and better MT results compared to the uniform one. Mnih and Teh (2012) also reported similar findings on perplexity in their study.

4.3. Dealing with unknown words

In order to reduce the training time and to learn better word representations, it is common to train neural language models on corpora containing frequent vocabulary words only and grouping rare (and unseen) words under a common class unk. This results in a large number of *n*-gram sequences containing at least one unk word and thereby, makes unk a highly probable word in the model.[6]

Recall that the adaptation models proposed in Section 3.2 rely on scoring out-domain sequences using models that are trained based on the in-domain vocabulary. To score out-domain sequences using a model, we need to generate the sequences using the same vocabulary on which the model was trained. The out-domain words that are not seen in the in-domain data also map to the unk class. As a result, out-domain sequences containing unks get higher probability although they are distant from the in-domain data.

A solution to this problem is to have an in-domain model that can differentiate between its own unk class (resulted due to the pruned vocabulary) and the unknown words coming from the out-domain data (i.e., the actual unknown words). We do this by introducing a new class unk_o to represent the latter.

We train the in-domain model by adding a few dummy sequences with unk_o occurring in both source and target sides. This enables the model to learn unk and unk_o separately, while giving low probability mass to unk_o. Note that

[5] More specifically, interpolation has few more parameters accounting for the mixture weights. In contrast, the combination in fusion models is done at the output layer of the network, thus does not increase the overall number of parameters in the composite model.

[6] 30% of *n*-gram sequences in our Arabic–English in-domain data contains at least one unk.

Table 1
Statistics of the English–German and Arabic–English training corpora in terms of Sentences and Tokens (Source/Target). Tokens are represented in Millions. ep = Europarl, cc = Common Crawl, un = United Nations.

English–German				Arabic–English			
Corpus	Sent.	Tok_{EN}	Tok_{DE}	Corpus	Sent.	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	2 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

the *n*-gram sequences in the out-domain data contain both unk and unk_o classes depending on whether a word is unknown to only pruned in-domain vocabulary (i.e., unk) or is unknown to full in-domain vocabulary (i.e., unk_o).

5. Experiments

In this section, we present the experimental setup along with the results. We first describe the datasets used and the system settings including the settings for the neural networks and the SMT pipeline, and then we discuss the results.

5.1. 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 (≈ 177K sentences) corpus. For Arabic-to-English pair, we used the multiUN (≈ 3.7M sentences) (Eisele and Chen, 2010) as our out-domain corpora. For English-to-German, we used data made available (≈ 4.4M sentences) for the 9*th* Workshop on Machine Translation[7] as our out-domain data. Table 1 shows the size of the data used. Language models were trained on all the available monolingual data (English: ≈ 287.3M and German: ≈ 59.5M sentences). We used *Farasa* segmenter (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.

Training NN models is expensive.[8] In the interest of time, we therefore reduced the NN 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.

We use the test set of IWSLT-2011 as the development set and the test sets of IWSLT-2012 and IWSLT-2013 as the test sets. The systems were tuned on the concatenation of the development and the test sets of IWSLT-2010. The tuning set was also used to measure the perplexities of different models.

5.2. System settings

NNJM and NDAM. The NNJM models were trained using the Neural Probabilistic Language Model (NPLM) toolkit (Vaswani et al., 2013).[9] We used a target history of 4 words and an aligned source window of 9 words, forming a joint stream of 14-grams. We restricted the source and the target side vocabularies to the 20*K* and 40*K* most frequent

[7] http://www.statmt.org/wmt14/translation-task.html.
[8] 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.
[9] http://nlg.isi.edu/software/nplm/

words. Larger vocabulary sizes increased the training time significantly, but did not improve the MT performance. The reason could be that since our in-domain data (IWSLT) is small (less than 200K sentences), the smaller vocabulary sizes we are using suffice the purpose.

The word vector size D and the hidden layer size were set to 150 and 750, respectively. Only one hidden layer is used to allow faster decoding. Training was done by the standard stochastic gradient descent (SGD) with NCE using $S = 100$ noise samples and a mini-batch size of 1000. All models were trained for 25 epochs. We used identical settings to train the NDAM models, except for the special handling of unk tokens.

Recently, Luong and Manning (2015) demonstrated considerable improvements by fine-tuning an out-of-domain attention-based NMT towards in-domain data (i.e., by training the neural network additional epochs on in-domain data). We also tried this strategy by training an out-domain NNJM and then fine-tuning it using in-domain data.

Machine translation systems. Baseline systems were trained by simply concatenating all the data shown in Table 1. To evaluate our work, 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, with the fusion models, where models are trained independently and adjusted towards in-domain data, with the interpolated models or with the fine-tuned NNJM model. Later, for comparison and completeness, we also experimented with training translation models from in and out-domain separately. We compared performance of our models against state-of-the-art model adaptation techniques, *linear phrase-table interpolation, instance weighting* and *fill-up* methods. Here phrase-tables were separately trained and then combined using above mentioned methods.[10] Finally, we compared our system against MML-filtering (Axelrod et al., 2011), although this technique falls in the array of data-selection methods. The optimal thresholds were found to be 20% and only 5% in English–German and Arabic–English, respectively, upon which full MT systems were then retrained.

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 operation sequence model (Durrani et al., 2015b), with supporting (gaps and jump penalty) features (Durrani et al., 2011) 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.

5.3. Results

In this section we report our results. First we perform an intrinsic evaluation of the models by comparing their perplexity values on the tune set. Then we present the results of different adaptation methods on the translation task.

5.3.1. Intrinsic evaluation

In Table 2, we compare the baseline NNJM models (i.e., NNJM_i and NNJM_c) with our two regularized adaptation models (i.e., NDAM-v1 and NDAM-v2 in Section 3.2), our deep fusion model (NFM-I in Section 3.3.2) and fine tuning (FT) method of Luong and Manning in terms of their perplexity numbers on the in-domain held-out tune set (i.e., development and test sets of IWSLT-2010).[11] Recall that all the neural models are trained with NCE objective instead of the standard log likelihood objective. Therefore, a question may arise whether one should evaluate the models based on their perplexity (or entropy) values on a held-out dataset. It can be shown that as the number of noise samples per observation (i.e., S) increases, the NCE gradient approaches the log likelihood gradient (Mnih and Teh, 2012). We use the same number of noise samples across the models (i.e., $S = 100$), which makes their perplexity values on a held-out dataset comparable.

The second column shows the perplexity values for the baseline NNJM (NNJM_i), when the model is trained on IWSLT or UN domains separately. The perplexity numbers demonstrate how different UN data is from the

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

[11] It does not make sense to compare perplexity numbers for the linear and log-linear models (Section 3.3). Because of the convex combination that mixture models perform, the mixture perplexity is always worse than the in-domain baseline.

Table 2
Comparison of Perplexities the in-domain tune set: $NNJM_i$ = IN Domain NNJM, $NNJM_c$ = IN+OUT Concatenated NNJM, NFM-I = Neural Fusion Model-1, $NDAM_{v*}$ = Neural Domain Adaptation Models, FT = Fine tuned models, $NNJM_m$ = NNJM trained on IN + MML Selected Data, $NDAM_m$ = NDAM model trained on IN + MML Selected Data.

Dom	$NNJM_i$	$NNJM_c$	NFM-I	$NDAM_{v1}$	$NDAM_{v2}$	FT	$NNJM_m$	$NDAM_m$
Perplexities on Tune Set (EN-DE)								
ID	10.20	-	-	-	-			
OD	10.70	6.71	6.59	**6.21**	6.37	6.71	6.65	6.29
Perplexities on Tune Set (AR-EN)								
ID	12.55	-	-	-	-			
OD	111.11	11.94	11.25	10.83	10.74	**8.5**	10.5	10.03

in-domain IWSLT — perplexity values of 12.55 vs. 111.11. In comparison, the out-domain data in English−German is less different than the in-domain 10.20 vs. 10.70.

The third column shows the result of the NNJM, when it is trained on the concatenation of IWSLT and UN data ($NNJM_c$). The perplexity improves significantly showing that there is useful information available in out-domain data, which can be utilized to improve the in-domain baseline. It also demonstrates the robustness of neural models as language models. Unlike (discrete) *n*-gram models, neural models yield better generalization with the increase of data without completely skewing towards the dominating part of the data. This could be attributed to the continuous representation of the neural models.

The fourth column shows the perplexity of the *fusion* model, NFM-I. Notice that by readjusting the parameters of the in- and out-domain models in the model combination process, NFM-I gets better perplexity value than $NNJM_c$.

The fifth and sixth columns show results of the regularized adaptation models NDAM-v1 and NDAM-v2. Both models give better perplexity values than the baselines and the fusion model. This demonstrates that the data dependent regularization of the loss function based on the in- (and out-) domain model(s) is more beneficial for the adaptation task. When we compare the two NDAM models, we observe that NDAM-v2 yields better perplexity than NDAM-v1 by penalizing instances that are favored by the out-domain model. The seventh column shows perplexity results for the fine tuning method of Luong and Manning (2015), which gives same perplexity as the baseline ($NNJM_c$ for English−German but much better perplexity in the Arabic-to-English pair.

Finally, we also retrained the NNJM and the NDAM models on the data that we selected using MML, rather than training on a randomly selected data. The last two columns in Table 2 show the perplexity values on the held-out tune set for these two models. We can observe that the perplexity numbers are better than their counterparts trained on the randomly selected data.

5.3.2. Machine translation results − model adaptation

In this section, we evaluate the performance of our model adaptation methods extrinsically, by replacing the baseline NNJM models (i.e., $NNJM_i$ and $NNJM_c$) with different adapted versions, and finally also comparing our approach against existing domain adaptation approaches.

First row in Table 3 shows results for the baseline system, which uses an NNJM trained on the plain concatenation of in-and out-domain data. The next block of rows shows results for systems, where the NNJM models are adapted using weighted concatenation, i.e., using in-domain model (NDAM-v1) or both in- and out-domain models (NDAM-v2) or by fine tuning. The fine-tuning method performs much better than the baseline and the NDAM models on the English−German task but was not as effective on the Arabic−English pair. The linear and log-linear interpolation of in- and out- NNJM models (see fifth and sixth rows) performed similarly and gave improvements of up to +0.4 BLEU points. Next set of rows show results from the fusion models which outperformed the baseline system as well as other neural adapted models. Fusion Model-I, which performs deeper fusion (i.e., till the embedding layer) worked better than Fusion Model-II, where backpropagation is restricted only to the out-most hidden layer, showing that there is an additional value in doing a deeper fusion.

Next we compare our model with the domain adaptation methods available in Moses (See Table 4). We will only compare the results against our best adaptation model (NFM-I). Here instead of adapting the

Table 3
Comparison against Neural Model Weighting Techniques – NDAM*, NFM = Neural Fusion Model, Fine-Tuning = Luong and Manning 2015.

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Base (NNJM$_c$)	27.3	22.9	24.5	24.9		26.1	29.4	30.5	28.7	
NDAM-v1	27.5	23.4	25.1	25.3	+0.4	26.1	29.6	30.9	28.9	+0.2
NDAM-v2	27.4	23.3	24.8	25.2	+0.3	26.3	30.0	30.9	29.1	+0.4
Fine-Tuning	27.7	23.9	25.3	25.6	+0.7	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
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

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 outperformed these methods in both language pairs and in most of the test-sets. Additional experiments combining phrasetable 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.

To gain further insights of the models, we analyzed some translations of the baseline system, Base (NNJM$_c$), and spotted several cases of lexical ambiguity caused by the out-domain data. For example, the Arabic phrase "الحمل الزائد للاختيار" can be translated to *choice overload* or *unwanted pregnancy*. The latter translation is incorrect given the in-domain context. The bias created by the out-domain data caused Base (NNJM$_c$) to choose the contextually incorrect translation *unwanted pregnancy*. However, our adapted systems were able to translate the phrase correctly. In another example "ماذا عن لياقة البدن؟" (*How about fitness?*), the word "لياقة" was translated to *proprietary* incorrectly by Base (NNJM$_c$), a translation frequently observed in the out-domain data. The adapted models translated it correctly to *fitness*, as preferred by the in-domain data.

5.3.3. Machine translation results – data selection methods

In this subsection we report our results for applying data selection methods to the training data. In this method we select *P*% top scoring data instances from the out-domain data to include it in the training set. To select the right threshold *P*, we experimented with 0%, 2.5%, 5%, 10%, 20%, 40% and 100%. Initially we tried to train n-gram language models from the selected data and computed perplexity on the held-out tuning set. But this did not give any concluding evidence. The perplexity numbers kept increasing (See Fig. 3). Only by training full MT pipeline on a concatenation of in-domain and the selected data, we found the optimal thresholds (5% for Arabic–English and 20% for English–German). We validated this on our dev-set (IWSLT-2011).

In Table 5, we experiment with the 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 helpful for English-to-

Table 4
Comparison against Translation Model Adaptation Techniques – Linear Interpolation and Instance Weighting (Sennrich 2012), Fill-up Method (Bisazza et al. 2011), NFM = Neural Fusion Model (this work)

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Base (NNJM$_c$)	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
+ Instance Wt.	28.0	23.8	25.3	25.7	+0.8	27.5	30.7	30.8	29.7	+1.0

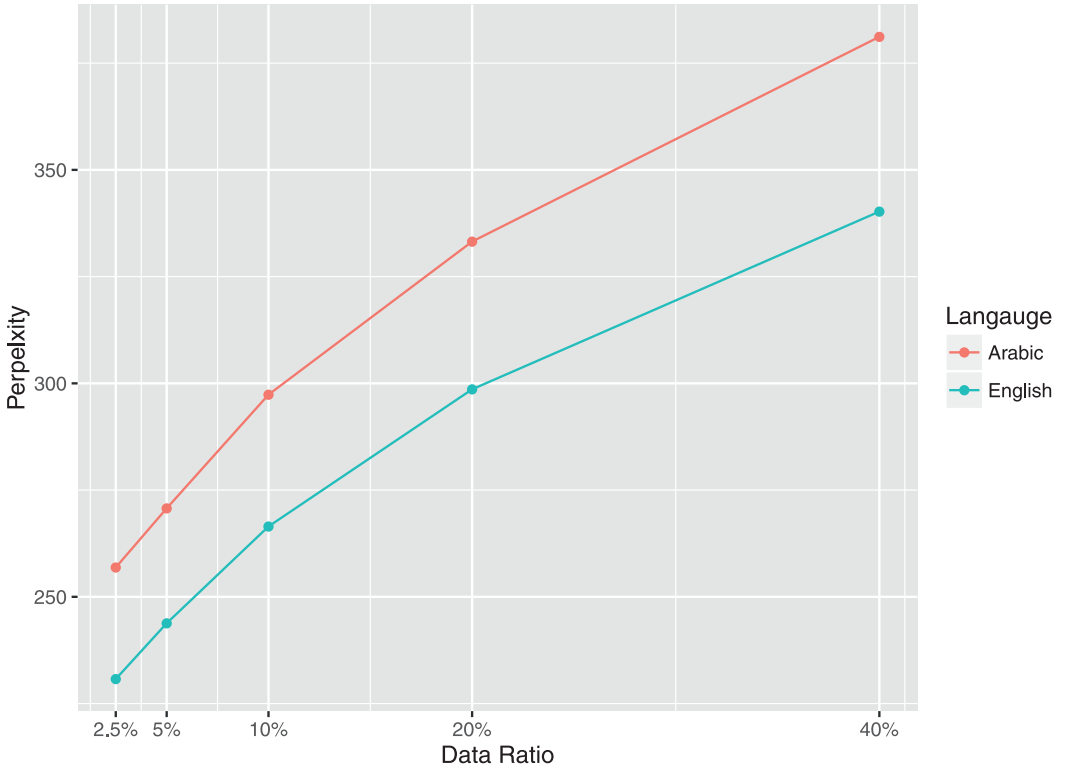

Fig. 3. Perplexity values for selected amounts from the out-domain data.

German and much of the out-domain data (UN corpus) used in these experiments is harmful in the case of Arabic-to-English. These observations are inline with the results reported in Duh et al. (2013) for English–German and Mansour and Ney (2013),Sajjad et al. (2013) for Arabic–English. Best MML results were obtained by selecting only 5% of the out-domain data. In comparison, data selection was found to be less useful in the case of English-to-German. NNJM-based selection gave similar improvement as the MML selection. The overlap between the data instances selected by the two methods was roughly 77%, which explains the similarity of the results. In an ideal scenario model adaptation techniques should not rely on data filtering, but should be able to select best hypotheses from the unpruned search space. However, this is not practical – given the large amounts of generic data, and the slow training time of neural networks, data selection is a mandatory first step. Also, the decoder makes more search errors when presented with a full search space.

In additional experimentation (Table 6), we found that using our NDAM-v1 and fusion models instead of baseline NNJM still gave improvements (up to +0.5 and +0.3 in English–German and Arabic–English, respectively) on top of MML-based systems. When trained on selected data, and with pruned search space, our model achieves optimal performance. These results show that there is a merit in combining model adaptation with data selection.

Table 5
Comparison of NNJM-based Selection against MML-based Selection (Axelrod et al 2011).

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Cat	27.3	22.9	24.5	24.9		26.1	29.4	30.5	28.7	
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	−0.2	27.4	30.8	30.9	29.7	+0.6
NNJM	27.0	23.1	23.9	24.7	−0.2	27.6	30.7	31.1	29.8	+0.7

Table 6
Comparison of models trained on the selected data.

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
MML	26.9	22.9	24.4	24.7		27.4	30.8	30.9	29.7	
+NDAM-v1	26.8	23.1	24.8	24.9	+0.2	27.5	31.0	31.2	29.9	+0.2
+NFM-I	27.6	23.1	25.0	25.2	+0.5	27.8	31.3	31.1	30.1	+0.3

6. 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 the MT systems, hurt its performance. The idea is to score the out-domain data instances using a model trained from the in-domain data and then to apply a cut-off threshold based on the resulting scores. The MT system can then be trained on the concatenation of the in-domain and the selected portion of the out-domain data that is closer to in-domain. Selection based methods can be helpful to reduce computational cost when training is expensive and also when memory is constrained.

Data selection was earlier done for language modeling using information retrieval techniques (Hildebrand et al., 2005) and using perplexity measure (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 side and target side language models. Duh et al. (2013) used recurrent neural network language model instead of an ngram-based language model to do the same. More recently, Liu et al. (2014) and Hoang and Sima'an (2014) use translation model features to do data selection.

The downside of data selection is that finding an optimal cut-off threshold is a time consuming process. Therefore rather than filtering less useful data, an alternative way is to down-weight it and boost the data closer to the in-domain. It is robust than selection since it takes advantage of the complete out-domain data with intelligent weighting towards the in-domain.

Federico (1999) proposed adaptation of n-gram language models based on the minimum discrimination information principle, where a background language model is adapted to fit constraints (e.g., in the form of a unigram distribution) on its marginal distributions that are derived from in-domain data. 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. (2013b) used vector space model for adaptation at phrase level. Every phrase pair is represented as a vector where every entry in the vector reflects its relatedness with each domain. Chen et al. (2013a) also applied mixture model adaptation for reordering model.

Domain adaptation in neural MT is carried out by the fine-tuning method (Luong and Manning, 2015). The idea is to train neural network model on the large out-domain corpus and then run additional iterations using only in-domain data. Do et al. (2015) applied their Continuous Translation Model (CTM) in adaptation scenario by training SMT system with large out-domain data and training CTM on the in-domain data. Although they only used the model for n-best rescoring.

7. Conclusions and future work

We proposed novel domain adaptation models by: (*i*) regularizing loss functions of the adapted model, and (*ii*) by fusing in- and out-domain models. The former uses data dependent regularization in their loss function to perform (soft) data selection inside the model, while the latter combines the in-domain and the out-domain models by readjusting their parameters. We also explored how the existing domain adaptation techniques such as mixture modelling and data selection can be used with the NNJM. Through extensive experimentation, we found our deep fusion model to outperform the other discussed neural adaptation methods as well as phrase-table adaptation techniques. Regularized adaptation models were also shown to improve, but their performance was on par with the existing techniques.

We also demonstrated that our models are complementary to the existing techniques and together the models can achieve a better translation quality.

Although this study focused on fusing multiple neural models for domain adaptation in phrase-based SMT, we would like to adopt the idea in the end-to-end NMT systems (Bahdanau et al., 2015), where the goal will be to fuse multiple NMT systems. We also intend to experiment with other neural network architecture including recurrent neural networks with long short term memory cells and gated recurrent units (Chung et al., 2014). We also plan to explore the utility of pre-trained word vectors (e.g., word2vec Mikolov et al., 2013a) in the embedding layer and the approximate softmax in the output layer of the neural network to improve the overall translation quality.

References

Abdelali, A., Darwish, K., Durrani, N., Mubarak, H., 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. Association for Computational Linguistics, San Diego, California, pp. 11–16. URL http://www.aclweb.org/anthology/N16-3003.

Abdelali, A., Guzman, F., Sajjad, H., Vogel, S., 2014. The AMARA corpus: Building parallel language resources for the educational domain. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). Reykjavik, Iceland.

Auli, M., Galley, M., Quirk, C., Zweig, G., 2013. Joint language and translation modeling with recurrent neural networks. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA.

Axelrod, A., He, X., Gao, J., 2011. Domain adaptation via pseudo in-domain data selection. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Edinburgh, United Kingdom.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate. ICLR. URL http://arxiv.org/pdf/1409.0473v6.pdf.

Bengio, Y., Ducharme, R., Vincent, P., Janvin, C., 2003. A neural probabilistic language model. J. Mach. Learn. Res. 3, 1137–1155. URL http://dl.acm.org/citation.cfm?id=944919.944966.

Birch, A., Huck, M., Durrani, N., Bogoychev, N., Koehn, P., 2014. Edinburgh SLT and MT system description for the IWSLT 2014 evaluation. In: Proceedings of the 11th International Workshop on Spoken Language Translation. Lake Tahoe, CA, USA.

Bisazza, A., Ruiz, N., Federico, M., 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In: Proceedings of the Seventh International Workshop on Spoken Language Translation (IWSLT), pp. 136–143.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., Federico, M., 2014. Report on the 11th IWSLT evaluation campaign. In: Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US.

Chen, B., Foster, G., Kuhn, R., 2013a. Adaptation of reordering models 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.

Chen, B., Kuhn, R., Foster, G., 2013b. 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.

Cherry, C., Foster, G., 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. Montréal, Canada.

Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Technical Report Arxiv report 1412.3555. Université de Montréal. Presented at the Deep Learning workshop at NIPS2014

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P., 2011. Natural language processing (almost) from scratch. JMLR.org, pp. 2493–2537.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., Makhoul, J., 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).

Do, Q.-K., Allauzen, A., Yvon, F., 2015. A discriminative training procedure for continuous translation models. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Lisbon, Portugal, pp. 1046–1052. URL http://aclweb.org/anthology/D15-1121.

Duh, K., Neubig, S.K., Graham, Tsukada, H., 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.

Durrani, N., Fraser, A., Schmid, H., Hoang, H., Koehn, P., 2013. Can Markov models over minimal translation units help phrase-based SMT? In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics, Sofia, Bulgaria, pp. 399–405. URL http://www.aclweb.org/anthology/P13-2071.

Durrani, N., Sajjad, H., Hoang, H., Koehn, P., 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.

Durrani, N., Sajjad, H., Joty, S., Abdelali, A., Vogel, S., 2015a. Using joint models for domain adaptation in statistical machine translation. In: Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV). AMTA, Florida, USA.

Durrani, N., Schmid, H., Fraser, A., 2011. A joint sequence translation model with integrated reordering. In: Proceedings of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11). Portland, OR, USA.

Durrani, N., Schmid, H., Fraser, A., Koehn, P., Schütze, H., 2015b. The Operation Sequence Model – Combining N-Gram-based and Phrase-based Statistical Machine Translation. Comput Ling 41 (2), 157–186.

Dyer, C., Chahuneau, V., Smith, N.A., 2013. A simple, fast, and effective reparameterization of ibm model 2. In: Proceedings of NAACL'13.

Eisele, A., Chen, Y., 2010. MultiUN: a multilingual corpus from united nation documents. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation. Valleta, Malta.

Federico, M., 1999. Efficient language model adaptation through MDI estimation. In Proceedings of EUROSPEECH, pp. 1583–1586.

Finch, A., Sumita, E., 2008. Dynamic model interpolation for statistical machine translation. In: Proceedings of the Third Workshop on Statistical Machine Translation. Columbus, Ohio.

Foster, G., Goutte, C., Kuhn, R., 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.

Foster, G., Kuhn, R., 2007. Mixture-model adaptation for SMT. In: Proceedings of the Second Workshop on Statistical Machine Translation.

Foster, G., Kuhn, R., 2009. Stabilizing minimum error rate training. In: Proceedings of the Fourth Workshop on Statistical Machine Translation. Athens, Greece.

Fujii, A., Utiyama, M., Yamamoto, M., Utsuro, T., 2010. Overview of the patent translation task at the ntcir-8 workshop. In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access, pp. 293–302.

Galley, M., Manning, C.D., 2008. A simple and effective hierarchical phrase reordering model. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. Honolulu, Hawaii, pp. 848–856. URL http://www.aclweb.org/anthology/D08-1089.

Gao, J., He, X., Yih, W.-t., Deng, L., 2014. Learning continuous phrase representations for translation modeling. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Baltimore, Maryland.

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

Heafield, K., 2011. KenLM: faster and smaller language model queries. In: Proceedings of the Sixth Workshop on Statistical Machine Translation. Edinburgh, Scotland, United Kingdom, pp. 187–197. URL http://kheafield.com/professional/avenue/kenlm.pdf.

Hildebrand, A.S., Eck, M., Vogel, S., Waibel, A., 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.

Hinton, G., Deng, L., Yu, D., Dahl, G., rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition. Signal Process. Mag 29 (6), 82–97.

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

Joty, S., Sajjad, H., Durrani, N., Al-Mannai, K., Abdelali, A., Vogel, S., 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.

Kalchbrenner, N., Blunsom, P., 2013. Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, USA.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E., 2007. Moses: open source toolkit for statistical machine translation. In: Proceedings of the Association for Computational Linguistics (ACL'07). Prague, Czech Republic.

Koehn, P., Schroeder, J., 2007. Experiments in domain adaptation for statistical machine translation. In: Proceedings of the Second Workshop on Statistical Machine Translation. Prague, Czech Republic.

Le, H.-S., Allauzen, A., Yvon, F., 2012. Continuous space translation models with neural networks. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, Montréal, Canada, pp. 39–48. URL http://www.aclweb.org/anthology/N12-1005.

Liu, L., Hong, Y., Liu, H., Wang, X., Yao, J., 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.

Luong, M.-T., Manning, C.D., 2015. Stanford neural machine translation systems for spoken language domain. International Workshop on Spoken Language Translation. Da Nang, Vietnam.

Mansour, S., Ney, H., 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.

Mariño, J.B., Banchs, R.E., Crego, J.M., de Gispert, A., Lambert, P., Fonollosa, J.A.R., Costa-jussà, M.R., 2006. N-gram-Based Machine Translation. Computational Linguistics 32 (4), 527–549 http://dx.doi.org/10.1162/coli.2006.32.4.527 .

Matsoukas, S., Rosti, A.-V.I., Zhang, B., 2009. Discriminative corpus weight estimation for machine translation. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2.

Mikolov, T., 2012. Statistical Language Models based on Neural Networks. Brno University of Technology Ph.D thesis.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013a. Efficient Estimation of Word Representations in Vector Space. CoRR http://arxiv.org/abs/1301.3781 .

Mikolov, T., Yih, W.-T., Zweig, G., 2013b. Linguistic Regularities in Continuous Space Word Representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, Atlanta, Georgia, pp. 746–751. URL http://www.aclweb.org/anthology/N13-1090.

Mnih, A., Teh, Y.W., 2012. A fast and simple algorithm for training neural probabilistic language models. In: Proceedings of the International Conference on Machine Learning.

Moore, R.C., Lewis, W., 2010. Intelligent selection of language model training data. In: Proceedings of the Association for Computational Linguistics (ACL'10). Uppsala, Sweden.

Nakov, P., Ng, H.T., 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.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. Morristown, NJ, USA, pp. 311–318.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16−21 June 2013.

Sajjad, H., Guzmn, F., Nakov, P., Abdelali, A., Murray, K., Obaidli, F.A., Vogel, S., 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).

Schwenk, H., 2012. Continuous space translation models for phrase-based statistical machine translation. In: Proceedings of COLING 2012: Posters. Mumbai, India.

Sennrich, R., 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.

Sennrich, R., Haddow, B., Birch, A., 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.

Socher, R., Bauer, J., Manning, C.D., Andrew Y., N., 2013. Parsing with compositional vector grammars. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Sofia, Bulgaria, pp. 455–465. URL http://www.aclweb.org/anthology/P13-1045.

Vaswani, A., Zhao, Y., Fossum, V., Chiang, D., 2013. Decoding with large-scale neural language models improves translation. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.