How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models

Shafiq Joty **Hassan Sajjad** **Nadir Durrani**
Kamla Al-Mannai **Ahmed Abdelali** **Stephan Vogel**
Qatar Computing Research Institute - Hamad Bin Khalifa University
{sjoty,hsajjad,ndurrani,kmannai,aabdelali,svogel}@qf.org.qa

Abstract

We present novel models for domain adaptation based on the neural network joint model (NNJM). Our models maximize the cross entropy by regularizing the loss function with respect to in-domain model. Domain adaptation is carried out by assigning higher weight to out-domain sequences that are similar to the in-domain data. In our alternative model we take a more restrictive approach by additionally penalizing sequences similar to the out-domain data. Our models achieve better perplexities than the baseline NNJM models and give improvements of up to 0.5 and 0.6 BLEU points in Arabic-to-English and English-to-German language pairs, on a standard task of translating TED talks.

1 Introduction

Rapid influx of digital data has galvanized the use of empirical methods in many fields including Machine Translation (MT). The increasing availability of bilingual corpora has made it possible to automatically learn translation rules that required years of linguistic analysis previously. While additional data is often beneficial for a general purpose Statistical Machine Translation (SMT) system, a problem arises when translating new domains such as lectures (Cettolo et al., 2014), patents (Fujii et al., 2010) or medical text (Bojar et al., 2014), where either the bilingual text does not exist or is available in small quantity. All domains have their own vocabulary and stylistic preferences which cannot be fully encompassed by a system trained on the general domain.

Machine translation 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 distant or overwhelmingly larger than the in-domain data. Additional data increases lexical ambiguity by introducing new senses to the existing in-domain vocabulary. 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 incorrect in the context of the in-domain data, 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. In this paper, we tackle this problem by proposing domain adaptation models that make use of all the data while preserving the in-domain preferences.

A significant amount of research has been carried out recently in domain adaptation. The complexity of the SMT pipeline, starting from corpus preparation to word-alignment, and then training a wide range of models opens a wide horizon to carry out domain specific adaptations. This is typically done using either *data selection* (Matsoukas et al., 2009) or *model adaptation* (Foster and Kuhn, 2007). In this paper, we further research in model adaptation using the neural network framework.

In recent years, there has been a growing interest in deep neural networks (NNs) and word embeddings with application to numerous NLP problems. A notably successful attempt on the SMT frontier was recently made by Devlin et al., 2014). They proposed a neural network joint model (NNJM), which augments streams of source with target n-grams and learns a NN model over vector representation of such streams. The model is then integrated into the decoder and used as an additional language model feature.

Our aim in this paper is to advance the state-of-the-art in SMT by extending NNJM for domain adaptation to leverage the huge amount of out-

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

To this end, we propose two novel extensions of NNJM for domain adaptation. Our first model minimizes the cross entropy by regularizing the loss function with respect to the in-domain model. The regularizer gives higher weight to the training instances that are similar to the in-domain data. Our second model takes a more conservative approach by additionally penalizing data instances similar to the out-domain data.

We evaluate our models on the standard task of translating Arabic-English and English-German language pairs. Our adapted models achieve better perplexities (Chen and Goodman, 1999) than the models trained on in- and in+out-domain data. Improvements are also reflected in BLEU scores (Papineni et al., 2002) as we compare these models within the SMT pipeline. We obtain gains of up to 0.5 and 0.6 on Arabic-English and English-German pairs over a competitive baseline system.

The remainder of this paper is organized as follows: Section 2 gives an account on related work. Section 3 revisits NNJM model and Section 4 discusses our models. Section 5 presents the experimental setup and the results. Section 6 concludes.

2 Related Work

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

2.1 Data Selection

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. The idea is to score the out-domain data using a model trained from the in-domain data and apply a cut-off based on the resulting scores. The MT system can then be trained on a subset 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 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., 2015) performed data selection using operation sequence model (OSM) (Durrani et al., 2011) and NNJM models.

2.2 Model Adaptation

The downside of data selection is that finding an optimal cut-off threshold is a time consuming process. 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.

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). Durrani et al., 2015) applied EM-based mixture modeling to OSM and NNJM models to perform model weighting. Chen et al., 2013b) 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. Chen et al., 2013a) also applied mixture model adaptation for reordering model.

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).

In this paper, we do model adaptation using a neural network framework. In contrast to previous work, we perform it at the (bilingual) n-gram level, where n is sufficiently large to cap-

ture long-range cross-lingual dependencies. The generalized vector representation of the neural network model reduces the data sparsity issue of traditional Markov-based models by learning better word classes. Furthermore, our specially designed loss functions for adaptation help the model to avoid deviation from the in-domain data without losing the ability to generalize.

3 Neural Network Joint Model

In recent years, there has been a great deal of effort dedicated to neural networks (NNs) and word embeddings with applications to SMT and other areas in NLP (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). Recently, Devlin et al., 2014) proposed a neural network joint model (NNJM) and integrated it into the decoder as an additional feature. They showed impressive improvements in Arabic-to-English and Chinese-to-English MT tasks. Let us revisit the NNJM model briefly.

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

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

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. As exemplified in Figure 1, this is essentially a $(p+q)$-gram neural network LM (NNLM) originally proposed by Bengio et al., 2003). Each input word i.e. source or target word 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.[1] 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 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 over the output vocabulary V_o of target words. Formally, the probability of getting k-th word in the output given the context $\mathbf{x_n}$ can be written as:

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

[1]Note that L is a model parameter to be learned.

where $\phi(\mathbf{x_n})$ 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 major bottleneck, however, 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 to address this issue.

3.1 Training by Noise Contrastive Estimation

The standard way to train NNLMs is to maximize the log likelihood of the training data:

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

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

$$P(y_n = k|\mathbf{x_n}, \theta) = \frac{\sigma(y_n = k|\mathbf{x_n}, \theta)}{Z(\phi(\mathbf{x_n}), \mathbf{W})} \quad (4)$$

where $\sigma(.)$ is the un-normalized score and $Z(.)$ is the normalization factor. In NCE, we consider

[2]This would take few weeks for a modern CPU machine to train a single NNJM model on the whole data.

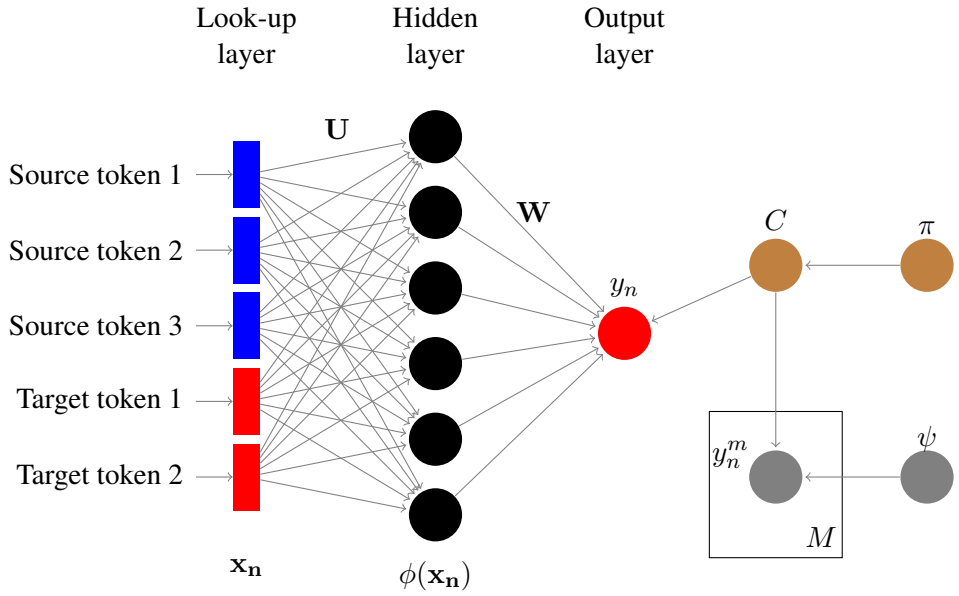

Figure 1: A simplified neural network joint model with noise contrastive loss, where we use 3-gram target words (i.e., 2-words history) and a source context window of size 3. For illustration, the output y_n is shown as a single categorical variable (scalar) as opposed to the traditional one-hot vector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

$Z(.)$ as an additional model parameter along with the regular parameters, i.e., weights, look-up vectors. However, it has been shown that fixing $Z(.)$ to 1 instead of learning it in training does not affect the model performance (Mnih and Teh, 2012).

For each training instance $(\mathbf{x_n}, y_n)$, we add M noise samples $(\mathbf{x_n}, y_n^m)$ by sampling y_n^m from a known noise distribution ψ (e.g., `unigram`, `uniform`) M many times (i.e., $m = 1 \ldots M$); see Figure 1. NCE loss is then defined 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:

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

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

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, Equation 6 can be further simplified as:

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

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 (Equation 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 comparison with $|V_o|$. In all our experiments we use NCE loss with $M = 100$ samples as suggested by Mnih and Teh, 2012).

4 Neural Domain Adaptation Models

The ability to generalize and learn complex semantic relationships (Mikolov et al., 2013b) and its compelling empirical results gives a strong motivation to use the NNJM model for the problem of domain adaptation in machine translation. However, the vanilla NNJM described above is limited in its ability to effectively learn from a large and diverse out-domain data in the best favor of an in-domain data. To address this, we propose two neural domain adaptation models (NDAM) extending the NNJM model. Our models add regularization to its loss function either with respect to in-domain or both in- and out-domains. In both cases, we first present the regularized loss function for the normalized output layer with the standard softmax,

followed by the corresponding un-normalized one using the noise contrastive estimation.

4.1 NDAM$_{v1}$

To improve the generalization of word embeddings, NNLMs are generally trained on very large datasets (Mikolov et al., 2013a; Vaswani et al., 2013). Therefore, we aim to train our neural domain adaptation models (NDAM) on in- plus out-domain data, while restricting it to drift away from in-domain. In our first model NDAM$_{v1}$, we achieve this by biasing the model towards the in-domain using a regularizer (or prior) based on the in-domain model. Let θ_i be an NNJM model already trained on the in-domain data. We train an adapted model θ_a on the whole data, but regularizing it with respect to θ_i. We redefine the normalized loss function of Equation 3 as follows:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, logP(y_n = k|\mathbf{x_n}, \theta_a) + (1-\lambda)\, y_{nk}\, P(y_n = k|\mathbf{x_n}, \theta_i)\, logP(y_n = k|\mathbf{x_n}, \theta_a)\Big] \quad (8)$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \hat{y}_{nk}(\theta_a) + (1-\lambda)\, y_{nk}\, p_{nk}(\theta_i)\, log\, \hat{y}_{nk}(\theta_a)\Big] \quad (9)$$

where $\hat{y}_{nk}(\theta_a)$ 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 of the current model θ_a with respect to the gold labels y_n and the in-domain model θ_i. The mixing parameter $\lambda \in [0, 1]$ determines the relative strength of the two components.[3] Similarly, we can re-define the NCE loss of Equation 7 as:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \sigma_{nk} + (1-\lambda)\, y_{nk}\, p_{nk}(\theta_i)\, log\, \sigma_{nk} + \sum_{m=1}^{M} y_{nk}^{m} log\, \psi_{nk}\Big] \quad (10)$$

We use SGA with backpropagation to train this model. The derivatives of $J(\theta_a)$ with respect to the final layer weight vectors $\mathbf{w}_j$ turn out to be:

$$\nabla_{\mathbf{w}_j} J(\theta_a) = \sum_{n=1}^{N}\Big[\lambda\,(y_{nj} - \sigma_{nj}) + (1-\lambda)\,[p_{nj}(\theta_i) - \sum_{k} y_{nk}\, p_{nk}(\theta_i)\, \sigma_{nj}]\Big] \quad (11)$$

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

4.2 NDAM$_{v2}$

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.

Let θ_i and θ_o be the two NNJMs already trained from the in- and out-domains, respectively, and θ_o is trained using the same vocabulary as θ_i. We define the new normalized loss function as follows:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \hat{y}_{nk}(\theta_a) + (1-\lambda)\, y_{nk}\, [p_{nk}(\theta_i) - p_{nk}(\theta_o)]\, log\, \hat{y}_{nk}(\theta_a)\Big] \quad (12)$$

where y_{nk}, $\hat{y}_{nk}(\theta_a)$, $p_{nk}(\theta_i)$ 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:

$$J(\theta_a) = \sum_{n=1}^{N}\sum_{k=1}^{|V_o|}\Big[\lambda\, y_{nk}\, log\, \sigma_{nk} + (1-\lambda)\, y_{nk}\, log\, \sigma_{nk}\, (p_{nk}(\theta_i) - p_{nk}(\theta_o)) + \sum_{m=1}^{M} y_{nk}^{m}\, log\, \psi_{nk}\Big] \quad (13)$$

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

$$\nabla_{\mathbf{w}_j} J(\theta_a) = \sum_{n=1}^{N}\Big[\lambda\,(y_{nj} - \sigma_{nj}) + (1-\lambda)[p_{nj}(\theta_i) - p_{nj}(\theta_o) - \sum_{k} y_{nk}\, \sigma_{nj}\; (p_{nk}(\theta_i) - p_{nk}(\theta_o))]\Big] \quad (14)$$

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 between the in- and the out-domain LMs to score out-domain sentences. However, our approach is quite different from them 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 NNs, while theirs utilize the traditional Markov-based generative models.

4.3 Technical Details

In this section, we describe some implementation details of NDAM that we found to be crucial, such as: using *gradient clipping* to handle vanishing/exploding gradient problem in SGA training with backpropagation, selecting appropriate *noise distribution* in NCE, and special handling of out-domain words that are unknown to the in-domain.

4.3.1 Gradient Clipping

Two common issues with training deep NNs on large data-sets 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 our NDAM quite often. One simple solution to this problem is to truncate the gradients, known as *gradient clipping* (Mikolov, 2012). In our experiments, we limit the gradients to be in the range $[-5; +5]$.

4.3.2 Noise Distribution in NCE

Training with NCE relies on sampling from a noise distribution (i.e., ψ in Equation 5), and the performance of the NDAM 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 noise. 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.

4.3.3 Handling of Unknown Words

In order to reduce the training time and to learn better word representations, NNLMs are often trained on most frequent vocabulary words only and low frequency words are represented under a class of unknown words, `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.[4]

Our NDAM models rely on scoring out-domain sequences (of word Ids) 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 based on which the model was trained. In doing so, the out-domain words that are unknown to the in-domain data map to the same `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 from the reduced in-domain vocabulary, and actual unknown words that come from the out-domain data. We introduce a new class $\texttt{unk}_o$ to represent the latter. We train the in-domain model by adding a few dummy sequences containing $\texttt{unk}_o$ occurring on both source and target sides. This enables the model to learn `unk` and $\texttt{unk}_o$ separately, where $\texttt{unk}_o$ is a less probable class according to the model. Later, the n-gram sequences of the out-domain data contain both `unk` and $\texttt{unk}_o$ 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., $\texttt{unk}_o$).

5 Evaluation

In this section, we describe the experimental setup (i.e., data, settings for NN models and MT pipeline) and the results. First we evaluate our models intrinsically by comparing the perplexities on a held-out in-domain testset against the baseline NNJM model. Then we carry out an extrinsic evaluation by using the NNJM and NDAM models as features in machine translation and compare the BLEU scores. Initial developmental experiments were done on the Arabic-to-English language pair.

[4]For our Arabic-English in-domain data, 30% of n-gram sequences contain at least one `unk` word.

We carried out further experiments on the English-to-German pair to validate our models.

5.1 Data

We experimented with the data made publicly 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 corpus. For Arabic-to-English, we used the QCRI Educational Domain (QED) – A bilingual collection of educational lectures[5] (Abdelali et al., 2014), the News, and the multiUN (UN) (Eisele and Chen, 2010) as our out-domain corpora. For English-to-German, we used the News, the Europarl (EP), and the Common Crawl (CC) corpora made available for the 9^{th} Workshop of Statistical Machine Translation.[6] Table 1 shows the size of the data used.

Training NN models is expensive. We, therefore, randomly selected subsets of about 300K sentences from the bigger domains (UN, CC and EP) to train the NN models.[7] The systems were tuned on concatenation of the dev. and test2010 and evaluated on test2011-2013 datasets. The tuning set was also used to measure the perplexities of different models.

5.2 System Settings

NNJM & NDAM: The NNJM models were trained using NPLM[8] toolkit (Vaswani et al., 2013) with the following settings. We used a target context of 5 words and an aligned source window of 9 words, forming a joint stream of 14-grams for training. We restricted source and target side vocabularies to the 20K and 40K most frequent words. 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 ascent with NCE using 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.

AR-EN			EN-DE		
Corpus	Sent.	Tok.	Corpus	Sent.	Tok.
IWSLT	150k	2.8/3.0	IWSLT	177K	3.5/3.3
QED	150k	1.4/1.5	CC	2.3M	57/53
NEWS	203k	5.6/6.3	NEWS	200K	2.8/3.4
UN	3.7M	129/125	EP	1.8M	51/48

Table 1: Statistics of the Arabic-English and English-German training corpora in terms of Sentences and Tokens (Source/Target). Tokens are represented in millions.

Machine Translation System: We trained a Moses system (Koehn et al., 2007), with the following settings: a maximum sentence length of 80, Fast-Aligner for word-alignments (Dyer et al., 2013), an interpolated Kneser-Ney smoothed 5-gram language model with KenLM (Heafield, 2011), lexicalized reordering model (Galley and Manning, 2008), a 5-gram operation sequence model (Durrani et al., 2013) and other default parameters. We also used an NNJM trained with the settings described above as an additional feature in our baseline system. In adapted systems, we replaced the NNJM model with the NDAM models. We used ATB segmentation using the Stanford ATB segmenter (Green and DeNero, 2012) for Arabic-to-English and the default tokenizer provided with the Moses toolkit (Koehn et al., 2007) for the English-to-German pair. Arabic OOVs were translated using an unsupervised transliteration module in Moses (Durrani et al., 2014). We used k-best batch MIRA (Cherry and Foster, 2012) for tuning.

5.3 Intrinsic Evaluation

In this section, we compare the NNJM model and our NDAM models in terms of their perplexity numbers on the in-domain held-out dataset (i.e., dev+test2010). We choose Arabic-English language pair for the development experiments and train domain-wise models to measure the relatedness of each domain with respect to the in-domain. We later replicated selective experiments for the English-German language pair.

The first part of Table 2 summarizes the results for Arabic-English. The perplexity numbers in the second column (NNJM_b) show that NEWS is the

[5]Guzmán et al., 2013) showed that the QED corpus is similar to IWSLT and adding it improves translation quality.

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

[7]Concatenating all the data results in a corpus of approximately 4.5 million sentences which requires roughly 18 days of wall-clock time (18 hours/epoch on a Linux Ubuntu 12.04.5 LTS running on a 16 Core Intel Xeon E5-2650 2.00Ghz and 64Gb RAM) to train NNJM models on our machines. We ran one baseline experiment with all the data and did not find it better than the system trained on randomly selected subset of the data. In the interest of time, we therefore reduced the NN training to a subset (800K and 1M sentences for AR-EN and EN-DE respectively).

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

Domain	NNJM$_b$	NNJM$_{cat}$	NDAM$_{v1}$	NDAM$_{v2}$
Arabic-English				
IWSLT	12.55	-	-	-
QED	61.34	11.72	**11.14**	11.15
NEWS	42.88	10.88	10.67	**10.59**
UN	111.11	11.25	10.83	**10.74**
ALL	-	10.31	**10.08**	10.22
English-German				
IWSLT	10.20	–	–	–
ALL	-	6.71	**6.21**	**6.37**

Table 2: Comparing the perplexity of NNJM and NDAM models. NNJM$_b$ represents the model trained on each individual domain separately.

most related domain from the perspective of in-domain data, whereas UN is the farthest having the worst perplexity. The third column (NNJM$_{cat}$) shows results of the models trained from concatenating each domain to the in-domain data. The perplexity numbers improved significantly in each case showing that there is useful information available in each domain which can be utilized to improve the baseline. It also shows the robustness of neural network models. Unlike the n-gram model, the NN-based model improves generalization with the increase in data without completely skewing towards the dominating part of the data.

Concatenating in-domain with the NEWS data gave better perplexities than other domains. Best results were obtained by concatenating all the data together (See row *ALL*). The third and fourth columns show results of our models (NDAM$_{v*}$). Both give better perplexities than NNJM$_{cat}$ in all cases. However, it is unclear which of the two is better. Similar observations were made for the English-to-German pair, where we only did experiments on the concatenation of all domains.

5.4 Extrinsic Evaluation

Arabic-to-English: For most language pairs, the conventional wisdom is to train the system with all available data. However, previously reported MT results on Arabic-to-English (Mansour and Ney, 2013) show that this is not optimal and the results are often worse than only using in-domain data. The reason for this is that the UN domain is found to be distant and overwhelmingly large as compared to the in-domain IWSLT data. We carried out domain-wise experiments and also found this to be true.

We considered three baseline systems: (*i*) B$_{in}$, which is trained on the in-domain data, (*ii*) B$_{cat}$, which is trained on the concatenation of in- and out-domain data, and (*ii*) B$_{cat,in}$, where the MT pipeline was trained on the concatenation but the NNJM model is trained only on the in-domain data. Table 3 reports average BLEU scores across three test sets on all domains. Adding QED and NEWS domains gave improvements on top of the in-domain IWSLT baseline. Concatenation of UN with in-domain made the results worse. Concatenating all out-domain and in-domain data achieves +0.4 BLEU gain on top of the baseline in-domain system. We will use B$_{cat}$ systems as our baseline to compare our adapted systems with.

SYS	IWSLT	QED	NEWS	UN	ALL
B$_{in}$	26.1	-	-	-	-
B$_{cat}$	-	26.2	26.7	25.8	26.5
B$_{cat,in}$	-	26.2	26.3	25.9	26.5

Table 3: Results of the baseline Arabic-to-English MT systems. The numbers are averaged over tst2011-2013.

Table 4 shows results of the MT systems S$_{v1}$ and S$_{v2}$ using our adapted models NDAM$_{v1}$ and NDAM$_{v2}$. We compare them to the baseline system B$_{cat}$, which uses the non-adapted NNJM$_{cat}$ as a feature. S$_{v1}$ achieved an improvement of up to +0.4 and S$_{v2}$ achieved an improvement of up to +0.5 BLEU points. However, S$_{v2}$ performs slightly worse than S$_{v1}$ on individual domains. We speculate this is because of the nature of the NDAM$_{v2}$, which gives high weight to out-domain sequences that are liked by the in-domain model and disliked by the out-domain model. In the case of individual domains, NDAM$_{v2}$ might be over penalizing out-domain since the out-domain model is only built on that particular domain and always prefers it more than the in-domain model. In case of *ALL*, the out-domain model is more diverse and has different level of likeness for each domain.

We analyzed the output of the baseline system (S$_{cat}$) and spotted several cases of lexical ambiguity caused by 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 in the context of in-domain. The bias created due to the out-domain data caused S$_{cat}$ to choose the contextually incorrect translation *unwanted pregnancy*. However, the adapted systems S$_{v*}$ were able to translate it

	QED			NEWS			UN			ALL		
	tst11	tst12	tst13	tst11	tst12	tst13	tst11	tst12	tst13	tst11	tst12	tst13
B_{cat}	25.0	27.3	26.2	25.4	27.6	27.1	24.7	27.0	25.8	25.0	27.5	27.0
S_{v1}	25.2	27.7	26.2	25.8	27.8	27.3	24.7	27.5	26.1	25.3	27.8	27.0
Δ	**+0.2**	**+0.4**	0.0	**+0.4**	**+0.2**	**+0.2**	0.0	**+0.5**	**+0.3**	**+0.3**	**+0.2**	0.0
S_{v2}	25.1	27.6	26.2	25.6	27.9	27.2	24.6	27.2	26.1	25.5	27.9	26.9
Δ	+0.1	**+0.3**	0.0	**+0.2**	**+0.3**	+0.1	-0.1	**+0.2**	**+0.3**	**+0.5**	**+0.4**	-0.1

Table 4: Arabic-to-English MT Results

SYS	tst11	tst12	tst13	Avg
	Baselines			
B_{in}	25.0	22.5	23.2	23.6
B_{cat}	25.7	22.9	24.1	24.2
$B_{cat,in}$	26.0	22.4	23.6	24.0
	Comparison against NDAM			
B_{cat}	25.7	22.9	24.1	24.2
S_{v1}	26.3	23.1	24.5	24.6
Δ	**+0.6**	**+0.2**	**+0.4**	**+0.4**
S_{v2}	26.2	23.0	24.6	24.6
Δ	**+0.5**	+0.1	**+0.5**	**+0.4**

Table 5: English-to-German MT Results

SYS	tst11	tst12	tst13	Avg
	Arabic-to-English			
B_{cat}	25.0	27.5	27.0	26.5
S_{v1}	25.3	27.8	27.0	26.7
B_{mml}	25.5	27.8	26.8	26.7
S_{v1+mml}	25.5	28.2	27.2	27.0
	English-to-German			
B_{cat}	25.7	22.9	24.1	24.2
S_{v1}	26.3	23.1	24.5	24.6
B_{mml}	25.1	22.7	23.9	23.9
S_{v1+mml}	25.4	22.8	23.9	24.0

Table 6: Comparison with Modified Moore-Lewis

correctly. In another example ماذا عن لياقة البدن؟ (*How about fitness?*), the word لياقة is translated to *proprietary* by S_{cat}, a translation frequently observed in the out-domain data. S_{v*} translated it correctly to *fitness*, as preferred by the in-domain.

English-to-German: Concatenating all training data to train the MT pipeline has been shown to give the best results for English-to-German (Birch et al., 2014). Therefore, we did not do domain-wise experiments, except for training a system on the in-domain IWSLT data for the sake of completeness. We also tried $B_{cat,in}$ variation, i.e. training an MT system on the entire data and using in-domain data to train the baseline NNJM. The baseline system B_{cat} gave better results and was used as our reference for comparison.

Table 5 shows the results of our systems, S_{v1} and S_{v2}, compared to the baselines, B_{in} and B_{cat}. Unlike Arabic-to-English, the baseline system B_{in} is much worse than B_{cat}. Our adapted MT systems S_{v1} and S_{v2} both outperformed the best baseline system (B_{cat}) with an improvement of up to 0.6 points. S_{v2} performed slightly better than S_{v1} on one occasion and slightly worse in others.

Comparison with Data Selection: We also compared our results with the MML-based data selection approach as shown in Table 6. The MML-based baseline systems (B_{mml}) used 20% selected data for training the MT system and the NNJM. On Arabic-English, both MML-based selection and our model (S_{v1}) gave similar gains on top of the baseline system (B_{cat}). Further results showed that both approaches are complementary. We were able to obtain an average gain of +0.3 BLEU points by training an $NDAM_{v1}$ model over the selected data (see S_{v1+mml}).

However, on English-German, the MML-based selection caused a drop in the performance (see Table 6). Training an adapted $NDAM_{v1}$ model over selected data gave improvements over MML in two test sets but could not restore the baseline performance, probably because the useful data has already been filtered by the selection process.

6 Conclusion

We presented two novel models for domain adaptation based on NNJM. Adaptation is performed by regularizing the loss function towards the in-domain model and away from the unrelated out-of-domain data. Our models show better perplexities than the non-adapted baseline NNJM models. When integrated into a machine translation system, gains of up to 0.5 and 0.6 BLEU points were obtained in Arabic-to-English and English-to-German systems over strong baselines.

References

Abdelali, A., Guzman, F., Sajjad, H., and 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., and 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., and Gao, J. (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.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Birch, A., Huck, M., Durrani, N., Bogoychev, N., and 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*, IWSLT '14, Lake Tahoe, CA, USA.

Bisazza, A., Ruiz, N., and 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)*, pages 136–143.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA.

Carpuat, M., Daume III, H., Henry, K., Irvine, A., Jagarlamudi, J., and Rudinger, R. (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.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. (2014). Report on the 11th IWSLT Evaluation Campaign. *Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US*.

Chen, B., Foster, G., and 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., and 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.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Cherry, C. and 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*, NAACL-HLT '12, Montréal, Canada.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. volume 12, pages 2493–2537. JMLR. org.

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

Duh, K., Neubig, Graham, S. K., and 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., and Schmid, H. (2013). Model With Minimal Translation Units, But Decode With Phrases. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'13)*, Atlanta, Georgia, USA.

Durrani, N., Sajjad, H., Hoang, H., and 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., and Vogel, S. (2015). Using Joint Models for Domain Adaptation in Statistical Machine Translation. In *Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV)*, Florida, USA. AMTA.

Durrani, N., Schmid, H., and 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.

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

Eidelman, V., Boyd-Graber, J., and Resnik, P. (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.

Eisele, A. and 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.

Finch, A. and 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., and 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. and Kuhn, R. (2007). Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07.

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

Fujii, A., Utiyama, M., Yamamoto, M., and Utsuro, T. (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.

Galley, M. and 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*, pages 848–856, Honolulu, Hawaii.

Gao, J., He, X., Yih, W.-t., and 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.

Green, S. and DeNero, J. (2012). A class-based agreement model for generating accurately inflected translations. In *Proceedings of the Association for Computational Linguistics (ACL'12)*, Jeju Island, Korea.

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

Guzmán, F., Sajjad, H., Vogel, S., and Abdelali, A. (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)*.

Hasler, E., Blunsom, P., Koehn, P., and Haddow, B. (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.

Heafield, K. (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.

Hildebrand, A. S., Eck, M., Vogel, S., and 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., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*.

Hoang, C. and Sima'an, K. (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*.

Kalchbrenner, N. and 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., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Liu, L., Hong, Y., Liu, H., Wang, X., and 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.

Mansour, S. and 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.

Mathur, P., Venkatapathy, S., and Cancedda, N. (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.

Matsoukas, S., Rosti, A.-V. I., and 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*, EMNLP '09.

Mikolov, T. (2012). *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. HLT-NAACL, pages 746–751.

Mnih, A. and 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. and Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the Association for Computational Linguistics (ACL'10)*, Uppsala, Sweden.

Nakov, P. and 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., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA, USA.

Pascanu, R., Mikolov, T., and 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*.

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., Schwenk, H., and Aransa, W. (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.

Socher, R., Bauer, J., Manning, C. D., and 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)*, pages 455–465, Sofia, Bulgaria.

Vaswani, A., Zhao, Y., Fossum, V., and 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*.