Alignment-Based Neural Machine Translation

Tamer Alkhouli, Gabriel Bretschner,
Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta and Hermann Ney
Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Aachen, Germany
{surname}@cs.rwth-aachen.de

Abstract

Neural machine translation (NMT) has emerged recently as a promising statistical machine translation approach. In NMT, neural networks (NN) are directly used to produce translations, without relying on a pre-existing translation framework. In this work, we take a step towards bridging the gap between conventional word alignment models and NMT. We follow the hidden Markov model (HMM) approach that separates the alignment and lexical models. We propose a neural alignment model and combine it with a lexical neural model in a log-linear framework. The models are used in a standalone word-based decoder that explicitly hypothesizes alignments during search. We demonstrate that our system outperforms attention-based NMT on two tasks: IWSLT 2013 German→English and BOLT Chinese→English. We also show promising results for re-aligning the training data using neural models.

1 Introduction

Neural networks have been gaining a lot of attention recently in areas like speech recognition, image recognition and natural language processing. In machine translation, NNs are applied in two main ways: In N-best rescoring, the neural model is used to score the first-pass decoding output, limiting the model to a fixed set of hypotheses (Le et al., 2012; Sundermeyer et al., 2014a; Hu et al., 2014; Guta et al., 2015). The second approach integrates the NN into decoding, potentially allowing it to directly determine the search space.

There are two approaches to use neural models in decoding. The first integrates the models into phrase-based decoding, where the models are used to score phrasal candidates hypothesized by the decoder (Vaswani et al., 2013; Devlin et al., 2014; Alkhouli et al., 2015). The second approach is referred to as neural machine translation, where neural models are used to hypothesize translations, word by word, without relying on a pre-existing framework. In comparison to the former approach, NMT does not restrict NNs to predetermined translation candidates, and it does not depend on word alignment concepts that have been part of building state-of-the-art phrase-based systems. In such systems, the HMM and the IBM models developed more than two decades ago are used to produce Viterbi word alignments, which are used to build standard phrase-based systems. Existing NMT systems either disregard the notion of word alignments entirely (Sutskever et al., 2014), or rely on a probabilistic notion of alignments (Bahdanau et al., 2015) independent of the conventional alignment models.

Most recently, Cohn et al. (2016) designed neural models that incorporate concepts like fertility and Markov conditioning into their structure. In this work, we also focus on the question whether conventional word alignment concepts can be used for NMT. In particular, (1) We follow the HMM approach to separate the alignment and translation models, and use neural networks to model alignments and translation. (2) We introduce a lexicalized alignment model to capture source reordering information. (3) We bootstrap the NN training using Viterbi word alignments obtained from the HMM and IBM model training, and use the trained neural models to generate new alignments. The new alignments are then used to re-train the neural networks. (4) We design an alignment-based decoder that hypothesizes the alignment path along with the associated translation. We show competitive results in comparison to attention-based

Proceedings of the First Conference on Machine Translation, Volume 1: Research Papers, pages 54–65,
Berlin, Germany, August 11-12, 2016.

models on the IWSLT 2013 German→English and BOLT Chinese→English task.

1.1 Motivation

Attention-based NMT computes the translation probability depending on an intermediate computation of an alignment distribution. The alignment distribution is used to choose the positions in the source sentence that the decoder attends to during translation. Therefore, the alignment model can be considered as an implicit part of the translation model. On the other hand, separating the alignment model from the lexical model has its own advantages: First, this leads to more flexibility in modeling and training: not only can the models be trained separately, but they can also have different model types, e.g. neural models, count-based models, etc. Second, the separation avoids propagating errors from one model to the other. In attention-based systems, the translation score is based on the alignment distribution, which risks propagating errors from the alignment part to the translation part. Third, using separate models makes it possible to assign them different weights. We exploit this and use a log-linear framework to combine them. We still retain the possibility of joint training, which can be performed flexibly by alternating between model training and alignment generation. The latter can be performed using forced-decoding.

In contrast to the count-based models used in HMMs, we use neural models, which allow covering long context without having to explicitly address the smoothing problem that arises in count-based models.

2 Related Work

Most recently, NNs have been trained on large amounts of data, and applied to translate independent of the phrase-based framework. Sutskever et al. (2014) introduced the pure encoder-decoder approach, which avoids the concept of word alignments. Bahdanau et al. (2015) introduced an attention mechanism to the encoder-decoder approach, allowing the decoder to attend to certain source words. This method was refined in (Luong et al., 2015) to allow for local attention, which makes the decoder attend to representations of source words residing within a window. These translation models have shown competitive results, outperforming phrase-based systems when using ensembles on tasks like IWSLT English→German 2015 (Luong and Manning, 2015).

In this work, we follow the same standalone neural translation approach. However, we have a different treatment of alignments. While the attention-based soft-alignment model computes an alignment distribution as an intermediate step within the neural model, we follow the hard alignment concept used in phrase extraction. We separate the alignment model from the lexical model, and train them independently. At translation time, the decoder hypothesizes and scores the alignment path in addition to the translation.

Cohn et al. (2016) introduce several modifications to the attention-based model inspired by traditional word alignment concepts. They modify the network architecture, adding a first-order dependence by making the attention vector computed for a target position directly dependent on that of the previous position. Our alignment model has a first-order dependence that takes place at the input and output of the model, rather than an architectural modification of the neural network.

Yang et al. (2013) use NN-based lexical and alignment models, but they give up the probabilistic interpretation and produce unnormalized scores instead. Furthermore, they model alignments using a simple distortion model that has no dependence on lexical context. The models are used to produce new alignments which are in turn used to train phrase systems. This leads to no significant difference in terms of translation performance. Tamura et al. (2014) propose a lexicalized RNN alignment model. The model still produces non-probabilistic scores, and is used to generate word alignments used to train phrase-based systems. In this work, we develop a feed-forward neural alignment model that computes probabilistic scores, and use it directly in standalone decoding, without constraining it to the phrase-based framework. In addition, we use the neural models to produce alignments that are used to re-train the same neural models.

Schwenk (2012) proposed a feed-forward network that computes phrase scores offline, and the scores were added to the phrase table of a phrase-based system. Offline phrase scoring was also done in (Alkhouli et al., 2014) using semantic phrase features obtained using simple neural networks. In comparison, our work does not rely on the phrase-based system, rather, the neural net-

works are used to hypothesize translation candidates directly, and the scores are computed online during decoding.

We use the feed-forward joint model introduced in (Devlin et al., 2014) as a lexical model, and introduce a lexicalized alignment model based on it. In addition, we modify the bidirectional joint model presented in (Sundermeyer et al., 2014a) and compare it to the feed-forward variant. These lexical models were applied in phrase-based systems. In this work, we apply them in a standalone NMT framework.

Forced alignment was applied to train phrase tables in (Wuebker et al., 2010; Peitz et al., 2012). We generate forced alignments using a neural decoder, and use them to re-train neural models.

Tackling the costly normalization of the output layer during decoding has been the focus of several papers (Vaswani et al., 2013; Devlin et al., 2014; Jean et al., 2015). We propose a simple method to speed up decoding using a class-factored output layer with almost no loss in translation quality.

3 Statistical Machine Translation

In statistical machine translation, the target word sequence $e_1^I = e_1, ..., e_I$ of length I is assigned a probability conditioned on the source word sequence $f_1^J = f_1, ..., f_J$ of length J. By introducing word alignments as hidden variables, the posterior probability $p(e_1^I|f_1^J)$ can be computed using a lexical and an alignment model as follows.

$$
\begin{aligned}
& p(e_1^I|f_1^J) \\
&= \sum_{b_1^I} p(e_1^I, b_1^I|f_1^J) \\
&= \sum_{b_1^I} \prod_{i=1}^{I} p(e_i, b_i|b_1^{i-1}, e_1^{i-1}, f_1^J) \\
&= \sum_{b_1^I} \prod_{i=1}^{I} \underbrace{p(e_i|b_1^i, e_1^{i-1}, f_1^J)}_{\text{lexical model}} \cdot \underbrace{p(b_i|b_1^{i-1}, e_1^{i-1}, f_1^J)}_{\text{alignment model}}
\end{aligned}
$$

where $b_1^I = b_1, ..., b_I$ denotes the alignment path, such that b_i aligns the target word e_i to the source word f_{b_i}. In this general formulation, the lexical model predicts the target word e_i conditioned on the source sentence, the target history, and the alignment history. The alignment model is lexicalized using the source and target context as well. The sum over alignment paths is replaced by the maximum during decoding (cf. Section 5).

4 Neural Network Models

There are two common network architectures used in machine translation: feed-forward NNs (FFNN) and recurrent NNs (RNN). In this section we will discuss alignment-based feed-forward and recurrent neural networks. These networks are conditioned on the word alignment, in addition to the source and target words.

4.1 Feed-forward Joint Model

We adopt the feed-forward joint model (FFJM) proposed in (Devlin et al., 2014) as the lexical model. The authors demonstrate the model has a strong performance when applied in a phrase-based framework. In this work we explore its performance in standalone NMT. The model was introduced along with heuristics to resolve unaligned and multiply aligned words. We denote the heuristic-based source alignment point corresponding to the target position i by $\hat{b}_i$. The model is defined as

$$p(e_i|b_1^i, e_1^{i-1}, f_1^J) = p(e_i|e_{i-n}^{i-1}, f_{\hat{b}_i-m}^{\hat{b}_i+m}) \qquad (1)$$

and it computes the probability of a target word e_i at position i given the n-gram target history $e_{i-n}^{i-1} = e_{i-n}, ..., e_{i-1}$, and a window of $2m+1$ source words $f_{\hat{b}_i-m}^{\hat{b}_i+m} = f_{\hat{b}_i-m}, ..., f_{\hat{b}_i+m}$ centered around the word $f_{\hat{b}_i}$.

As the heuristics have implications on our alignment-based decoder, we explain them by the examples shown in Figure 1. We mark the source and target context by rectangles on the x- and y-axis, respectively. The left figure shows a single source word 'Jungen' aligned to a single target word 'offspring', in which case, the original source position is used, i.e., $\hat{b}_i = b_i$. If the target word is aligned to multiple source words, as it is the case with the words 'Mutter Tiere' and 'Mothers' in the middle figure, then $\hat{b}_i$ is set to the middle alignment point. In this example, the left alignment point associated with 'Mutter' is selected. The right figure shows the case of the unaligned target word 'of'. $\hat{b}_i$ is set to the source position associated with the closest aligned target word 'full', preferring right to left. Note that this model does not have special handling of unaligned source words. While these words can be covered indirectly by source windows associated with aligned source words, the model does not explicitly score them.

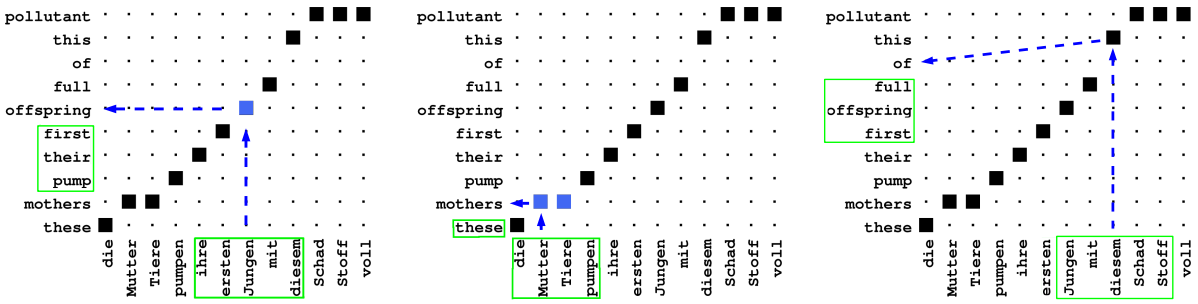

Figure 1: Examples on resolving word alignments to obtain word affiliations.

Computing normalized probabilities is done using the softmax function, which requires computing the full output layer first, and then computing the normalization factor by summing over the output scores of the full vocabulary. This is very costly for large vocabularies. To overcome this, we adopt the class-factored output layer consisting of a class layer and a word layer (Goodman, 2001; Morin and Bengio, 2005). The model in this case is defined as

$$p(e_i|e_{i-n}^{i-1}, f_{\hat{b}_i-m}^{\hat{b}_i+m}) = p(e_i|c(e_i), e_{i-n}^{i-1}, f_{\hat{b}_i-m}^{\hat{b}_i+m}) \cdot p(c(e_i)|e_{i-n}^{i-1}, f_{\hat{b}_i-m}^{\hat{b}_i+m})$$

where c denotes a word mapping that assigns each target word to a single class, where the number of classes is chosen to be much smaller than the vocabulary size $|C| << |V|$. Even though the full class layer needs to be computed, only a subset of the significantly-larger word layer has to be considered, namely the words that share the same class $c(e_i)$ with the target word e_i. This helps speeding up training on large-vocabulary tasks.

4.2 Bidirectional Joint Model

The bidirectional RNN joint model (BJM) presented in (Sundermeyer et al., 2014a) is another lexical model. The BJM uses the full source sentence and the full target history for prediction, and it is computed by reordering the source sentence following the target order. This requires the complete alignment information to compute the model scores. Here, we introduce a variant of the model that is conditioned on the alignment history instead of the full alignment path. This is achieved by computing forward and backward representations of the source sentence in its original order, as done in (Bahdanau et al., 2015). The model is given by

$$p(e_i|b_1^i, e_1^{i-1}, f_1^J) = p(e_i|\hat{b}_1^i, e_1^{i-1}, f_1^J)$$

Note that we also use the same alignment heuristics presented in Section 4.1. As this variant does not require future alignment information, it can be applied in decoding. However, in this work we apply this model in rescoring and leave decoder integration to future work.

4.3 Feed-forward Alignment Model

We propose a neural alignment model to score alignment paths. Instead of predicting the absolute positions in the source sentence, we model the jumps from one source position to the next position to be translated. The jump at target position i is defined as $\Delta_i = \hat{b}_i - \hat{b}_{i-1}$, which captures the jump from the source position $\hat{b}_{i-1}$ to $\hat{b}_i$. We modify the FFNN lexical model to obtain a feed-forward alignment model. The feed-forward alignment model (FFAM) is given by

$$p(b_i|b_1^{i-1}, e_1^{i-1}, f_1^J) = p(\Delta_i|e_{i-n}^{i-1}, f_{\hat{b}_{i-1}-m}^{\hat{b}_{i-1}+m}) \quad (2)$$

This is a lexicalized alignment model conditioned on the n-gram target history and the $(2m+1)$-gram source window. Note that, different from the FFJM, the source window of this model is centered around the source position $\hat{b}_{i-1}$. This is because the model needs to predict the jump to the next source position $\hat{b}_i$ to be translated. The alignment model architecture is shown in Figure 2.

In contrast to the lexical model, the output vocabulary of the alignment model is much smaller, and therefore we use a regular softmax output layer for this model without class-factorization.

4.4 Feed-forward vs. Recurrent Models

RNNs have been shown to outperform feed-forward variants in language and translation modeling. Nevertheless, feed-forward networks have their own advantages: First, they are typically

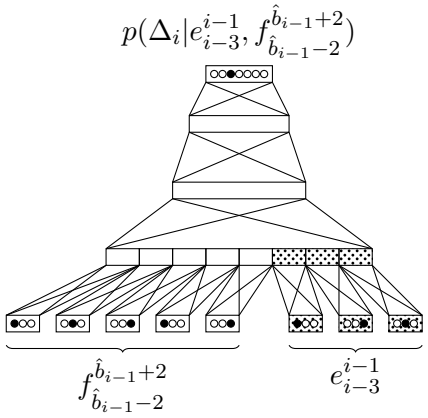

Figure 2: A feed-forward alignment NN, with 3 target history words, 5-gram source window, a projection layer, 2 hidden layers, and a small output layer to predict jumps.

faster to train due to their simple architecture, and second, they are more flexible to integrate into beam search decoders. This is because feed-forward networks only depend on a limited context. RNNs, on the other hand, are conditioned on an unbounded context. This means that the complete hypotheses during decoding have to be maintained without any state recombination. Since feed-forward networks allow the use of state recombination, they are potentially capable of exploring more candidates during beam search.

5 Alignment-based Decoder

In this section we present the alignment-based decoder. This is a beam-search word-based decoder that predicts one target word at a time. As the models we use are alignment-based, the decoder hypothesizes the alignment path. This is different from the NMT approaches present in the literature, which are based on models that either ignore word alignments or compute alignments as part of the attention-based model.

In the general case, a word can be aligned to a single word, multiple words, or it can be unaligned. However, we do not use the general word alignment notion, rather, the models are based on alignments derived using the heuristics discussed in Section 4. These heuristics simplify the task of the decoder, as they induce equivalence classes over the alignment paths, reducing the number of possible alignments the decoder has to hypothesize significantly. As a result of using these heuristics, the task of hypothesizing alignments is reduced to enumerating all J source positions a target word can be aligned to. The following is a list of the possible alignment scenarios and how the decoder covers them.

Algorithm 1 Alignment-based Decoder

procedure TRANSLATE(f_1^J, $beamSize$)
  $hyps \leftarrow initHyp$ ▷previous set of partial hypotheses
  $newHyps \leftarrow \emptyset$ ▷current set of partial hypotheses
  while GETBEST($hyps$) **not** terminated **do**
    ▷compute alignment distribution in batch mode
    $alignDists \leftarrow$ ALIGNMENTDISTRIBUTION($hyps$)
    ▷hypothesize source alignment points
    for pos **From** 1 **to** J **do**
      ▷compute lexical distributions of all
      ▷hypotheses in hyps in batch mode
      $dists \leftarrow$ LEXICALDISTRIBUTION($hyps$, pos)
      ▷expand each of the previous hypotheses
      for hyp **in** $hyps$ **do**
        $jmpCost \leftarrow$ SCORE($alignDists$, hyp, pos)
        $dist \leftarrow$ GETDISTRIBUTION($dists$, hyp)
        $dist \leftarrow$ PARTIALSORT($dist$,$beamSize$)
        $cnt \leftarrow 0$
        ▷hypothesize new target word
        for $word$ **in** $dist$ **do**
          if $cnt > beamSize$ **then**
            break
          $newHyp \leftarrow$ EXTEND(hyp,$word$,pos,$jmpCost$)
          $newHyps$.INSERT($newHyp$)
          $cnt \leftarrow cnt + 1$
    PRUNE($newHyps$, $beamSize$)
    $hyps \leftarrow newHyps$
  ▷return the best scoring hypothesis
  return GETBEST($hyps$)

- Multiply-aligned target words: the heuristic chooses the middle link as an alignment point. Therefore, the decoder is able to cover these cases by hypothesizing J many source positions for each target word hypothesis.
- Unaligned target words: the heuristic aligns these words using the nearest aligned target word in training (cf. Figure 1, right). In decoding, these words are handled as aligned words.
- Multiply-aligned source words: covered by revisiting a source position that has already been translated.
- Unaligned source words: result if no target word is generated using a source window centered around the source word in question.

The decoder is shown in Algorithm 1. It involves hypothesizing alignments and translation

words. Alignments are hypothesized in the loop starting at line 8. Once an alignment point is set to position pos, the lexical distribution over the full target vocabulary is computed using this position in line 11. The distribution is sorted and the best candidate translations lying within the beam are used to expand the partial hypotheses.

We batch the NN computations, calling the alignment and lexical networks for all partial hypotheses in a single call to speed up computations as shown in lines 6 and 11. We also exploit the beam and apply partial sorting in line 16, instead of completely sorting the list. Partial sorting has a linear complexity on average, and it returns a list whose first $beamSize$ words have better scores compared to the rest of the list.

We terminate translation if the best scoring partial hypothesis ends with the sentence end symbol. If a hypothesis terminates but it scores worse than other hypotheses, it is removed from the beam, but it still competes with non-terminated hypotheses. Note that we do not have any explicit coverage constraints. This means that a source position can be revisited many times, hence generating one-to-many alignment cases. This also allows having unaligned source words.

In the alignment-based decoder, an alignment distribution is computed, and word alignments are hypothesized and scored using this distribution, leading alignment decisions to become part of beam search. The search space is composed of both alignment and translation decisions. In contrast, the search space in attention-based decoding is composed of translation decisions only.

Class-Factored Output Layer in Decoding

The large output layer used in language and translation modeling is a major bottleneck in evaluating the network. Several papers discuss how to evaluate it efficiently during decoding using approximations. In this work, we exploit the class-factored output layer to speed up training. At decoding time, the network needs to hypothesize all target words, which means the full output layer should be evaluated. In the case of using a class-factored output layer, this results in an additional computational overhead from computing the class layer. In order to speed up decoding, we propose to use the class layer to choose the top scoring k classes, then we evaluate the word layer for each of these classes only. We show this leads to a significant speed up with minimal loss in translation quality.

Model Combination

We embed the models in a log-linear framework, which is commonly used in phrase-based systems. The goal of the decoder is to find the best scoring hypothesis as follows.

$$\hat{e}_1^{\hat{I}} = \arg\max_{I, e_1^I} \left\{ \max_{\hat{b}_1^I} \sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I, \hat{b}_1^I) \right\}$$

where λ_m is the model weight associated with the model h_m, and M is the total number of models. The model weights are automatically tuned using minimum error rate training (MERT) (Och, 2003). Our main system includes a lexical neural model, an alignment neural model, and a word penalty, which is the count of target words. The word penalty becomes important at the end of translation, where hypotheses in the beam might have different final lengths.

6 Forced-Alignment Training

Since the models we use require alignments for training, we initially use word alignments produced using HMM/IBM models using GIZA++ as initial alignments. At first, the FFJM and the FFAM are trained separately until convergence, then the models are used to generate new word alignments by force-decoding the training data as follows.

$$\tilde{b}_1^I(f_1^J, e_1^I) = arg\max_{b_1^I} \prod_{i=1}^{I} p^{\lambda_1}(\Delta_i | e_{i-n}^{i-1}, f_{b_{i-1}-m}^{b_{i-1}+m}) \cdot p^{\lambda_2}(e_i | e_{i-n}^{i-1}, f_{b_i-m}^{b_i+m})$$

where λ_1 and λ_2 are the model weights. We modify the decoder to only compute the probabilities of the target words in the reference sentence. The for loop in line 19 of Algorithm 1 collapses to a single iteration. We use both the the feed-forward joint model (FFJM) and the feed-forward alignment model (FFAM) to perform force-decoding, and the new alignments are used to retrain the models, replacing the initial GIZA++ alignments.

Retraining the neural models using the forced-alignments has two benefits. First, since the alignments are produced using both of the lexical and alignment models, this can be viewed as joint training of the two models. Second, since the neural decoder generates these alignments, training neural models based on them yields models that are more consistent with the neural decoder. We verify this claim in the experiments section.

	IWSLT		BOLT	
	De	**En**	**Zh**	**En**
Sentences	4.32M		4.08M	
Run. Words	108M	109M	78M	86M
Vocab.	836K	792K	384K	817K
FFNN/BJM Vocab.	173K	149K	169K	128K
Attention Vocab.	30K	30K	30K	30K
FFJM params	177M		159M	
BJM params	170M		153M	
FFAM params	101M		94M	
Attention params	84M		84M	

Table 1: Corpora and NN statistics.

7 Experiments

We carry out experiments on two tasks: the IWSLT 2013 German→English shared translation task,[1] and the BOLT Chinese→English task. The corpora statistics are shown in Table 1. The IWSLT phrase-based baseline system is trained on all available bilingual data, and uses a 4-gram LM with modified Kneser-Ney smoothing (Kneser and Ney, 1995; Chen and Goodman, 1998), trained with the SRILM toolkit (Stolcke, 2002). As additional data sources for the LM, we selected parts of the Shuffled News and LDC English Gigaword corpora based on the cross-entropy difference (Moore and Lewis, 2010), resulting in a total of 1.7 billion running words for LM training. The phrase-based baseline is a standard phrase-based SMT system (Koehn et al., 2003) tuned with MERT (Och, 2003) and contains a hierarchical reordering model (Galley and Manning, 2008). The in-domain data consists of 137K sentences.

The BOLT Chinese→English task is evaluated on the "discussion forum" domain. We use a 5-gram LM trained on 2.9 billion running words in total. The in-domain data consists of a subset of 67.8K sentences. We used a set of 1845 sentences as a tune set. The evaluation set `test1` contains 1844 and `test2` contains 1124 sentences.

We use the FFNN architecture for the lexical and alignment models. Both models use a window of 9 source words, and 5 target history words. Both models use two hidden layers, the first has 1000 units and the second has 500 units. The lexical model uses a class-factored output layer, with 1000 singleton classes dedicated to the most frequent words, and 1000 classes shared among the rest of the words. The classes are trained using a separate tool to optimize the maximum likelihood training criterion with the bigram assumption. The alignment model uses a small output layer of 201 nodes, determined by a maximum jump length of 100 (forward and backward). 300 nodes are used for word embeddings. Each of the FFNN models is trained on CPUs using 12 threads, which takes up to 3 days until convergence. We train with stochastic gradient descent using a batch size of 128. The learning rate is halved when the development perplexity increases.

Each BJM has 4 LSTM layers: two for the forward and backward states, one for the target state, and one after merging the source and target states. The size of the word embeddings and hidden layers is 350 nodes. The output layers are identical to those of the FFJM models.

We compare our system to an attention-based baseline similar to the networks described in (Bahdanau et al., 2015). All such systems use single models, rather than ensembles. The word embedding dimension is 620, each direction of the encoder and the decoder has a layer of 1000 gated recurrent units (Cho et al., 2014). Unknowns and numbers are carried out from the source side to the target side based on the largest attention weight.

To speed up decoding of long sentences, the decoder hypothesizes 21 and 41 source positions around the diagonal, for the IWSLT and the BOLT tasks, respectively. We choose these numbers such that the translation quality does not degrade. The beam size is set to 16 in all experiments. Larger beam sizes did not lead to improvements. We apply part-of-speech-based long-range verb reordering rules to the German side in a preprocessing step for all German→English systems (Popović and Ney, 2006), including the baselines. The Chinese→English systems use no such preordering. We use the GIZA++ word alignments to train the models. The networks are fine-tuned by training additional epochs on the in-domain data only (Luong and Manning, 2015). The LMs are only used in the phrase-based systems in both tasks, but not in the NMT systems.

All translation experiments are performed with the *Jane* toolkit (Vilar et al., 2010; Wuebker et al., 2012). The alignment-based NNs are trained using an extension of the *rwthlm* toolkit (Sundermeyer et al., 2014b). We use an implementation based on Blocks (van Merriënboer et al., 2015) and Theano (Bergstra et al., 2010; Bastien et al., 2012) for the attention-based experiments. All results are mea-

[1] `http://www.iwslt2013.org`

#	system	test 2010 BLEU	test 2010 TER	eval 2011 BLEU	eval 2011 TER
1	phrase-based system	28.9	51.0	32.9	46.3
2	+ monolingual data	30.4	49.5	35.4	44.2
3	attention-based RNN	27.9	51.4	31.8	46.5
4	+fine-tuning	29.8	48.9	32.9	45.1
5	FFJM+dp+wp	21.6	56.9	24.7	53.8
6	FFJM+FFAM+wp	26.1	53..1	29.9	49.4
7	+fine-tuning	29.3	50.5	33.2	46.5
8	+BJM Rescoring	**30.0**	**48.7**	**33.8**	**44.8**
9	BJM+FFAM+wp+fine-tuning	29.8	49.5	33.7	45.8

Table 2: IWSLT 2013 German→English results in BLEU [%] and TER [%].

sured in case-insensitive BLEU [%] (Papineni et al., 2002) and TER [%] (Snover et al., 2006) on a single reference. We used the multeval toolkit (Clark et al., 2011) for evaluation.

7.1 IWSLT 2013 German→English

Table 2 shows the IWSLT German→English results. FFJM refers to feed-forward lexical model. We compare against the phrase-based system with an LM trained on the target side of the bilingual data (row #1), the phrase-based system with an LM trained on additional monolingual data (row #2), the attention-based system (row #3), and the attention-based system after fine-tuning towards the in-domain data (row #4). First, we experiment with a system using the FFJM as a lexical model and a linear distortion penalty (dp) to encourage monotone translation as the alignment model. We also include a word penalty (wp). This system is shown in row #5. In comparison, if the distortion penalty is replaced by the feed-forward alignment model (FFAM), we observe large improvements of 4.5% to 5.2% BLEU (row #5 vs. #6). This highlights the significant role of the alignment model in our system. Moreover, it indicates that the FFAM is able to model alignments beyond the simple monotone alignments preferred by the distortion penalty.

Fine-tuning the neural networks towards in-domain data improves the system by up to 3.3% BLEU and 2.9% TER (row #6 vs #7). The gain from fine-tuning is larger than the one observed for the attention-based system. This is likely due to the fact that our system has two neural models, and each of them is fine-tuned.

We apply the BJM in 1000-best list rescoring (row #8). Which gives another boost, leading our system to outperform the attention-based system by 0.9% BLEU on eval 2011, while a comparable performance is achieved on test 2010. In order to highlight the difference between using the FFJM and the BJM, we replace the FFJM scores after obtaining the N-best lists with the BJM scores and apply rescoring (row #9). In comparison to row #7, we observe up to 0.5% BLEU and 1.0% TER improvement. This is expected as the BJM captures unbounded source and target context in comparison to the limited context of the FFJM. This calls for a direct integration of the BJM into decoding, which we intend to do in future work. Our best system (row #8) outperforms the phrase-based system (row #1) by up to 1.1% BLEU and 2.3% TER. While the phrase-based system can benefit from training the LM on additional monolingual data (row #1 vs. #2), exploiting monolingual data in NMT systems is still an open research question.

7.2 BOLT Chinese→English

The BOLT Chinese→English experiments are shown in Table 3. Again, we observe large improvements when including the FFAM in comparison to the distortion penalty (row #5 vs #6), and fine-tuning improves the results considerably. Including the BJM in rescoring improves the system by up to 0.4% BLEU. Our best system (row #8) outperforms the attention-based model by up to 0.4% BLEU and 2.8% TER. We observe that the length ratio of our system's output to the reference is 93.3-94.9%, while it is 99.1-102.6% for the attention-based system. In light of the BLEU and TER scores, the attention-based model does not benefit from matching the reference length. Our system (row #8) still lags behind the phrase-based system (row #1). Note, however, that in the WMT 2016 evaluation campaign,[2] it was demonstrated that NMT can outperform phrase-based systems on several tasks including German→English and English→German. Including monolingual data (Sennrich et al., 2016) in training neural translation models can boost performance, and this can be applied to our system.

7.3 Neural Alignments

Next, we experiment with re-aligning the training data using neural networks as described in Section 6. We use the fine-tuned FFJM and FFAM to realign the in-domain data of the IWSLT German→English task. These models are initially

[2] http://matrix.statmt.org/

#	system	test1 BLEU	test1 TER	test2 BLEU	test2 TER
1	phrase-based system	17.6	68.3	16.9	67.4
2	+ monolingual data	17.9	67.9	17.0	67.1
3	attention-based RNN	14.8	76.1	13.6	76.9
4	+fine-tuning	16.1	73.1	15.4	72.3
5	FFJM+dp+wp	10.1	77.2	9.8	75.8
6	FFJM+FFAM+wp	14.4	71.9	13.7	71.3
7	+fine-tuning	15.8	70.3	15.4	69.4
8	+BJM Rescoring	**16.0**	**70.3**	**15.8**	**69.5**
9	BJM+FFAM+wp+fine-tuning	**16.0**	70.4	15.7	69.7

Table 3: BOLT Chinese→English results in BLEU [%] and TER [%].

Alignment Source	test 2010 BLEU	test 2010 TER	eval 2011 BLEU	eval 2011 TER
GIZA++	25.6	53.6	29.3	49.7
Neural Forced decoding	25.9	52.4	29.5	49.4

Table 4: Re-alignment results in BLEU [%] and TER [%] on the IWSLT 2013 German→English in-domain data. Each system includes FFJM, FFAM and word penalty.

trained using GIZA++ alignments. We train new models using the re-aligned data and compare the translation quality before and after re-alignment. We use 0.7 and 0.3 as model weights for the FFJM and FFAM, respectively. These values are based on the model weights obtained using MERT. The results are shown in Table 4. Note that the baseline is worse than the one in Table 2 as the models are only trained on the in-domain data. We observe that re-aligning the data improves translation quality by up to 0.3% BLEU and 1.2% TER. The new alignments are generated using the neural decoder, and using them to train the neural networks results in training that is more consistent with decoding. As future work, we intend to re-align the full bilingual data and use it for neural training.

7.4 Class-Factored Output Layer

Figure 3 shows the trade-off between speed and performance when evaluating words belonging to the top classes only. Limiting the evaluation to words belonging to the top class incurs a performance loss of 0.4% BLEU only when compared to the full evaluation of the output layer. However, this corresponds to a large speed-up. The system is about 30 times faster, with a translation speed of 0.4 words/sec. In conclusion, not only does the class layer speed up training, but it can also be used to speed up decoding considerably. We use the top 3 classes throughout our experiments.

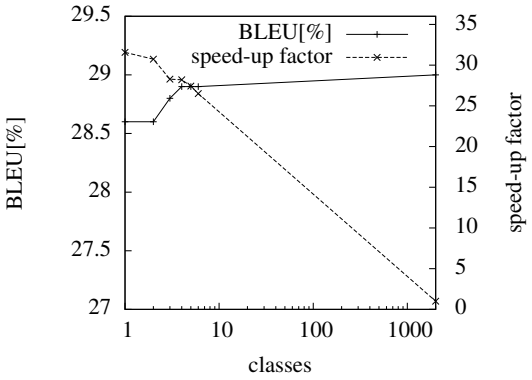

Figure 3: Decoding speed-up and translation quality using top scoring classes in a class-factored output layer. The results are computed for the IWSLT German→English dev dataset.

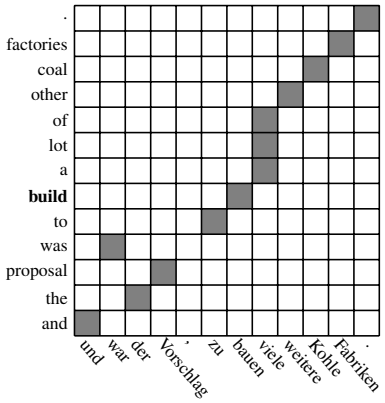

Figure 4: A translation example produced by our system. The shown German sentence is pre-ordered.

8 Analysis

We show an example from the German→English task in Figure 4, along with the alignment path. The reference translation is 'and the proposal has been to build a lot more coal plants .'. Our system handles the local reordering of the word 'was', which is produced in the correct target order. An example on the one-to-many alignments is given by the correct translation of 'viele' to 'a lot of'.

As an example on handling multiply-aligned target words, we observe the translation of 'Nord Westen' to 'northwest' in our output. This is possible because the source window allows the FFNN to translate the word 'Westen' in context of the word 'Nord'.

Table 5 lists some translation examples produced by our system and the attention-based system, where maximum attention weights are used

1	source	sie würden verhungern nicht , und wissen Sie was ?
	reference	they wouldn 't *starve* , and you know what ?
	attention NMT	you wouldn 't **interview** , and guess what ?
	our system	they wouldn 't *starve* , and you know what ?
2	source	denn sie sind diejenigen , die sind auch Experten für Geschmack .
	reference	because they 're the ones that are *experts in flavor* , too .
	attention NMT	because they 're the ones who are also **experts** .
	our system	because they 're the ones who are also *experts in flavor* .
3	source	es ist ein Online Spiel , in dem Sie müssen überwinden eine Ölknappheit .
	reference	this is an online game in which you try to survive an *oil shortage* .
	attention NMT	it 's an online game where you need to get through a **UNKOWN** .
	our system	it 's an online game in which you have to overcome an **astrolabe** .
4	source	es liegt daran , dass gehen nicht Möglichkeiten auf diesem Planeten zurück, sie gehen vorwärts .
	reference	it 's because *possibilities* on this planet , they *don 't go back* , they go forward .
	attention NMT	it 's because **there 's no way back** on this planet , they 're going to move forward .
	our system	it 's because *opportunities don 't* **go** on this planet , they go forward .

Table 5: Sample translations from the IWSLT German→English `test` set using the attention-based system (Table 2, row #4) and our system (Table 2, row #7). We highlight the (pre-ordered) source words and their aligned target words. We underline the source words of interest, italicize *correct* translations, and use bold-face for **incorrect** translations.

as alignment. While we use larger vocabularies compared to the attention-based system, we observe incorrect translations of rare words. E.g., the German word Ölknappheit in sentence 3 occurs only 7 times in the training data among 108M words, and therefore it is an unknown word for the attention system. Our system has the word in the source vocabulary but fails to predict the right translation. Another problem occurs in sentence 4, where the German verb "zurückgehen" is split into "gehen ... zurück". Since the feed-forward model uses a source window of size 9, it cannot include both words when it is centered at any of them. Such insufficient context might be resolved when integrating the bidirectional RNN in decoding. Note that the attention-based model also fails to produce the correct translation here.

9 Conclusion

This work takes a step towards bridging the gap between conventional word alignment concepts and NMT. We use an HMM-inspired factorization of the lexical and alignment models, and employ the Viterbi alignments obtained using conventional HMM/IBM models to train neural models. An alignment-based decoder is introduced and a log-linear framework is used to combine the models. We use MERT to tune the model weights. Our system outperforms the attention-based system on the German→English task by up to 0.9% BLEU, and on Chinese→English by up to 2.8% TER. We also demonstrate that re-aligning the training data using the neural decoder yields better translation quality.

As future work, we aim to integrate alignment-based RNNs such as the BJM into the alignment-based decoder. We also plan to develop a bidirectional RNN alignment model to make jump decisions based on unbounded context. In addition, we want to investigate the use of coverage constraints in alignment-based NMT. Furthermore, we consider the re-alignment experiment promising and plan to apply re-alignment on the full bilingual data of each task.

Acknowledgments

This paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 645452 (QT21). Tamer Alkhouli was partly funded by the 2016 Google PhD Fellowship for North America, Europe and the Middle East.

References

Tamer Alkhouli, Andreas Guta, and Hermann Ney. 2014. Vector space models for phrase-based machine translation. In *EMNLP 2014 Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 1–10, Doha, Qatar, October.

Tamer Alkhouli, Felix Rietig, and Hermann Ney. 2015. Investigations on phrase-based decoding with recurrent neural network language and translation mod-

els. In *Proceedings of the EMNLP 2015 Tenth Workshop on Statistical Machine Translation*, pages 294–303, Lisbon, Portugal, September.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, San Diego, Calefornia, USA, May.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. December.

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

Stanley F. Chen and Joshua Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, August.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar, October.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *49th Annual Meeting of the Association for Computational Linguistics*, pages 176–181, Portland, Oregon, June.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California, June.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, MD, USA, June.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, USA, October.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 561–564, Utah, USA, May.

Andreas Guta, Tamer Alkhouli, Jan-Thorsten Peter, Joern Wuebker, and Hermann Ney. 2015. A Comparison between Count and Neural Network Models Based on Joint Translation and Reordering Sequences. In *Conference on Empirical Methods on Natural Language Processing*, pages 1401–1411, Lisbon, Portugal, September.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29, Gothenburg, Sweden, April.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1–10, Beijing, China, July.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184, May.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics*, pages 127–133, Edmonton, Canada, May/June.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous Space Translation Models with Neural Networks. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montreal, Canada, June.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, Da Nag, Vietnam, December.

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

R.C. Moore and W. Lewis. 2010. Intelligent Selection of Language Model Training Data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 220–224, Uppsala, Sweden, July.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252, Barbados, January.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

Stephan Peitz, Arne Mauser, Joern Wuebker, and Hermann Ney. 2012. Forced derivations for hierarchical machine translation. In *International Conference on Computational Linguistics*, pages 933–942, Mumbai, India, December.

Maja Popović and Hermann Ney. 2006. POS-based word reorderings for statistical machine translation. In *Language Resources and Evaluation*, pages 1278–1283, Genoa, Italy, May.

Holger Schwenk. 2012. Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. In *25th International Conference on Computational Linguistics*, pages 1071–1080, Mumbai, India, December.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of 54th Annual Meeting of the Association for Computational Linguistics*, August.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Speech and Language Processing*, volume 2, pages 901–904, Denver, CO, September.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014a. Translation Modeling with Bidirectional Recurrent Neural Networks. In *Conference on Empirical Methods on Natural Language Processing*, pages 14–25, Doha, Qatar, October.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2014b. rwthlm - the RWTH Aachen university neural network language modeling toolkit. In *Interspeech*, pages 2093–2097, Singapore, September.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112, Montréal, Canada, December.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *52nd Annual Meeting of the Association for Computational Linguistics*, pages 1470–1480, Baltimore, MD, USA.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October.

David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. 2010. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, pages 262–270, Uppsala, Sweden, July.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 475–484, Uppsala, Sweden, July.

Joern Wuebker, Matthias Huck, Stephan Peitz, Malte Nuhn, Markus Freitag, Jan-Thorsten Peter, Saab Mansour, and Hermann Ney. 2012. Jane 2: Open source phrase-based and hierarchical statistical machine translation. In *International Conference on Computational Linguistics*, pages 483–491, Mumbai, India, December.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 166–175, Sofia, Bulgaria, August.