A Comparison between Count and Neural Network Models Based on Joint Translation and Reordering Sequences

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

Abstract

We propose a conversion of bilingual sentence pairs and the corresponding word alignments into novel linear sequences. These are joint translation and reordering (JTR) uniquely defined sequences, combining interdepending lexical and alignment dependencies on the word level into a single framework. They are constructed in a simple manner while capturing multiple alignments and empty words. JTR sequences can be used to train a variety of models. We investigate the performances of n-gram models with modified Kneser-Ney smoothing, feed-forward and recurrent neural network architectures when estimated on JTR sequences, and compare them to the operation sequence model (Durrani et al., 2013b). Evaluations on the IWSLT German→English, WMT German→English and BOLT Chinese→English tasks show that JTR models improve state-of-the-art phrase-based systems by up to 2.2 BLEU.

1 Introduction

Standard phrase-based machine translation (Och et al., 1999; Zens et al., 2002; Koehn et al., 2003) uses relative frequencies of phrase pairs to estimate a translation model. The phrase table is extracted from a bilingual text aligned on the word level, using e.g. GIZA++ (Och and Ney, 2003). Although the phrase pairs capture internal dependencies between the source and target phrases aligned to each other, they fail to model dependencies that extend beyond phrase boundaries. Phrase-based decoding involves concatenating target phrases. The burden of ensuring that the result is linguistically consistent falls on the language model (LM).

This work proposes word-based translation models that are potentially capable of capturing long-range dependencies. We do this in two steps: First, given bilingual sentence pairs and the associated word alignments, we convert the information into uniquely defined linear sequences. These sequenecs encode both word reordering and translation information. Thus, they are referred to as *joint translation and reordering* (JTR) sequences. Second, we train an n-gram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the resulting JTR sequences. This yields a model that fuses interdepending reordering and translation dependencies into a single framework.

Although JTR n-gram models are closely related to the operation sequence model (OSM) (Durrani et al., 2013b), there are three main differences. To begin with, the OSM employs minimal translation units (MTUs), which are essentially atomic phrases. As the MTUs are extracted sentence-wise, a word can potentially appear in multiple MTUs. In order to avoid overlapping translation units, we define the JTR sequences on the level of words. Consequently, JTR sequences have smaller vocabulary sizes than OSM sequences and lead to models with less sparsity. Moreover, we argue that JTR sequences offer a simpler reordering approach than operation sequences, as they handle reorderings without the need to predict gaps. Finally, when used as an additional model in the log-linear framework of phrase-based decoding, an n-gram model trained on JTR sequences introduces only one single feature to be tuned, whereas the OSM additionally uses 4 supportive features (Durrani et al., 2013b). Experimental results confirm that this simplification does not make JTR models less expressive, as their performance is on par with the OSM.

Due to data sparsity, increasing the n-gram order of count-based models beyond a certain point becomes useless. To address this, we resort to neu-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1413–1423,
Lisbon, Portugal, 17-21 September 2015.

ral networks (NNs), as they have been successfully applied to machine translation recently (Sundermeyer et al., 2014; Devlin et al., 2014). They are able to score any word combination without requiring additional smoothing techniques. We experiment with feed-forward and recurrent translation networks, benefiting from their smoothing capabilities. To this end, we split the linear sequence into two sequences for the neural translation models to operate on. This is possible due to the simplicity of the JTR sequence. We show that the count and NN models perform well on their own, and that combining them yields even better results.

In this work, we apply n-gram models with modified Kneser-Ney smoothing during phrase-based decoding and neural JTR models in rescoring. However, using a phrase-based system is not required by the model, but only the initial step to demonstrate the strength of JTR models, which can be applied independently of the underlying decoding framework. While the focus of this work is on the development and comparison of the models, the long-term goal is to decode using JTR models without the limitations introduced by phrases, in order to exploit the full potential of JTR models. The JTR models are estimated on word alignments, which we obtain using GIZA++ in this paper. The future aim is to also generate improved word alignments by a joint optimization of both the alignments and the models, similar to the training of IBM models (Brown et al., 1990; Brown et al., 1993). In the long run, we intend to achieve a consistency between decoding and training using the introduced JTR models.

2 Previous Work

In order to address the downsides of the phrase translation model, various approaches have been taken. Mariño et al. (2006) proposed a bilingual language model (BILM) that operates on bilingual n-grams, with an own n-gram decoder requiring monotone alignments. The lexical reordering model introduced in (Tillmann, 2004) was integrated into phrase-based decoding. Crego and Yvon (2010) adapted the approach to BILMs. The bilingual n-grams are further advanced in (Niehues et al., 2011), where they operate on non-monotone alignments within a phrase-based translation framework. Compared to our JTR models, their BILMs treat jointly aligned source words as minimal translation units, ignore unaligned source words and do not include reordering information.

Durrani et al. (2011) developed the OSM which combined dependencies on bilingual word pairs and reordering information into a single framework. It used an own decoder that was based on n-grams of MTUs and predicted single translation or reordering operations. This was further advanced in (Durrani et al., 2013a) by a decoder that was capable of predicting whole sequences of MTUs, similar to a phrase-based decoder. In (Durrani et al., 2013b), a slightly enhanced version of OSM was integrated into the log-linear framework of the Moses system (Koehn et al., 2007). Both the BILM (Stewart et al., 2014) and the OSM (Durrani et al., 2014) can be smoothed using word classes.

Guta et al. (2015) introduced the extended translation model (ETM), which operates on the word level and augments the IBM models by an additional bilingual word pair and a reordering operation. It is implemented into the log-linear framework of a phrase-based decoder and shown to be competitive with a 7-gram OSM.

The JTR n-gram models proposed within this work can be seen as an extension of the ETM. Nevertheless, JTR models utilize linear sequences of dependencies and combine the translation of bilingual word pairs and reoderings into a single model. The ETM, however, features separate models for the translation of individual words and reorderings and provides an explicit treatment of multiple alignments. As they operate on linear sequences, JTR count models can be implemented using existing toolkits for n-gram language models, e.g. the KenLM toolkit (Heafield et al., 2013).

An HMM approach for word-to-phrase alignments was presented in (Deng and Byrne, 2005), showing performance similar to IBM Model 4 on the task of bitext alignment. Feng et al. (2013) propose several models which rely only on the information provided by the source side and predict reorderings. Contrastingly, JTR models incorporate target information as well and predict both translations and reorderings jointly in a single framework.

Zhang et al. (2013) explore different Markov chain orderings for an n-gram model on MTUs in rescoring. Feng and Cohn (2013) present another generative word-based Markov chain translation model which exploits a hierarchical Pitman-Yor process for smoothing, but it is only applied to induce word alignments. Their follow-up work (Feng et al., 2014) introduces a Markov-model on

MTUs, similar to the OSM described above.

Recently, neural machine translation has emerged as an alternative to phrase-based decoding, where NNs are used as standalone models to decode source input. In (Sutskever et al., 2014), a recurrent NN was used to encode a source sequence, and output a target sentence once the source sentence was fully encoded in the network. The network did not have any explicit treatment of alignments. Bahdanau et al. (2015) introduced soft alignments as part of the network architecture. In this work, we make use of hard alignments instead, where we encode the alignments in the source and target sequences, requiring no modifications of existing feed-forward and recurrent NN architectures. Our feed-forward models are based on the architectures proposed in (Devlin et al., 2014), while the recurrent models are based on (Sundermeyer et al., 2014). Further recent research on applying NN models for extended context was carried out in (Le et al., 2012; Auli et al., 2013; Hu et al., 2014). All of these works focus on lexical context and ignore the reordering aspect covered in our work.

3 JTR Sequences

The core idea of this work is the interpretation of a bilingual sentence pair and its word alignment as a linear sequence of K joint translation and reordering (JTR) tokens g_1^K. Formally, the sequence $g_1^K(f_1^J, e_1^I, b_1^I)$ is a uniquely defined interpretation of a given source sentence f_1^J, its translation e_1^I and the inverted alignment b_1^I, where b_i denotes the ordered sequence of source positions j aligned to target position i. We drop the explicit mention of (f_1^J, e_1^I, b_1^I) to allow for a better readability. Each JTR token is either an aligned bilingual word pair $\langle f, e \rangle$ or a reordering class $\Delta_{j'j}$.

Unaligned words on the source and target side are processed as if they were aligned to the empty word ε. Hence, an unaligned source word f generates the token $\langle f, \varepsilon \rangle$, and an unaligned target word e the token $\langle \varepsilon, e \rangle$.

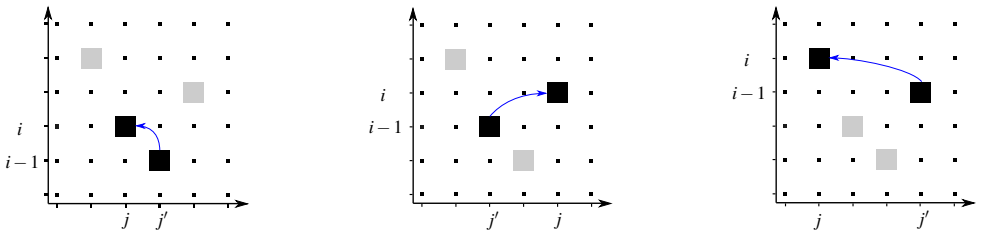
Each word of the source and target sentences is to appear in the corresponding JTR sequence exactly once. For multiply-aligned target words e, the first source word f that is aligned to e generates the token $\langle f, e \rangle$. All other source words f', that are also aligned to e, are processed as if they were aligned to the artificial word σ. Thus, each of these f' generates a token $\langle f', \sigma \rangle$. The same approach is applied to multiply-aligned source words. Similar to Feng and Cohn (2013), we classify the reordered source positions j' and j by $\Delta_{j'j}$:

$$\Delta_{j'j} = \begin{cases} \text{step backward } (\leftarrow), & j = j' - 1 \\ \text{jump forward } (\curvearrowright), & j > j' + 1 \\ \text{jump backward } (\curvearrowleft), & j < j' - 1. \end{cases}$$

The reordering classes are illustrated in Figure 1.

Algorithm 1 JTR Conversion Algorithm

```
1:  procedure JTRCONVERSION(f_1^J, e_1^I, b_1^I)
2:      g_1^K ← ∅
3:      // last translated source position j'
4:      j' ← 0
5:      for i ← 1 to I do
6:          if e_i is unaligned then
7:              // align e_i to the empty word ε
8:              APPEND(g_1^K, ⟨ε, e_i⟩)
9:              continue
10:         // e_i is aligned to at least one source word
11:         j ← first source position in b_i
12:         if j = j' then
13:             // e_i is aligned to the same f_j as e_{i-1}
14:             APPEND(g_1^K, ⟨σ, e_i⟩)
15:             continue
16:         if j ≠ j' + 1 then
17:             // alignment step is non-monotone
18:             REORDERINGS(f_1^J, b_1^I, g_1^K, j', j)
19:         // 1-to-1 translation: f_j is aligned to e_i
20:         APPEND(g_1^K, ⟨f_j, e_i⟩)
21:         j' ← j
22:         // generate all other f_j that are also
23:         // aligned to the current target word e_i
24:         for all remaining j in b_i do
25:             APPEND(g_1^K, ⟨f_j, σ⟩)
26:             j' ← j
27:     // check last alignment step at sentence end
28:     if j' ≠ J then
29:         // last alignment step is non-monotone
30:         REORDERINGS(f_1^J, b_1^I, g_1^K, j', J + 1)
31:     return g_1^K
32:
33: // called when a reordering class is appended
34: procedure REORDERINGS(f_1^J, b_1^I, g_1^K, j', j)
35:     // check if the predecessor is unaligned
36:     if f_{j-1} is unaligned then
37:         // get unaligned predecessors
38:         f_{j_0}^{j-1} ← unaligned predecessors of f_j
39:         // check if the alignment step to the first
40:         // unaligned predecessor is monotone
41:         if j_0 ≠ j' + 1 then
42:             // non-monotone: add reordering class
43:             APPEND(g_1^K, Δ_{j',j_0})
44:         // translate unaligned predecessors by ε
45:         for f ← f_{j_0} to f_{j-1} do
46:             APPEND(g_1^K, ⟨f, ε⟩)
47:     else
48:         // non-monotone: add reordering class
49:         APPEND(g_1^K, Δ_{j',j})
```



(a) step backward (←) (b) jump forward (↷) (c) jump backward (↶)

Figure 1: Overview of the different reordering classes in JTR sequences.

3.1 Sequence Conversion

Algorithm 1 presents the formal conversion of a bilingual sentence pair and its alignment into the corresponding JTR sequence g_1^K. At first, g_1^K is initialized by an empty sequence (line 2). For each target position $i = 1, \ldots, I$ it is extended by at least one token. During the generation process, we store the last visited source position j' (line 4). If a target word e_i is

- unaligned, we align it to the empty word ε and append $\langle \varepsilon, e_i \rangle$ to the current g_1^K (line 8),
- if it is aligned to the same f_j as e_{i-1}, we only add $\langle \sigma, e_i \rangle$ (line 14),
- otherwise we append $\langle f_j, e_i \rangle$ (line 20) and
- in case there are more source words aligned to e_i, we additionally append $\langle f_j, \sigma \rangle$ for each of these (line 24).

Before a token $\langle f_j, e_i \rangle$ is generated, we have to check whether the alignment step from j' to j is monotone (line 16). In case it is not, we have to deal with reorderings (line 34). We define that a token $\langle f_{j-1}, \varepsilon \rangle$ is to be generated right before the generation of the token containing f_j. Thus, if f_{j-1} is not aligned, we first determine the contiguous sequence of unaligned predecessors $f_{j_0}^{j-1}$ (line 38). Next, if the step from j' to j_0 is not monotone, we add the corresponding reordering class (line 43). Afterwards we append all $\langle f_{j_0}, \varepsilon \rangle$ to $\langle f_{j-1}, \varepsilon \rangle$. If f_{j-1} is aligned, we do not have to process unaligned source words and only append the corresponding reordering class (line 49).

Figure 2 illustrates the generation steps of a JTR sequence, whose result is presented in Table 1. The alignment steps are denoted by the arrows connecting the alignment points. The first dashed alignment point indicates the $\langle \varepsilon, , \rangle$ token that is generated right after the $\langle \texttt{Feld}, \texttt{field} \rangle$ token. The second dashed alignment point indicates the $\langle \texttt{ein}, \varepsilon \rangle$ token, which corresponds to the unaligned source word `ein`. Note, that the $\langle \texttt{ein}, \varepsilon \rangle$ token has to be generated right before $\langle ., . \rangle$ is generated. Therefore, there is no forward jump from $\langle \texttt{Code}, \texttt{code} \rangle$ to $\langle ., . \rangle$, but a monotone step to $\langle \texttt{ein}, \varepsilon \rangle$ followed by $\langle ., . \rangle$.

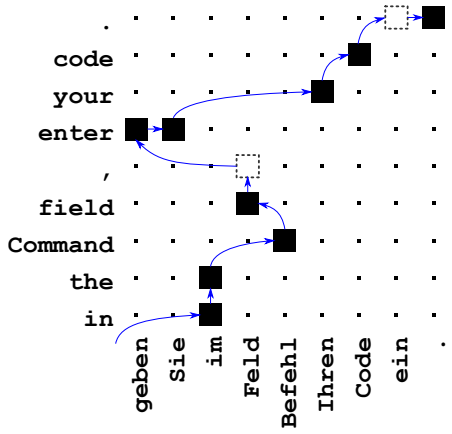


Figure 2: This example illustrates the JTR sequence g_1^K for a German→English sentence pair including the word-to-word alignment.

3.2 Training of Count Models

As the JTR sequence g_1^K is a unique interpretation of a bilingual sentence pair and its alignment, the probability $p(f_1^J, e_1^I, b_1^I)$ can be computed as:

$$p(f_1^J, e_1^I, b_1^I) = p(g_1^K). \tag{1}$$

The probability of g_1^K can be factorized and approximated by an n-gram model.

$$p(g_1^K) = \prod_{k=1}^{K} p(g_k | g_{k-n+1}^{k-1}) \tag{2}$$

Within this work, we first estimate the Viterbi alignment for the bilingual training data using GIZA++ (Och and Ney, 2003). Secondly, the conversion presented in Algorithm 1 is applied to obtain the JTR sequences, on which we estimate an n-gram model with modified Kneser-Ney smoothing as described in (Chen and Goodman, 1998) using the KenLM toolkit[1] (Heafield et al., 2013).

[1] https://kheafield.com/code/kenlm/

k	g_k	s_k	t_k
1	$\curvearrowright$	δ	$\curvearrowright$
2	$\langle$im, in$\rangle$	im	in
3	$\langle\sigma$, the$\rangle$	σ	the
4	$\curvearrowright$	δ	$\curvearrowright$
5	$\langle$Befehl, Command$\rangle$	Befehl	Command
6	$\leftarrow$	δ	$\leftarrow$
7	$\langle$Feld, field$\rangle$	Feld	field
8	$\langle\varepsilon$, ,$\rangle$	ε	,
9	$\curvearrowleft$	δ	$\curvearrowleft$
10	$\langle$geben, enter$\rangle$	geben	enter
11	$\langle$Sie, $\sigma\rangle$	Sie	σ
12	$\curvearrowright$	δ	$\curvearrowright$
13	$\langle$Ihren, your$\rangle$	Ihren	your
14	$\langle$Code, code$\rangle$	Code	code
15	$\langle$ein, $\varepsilon\rangle$	ein	ε
16	$\langle$.,.$\rangle$	.	.

Table 1: The left side of this table presents the JTR tokens g_k corresponding to Figure 2. The right side shows the source and target tokens s_k and t_k obtained from the JTR tokens g_k. They are used for the training of NNs (cf. Section 4).

3.3 Integration into Phrase-based Decoding

Basically, each phrase table entry is annotated with both the word alignment information, which also allows to identify unaligned source words, and the corresponding JTR sequence. The JTR model is added to the log-linear framework as an additional n-gram model. Within the phrase-based decoder, we extend each search state such that it additionally stores the JTR model history.

In comparison to the OSM, the JTR model does not predict gaps. Local reorderings within phrases are handled implicitly. On the other hand, we represent long-range reorderings between phrases by the coverage vector and limit them by reordering constraints.

Phrase-pairs ending with unaligned source words at their right boundary prove to be a problem during decoding. As shown in Subsection 3.1, the conversion from word alignments to JTR sequences assumes that each token corresponding to an unaligned source word is generated immediately before the token corresponding to the closest aligned source position to its right. However, if a phrase ends with an unaligned f_j as its rightmost source word, the generation of the $\langle f_j, \varepsilon\rangle$ token has to be postponed until the next word f_{j+1} is to be translated or, even worse, f_{j+1} has already been translated before.

To address this issue, we constrained the phrase table extraction to discard entries with unaligned source tokens at the right boundary. For IWSLT De→En, this led to a baseline weaker by 0.2 BLEU than the one described in Section 5. In order to have an unconstrained and fair baseline, we thereafter removed this constraint and forced such deletion tokens to be generated at the end of the sequence. Hence, we accept that the JTR model might compute the wrong score in these special cases.

4 Neural Networks

Usually, smoothing techniques are applied to count-based models to handle unseen events. A neural network does not suffer from this, as it is able to score unseen events without additional smoothing techniques. In the following, we will describe how to adapt JTR sequences to be used with feed-forward and recurrent NNs.

The first thing to notice is the vocabulary size, mainly determined by the number of bilingual word pairs, which constituted atomic units in the count-based models. NNs that compute probability values at the output layer evaluate a softmax function that produces normalized scores that sum up to unity. The softmax function is given by:

$$p(e_i|e_1^{i-1}) = \frac{e^{o_{e_i}(e_1^{i-1})}}{\sum_{w=1}^{|V|} e^{o_w(e_1^{i-1})}} \quad (3)$$

where o_{e_i} and o_w are the raw unnormalized output layer values for the words e_i and w, respectively, and $|V|$ is the vocabulary size. The output layer is a function of the context e_1^{i-1}. Computing the denominator is expensive for large vocabularies, as it requires computing the output for all words. Therefore, we split JTR tokens g_k and use individual words as input and output units, such that the NN receives jumps, source and target words as input and outputs target words and jumps. Hence, the resulting neural model is not a LM, but a translation model with different input and output vocabularies. A JTR sequence g_1^K is split into its source and target parts s_1^K and t_1^K. The construction of the JTR source sequence s_1^K proceeds as follows: Whenever a bilingual pair is encountered, the source word is kept and the target word is discarded. In addition, all jump classes are replaced by a special token δ. The JTR target sequence t_1^K is constructed similarly by keeping the target words and dropping source words, and the jump classes are also kept. Table 1 shows the JTR source and target sequences corresponding to JTR sequence of Figure 2.

Due to the design of the JTR sequence, producing the source and target JTR sequences is straightforward. The resulting sequences can then be used with existing NN architectures, without further modifications to the design of the networks. This results in powerful models that require little effort to implement.

4.1 Feed-forward Neural JTR

First, we will apply a feed-forward NN (FFNN) to the JTR sequence. FFNN models resemble count-based models in using a predefined limited context size, but they do not encounter the same smoothing problems. In this work, we use a FFNN similar to that proposed in (Devlin et al., 2014), defined as:

$$p(t_1^K|s_1^K) \approx \prod_{k=1}^{K} p(t_k|t_{k-n}^{k-1}, s_{k-n}^{k}). \quad (4)$$

It scores the JTR target word t_k at position k using the current source word s_k, and the history of n JTR source words. In addition, the n JTR target words preceding t_k are used as context. The FFNN computes the score by looking up the vector embeddings of the source and target context words, concatenating them, then evaluating the rest of the network. We reduce the output layer to a shortlist of the most frequent words, and compute word class probabilities for the remaining words.

4.2 Recurrent Neural JTR

Unlike feed-forward NNs, recurrent NNs (RNNs) enable the use of unbounded context. Following (Sundermeyer et al., 2014), we use bidirectional recurrent NNs (BRNNs) to capture the full JTR source side. The BRNN uses the JTR target side as well as the full JTR source side as context, and it is given by:

$$p(t_1^K|s_1^K) = \prod_{k=1}^{K} p(t_k|t_1^{k-1}, s_1^K) \quad (5)$$

This equation is realized by a network that uses forward and backward recurrent layers to capture the complete source sentence. By a forward layer we imply a recurrent hidden layer that processes a given sequence from left to right, while a backward layer does the processing backwards, from right to left. The source sentence is basically split at a given position k, then past and future representations of the sentence are recursively computed by the forward and backward layers, respectively. To include the target side, we provide the forward layer with the target input t_{k-1} as well, that is, we aggregate the embeddings of the input source word s_k and the input target word t_{k-1} before they are fed into the forward layer. Due to recurrency, the forward layer encodes the parts (t_1^{k-1}, s_1^k), and the backward layer encodes s_k^K, and together they encode (t_1^{k-1}, s_1^K), which is used to score the output target word t_k. For the sake of comparison to FFNN and count models, we also experiment with a recurrent model that does not include future source information, this is obtained by replacing the term s_1^K with s_1^k in Eq. 5. It will be referred to as the unidirectional recurrent neural network (URNN) model in the experiments.

Note that the JTR source and target sides include jump information, therefore, the RNN model described above explicitly models reordering. In contrast, the models proposed in (Sundermeyer et al., 2014) do not include any jumps, and hence do not provide an explicit way of including word reordering. In addition, the JTR RNN models do not require the use of IBM-1 lexica to resolve multiply-aligned words. As discussed in Section 3, these cases are resolved by aligning the multiply-aligned word to the first word on the opposite side.

The integration of the NNs into the decoder is not trivial, due to the dependence on the target context. In the case of RNNs, the context is unbounded, which would affect state recombination, and lead to less variety in the beam used to prune the search space. Therefore, the RNN scores are computed using approximations instead (Auli et al., 2013; Alkhouli et al., 2015). In (Alkhouli et al., 2015), it is shown that approximate RNN integration into the phrase-based decoder has a slight advantage over n-best rescoring. Therefore, we apply RNNs in rescoring in this work, and to allow for a direct comparison between FFNNs and RNNs, we apply FFNNs in rescoring as well.

5 Evaluation

We perform experiments on the large-scale IWSLT 2013[2] (Cettolo et al., 2014) German→English, WMT 2015[3] German→English and the DARPA BOLT Chinese→English tasks. The statistics for the bilingual corpora are shown in Table 2. Word alignments are generated with the GIZA++ toolkit

[2] http://www.iwslt2013.org

[3] http://www.statmt.org/wmt15/

	IWSLT		WMT		BOLT	
	German	**English**	**German**	**English**	**Chinese**	**English**
Sentences	4.32M		4.22M		4.08M	
Run. Words	108M	109M	106M	108M	78M	86M
Vocabulary	836K	792K	814K	773K	384K	817K

Table 2: Statistics for the bilingual training data of the IWSLT 2013 German→English, WMT 2015 German→English, and the DARPA BOLT Chinese→English translation tasks.

(Och and Ney, 2003). We use a standard phrase-based translation system (Koehn et al., 2003). The decoding process is implemented as a beam search. All baselines contain phrasal and lexical smoothing models for both directions, word and phrase penalties, a distance-based reordering model, enhanced low frequency features (Chen et al., 2011), a hierarchical reordering model (HRM) (Galley and Manning, 2008), a word class LM (Wuebker et al., 2013) and an n-gram LM. The lexical and phrase translation models of all baseline systems are trained on all provided bilingual data. The log-linear feature weights are tuned with minimum error rate training (MERT) (Och, 2003) on BLEU (Papineni et al., 2001). All systems are evaluated with *MultEval* (Clark et al., 2011). The reported BLEU scores are averaged over three MERT optimization runs.

All LMs, OSMs and count-based JTR models are estimated with the KenLM toolkit (Heafield et al., 2013). The OSM and the count-based JTR model are implemented in the phrasal decoder. NNs are used only in rescoring. The 9-gram FFNNs are trained with two hidden layers. The short lists contain the 10k most frequent words, and all remaining words are clusterd into 1000 word classes. The projecton layer has 17×100 nodes, the first hidden layer 1000 and the second 500. The RNNs have LSTM architectures. The URNN has 2 hidden layers while the BRNN has one forward, one backward and one additional hidden layer. All layers have 200 nodes, while the output layer is class-factored using 2000 classes. For the count-based JTR model and OSM we tuned the n-gram size on the tuning set of each task. For the full data, 7-grams were used for the IWSLT and WMT tasks, and 8-grams for BOLT. When using in-domain data, smaller n-gram sizes were used. All rescoring experiments used 1000-best lists without duplicates.

5.1 Tasks description

The domain of IWSLT consists of lecture-type talks presented at TED conferences which are also available online[4]. All systems are optimized on the `dev2010` corpus, named `dev` here. Some of the OSM and JTR systems are trained on the TED portions of the data containing 138K sentences. To estimate the 4-gram LM, we additionally make use of parts of the Shuffled News, LDC English Gigaword and 10^9-French-English corpora, selected by a cross-entropy difference criterion (Moore and Lewis, 2010). In total, 1.7 billion running words are taken for LM training. The BOLT Chinese→English task is evaluated on the "discussion forum" domain. The 5-gram LM is trained on 2.9 billion running words in total. The in-domain data consists of a subset of 67.8K sentences and we used a set of 1845 sentences for tuning. The evaluation set `test1` contains 1844 and `test2` 1124 sentences. For the WMT task, we used the target side of the bilingual data and all monolingual data to train a pruned 5-gram LM on a total of 4.4 billion running words. We concatenated the `newstest2011` and `newstest2012` corpora for tuning the systems.

5.2 Results

We start with the IWSLT 2013 German→ English task, where we compare between the different JTR and OSM models. The results are shown in Table 3. When comparing the in-domain n-gram JTR model trained using Kneser-Ney smoothing (KN) to OSM, we observe that the n-gram KN JTR model improves the baseline by 1.4 BLEU on both `test` and `eval11`. The OSM model performs similarly, with a slight disadvantage on `eval11`. In comparison, the FFNN of Eq. (4) improves the baseline by 0.7–0.9 BLEU, compared to the slightly better 0.8–1.1 BLEU achieved by the URNN. The difference between the FFNN and the

[4] `http://www.ted.com/`

	data	dev	test	eval11
baseline	full	33.3	30.8	35.7
+OSM	TED	34.5	**32.2**	36.8
+FFNN	TED	34.0	31.7	36.4
+URNN	TED	34.2	31.9	36.5
+BRNN	TED	34.4	32.1	36.8
+KN	TED	**34.6**	**32.2**	**37.1**
+BRNN	TED	**35.0**	**32.8**	**37.7**
+OSM	full	34.1	**31.6**	36.5
+FFNN	full	33.9	31.5	36.0
+KN	full	**34.2**	**31.6**	**36.6**
+KN	TED	34.9	32.4	37.1
+FFNN	TED	35.2	32.7	37.2
+FFNN	full	35.1	32.7	37.2
+BRNN	TED	**35.5**	**33.0**	**37.4**
+BRNN	TED	35.4	**33.0**	37.3

Table 3: Results measured in BLEU for the IWSLT German→English task.

	train data	test1	test2
baseline		18.1	17.0
+OSM	indomain	**18.8**	17.2
+FFNN	indomain	18.6	**17.6**
+BRNN	indomain	18.6	**17.6**
+KN	indomain	**18.8**	17.5
+OSM	full	18.5	17.2
+FFNN	full	18.4	**17.4**
+KN	full	**18.8**	17.3
+KN	indomain	19.0	17.7
+FFNN	full	19.2	18.3
+RNN	indomain	**19.3**	**18.4**

Table 4: Results measured in BLEU for the BOLT Chinese→English task.

URNN is that the latter captures the unbounded source and target history that extends until the beginning of the sentences, giving it an advantage over the FFNN. The performance of the URNN can be improved by including the future part of the source sentence, as described in Eq. (5), resulting in the BRNN model. Next, we explore whether the models are additive. When rescoring the *n*-gram KN JTR output with the BRNN, an additional improvement of 0.6 BLEU is obtained. There are two reasons for this: The BRNN includes the future part of the source input when scoring target words. This information is not used by the KN model. Moreover, the BRNN is able to score word combinations unseen in training, while the KN model uses backing off to score unseen events.

When training the KN, FFNN, and OSM models on the full data, we observe less gains in comparison to in-domain data training. However, combining the KN models trained on in-domain and full data gives additional gains, which suggests that although the in-domain model is more adapted to the task, it still can gain from out-of-domain data. Adding the FFNN on top improves the combination. Note here that the FFNN sees the same information as the KN model, but the difference is that the NN operates on the word level rather than the word-pair level. Second, the FFNN is able to handle unseen sequences by design, without the need for the backing off workaround. The BRNN improves the combination more than the FFNN, as the model captures an unbounded source and target history in addition to an unbounded future source context. Combining the KN, FFNN and BRNN JTR models leads to an overall gain of 2.2 BLEU on both dev and test.

Next, we present the BOLT Chinese→English results, shown in Table 4. Comparing *n*-gram KN JTR and OSM trained on the in-domain data shows they perform equally well on test1, improving the baseline by 0.7 BLEU, with a slight advantage for the JTR model on test2. The feed-forward and the recurrent in-domain networks yield the same results in comparison to each other. Training the OSM and JTR models on the full data yields slightly worse results than in-domain training. However, combining the two types of training improves the results. This is shown when adding the in-domain KN JTR model on top of the model trained on full data, improving it by up to 0.4 BLEU. Rescoring with the feed-forward and the recurrent network improves this even further, supporting the previous observation that the *n*-gram KN JTR and NNs complement each other. The combination of the 4 models yields an overall improvement of 1.2–1.4 BLEU.

Finally, we compare KN JTR and OSM models on the WMT German→English task in Table 5. The two models perform almost similar to each other. The JTR model improves the baseline by up to 0.7 BLEU. Rescoring the KN JTR with the FFNN improves it by up to 0.3 BLEU leading to an overall improvement between 0.5 and 1.0 BLEU.

	newstest		
	2013	2014	2015
baseline	28.1	28.6	29.4
+OSM	28.6	**28.9**	**30.0**
+FFNN	28.7	**28.9**	29.7
+KN	**28.8**	**28.9**	29.9
+FFNN	**29.1**	**29.1**	**30.0**

Table 5: Results measured in BLEU for the WMT German→English task.

5.3 Analysis

To investigate the effect of including jump information in the JTR sequence, we trained a BRNN using jump classes and another excluding them. The BRNNs were used in rescoring. Below, we demonstrate the difference between the systems:

source: wir kommen später noch auf diese Leute zurück .
reference: We'll come back to these people later .
Hypothesis 1:
JTR source: wir kommen δ zurück δ später noch auf diese Leute δ .
JTR target: we come $\curvearrowright$ back $\curvearrowleft$ later σ to these people $\curvearrowright$.
Hypothesis 2:
JTR source: wir kommen später noch auf diese Leute zurück .
JTR target: we come later σ on these guys back .

Note the German verb "zurückkommen", which is split into "kommen" and "zurück". German places "kommen" at the second position and "zurück" towards the end of the sentence. Unlike German, the corresponding English phrase "come back" has the words adjacent to each other. We found that the system including jumps prefers the correct translation of the verb, as shown in Hypothesis 1 above. The system translates "kommen" to "come", jumps forward to "zurück", translates it to "back", then jumps back to continue translating the word "später". In contrast, the system that excludes jump classes is blind to this separation of words. It favors Hypothesis 2 which is a strictly monotone translation of the German sentence. This is also reflected by the BLEU scores, where we found the system including jump classes outperforming the one without by up to 0.8 BLEU.

6 Conclusion

We introduced a method that converts bilingual sentence pairs and their word alignments into joint translation and reordering (JTR) sequences. They combine interdepending lexical and alignment dependencies into a single framework. A main advantage of JTR sequences is that a variety of models can be trained on them. Here, we have estimated n-gram models with modified Kneser-Ney smoothing, FFNN and RNN architectures on JTR sequences.

We compared our count-based JTR model to the OSM, both used in phrase-based decoding, and showed that the JTR model performed at least as good as OSM, with a slight advantage for JTR. In comparison to the OSM, the JTR model operates on words, leading to a smaller vocabulary size. Moreover, it utilizes simpler reordering structures without gaps and only requires one log-linear feature to be tuned, whereas the OSM needs 5. Due to the flexibility of JTR sequences, we can apply them also to FFNNs and RNNs. Utilizing two count models and applying both networks in rescoring gains the overall highest improvement over the phrase-based system by up to 2.2 BLEU, on the German→English IWSLT task. The combination outperforms OSM by up to 1.2 BLEU on the BOLT Chinese→English tasks.

The JTR models are not dependent on the phrase-based framework, and one of the long-term goals is to perform standalone decoding with the JTR models independently of phrase-based systems. Without the limitations introduced by phrases, we believe that JTR models could perform even better. In addition, we aim to use JTR models to obtain the alignment, which would then be used to train the JTR models in an iterative manner, achieving consistency and hoping for improved models.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 645452 (QT21). This material is partially based upon work supported by the DARPA BOLT project under Contract No. HR0011- 12-C-0015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Tamer Alkhouli, Felix Rietig, and Hermann Ney. 2015. Investigations on phrase-based decoding with recurrent neural network language and translation models. In *Proceedings of the EMNLP 2015 Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September. to appear.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation Modeling with Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, USA, October.

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.

Peter F. Brown, John Cocke, Stephan A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Rossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, June.

Peter F. Brown, Stephan A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, June.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *International Workshop on Spoken Language Translation*, pages 2–11, Lake Tahoe, CA, USA, December.

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

Boxing Chen, Roland Kuhn, George Foster, and Howard Johnson. 2011. Unpacking and transforming feature functions: New ways to smooth phrase tables. In *MT Summit XIII*, pages 269–275, Xiamen, China, September.

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:shortpapers*, pages 176–181, Portland, Oregon, June.

Josep Maria Crego and François Yvon. 2010. Improving reordering with linguistically informed bilingual n-grams. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010: Posters)*, pages 197–205, Beijing, China.

Yonggang Deng and William Byrne. 2005. Hmm word and phrase alignment for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 169–176, Vancouver, British Columbia, Canada, October.

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.

Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1045–1054, Portland, Oregon, USA, June.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013a. Model with minimal translation units, but decode with phrases. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Atlanta, Georgia, June.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013b. Can markov models over minimal translation units help phrase-based smt? In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 399–405, Sofia, Bulgaria, August.

Nadir Durrani, Philipp Koehn, Helmut Schmid, and Alexander Fraser. 2014. Investigating the usefulness of generalized word representations in smt. In *COLING*, Dublin, Ireland, August.

Yang Feng and Trevor Cohn. 2013. A markov model of machine translation using non-parametric bayesian inference. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 333–342, Sofia, Bulgaria, August.

Minwei Feng, Jan-Thorsten Peter, and Hermann Ney. 2013. Advancements in reordering models for statistical machine translation. In *Annual Meeting of the Assoc. for Computational Linguistics*, pages 322–332, Sofia, Bulgaria, August.

Yang Feng, Trevor Cohn, and Xinkai Du. 2014. Factored markov translation with robust modeling. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 151–159, Ann Arbor, Michigan, 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, EMNLP '08, pages 848–856, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andreas Guta, Joern Wuebker, Miguel Graça, Yunsu Kim, and Hermann Ney. 2015. Extended translation models in phrase-based decoding. In *Proceedings of the EMNLP 2015 Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September. to appear.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

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.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, pages 127–133, Edmonton, Alberta.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantine, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. pages 177–180, Prague, Czech Republic, 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.

José B Mariño, Rafael E Banchs, Josep M Crego, Adrià de Gispert, Patrik Lambert, José A R Fonollosa, and Marta R Costa-jussà. 2006. N-gram-based Machine Translation. *Comput. Linguist.*, 32(4):527–549, December.

R.C. Moore and W. Lewis. 2010. Intelligent Selection of Language Model Training Data. In *ACL (Short Papers)*, pages 220–224, Uppsala, Sweden, July.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel, 2011. *Proceedings of the Sixth Workshop on Statistical Machine Translation*, chapter Wider Context by Using Bilingual Language Models in Machine Translation, pages 198–206.

Franz J. Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Franz J. Och, Christoph Tillmann, and Hermann Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proc. Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, University of Maryland, College Park, MD, June.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. IBM Research Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, September.

Darelene Stewart, Roland Kuhn, Eric Joanis, and George Foster. 2014. Coarse split and lump bilingual languagemodels for richer source information in smt. In *AMTA*, Vancouver, BC, Canada, October.

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

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.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, Stroudsburg, PA, USA.

Joern Wuebker, Stephan Peitz, Felix Rietig, and Hermann Ney. 2013. Improving statistical machine translation with word class models. In *Conference on Empirical Methods in Natural Language Processing*, pages 1377–1381, Seattle, USA, October.

Richard Zens, Franz Josef Och, and Hermann Ney. 2002. Phrase-Based Statistical Machine Translation. In *25th German Conf. on Artificial Intelligence (KI2002)*, pages 18–32, Aachen, Germany, September.

Hui Zhang, Kristina Toutanova, Chris Quirk, and Jianfeng Gao. 2013. Beyond left-to-right: Multiple decomposition structures for smt. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21, Atlanta, Georgia, June.