Coverage Embedding Models for Neural Machine Translation

Haitao Mi Baskaran Sankaran Zhiguo Wang Abe Ittycheriah[*]
T.J. Watson Research Center
IBM
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{hmi, bsankara, zhigwang, abei}@us.ibm.com

Abstract

In this paper, we enhance the attention-based neural machine translation (NMT) by adding explicit coverage embedding models to alleviate issues of repeating and dropping translations in NMT. For each source word, our model starts with a *full* coverage embedding vector to track the *coverage* status, and then keeps updating it with neural networks as the translation goes. Experiments on the large-scale Chinese-to-English task show that our enhanced model improves the translation quality significantly on various test sets over the strong large vocabulary NMT system.

1 Introduction

Neural machine translation (NMT) has gained popularity in recent years (e.g. (Bahdanau et al., 2014; Jean et al., 2015; Luong et al., 2015; Mi et al., 2016b; Li et al., 2016)), especially for the attention-based models of Bahdanau et al. (2014). The attention at each time step shows which source word the model should focus on to predict the next target word. However, the attention in each step only looks at the previous hidden state and the previous target word, there is no history or coverage information typically for each source word. As a result, this kind of model suffers from issues of repeating or dropping translations.

The traditional statistical machine translation (SMT) systems (e.g. (Koehn, 2004)) address the above issues by employing a source side "coverage vector" for each sentence to indicate explicitly which words have been translated, which parts have not yet. A coverage vector starts with all zeros, meaning no word has been translated. If a source word at position j got translated, the coverage vector sets position j as 1, and they won't use this source word in future translation. This mechanism avoids the repeating or dropping translation problems.

However, it is not easy to adapt the "coverage vector" to NMT directly, as attentions are soft probabilities, not 0 or 1. And SMT approaches handle one-to-many fertilities by using phrases or hiero rules (predict several words in one step), while NMT systems only predict one word at each step.

In order to alleviate all those issues, we borrow the basic idea of "coverage vector", and introduce a coverage embedding vector for each source word. We keep updating those embedding vectors at each translation step, and use those vectors to track the *coverage* information.

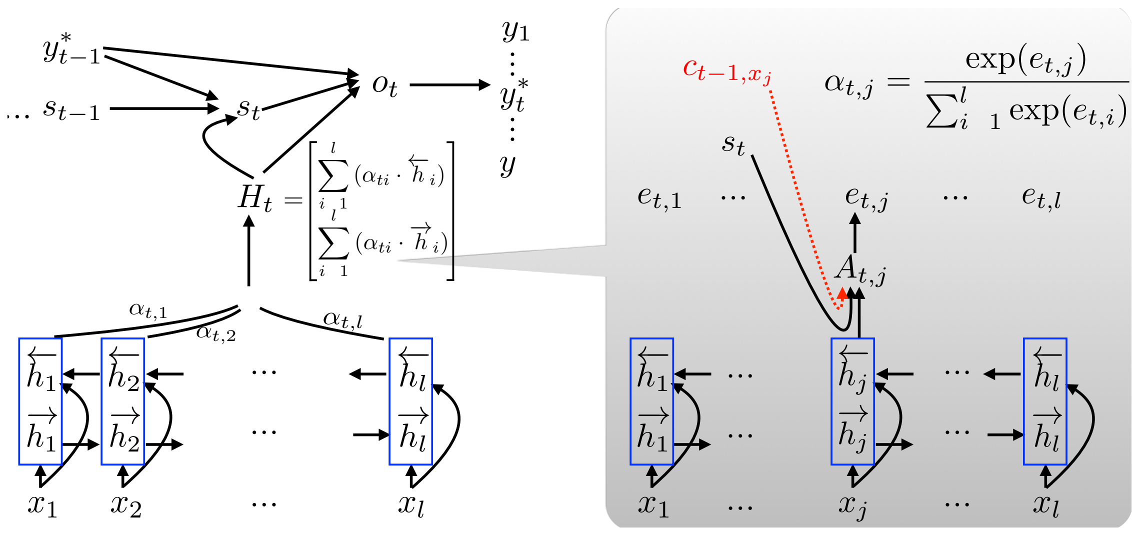
Here is a brief description of our approach. At the beginning of translation, we start from a *full* coverage embedding vector for each source word. This is different from the "coverage vector" in SMT in following two aspects:

- each source word has its own coverage embedding vector, instead of 0 or 1, a scalar, in SMT,
- we start with a *full* embedding vector for each word, instead of 0 in SMT.

After we predict a translation word y_t at time step t, we need to update each coverage embedding vector accordingly based on the attentions in the current step. Our motivation is that if we observe a very high attention over x_i in this step, there is a high chance that x_i and y_t are translation equivalent. So the embedding vector of x_i should come to *empty* (a zero vector) in a one-to-one translation case, or subtract the embedding of y_t for the one-to-many translation case. An *empty* coverage embedding of a word x_i indicates this word is translated, and we can not translate x_i again in future. Empirically, we model this procedure by using neural networks (gated recurrent unit (GRU) (Cho et al., 2014) or direct subtraction).

Large-scale experiments over Chinese-to-English on various test sets show that our method improves the translation quality significantly over the large vocabulary NMT system (Section 5).

[*]Work done while at IBM. To contact Abe, aittycheriah@google.com.



y^*_{t-1}
$\ldots\ s_{t-1}$
s_t
o_t
y_1
y^*_t
y
$H_t = \begin{bmatrix} \sum_{i}^{l} (\alpha_{ti} \cdot \overleftarrow{h}_i) \\ \sum_{i}^{l} (\alpha_{ti} \cdot \overrightarrow{h}_i) \end{bmatrix}$
$\alpha_{t,1}$
$\alpha_{t,2}$
$\alpha_{t,l}$
$\overleftarrow{h}_1$ $\overrightarrow{h}_1$ $\overleftarrow{h}_2$ $\overrightarrow{h}_2$ $\overleftarrow{h}_l$ $\overrightarrow{h}_l$
x_1 x_2 x_l
c_{t-1,x_j}
$\alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_{i}^{l} \exp(e_{t,i})}$
$e_{t,1}$ $e_{t,j}$ $e_{t,l}$
$A_{t,j}$
$\overleftarrow{h}_j$ $\overrightarrow{h}_j$
x_j

Figure 1: The architecture of attention-based NMT.The source sentence is $\mathbf{x} = (x_1, ..., x_l)$ with length l, the translation is $\mathbf{y}^* = (y^*_1, ..., y^*_m)$ with length m. $\overleftarrow{h_i}$ and $\overrightarrow{h_i}$ are bi-directional encoder states. $\alpha_{t,j}$ is the attention probability at time t, position j. H_t is the weighted sum of encoding states. s_t is a hidden state. o_t is an output state. Another one layer neural network projects o_t to the target output vocabulary, and conducts softmax to predict the probability distribution over the output vocabulary. The attention model (in right gray box) is a two layer feedforward neural network, $A_{t,j}$ is an intermediate state, then another layer converts it into a real number $e_{t,j}$, the final attention probability at position j is $\alpha_{t,j}$. We plug coverage embedding models into NMT model by adding an input c_{t-1,x_j} to $A_{t,j}$ (the red dotted line).

2 Neural Machine Translation

As shown in Figure 1, attention-based neural machine translation (Bahdanau et al., 2014) is an encoder-decoder network. the encoder employs a bi-directional recurrent neural network to encode the source sentence $\mathbf{x} = (x_1, ..., x_l)$, where l is the sentence length, into a sequence of hidden states $\mathbf{h} = (h_1, ..., h_l)$, each h_i is a concatenation of a left-to-right $\overrightarrow{h_i}$ and a right-to-left $\overleftarrow{h_i}$,

$$h_i = \begin{bmatrix} \overleftarrow{h}_i \\ \overrightarrow{h}_i \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix},$$

where $\overleftarrow{f}$ and $\overrightarrow{f}$ are two GRUs.

Given the encoded $\mathbf{h}$, the decoder predicts the target translation by maximizing the conditional log-probability of the correct translation $\mathbf{y}^* = (y^*_1, ...y^*_m)$, where m is the sentence length. At each time t, the probability of each word y_t from a target vocabulary V_y is:

$$p(y_t|\mathbf{h}, y^*_{t-1}..y^*_1) = g(s_t, y^*_{t-1}), \qquad (1)$$

where g is a two layer feed-forward neural network (o_t is a intermediate state) over the embedding of the previous word y^*_{t-1}, and the hidden state s_t. The s_t is computed as:

$$s_t = q(s_{t-1}, y^*_{t-1}, H_t) \qquad (2)$$

$$H_t = \begin{bmatrix} \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overleftarrow{h}_i) \\ \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overrightarrow{h}_i) \end{bmatrix}, \qquad (3)$$

where q is a GRU, H_t is a weighted sum of $\mathbf{h}$, the weights, α, are computed with a two layer feed-forward neural network r:

$$\alpha_{t,i} = \frac{\exp\{r(s_{t-1}, h_i, y^*_{t-1})\}}{\sum_{k=1}^{l} \exp\{r(s_{t-1}, h_k, y^*_{t-1})\}} \qquad (4)$$

3 Coverage Embedding Models

Our basic idea is to introduce a coverage embedding for each source word, and keep updating this embedding at each time step. Thus, the coverage embedding for a sentence is a matrix, instead of a vector in SMT. As different words have different fertilities (one-to-one, one-to-many, or one-to-zero), similar to word embeddings, each source word has its own coverage embedding vector. For simplicity, the number of coverage embedding vectors is the same as the source word vocabulary size.

At the beginning of our translation, our coverage embedding matrix $(c_{0,x_1}, c_{0,x_2}, ...c_{0,x_l})$ is initialized with the coverage embedding vectors of all the source words.

Then we update them with neural networks (a GRU (Section 3.1.1) or a subtraction (Section 3.1.2))

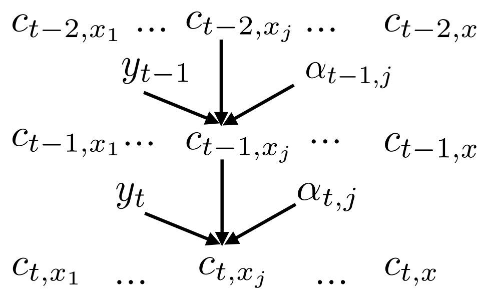

c_{t-2,x_1} ⋯ c_{t-2,x_j} ⋯ $c_{t-2,x}$
y_{t-1}
$\alpha_{t-1,j}$
c_{t-1,x_1} ⋯ c_{t-1,x_j} ⋯ $c_{t-1,x}$
y_t
$\alpha_{t,j}$
c_{t,x_1} … c_{t,x_j} … $c_{t,x}$

Figure 2: The coverage embedding model with a GRU at time step $t-1$ and t. $c_{0,1}$ to $c_{0,l}$ are initialized with the word coverage embedding matrix

until we translation all the source words.

In the middle of translation, some coverage embeddings should be close to *zero*, which indicate those words are covered or translated, and can not be translated in future steps. Thus, in the end of translation, the embedding matrix should be close to zero, which means all the words are covered.

In the following part, we first show two updating methods, then we list the NMT objective that takes into account the embedding models.

3.1 Updating Methods

3.1.1 Updating with a GRU

Figure 2 shows the updating method with a GRU. Then, at time step t, we feed y_t and $\alpha_{t,j}$ to the coverage model (shown in Figure 2),

$$\begin{aligned}
z_{t,j} &= \sigma(W^{zy} y_t + W^{z\alpha} \alpha_{t,j} + U^z c_{t-1,x_j}) \\
r_{t,j} &= \sigma(W^{ry} y_t + W^{r\alpha} \alpha_{t,j} + U^r c_{t-1,x_j}) \\
\tilde{c}_{t,x_j} &= \tanh(W y_t + W^{\alpha} \alpha_{t,j} + r_{t,j} \circ U c_{t-1,x_j}) \\
c_{t,x_j} &= z_{t,j} \circ c_{t-1,x_j} + (1 - z_{t,j}) \circ \tilde{c}_{t,x_j},
\end{aligned}$$

where, z_t is the update gate, r_t is the reset gate, $\tilde{c}_t$ is the new memory content, and c_t is the final memory. The matrix W^{zy}, $W^{z\alpha}$, U^z, W^{ry}, $W^{r\alpha}$, U^r, W^y, W^α and U are shared across different position j. $\circ$ is a pointwise operation.

3.1.2 Updating as Subtraction

Another updating method is to subtract the embedding of y_t directly from the coverage embedding c_{t,x_j} with a weight $\alpha_{t,j}$ as

$$c_{t,x_j} = c_{t-1,x_j} - \alpha_{t,j} \circ (W^{y \to c} y_t), \tag{5}$$

where $W^{y\to c}$ is a matrix that coverts word embedding of y_t to the same size of our coverage embedding vector c.

3.2 Objectives

We integrate our coverage embedding models into the attention NMT (Bahdanau et al., 2014) by adding c_{t-1,x_j} to the first layer of the attention model (shown in the red dotted line in Figure 1).

Hopefully, if y_t is partial translation of x_j with a probability $\alpha_{t,j}$, we only remove partial information of c_{t-1,x_j}. In this way, we enable coverage embedding c_{0,x_j} to encode fertility information of x_j.

As we have mentioned, in the end of translation, we want all the coverage embedding vectors to be close to *zero*. So we also minimize the absolute values of embedding matrixes as

$$\begin{aligned}
\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \Bigg\{ & \sum_{t=1}^{m} \log p(y_t^{*n} | \mathbf{x}^n, y_{t-1}^{*n}..y_1^{*n}) \\
& - \lambda \sum_{i=1}^{l} ||c_{m,x_i}|| \Bigg\},
\end{aligned} \tag{6}$$

where λ is the coefficient of our coverage model.

As suggested by Mi et al. (2016a), we can also use some supervised alignments in our training. Then, we know exactly when each c_{t,x_j} should become close to *zero* after step t. Thus, we redefine Equation 6 as:

$$\begin{aligned}
\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \Bigg\{ & \sum_{t=1}^{m} \log p(y_t^{*n} | \mathbf{x}^n, y_{t-1}^{*n}..y_1^{*n}) \\
& - \lambda \sum_{i=1}^{l} (\sum_{j=a_{x_i}}^{m} ||c_{j,x_i}||) \Bigg\},
\end{aligned} \tag{7}$$

where a_{x_i} is the maximum index on the target sentence x_i can be aligned to.

4 Related Work

There are several parallel and independent related work (Tu et al., 2016; Feng et al., 2016; Cohn et al., 2016). Tu et al. (2016) is the most relevant one. In their paper, they also employ a GRU to model the coverage vector. One main difference is that our model introduces a specific coverage embedding vector for each source word, in contrast, their work initializes the word coverage vector with a scalar with a uniform distribution. Another difference lays in the fertility part, Tu et al. (2016) add an accumulate operation and a fertility function to simulate the process of one-to-many alignments. In our approach, we add fertility information directly to coverage embeddings, as each source word has its own embedding. The last difference is that our baseline system (Mi et al., 2016b) is an extension of the large vocabulary NMT of Jean et al. (2015) with candidate list decoding and UNK replacement, a much stronger baseline system.

Cohn et al. (2016) augment the attention model with well-known features in traditional SMT, including positional bias, Markov conditioning, fertility and agreement over translation directions. This work is orthogonal to our work.

5 Experiments

5.1 Data Preparation

We run our experiments on Chinese to English task. We train our machine translation systems on two training sets. The first training corpus consists of approximately 5 million sentences available within the DARPA BOLT Chinese-English task. The second training corpus adds HK Law, HK Hansard and UN data, the total number of training sentence pairs is 11 million. The Chinese text is segmented with a segmenter trained on CTB data using conditional random fields (CRF).

Our development set is the concatenation of several tuning sets (GALE Dev, P1R6 Dev, and Dev 12) released under the DARPA GALE program. The development set is 4491 sentences in total. Our test sets are NIST MT06, MT08 news, and MT08 web.

For all NMT systems, the full vocabulary sizes for thr two training sets are $300k$ and $500k$ respectively. The coverage embedding vector size is 100. In the training procedure, we use AdaDelta (Zeiler, 2012) to update model parameters with a mini-batch size 80. Following Mi et al. (2016b), the output vocabulary for each mini-batch or sentence is a sub-set of the full vocabulary. For each source sentence, the sentence-level target vocabularies are union of top $2k$ most frequent target words and the top 10 candidates of the word-to-word/phrase translation tables learned from 'fast_align' (Dyer et al., 2013). The maximum length of a source phrase is 4. In the training time, we add the reference in order to make the translation reachable.

Following Jean et al. (2015), We dump the alignments, attentions, for each sentence, and replace UNKs with the word-to-word translation model or the aligned source word.

Our traditional SMT system is a hybrid syntax-based tree-to-string model (Zhao and Al-onaizan, 2008), a simplified version of Liu et al. (2009) and Cmejrek et al. (2013). We parse the Chinese side with Berkeley parser, and align the bilingual sentences with GIZA++. Then we extract Hiero and tree-to-string rules on the training set. Our two 5-gram language models are trained on the English side of the parallel corpus, and on monolingual corpora (around 10 billion words from Gigaword (LDC2011T07)), respectively. As suggestion by Zhang (2016), NMT systems can achieve better results with the help of those monolingual corpora. We tune our system with PRO (Hopkins and May, 2011) to minimize (TER- BLEU)/2 on the development set.

5.2 Translation Results

Table 1 shows the results of all systems on 5 million training set. The traditional syntax-based system achieves 9.45, 12.90, and 17.72 on MT06, MT08 News, and MT08 Web sets respectively, and 13.36 on average in terms of (TER- BLEU)/2. The large-vocabulary NMT (LVNMT), our baseline, achieves an average (TER- BLEU)/2 score of 15.74, which is about 2 points worse than the hybrid system.

We test four different settings for our coverage embedding models:

- $\mathbf{U}_{GRU}$: updating with a GRU;
- $\mathbf{U}_{Sub}$: updating as a subtraction;
- $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$: combination of two methods (do not share coverage embedding vectors);
- **+Obj.**: $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$ plus an additional objective in Equation 6[1].

$\mathbf{U}_{GRU}$ improves the translation quality by 1.3 points on average over LVNMT. And $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$ achieves the best average score of 13.14, which is about 2.6 points better than LVNMT. All the improvements of our coverage embedding models over LVNMT are statistically significant with the sign-test of Collins et al. (2005). We believe that we need to explore more hyper-parameters of **+Obj.** in order to get even better results over $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$.

[1]We use two λs for $\mathbf{U}_{GRU}$ and $\mathbf{U}_{Sub}$ separately, and we test $\lambda_{GRU} = 1 \times 10^{-4}$ and $\lambda_{Sub} = 1 \times 10^{-2}$ in our experiments.

single system		MT06			MT08						avg.
					News			Web			
		BP	BLEU	T-B	BP	BLEU	T-B	BP	BLEU	T-B	T-B
Tree-to-string		0.95	34.93	9.45	0.94	31.12	12.90	0.90	23.45	17.72	13.36
LVNMT		0.96	34.53	12.25	0.93	28.86	17.40	0.97	26.78	17.57	15.74
Ours	$\mathbf{U}_{GRU}$	0.92	35.59	10.71	0.89	30.18	15.33	0.97	27.48	16.67	14.24
	$\mathbf{U}_{Sub}$	0.91	35.90	10.29	0.88	30.49	15.23	0.96	27.63	16.12	13.88
	$\mathbf{U}_{GRU}$+$\mathbf{U}_{Sub}$	0.92	36.60	9.36	0.89	31.86	13.69	0.95	27.12	16.37	13.14
	+Obj.	0.93	36.80	9.78	0.90	31.83	14.20	0.95	28.28	15.73	13.24

Table 1: Single system results in terms of (TER-BLEU)/2 (the lower the better) on 5 million Chinese to English training set. NMT results are on a large vocabulary ($300k$) and with UNK replaced. $\mathbf{U}_{GRU}$: updating with a GRU; $\mathbf{U}_{Sub}$: updating as a subtraction; $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$: combination of two methods (do not share coverage embedding vectors); **+Obj.**: $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$ with an additional objective in Equation 6, we have two λs for $\mathbf{U}_{GRU}$ and $\mathbf{U}_{Sub}$ separately, and we test $\lambda_{GRU} = 1 \times 10^{-4}$ and $\lambda_{Sub} = 1 \times 10^{-2}$.

single system	MT06		MT08				avg.
			News		Web		
	BP	T-B	BP	T-B	BP	T-B	T-B
Tree-to-string	0.90	8.70	0.84	12.65	0.84	17.00	12.78
LVNMT	0.96	9.78	0.94	14.15	0.97	15.89	13.27
$\mathbf{U}_{GRU}$	0.97	8.62	0.95	12.79	0.97	15.34	12.31

Table 2: Single system results in terms of (TER-BLEU)/2 on 11 million set. NMT results are on a large vocabulary ($500k$) and with UNK replaced. Due to the time limitation, we only have the results of $\mathbf{U}_{GRU}$ system.

system		pre.	rec.	F1
MaxEnt		74.86	77.10	75.96
LVNMT		47.88	41.06	44.21
Ours	$\mathbf{U}_{GRU}$	51.11	41.42	45.76
	$\mathbf{U}_{Sub}$	49.07	42.49	45.55
	$\mathbf{U}_{GRU}$+$\mathbf{U}_{Sub}$	49.46	43.83	46.47
	+Obj.	49.78	41.73	45.40

Table 3: Alignment F1 scores of different models.

Table 2 shows the results of 11 million systems, LVNMT achieves an average (TER-BLEU)/2 of 13.27, which is about 2.5 points better than 5 million LVNMT. The result of our $\mathbf{U}_{GRU}$ coverage model gives almost 1 point gain over LVNMT. Those results suggest that the more training data we use, the stronger the baseline system becomes, and the harder to get improvements. In order to get a reasonable or strong NMT system, we have to conduct experiments over a large-scale training set.

5.3 Alignment Results

Table 3 shows the F1 scores on the alignment test set (447 hand aligned sentences). The MaxEnt model is trained on $67k$ hand-aligned data, and achieves an F1 score of 75.96. For NMT systems, we dump alignment matrixes, then, for each target word we only add the highest probability link if it is higher than 0.2. Results show that our best coverage model, $\mathbf{U}_{GRU}$ + $\mathbf{U}_{Sub}$, improves the F1 score by 2.2 points over the sorce of LVNMT.

We also check the repetition statistics of NMT outputs. We simply compute the number of repeated phrases (length longer or equal than 4 words) for each sentence. On MT06 test set, the 5 million LVNMT has 209 repeated phrases, our $\mathbf{U}_{GRU}$ system reduces it significantly to 79, $\mathbf{U}_{GRU}+\mathbf{U}_{Sub}$ and **+Obj.** only have 50 and 47 repeated phrases, respectively. The 11 million LVNMT gets 115 repeated phrases, and $\mathbf{U}_{GRU}$ reduces it further down to 16. Those trends hold across other test sets. Those statistics show that a larger training set or coverage embedding models alleviate the repeating problem in NMT.

6 Conclusion

In this paper, we propose simple, yet effective, coverage embedding models for attention-based NMT. Our model learns a special coverage embedding vector for each source word to start with, and keeps updating those coverage embeddings with neural networks as the translation goes. Experiments on the large-scale Chinese-to-English task show significant improvements over the strong LVNMT system.

Acknowledgment

We thank reviewers for their useful comments.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv e-prints*, September.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Martin Cmejrek, Haitao Mi, and Bowen Zhou. 2013. Flexible and efficient hypergraph interactions for joint hierarchical and forest-to-string decoding. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 545–555, Seattle, Washington, USA, October. Association for Computational Linguistics.

T. Cohn, C. D. V. Hoang, E. Vymolova, K. Yao, C. Dyer, and G. Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. *ArXiv e-prints*, January.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June. Association for Computational Linguistics.

S. Feng, S. Liu, M. Li, and M. Zhou. 2016. Implicit Distortion and Fertility Models for Attention-based Encoder-Decoder NMT Model. *ArXiv e-prints*, January.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

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

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *Proceedings of IJCAI 2016*, pages 2852–2858, New York, NY, USA, July.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009. Joint decoding with multiple translation models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 576–584, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016a. Supervised attentions for neural machine translation. In *Proceedings of EMNLP*, Austin, USA, November.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016b. Vocabulary manipulation for neural machine translation. In *Proceedings of ACL*, Berlin, Germany, August.

Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. 2016. Coverage-based Neural Machine Translation. *ArXiv e-prints*, January.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*.

Jiajun Zhang. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP 2016*, Austin, Texas, USA, November.

Bing Zhao and Yaser Al-onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 572–581, Stroudsburg, PA, USA. Association for Computational Linguistics.