Journal of Artificial Intelligence Research 67 (2020) 581-606 Submitted 02/2017; published 03/2020

Agreement on Target-Bidirectional Recurrent Neural Networks for Sequence-to-Sequence Learning

Lemao Liu LEMAOLIU@GMAIL.COM
Andrew Finch ANDREW.FINCH@GMAIL.COM
Masao Utiyama MASAO.UTIYAMA@NICT.GO.JP
Eiichiro Sumita EIICHIRO.SUMITA@NICT.GO.JP
National Institute of Information & Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan

Abstract

Recurrent neural networks are extremely appealing for sequence-to-sequence learning tasks. Despite their great success, they typically suffer from a shortcoming: they are prone to generate unbalanced targets with good prefixes but bad suffixes, and thus performance suffers when dealing with long sequences. We propose a simple yet effective approach to overcome this shortcoming. Our approach relies on the *agreement* between a pair of target-directional RNNs, which generates more balanced targets. In addition, we develop two efficient approximate search methods for agreement that are empirically shown to be *almost optimal* in terms of either sequence level or non-sequence level metrics. Extensive experiments were performed on three standard sequence-to-sequence transduction tasks: machine transliteration, grapheme-to-phoneme transformation and machine translation. The results show that the proposed approach achieves consistent and substantial improvements, compared to many state-of-the-art systems.

1. Introduction

Recurrent neural networks (RNNs) (Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010), particularly RNNs with Long Short-term Memory (LSTM) (Hochreiter & Schmidhuber, 1997; Graves, 2013) or those with Gated Recurrent Units (Cho, Van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk, & Bengio, 2014), provide a universal and powerful solution for various tasks that have traditionally required carefully designed, task-specific solutions. On classification tasks (Graves & Schmidhuber, 2008; Tai, Socher, & Manning, 2015), they can readily summarize an unbounded context which is difficult for tranditional solutions, and this leads to more reliable prediciton. They have advantages over traditional solutions on more general and challenging tasks such as sequence-to-sequence learning (Sutskever, Vinyals, & Le, 2014), where a series of local but *dependent* predictions are required. RNNs make use of the contextual information for the entire source sequence and also critically are able to exploit the entire sequence of previous predictions. On various sequence-to-sequence transduction tasks, RNNs have been shown to be comparable to the state-of-the-art (Bahdanau, Cho, & Bengio, 2015; Meng, Lu, Tu, Li, & Liu, 2015) or superior (Jean, Cho, Memisevic, & Bengio, 2015; Luong, Sutskever, Le, Vinyals, & Zaremba, 2015).

Despite their sucesses on sequence-to-sequnce learning, RNNs suffer from an important shortcoming, which has been overlooked. When making predictions (in decoding), an LSTM

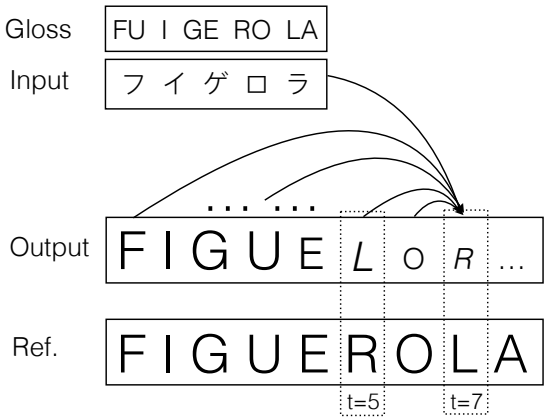


Figure 1: Illustraction of a shortcoming of an LSTM in decoding.

needs to encode the previous local predictions as a part of the contextual information. If some of previous predictions are incorrect, the context for subsequent predictions might include some noise, which will undermine the quality of these predicitons, as shown in Figure 1.

In the figure, larger fonts in the output box indicate greater confidence in the predicted target character. The prediction at $t = 7$ uses a context consisting of the input and all previous predictions. Since at $t = 5$ the prediction is incorrect, i.e. it should be 'R' (in the reference) instead of 'L', it leads to an incorrect prediction at $t = 7$. In this way, an LSTM is more likely to generate an unbalanced sequence deteriorating in quality as the target sequence is generated.

A statistical analysis on the real prediction results from an LSTM was performed in order to motivate the work reported here. The analysis supports our hypothesis: on test examples from real machine transliteration task, we found that for sequences longer than 10 characters, the precision of predictions for the first two characters was higher than 77%, while for the last two it was only about 65% (see Section §5.1.2). Therefore this shortcoming may limit the potential of an RNN, especially for long sequences.

To address the above shortcoming, in this paper, we propose a simple yet efficient approach. Its basic idea relies on the **agreement** between two target-specific directional LSTM models: one generates target sequences from left-to-right as usual, while the other generates from right-to-left. Specifically, we first jointly train both directional LSTM models; and then for testing we search for target sequences which have support from both of the models. In this way, it is expected that the final outputs contain both good prefixes and good suffixes. Since the agreement search problem involves in a pair of models with two opposite generation orders, its exact solution is intractable, and we have therefore developed two approximate alternatives which are simple yet efficient. The proposed approximate search techniques consider only a tiny subset of the entire search space, and thus they might introduce some search errors, which are well-known to undermine the performance of feature-rich linear models in NLP tasks (Collins & Roark, 2004; Huang, Fayong, & Guo, 2012; Liu & Huang, 2014). In this paper, we analyze the search quality according to both model scores and evaluation metrics. In particular, we provide some theoretical basis for the condition

when the search errors undermine the performance according to different types of evaluation metrics. In addition, we empirically evaluate our approximate search approaches under this condition on the real dataset, and the results show that both search approaches are almost optimal in terms of both sequence-level and non-sequence-level evaluation metrics.

This paper makes the following contributions:

- It points out and formally analyzes a shortcoming affecting recurrent neural networks in sequence-to-sequence learning.

- It proposes an *efficient* approach to address the shortcoming, by encouraging the agreement between two directional (left-to-right and right-to-left) models. In addition, two approximate methods are proposed to search for the agreement. This approach is *general* enough to be applied to any deep recurrent neural networks.

- It intensively and systematically explores the problem of search errors from different points of view, and proposes several methods and algorithms to analyze and evaluate the search errors. With these methods, we empirically demonstrate that our approximate search methods perform similarly to the optimal search method according to different evaluation metrics. To the best of our knowledge, it is the first time search errors have been evaluated for recurrent neural networks from multiple views.

- On three standard sequence-to-sequence learning tasks including machine transliteration, grapheme-to-phoneme transduction and machine translation, the proposed approach delivered substantial improvements and consistently outperformed several state-of-the-art systems.

The rest of this paper is organized as follows. Section §2 revisits RNN models for sequence-to-sequence learning and points out one of their shortcomings; Section §3 proposes the agreement model based on target-bidirectional RNNs and its approximate search approaches to address this shortcomings; Section §4 analyzes the optimality of both approximate search approaches; the effectiveness of our model is empirically verified by experiments on real sequence-to-sequence learning tasks in Section §5; and Section §6 presents the related work with respect to our model followed by the conclusion at the last section. Our toolkit is publicly available on https://github.com/lemaoliu/Agtarbidir.

2. Background on Sequence-to-Sequence Learning with RNNs

Suppose $\mathbf{x} = \langle x_1, x_2, \cdots, x_{|\mathbf{x}|} \rangle$ denotes a sequence of characters, its t^{th} character (at time step t) is x_t and its length is $|\mathbf{x}|$. In addition, let $x_{<t} = \langle x_1, x_2, \cdots, x_{t-1} \rangle$ denote a prefix of $\mathbf{x}$, and $\mathbf{f}$ denote a source sequence while $\mathbf{e}$ denote a target sequence. θ denotes the set of model parameters of a recurrent neural network: $\theta^{superscript}$ denotes a component parameter of θ depending on *superscript*, and it is either a bias vector (if *superscript* includes b) or a matrix; $\theta(x_t)$ is a vector representing embedding of x_t, which is either a source character or a target character. Note that in the rest of this paper, the subscript is reserved as the time step in a sequence for easier reading.

2.1 Model Definition

The sequence to sequence learning model maps a source sequence $\mathbf{f}$ into a target $\mathbf{e}$ in a probabilistic way. Generally, this probabilistic model relies on two RNNs under the encode-decode framework, where one RNN is used for encoding the source sequence while the other is used for decoding the target sequence. For easier description, h_t^E denotes the hidden state at time step t for encoding RNN, h_t^D denotes the hidden state for decoding RNN and $\mathbf{h}^E = \left\langle h_1^E, h_2^E, \cdots, h_{|\mathbf{h}^E|}^E \right\rangle$ denotes the sequence of hidden states for encoding RNN.

Formally, it defines a conditional probability over a pair of sequences $\mathbf{f}$ and $\mathbf{e}$ via RNNs as follows:

$$\begin{aligned} P(\mathbf{e} \mid \mathbf{f}; \theta) &= \prod_t P(e_t \mid e_{<t}; \theta) \\ &= \prod_t \mathbf{softmax}\Big(g\big(h_t^D, \mathbf{h}^E, \mathbf{f}\big)\Big)\big[I(\theta, e_t)\big] \end{aligned} \tag{1}$$

where g is a multi-layer peceptron transforming h_t^D, $\mathbf{h}^E$ and $\mathbf{f}$ to a vector with dimension of the target-side vocabulary size, $I(\theta, e_t)$ denotes the index of e_t in the target vocabulary, $vec[I]$ is a real number representing the I_{th} component of vector vec.

In order to further specify the definition of sequence to sequence learning architecture, we introduce two different models in this paper. The main difference between them is whether the attention mechanism is applied or not. Therefore, they are called **A**ttention-**f**ree Recurrent Neural Networks (AfRNN) and **A**ttention-**b**ased **R**ecurrent **N**eural **N**etworks (AbRNN) hereafter.

2.1.1 Attention-Free Recurrent Neural Networks

Sutskever et al. (2014) proposed an attention-free RNN model for sequence-to-sequence learning. Both encoding and decoding use the same type of RNN models but they are represented by different parameters θ^E and θ^D. Suppose $\Re$ denotes a recurrent function, then the hidden state h_t^E and h_t^D are recursively defined as follows:

$$h_t^E = \Re(f_t, h_{t-1}^E; \theta^E)$$
$$h_t^D = \Re(e_{t-1}, h_{t-1}^D; \theta^D)$$

where the base case for encoding is $h_0^E = \mathbf{0}$, and the base cases for decoding are $h_0^D = h_{|\mathbf{e}|}^E$ and e_0, which denotes a special token with embedding $\theta(e_0) = \mathbf{0}$.

Let $h_t = \langle h_t^1, h_t^2 \rangle$ be a hidden unit, $\langle h_t^1, h_t^2 \rangle = \Re(x_t, h_{t-1}, \theta)$ is defined by an LSTM unit (Graves, 2013) via the following functions:

$$\begin{aligned} i_t &= \sigma(\theta^i \theta(x_t) + \theta^{1,i} h_{t-1}^1 + \theta^{2,i} h_{t-1}^2 + \theta^{b,i}) \\ j_t &= \sigma(\theta^j \theta(x_t) + \theta^{1,j} h_{t-1}^1 + \theta^{2,j} h_{t-1}^2 + \theta^{b,j}) \\ h_t^2 &= j_t \odot h_{t-1}^2 + i_t \odot \tanh(\theta^2 \theta(x_t) + \theta^{1,2} h_{t-1}^1 + \theta^{b,2}) \\ o_t &= \sigma(\theta^{x,o} \theta(x_t) + \theta^{1,o} h_{t-1}^1 + \theta^{2,o} h_{t-1}^2 + \theta^{b,o}) \\ h_t^1 &= o_t \odot \tanh(h_t^2) \end{aligned}$$

where $\theta(x_t)$ denotes the embedding of x_t as claimed before, σ denotes the sigmoid function and $\odot$ denotes the element-wise product over a pair of vectors.

2.1.2 Attention-Based Recurrent Neural Networks

Note that an AfRNN encodes the entire source sequence into a single fixed-size vector (i.e. $h^E_{|\mathbf{f}|}$) for calculation of h^D_t in decoding. In contrast, an AbRNN is able to dynamically summarize information from source hidden units for calculation of h^D_t, by using an attention mechanism in decoding. In details, h^D_t is defined as follows:

$$h^D_t = \Re(h^D_{t-1}, e_{t-1}, c_t; \theta) \tag{2}$$

where $c_t = \sum_i \alpha_{t,i} h^E_i$ denotes the context at time step t with its weight $\alpha_{t,i}$ being specified by

$$\alpha_{t,i} = \frac{\exp(\theta^v \tanh(\theta^w h^D_{t-1} + \theta^u h^E_i)))}{\sum_k \exp(\theta^v \tanh(\theta^w h^D_{t-1} + \theta^u h^E_k))},$$

and $\Re$ is defined by the GRU unit (Cho et al., 2014) instead of LSTM unit as in AfRNN according to Bahdanau et al. (2015):

$$\Re(h_{t-1}, f_{t-1}, c_t) = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{3}$$

where $\widetilde{h}_t$ is defined by the following equations:

$$\begin{aligned} \widetilde{h}_t &= \tanh(\theta^{h,1}\theta(f_{t-1}) + \theta^{h,2}[r_t \odot h_{t-1}] + \theta^{h,3} c_t) \\ z_t &= \sigma(\theta^{z,1}\theta(f_{t-1}) + \theta^{z,2} h_{t-1} + \theta^{z,3} c_t) \\ r_t &= \sigma(\theta^{r,1}\theta(f_{t-1}) + \theta^{r,2} h_{t-1} + \theta^{r,3} c_t) \end{aligned} \tag{4}$$

Furthermore, in order to capture abundant information during encoding, Bahdanau et al. (2015) employ a bidirectional RNN to encode the source sequence in both forward and backward directions. In other words, $h^E_t = \langle h^{1,E}_t, h^{2,E}_t \rangle$, where $h^{1,E}_t$ is recursively defined in an ascending manner, while $h^{2,E}_t$ is defined in a descending manner:

$$\begin{aligned} h^{1,E}_t &= \Re(f_t, h^{1,E}_{t-1}; \theta) \\ h^{2,E}_t &= \Re(f_t, h^{2,E}_{t+1}; \theta) \end{aligned} \tag{5}$$

where the initial values $h^{1,E}_0$ and $h^{2,E}_{|\mathbf{f}|+1}$ are $\mathbf{0}$, and $\Re$ denotes a recurrent function similar to that in Eq.(3) with a slight modification to consider c_t as zero in Eq.(3), since it is defined over two variables rather than three.

2.2 Decoding

Given a source sequence $\mathbf{f}$ and parameters θ, decoding can be formulated as follows:

$$\hat{\mathbf{e}}\big(\mathbf{f}; \theta, \Omega(\mathbf{f})\big) = \underset{\mathbf{e} \in \Omega(\mathbf{f})}{\mathbf{argmax}}\, P(\mathbf{e} \mid \mathbf{f}; \theta) \tag{6}$$

where P is given by Equation (1), and $\Omega(\mathbf{f})$ is the set of all possible target sequences $\mathbf{e}$ that can be generated using the target vocabulary. Since the prediciton at time step t (i.e. e_t) is dependent on all the previous predictions, it is NP-hard to optimize the exact solution of Equation (6). Instead, an approximate solution, beam search, which generates the target

Algorithm 1 Beam Search Algorithm

Require: initial state S_0 (an empty state), beam size b
Ensure: S_g
1: FBIN = [], BEAM = $[S_0]$, $S_g = S_0$
2: **while** BEAM **do**
3: OPEN = []
4: **for all** S in BEAM **do**
5: **for all** S' in **get_successor**(S) and not **is_explored**(S') **do**
6: **if is_goal_state**(S') **then**
7: **add**(FBIN,S')
8: $S_g = \mathbf{min}(S', S_g)$
9: **if** |FBIN| = b **then**
10: return S_g
11: **else**
12: **add**(OPEN,S')
13: BEAM = $\mathbf{min\_pop}_b$(OPEN)
14: return S_g

sequence by extending one token each time from *left-to-right*, is widely applied (Sutskever et al., 2014).

Algorithm 1 shows the generic beam search algorithm. In this algorithm, its initial state S_0 is an empty sequence, **get_successor**(S) consists of all possible states extending S with only one token from left-to-right as shown in Figure 1, **is_explored**(S') is true if S' has been explored already, and **is_goal_state**(S) is true if and only if S is terminated with a special token "eos". Finally, the algorithm returns the target state S_g in line 14.

The main part of this algorithm is the loop between line 2 and 13. It enumerates each successor S' of each state S in BEAM in line 5 and checks whether S' is a goal state or not in line 6. If S' is a goal state, S' is added into final bin FBIN and the best state S_g is added in line 7-8; otherwise S' is added into OPEN list. Note that if the size of FBIN exceeds b, then it returns the current S_g in line 9-10. In line 13, it performs pruning by $\mathbf{min\_pop}_b$(OPEN), which maintains a BEAM including at most b states with lower model scores from OPEN.

2.3 Shortcoming of RNN

Despite their successes on various tasks, both AfRNN and AbRNN still suffer from a shortcoming. Suppose at time step t when predicting e_t, there is an incorrect prediciton $e_{t'}$ for t' with $0 \leqslant t' < t$. In other words, the hidden states $h^D_{t''}$ makes use of this incorrect information for each t'' in the range $t' < t'' \leqslant t$; and this can be expected to degrade the quality of all the predictions made using the noisy $h^D_{t''}$. Ideally, at any timestep t'' if the probability of a correct prediction is $p_{t''}$, then will h^D_t contain noisy information with a probability of: $1 - \prod_{0 \leqslant t' < t} p_{t'}$. As t increases the probability of noise in the context increases quickly, and therefore it is more difficult for an RNN to make correct predictions as the sequence length increases. As a result, generic LSTMs cannot maintain the quality of their earlier

predictions in their later predictions, and this is a serious problem especially when the input sequence is long.

3. Agreement on Target-Bidirectional RNNs

As explained in the previous section, although the generic (left-to-right) RNN struggles when predicting suffixes, fortunately, it is very capable at predicting prefixes. On the other hand, a complementary RNN which generates targets from right-to-left, is proficient at predicting suffixes. Inspired by work in the field of word alignment (Liang, Bouchard-Côté, Klein, & Taskar, 2006), we propose an agreement model for sequence-to-sequence learning to overcome the shortcoming described in §2. It encourages the agreement between both target-directional RNN models.

3.1 Agreement Model

Formally, we develop the joint training objective based on the agreement over a pair of target-bidirectional RNNs following Liang et al. (2006):[1]

$$\min_{\overrightarrow{\theta},\overleftarrow{\theta}} \sum_{\langle \mathbf{f},\mathbf{e}\rangle} \Big(\log\big(\overrightarrow{\mathbf{P}}(\mathbf{e} \mid \mathbf{f}; \overrightarrow{\theta})\big) + \log\big(\overleftarrow{\mathbf{P}}(\mathbf{e} \mid \mathbf{f}; \overleftarrow{\theta})\big) \Big) \tag{7}$$

where the example $\langle \mathbf{f}, \mathbf{e} \rangle$ ranges over a given training set. $\overrightarrow{P}$ and $\overleftarrow{P}$ are the left-to-right (l2r) and right-to-left (r2l) RNN models respectively, with definitions similar to Equation (1); $\overrightarrow{\theta}$ and $\overleftarrow{\theta}$ denote their parameters. This model is called an **agreement** model or **bidirectional** model in this paper. To perform the optimization, we employ AdaDelta (Zeiler, 2012), a mini-batch stochastic gradient method. The gradient is calculated using backpropogation through time (Rumelhart, Hinton, & Williams, 1986), where the time is unlimited in our experiments.

Suppose $\overleftrightarrow{\theta} = \langle \overrightarrow{\theta}, \overleftarrow{\theta} \rangle$ denotes the over all parameters in Eq.(7), and $P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta})$ be the product of a pair of bidirectional models as follows:

$$P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta}) = \overrightarrow{\mathbf{P}}(\mathbf{e} \mid \mathbf{f}; \overrightarrow{\theta}) \times \overleftarrow{\mathbf{P}}(\mathbf{e} \mid \mathbf{f}; \overleftarrow{\theta}) \tag{8}$$

Then decoding can be formulated similarly by plugging Eq.(8) into Eq.(6). Note that although the right hand of Eq.(8) involves in two distributions, their product is not a distribution any more since it may not be normalized to 1 with respect to $\mathbf{e}$. However, it does not matter as a criteria for search and thus we still use P to denote the product of two probabilities in Eq.(8) in this paper for the consistency.

The above directional RNN is different from the ideas in many works, for example, Sundermeyer, Alkhouli, Wuebker, and Ney (2014), Bahdanau et al. (2015), Rao, Peng, Sak, and Beaufays (2012), Yao and Zweig (2015), where directions are specified by the source side instead of the target side as in our approach. Therefore, their bidirectional RNNs will still suffer from the shortcoming mentioned before. The source-side bidirectional method has been proven to be a basic and practical technique, and it can be easily employed in our

1. It might be better to constrain the embedding parameters across the two directional models as in Tamura, Watanabe, and Sumita (2014), and this remains future work.

models for potential improvements, but we skip it to instead to highlight the novelty of our model in this paper.

In addition, our agreement model employs a pair of RNNs and thus it is in some sense an ensemble. However, there are major differences between our idea and the neural network ensembles reported in the literature to date. Firstly, the decoding for each RNN in an ensemble of RNNs is straightforward to implement in the standard manner, whereas the decoding for our agreement model with different directional RNNs is challenging, as will be shown in the next section. Secondly, our idea is orthogonal to an ensemble, since the left-to-right and right-to-left RNNs of our agreement model can themselves be an ensemble of RNNs, and in fact this approach was taken in the experiments reported here.

3.2 Approximate Search

In the rest of this section, we firstly discuss the challenges in optimal search for the agreement model and then we propose two approximate solutions to address its search problem.

3.2.1 Challenges in Agreement Search

The exact inference for an agreement model is usually intractable, even in the cases where the individual models can be factorized locally. On an agreement task using HMMs, Liang et al. (2006) apply an approximate inference method which depends on the tractable calculation of the marginal probability. Unfortunately, this approximate method can not be used in our case, because our individual model (the RNN) is globally dependent and therefore such marginal calculations are not tractable.

Bidirectional search (Kaindl & Kainz, 1997) for agreement search is also impracticable for our agreement model. The reason being that the generation processes proceed in different directions; the agreement model generates partial sequences either in a left-to-right or in a right-to-left manner during the search. It is impossible to calculate both left-to-right and right-to-left model scores simultaneously for each partial sequence because the prediction of a latter token depends on the former token predictions in RNNs.

We propose two simple approximate methods for agreement search, which explore a smaller space than that of beam search. Their basic idea is aggresive pruning followed by exhaustive search: we first aggressively prune the entire exponential search space and then obtain the 1-best result via exhaustive search over the pruned space $\mathcal{S}$ with respect to the agreement model. Critical to the success of this approach is that the aggressive pruning must not eliminate the truly optimal hypothesis from the search space prior to the exhaustive search phase. In the rest of this section, we provide two methods to construct the pruned space $\mathcal{S}$.

3.2.2 Union b-best Approximation

Suppose b denotes the beam size in beam search algorithm, and $\mathrm{FBIN}_{l2r}(b)$ and $\mathrm{FBIN}_{r2l}(b)$ are two FBINs for the left-to-right and right-to-left RNN models in Algorithm 1, respectively. Then we construct the first search space $\mathcal{S}_1(b)$ as the union of these two sets:

$$\mathcal{S}_1(b) = \mathrm{FBIN}_{l2r}(b) \cup \mathrm{FBIN}_{r2l}(b)$$

Since both $\text{FBIN}_{l2r}(b)$ and $\text{FBIN}_{r2l}(b)$ contain at most b target sequences, exhaustively rescoring S_1 with the agreement model has complexity $O(b)$. One advantage of this method is that the search space is at most twice the size of that of its component RNN models, and since the b-best size for generic RNNs is typically very small, this method is computationally light. To make this explicit, in all the experiments reported here, the b-best size was 12, and the additional rescoring time was negligible.

3.2.3 Polynomial Approximation

Observing that both the prefixes of sequences in $\text{FBIN}_{l2r}(b)$ and the suffixes of sequences in FBIN_{r2l} are of high quality, we construct the second search space $\mathcal{S}_2(b)$ as follows:

$$\mathcal{S}_2(b) = \left\{ \mathbf{e}[:t] \circ \mathbf{e}'[t':] \;\middle|\; \mathbf{e} \in \text{FBIN}_{l2r}(b), \mathbf{e}' \in \text{FBIN}_{r2l}(b), 0 \leqslant t \leqslant |\mathbf{e}|, 0 \leqslant t' \leqslant |\mathbf{e}'| \right\}$$

where $\circ$ is a string concatenation operator, $[:t]$ is a prefix operator that yields the first t characters of a string, and $[t:]$ is a suffix operator that yields the last t characters. Since $\mathcal{S}_2(b)$ contains at most b^2N^2 target sequences, exhaustively rescoring over this space has complexity $O(b^2N^2)$, where N is length of the longest target sequence[2]. In our implementation, the speed for rescoring over this space was approximately 0.1 seconds per sentence, thanks to efficient use of a GPU. We can see that the search space of this method includes that of the first method as a proper subset ($\mathcal{S}_2(b) \supset \mathcal{S}_1(b)$), and thus this method can be expected to lead to higher 1-best agreement model scores than the previous method.

4. Analysis on Agreement Search

As discussed earlier, our agreement search only explores a tiny subset of the entire exponentially large space. Therefore, one may argue that our agreement search may cause serious search errors, and thus its performance is limited due to these search errors, which are known to undermine linear models on many sequence-to-sequence learning tasks (Collins & Roark, 2004; Huang et al., 2012; Liu & Huang, 2014). In this section, we firstly measure search errors for our agreement serach approaches according to model scores; and then we show whether these search errors largely degrade end-to-end performance according to evaluation metrics. To quantify the search errors in terms of both methods, we empirically conduct analysis on machine transliteration Jp-En task (see the details of the dataset in the Section §5.1 later), and use ACC (Word **Acc**uracy in Top-1) and Fscore (Fuzziness in Top-1, or Mean F-score) as two standard evaluation metrics following Zhang, Li, Liu, and Kumaran (2012).

4.1 Measuring Search Errors

Inspired by Huang and Chiang (2007), we employ the Expected Relative Scores (ERS) to measure the search errors for each agreement search approach represented by $\mathcal{S}_i(b)$ $(i = 1, 2)$ as follows:

$$\text{ERS}_i(b) = E_{\mathbf{f}}[\max_{e \in \Omega(\mathbf{f})} P(\mathbf{e} \mid \mathbf{f}) - \max_{\mathbf{e} \in \mathcal{S}_i(b)} P(\mathbf{e} \mid \mathbf{f})]$$

2. One can also design some methods to filter some undesirable concatenations safely, for example, those leading to too long or too short sequences. This will make rescoring much faster.

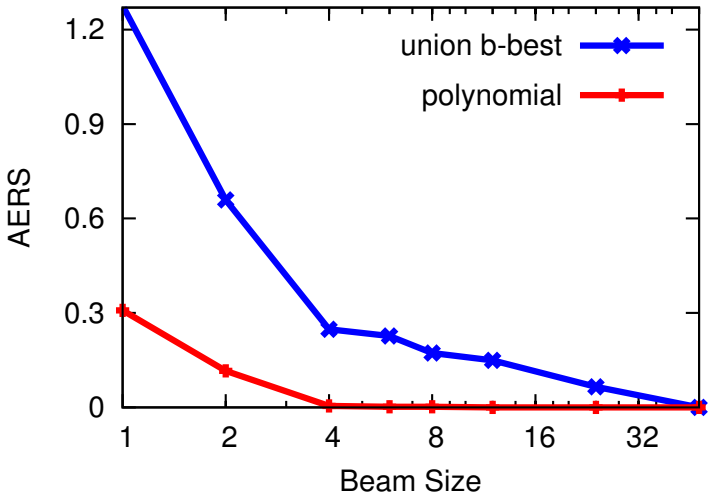


Figure 2: AERS with respect to different beam sizes 1, 2, 4, 8, 12, 24, and 48 in log scale.

where b denotes beam size. It is trivial to know $\text{ERS}_i \geqslant 0$, and if $\text{ERS}_i = 0$, then the search approach represented by $\mathcal{S}_i(b)$ has no search errors. However, it is intractable to calculate ERS_i exactly due to $\max_{\mathbf{e}\in\Omega(\mathbf{f})} P(\mathbf{e} \mid \mathbf{f})$. Instead, we adopt an Approximate Expected Relative Scores (AERS), which is a lower bound to measure the search errors:

$$\text{AERS}_i(b) = E_{\mathbf{f}}[\max_{\mathbf{e}\in\mathcal{S}_1(b_0)} P(\mathbf{e} \mid \mathbf{f}) - \max_{\mathbf{e}\in\mathcal{S}_i(b)} P(\mathbf{e} \mid \mathbf{f})] \leqslant \text{ERS}_i(b)$$

where $b_0 \gg b$ is a sufficiently large constant. Note that if $b_0 \to \infty$, then $\mathcal{S}_1(b_0) \to \Omega(\mathbf{f})$ and thereby $\text{AERS}_i(b)$ is close to $\text{ERS}_i(b)$ for any b. In this section, b_0 is set to 2000.

In order to measure the search errors, we randomly pick up some examples $\mathbf{f}$ from a machine transliteration task, and then calculate $\text{AERS}_i(b)$ for different beam sizes b. Figure 2 shows the $\text{AERS}_i(b)$ for both agreement search approaches according to b. We can see that $\text{AERS}_2(b)$ corresponding to polynomial approximation is smaller than $\text{AERS}_1(b)$ corresponding to union b-best, which shows that the former agreement search approach is better than the latter one according to search errors. In addition, with a beam size of 12, for both agreement search approaches, their ARES are good enough and thus their search quality is promising on this task. Note that since the vocabulary size is only 28, the AERS is relatively small; while it would be much larger for the task with large vocabulary size. [3]

4.2 Optimality towards Evaluation Metrics

Although our approximate agreement search approach leads to some search errors indicated by ARES, in this section we try to answer the question of how much improvement the optimal agreement search has over our approximate approach according to the evaluation metrics.

3. We did not calculate AERS on machine translation task, because it needs a very large b_0, which is impossible due to the computational limitation.

Suppose the parameters of our agreement model $\overleftrightarrow{\theta}$ are fixed after training, $\mathbf{r}$ is the reference sequence of $\mathbf{f}$, $\mathcal{S}_i(b)$ represents each of our approximate search approaches as before; let $\mathbf{e}^*$ be the optimal sequence, and $\hat{\mathbf{e}}_i$ be the sequence optimized via $\mathcal{S}_i(b)$, i.e. $\mathbf{e}^* = \mathbf{argmax}_{\mathbf{e}\in\Omega(f)} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta})$, and $\hat{\mathbf{e}}_i = \mathbf{argmax}_{\mathbf{e}\in\mathcal{S}_i(b)} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta})$. In addition, let $\mathcal{D} = \{\mathbf{e} \mid M(\mathbf{e}) > M(\hat{\mathbf{e}}_i)\}$ denote a domain of $\mathbf{e}$, where the metric represented by M (the higher is the better) of each element is better than that of $\hat{\mathbf{e}}_i$.

Firstly, we define the quantity Q as follows:

$$Q(\overleftrightarrow{\theta}) = \max_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta}) - P(\hat{\mathbf{e}}_i \mid \mathbf{f}; \overleftrightarrow{\theta})$$

We can obtain the following theorem:

Theorem 1. *Suppose* $P(\hat{\mathbf{e}}_i \mid \mathbf{f}; \overleftrightarrow{\theta}) < P(\mathbf{e}^* \mid \mathbf{f}; \overleftrightarrow{\theta})$*, if* $Q(\overleftrightarrow{\theta}) \leqslant 0$*, then* $M(\mathbf{e}^*) \leqslant M(\hat{\mathbf{e}}_i)$*.*

Proof. To prove it by contradiction, we assume $M(\mathbf{e}^*) > M(\hat{\mathbf{e}}_i)$, i.e. $\mathbf{e}^* \in \mathcal{D}$. Then $\max_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta}) \geqslant P(\mathbf{e}^* \mid \mathbf{f}; \overleftrightarrow{\theta})$.

On the other hand, $Q(\overleftrightarrow{\theta}) \leqslant 0$ induces $P(\hat{\mathbf{e}}_i \mid \mathbf{f}; \overleftrightarrow{\theta}) \geqslant \max_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta})$. Since $P(\mathbf{e}^* \mid \mathbf{f}; \overleftrightarrow{\theta}) > P(\hat{\mathbf{e}}_i \mid \mathbf{f}; \overleftrightarrow{\theta})$, one has $\max_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta}) < P(\mathbf{e}^* \mid \mathbf{f}; \overleftrightarrow{\theta})$. This is a contradiction. □

The above theorem shows that if $\hat{\mathbf{e}}_i$ is not optimal and $Q(\overleftrightarrow{\theta}) \leqslant 0$, then $\mathbf{e}^*$ is worse than $\hat{\mathbf{e}}_i$ according to the metric M. In other words, in this case, the optimal agreement search approach does not lead to improvements over our approximate approach. In contrast, if $Q(\overleftrightarrow{\theta}) > 0$, it is possible that $\mathbf{e}^*$ is equal to $\mathbf{argmax}_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta})$ and thus $M(\mathbf{e}^*) > M(\hat{\mathbf{e}}_i)$, where our approximate search undermines the performance in terms of M.

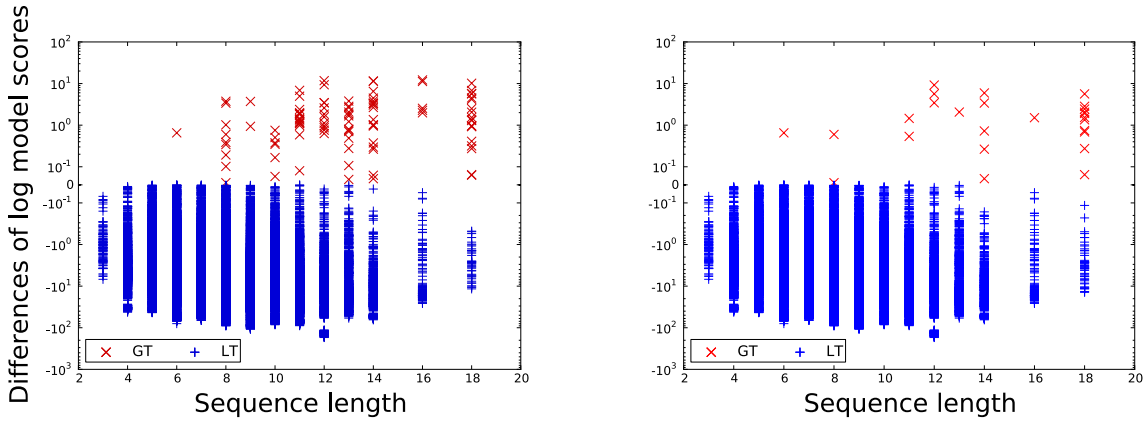
Using the above as a basis, we employ the following scheme to evaluate the potential of our search methods as follows: randomly select examples from the development set and calculate the distribution of the two cases $Q(\overleftrightarrow{\theta}) > 0$ and $Q(\overleftrightarrow{\theta}) \leqslant 0$, which are represented as **GT** (i.e. Greater Than) and **LT** (i.e. Less or equal Than), respectively. In addition, to alleviate the dependency on $\overleftrightarrow{\theta}$, we try 100 parameter sets optimized by our training algorithm starting from different initializations[4]. For simplicity, we run our bidirectional RNN model based a pair of AfRNNs on the machine transliteration task in this section.

However, there still is a problem to be addressed, which is to solve the constrained decoding, i.e. $\max_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta})$. Let us consider this problem in terms of different kinds of metrics M in the next.

4.2.1 Evaluation on Sequence-Level Metrics

If M is a sequence-level metric, for example, ACC for machine transliteration. It is a 0-1 metric, i.e. if e exactly matches its reference $\mathbf{r}$, then $M(\mathbf{e}) = 1$ otherwise $M(\mathbf{e}) = 0$. Therefore, $\mathcal{D} = \{\mathbf{e} \mid M(\mathbf{e}) > M(\hat{\mathbf{e}}_i)\}$ is either $\{\mathbf{r}\}$ or $\emptyset$ depending on $M(\hat{\mathbf{e}}_i)\} = 0$ or not. In this case, it is trivial to solve the constrained maximization problem and thus calculate the distribution of GT and LT based on the value of $Q(\overleftrightarrow{\theta})$. [5]

4. These parameters are from independently training the agreement model with 10 different initializations, as it is too costly to train with 100 initializations.
5. If $\mathcal{D} = \emptyset$, we assume that the maximization problem over $\mathcal{D}$ is well defined with value of $-\infty$.



(a) union b-best approximation (b) polynomial approximation

Figure 3: Potential estimations based on the distribution of GT (red 'x') and LT (blue '+') for union b-best (a) and polynomial (b) approximations along sequences with different length in terms of sequence level metrics. The ratio between GT and LT (less than 0.2%) indicates the search errors.

	Sequences	Scores	Acc	Fscore
$\hat{\mathbf{e}}_i$	F I G U E L O R A	-0.3	0	0.78
$\mathbf{e}^*$	F I G U E R O R A	-0.2	0	0.89
$\mathbf{r}$	F I G U E R O L A	-0.4	1	1.0

Table 1: The optimal sequence $\mathbf{e}^*$ can not lead to better Acc than $\hat{\mathbf{e}}_i$ obtained by our approximate search, but it leads to better Fscore. 'Scores' denotes the model scores, red color denotes an error regarding to reference $\mathbf{r}$.

Figure 3 shows that the distribution of GT and LT with respect to source sequence length for both approximate search methods. It is clear that there were some LT (plotted with a blue '+') cases for union b-best approximation method and many of them were eliminated by using polynomial approximation method. This fact shows that search errors do undermine the performance in terms of sequence-level metrics. Fortunately, the cases of LT (plotted with a blue '+') far outnumber the GT cases, and only 0.2% of all cases were GT, even for union b-best approximation. This 0.2% represents all this is possible to be gained by improving the search technique, and therefore both approximate methods can be said to be "almost optimal". In addition, we can see that the search quality for short sequences is much better than that for long sequences. This is as one expected, because the search space for short sequences is much smaller.

The above scheme relates to sequence-level metrics like Acc, but it can not give an indication of the effect on non sequence-level metrics such as Fscore. For example, in Table 1, since $\mathbf{e}^* \in \mathcal{D}$, $\max_{\mathbf{e}\in\mathcal{D}} P(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta}) \geqslant P(\mathbf{e}^* \mid \mathbf{f}; \overleftrightarrow{\theta}) > P(\hat{\mathbf{e}} \mid \mathbf{f}; \overleftrightarrow{\theta})$ and thus $Q(\overleftrightarrow{\theta}) > 0$.

Operations	Positions	Sequences
Substitute with R	6	F I G U E R O R A
Insert A	5	F I G U E A L O R A
Delete L	6	F I G U E O R A

Table 2: The operations to get successors for the state of sequence "F I G U E L O R A". "Positions" denotes the position at the sequence to be operated; and "Sequences" denotes the resulted sequences after the corresponding operation.

The optimal sequence e^* thus can not obtain gains in Acc, while it still gains in Fscore. Therefore, we need to consider the case of non-sequence-level metrics such as Fscore.

4.2.2 Evaluation on Non-Sequence-Level Metrics

Algorithm 2 Variant Beam Search Algorithm

Require: $\hat{\mathbf{e}}_i$, state $S_{\mathbf{r}}$ regarding to reference $\mathbf{r}$, beam size b, a large threshold B
Ensure: S_g

$i = 0$, BEAM $= [S_{\mathbf{r}}]$, $S_g = S_{\mathbf{r}}$
while BEAM **do**
  OPEN $= []$
  for all S in BEAM **do**
    for all S' in **get_successor**(S) and not **is_explored**(S') **do**
      if is_goal_state$(S'; \hat{\mathbf{e}}_i)$ **then**
        $i = i + 1$
        $S_g = \mathbf{min}(S', S_g)$
        add(OPEN,S')
        if $i \geqslant B$ **then**
          return S_g
  BEAM $= \mathbf{min\_pop}_b$(OPEN)
return S_g

Suppose M is a kind of non-sequence-level metrics such as Fscore, $\mathbf{e} \in \mathcal{D}$ contains exponentially many elements, and thus it is impracticable to enumerate the elements to solve the maximization problem. Furthermore, since $P(\mathbf{e} \mid \mathbf{f}, \overleftrightarrow{\theta})$ is defined upon bidirectional RNNs, it is intractable to exactly solve the maximization problem, and thus we develop a variant beam search approximate method as shown in Algorithm (2).

Algorithm 2 is similar to the standard beam search in Algorithm 1, but there are some differences. Firstly, we start from the state $S_{\mathbf{r}}$ corresponding to reference $\mathbf{r}$ instead of an empty state as in Algorithm 1 for search. Secondly, **get_successor**(S), **is_goal_state**$(S'; \hat{\mathbf{e}}_i)$ and $\mathbf{min}(S', S_g)$ have different definitions. **get_successor**(S) denotes a set of S' whose sequence differs from the sequence of S by an operation among substitution, insertion, and deletion as shown in Table 2; **is_goal_state**$(S'; \hat{\mathbf{e}}_i)$ means that $\hat{\mathbf{e}}_i$ is better than the sequence of S' according to a metric M; and $\mathbf{min}(S', S_g)$ is compared with the bidirectional RNNs

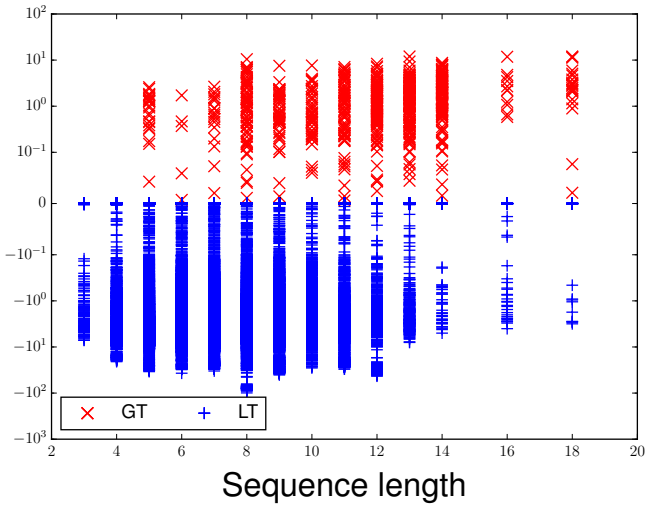


Figure 4: Potential estimations based on the distribution of GT (red 'x') and LT (blue '+') for union b-best approximation along sequences with different length in terms of non-sequence (token) level metric. The ratio between GT and LT (less than 2%) indicates the search errors.

rather than unidirectional RNN as in Algorithm 1. In addition, we use a large threshold B for better search quality. In this paper, b and B are set to 12 and 20000 to trade off the efficiency and search quality.

Figure 4 shows that the distribution of GT and LT with respect to source sequence length for both approximate search methods. We can see that there are more LT (plotted with a blue '+') cases in Figure 3 than before, which means that search errors lead to more drops in non-sequence-level metrics than those in sequence-level metrics. Fortunately, the cases of LT (plotted with a blue '+') are still dominated by GT cases. Their ratio of 2% indicates that the optimal search method can only achieve relative gains of 2% in non-sequence-level metrics, compared with our approximate method.

5. Experiments

We evaluated our approach on three standard sequence-to-sequence learning tasks: machine transliteration, grapheme-to-phoneme conversion, and machine translation. The first two are relatively easier sequence-to-sequence learning tasks, since there are no reorderings between source and target sequences; while the last one is difficult due to reorderings.

On evaluation, we use Acc and Fscore for machine transliteration, Wer (word error rate) and Per (phoneme error rate) for grapheme-to-phoneme, following Zhang et al. (2012), Kubo, Sakti, Neubig, Toda, and Nakamura (2014), Finch, Liu, Wang, and Sumita (2015), and Bleu for machine translation (Papineni, Roukos, Ward, & Zhu, 2002). For both Wer and Per, lower is better; while for Acc, Fscore and Bleu, higher is better. Note that Acc

and Wer are sequence-level metrics, while Fscore, Per and Bleu are non-sequence-level metrics.

In the experiments, we used several systems as baselines, which are listed below. The first four used open source implementations, and the last two were re-implemented:

- **Moses**: a phrase based machine translation model (Koehn et al., 2007) used with default settings except the monotonic decoding as in Finch and Sumita (2008); the reported results are the best from five independent runs of MERT.
- **Moses_hier**: a herarchical phrase based machine translation model (Chiang, 2005) implemented with the Moses toolkit.
- **DirecTL+**: a feature-rich linear model trained and run with default settings (Jiampojamarn, Cherry, & Kondrak, 2008)[6].
- **Sequitur G2P**: a joint n-gram model trained and run with default settings (Bisani & Ney, 2008).
- **AfRNN**: a attention-free RNN model with LSTM as its RNN unit, which is unidirectional model and re-implemented with Theano (Bergstra et al., 2010) following Sutskever et al. (2014).
- **AbRNN**: a neural translation model (Bahdanau et al., 2015) with an attention mechanism, which is unidirectional and from the open source NMT system.
- **EAfRNN**: an unidirectional ensemble of several AfRNNs with the same direction.
- **EAbRNN**: an unidirectional ensemble of several AbRNNs with the same direction.

In implementation of AfRNN, we reverse the source sequences for encoding as Sutskever et al. (2014). Our proposed bidirectional (agreement) RNN models are implemented on both AfRNN and AbRNN and denoted as follows:

- **BAfRNN**: bidirectional model including a single left-to-right (l2r) AfRNN and a single right-to-left (r2l) AfRNN.
- **BAbRNN**: bidirectional model including a single left-to-right (l2r) AbRNN and a single right-to-left (r2l) AbRNN.

Since our bidirectional model is orthognal to ensemble, and we develop bidirectional models based on ensemble of AfRNN and AbRNN for better performance:

- **BEAfRNN**: bidirectional model on the ensemble of AfRNN in both directions.
- **BEAbRNN**: bidirectional model on the ensemble of AbRNN in both directions.

In order to indicate the number of individual RNNs and its directions, we adopted the notations with format m-d-n, where m denotes the specific model, d denotes either left-to-right or right-to-left and n denotes the number of individual RNNs. For example, EAfRNN-l2r-5 denotes the unidirectional ensemble of five left-to-right AfRNNs and BEAbRNN-5 denotes the bidirectional model including five left-to-right AbRNNs and five right-to-left AbRNNs.

For all open source systems, their configurations and hyperparameters were set to their default settings, if not explicitly stated. In order to make the comparison fair, the stopping iteration was selected using the development set for all systems except Moses (which has its own termination criteria).

6. We tried various different settings but the default settings proved to be the most effective.

Approximations	Acc	Fscore
union b-best	33.3	85.1
polynomial	33.4	85.1
vanilla	32.7	84.9

Table 3: Performance of both search methods over the vanilla method using BAfRNN on the Jp-En test set. For the vanilla method, its rescoring space is fixed as the b-best list from one left-to-right AfRNN.

5.1 Machine Transliteration

For the machine transliteration task, we conducted both Japanese-to-English (Jp-En) and English-to-Japanese (En-Jp) directional subtasks. The transliteration training, development and test sets were taken from Wikipedia inter-language link titles[7] from Fukunishi, Finch, Yamamoto, and Sumita (2013): the training data consisted of 59000 sequence pairs composed of 313378 Japanese katakana characters and 445254 English characters; the development and test data were manually cleaned and each of them consisted of 1000 sequence pairs.

For all of the re-implemented models based on AfRNN, the number of word embedding units and hidden units were set to 500. We use the adadelta for training RNN based systems: the decay rate ρ and constant ϵ were set as 0.95 and 10^{-6} as suggested by Zeiler (2012), and minibatch sizes were 16.

In our experiments, we found one layer RNN works well for AfRNN, thanks to the limited vocabulary in this task. Therefore, we employ one layer RNN for all AfRNN based models including both unidirectional and bidirectional models.

5.1.1 Results

Table 3 shows the performance of the approximate search methods on the Jp-En test set, and their comparisons with a vanilla approximate method, which defines the rescoring space as the b-best list from left-to-right AfRNN model. We can see that both (i.e. union b-best and polynomial) methods achieve some improvements over the vanilla method. In addition, both proposed methods perform almost identically in terms of Acc and Fscore. This result is not surprising, because both of them are near optimal (as illustrated in the previous section). Therefore, in the remainder of the experiments, we only report the results using the union b-best approximate search.

Table 4 shows the results on the test sets of both Jp-En and En-Jp tasks. Firstly, we can see that the undirectional neural networks (AbRNN and AfRNN) have lower performance than the strongest non-neural network baselines (Sequitur G2P), even when they achieve comparable performance on En-Jp. Our agreement model BAfRNN shows substantial gains over both the AbRNN and AfRNN on both tasks. More specifically, the gain was up to 4.1 percentage points in terms of Acc and up to 4.8 percentage points in terms of Fscore. Moreover, BAfRNN showed comparable performance relative to Sequitur G2P on Jp-En

7. www.dumps.wikimedia.org

Systems	Detailed Model	Jp-En		En-Jp	
		Acc	Fscore	Acc	Fscore
Moses	log-linear	29.8	83.3	37.1	80.8
DirecTL+	feature-rich linear	11.1	75.1	31.7	79.9
Sequitur G2P	joint n-gram	34.6	84.6	39.8	81.6
AbRNN	RNN models	29.2	82.8	40.0	81.2
AfRNN		28.3	83.0	40.1	81.0
BAfRNN		33.3	85.1	43.8	85.0
EAfRNN-5		34.2	85.4	44.5	86.0
BEAfRNN-5		**36.3**	**86.0**	**45.3**	**86.3**

Table 4: The comparison of different systems on machine transliteration (Jp-En and En-Jp) tasks.

Systems	Prefix	Suffix
AfRNN-l2r	77%	65%
AfRNN-r2l	76%	74%
BAfRNN	80%	74%
EAfRNN-l2r-5	82%	73%
EAfRNN-r2l-5	82%	77%
BEAfRNN-5	82%	78%

Table 5: Precision of prefixes (the first two characters) and suffixes (the last two characters) on long sequences longer than 10 characters from the Jp-En test set.

task, and was markedly better on the En-Jp task. Note that since AfRNN is comparable to AbRNN, we did not run bidirectional models based on AbRNN in Table 4.

Secondly, the BEAfRNN-5 which used ensembles of five AfRNNs in both directions consistently achieved the best performance on both tasks, and outperformed Sequitur G2P by up to absolute gains of 5.5 points and 14% relative gains. In addition, BEAfRNN-5 outperformed the EAfRNN-5 by a substantial margin on all tasks, showing that our bidirectional agreement is effective in improving the performance of the unidirectional AfRNN on which it is based.

Furthermore it is clear that the gains of the BEAfRNN-5 relative to the EAfRNN-5 on Jp-En were larger than those on En-Jp. We believe the explanation is likely to be that the relative length of target sequences with respect to the source sequences on Jp-En is much larger than that on En-Jp, and our agreement model is able to draw greater advantage from the relatively longer target sequences. The relative length of the target for Jp-En was 1.43, whereas the relative length for En-Jp was only 0.70.

Systems	Acc	Fscore
AfRNN-l2r	17.3	82.2
AfRNN-r2l	18.5	83.5
BAfRNN	25.0	85.3
EAfRNN-l2r-5	24.4	86.8
EAfRNN-r2l-5	28.6	87.0
BEAfRNN-5	28.6	88.2

Table 6: Performance comparison on long sequences (> 10 tokens) on the Jp-En test set.

Systems	Acc	Fscore
AfRNN-l2r	28.3	83.4
AfRNN-r2l	29.7	83.6
EAfRNN-r2l-2	31.2	84.2
BAfRNN	33.3	85.1
EAfRNN-l2r-5	34.2	85.4
EAfRNN-r2l-10	34.0	85.2
EAfRNN-l2r-10	34.5	85.6
BEAfRNN-5	36.3	86.0
BEAfRNN-10	36.5	86.2

Table 7: Ensemble Uni- and Bidirectional AfRNNs compared on the Jp-En test set.

5.1.2 Analysis on Jp-En

One of the main weaknesses of RNNs is their unbalanced outputs which have high quality prefixes but low quality suffixes, as discussed earlier. Table 5 shows that the difference in precision is 12% for AfRNN-l2r between prefixes and suffixes. This gap narrowed using the BAfRNN, which outperformed the AfRNN-l2r on both prefix and suffix (with the largest difference on the suffix) and outperformed the AfRNN-r2l on the prefix. A similar effect was observed with the BEAfRNN-5, which generated the better, more balanced outputs compared to EAfRNN-l2r-5 and EAfRNN-r2l-5 models.

Our agreement model worked well for long sequences, and this is shown in Table 6. The BAfRNN obtained large gains over AfRNN-l2r and AfRNN-r2l, (the gains were up to 7.7 and 3.1 in terms of Acc and Fscore, respectively). Furthermore, the BEAfRNN-5 obtained gains of 1.2 points in terms of Fscore over the EAfRNN-r2l-5, but gave no improvements in terms of Acc. This is to be expected, since for long sequences it is hard to exactly match the references and thus it is more difficult to improve Acc.

To ensure a fairer comparison, the number of individual AfRNNs in both the ensemble and our agreement model were identical in the experiments. As shown in Table 7, although the BAfRNN explores a much smaller search space than the AfRNN-r2l-2, it substantially outperformed it. As the number of total number of AfRNNs used was increased to ten, the BEAfRNN-5 still obtained substantial gains over the EAfRNN-l2r-10. Incorporating more directional AfRNNs in the BEAfRNN-10 further increased the performance of the BEAfRNN.

Systems	WER	PER
Moses	31.0	7.0
DirecTL+	33.0	8.1
Sequitur G2P	25.0	6.0
AbRNN	29.4	7.9
AfRNN	29.6	8.0
BAfRNN	23.8	5.8
EAfRNN-5	22.1	5.3
BEAfRNN-5	**21.2**	**5.0**

Table 8: The comparison on machine transliteration on grapheme-to-phoneme (GM-PM) task.

	Model	WER	PER
Wu, Allauzen, Hall, Riley, and Roark (2014)	non-nn	23.4	5.5
Yao and Zweig (2015)	nn	23.6	5.5
Rao et al. (2012)	hybrid	21.3	-
This paper	nn	**21.2**	**5.0**

Table 9: Comparison with the reported results from different models on GM-PM task. '-' denotes no result was reported in the corresponding paper. 'nn' denotes a neural network model, 'non-nn' denotes a non-neural network model, and 'hybrid' denotes a linear combination between neural network and non-neural network models.

5.2 Graphenem to Phoneme Conversion

For grapheme-to-phoneme (GM-PM) conversion, the standard CMUdict[8] data sets were used: the original training set was randomly split into our training set (about 110000 sequence pairs) and development set (2000 pairs); the original test set consisting of about 12000 pairs was used for testing. For all of RNN based models, we used the same configuration and hyperparameters as in machine transliteration task except that the minibatch size was 64 for GM-PM task. In addition, we use the same baselines as in machine transliteration task to conduct experiments.

Table 4 shows the results on the test set of GM-PM. We can clearly see that the undirectional neural networks (AbRNN and AfRNN) are worse than the strongest non-neural network baselines (Sequitur G2P). Fortunately, with the help of bidirectional models, one can obtain substantial gains over both the AbRNN and AfRNN. Moreover, the BEAfRNN-5 achieved the best performance and outperformed Sequitur G2P by up to absolute gains of 5.5 points and 17% relative gains. In addition, BEAfRNN-5 outperformed the EAfRNN-5 by a substantial margin on all tasks. These results show that our bidirectional model is very effective in improving the unidirectional AfRNN.

8. http://www.speech.cs.cmu.edu/cgi-bin/cmudict

We also compare our results with the reported ones taken from other papers on Gm-Pm task [9], and these results are summarized in Table 9. This table shows that our results outperform these reported results from both non-neural network and neural network methods with a relative gains of up to 9%. Additionally, our end-to-end neural network method is slightly better than a hybrid method (Rao et al., 2012), which is a linear combination of WFST (Novak, Minematsu, & Hirose, 2012) and neural network models[10]. One of the our benefits over Rao et al. (2012), Yao and Zweig (2015) is that it does not need any external modules such as WFST or Aligner, and this makes ours more flexible.

5.3 Machine Translation

We conducted experiments on two challenging translation tasks: Japanese-to-English (Jp-En) and Chinese-to-English (Ch-En), using case-insensitive Bleu for evaluation. For the Jp-En task, we use the data from NTCIR-9 (Goto, Lu, Chow, Sumita, & Tsou, 2011): the training data consisted of 2.0M sentence pairs, The development and test sets contained 2K sentences with a single referece, respectively. For the Ch-En task, we used the data from the NIST2008 Open Machine Translation Campaign: the training data consisted of 1.8M sentence pairs, the development set was nist02 (878 sentences), and the test sets were nist05 (1082 sentences), nist06 (1664 sentences) and nist08 (1357 sentences).

Since the vocabulary size is much larger than that in machine transliteration, AfRNN with a single layer LSTM does not work well in our preliminary experiment. However, it is costly to train AfRNN with multiple layers on our large scale tasks, and thus we did not implement our bidirectional model on top of AfRNN but on top of AbRNN. As comparison, four baselines were used. The first two were the conventional phrase-based model Moses and hierarchical phrase-based model Moses_hier. and are respectively denoted as Moses and Moses-hier. The other two were AbRNN and its ensemble EAbRNN.

We followed the standard pipeline to train and run Moses. GIZA++ (Och & Ney, 2000) with grow-diag-final-and was used to build the translation model. We trained 5-gram target language models with srilm (Stolcke et al., 2002) using the training set for Jp-En and the Gigaword corpus for Ch-En, and used a lexicalized distortion model. All experiments were run with the default settings except for a distortion-limit of 12 in the Jp-En experiment, as suggested by Goto, Utiyama, Sumita, Tamura, and Kurohashi (2013).[11] To alleviate the negative effects of randomness, the final reported results are averaged over five runs of MERT as suggested by Clark, Dyer, Lavie, and Smith (2011). We used the following settings for AbRNN-based systems: the dimension of word embedding was 620, the dimension of hidden units was 1000, the batch size was 80, the source and target side vocabulary sizes were 30000, the maximum sequence length was set to 80, and the beam size for decoding was 12. Training was conducted on a single Tesla K80 GPU, and it took about 6 days to train a single AbRNN system on our large-scale data.

9. We can not present the similar comparison on machine transliteration tasks, since there are no previous publications conducting experiments on the same datasets as ours.
10. We believe that the linear combination of ours and non-neural network methods will lead to more improvements, but this is beyond the scope of this paper.
11. This configuration achieved the significant improvements over the default setting on Jp-En.

Systems	dev	test
Moses	27.9	29.4
Moses-hier	28.6	30.2
AbRNN-l2r	31.5	32.4
AbRNN-r2l	31.5	32.6
BAbRNN	33.0	34.1
EAbRNN-l2r-5	32.6	33.7
EAbRNN-r2l-5	33.0	34.3
BEAbRNN-5	**33.8**	**35.0**
EAbRNN-l2r-10	32.5	33.6
EAbRNN-r2l-10	33.0	34.2

Table 10: Bleu comparison of the proposed bidirectional models with baselines on Jp-En task.

Systems	nist05	nist06	nist08
Moses	35.4	33.7	25.0
Moses-hier	35.6	33.8	25.3
AbRNN-l2r	34.2	34.9	27.7
AbRNN-r2l	34.0	34.1	26.9
BAbRNN	36.8	36.9	28.5
EAbRNN-l2r-5	37.0	37.5	28.2
EAbRNN-r2l-5	36.9	37.1	27.3
BEAbRNN-5	**37.5**	**38.9**	**28.8**

Table 11: Bleu comparison of the proposed bidirectional models with baselines on Ch-En task.

5.3.1 Results

Table 10 shows the main results on the Jp-En task. From this table, we can see that, although a single AbRNN model (either left-to-right or right-to-left) comfortably outperforms the Moses and Moses-hier baselines, our simple BAbRNN (with one l2r and one r2l AbRNN model) obtain gains of 1.5 Bleu points over a single AbRNN. In addition, the more powerful bidirectional model BEAbRNN-5, which is an ensemble of five l2r and five r2l AbRNN models, gains 0.7 Bleu points over the strongest AbRNN ensemble EAbRNN-r2l-5, i.e. an ensemble of five r2l AbRNN models. The ensemble of bidirectional models achieved considerable gains of 5.6 and 4.8 Bleu points over the state-of-the-art Moses and Moses-hier, respectively.

One might argue that our BEAbRNN-5 contained ten AbRNN models in total, while the EAbRNN-l2r-5 or EAbRNN-r2l-5 only used five models, and thus such a comparison is unfair. Therefore, we integrated ten AbRNN models into the EAbRNN-r2l-10 ensemble. In Table 10, we can see that EAbRNN-r2l-10 is not necessarily better than EAbRNN-r2l-5, which is consistent with the findings reported in Zhou, Wu, and Tang (2002).

Table 11 shows the comparison between our method and the baselines on the Ch-En task.[12] The results were similar in character to the results for Jp-En. The proposed bidirectional model (BEAbRNN-5) consistently outperformed the strongest neural baseline (AbRNN-l2r-5), an ensemble of five l2r AbRNN models, on all the test sets with gains up to 1.4 Bleu points. Furthermore, our model again achieved substantial gains over the Moses and Moses-hier systems, in the range 1.9∼5.2 Bleu points, depending on the test set.

6. Related Work

Target-bidirectional decoding techniques were pioneered in statistical machine translation. For example, Watanabe and Sumita (2002), Finch and Sumita (2009), Zhang, Toutanova, Quirk, and Gao (2013) proposed these techniques for traditional SMT models instead of neural network models as ours. In particular, target-directional neural network models were also employed in Devlin, Zbib, Huang, Lamar, Schwartz, and Makhoul (2014). However, their approach was concerned with feedforward networks, which can not make full use of contextual information. Furthermore, their models were implemented using features (i.e. submodels) to augment traditional methods (for example, a hierarchical phrase-based translation model) in contrast to the end-to-end neural network model for sequence-to-sequence learning in our proposal.

Our work is closely related to the work of Liang et al. (2006) in the field of word alignment. However, we use the agreement RNNs that exploit the global context as opposed to the local HMM models; furthermore, the proposed approach combines left and right generation directions on the target side instead of source and target directions. In Tamura et al. (2014) a form of agreement for globally dependent RNN models was proposed for word alignment. Similar to the proposed method their models are trained jointly. Their approach differs from method in the directions used for agreement, and moreover their method does not consider decoding with the agreement model, which is a very challenging problem as discussed before in this paper.

Recurrent neural networks have become very popular for many sequence-to-sequence learning tasks. For example, Watanabe and Sumita (2015) and Dyer, Ballesteros, Ling, Matthews, and Smith (2015) employed RNNs and LSTMs for constituent and dependency parsing, where parse trees are generated as sequences of shift-reduce actions. For machine translation, Sutskever et al. (2014) introduced neural network based on LSTMs and Bahdanau et al. (2015) proposed an effective attention mechanism under the RNN framework. All of these works have advanced the state-of-the-art RNNs by notable improvements of a basic technique; one key benefit of proposed method is that it does not vertically extend these methods, but can be generally applied on top of them all.

Particularly, Yao and Zweig (2015) and Rao et al. (2012) also employed the LSTM models for the grapheme-to-phoneme conversion task. However, our model is directly oriented to an issue of recurrent neural network rather than a specific task itself. In addition, one of our advantages is that our model is very flexible: it is independent on any external toolkits such as the alignment toolkit in Yao and Zweig (2015) or WFST (weighted finite-state

12. We did not run AbRNN-l2r-10 and AbRNN-r2l-10, because it is too time-consuming to train 10 AbRNN models on both target directions and especially AbRNN-r2l-10 is not necessarily better than AbRNN-r2l-5 as shown in Table 10.

transducer) in Rao et al. (2012). Anyway, our model can be easily applied on top of them for potential improvements.

Finally, our work is related to Bengio, Vinyals, Jaitly, and Shazeer (2015) in some sense. Both approaches can alleviate the mismatch between the training and testing stages: the history predictions are always correct in training while may be incorrect in testing. Bengio et al. (2015) introduce noise into history predictions in training to balance the mistmatch, while we try to make the history predictions in testing as accurate as those in training by using of two directional models. Therefore, theirs focuses on this problem from the view of training instead of both modeling and training as ours, but it is possible and promising to apply their approach to optimize our joint model.

7. Conclusion

When generating the target in a unidirectional process for RNNs, the precision falls off with distance from the start of the sequence, and the generation of long sequences therefore becomes an issue. We propose an agreement model on target-bidirectional LSTMs that symmetrize the generative process. The exact search for this agreement model is NP-hard, and therefore we developed two approximate search alternatives, and analyze their behavior empirically, finding them to be near optimal. Extensive experiments showed our approach to be very promising, delivering substantial gains over a range of strong baselines on three standard sequence-to-sequence learning tasks: machine transliteration, grapheme-to-phoneme conversion and machine translation.

Acknowledgments

This article is an extended version of the work in Liu, Finch, Utiyama, and Sumita (2016) and Liu, Utiyama, Finch, and Sumita (2016). We mainly extend this work in two points: we measure the search errors for our approximate agreement search in terms of model scores; and we extend its optimality analysis from the case of sequence-level metrics to that of non-sequence-level metrics, after presenting a general theorem and a variant beam search algorithm for constrained decoding. The main technique of this article has achieved the top performance among several tasks in NEWS workshop (Finch, Liu, Wang, & Sumita, 2016). We would like to thank anonymous reviewers for helpful comments and suggestions. Particularly, we thank the reviewer who points out the bug on the inconsistency between $Q(\overleftrightarrow{\theta})$ and "GT" in section §4.2. In addition, we are grateful to Taro Watanabe, Liang Huang, Xugang Lu, Haitao Mi, and Shumpei Kubosawa for invaluable discussions on this work. Lemao Liu is currently affiliated with Tencent AI Lab, Shenzhen, China.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Pro-*

cessing Systems, pp. 1171–1179.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., & Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Bisani, M., & Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Commun.*

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Clark, J. H., Dyer, C., Lavie, A., & Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pp. 176–181. Association for Computational Linguistics.

Collins, M., & Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., & Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*.

Finch, A., Liu, L., Wang, X., & Sumita, E. (2015). Neural network transduction models in transliteration generation. In *Proceedings of ACL Workshop on NEWS*, pp. 61–66.

Finch, A., Liu, L., Wang, X., & Sumita, E. (2016). Target-bidirectional neural models for machine transliteration. In *Proceedings of ACL Workshop on NEWS*, pp. 78–82.

Finch, A., & Sumita, E. (2009). Bidirectional phrase-based statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 1124–1132, Singapore. Association for Computational Linguistics.

Finch, A. M., & Sumita, E. (2008). Phrase-based machine transliteration. In *Proceedings of IJCNLP*.

Fukunishi, T., Finch, A., Yamamoto, S., & Sumita, E. (2013). A bayesian alignment approach to transliteration mining.. *12*(3), 9:1–9:22.

Goto, I., Lu, B., Chow, K., Sumita, E., & Tsou, B. K. (2011). Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR-9*.

Goto, I., Utiyama, M., Sumita, E., Tamura, A., & Kurohashi, S. (2013). Distortion model considering rich context for statistical machine translation. In *Proceedings of ACL*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*.

Graves, A., & Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. In *Proceedings of NIPS*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*.

Huang, L., & Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 144–151, Prague, Czech Republic.

Huang, L., Fayong, S., & Guo, Y. (2012). Structured perceptron with inexact search. In *Proceedings of NAACL*.

Jean, S., Cho, K., Memisevic, R., & Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of ACL-IJCNLP*.

Jiampojamarn, S., Cherry, C., & Kondrak, G. (2008). Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-HLT*.

Kaindl, H., & Kainz, G. (1997). Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research*, *7*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., & Herbst, E. (2007). Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

Kubo, K., Sakti, S., Neubig, G., Toda, T., & Nakamura, S. (2014). Structured soft margin confidence weighted learning for grapheme-to-phoneme conversion. In *Proceedings of InterSpeech*.

Liang, P., Bouchard-Côté, A., Klein, D., & Taskar, B. (2006). An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*, Sydney, Australia.

Liu, L., Finch, A., Utiyama, M., & Sumita, E. (2016). Agreement on target-bidirectional lstms for sequence-to-sequence learning. In *Proceedings of AAAI*.

Liu, L., & Huang, L. (2014). Search-aware tuning for machine translation. In *Proceedings of EMNLP*.

Liu, L., Utiyama, M., Finch, A., & Sumita, E. (2016). Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 411–416, San Diego, California. Association for Computational Linguistics.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., & Zaremba, W. (2015). Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

Meng, F., Lu, Z., Tu, Z., Li, H., & Liu, Q. (2015). Neural transformation machine: A new architecture for sequence-to-sequence learning. *CoRR*.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

Novak, J. R., Minematsu, N., & Hirose, K. (2012). WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of ACL Workshop on FSMNLP*.

Och, F. J., & Ney, H. (2000). Improved statistical alignment models. In *Proceedings of ACL*, pp. 440–447.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics.

Rao, K., Peng, F., Sak, H., & Beaufays, F. (2012). Graphemeto-phoneme conversion using long short-term memory recurrent neural networks. In *IWSLT*, pp. 69–76.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Parallel distributed processing.. chap. Learning Internal Representations by Error Propagation.

Stolcke, A., et al. (2002). Srilm-an extensible language modeling toolkit.. In *Interspeech*, Vol. 2002, p. 2002.

Sundermeyer, M., Alkhouli, T., Wuebker, J., & Ney, H. (2014). Translation modeling with bidirectional recurrent neural networks. In *Proceedings of EMNLP*.

Sutskever, I., Vinyals, O., & Le, Q. V. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*.

Tamura, A., Watanabe, T., & Sumita, E. (2014). Recurrent neural networks for word alignment model. In *Proceedings of ACL*.

Watanabe, T., & Sumita, E. (2002). Bidirectional decoding for statistical machine translation. In *Proceeding of COLING*.

Watanabe, T., & Sumita, E. (2015). Transition-based neural constituent parsing. In *Proceedings of ACL*.

Wu, K., Allauzen, C., Hall, K. B., Riley, M., & Roark, B. (2014). Encoding linear models as weighted finite-state transducers. In *INTERSPEECH*.

Yao, K., & Zweig, G. (2015). Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *CoRR*.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*.

Zhang, H., Toutanova, K., Quirk, C., & Gao, J. (2013). Beyond left-to-right: Multiple decomposition structures for smt.. In *HLT-NAACL*, pp. 12–21.

Zhang, M., Li, H., Liu, M., & Kumaran, A. (2012). Whitepaper of news 2012 shared task on machine transliteration. In *Proceedings of NEWS Workshop*.

Zhou, Z., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artif. Intell.*