Agreement on Target-Bidirectional LSTMs for Sequence-to-Sequence Learning

Lemao Liu, Andrew Finch, Masao Utiyama, Eiichiro Sumita
National Institute of Information and Communications Technology (NICT)
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan
{lmliu,first.last}@nict.go.jp

Abstract

Recurrent neural networks, particularly the long short-term memory networks, are extremely appealing for sequence-to-sequence learning tasks. Despite their great success, they typically suffer from a fundamental 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 LSTMs, 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 sequence-level losses. Extensive experiments were performed on two standard sequence-to-sequence transduction tasks: machine transliteration and grapheme-to-phoneme transformation. The results show that the proposed approach achieves consistent and substantial improvements, compared to six state-of-the-art systems. In particular, our approach outperforms the best reported error rates by a margin (up to 9% relative gains) on the grapheme-to-phoneme task. Our toolkit is publicly available on https://github.com/lemaoliu/Agtarbidir.

Recurrent neural networks (RNNs) (Mikolov et al. 2010), particularly Long Short-term Memory networks (LSTMs)[1] (Hochreiter and Schmidhuber 1997; Graves 2013), provide a universal and powerful solution for various tasks that have traditionally required carefully designed, task-specific solutions. On classification tasks (Graves and Schmidhuber 2008; Tai, Socher, and 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 a more general and challenging tasks such as sequence-to-sequence learning (Sutskever, Vinyals, and 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, and Bengio 2015; Meng et al. 2015) or superior (Jean et al. 2015; Luong et al. 2015).

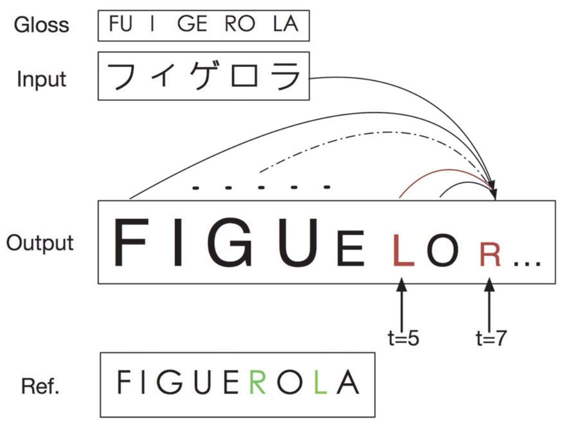


Figure 1: Illustraction of the fundamental shortcoming of an LSTM in decoding.

Despite their sucesses on sequence-to-sequnce learning, RNNs suffer from a fundamental and crucial shortcoming, which has surprisingly been overlooked. When making predictions (in decoding), an LSTM 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 noises, which undermine the quality of subsequent predicitons, as shown in Figure 1.

In the figure, larger fonts 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' (the green character 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, and found that on test examples longer than 10 characters, the precision of predictions for the first two characters was higher than 77%,

[1]Throughout this paper, an LSTM network denote a particular RNN with LSTM hidden units.

while for the last two it was only about 65% (see Experiments Section). 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 try to 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 joint search problem has been shown to be NP-hard, its exact solution is intractable, and we have therefore developed two approximate alternatives which are simple yet efficient. Even though the proposed search techniques consider only a tiny subset of the entire search space, our empirical results show them to be almost optimal in terms of sequence-level losses.

This paper makes the following contributions:

- It, for the first time, points out and formally analyzes a *fundamental* shortcoming, affecting recurrent neural networks in sequence-to-sequence learning.
- It proposes an *efficient* approximation of the joint search problem, and demonstrates empirically that it can achieve close to optimal performance. This approach is *general* enough to be applied to any deep recurrent neural networks.
- On both machine transliteration and grapheme-to-phoneme transduction tasks, the proposed approach consistently and substantially outperformed six state-of-the-art systems and particularly it advances the best reported error rates by a margin (up to 9% relative gains) on grapheme-to-phoneme task.

Although this paper focuses on machine transliteration and grapheme-to-phoneme tasks for reasons of simplicity, our work has the potential to be applied to any sequence-to-sequence learning tasks including machine translation, in which the generation of long sequences is challenging.

Revisiting the Generic LSTM

Suppose $\mathbf{x}$ denotes a general (either source or target) sequence of characters, its t^{th} character (at time step t) is $\mathbf{x}_t$ and its length is $|\mathbf{x}|$. In particular, a source sequence is denoted by $\mathbf{f}$ while a target sequence is denoted by $\mathbf{e}$. θ denotes the overall model parameter of recurrent neural networks: $\theta^{superscript}$ denotes a component parameter of θ depending on $superscript$, and it is either a bias vector (if $superscript$ includes b) or a matrix; $\theta(\mathbf{x}_t)$ is a vector representing embedding of $\mathbf{x}_t$, which is either a source character or a target character; $I(\theta, \mathbf{x}_t)$ denotes the index of $\mathbf{x}_t$ in the source or target vocabulary specified by $\mathbf{x}$. Note that in the rest of this paper, the subscript is reserved as the time step in a sequence for easier reading.

Model Definition

The sequence to sequence learning model by RNNs is defined as follows (Graves 2013):

$$\begin{aligned}\mathbf{P}(\mathbf{e} \mid \mathbf{f}; \theta) &= \prod_t \mathbf{P}(\mathbf{e}_t \mid h_t(\mathbf{e}); \theta) \\ &= \prod_t g\Big(\theta^l p\big(h_t(\mathbf{e})\big)\Big)\big[I(\theta, \mathbf{e}_t)\big]\end{aligned} \tag{1}$$

where g is a softmax function, p is an operator over a vector dependent on specific instances of RNNs, $vec[I]$ is a real number representing the I_{th} component of vector vec, and a vector $h_t(\mathbf{x}) = \Re(\mathbf{x}_t, h_{t-1}(\mathbf{x}); \theta)$ is the recurrent hidden state of sequence $\mathbf{x}$ at time step t with base cases: $h_{-1}(\mathbf{f}) = 0$ and $h_{-1}(\mathbf{e}) = h_{|\mathbf{f}|-1}(\mathbf{f})$.

For most RNNs (Mikolov et al. 2010), p is an identity function and $\Re(\mathbf{x}_t, h_{t-1}; \theta) = \phi(\theta^{\mathbf{x},h}\theta(\mathbf{x}_t) + \theta^{h,h}h_{t-1} + \theta^b)$; here ϕ as an activation function such as a sigmoid function, tanh, or a rectifier. In particular, for LSTM-based RNNs (Graves 2013), p is the projection funciton $p(h_t) = h_t^1$ (where h_t^1 is a component of h_t, i.e. $h_t = \langle h_t^1, h_t^2 \rangle$), and $\Re(\mathbf{x}_t, h_{t-1}; \theta) = \langle h_t^1, h_t^2 \rangle$ is obtained by following functions:

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

where σ denotes the sigmoid function and $\odot$ denotes the element-wise product over a pair of vectors.

Decoding

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

$$\hat{\mathbf{e}}\big(\mathbf{f}; \theta, \Omega(\mathbf{f})\big) = \underset{\mathbf{e} \in \Omega(\mathbf{f})}{\mathbf{argmax}}\, \mathbf{P}(\mathbf{e} \mid \mathbf{f}; \theta) \tag{2}$$

where $\mathbf{P}$ is given by Equation (1), and $\Omega(\mathbf{f})$ is the set of all possible target sequences $\mathbf{f}$ that can be generated using the target vocabulary. Since the prediciton at time step t (i.e. $\mathbf{e}_t$) is dependent on all the previous predictions, it is NP-hard to optimize the exact solution of Equation (2). Instead, an approximate solution, beam search, which generates the target sequence by extending one token each time from *left-to-right*, is widely applied (Sutskever, Vinyals, and Le 2014).

Fundamental Shortcoming

Despite their successes on various tasks, RNNs still suffer from a fundamental 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 $h_{t''}$ encode 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 $h_{t''}$. Ideally, if the probability of a correct prediction at t'' is $p_{t''}$, then will h_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.

Agreement on Target-bidirectional LSTMs

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

Formally, we develop the following joint target-bidirectional LSTM model:

$$\mathbf{P}_{\text{jnt}}(\mathbf{e} \mid \mathbf{f}; \overrightarrow{\theta}, \overleftarrow{\theta}) = \overrightarrow{\mathbf{P}}(\mathbf{e} \mid \mathbf{f}; \overrightarrow{\theta}) \times \overleftarrow{\mathbf{P}}(\mathbf{e} \mid \mathbf{f}; \overleftarrow{\theta}) \quad (3)$$

where $\overrightarrow{\mathbf{P}}$ and $\overleftarrow{\mathbf{P}}$ are the left-to-right and right-to-left LSTM models respectively, with definitions similar to Equation (1); $\overrightarrow{\theta}$ and $\overleftarrow{\theta}$ denote their parameters. This model is called an **agreement** model or **joint model** in this paper, and $\overleftrightarrow{\theta} = \left\langle \overrightarrow{\theta}, \overleftarrow{\theta} \right\rangle$ denotes its parameters for simplicity.

The training can be written as the minimization of the following equation[2]:

$$\min_{\overleftrightarrow{\theta}} \sum_{\langle \mathbf{f}, \mathbf{e} \rangle} \log \left(\mathbf{P}_{\text{jnt}}(\mathbf{e} \mid \mathbf{f}; \overleftrightarrow{\theta}) \right) \quad (4)$$

where the example $\langle \mathbf{f}, \mathbf{e} \rangle$ ranges over a given training set. To perform the optimization, we employ AdaDelta (Zeiler 2012), a mini-batch stochastic gradient method. The gradient is calculated using back-propogation through time (Rumelhart, Hinton, and Williams 1986), where the time is unlimited in our experiments. We employ the MAP strategy for testing, which is in essence Equation (2) with $\mathbf{P}_{\text{jnt}}$ (using the trained $\overleftrightarrow{\theta}$ parameters) plugged in.

Note that our directional LSTM is different from the ideas in many works, for example, (Sundermeyer et al. 2014; Bahdanau, Cho, and Bengio 2015; Rao et al. 2015; Yao and Zweig 2015), where directions are specified by the source side instead of the target side as in our approach. Therefore, their bidirectional LSTMs will still suffer from the shortcoming mentioned before. Anyway, source-side bidirectional method has been proved to be a basic and practical technique, and it can be easily employed in our models for potential improvements. But we skip it instead to highlight the novelty of our model in this paper.

[2]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.

In addition, our agreement model employs a pair of LSTMs 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 LSTM in an ensemble of LSTMs is straightforward to implement in the standard manner, whereas the decoding for our agreement model with different directional LSTMs 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 LSTMs of our agreement model can themselves be an ensemble of LSTMs, and in fact this approach was taken in the experiments reported here.

Approximations of Joint Search

Challenges in Joint 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) applies 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 LSTM) is globally dependent and therefore such marginal calculations are not possible (tractable).

Bidirectional search (Kaindl and Kainz 1997) for joint 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 due to the essence of RNNs.

We propose two simple approximate methods for joint search, which explore a smaller space than that of beam search. Their basic idea is aggresive pruning followed by exhaustive search: we first aggresively prune the entire exponential search space and then obtain the 1-best result via exhaustive search over the pruned space with respect to the agreement model. Critical to the success of this approach is that the aggressive pruning must not eliminate the promising hypothesis from the search space prior to the exhaustive search phase.

Joint k-best Approximation

Suppose L_{l2r} and L_{r2l} are two top-k target sequence sets from the generic left-to-right and right-to-left LSTM models, respectively. Then we construct the first search space $\mathcal{S}_1$ as the union of these two sets:

$$\mathcal{S}_1 = L_{l2r} \cup L_{r2l}$$

In this way, exhaustively re-scoring S_1 with the agreement model has complexity $O(k)$. One advantage of this method is that the search space is at most twice the size of that of its component LSTM models, and since the k-best size for generic LSTMs is typically very small, this method is computationally light. To make this explicit, in all the experiments reported here, the k-best size was 12, and the additional rescoring time was negligible.

Polynomial Approximation

Observing that both the prefixes of sequences in L_{l2r} and the suffixes of sequences in L_{r2l} are of high quality, we construct the second search space $\mathcal{S}_2$ as follows:

$$\mathcal{S}_2 = \Big\{ \mathbf{e}[:t] \circ \mathbf{e}'[t':] \;\Big|\; \mathbf{e} \in L_{l2r}, \mathbf{e}' \in L_{r2l}, \\ 0 \leqslant t \leqslant |\mathbf{e}|, 0 \leqslant t' \leqslant |\mathbf{e}'| \Big\}$$

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. Exhaustively rescoring over this space has complexity $O(k^2N^2)$, where N is length of the longest target sequence[3]. 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 ($\mathcal{S}_2 \supset \mathcal{S}_1$), and thus this method can be expected to lead to higher 1-best agreement model scores than the previous method.

Both approximate methods explore only a tiny subset of the exponential space $\Omega(\mathbf{f})$, and thus they may introduce search errors, that is, there are some sequences beyond our pruned search space $\mathcal{S}$, whose agreement model scores are higher than those of our 1-best sequence. Therefore, one might argue that their performance is limited due to these search errors, which are known to undermine linear models on many sequence-to-sequence learning tasks (Collins and Roark 2004; Huang, Fayong, and Guo 2012; Liu and Huang 2014). However, our empirical analysis later on examines the extent of this problem and will show that even an optimal (i.e. search error free) method, was not able to achieve significantly better performance than either of our approximate methods.

Experiments

Experimental Methodology

We evaluated our approach on machine transliteration and grapheme-to-phoneme conversion tasks. 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[4]: 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 grapheme-to-phoneme (GM-PM) conversion, the standard CMUdict[5] 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. On evaluation, we use ACC and FSCORE for machine transliteration and WER and PER for grapheme-to-phoneme, following (Zhang et al. 2012; Kubo et al. 2014). Note that ACC and WER are sequence-level metrics, while FSCORE and PER are non-sequence-level.

Six baseline systems, were used and are listed below. The first 4 used open source implementations, and the last 2 were re-implemented:

- **Moses**: a machine translation system (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.
- **DirecTL+**: a feature-rich linear model trained and run with default settings (Jiampojamarn, Cherry, and Kondrak 2008)[6].
- **Sequitur G2P**: a joint n-gram model trained and run with default settings (Bisani and Ney 2008).
- **NMT**: a neural translation model (Bahdanau, Cho, and Bengio 2015), RNN with Gated Recurrent Units (GRU), with default settings except the word embedding dimension of 500.
- **GLSTM**: a single generic LSTM and that was re-implemented with Theano (Bergstra et al. 2010) following (Sutskever, Vinyals, and Le 2014).
- **ELSTM**: an ensemble of several GLSTMs with the same direction.

In implementation of GLSTM, we reverse the source sequences for encoding as (Sutskever, Vinyals, and Le 2014). In our experiments, we found one layer LSTM works well. A possible reason might be our limited vocabulary in both tasks.

Our proposed bidirectional (agreement) LSTM models are denoted:

- **BLSTM**: a single left-to-right (l2r) LSTM and a single right-to-left (r2l) LSTM.
- **BELSTM**: ensembles of LSTMs in both directions.

In addition we use the following notation: nl2r or nr2l denotes the number of left-to-right or right-to-left LSTMs in the ensembles of the ELSTM and BELSTM. For example, BELSTM (5l2r+5r2l) denotes ensembles of five l2r and five r2l LSTMs in the BELSTM.

For fair comparison, the stopping iteration for all systems was selected using the development set for all systems except Moses (which has its own termination criteria). For all of the re-implemented models, the number of word embedding units and hidden units were set to 500 to match the configuration using in the NTM. We use the adadelta for training both GLSTM and proposed 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 and 64 for machine transliteration and GM-PM tasks, respectively.

Evaluation of the Joint Search Strategies

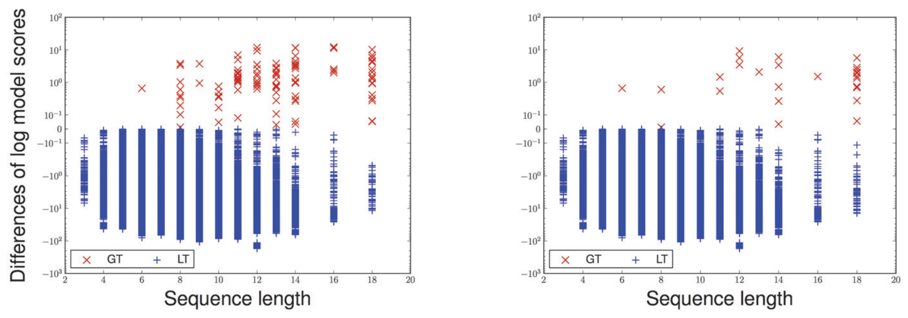
Suppose the parameters of our agreement model $\overleftrightarrow{\theta}$ are fixed after training, $\mathbf{e}$ is the reference sequence of $\mathbf{f}$, $\mathcal{S}$ denotes the search space (either $\mathcal{S}_1$ or $\mathcal{S}_2$) of our approximate methods,

[3]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.

[4]www.dumps.wikimedia.org

[5]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

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



(a) joint k-best approximation (b) polynomial approximation

Figure 2: Potential estimations based on the distribution of GT (red 'x') and LT (blue '+') for joint k-best (a) and polynomial (b) approximations along sequences with different length. The number of GT indicates the search errors.

and $\hat{\mathrm{e}}\big(\mathbf{f};\overleftrightarrow{\theta},\Omega\big)$, defined as in Equation (2), is the best target sequence of f in the search space $\Omega \in \{\mathcal{S}_1, \mathcal{S}_2, \Omega(f)\}$.

If $\mathbf{P}_{\mathrm{jnt}}(\mathrm{e} \mid \mathbf{f};\overleftrightarrow{\theta}) == \mathbf{P}_{\mathrm{jnt}}\big(\hat{\mathrm{e}}(\mathbf{f};\overleftrightarrow{\theta},\mathcal{S}) \mid \mathbf{f};\overleftrightarrow{\theta}\big)$, then our approximate search has resulted in the reference, as desired [7]. Otherwise, we have the following possible outcomes:

- **GT**: if $\mathbf{P}_{\mathrm{jnt}}(\mathrm{e} \mid \mathbf{f};\overleftrightarrow{\theta}) > \mathbf{P}_{\mathrm{jnt}}\big(\hat{\mathrm{e}}(\mathbf{f};\overleftrightarrow{\theta},\mathcal{S}) \mid \mathbf{f};\overleftrightarrow{\theta}\big)$, our approximate search is insufficient, since $\mathbf{P}_{\mathrm{jnt}}(\mathrm{e} \mid \mathbf{f};\overleftrightarrow{\theta})$ might be equal to $\mathbf{P}_{\mathrm{jnt}}\big(\hat{\mathrm{e}}(\mathbf{f};\overleftrightarrow{\theta},\Omega(\mathbf{f})) \mid \mathbf{f};\overleftrightarrow{\theta}\big)$, which is the upper bound of $\mathbf{P}_{\mathrm{jnt}}\big(\hat{\mathrm{e}}(\mathbf{f};\overleftrightarrow{\theta},\mathcal{S}) \mid f;\overleftrightarrow{\theta}\big)$ and the globally optimal probability. Therefore, this case is concerned with search errors.
- **LT**: if $\mathbf{P}_{\mathrm{jnt}}(\mathrm{e} \mid \mathbf{f};\overleftrightarrow{\theta}) < \mathbf{P}_{\mathrm{jnt}}\big(\hat{\mathrm{e}}(\mathbf{f};\overleftrightarrow{\theta},\mathcal{S}) \mid \mathbf{f};\overleftrightarrow{\theta}\big)$, then $\mathbf{P}_{\mathrm{jnt}}(\mathrm{e} \mid \mathbf{f};\overleftrightarrow{\theta})$ is definitely less than $\mathbf{P}_{\mathrm{jnt}}\big(\hat{\mathrm{e}}(\mathbf{f};\overleftrightarrow{\theta},\Omega(\mathbf{f})) \mid \mathbf{f};\overleftrightarrow{\theta}\big)$. In other words, even if we have the optimal joint search method, it still cannot find the correct sequence as the reference. The quality of the model rather than the search is the issue in this case.

Using this as a basis, we designed a scheme to evaluate the potential of our search methods as follows: we randomly select examples from the development set and compare the model scores of the references and the 1-best results from the approximate search methods; then analyze the distributions of the two cases GT and LT, where our model fails. In addition, to alleviate the dependency on $\overleftrightarrow{\theta}$, we tried 100 parameter sets optimized by our training algorithm starting from different initializations[8].

Figure 2 shows that the distribution of GT and LT with respect to source sequence length for both approximate search methods. It is clear that joint k-best approximation suffers from some search errors shown on the graph as GT (a red 'x'), many of which were eliminated by using polynomial approximation method. 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 joint k-best approximation. This 0.2% represents all this is possible to gained by improving the search technique, and therefore both approximate methods can be said to be "almost optimal".

[7]It is possible for $\mathbf{e}$ and $\hat{\mathbf{e}}$ to have the same score, even though they are not equal, but this is very unlikely in our experiment.

[8]These parameters were from independently training the joint model with 10 different initializations, as it is too costly to train with 100 initializations.

Approximations	ACC	FSCORE
joint k-best	33.3	85.1
polynomial	33.4	85.1
vanilla	32.7	84.9

Table 1: Performance of both search methods over the vanilla method using BLSTM (l2r+r2l) on the JP-EN test set. For the vanilla method, its rescoring space is fixed as the kbest list from left-to-right LSTM.

The above scheme relates to sequence-level losses like ACC, but it can not give an indication of the effect on non sequence-level losses. Empirically however, our approximate search methods appeared to be effective when performance was measured using non-sequence-level losses (FSCORE). The reason may be that non-sequence-level losses are usually positively correlated to sequence-level losses.

Main Results

Table 1 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 kbest list from left-to-right LSTM model. We can see that both (i.e. joint k-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 joint k-best approximate search.

Table 2 shows the results on the test sets of all three tasks: JP-EN, EN-JP and GM-PM. Firstly, we can see that the undirectional neural networks (NMT and GLSTM) have lower performance than the strongest non-neural network baselines (Sequitur G2P), even when they achieve comparable performance on EN-JP. Our agreement model BLSTM

Systems	Detailed Model	JP-EN		EN-JP		GM-PM	
		ACC↑	FSCORE↑	ACC↑	FSCORE↑	WER↓	PER↓
Moses	log-linear	29.8	83.3	37.1	80.8	31.0	7.0
DirecTL+	feature-rich linear	11.1	75.1	31.7	79.9	33.0	8.1
Sequitur G2P	joint n-gram	34.6	84.6	39.8	81.6	25.0	6.0
NMT	GRU RNN	29.2	82.8	40.0	81.2	29.4	7.9
GLSTM	(l2r) LSTM	28.3	83.0	40.1	81.0	29.6	8.0
BLSTM	(l2r+r2l) LSTMs	33.3	85.1	43.8	85.0	23.8	5.8
ELSTM	5 LSTMs	34.2	85.4	44.5	86.0	22.1	5.3
BELSTM	(5l2r+5r2l) LSTMs	**36.3**	**86.0**	**45.3**	**86.3**	**21.2**	**5.0**

Table 2: The comparison on machine transliteration (JP-EN and EN-JP) and grapheme-to-phoneme (GM-PM) tasks. ↑ denotes the higher is the better, while ↓ the lower the better.

	Model	ACC	FSCORE
(Wu et al. 2014)	non-nn	23.4	5.5
(Yao and Zweig 2015)	nn	23.6	5.5
(Rao et al. 2015)	hybrid	21.3	-
This paper	nn	**21.2**	**5.0**

Table 3: Comparison with the best 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.

Systems	Prefix	Suffix
GLSTM(l2r)	77%	65%
GLSTM(r2l)	76%	74%
BLSTM	80%	74%
ELSTM(5l2r)	82%	73%
ELSTM(5r2l)	82%	77%
BELSTM	82%	78%

Table 4: 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.

shows substantial gains over both the GLSTM and NMT on all three tasks. More specifically, the gain was up to 5.8 percentage points in terms of a sequence-level loss (WER) and up to 4.0 percentage points in terms of a non-sequence-level loss (FSCORE). Moreover, BLSTM showed comparable performance relative to Sequitur G2P on both JP-EN and GM-PM, and was markedly better on the EN-JP task.

Secondly, the BELSTM which used ensembles of five LSTMs in both directions consistently achieved the best performance on all the three tasks, and outperformed Sequitur G2P by up to absolute gains of 5.5 points and 18% relative gains. In addition, BELSTM outperformed the ELSTM by a substantial margin on all tasks, showing that our bidirectional agreement is effective in improving the performance of the unidirectional ELSTM on which it is based.

Furthermore it is clear that the gains of the BELSTM relative to the ELSTM on JP-EN were larger than those on both EN-JP and GM-PM. 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 those on EN-JP and GM-PM, 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 lengths for EN-JP and GM-PM were only 0.70 and 0.85 respectively.

We also compare our results with the best reported ones on GM-PM task to date[9], and these results are summarized in Table 3. To the best of our knowledge, our results outperform the best 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. 2015), which is a linear combination of WFST (Novak, Minematsu, and Hirose 2012) and neural network models[10]. One of the our benefits over (Rao et al. 2015; Yao and Zweig 2015) is that it does not need any external modules such as WFST or Aligner, and this makes ours more flexible.

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 4 shows that the difference in precision is 12% for GLSTM (l2r) between prefixes and suffixes. This gap narrowed using the BLSTM, which outperformed the GLSTM (l2r) on both prefix and suffix (with the largest difference on the suffix) and outperformed the GLSTM (r2l) on the prefix. A similar effect was observed with the BELSTM, which generated the better, more balanced outputs compared to ELSTM(5l2r) and ELSTM(5r2l) models.

Our agreement model worked well for long sequences, and this is shown in Table 5. The BLSTM obtained large gains over GLSTM(l2r) and GLSTM(r2l), (the gains were

[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.

Systems	ACC	FSCORE
GLSTM(l2r)	17.3	82.2
GLSTM(r2l)	18.5	83.5
BLSTM	25.0	85.3
ELSTM(5l2r)	24.4	86.8
ELSTM(5r2l)	28.6	87.0
BELSTM	28.6	88.2

Table 5: Performance comparison on long sequences (>10 tokens) on the JP-EN test set.

Systems	ACC	FSCORE
GLSTM(l2r)	28.3	83.4
GLSTM(r2l)	29.7	83.6
ELSTM(2r2l)	31.2	84.2
BLSTM(l2r+r2l)	33.3	85.1
ELSTM(5l2r)	34.2	85.4
ELSTM(5r2l)	34.0	85.2
ELSTM(10l2r)	34.5	85.6
BELSTM(5l2r+5r2l)	36.3	86.0
BELSTM(10l2r+10r2l)	36.5	86.2

Table 6: Ensemble Uni- and Bidirectional LSTMs compared on the JP-EN test set.

up to 7.7 and 3.1 in terms of ACC and FSCORE, respectively). Furthermore, the BELSTM obtained gains of 1.2 points in terms of FSCORE over the ELSTM(5r2l), but gave no improvements in terms of ACC. This is to be expected, since for long sequences it is hard to generate targets that exactly match the references and thus it is more difficult to improve ACC.

To ensure a more fair comparison, the number of individual LSTMs in both the ensemble and our agreement model were identical in the experiments. As shown in Table 6, although the BLSTM(r2l+l2r) explores a much smaller search space than the ELSTM(2r2l), it substantially outperformed it. As the number of total number of LSTMs used was increased to ten, the BELSTM(5l2r+5r2l) still obtained substantial gains over the ELSTM(10l2r). Incorporating more directional LSTMs in the BELSTM(10l2r+10r2l) further increased the performance of the BELSTM.

Related Work

Target-bidirectional decoding techniques were pioneered in statistical machine translation. For example, (Watanabe and Sumita 2002; Finch and Sumita 2009; Zhang et al. 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 et al. 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, Watanabe, and Sumita 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 et al. 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, Vinyals, and Le 2014) introduced neural network based on LSTMs and (Bahdanau, Cho, and Bengio 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. 2015) also employed the LSTM models for the grapheme-to-phoneme conversion task. However, our model is directly oriented to the fundamental 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 in (Rao et al. 2015). Anyway, our model can be easily applied on top of them for potential improvements.

Conclusions

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 both machine transliteration and grapheme-to-phoneme conversion. Furthermore, our method has achieved the best reported results to date, on a standard grapheme-to-phoneme conversion dataset.

Acknowledgments

We would like to thank the three anonymous reviewers for helpful comments and suggestions. Especially, we thank

Taro Watanabe, Liang Huang, Haitao Mi, and Shumpei Kubosawa for invaluable discussions on this work.

References

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

Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D.; and Bengio, Y. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of SciPy*.

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

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

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

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

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

Finch, A., and Sumita, E. 2009. Bidirectional phrase-based statistical machine translation. In *Proceedings of EMNLP*.

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

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

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

Huang, L.; Fayong, S.; and Guo, Y. 2012. Structured perceptron with inexact search. In *Proceedings of NAACL*.

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

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

Kaindl, H., and 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.; and Herbst, E. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

Kubo, K.; Sakti, S.; Neubig, G.; Toda, T.; and 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.; and Taskar, B. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*.

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

Luong, T.; Sutskever, I.; Le, Q.; Vinyals, O.; and Zaremba, W. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL-IJCNLP*.

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

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

Novak, J. R.; Minematsu, N.; and 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*.

Rao, K.; Peng, F.; Sak, H.; and Beaufays, F. 2015. Graphemeto-phoneme conversion using long short-term memory recurrent neural networks. In *ICASSP*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Parallel distributed processing. chapter Learning Internal Representations by Error Propagation.

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

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

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

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

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

Watanabe, T., and Sumita, E. 2015. Transition-based neural constituent parsing. In *Proceedings of ACL-IJCNLP*.

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

Yao, K., and 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, M.; Li, H.; Liu, M.; and Kumaran, A. 2012. Whitepaper of news 2012 shared task on machine transliteration. In *Proceedings of NEWS Workshop*.

Zhang, H.; Toutanova, K.; Quirk, C.; and Gao, J. 2013. Beyond left-to-right: Multiple decomposition structures for smt. In *HLT-NAACL*.