NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1 INTRODUCTION

Neural machine translation is a newly emerging approach to machine translation, recently proposed by Kalchbrenner and Blunsom (2013), Sutskever *et al.* (2014) and Cho *et al.* (2014b). Unlike the traditional phrase-based translation system (see, e.g., Koehn *et al.*, 2003) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Most of the proposed neural machine translation models belong to a family of *encoder–decoders* (Sutskever *et al.*, 2014; Cho *et al.*, 2014a), with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared (Hermann and Blunsom, 2014). An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.

In order to address this issue, we introduce an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

*CIFAR Senior Fellow

The most important distinguishing feature of this approach from the basic encoder–decoder is that it does not attempt to encode a whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. This frees a neural translation model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector. We show this allows a model to cope better with long sentences.

In this paper, we show that the proposed approach of jointly learning to align and translate achieves significantly improved translation performance over the basic encoder–decoder approach. The improvement is more apparent with longer sentences, but can be observed with sentences of any length. On the task of English-to-French translation, the proposed approach achieves, with a single model, a translation performance comparable, or close, to the conventional phrase-based system. Furthermore, qualitative analysis reveals that the proposed model finds a linguistically plausible (soft-)alignment between a source sentence and the corresponding target sentence.

2 Background: Neural Machine Translation

From a probabilistic perspective, translation is equivalent to finding a target sentence $\mathbf{y}$ that maximizes the conditional probability of $\mathbf{y}$ given a source sentence $\mathbf{x}$, i.e., $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$. In neural machine translation, we fit a parameterized model to maximize the conditional probability of sentence pairs using a parallel training corpus. Once the conditional distribution is learned by a translation model, given a source sentence a corresponding translation can be generated by searching for the sentence that maximizes the conditional probability.

Recently, a number of papers have proposed the use of neural networks to directly learn this conditional distribution (see, e.g., Kalchbrenner and Blunsom, 2013; Cho *et al.*, 2014a; Sutskever *et al.*, 2014; Cho *et al.*, 2014b; Forcada and Ñeco, 1997). This neural machine translation approach typically consists of two components, the first of which encodes a source sentence $\mathbf{x}$ and the second decodes to a target sentence $\mathbf{y}$. For instance, two recurrent neural networks (RNN) were used by (Cho *et al.*, 2014a) and (Sutskever *et al.*, 2014) to encode a variable-length source sentence into a fixed-length vector and to decode the vector into a variable-length target sentence.

Despite being a quite new approach, neural machine translation has already shown promising results. Sutskever *et al.* (2014) reported that the neural machine translation based on RNNs with long short-term memory (LSTM) units achieves close to the state-of-the-art performance of the conventional phrase-based machine translation system on an English-to-French translation task.[1] Adding neural components to existing translation systems, for instance, to score the phrase pairs in the phrase table (Cho *et al.*, 2014a) or to re-rank candidate translations (Sutskever *et al.*, 2014), has allowed to surpass the previous state-of-the-art performance level.

2.1 RNN Encoder–Decoder

Here, we describe briefly the underlying framework, called *RNN Encoder–Decoder*, proposed by Cho *et al.* (2014a) and Sutskever *et al.* (2014) upon which we build a novel architecture that learns to align and translate simultaneously.

In the Encoder–Decoder framework, an encoder reads the input sentence, a sequence of vectors $\mathbf{x} = (x_1, \cdots, x_{T_x})$, into a vector c.[2] The most common approach is to use an RNN such that

$$h_t = f\left(x_t, h_{t-1}\right) \tag{1}$$

and

$$c = q\left(\{h_1, \cdots, h_{T_x}\}\right),$$

where $h_t \in \mathbb{R}^n$ is a hidden state at time t, and c is a vector generated from the sequence of the hidden states. f and q are some nonlinear functions. Sutskever *et al.* (2014) used an LSTM as f and $q\left(\{h_1, \cdots, h_T\}\right) = h_T$, for instance.

[1] We mean by the state-of-the-art performance, the performance of the conventional phrase-based system without using any neural network-based component.

[2] Although most of the previous works (see, e.g., Cho *et al.*, 2014a; Sutskever *et al.*, 2014; Kalchbrenner and Blunsom, 2013) used to encode a variable-length input sentence into a *fixed-length* vector, it is not necessary, and even it may be beneficial to have a *variable-length* vector, as we will show later.

The decoder is often trained to predict the next word $y_{t'}$ given the context vector c and all the previously predicted words $\{y_1, \cdots, y_{t'-1}\}$. In other words, the decoder defines a probability over the translation $\mathbf{y}$ by decomposing the joint probability into the ordered conditionals:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c), \tag{2}$$

where $\mathbf{y} = (y_1, \cdots, y_{T_y})$. With an RNN, each conditional probability is modeled as

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c), \tag{3}$$

where g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t, and s_t is the hidden state of the RNN. It should be noted that other architectures such as a hybrid of an RNN and a de-convolutional neural network can be used (Kalchbrenner and Blunsom, 2013).

3 Learning to Align and Translate

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 Decoder: General Description

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \tag{4}$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

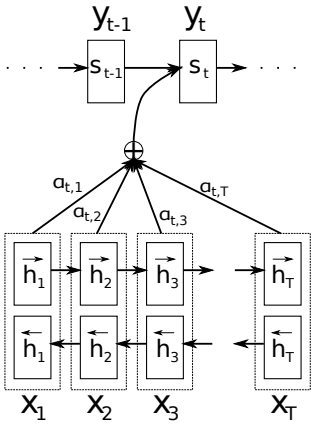

Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{5}$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{6}$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

is an *alignment model* which scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden state s_{i-1} (just before emitting y_i, Eq. (4)) and the j-th annotation h_j of the input sentence.

We parametrize the alignment model a as a feedforward neural network which is jointly trained with all the other components of the proposed system. Note that unlike in traditional machine translation,

the alignment is not considered to be a latent variable. Instead, the alignment model directly computes a soft alignment, which allows the gradient of the cost function to be backpropagated through. This gradient can be used to train the alignment model as well as the whole translation model jointly.

We can understand the approach of taking a weighted sum of all the annotations as computing an *expected annotation*, where the expectation is over possible alignments. Let α_{ij} be a probability that the target word y_i is aligned to, or translated from, a source word x_j. Then, the i-th context vector c_i is the expected annotation over all the annotations with probabilities α_{ij}.

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed-length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.

3.2 Encoder: Bidirectional RNN for Annotating Sequences

The usual RNN, described in Eq. (1), reads an input sequence $\mathbf{x}$ in order starting from the first symbol x_1 to the last one x_{T_x}. However, in the proposed scheme, we would like the annotation of each word to summarize not only the preceding words, but also the following words. Hence, we propose to use a bidirectional RNN (BiRNN, Schuster and Paliwal, 1997), which has been successfully used recently in speech recognition (see, e.g., Graves *et al.*, 2013).

A BiRNN consists of forward and backward RNN's. The forward RNN $\overrightarrow{f}$ reads the input sequence as it is ordered (from x_1 to x_{T_x}) and calculates a sequence of *forward hidden states* $(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_{T_x})$. The backward RNN $\overleftarrow{f}$ reads the sequence in the reverse order (from x_{T_x} to x_1), resulting in a sequence of *backward hidden states* $(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_{T_x})$.

We obtain an annotation for each word x_j by concatenating the forward hidden state $\overrightarrow{h}_j$ and the backward one $\overleftarrow{h}_j$, i.e., $h_j = \left[\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top\right]^\top$. In this way, the annotation h_j contains the summaries of both the preceding words and the following words. Due to the tendency of RNNs to better represent recent inputs, the annotation h_j will be focused on the words around x_j. This sequence of annotations is used by the decoder and the alignment model later to compute the context vector (Eqs. (5)–(6)).

See Fig. 1 for the graphical illustration of the proposed model.

4 Experiment Settings

We evaluate the proposed approach on the task of English-to-French translation. We use the bilingual, parallel corpora provided by ACL WMT '14.[3] As a comparison, we also report the performance of an RNN Encoder–Decoder which was proposed recently by Cho *et al.* (2014a). We use the same training procedures and the same dataset for both models.[4]

4.1 Dataset

WMT '14 contains the following English-French parallel corpora: Europarl (61M words), news commentary (5.5M), UN (421M) and two crawled corpora of 90M and 272.5M words respectively, totaling 850M words. Following the procedure described in Cho *et al.* (2014a), we reduce the size of the combined corpus to have 348M words using the data selection method by Axelrod *et al.* (2011).[5] We do not use any monolingual data other than the mentioned parallel corpora, although it may be possible to use a much larger monolingual corpus to pretrain an encoder. We concatenate news-test-

[3] http://www.statmt.org/wmt14/translation-task.html

[4] Implementations are available at https://github.com/lisa-groundhog/GroundHog.

[5] Available online at http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/.

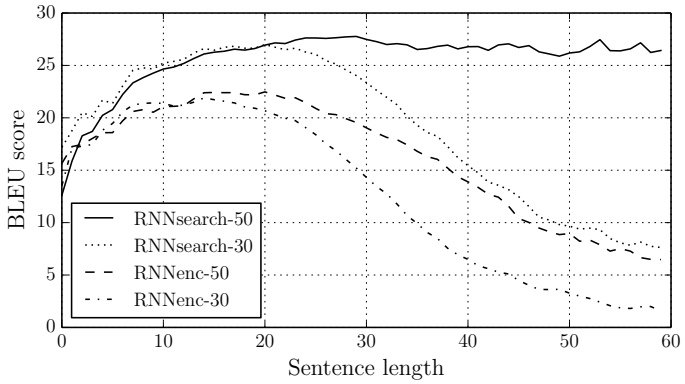

Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

2012 and news-test-2013 to make a development (validation) set, and evaluate the models on the test set (news-test-2014) from WMT '14, which consists of 3003 sentences not present in the training data.

After a usual tokenization[6], we use a shortlist of 30,000 most frequent words in each language to train our models. Any word not included in the shortlist is mapped to a special token ([UNK]). We do not apply any other special preprocessing, such as lowercasing or stemming, to the data.

4.2 Models

We train two types of models. The first one is an RNN Encoder–Decoder (RNNencdec, Cho *et al.*, 2014a), and the other is the proposed model, to which we refer as RNNsearch. We train each model twice: first with the sentences of length up to 30 words (RNNencdec-30, RNNsearch-30) and then with the sentences of length up to 50 word (RNNencdec-50, RNNsearch-50).

The encoder and decoder of the RNNencdec have 1000 hidden units each.[7] The encoder of the RNNsearch consists of forward and backward recurrent neural networks (RNN) each having 1000 hidden units. Its decoder has 1000 hidden units. In both cases, we use a multilayer network with a single maxout (Goodfellow *et al.*, 2013) hidden layer to compute the conditional probability of each target word (Pascanu *et al.*, 2014).

We use a minibatch stochastic gradient descent (SGD) algorithm together with Adadelta (Zeiler, 2012) to train each model. Each SGD update direction is computed using a minibatch of 80 sentences. We trained each model for approximately 5 days.

Once a model is trained, we use a beam search to find a translation that approximately maximizes the conditional probability (see, e.g., Graves, 2012; Boulanger-Lewandowski *et al.*, 2013). Sutskever *et al.* (2014) used this approach to generate translations from their neural machine translation model.

For more details on the architectures of the models and training procedure used in the experiments, see Appendices A and B.

5 Results

5.1 Quantitative Results

In Table 1, we list the translation performances measured in BLEU score. It is clear from the table that in all the cases, the proposed RNNsearch outperforms the conventional RNNencdec. More importantly, the performance of the RNNsearch is as high as that of the conventional phrase-based translation system (Moses), when only the sentences consisting of known words are considered. This is a significant achievement, considering that Moses uses a separate monolingual corpus (418M words) in addition to the parallel corpora we used to train the RNNsearch and RNNencdec.

[6] We used the tokenization script from the open-source machine translation package, Moses.

[7] In this paper, by a 'hidden unit', we always mean the gated hidden unit (see Appendix A.1.1).

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.