Supplementary Material
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho[1], Bart van Merriënboer[1], Caglar Gulcehre[1], Dzmitry Bahdanau[2], Fethi Bougares[3], Holger Schwenk[3], and Yoshua Bengio[1]

[1]Université de Montréal, Canada
[2]Jacobs University, Germany
[3]Université du Maine, France

1 RNN Encoder–Decoder

In this document, we describe in detail the architecture of the RNN Encoder–Decoder used in the experiments.

Let us denote an source phrase by $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ and a target phrase by $Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$. Each phrase is a sequence of K-dimensional one-hot vectors, such that only one element of the vector is 1 and all the others are 0. The index of the active (1) element indicates the word represented by the vector.

1.1 Encoder

Each word of the source phrase is embedded in a 500-dimensional vector space: $e(\mathbf{x}_i) \in \mathbb{R}^{500}$. $e(\mathbf{x})$ is used in Sec. 4.4 to visualize the words.

The hidden state of an encoder consists of 1000 hidden units, and each one of them at time t is computed by

$$h_j^{\langle t \rangle} = z_j h_j^{\langle t-1 \rangle} + (1 - z_j)\tilde{h}_j^{\langle t \rangle},$$

where

$$\begin{aligned}
\tilde{h}_j^{\langle t \rangle} =& \tanh\left(\left[\mathbf{W} e(\mathbf{x}_t)\right]_j + r_j \left[\mathbf{U}\mathbf{h}_{\langle t-1 \rangle}\right] \right), \\
z_j =& \sigma\left(\left[\mathbf{W}_z e(\mathbf{x}_t)\right]_j + \left[\mathbf{U}_z \mathbf{h}_{\langle t-1 \rangle}\right]_j \right), \\
r_j =& \sigma\left(\left[\mathbf{W}_r e(\mathbf{x}_t)\right]_j + \left[\mathbf{U}_r \mathbf{h}_{\langle t-1 \rangle}\right]_j \right).
\end{aligned}$$

σ is a logistic sigmoid function. To make the equations uncluttered, we omit biases. The initial hidden state $h_j^{\langle 0 \rangle}$ is fixed to 0.

Once the hidden state at the N step (the end of the source phrase) is computed, the representation of the source phrase $\mathbf{c}$ is

$$\mathbf{c} = \tanh\left(\mathbf{V}\mathbf{h}^{\langle N \rangle} \right).$$

1.1.1 Decoder

The decoder starts by initializing the hidden state with

$$\mathbf{h}'^{\langle 0 \rangle} = \tanh\left(\mathbf{V}'\mathbf{c} \right),$$

where we will use $\cdot'$ to distinguish parameters of the decoder from those of the encoder.

The hidden state at time t of the decoder is computed by

$$h'^{\langle t \rangle}_j = z'_j h'^{\langle t-1 \rangle}_j + (1 - z'_j)\tilde{h}'^{\langle t \rangle}_j,$$

where

$$\begin{aligned}
\tilde{h}'^{\langle t \rangle}_j &= \tanh\left([\mathbf{W}' e(\mathbf{y}_{t-1})]_j + r'_j \left[\mathbf{U}' \mathbf{h}'_{\langle t-1 \rangle} + \mathbf{C}\mathbf{c} \right] \right), \\
z'_j &= \sigma\left([\mathbf{W}'_z e(\mathbf{y}_{t-1})]_j + \left[\mathbf{U}'_z \mathbf{h}'_{\langle t-1 \rangle} \right]_j + [\mathbf{C}_z \mathbf{c}]_j \right), \\
r'_j &= \sigma\left([\mathbf{W}'_r e(\mathbf{y}_{t-1})]_j + \left[\mathbf{U}'_r \mathbf{h}'_{\langle t-1 \rangle} \right]_j + [\mathbf{C}_r \mathbf{c}]_j \right),
\end{aligned}$$

and $e(\mathbf{y}_0)$ is an all-zero vector. Similarly to the case of the encoder, $e(\mathbf{y})$ is an embedding of a target word.

Unlike the encoder which simply encodes the source phrase, the decoder is learned to generate a target phrase. At each time t, the decoder computes the probability of generating j-th word by

$$p(y_{t,j} = 1 \mid \mathbf{y}_{t-1}, \ldots, \mathbf{y}_1, X) = \frac{\exp\left(\mathbf{g}_j \mathbf{s}_{\langle t \rangle}\right)}{\sum_{j'=1}^{K} \exp\left(\mathbf{g}_{j'} \mathbf{s}_{\langle t \rangle}\right)},$$

where the i-element of $\mathbf{s}_{\langle t \rangle}$ is

$$s_i^{\langle t \rangle} = \max\left\{ s'^{\langle t \rangle}_{2i-1}, s'^{\langle t \rangle}_{2i} \right\}$$

and

$$\mathbf{s}'^{\langle t \rangle} = \mathbf{O}_h \mathbf{h}'^{\langle t \rangle} + \mathbf{O}_y \mathbf{y}_{t-1} + \mathbf{O}_c \mathbf{c}.$$

In short, the $s_i^{\langle t \rangle}$ is a so-called *maxout* unit.

For the computational efficiency, instead of a single-matrix output weight $\mathbf{G}$, we use a product of two matrices such that

$$\mathbf{G} = \mathbf{G}_l \mathbf{G}_r,$$

where $\mathbf{G}_l \in \mathbb{R}^{K \times 500}$ and $\mathbf{G}_r \in \mathbb{R}^{500 \times 1000}$.

2 Word and Phrase Representations

Here, we show enlarged plots of the word and phrase representations in Figs. 4–5.

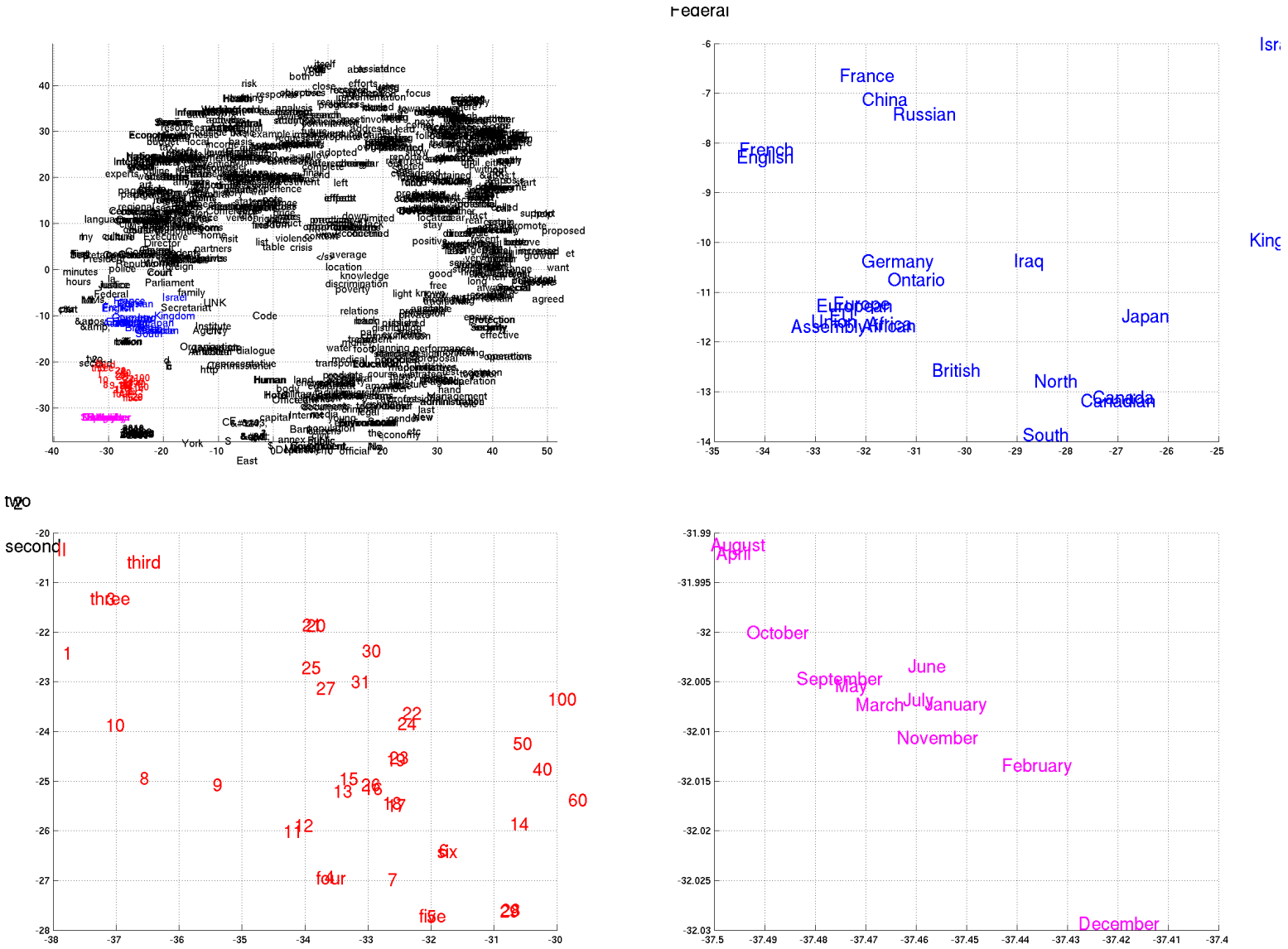

Figure 1: 2–D embedding of the learned word representation. The top left one shows the full embedding space, while the other three figures show the zoomed-in view of specific regions (color–coded).

Figure 2: 2–D embedding of the learned phrase representation. The top left one shows the full representation space (1000 randomly selected points), while the other three figures show the zoomed-in view of specific regions (color–coded).