Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

M Saiful Bari

Research Assistant
School of Computer Science and Engineering
Nanyang Technological University

March 12, 2018

Outline

Recap: Recurrent Neural Networks

- for a seq. , $\mathbf{x} = (x_1, x_2, ..., x_T)$ at time step t, the hidden state is

$$h_{<t>} = f(h_{<t-1>}, x_t)$$

Recap: Recurrent Neural Networks

- for a seq. , $\mathbf{x} = (x_1, x_2, ..., x_T)$ at time step t, the hidden state is

$$h_{<t>} = f(h_{<t-1>}, x_t)$$

- here f is non-linear activation function.

Recap: Recurrent Neural Networks

- for a seq. , $\mathbf{x} = (x_1, x_2, ..., x_T)$ at time step t, the hidden state is

$$h_{<t>} = f(h_{<t-1>}, x_t)$$

- here f is non-linear activation function.
- f may be a logistic sigmoid function or a long short-term memory (LSTM)

Recap: Recurrent Neural Networks

- for a seq. , $\mathbf{x} = (x_1, x_2, ..., x_T)$ at time step t, the hidden state is

$$h_{<t>} = f(h_{<t-1>}, x_t)$$

- here f is non-linear activation function.
- f may be a logistic sigmoid function or a long short-term memory (LSTM)
- The output at each timestep t is the conditional distribution $p(x_t|x_{t-1}, ..., x_1)$. for softmax activation function $p(x_{t,j} = 1|x_{t-1}, ..., x_1) = \frac{exp(w_j h_{<t>})}{\sum_{j'=1}^{K} exp(w_j{}' h_{<t>})}$

Recap: Recurrent Neural Networks

- for a seq. , $\mathbf{x} = (x_1, x_2, ..., x_T)$ at time step t, the hidden state is

$$h_{<t>} = f(h_{<t-1>}, x_t)$$

- here f is non-linear activation function.
- f may be a logistic sigmoid function or a long short-term memory (LSTM)
- The output at each timestep t is the conditional distribution $p(x_t|x_{t-1}, ..., x_1)$. for softmax activation function $p(x_{t,j} = 1|x_{t-1}, ..., x_1) = \frac{exp(w_j h_{<t>})}{\sum_{j'=1}^{K} exp(w_j{}' h_{<t>})}$
- $$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, ..., x_1)$$

Recap: Recurrent Neural Networks

- for a seq. , $\mathbf{x} = (x_1, x_2, ..., x_T)$ at time step t, the hidden state is

$$h_{<t>} = f(h_{<t-1>}, x_t)$$

- here f is non-linear activation function.
- f may be a logistic sigmoid function or a long short-term memory (LSTM)
- The output at each timestep t is the conditional distribution $p(x_t|x_{t-1}, ..., x_1)$. for softmax activation function $p(x_{t,j} = 1|x_{t-1}, ..., x_1) = \frac{exp(w_j h_{<t>})}{\sum_{j'=1}^{K} exp(w_j{}' h_{<t>})}$
- $$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, ..., x_1)$$

Recap: Recurrent Neural Networks

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_{t-1}, ..., x_1)$$

After learning this distribution

How to sample a new sequence?

iteratively sample a symbol at each time step.

Recap: Recurrent Neural Networks

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_{t-1}, ..., x_1)$$

After learning this distribution

How to sample a new sequence?

iteratively sample a symbol at each time step.

Outline

Model

- **2 RNN**

Model

- **2 RNN**

1. RNN as Encoder

encodes a variable length seq. of symbols into a **fixed length** vector representation.

Model

- **2 RNN**

1. RNN as Encoder

encodes a variable length seq. of symbols into a **fixed length** vector representation.

2. RNN as Decoder

decodes the **fixed length** representation into another variable length seq. of symbols

Model

- **2 RNN**

1. RNN as Encoder

encodes a variable length seq. of symbols into a **fixed length** vector representation.

2. RNN as Decoder

decodes the **fixed length** representation into another variable length seq. of symbols

The encoder and decoder are trained **jointly**.

Probabilistic Description

from probabilistic perspective the model computes,

$$p(y_1, ..., y_{T'} | x_1, ..., x_T)$$

Probabilistic Description

from probabilistic perspective the model computes,

$$p(y_1, ..., y_{T'} | x_1, ..., x_T)$$

general method to learn the **conditional distribution** over a **variable-length** sequence conditioned on yet another **variable-length** sequence

I/0 Structure

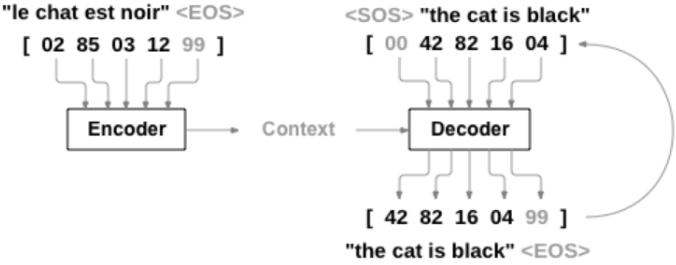


Figure: $< EOS >$ & $< SOS >$ additional tags are added with sentences.

RNN-Encoder

- The encoder reads each symbol of an input sequence x sequentially.
- After reading $< EOS >$, the hidden state of the RNN is a summary c of the whole input sequence.
- This c is used as a context vector for Decoder

RNN-Decoder

- RNN-Decoder is not similar to RNN-Encoder.

RNN-Decoder

- RNN-Decoder is not similar to RNN-Encoder.
-
$$h_{<t>} = f(h_{<t-1>}, y_{t-1}, c)$$
$h_{<t>}$ is conditioned on y_{t-1} and contex vector c.

RNN-Decoder

- RNN-Decoder is not similar to RNN-Encoder.
- $$h_{<t>} = f(h_{<t-1>}, y_{t-1}, c)$$
  $h_{<t>}$ is conditioned on y_{t-1} and contex vector c.
- the **conditional distribution** of the next symbol is,

$$p(y_t | y_{t-1}, y_{t-2}, ..., y_1, c) = g(h_{<t-1>}, y_{t-1}, c)$$

Diagram

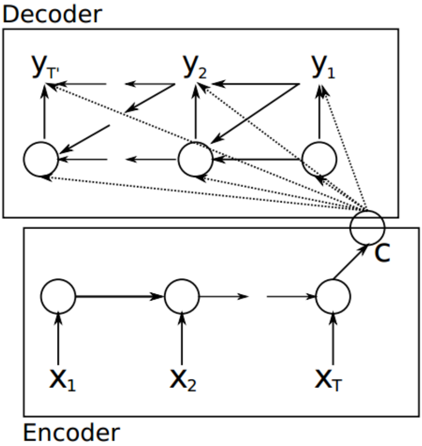


Figure: An illustration of the proposed RNN Encoder-Decoder

Loss Function

Encoder Decoder is trained **jointly** to maximize the **conditional log-likelihood**.

$$max_\theta \frac{1}{N}\sum_{i=1}^{N}\log p_\theta(y_n|\mathbf{x}_n)$$

where,

- θ is model parameter
- x_n is input sequence
- y_n is output sequence

Decoder

Slightly changed decoder structure,

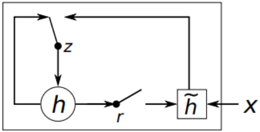
$$h'^{\langle t\rangle}_j = z'_j h'^{\langle t-1\rangle}_j + (1 - z'_j)\tilde{h}'^{\langle t\rangle}_j,$$

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

Cell type

- Model uses GRU Cell.
- The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$.
- The reset/forget gate r decides whether the previous hidden state is ignored.

$$
\begin{aligned}
r_j &= \sigma([W_r x]_j + [U h_{<t-1>}]_j) \\
z_j &= \sigma([W_z \mathbf{x}]_j + [U_z h_{<t-1>}]_j) \\
h^{<t>} &= z_j h_j^{<t-1>} + (1 - z_j)\tilde{h}_j^{<t-1>} \\
\tilde{h}_j &= \phi([W x]_j + [U(r \odot h_{<t-1>})]_j)
\end{aligned}
$$



Usages

- Model can be used in two ways.
  - generate a target sequence
  - to score a given pair of input and output seq. $p_\theta(y|\mathbf{x})$

Outline

Statistical Machine Translation

For SMT,

$$p(\mathbf{f}|\mathbf{e}) \propto p(\mathbf{e}|\mathbf{f})p(\mathbf{f})$$

where,

- $\mathbf{f}$ is translation

Statistical Machine Translation

For SMT,

$$p(\mathbf{f}|\mathbf{e}) \propto p(\mathbf{e}|\mathbf{f})p(\mathbf{f})$$

where,

- **f** is translation
- **e** is source

Statistical Machine Translation

For SMT,

$$p(\mathbf{f}|\mathbf{e}) \propto p(\mathbf{e}|\mathbf{f})p(\mathbf{f})$$

where,

- $\mathbf{f}$ is translation
- $\mathbf{e}$ is source
- $p(\mathbf{e}|\mathbf{f})$ is called **translation model**.

Statistical Machine Translation

For SMT,

$$p(\mathbf{f}|\mathbf{e}) \propto p(\mathbf{e}|\mathbf{f})p(\mathbf{f})$$

where,

- **f** is translation
- **e** is source
- $p(\mathbf{e}|\mathbf{f})$ is called **translation model**.
- $p(\mathbf{f})$ is **language model**

Statistical Machine Translation

$$p(\mathbf{f}|\mathbf{e}) \propto p(\mathbf{e}|\mathbf{f})p(\mathbf{f})$$

In practice, however, most SMT systems model $p(\mathbf{f}|\mathbf{e})$ as a *log* linear model with additional features and corresponding weights

$$\log p(\mathbf{f}|\mathbf{e}) = \sum_{i=1}^{N} w_n f_n(\mathbf{f}, \mathbf{e}) + \log Z(\mathbf{e})$$

where,

- f_n and w_n are the n-th feature and weight
- $Z(e)$ is a normalization constant that
- The weights are often optimized to maximize the BLEU score on a dev. set.

Scoring Phrase Pairs

The proposed RNN encoderdecoder model has trained on a table of phrase pairs.

Scoring Phrase Pairs

The proposed RNN encoderdecoder model has trained on a table of phrase pairs.

Use its scores as additional features in the log-linear model when tuning the SMT decode.

Scoring Phrase Pairs

The proposed RNN encoderdecoder model has trained on a table of phrase pairs.

Use its scores as additional features in the log-linear model when tuning the SMT decode.

Model is trained without frequency information of each phrase pair.

Scoring Phrase Pairs

The proposed RNN encoderdecoder model has trained on a table of phrase pairs.

Use its scores as additional features in the log-linear model when tuning the SMT decode.

Model is trained without frequency information of each phrase pair.

Objective of the model is to capture **linguistic information** rather than **statistical information**.

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.
- The computation from the hidden state in the decoder to the output is implemented as a deep neural network with a single intermediate layer having 500 maxout units each pooling 2 inputs.

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.
- The computation from the hidden state in the decoder to the output is implemented as a deep neural network with a single intermediate layer having 500 maxout units each pooling 2 inputs.
- Non-recurrent weights initialized by sampling from an **isotropic Gaussian distribution** (mean $= 0$, sd $= 0.01$)

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.
- The computation from the hidden state in the decoder to the output is implemented as a deep neural network with a single intermediate layer having 500 maxout units each pooling 2 inputs.
- Non-recurrent weights initialized by sampling from an **isotropic Gaussian distribution** (mean $= 0$, sd $= 0.01$)
- Recurrent weights initialized by sampling from **white Gaussian distribution** and using its left singular vectors.

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.
- The computation from the hidden state in the decoder to the output is implemented as a deep neural network with a single intermediate layer having 500 maxout units each pooling 2 inputs.
- Non-recurrent weights initialized by sampling from an **isotropic Gaussian distribution** (mean = 0, sd = 0.01)
- Recurrent weights initialized by sampling from **white Gaussian distribution** and using its left singular vectors.
- Learning algorithm : $Adadelta(\eta = 10^{-6}, \rho = .95)$

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.
- The computation from the hidden state in the decoder to the output is implemented as a deep neural network with a single intermediate layer having 500 maxout units each pooling 2 inputs.
- Non-recurrent weights initialized by sampling from an **isotropic Gaussian distribution** (mean = 0, sd = 0.01)
- Recurrent weights initialized by sampling from **white Gaussian distribution** and using its left singular vectors.
- Learning algorithm : $Adadelta(\eta = 10^{-6}, \rho = .95)$
- Activation function : $tanh$

Experiments

- Evaluated the approach on the English/French translation task of the WMT14.
- embeddings dimension 100.
- The computation from the hidden state in the decoder to the output is implemented as a deep neural network with a single intermediate layer having 500 maxout units each pooling 2 inputs.
- Non-recurrent weights initialized by sampling from an **isotropic Gaussian distribution** (mean = 0, sd = 0.01)
- Recurrent weights initialized by sampling from **white Gaussian distribution** and using its left singular vectors.
- Learning algorithm : $Adadelta(\eta = 10^{-6}, \rho = .95)$
- Activation function : $tanh$
- 384M words, 3 days training time

Quantitative Analysis

Models	BLEU	
	dev	test
Baseline	30.64	33.30
RNN	31.20	33.87
CSLM + RNN	31.48	34.64
CSLM + RNN + WP	31.50	34.54

Figure: BLEU scores computed on the development and test sets using different combinations of approaches.)

*CSLM-Continuous Space Language Model (Schwenk, 2007)

Quantitative Analysis

adding features computed by neural networks consistently improves the performance over the baseline performance.

Quantitative Analysis

adding features computed by neural networks consistently improves the performance over the baseline performance.

contributions of the CSLM and the RNN EncoderDecoder are not too correlated

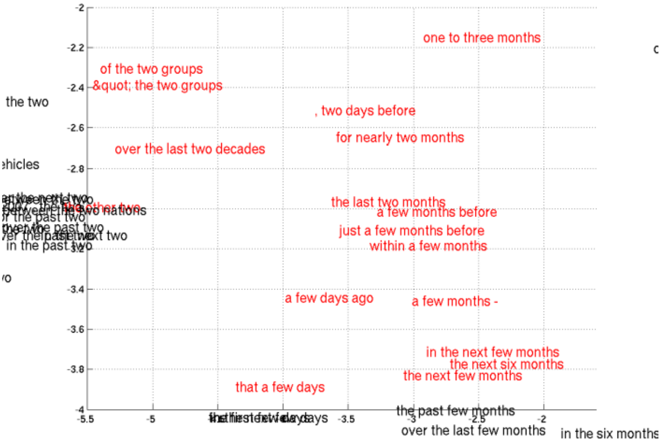
Quantitative Analysis

adding features computed by neural networks consistently improves the performance over the baseline performance.

contributions of the CSLM and the RNN EncoderDecoder are not too correlated

WP - Word Penalty. (penalizing the number of words that are unknown to the neural networks)

Qualitative Analysis



Qualitative Analysis

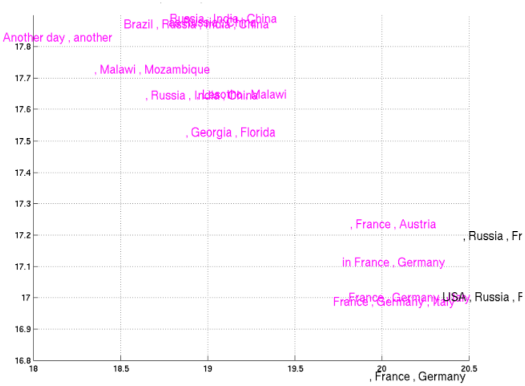


Figure: 2D embedding of the learned phrase representation

Qualitative Analysis

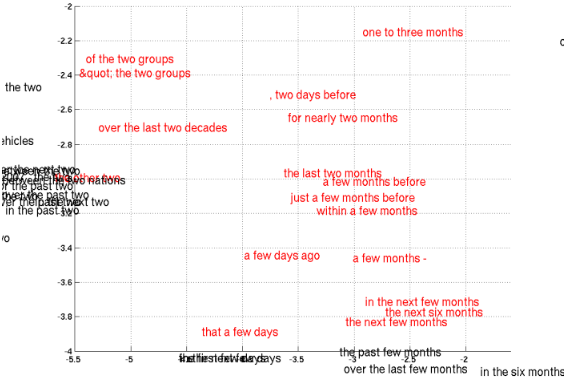


Figure: 2D embedding of the learned phrase representation

Outline

Summary

RNN EncoderDecoder model captures linguistic features.

RNN Encoder-Decoder learns a continuous space representation for phrases that preserves both the semantic and syntactic structure.

Summary

RNN EncoderDecoder model captures linguistic features.

RNN Encoder-Decoder learns a continuous space representation for phrases that preserves both the semantic and syntactic structure.

Final words

sentence level encoding vector which can be decoded

Implementation Details

- Paper (After Acknowledgments)
- Code (library implementation)
- A Simple NMT model (Without Attention)