Paper Reading
on
Sequence to Sequence Learning with Neural Networks

Authors:

Ilya Sutskever Oriol Vinyals Quoc V. Le

Abstract:

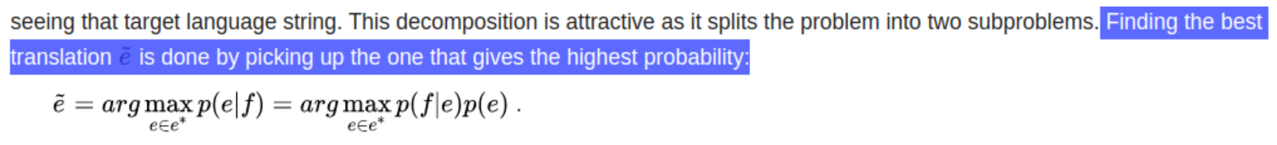
Used **a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector** of a fixed dimensionality, and then **another deep LSTM to decode the target sequence from the vector.**

1. On an English to French translation task from the **WMT-14 dataset**, the translations produced by the LSTM (5 Deep LSTMs with 380M parameters each) achieve a BLEU score of **34.8**

2. LSTM's **BLEU (Bilingual Evaluation Understudy)** score was penalized on out-of-vocabulary words (vocabulary had a collection of 80K words).

3. A phrase-based SMT system achieves a BLEU score of **33.3** on the same dataset.

SMT System:
Statistical Machine Translation:



seeing that target language string. This decomposition is attractive as it splits the problem into two subproblems. Finding the best translation $\tilde{e}$ is done by picking up the one that gives the highest probability:

$$\tilde{e} = arg\max_{e \in e^*} p(e|f) = arg\max_{e \in e^*} p(f|e)p(e) \; .$$

4. When LSTM was used to rerank the **1000 hypotheses** produced by the aforementioned SMT system, its BLEU score increases to **36.5**

5. Also, reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly

Introduction:

DNN can only be applied to problems whose inputs and outputs can be represented with a vector of fixed Dimensionality.
So, the paper is all about showing that the LSTM can be used to solve the sequence to sequence problem.
The idea is to use **one LSTM to read the input sequence, one timestep at a time, to obtain large fixed- dimensional vector representation**, and then to use **another LSTM to extract the output sequence from that vector**

Choice of using LSTM because it can work on Vanishing Gradient Problems.

LSTM did not suffer on very Long sentences with **a simple trick of reversing the words** in the source sentences which is one of the key technical contribution of the work

On evaluation it showed that the model is aware of word order and is fairly invariant to the active and passive voice.

Model:

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

The LSTM computes this conditional probability by **first obtaining the fixed-dimensional representation v of the input sequence (x 1 , . . . , x T)** given by the last hidden state of the LSTM, and **then computing the probability of y 1 , . . . , y T ′ with a standard LSTM-LM formulation** whose initial hidden state is set to the representation v of x 1 , . . . , x T in the above Snap.

DataSet Details:

WMT’14 English to French dataset
- 12M sentences consisting of 348M French words and 304M English words
- Used 160,000 of the most frequent words for the source language and 80,000 of the most frequent words for the target language
- Out of vocabulary words was replaced with “UNK” token

Decoding and Rescoring

The core of our experiments involved training a large deep LSTM on many sentence pairs. We trained it by maximizing the log probability of a correct translation T given the source sentence S, so the training objective is

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

where $\mathcal{S}$ is the training set. Once training is complete, we produce translations by finding the most likely translation according to the LSTM:

$$\hat{T} = \arg\max_{T} p(T|S) \qquad (2)$$

LSTM was also **used to rescore the 1000-best lists** produced by the baseline system.

To rescore an n-best list, log probability of every hypothesis was computed with the LSTM and an average was taken with their and the LSTM's score

Reversing the Source Sentences

Since LSTM is already capable of solving the problems with long term dependencies and works even on vanishing gradient problems, to have a much better accuracy the sentences on the source were reversed keeping the sentences on the target the same following result was obtained:
- The LSTM's test perplexity dropped from 5.8 to 4.7
- The test BLEU scores of its decoded translations increased from 25.9 to 30.6

The reason for improvement might have been introduction of many short term dependencies among the words on the dataset.

Training Details

- Deep LSTMs with 4 layers with 1000 cells at each layer and 1000 dimensional word embeddings
- Input Vocabulary of 160,000 words and output vocabulary of 80,000 words
- Naive Softmax over each outputs

Overally, LSTM had **380M parameters** out of which **64M parameters are pure recurrent connections** (32M for encoder and 32M for decoder LSTM)

Training Architecure Details:

- All of the LSTM's parameters were initialized with the uniform distribution between -0.08 and 0.08

- Stochastic gradient descent without momentum was used with a fixed learning rate of 0.7
After 5 epochs, halving the learning rate every half epoch was done. The model was trained for a total of 7.5 epochs.

- Batches of 128 sequences for the gradient and divided it the size of the batch (namely, 128).

- Although LSTMs tend to not suffer from the vanishing gradient problem, they can have exploding gradients. Thus a hard constraint was enforced on the norm of the gradient [10, 25] by scaling it when its norm exceeded a threshold. For each training batch, we compute $s = ||g||2$, where g is the gradient divided by 128. If $s > 5$, we set $g = 5g/s$.

- Different sentences have different lengths. Most sentences are short (e.g., length 20-30) but some sentences are long (e.g., length > 100), so a minibatch of 128 randomly chosen training sentences will have many short sentences and few long sentences, and as a result, much of the computation in the minibatch is wasted. To address this problem, it was made sure that all sentences within a minibatch were roughly of the same length, which a 2x speedup.

3.8 Model Analysis

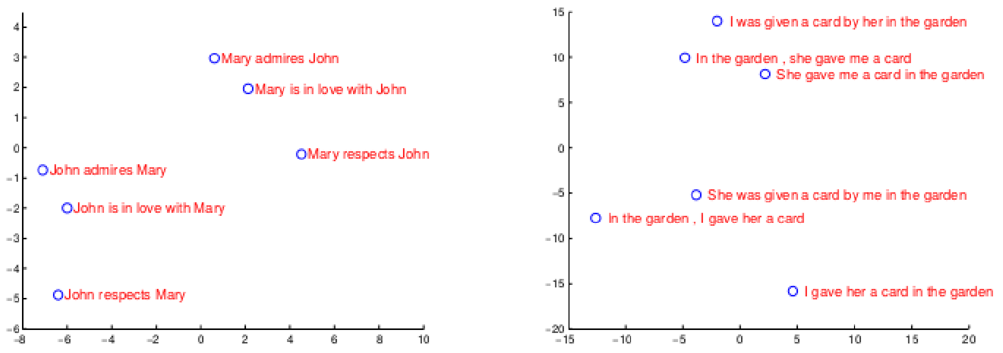


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

The model was able to perform well on both the short and Long Sentences. Which was seen through the PCA projection.