Research Paper Summary/Notes

Sequence to Sequence Learning with Neural Networks, Google.

Laxman Singh Tomar

November 2019

Research Paper Summary/Notes

Sequence to Sequence Learning with Neural Networks, Google.

Laxman Singh Tomar

November 2019

Contents

Dedication

I would like to dedicate this work to teachers whom I've never met in real but their valuable contribution have made this work possible.

Acknowledgement

(I encountered several resources which helped me in getting through this work)

I would like to express my sincere gratitude to:

- Ilya Sutskever et. al.
- Jay Alammar
- Adrian Coyler

Abstract

Deep Neural Nets work astoundingly well when provided with large labeled training sets; but seem to perform poorly when it comes to mapping input sequences to output ones. This paper proposes a general and end-to-end approach for sequence learning that uses two deep LSTMs, one to map input sequence to vector space and another to map vector to the output sequence.

Keywords: DNNs; RNNs; LSTMs

Chapter 1

Introduction

DNNs are extremely powerful model architectures due to their ability of performing arbitrary parallel number of computations with modal number of steps. For example, sorting N N bit numbers. If there exists a parameter setting for a DNN, supervised backpropagation can be implemented to achieve good results.

But Deep Neural Networks (DNNs) require the dimensionality of input and output sequences to be known and fixed. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priory. For example: Speech Recognition and Question Answering.

Here we're proposing a LSTM based architecture which solves the general sequence to sequence problems. 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. The second LSTM is essentially a recurrent neural network language model except that it is conditioned on the input sequence. The LSTM's ability to successfully learn on data with long range temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and their corresponding outputs.

When feeding the input sentence into the encoding LSTM, we discovered that the

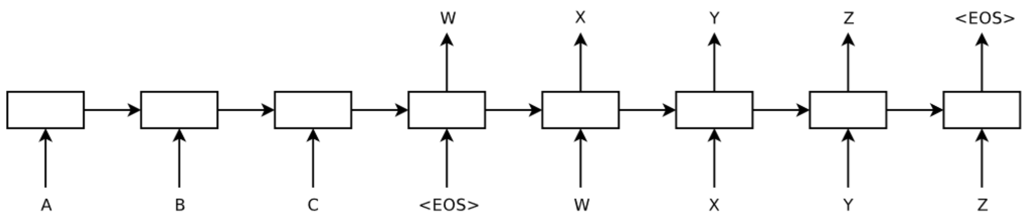

Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

end-to-end process works much better if the sentence is input in reverse word order.

1.1 Motivation

DNNs though being powerful architectures didn't perform well when it comes to sequence to sequence problems.

1.2 Objectives

Introduce an architecture that can be used to solve the general sequence to sequence problems.

Chapter 2

Methods

2.1 Models

Recurrent Neural Networks (RNNs) generalizes feed forward neural networks to sequences. Given a sequence of inputs $(x_1, x_2 ... x_T)$, RNN computes a sequence of outputs $(y_1, y_2 ... y_T)$ by iterating over the following equation:

$$h_t = sigm(W^{hx} x_t + W^{hh} h_{t-1})$$

$$y_t = W^{yh} h_t$$

RNNs can be used in sequence to sequence problems if alignment of input and output sequence is known beforehand, but there's no specific solution to when it comes to sequences with variable lengths. To map variable length sequences, the input is mapped to a fixed size vector using an RNN and this fixed size vector is mapped to output sequence using another RNN.

Given the long-term dependencies between the two sequences, LSTMs are preferred over RNNs. LSTMs estimate the conditional probability p(output sequence | input sequence) i.e. $p(y_1, ..., y_{T'} | x_1, ..., x_T)$ by first mapping the input sequence $(x_1, ..., x_T)$ to a fixed dimensional representation v given by last hidden state of LSTM and then computing the probability of output $y_1, ..., y_{T'}$ with a standard LST-LM formulation.

Notable thing here is that input and output sequences have variable length T and T' respectively.

2.1.1 Difference between Model and Standard LSTMs

1. The model uses two LSTMs (one for input sequence and another for output sequence), thereby increasing the number of model parameters at negligible computing cost.

2. Model uses Deep LSTMs (4 layers).

3. The words in the input sequences are reversed to introduce short-term dependencies and to reduce the "minimal time lag". By reversing the word order, the first few words in the source sentence (input sentence) are much closer to first few words in the target sentence (output sentence) thereby making it easier for LSTM to "establish" communication between input and output sentences.

Chapter 3

Results

3.1 Datset Details

We used WMT'14 English to French dataset containing 12 million sentences consisting of 348 million French words and 304 million English words.

3.2 Decoding and Rescoring

Our Model was tested on the task of re-scoring the n-best results of baseline approach. Deep LSTMs were trained on many sentence pairs by maximizing the log probability of a correct translation T, given the source sentence S, thus training objective is to maximize this log probability, averaged over all the pairs in the training set:

$$1/|S| \sum_{(T,S)\in S} logp(T|S)$$

where S is the training set. Once training is complete, most likely translation is found by performing a simple, left-to-right beam search for translation:

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

3.3 Reversing the Source Sentences

While we do not have a complete explanation to this phenomenon, we believe that it is caused by the introduction of many short term dependencies to the dataset. Normally, when we concatenate a source sentence with a target sentence, each word in the source sentence is far from its corresponding word in the target sentence. As a result, the problem has a large "minimal time lag". By reversing the words in the source sentence, the average distance between corresponding words in the source and target language is unchanged. However, the first few words in the source language are now very close to the first few words in the target language, so the problem's minimal time lag is greatly reduced. Thus, backpropagation has an easier time "establishing communication" between the source sentence and the target sentence, which in turn results in substantially improved overall performance.

The beneficial effects apply not just to the early parts of the target sentence, but also to long sentences.

3.4 Training Details

We built deep LSTMs with 4 layers of 1000 cells, and 1000-dimensional word embeddings, an input vocabulary of 160,000 words, and an output vocabularly of 80,000 words. "Thus the deep LSTM uses 8000 real numbers to represent a sentence." Deep LSTMs outperformed shallow ones, reducing 'perplexity' by nearly 10% with each additional layer. A naive softmax over the 80,000 words is used at each output. "The resulting LSTM has 384M parameters, of which 64M are pure recurrent connections."

A hard constraint is enforced on the norm of the gradient to avoid the exploding gradient problem as LSTMs don't suffer from vanishing gradient problem. Mini batches are selected to have sentences of similar lengths to reduce training time.

3.5 Parallelization

For training, each layer resides on a separate GPU, with another 4 GPUs used to parallelize the softmax. This configuration achieved a speed of 6,300 words per second and training took about 10 days.

3.6 Experimental Results

The model was trained on the WMT '14 English to French translation task and evaluated using the BLEU algorithm to produce a BLEU score(34.81). BLEU achieves high correlation with human judgements of quality. While the model does not beat the state-of-the-art(best overall WMT'14 result was 37.0, using a method that combines neural networks and an SMT system), it is the first pure neural translation system to outperform a phrase-based SMT baseline(33.30).

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Method	**test BLEU score (ntst14)**
Bahdanau et al.	28.45
Baseline System	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

3.7 Performance on Long Sentences

The model performs well on long sentences as well with only a minor degradation for the largest sentences.

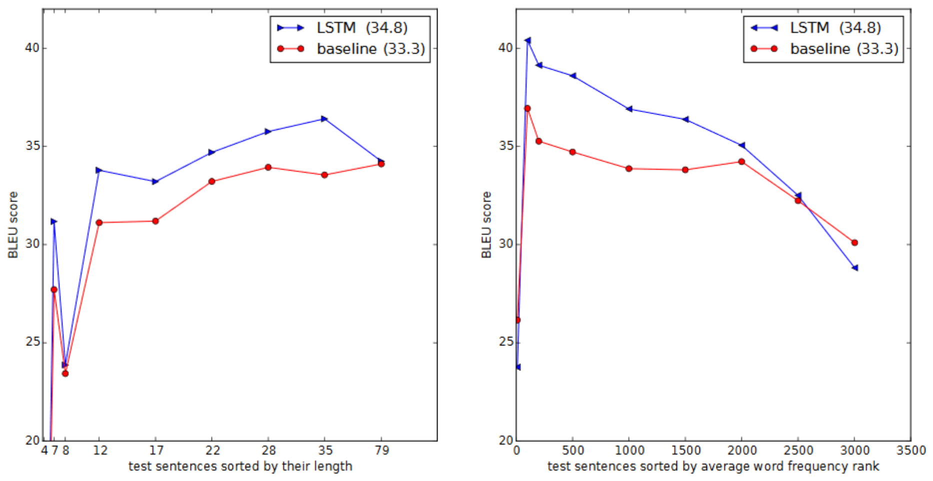

Figure 2: The left plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length and is marked by the actual sequence lengths.There is no degradation on sentences with less than 35 words, there is only a minor degradation on the longest sentences. The right plot shows the LSTM's performance on sentences with progressively more rare words,where the x-axis corresponds to the test sentences sorted by their "average word frequency rank".

3.8 Model Analysis

One of the attractive features of our model is its ability to turn a sequence of words into a vector of fixed dimensionality. A PCA projection of the LSTM hidden states shows that phrases are indeed clustered by meaning:

3.9 Related Work

Our work is closely related to Kalchbrenner and Blunsom[1], who were the first to map the input sentence into a vector and then back to a sentence, although they map sentences to vectors using convolutional neural networks, which lose the ordering of the words.

Similarly to this work, Cho et. al.[2] used an LSTM-like RNN architecture to map sentences into vectors and back, although their primary focus was on integrating their neural network into an SMT system. Bahdanau et al.[3] also attempted direct

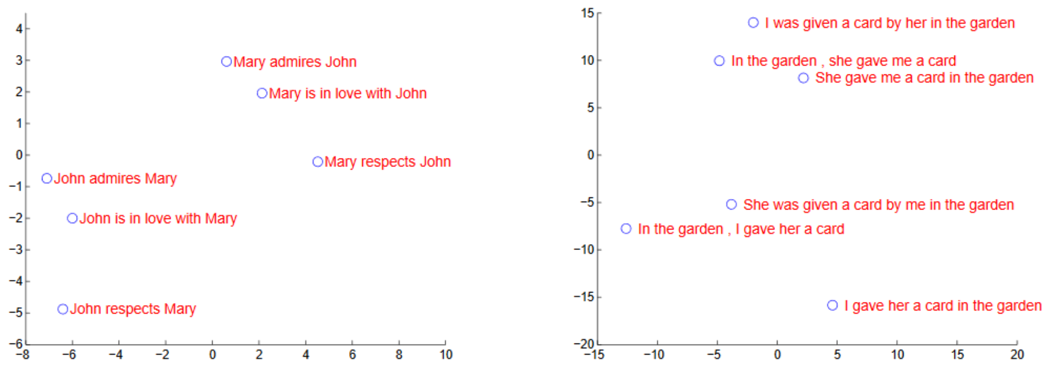

Figure 3: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

translations with a neural network that used an attention mechanism to overcome the poor performance on long sentences experienced by Cho et. al. [5] and achieved encouraging results.

Chapter 4

Discussion

4.1 Discussion

Some of the results we didn't anticipate:

1. Improvement obtained by reversing the order of words in a sentence due to the complexity being largely dependent on number of short-term dependencies.

2. Broke our assumption that due to limited memory, LSTM woud fail on large sentences and performed quite well when trained over reversed dataset.

3. A large deep LSTM with a limited vocabulary can outperform a standard SMT-based system whose vocabulary is unlimited on a large-scale MT task.

4.2 Conclusions

The paper prepares ground for the application of sequence-to-sequence based learning models in other domains by demonstrating how a simple and relatively unoptimised neural model could outperform a mature SMT system on translation tasks.

List of Figures

List of Tables

Bibliography

[1] Kalchbrenner, N. & Blunsom., P. Recurrent continuous translation models. *EMNLP* (2013).

[2] K. Cho, C. G. F. B. H. S., B. Merrienboer & Bengio, Y. Learning phrase represen-tations using rnn encoder-decoder for statistical machine translation. *Arxiv preprint arXiv:1406.1078* (2014).

[3] D. Bahdanau, K. C. & Bengio, Y. Neural machine translation by jointly learning to align and translate. *EMNLP* (2013).

Appendix A

Google's Seq2Seq Implementation

`https://github.com/tensorflow/nmt`

You can see the Tensorflow implementation tutorial on Neural Machine Translation(Seq2Seq).