Machine Translation using Deep Learning Methods

Max Fomin
Michael Zolotov

- Sequence to Sequence Learning with Neural Networks
Topics Ahead

01. Problem Definition
02. Network Architecture
03. Network Training
04. Results
History of Machine Translation

A few years ago

Recently

More recently
Problem Definition
Types of RNN Problems

- One to one: Regular CNN Model
- One to many: Image Captioning
- Many to one: Sentiment Analysis
- Many to many: Machine Translation
- Many to many: Video Classification
Limitations of current methods

Only fixed inputs!

Only problems whose inputs and targets can be encoded with fixed dimensionality.
**English to French Translation**

The WMT'14 English to French dataset was used. The models were trained on a subset of 12M sentences consisting of 348M French words and 304M English words.

**Vocabulary Filtering**

As typical neural language models rely on a vector representation for each word, we used a fixed vocabulary for both languages. We used 160,000 of the most frequent words for the source language and 80,000 of the most frequent words for the target language. Every out-of-vocabulary word was replaced with a special “UNK” token.
The BLEU Score

**Higher is Better**

More reference human translations $\rightarrow$ Better and more accurate scores

Scores over 30: Understandable translations

Scores over 50: Good and fluent translations

\[
p_n = \frac{\sum_{S \in C} \sum_{ngram \in S} Count_{matched}(ngram)}{\sum_{S \in C} \sum_{ngram \in S} Count(ngram)}
\]

Example of poor machine translation output with high precision

<table>
<thead>
<tr>
<th>Candidate</th>
<th>the</th>
<th>the</th>
<th>the</th>
<th>the</th>
<th>the</th>
<th>the</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference 1</td>
<td>the</td>
<td>cat</td>
<td>is</td>
<td>on</td>
<td>the</td>
<td>mat</td>
<td></td>
</tr>
<tr>
<td>Reference 2</td>
<td>there</td>
<td>is</td>
<td>a</td>
<td>cat</td>
<td>on</td>
<td>the</td>
<td>mat</td>
</tr>
</tbody>
</table>

Comparing metrics for candidate "the the cat"

<table>
<thead>
<tr>
<th>Model</th>
<th>Set of grams</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram</td>
<td>&quot;the&quot;, &quot;the&quot;, &quot;cat&quot;</td>
<td>$\frac{1 + 1 + 1}{3} = 1$</td>
</tr>
<tr>
<td>Bigram</td>
<td>&quot;the the&quot;, &quot;the cat&quot;</td>
<td>$\frac{0 + 1}{2} = \frac{1}{2}$</td>
</tr>
</tbody>
</table>
Some Background
“Classical” RNNs

Memory is a powerful tool!
Humans don’t start their thinking from scratch every second.

Sequential Data
A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. This chain-like nature makes them a natural architecture for sequential data.
“the clouds are in the sky,”

“I grew up in France… I speak fluent French.”
LSTMs

Long Short-Term Memory Networks
A special kind of RNN, capable of learning long-term dependencies.
They work tremendously well on a large variety of problems, and are now widely used.

Long-Term Dependencies
LSTMs are explicitly designed to avoid the long-term dependency problem.
Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!
LSTMs
LSTMs

The diagram illustrates the Long Short-Term Memory (LSTM) cell, which is a type of recurrent neural network (RNN) used in deep learning. LSTMs are designed to address the vanishing gradient problem in traditional RNNs by using three gates: the input gate ($i_t$), the forget gate ($f_t$), and the output gate ($o_t$). These gates control the flow of information by allowing the network to selectively retain or discard information from previous time steps. The LSTM cell also includes a cell state ($C_t$) and a hidden state ($h_t$) that evolve over time through the following equations:

$$C_t = f_t \odot C_{t-1} + i_t \odot \text{tanh}(W_x \odot x_t + W_h \odot h_{t-1})$$

$$h_t = o_t \odot \text{tanh}(C_t)$$

Where $x_t$ is the input at time step $t$, $W_x$ and $W_h$ are weight matrices, and $\sigma$ represents the sigmoid activation function.
LSTMs

1. \[ f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \]
2. \[ i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \]
   \[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]
3. \[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]
4. \[ o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \]
   \[ h_t = o_t \cdot \tanh(C_t) \]
A slightly more dramatic variation on the LSTM

It combines the forget and input gates into a single “update gate.” It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

\[
\begin{align*}
  z_t &= \sigma (W_z \cdot [h_{t-1}, x_t]) \\
  r_t &= \sigma (W_r \cdot [h_{t-1}, x_t]) \\
  \tilde{h}_t &= \tanh (W \cdot [r_t \cdot h_{t-1}, x_t]) \\
  h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t
\end{align*}
\]
Network Architecture
**High Level Architecture**

**Sequence Input**
The idea is to use one LSTM to read the input sequence, one time step at a time, to obtain large fixed dimensional vector representation.

**Sequence Output**
We 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.
Overall Process
Our method uses 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.
A Similar Concept: Word Embeddings
A Similar Concept: Word Embeddings

Male-Female

Verb tense

Country-Capital

- Spain
- Italy
- Germany
- Turkey
- Russia
- Canada
- Japan
- Vietnam
- China

- Madrid
- Rome
- Berlin
- Ankara
- Moscow
- Ottawa
- Tokyo
- Hanoi
- Beijing
A Similar Concept: Image Embeddings
A Similar Concept: Multiple Object Embeddings
A Classical Approach: Statistical Machine Translation

**Definition**
A machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual text corpora.

**Goal**
Finding a translation $f$, given a source sentence $e$, which maximizes the $p(f \mid e) \propto p(e \mid f) p(f)$.

**Phrase Based**
Creating translation probabilities of matching phrases in the source and target sentences in order to factorize $p(e \mid f)$. 
Network Training
we used two different LSTMs: one for the input sequence and another for the output sequence

Deep LSTMs significantly outperformed shallow LSTMs, so we chose an LSTM with four layers

It was extremely valuable to reverse the order of the words of the input sentence

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

where,

- $x_1, ..., x_T$ – the input sequence
- $y_1, ..., y_{T'}$ – the output sequence
Reversed Word Order!
Training Details

- 4 layers of LSTMs
- 1000 cells at each layer
- 1000 dimensional word embeddings
- An input vocabulary of 160,000
- An output vocabulary of 80,000
Training Details

- Each additional layer reduced perplexity by nearly 10%.
- We used a naive softmax over 80,000 words at each output.
- The resulting LSTM has 380M parameters of which 64M are pure recurrent connections (32M for the “encoder” LSTM and 32M for the “decoder” LSTM).
Training Details

- We initialized all of the LSTM’s parameters with the uniform distribution between -0.08 and 0.08.
- We used SGD without momentum, with a fixed learning rate of 0.7.
- After 5 epochs, we begun halving the learning rate every half epoch. We trained our models for a total of 7.5 epochs.
- We used batches of 128 sequences for the gradient and divided it the size of the batch.
- Thus we enforced a hard constraint on the norm of the gradient by scaling it when its norm exceeded a threshold.
- Different sentences have different lengths. Most sentences are short but some sentences are long. We made sure that all sentences within a mini-batch were roughly of the same length, resulting in a 2x speedup.
A C++ implementation of deep LSTM with the configuration from the previous section on a single GPU processes a speed of approximately 1,700 words per second.

We parallelized our model using an 8-GPU machine.

Each layer of the LSTM was executed on a different GPU and communicated its activations to the next GPU (or layer) as soon as they were computed.

The remaining 4 GPUs were used to parallelize the softmax, so each GPU was responsible for multiplying by a $1000 \times 20000$ matrix.

The resulting implementation achieved a speed of 6,300 (both English and French) words per second with a minibatch size of 128.

Training took about a ten days with this implementation.
Heuristic Search Algorithm
Explores a graph by expanding the most promising node in a limited set. Beam search is an optimization of best-first search that reduces its memory requirements.

Greedy Algorithm
Best-first search is a graph search which orders all partial solutions (states) according to some heuristic which attempts to predict how close a partial solution is to a complete solution. In beam search, only a predetermined number of best partial solutions are kept as candidates.
Beam-Search Decoder

We search for the most likely translation using a simple left-to-right beam search decoder.

We maintain a small number $B$ of partial hypotheses.

At each time step, we extend each partial hypothesis in the beam with every possible word.

we discard all but the $B$ most likely hypotheses according to the model's log probability.

As soon as the “ symbol is appended to a hypothesis, it is removed from the beam.

A beam of size 2 provides most of the benefits of beam search.
Results
## Some Tables

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bahdanau et al. [2]</td>
<td>28.45</td>
</tr>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Single forward LSTM, beam size 12</td>
<td>26.17</td>
</tr>
<tr>
<td>Single reversed LSTM, beam size 12</td>
<td>30.59</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 1</td>
<td>33.00</td>
</tr>
<tr>
<td>Ensemble of 2 reversed LSTMs, beam size 12</td>
<td>33.27</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 2</td>
<td>34.50</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 12</td>
<td><strong>34.81</strong></td>
</tr>
</tbody>
</table>
## Some Tables

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Cho et al. [5]</td>
<td>34.54</td>
</tr>
<tr>
<td>State of the art [9]</td>
<td><strong>37.0</strong></td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with a single forward LSTM</td>
<td>35.61</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with a single reversed LSTM</td>
<td>35.85</td>
</tr>
<tr>
<td>Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs</td>
<td><strong>36.5</strong></td>
</tr>
<tr>
<td>Oracle Rescoring of the Baseline 1000-best lists</td>
<td>~45</td>
</tr>
</tbody>
</table>
Some Plots
The figure shows a 2D PCA projection of the LSTM hidden states. Notice that both clusters have similar internal structure.
Conclusions

A large deep LSTM with a limited vocabulary can outperform a standard SMT-based system with an unlimited vocabulary.

The ability of the LSTM to correctly translate very long sentences was surprising.

Reversing the words in the source sentences gave surprising results.

A simple straightforward approach can outperform a mature SMT system.
Thank You!