Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation


Google

arXiv:1609.08144v2
Reviewed by: Bill Zhang
University of Virginia
https://qdata.github.io/deep2Read/
Outline

Introduction

Model Architecture

Segmentation

Training Criteria

Quantized Model

Decoder

Experiments & Results

Summary
Introduction
Basic Premise and Motivation

- Standard NMTs have slow training and inference speeds, are ineffective at dealing with rare words, and sometimes fail to translate all source words; GNMT aims to improve upon all these problems.
- GNMT is robust and works for a variety of language pairs and reduces error by 60% based on human evaluations.
Model Architecture
Overview

- Seq-to-seq model with attention
  - Encoder, decoder, and attention networks
- Define $X = x_1, ..., x_M$ and $Y = y_1, ..., y_N$ as the source and target sentences
  - Encoder: $x_1, ..., x_M = EncoderRNN(x_1, ..., x_M)$
  - $P(Y|X) = P(Y|x_1, ..., x_M)$
    $= \prod_{i=1}^{N} P(y_i|y_0, y_1, ..., y_{i-1}; x_1, ..., x_M)$
  - Decoder: RNN with softmax; RNN outputs a hidden state $y_i$ then generates a probability distribution using softmax
- Deep networks perform better, so both encoder and decoder have multiple layers
Model Architecture

Overview

- \( s_t = \text{AttentionFunction}(y_{t-1}, x_t) \); \text{AttentionFunction} is a 1 layer feed-forward network

- \( p_t = \frac{\exp(s_t)}{\sum_{t=1}^{M} \exp(s_t)} \)

- \( a_t = \sum_{t=1}^{M} p_t x_t \)
Stacking LSTM layers only improves performance initially; for translation, at around 6-8 layers, it becomes too difficult and slow to train.

Add residual connections, which greatly improves gradient flow in backward pass and allows for much deeper networks to be trained.

\[
\begin{align*}
\mathbf{c}_t^i, \mathbf{m}_t^i &= \text{LSTM}_i(\mathbf{c}_{t-1}^i, \mathbf{m}_{t-1}^i, \mathbf{x}_{t-1}^i; \mathbf{W}_i) \\
\mathbf{x}_t^i &= \mathbf{m}_t^i(+\mathbf{x}_{t-1}^i) \\
\mathbf{c}_{t+1}^{i+1}, \mathbf{m}_{t+1}^{i+1} &= \text{LSTM}_{i+1}(\mathbf{c}_{t-1}^{i+1}, \mathbf{m}_{t-1}^{i+1}, \mathbf{x}_t^i; \mathbf{W}_{i+1})
\end{align*}
\]
Model Architecture

Bi-directional Encoder for First Layer

- Context not necessarily left-to-right; could be in either direction depending on language
- Use a bi-directional encoder to take this into account; only in first layer to maximize parallelism
Parallelism

- **Data**
  - Train $n$ model replicas concurrently using Downpour SGD; all
    $n$ models share the same parameters and update
    asynchronously; generally, $n = 10$
  - Each replica works on minibatch of $m = 128$

- **Model**
  - Each layer runs on separate GPU; since most layers are
    unidirectional, $(i + 1)$th layer can start running before $i$th layer
    is finished; softmax layer also partitioned
  - Cannot have all bi-directional layers since both directions
    would have to be finished before next layers could start
  - Attention connected to bottom decoder layer, not top, because
    otherwise, no parallelism possible in decoding step
Segmentation

Wordpiece Model

- Used to solve Japanese/Korean segmentation problem; deterministic results
- Given a training corpus, select $D$ wordpieces which maximize language-model likelihood on training data; new wordpieces are added in a greedy manner
- Rare entity names and numbers are handled by shared wordpiece model between source and target language
- Wordpieces combine efficiency of words with flexibility of characters
Segmentation
Mixed Word/Character Model

- Similar to normal word model, except OOV words are not collapsed into \texttt{<unk>} character, but rather into a sequence of characters.
- Special prefixes added before characters to indicate position in the word: \texttt{<B>} (Beginning), \texttt{<M>} (Middle), and \texttt{<E>} (End).
Training Criteria

Objective Function

- Given $N$ pairs of input-output pairs $(X^{(i)}, Y^{*(i)})$,
  
  $O_{ML}(\theta) = \sum_{i=1}^{N} \log P_{\theta}(Y^{*(i)}|X^{(i)})$

  - Does not reward sentences close to but not exactly matching target sentence

- $O_{RL}(\theta) = \sum_{i=1}^{N} \sum_{Y \in y} P_{\theta}(Y|X^{(i)}) r(Y, Y^{*(i)})$

  - $r(Y, Y^{*(i)})$ is calculated using custom GLEU score instead of standard BLEU, which is more appropriate for an entire corpus

  - GLEU is calculated by taking all subsequences of size 1, 2, 3, or 4 tokens and taking minimum of recall ($\frac{\text{matching n-grams}}{\text{total n-grams in target}}$) and precision ($\frac{\text{matching n-grams}}{\text{total n-grams in generated}}$)

- First train model using standard likelihood objective until convergence, then refine using mixed objective

  - $O_{mixed}(\theta) = \alpha O_{ML}(\theta) + O_{RL}(\theta), \alpha = 0.017$
NMT too computationally intensive for inference

Constrain LSTM accumulators to \([-\delta, \delta]\), \(\delta\) ranges from 8.0 to 1.0 from beginning to end of training

\[
c_t^i, m_t^i = LSTM_i(c_{t-1}^i, m_{t-1}^i, x_t^{i-1}; W^i)
\]

\[
c_t^i = \max(-\delta, \min(\delta, c_t^i))
\]

\[
x_t'^i = m_t^i + x_t^{i-1}
\]

\[
x_t^i = \max(-\delta, \min(\delta, x_t'^i))
\]

\[
c_t'^{i+1}, m_t^{i+1} = LSTM_{i+1}(c_{t-1}^{i+1}, m_{t-1}^{i+1}, x_t^i; W^{i+1})
\]

Bound softmax layer output to \([-\gamma, \gamma]\), \(\gamma\) empirically determined to be 25.0

\[
v_t = W_s * y_t
\]

\[
v_t' = \max(-\gamma, \min(\gamma, v_t))
\]

\[
p_t = \text{softmax}(v_t')
\]
Quantized Model

Quantized Inference

- Replace all floating point operations in previous equations and also within LSTM with fixed-point 8 to 16-bit integer operations
- All weight matrices converted to 8-bit integer matrices
- All accumulator values become 16-bit integers
- Sigmoid, tanh, and element-wise operations become 16-bit integer operations
- During training, still keep floating-point precision; only clipping occurs during training
Quantized Model

Training Perplexity

- Quantized model similar to normal model; slightly better performance possibly due to regularization caused by clipping
- Model trained only on ML objective function, not refined version
Decoder
Beam Search

- Add length normalization and coverage penalty to traditional beam search
  - Length normalization: shorter sentences tend to be favored by regular beam search
    - \( lp(Y) = \frac{(5+|Y|)^\alpha}{(5+1)^\alpha} \)
  - Coverage penalty: favor sentences which cover source sentence according to attention module
    - \( cp(X; Y) = \beta \sum |X| \log(min(\sum |Y| p_{i,j}, 1.0)) \)
    - \( s(Y, X) = \log(P(Y|X))/lp(Y) + cp(X; Y) \)

- Pruning
  - Instead of 8-12 hypotheses for beam search, only consider 2-4
  - Only consider tokens within \textit{beamsize} of the best token score
  - Once normalized best score is found, prune all hypotheses more than \textit{beamsize} from score
Experiments & Results
Datasets and Setup

- WMT’14 English-to-French
- WMT’14 English-to-German
- Google’s translation production corpora
- Tested word-based, character-based, and wordpiece-based models
- Tested effects of objective refining and model ensembling
- 8 encoder layers, 8 decoder layers, attention is feed-forward with 1024 nodes, each layer has 1024 LSTM nodes
- Used BLEU as well as human evaluated side by side scores as metrics
Experiments & Results

Training Procedure

- Implemented with Tensorflow, 12 replicas running concurrently on separate machines, parameters updated asynchronously
- Initialize all trainable parameters within $[-0.04, 0.04]$, gradients clipped to 5.0 norm
- Stage 1 (ML objective): Each step is mini-batch of 128 examples; start with Adam ($\alpha = 0.0002$) for first 60k steps, then switch to SGD ($\alpha = 0.5$); start halving rate after 1.2M steps
- Stage 2 (RL objective): Simply run SGD until convergence
- Dropout applied to prevent overfitting; only on ML phase, not RL phase
Experiments & Results
ML and RL Objective Results

- BLEU and decoding time compared for GNMT across different models, then compared with other strong baselines

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>CPU decoding time per sentence (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>37.90</td>
<td>0.2226</td>
</tr>
<tr>
<td>Character</td>
<td>38.01</td>
<td>1.0530</td>
</tr>
<tr>
<td>WPM-8K</td>
<td>38.27</td>
<td>0.1919</td>
</tr>
<tr>
<td>WPM-16K</td>
<td>37.60</td>
<td>0.1874</td>
</tr>
<tr>
<td>WPM-32K</td>
<td>38.95</td>
<td>0.2118</td>
</tr>
<tr>
<td>Mixed Word/Character</td>
<td>38.39</td>
<td>0.2774</td>
</tr>
<tr>
<td>PBMT [15]</td>
<td>37.0</td>
<td></td>
</tr>
<tr>
<td>LSTM (6 layers) [31]</td>
<td>31.5</td>
<td></td>
</tr>
<tr>
<td>LSTM (6 layers + PosUnk) [31]</td>
<td>33.1</td>
<td></td>
</tr>
<tr>
<td>Deep-Att [45]</td>
<td>37.7</td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk [45]</td>
<td>39.2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>CPU decoding time per sentence (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>23.12</td>
<td>0.2972</td>
</tr>
<tr>
<td>Character</td>
<td>22.62</td>
<td>0.8011</td>
</tr>
<tr>
<td>WPM-8K</td>
<td>23.50</td>
<td>0.2079</td>
</tr>
<tr>
<td>WPM-16K</td>
<td>24.36</td>
<td>0.1931</td>
</tr>
<tr>
<td>WPM-32K</td>
<td>26.41</td>
<td>0.1882</td>
</tr>
<tr>
<td>Mixed Word/Character</td>
<td>24.17</td>
<td>0.3268</td>
</tr>
<tr>
<td>RNNSearch [37]</td>
<td>16.5</td>
<td></td>
</tr>
<tr>
<td>RNNSearch-LV [37]</td>
<td>16.9</td>
<td></td>
</tr>
<tr>
<td>RNNSearch-LV [37]</td>
<td>16.9</td>
<td></td>
</tr>
<tr>
<td>Deep-Att [45]</td>
<td>20.6</td>
<td></td>
</tr>
</tbody>
</table>

- Further RL refinement results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Trained with log-likelihood</th>
<th>Refined with RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>En→Fr</td>
<td>38.95</td>
<td>39.92</td>
</tr>
<tr>
<td>En→De</td>
<td>24.67</td>
<td>24.60</td>
</tr>
</tbody>
</table>
Experiments & Results
Ensembling

- Ensembling was performed with 8 models to produce final BLEU scores; RL-refined ensemble had slightly better scores than ML ensemble
- Humans were asked to rate translation quality on scale of 0-6; RL-refined ensemble had slightly worse scores than ML ensemble
Experiments & Results

Google Production Results

- No dropout because of large training set size, no RL-refinement because of dubious significance
- GNMT: Wordpiece models, no ensembling, shared vocabulary of 32k
- Evaluation data: 500 randomly sampled sentences and translations from Wikipedia and news websites

<table>
<thead>
<tr>
<th>Table 10: Mean of side-by-side scores on production data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>English → Spanish</td>
</tr>
<tr>
<td>English → French</td>
</tr>
<tr>
<td>English → Chinese</td>
</tr>
<tr>
<td>Spanish → English</td>
</tr>
<tr>
<td>French → English</td>
</tr>
<tr>
<td>Chinese → English</td>
</tr>
</tbody>
</table>
Summary

- GNMT approaches or surpasses all previously published results
- Key results
  - Wordpiece model effectively handles large, open vocabularies
  - Parallelism can improve efficiency of training large-scale models
  - Model quantization drastically improves inference speed
- GNMT approaches average human translator results and improves upon previous phrase-based translators by around 60%
References