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

Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, et al.

Google

arXiv:1609.08144v2

Reviewed by : Bill Zhang
University of Virginia
https://qdata.github.io/deep2Read/

Outline

- 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

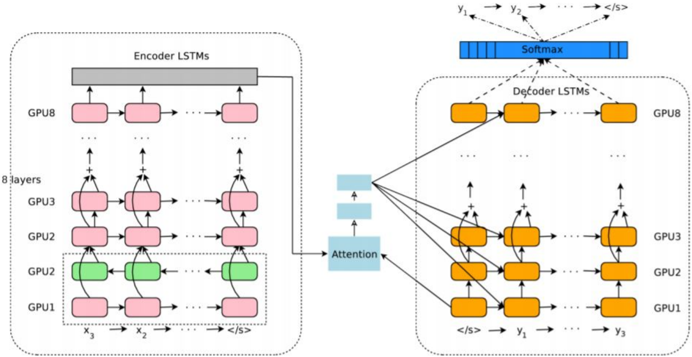
Overview

- 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: $\boldsymbol{x_1, ..., x_M} = EncoderRNN(x_1, ..., x_M)$
  - $P(Y|X) = P(Y|\boldsymbol{x_1, ..., x_M})$
    $= \Pi_{i=1}^{N} P(y_i|y_0, y_1, ..., y_{i-1}; \boldsymbol{x_1, ..., x_M})$
  - Decoder: RNN with softmax; RNN outputs a hidden state $\boldsymbol{y_i}$ then generates a probability distribution using softmax
- Deep networks perform better, so both encoder and decoder have multiple layers

Model Architecture

Overview

- $s_t = AttentionFunction(\boldsymbol{y_{t-1}}, \boldsymbol{x_t})$; $AttentionFunction$ is a 1 layer feed-forward network
- $p_t = \frac{\exp(s_t)}{\sum_{t=1}^{M} \exp(s_t)}$
- $\boldsymbol{a_t} = \sum_{t=1}^{M} p_t \boldsymbol{x_t}$



Model Architecture

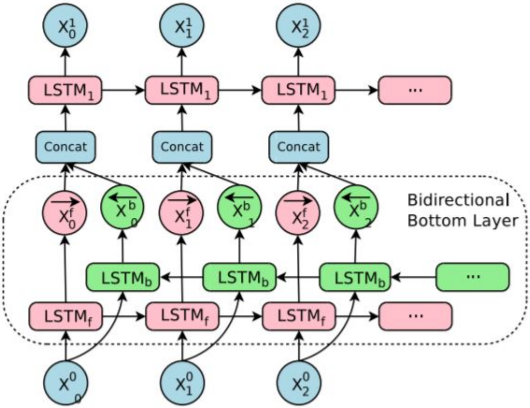
Residual Connections

- 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
  - $\boldsymbol{c}_t^i, \boldsymbol{m}_t^i = LSTM_i(\boldsymbol{c}_{t-1}^i, \boldsymbol{m}_{t-1}^i, \boldsymbol{x}_t^{i-1}; \boldsymbol{W}^i)$
  - $\boldsymbol{x}_t^i = \boldsymbol{m}_t^i (+\boldsymbol{x}_t^{i-1})$
  - $\boldsymbol{c}_t^{i+1}, \boldsymbol{m}_t^{i+1} = LSTM_{i+1}(\boldsymbol{c}_{t-1}^{i+1}, \boldsymbol{m}_{t-1}^{i+1}, \boldsymbol{x}_t^i; \boldsymbol{W}^{i+1})$

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 ith 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 <unk> character, but rather into a sequence of characters
- Special prefixes added before characters to indicate position in the word: <B> (Beginning), <M> (Middle), and <E> (End)

Training Criteria

Objective Function

- Given N pairs of input-output pairs $(X^{(i)}, Y^{*(i)})$, $O_{ML}(\theta) = \Sigma_{i=1}^{N} \log P_\theta(Y^{*(i)}|X^{(i)})$
  - Does not reward sentences close to but not exactly matching target sentence
- $O_{RL}(\theta) = \Sigma_{i=1}^{N} \Sigma_{Y \in y} P_\theta(Y|X^{(i)}) r(Y, Y^{*(i)})$
  - $r(Y, Y^{*(i)})$ 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
  - $O_{mixed}(\theta) = \alpha O_{ML}(\theta) + O_{RL}(\theta)$, $\alpha = 0.017$

Quantized Model

Constraints

- NMT too computationally intensive for inference
- Constrain LSTM accumulators to $[-\delta, \delta]$, δ ranges from 8.0 to 1.0 from beginning to end of training
  - $\boldsymbol{c}_t'^{\boldsymbol{i}}, \boldsymbol{m}_t^{\boldsymbol{i}} = LSTM_i(\boldsymbol{c}_{t-1}^{\boldsymbol{i}}, \boldsymbol{m}_{t-1}^{\boldsymbol{i}}, \boldsymbol{x}_t^{\boldsymbol{i-1}}; \boldsymbol{W}^{\boldsymbol{i}})$
  - $\boldsymbol{c}_t^{\boldsymbol{i}} = max(-\delta, min(\delta, \boldsymbol{c}_t'^{\boldsymbol{i}}))$
  - $\boldsymbol{x}_t'^{\boldsymbol{i}} = \boldsymbol{m}_t^{\boldsymbol{i}} + \boldsymbol{x}_t^{\boldsymbol{i-1}}$
  - $\boldsymbol{x}_t^{\boldsymbol{i}} = max(-\delta, min(\delta, \boldsymbol{x}_t'^{\boldsymbol{i}}))$
  - $\boldsymbol{c}_t'^{\boldsymbol{i+1}}, \boldsymbol{m}_t^{\boldsymbol{i+1}} = LSTM_{i+1}(\boldsymbol{c}_{t-1}^{\boldsymbol{i+1}}, \boldsymbol{m}_{t-1}^{\boldsymbol{i+1}}, \boldsymbol{x}_t^{\boldsymbol{i}}; \boldsymbol{W}^{\boldsymbol{i+1}})$
- Bound softmax layer output to $[-\gamma, \gamma]$, γ empirically determined to be 25.0
  - $\boldsymbol{v_t} = \boldsymbol{W_s} * \boldsymbol{y_t}$
  - $\boldsymbol{v}_t' = max(-\gamma, min(\gamma, \boldsymbol{v_t}))$
  - $\boldsymbol{p_t} = softmax(\boldsymbol{v}_t')$

Quantized Model

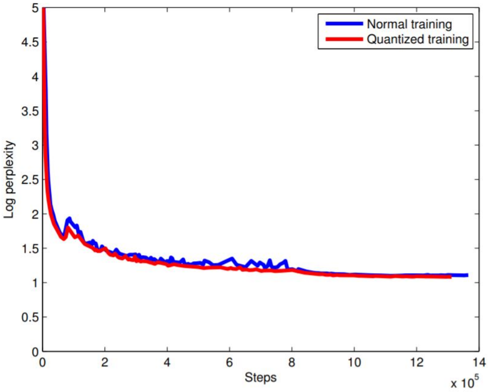
Quantized Inference

- 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
    - $cp(X;Y) = \beta * \Sigma_{i=1}^{|X|} \log(min(\Sigma_{i=1}^{|Y|} p_{i,j}, 1.0))$
  - $s(Y,X) = \log(P(Y|X))/lp(Y) + cp(X;Y)$
- Pruning
  - Instead of 8-12 hypotheses for beam search, only consider 2-4
  - Only consider tokens within *beamsize* of the best token score
  - Once normalized best score is found, prune all hypotheses more than *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 4: Single model results on WMT En→Fr (newstest2014)

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

Table 5: Single model results on WMT En→De (newstest2014)

Model	BLEU	CPU decoding time per sentence (s)
Word	23.12	0.2972
Character (512 nodes)	22.62	0.8011
WPM-8K	23.50	0.2079
WPM-16K	24.36	0.1931
WPM-32K	24.61	0.1882
Mixed Word/Character	24.17	0.3268
PBMT [6]	20.7	
RNNSearch [37]	16.5	
RNNSearch-LV [37]	16.9	
RNNSearch-LV [37]	16.9	
Deep-Att [45]	20.6	

- Further RL refinement results

Table 6: Single model test BLEU scores, averaged over 8 runs, on WMT En→Fr and En→De

Dataset	Trained with log-likelihood	Refined with RL
En→Fr	38.95	39.92
En→De	24.67	24.60

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 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

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

- https://arxiv.org/pdf/1609.08144v2.pdf