집현전 Season2 중급반
https://github.com/jiphyeonjeon
#1 GNMT, Machine Translation, Wordpiece model, Parallelism, Seq2Seq

Google's Neural Machine Translation System

Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi # 2016.10.08

2021.06.13

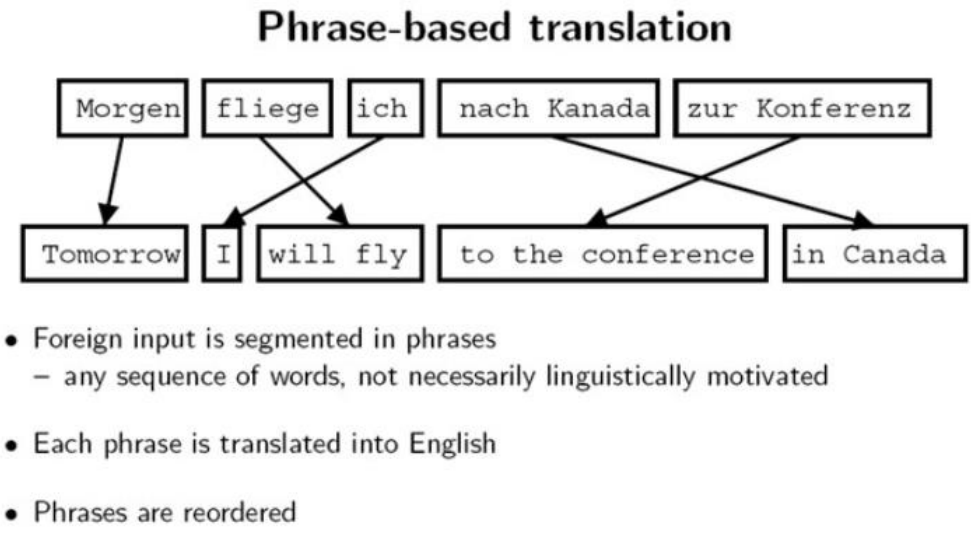
집현전 중급반 1조: 진명훈(발표자), 송지현, 지우석

Introduction & Related Works

Quick Research History

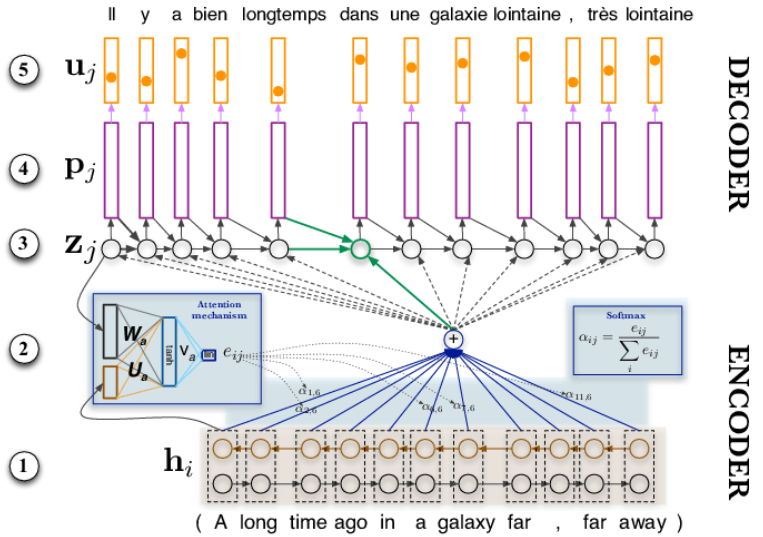
Old: Phrase-based translation

- Lots of individual pieces
- Optimized somewhat independently
- Choice design problem



New: Neural Machine Translation

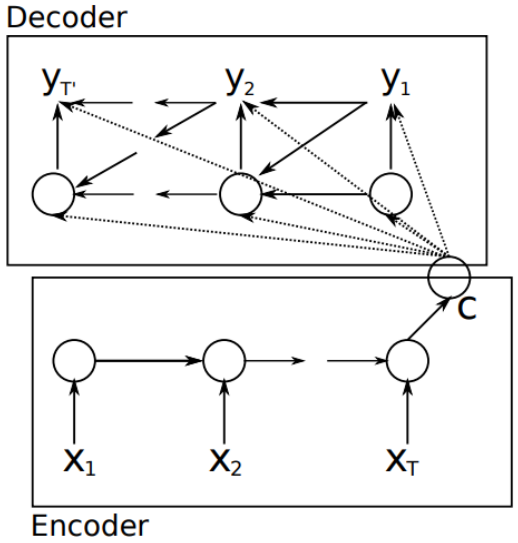
- End-to-end learning
- Simpler architecture!
- Plus results are much better!



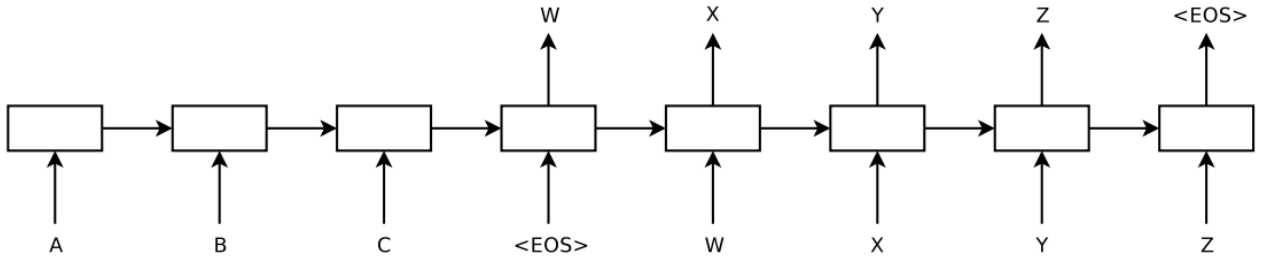
https://www.slideshare.net/ChesterChen/moving-to-neural-machine-translation-at-google-gopromeetup?from_action=save
https://www.researchgate.net/figure/Architecture-of-the-NMT-system-equipped-with-an-attention-mechanism_fig1_308152598

Quick Research History

- Sequence –To–Sequence Models (NIPS 2014)
  - Based on many earlier approaches to estimate P(Y|X) directly
  - SOTA on WMT En–〉Fr using custom software, very long training
  - Translation could be learning w/o explicit alignment!
  - Drawback: all information needs to be carried in internal state
    - Translation breaks down for long sentences!



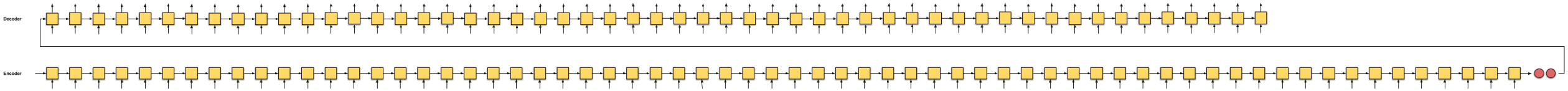
RNN Encoder–decoder
Kyunghyun Cho et al., 2014

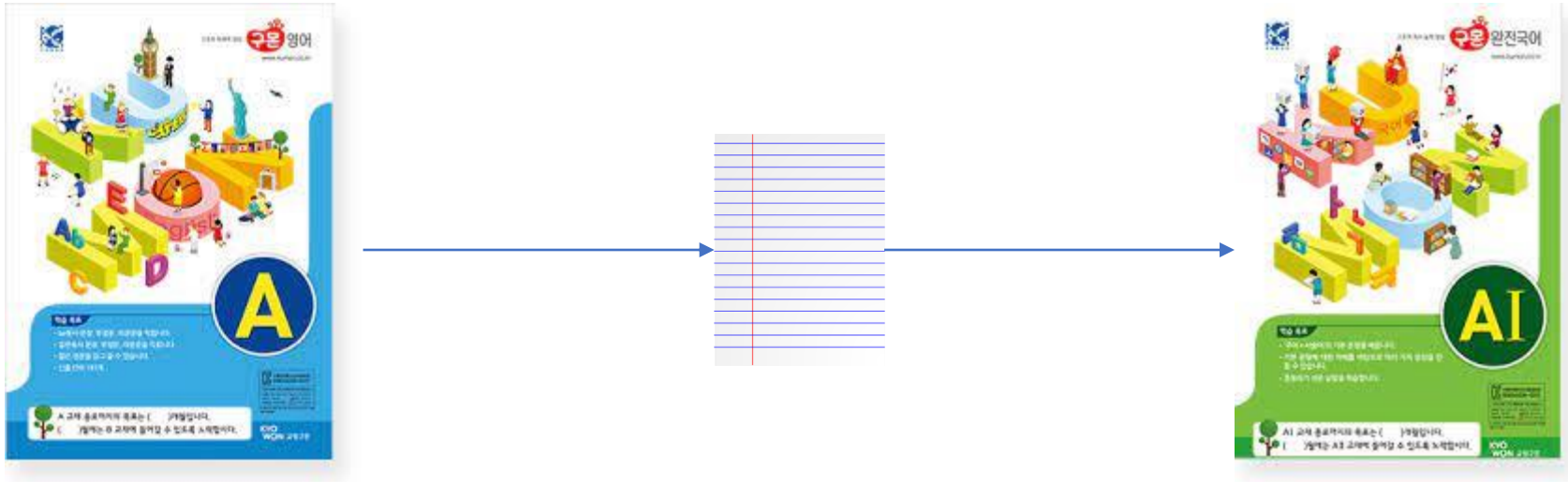


Seq2Seq
Ilya Sutskever et al., 2014

Quick Research History

- Sequence –To–Sequence Models (NIPS 2014)
  - Based on many earlier approaches to estimate P(Y|X) directly
  - SOTA on WMT En–〉Fr using custom software, very long training
  - Translation could be learning w/o explicit alignment!
  - Drawback: all information needs to be carried in internal state
    - Translation breaks down for long sentences!

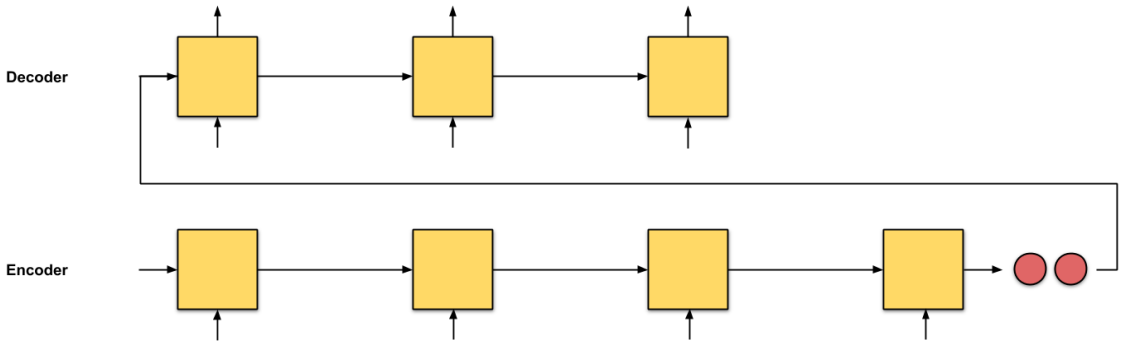




https://towardsdatascience.com/attn–illustrated–attention–5ec4ad276ee3

Quick Research History

- Sequence –To–Sequence Models (NIPS 2014)
  - Based on many earlier approaches to estimate P(Y|X) directly
  - SOTA on WMT En–〉Fr using custom software, very long training
  - Translation could be learning w/o explicit alignment!
  - Drawback: all information needs to be carried in internal state
    - Translation breaks down for long sentences!

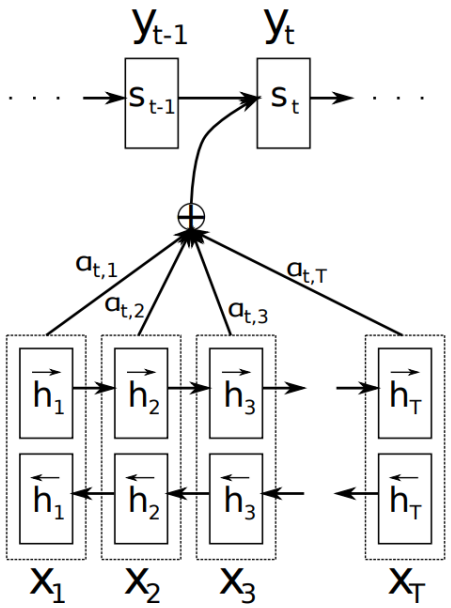


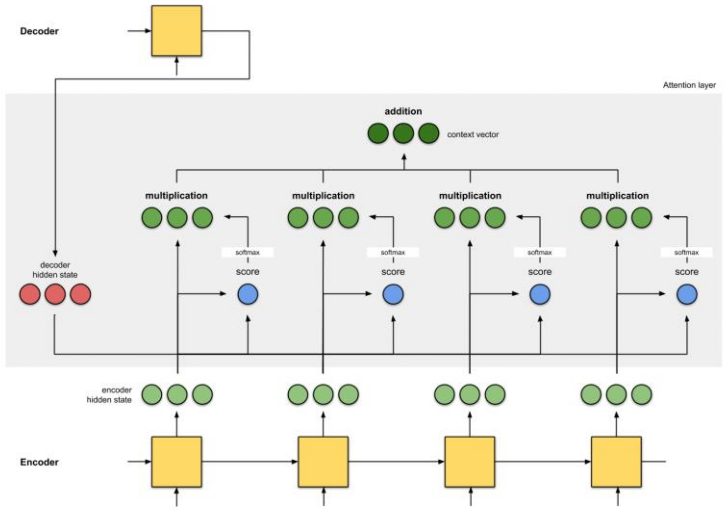
Encoder

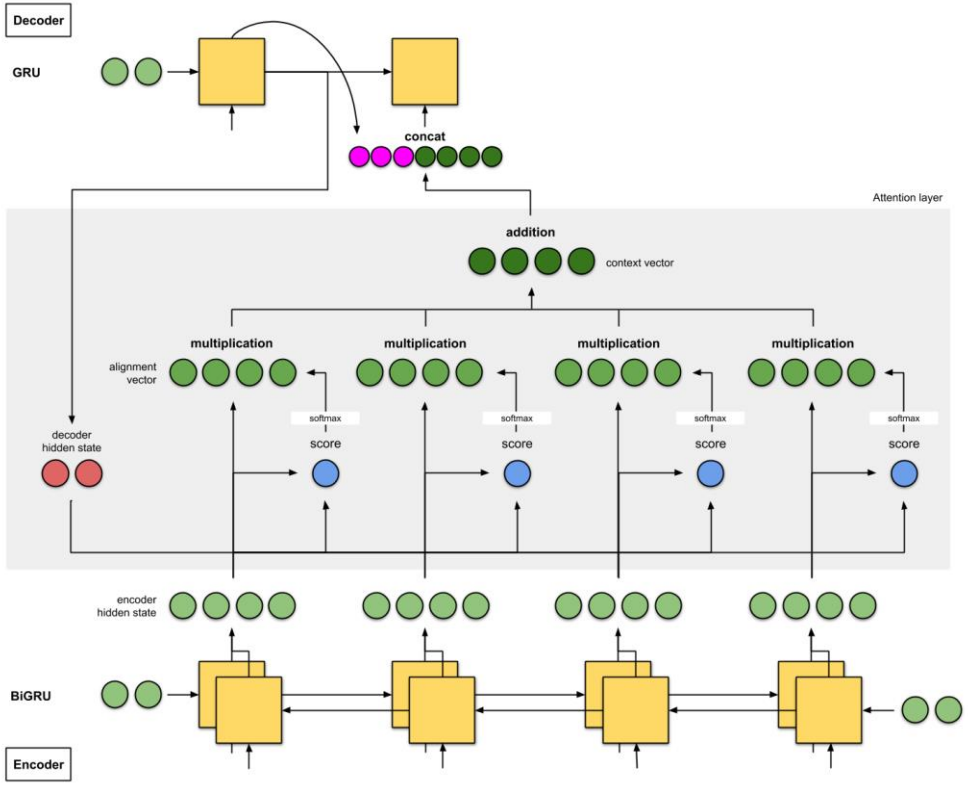
https://towardsdatascience.com/attn–illustrated–attention–5ec4ad276ee3

Quick Research History

- Attention Models (2014)
  - Removes drawback by giving access to all encoder states
    - **Translation quality is now independent of sentence length!**
  - RNNSearch (Bahdanau Attention, NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE)
    - Attention module 제안 paper
  - RNNSearch-LV (Jean et al., On Using Very Large Target Vocabulary for Neural Machine Translation)
    - Importance Sampling
    - Larger vocabulary 한계를 극복하여 성능을 개선한 paper







https://arxiv.org/pdf/1409.0473.pdf

Production launching

Expected time to launch:

3 years

Actual time to launch:

13.5 months

Sept 2015: Began project using TensorFlow

Feb 2016: First production data results

Sept 2016: zh->en launched

Nov 2016: 8 languages launched (16 pairs to/from English)

Mar 2017: 7 more launched (Hindi, Russian, Vietnamese, Thai, Polish, Arabic, Hebrew)

Apr 2017: 26 more launched (16 European, 8 Indish, Indonesian, Afrikaans)

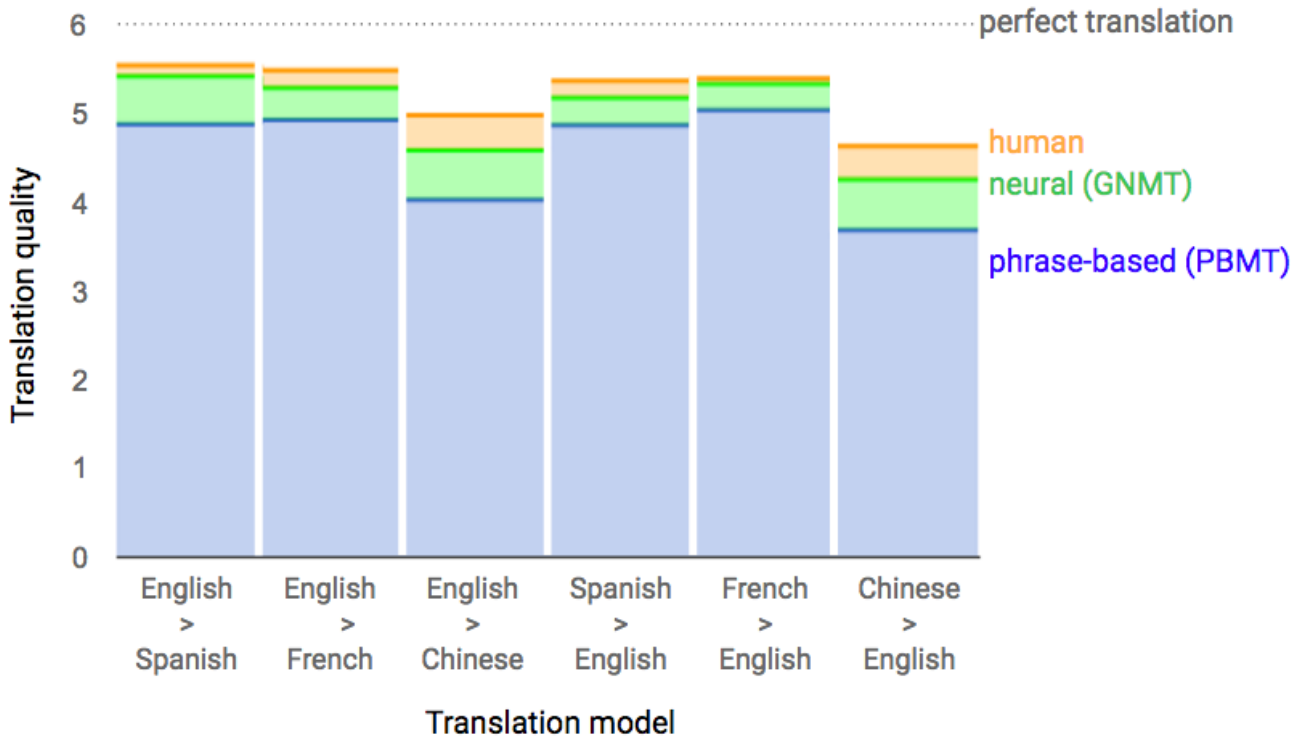
Jun/Aug 2017: 36/20 more launched

97 launched!

https://www.slideshare.net/ChesterChen/moving-to-neural-machine-translation-at-google-gopromeetup?from_action=save

Production launching

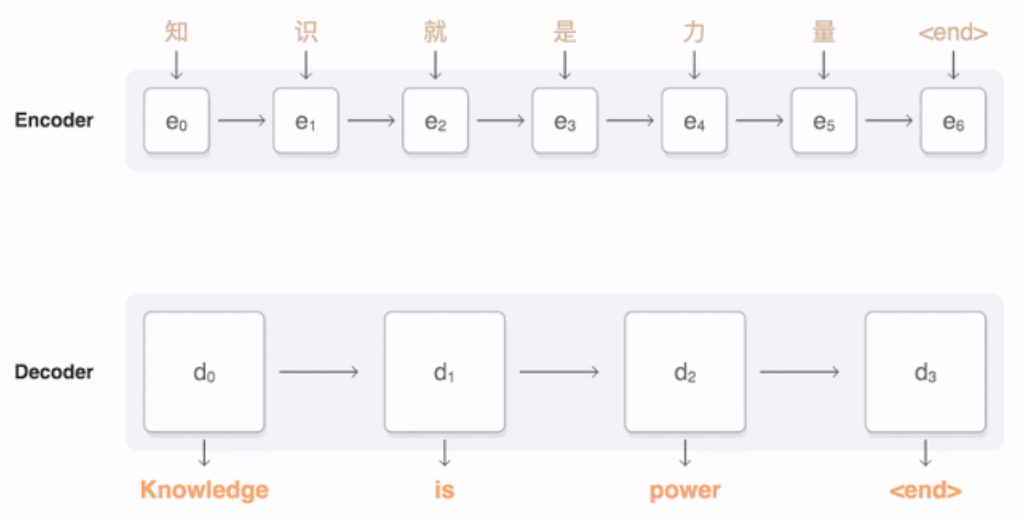
- 몇 년 전부터 RNN을 사용하여 Seq2Seq 매핑 연구를 시작
- PBMT는 입력 문장을 단어와 구문으로 분리하여 독립적으로 번역하지만
- NMT는 전체 입력 문장을 번역 단위로 간주!
- 최초엔 PBMT와 NMT의 성능이 유사했음



https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

Production launching

- 이 후, external alignment model을 모방하여 rare word를 처리하는 작업
  - Addressing the rare word problem in nmt, Luong et al., ACL 2015
- 입력과 출력 단어를 정렬하는 데 attention하는 방법
  - Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau et al., ICLR 2015
- Rare word 처리를 위해 단어를 더 작은 단위 (subword-units)로 나누는 등의 기술 제안
  - Japanese and Korean voice search, Schuster et al., ICASSP 2012
  - Neural Machine Translation of Rare Words with Subword Units, Sennrich et al., ACL 2016
- 그럼에도 불구, production 단에서 사용할 만큼의 성능이 나오지 못함 (성능 및 속도)
  - 이를 GNMT paper에서 어떻게 개선했는지 설명! (quantization & parallelism)



2016 당시 NMT의 세 가지 단점?

1. 느린 학습/추론 속도
2. 비효율적인 rare words 처리
3. Source 문장 coverage

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

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Abstract

Neural Machine Translation (NMT) is an end-to-end learning approach for automated translation, with the potential to overcome many of the weaknesses of conventional phrase-based translation systems. Unfortunately, NMT systems are known to be computationally expensive both in training and in translation inference – sometimes prohibitively so in the case of very large data sets and large models. Several authors have also charged that NMT systems lack robustness, particularly when input sentences contain rare words. These issues have hindered NMT's use in practical deployments and services, where both accuracy and speed are essential. In this work, we present GNMT, Google's Neural Machine Translation system, which attempts to address many of these issues. Our model consists of a deep LSTM network with 8 encoder and 8 decoder layers using residual connections as well as attention connections from the decoder network to the encoder. To improve parallelism and therefore decrease training time, our attention mechanism connects the bottom layer of the decoder to the top layer of the encoder. To accelerate the final translation speed, we employ low-precision arithmetic during inference computations. To improve handling of rare words, we divide words into a limited set of common sub-word units ("wordpieces") for both input and output. This method provides a good balance between the flexibility of "character"-delimited models and the efficiency of "word"-delimited models, naturally handles translation of rare words, and ultimately improves the overall accuracy of the system. Our beam search technique employs a length-normalization procedure and uses a coverage penalty, which encourages generation of an output sentence that is most likely to cover all the words in the source sentence. To directly optimize the translation BLEU scores, we consider refining the models by using reinforcement learning, but we found that the improvement in the BLEU scores did not reflect in the human evaluation. On the WMT'14 English-to-French and English-to-German benchmarks, GNMT achieves competitive results to state-of-the-art. Using a human side-by-side evaluation on a set of isolated simple sentences, it reduces translation errors by an average of 60% compared to Google's phrase-based production system.

GNMT 두둥

- **model architecture**
  - 8 Encoder (1st bi-directional, 2nd~ uni-directional)
  - 8 Decoder (uni-directional)
  - Bahdanau attention
  - Residual connection
- **Training speed**
  - model parallelism
  - data parallelism
  - last encoder → first decoder connect
- **Inference speed**
  - quantization with low-precision and clipping
- **Rare words**
  - Wordpiece subword tokenizer
- **Source sentence coverage**
  - Beam Search with length penalty
  - + coverage penalty
- **Performance**
  - BLEU with REINFORCE

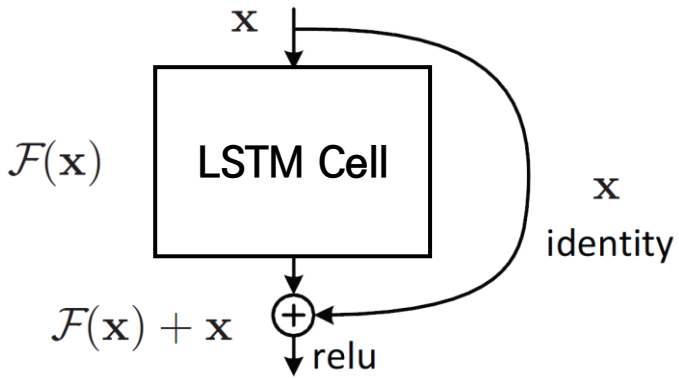
Related Work (@ 2016)

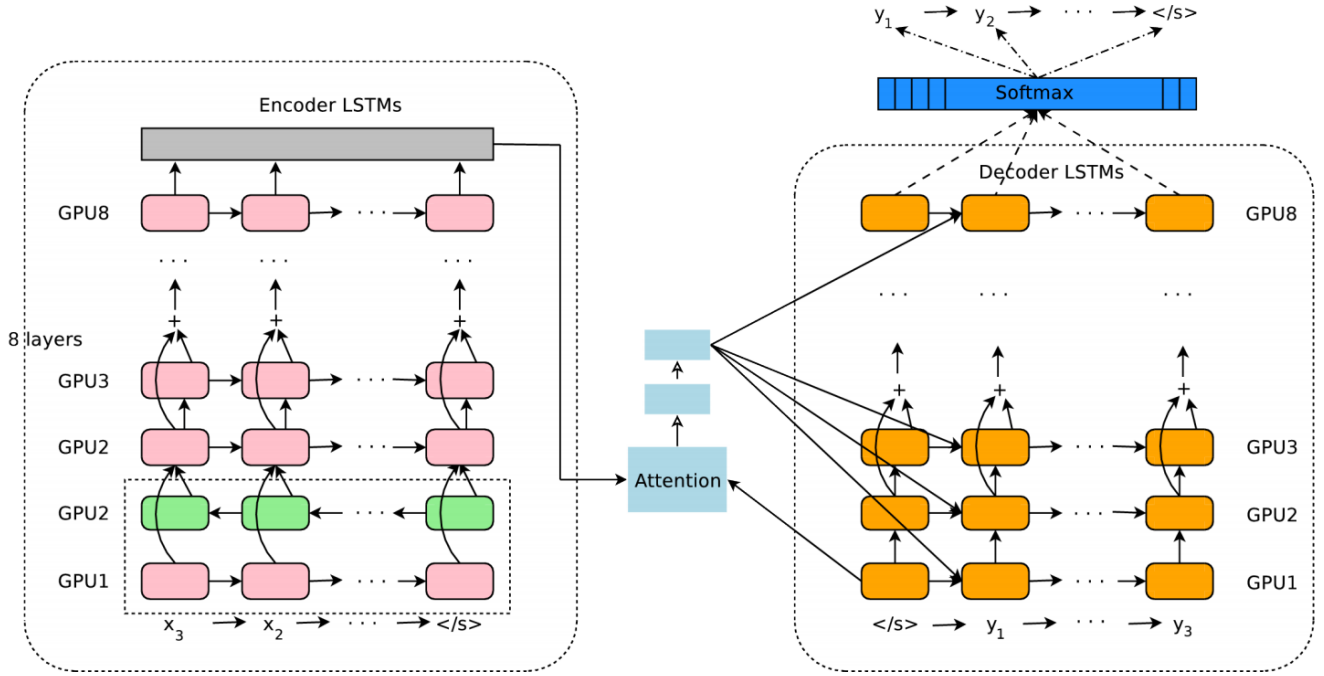
- SMT는 수 십년 간 우세한 번역 paradigm
  - A statistical approach to language translation. Brown et al., COLING 1988
  - A statistical approach to machine translation, Brown et al., Computational linguistics 1990
  - The mathematics of statistical machine translation: Parameter estimation, Brown et al., Computational linguistics 1993
- 실용적인 구현체는 PBSMT, 이는 다른 길이의 단어 혹은 구(phrase)의 sequence를 번역
  - Statistical phrase-based translation, Koehn et al., NAACL 2003
- Devlin et al., 연구처럼 phrase repr를 학습하기 위해 joint language model을 포함한 모델도 성공적이었음
  - 그러나 여전히 SMT의 단점이 존재함
  - Fast and robust neural network joint models for statistical machine translation, Devlin et al., ACL 2014
- Cho et al., 연구와 같이 learning phrase 표현을 학습하거나 Kalchbrenner et al., 연구와 같이 E2E 번역을 배우는 접근도 성공적
  - (RNN EncDec) Learning phrase representations using RNN encoder-decoder for statistical machine translation, Cho et al., EMNLP 2014
  - Recurrent continuous translation models, Kalchbrenner et al., EMNLP 2013
- 아래 두 paper에 의하면, NMT 번역 품질은 PBMT 목 밑까지 따라왔다고 함 ~~(Recall: 5년전이면 마스크도 안쓰고 돌아댕겼…)~~
  - (seq2seq) Sequence to sequence learning with neural networks, Sutskever et al., NIPS 2014
  - (RNNSearch) Neural machine translation by jointly learning to align and translate, Bahdanau et al., ICRL 2015
- 아마 처음으로 PBMT의 성능을 뛰어넘은 연구는 Luong et al., 의 rare words를 처리한 모델일 것임 (BLEU += 0.5)
  - Addressing the rare word problem in neural machine translation, Luong et al., ACL-IJCNLP 2015
- 아래와 같이 여러 NMT technique들을 적용하여 성능은 향상되었으나 production 단으로 쓰기엔 아직 부족하다고 판단
  - (rare word를 attention으로 처리) On using very large target vocabulary for neural machine translation, Sebastien et al., ACL-IJCNLP 2015
  - (Coverage mechanism) Coverage-based neural machine translation, Tu et al., ACL 2016
  - (Multi-task and SSL + more data) Multi-task learning for multiple language translation, Dong et al., ACL 2015
  - (Character decoder) A charcter-level decoder without explicit segmentation for neural machine translation, Chung et al., arxiv print 2016
  - (Character encoder) Character-based neural machine translation, Costa-jussa et al., CoRR 2016
  - (Subword units) A Neural machine translation of rare words with subword units, Sennrich et al., ACL 2016
  - (Luong Attention) Effective approaches to attention-based neural machine translation, EMNLP 2015
  - (sentence-level loss minimization) Sequence level training with recurrent neural networks, Ranzato et al.,^ ICLR 2015

Model Architecture

Model Architecture

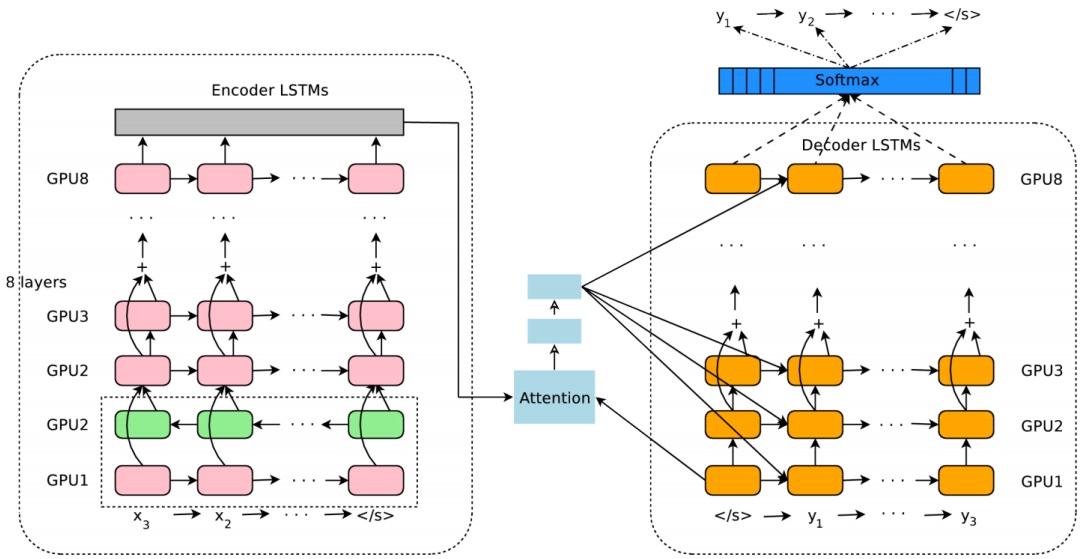
- By seq2seq paper, shallow ⟨⟨ deep lstm
- Bahdanau attention 사용 (RNNSearch)
- 8 replica를 사용하며 data parallelism 실시
- Residual connection 사용
- Left-to-right + right-to-left information → Encoder의 첫 번째 layer를 bidirectional하게!
  - 왜 첫 번째 layer만 양방향으로 구축했는가? → model parallelism

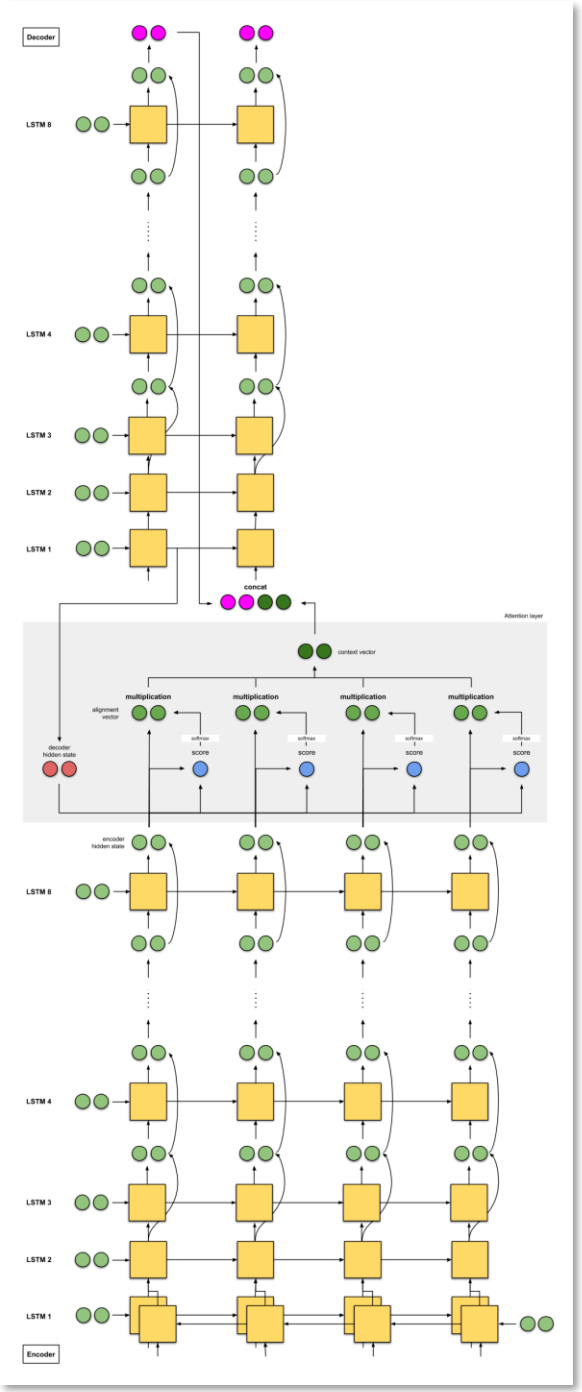




Model Architecture

- Google‘s Neural Machine Translation

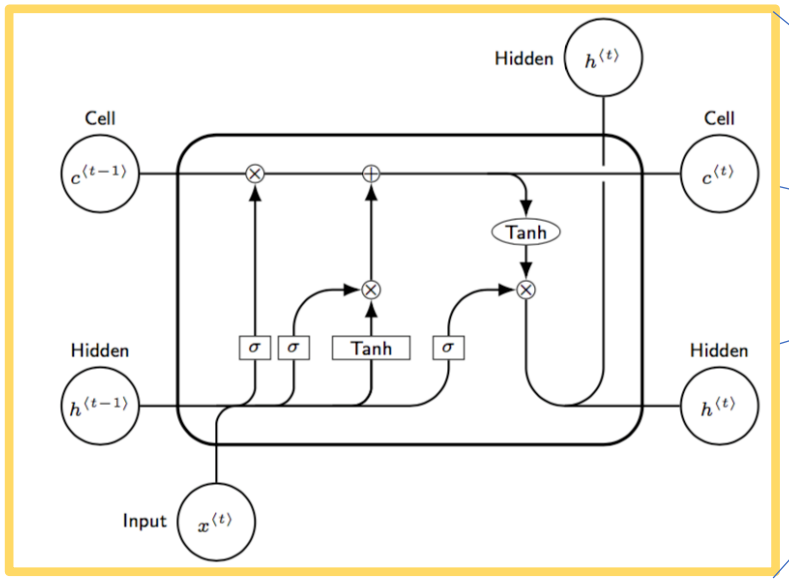


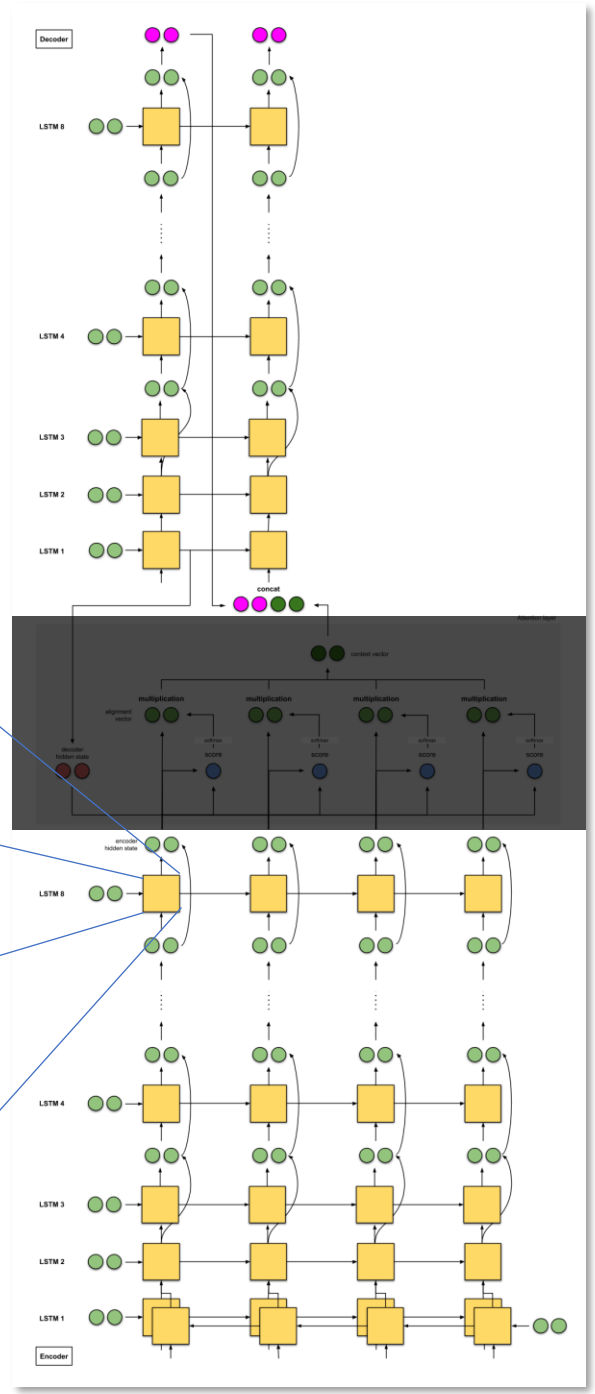


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#7eef

Model Architecture

- Google's Neural Machine Translation
  - Encoder
    - 8 layers
    - 1st layer만 bi-directional
    - 3rd layer부터 residual connection 적용
    - LSTM cell
  - Decoder
    - 8 layers
    - 3rd layer부터 residual connection 적용
    - LSTM cell

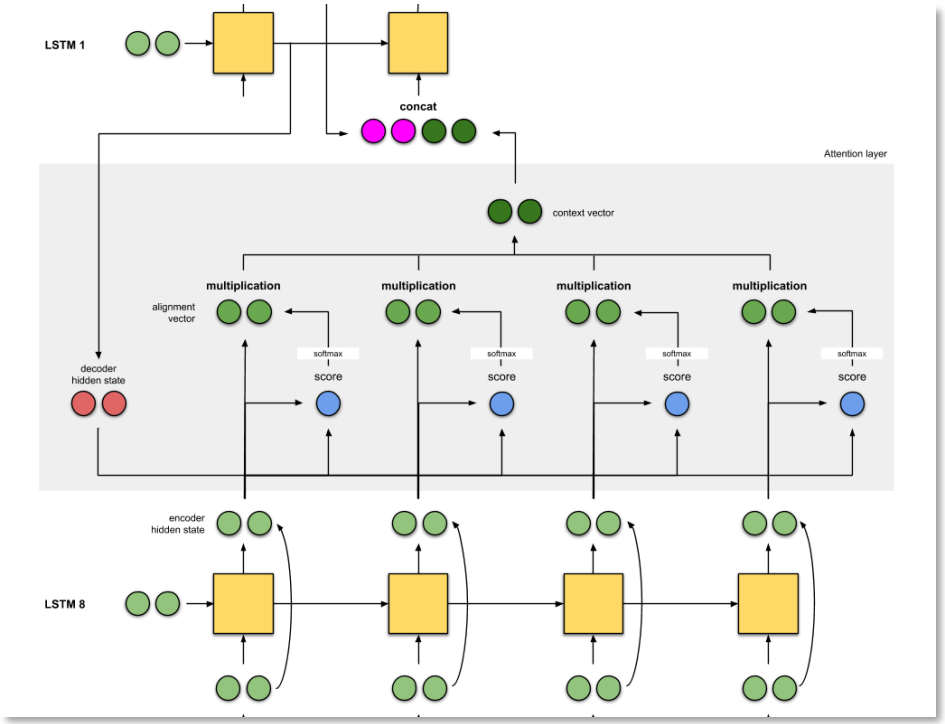


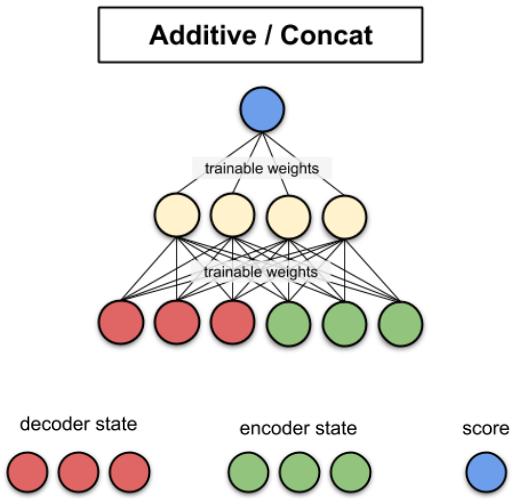


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#7eef

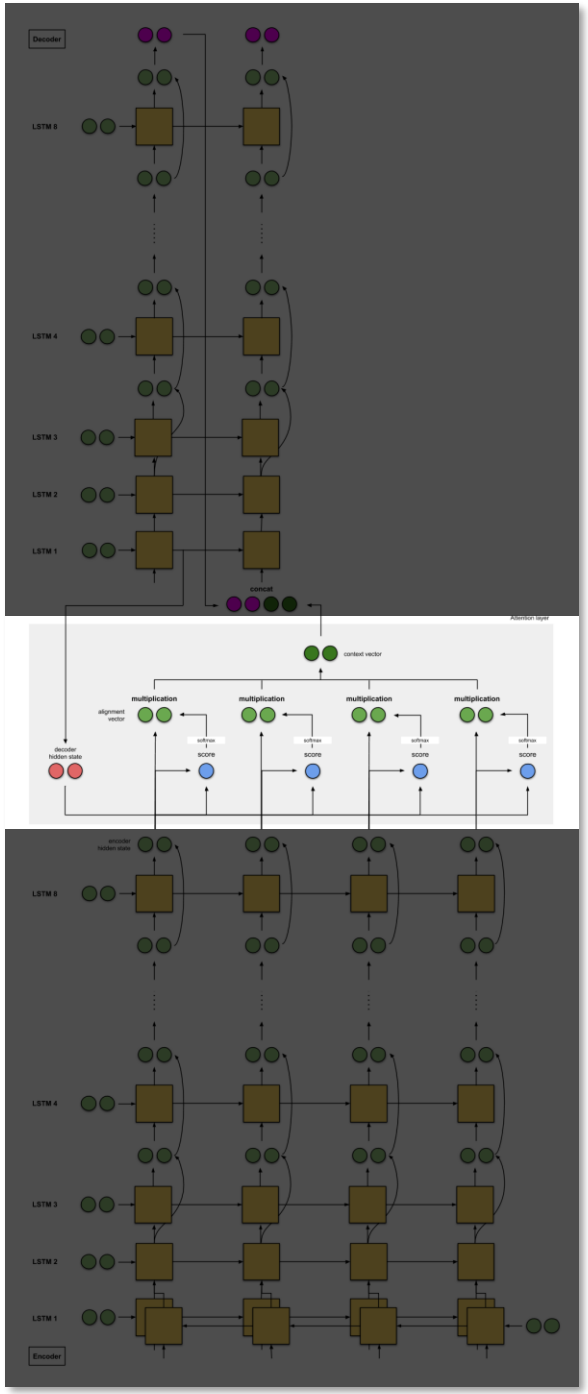
Model Architecture

- Google's Neural Machine Translation
  - Attention Module
    - 8th Encoder hidden states와 1st Decoder hidden states로 score 계산 (효율적인 병렬처리)
    - 아래의 alignment를 최상층의 인코더 레이어 아웃풋과 최하위 디코더 레이어 아웃풋으로!





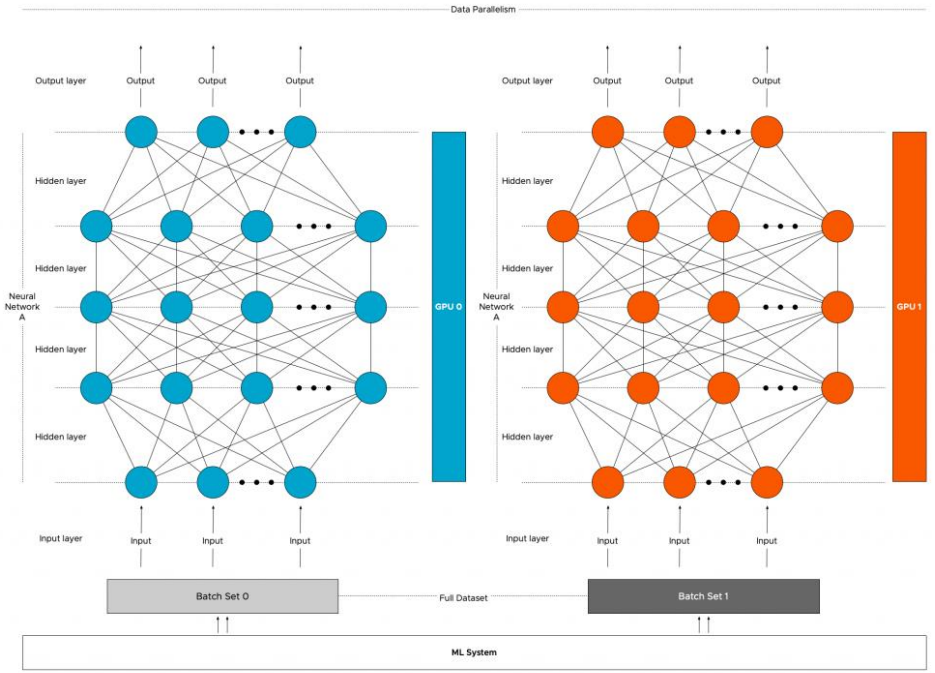
$$score(s_t, h_t) = v_a^T \tanh(W_a[s_t; h_i])$$

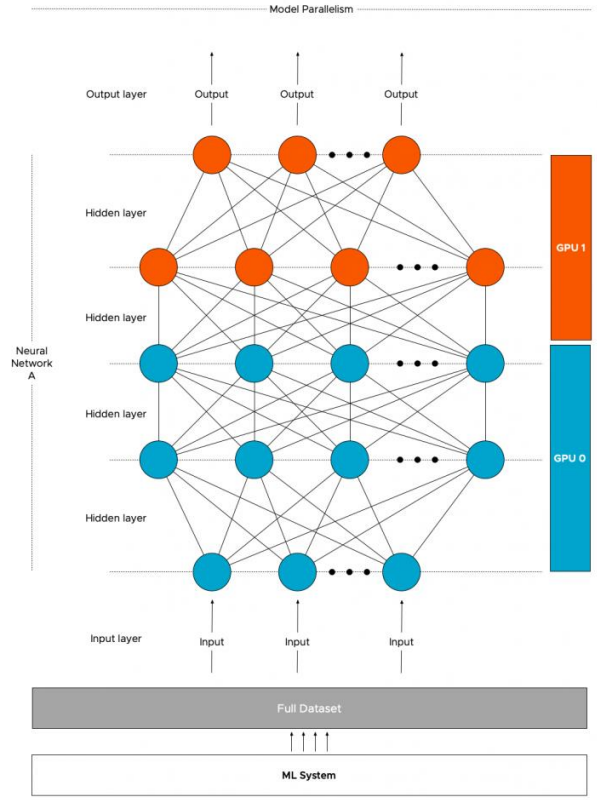


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#7eef

Model Architecture

- Google's Neural Machine Translation
  - Data Parallelism
    - n개의 replica를 downpour SGD 알고리즘을 이용하여 학습 (여러 모델에서 계산된 gradient를 평균내서 각각의 모델에 적용)
    - n개의 replica들은 parameter를 공유하며 Adam + SGD로 update 실시 (n=10, minibatch=128)
  - Model Parallelism
    - 각 replica별 gradient computation 속도를 증진시키고자 사용됨

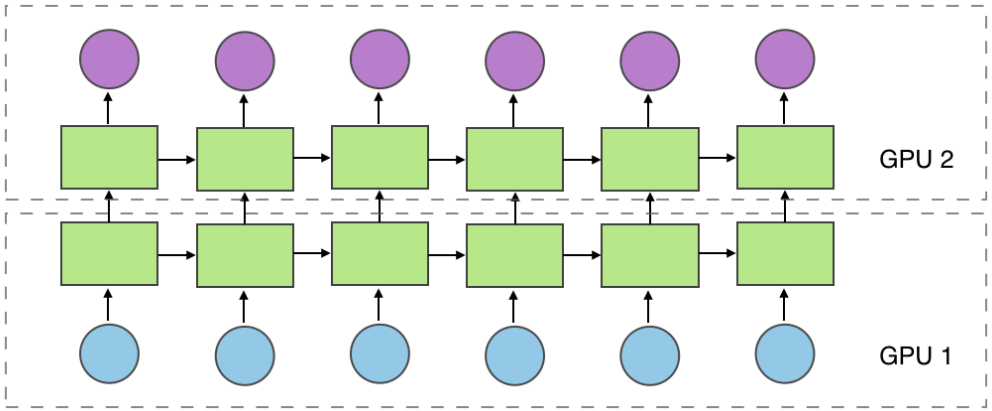


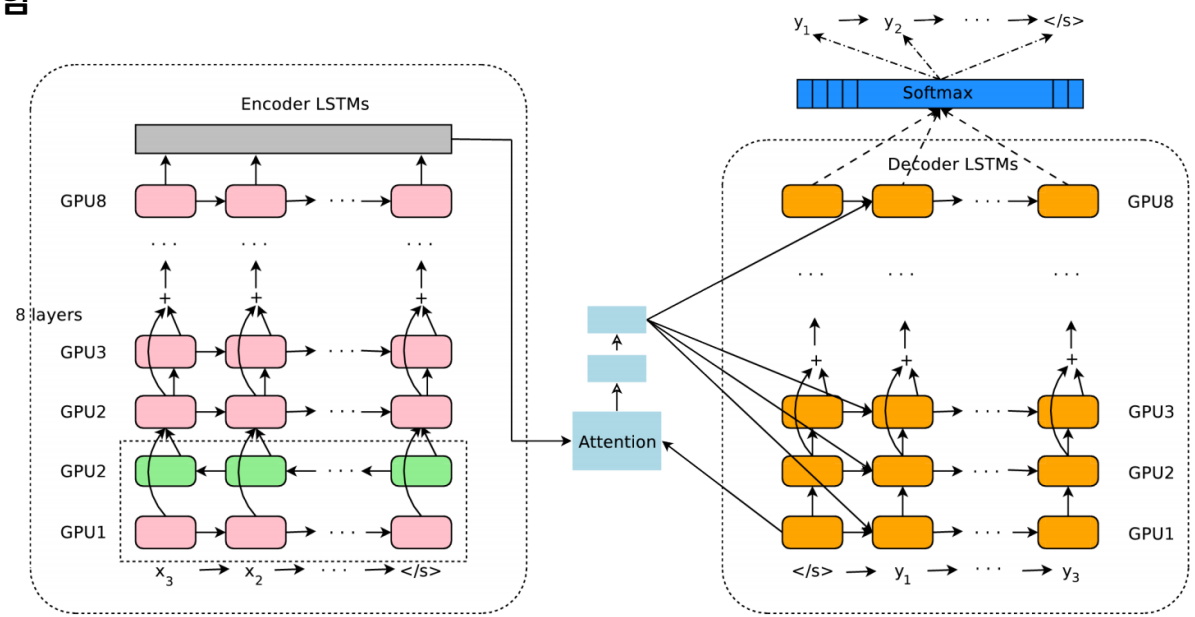


http://static.googleusercontent.com/media/research.google.com/ko//archive/large_deep_networks
https://frankdenneman.nl/2020/02/19/multi-gpu-and-distributed-deep-learning/

Model Architecture

- Google's Neural Machine Translation
  - Model Parallelism
    - Depth dimension에 따라 나눠지고 multiple-GPU에 할당됨
    - 1st Encoder layer를 제외한 모든 계층은 uni-directional이기 때문에 i+1th layer는 ith layer가 완전히 종료되기 전에 계산을 시작할 수 있다.
    - 본 논문의 model parallelism은 모델 구조로 인한 제약이 존재함
    - 8th Encoder hidden states와 1st Decoder hidden states로 score 계산 (효율적인 병렬처리)
    - 위처럼 하지 않으면, 둘 이상의 GPU를 사용하는 이점을 얻지 못함





https://mxnet.apache.org/versions/1.2.1/faq/model_parallel_lstm.html

Segmentation Approaches

Segmentation Approaches

- NMT에선 open vocabulary problem이 있지만 fixed word vocabulary로 연산함
- 그럼 oov를 어떻게 처리할 것인가?
  1. Copy model-based
     - (attention based) On using very large target vocabulary for neural machine translation. Sebastien et al., ACL-IJCNLP 2015
     - (external alignment model) Addressing the rare word problem in neural machine translation, Luong et al., ACL-IJCNLP 2015
     - (point networks) Pointing the unknown words, Gulcehre et al., CoRR 2016
  2. subword units
     - (characters) A character-level decoder without explicit segmentation for neural machine translation, Chung et al., CoRR 2016
     - (mixed word/chars) Achieving open vocabulary neural machine translation with hybrid word-character models, Luong et al., CoRR 2016
     - (intelligent sub-words) Neural machine translation of rare words with subword units, Sennrich et al., ACL 2016

Segmentation Approaches

- Wordpiece Model은 위의 2번 subword units를 활용하는 방법론
- 아래의 처리 과정을 거침
    1. 충분히 많은 training data (corpus)를 준비
    2. Desired subword vocabulary size를 정의
    3. 단어를 character의 sequence로 분해
    4. Step 3의 데이터를 기반으로 Language Model 구축
    5. 모델에 추가될 때 training data에 대한 likelihood를 가장 많이 증가시키는 모든 가능한 단어 중에서 새로운 단어 단위를 선택
    6. Step 2에서 정의된 subword vocabulary size에 도달하거나 likelihood 증가가 특정 임계값 아래로 떨어질 때 까지 Step 5 반복
- 즉, Character 단위로 분리 후 언어 모델을 학습, 결합되었을 때 likelihood를 maximize하는 경우! 추가하도록 함
- BPE이랑은 다른 알고리즘!
    - BPE는 character 단위로 분리 후 등장 빈도에 따라 결합하는 과정을 n회 반복하는 알고리즘
    - Sennrich의 addressing rare word paper에서 적용된 알고리즘
    - GPT-2의 tokenizer이기도 하죠!

https://www.youtube.com/watch?v=XMOtdAP_AHY
https://medium.com/@makcedward/how-subword-helps-on-your-nlp-model-83dd1b836f46

Segmentation Approaches

- Wordpiece
  - Original

1. ” 京都 清水寺の写真” (original text)
2. ” 京都 清水寺 の写真” (after segmentation)
3. ” __京都__ __清水寺 の写真__” (after addition of under-scores, used for LM training, dictionary etc.)
4. ” __京都__ __清水寺 の写真__” (decoder result)
5. ” 京都 清水寺の写真” (displayed output)

  - Ours

- **Word**: Jet makers feud over seat width with big orders at stake
- **wordpieces**: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

Segmentation Approaches

- Wordpiece
  - 또한 데이터에 따라 기본 char수를 관리할 수 있는 수로 줄이고 (서구 언어의 경우 500, 아시아 언어의 경우 더 많음)
  - 나머지 를 special unkonwn char로 mapping한다. 왜? 매우 드문 char로 주어진 wordpiece vocab이 오염되는 것을 방지하기 위해서
  - 8k~32k를 사용하는 것이 BLEU, decoding speed 측면에서 굉장히 좋았음
  - 위에서 언급한 바와 같이, 종종 드문 entity name 혹은 number를 source에서 target으로 바로 copy하는게 보통임
  - direct copy를 사용하기 위해 source language, target language에 share오후 8:11 2021-06-13d wordpiece model 사용
  - 위로 하여 같은 string은 바로 복사될 수 있게 해줌
  - wordpiece는 char의 유연성과 word의 효율성 사이에 벨런스를 갖춤
  - WPM은 resorting to characters only를 제외한 무한한 vocab을 가지고 효율적으로 처리할 수 있기에 효과적?
  - resorting to characterse only는 평균 인풋 아웃풋 길이를 더 길게 만들어주고 더 많은 연산을 요구함

Segmentation Approaches

- Mixed Word/Character Model
  - word -> fixed-size vocab 사용 | 일반적인 word와 다르게 OOV를 constituent characters로 치환
  - 각 char에 special prefix를 붙일거임! 왜냐? 1) 단어에서 char 위치에 대한 정보 2) in-vocab char에서 구분지으려고
  - <B>, <M>, E> tag를 사용하여 붙일 것임
  - 이 처리는 source, target 둘 다에서 처리됨
  - decoding 중, 출력은 오직 special tokens들의 sequence만 포함하게 됨
  - prefixes로 post-processing에서 복원함

Training Criteria

Training Criteria

- Maximum Likelihood Estimation + REINFORCE

$$D = \{(X^{(i)}, Y^{*(i)})\}_{i=1}^{N}$$

$$\mathcal{O}_{ML}(\theta) = \sum_{i=1}^{N} \log P_\theta(Y^{*(i)}|X^{(i)})$$

Task reward function (e.g., BLEU)를 반영하지 못함

Incorrect output sequence는 학습 중에 나타나지 않음…

즉, decoding 중 Error에 강건해지는 법을 학습할 수 없음!

→ Training / Testing Procedure이 다르기 때문 (Exposure Bias)

$$\mathcal{O}_{RL}(\theta) = \sum_{i=1}^{N} \sum_{Y \in \mathcal{Y}} P_\theta(Y^{*(i)}|X^{(i)})\, r(Y, Y^{*(i)})$$

$$\mathcal{O}_{mixed}(\theta) = \alpha * \mathcal{O}_{ML}(\theta) + \mathcal{O}_{RL}(\theta)$$

$$GLEU = \min(recall, precision)$$

RL setting에서 실용적으로 reward – mean(reward)해줌 (actor-critic)

평균은 분포에서 독립적으로 추출한 m개의 sequence의 평균으로 추정 (보통 m=15)

Alpha=0.0017

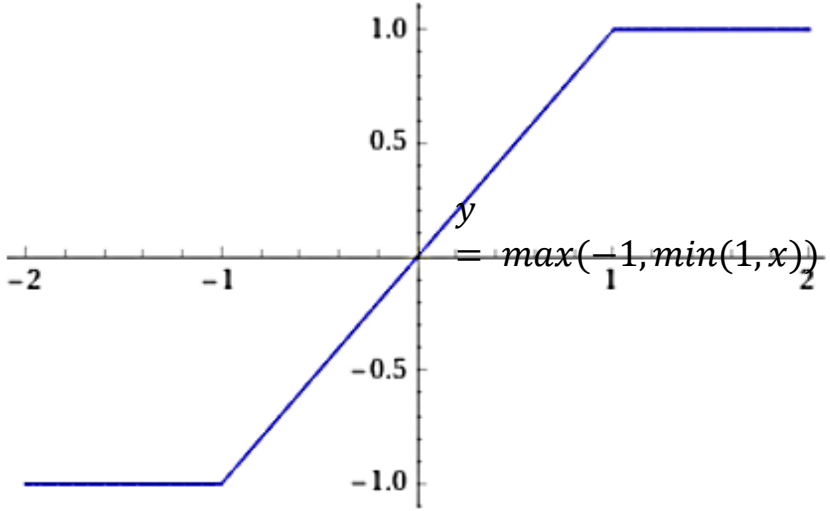
MLE가 수렴할 때까지 모델 학습을 실시하고 mixed objective로 BLEU score가 dev set에서 개선을 멈추기 전까지 학습

Quantization

Quantizable Model and Quantized Inference

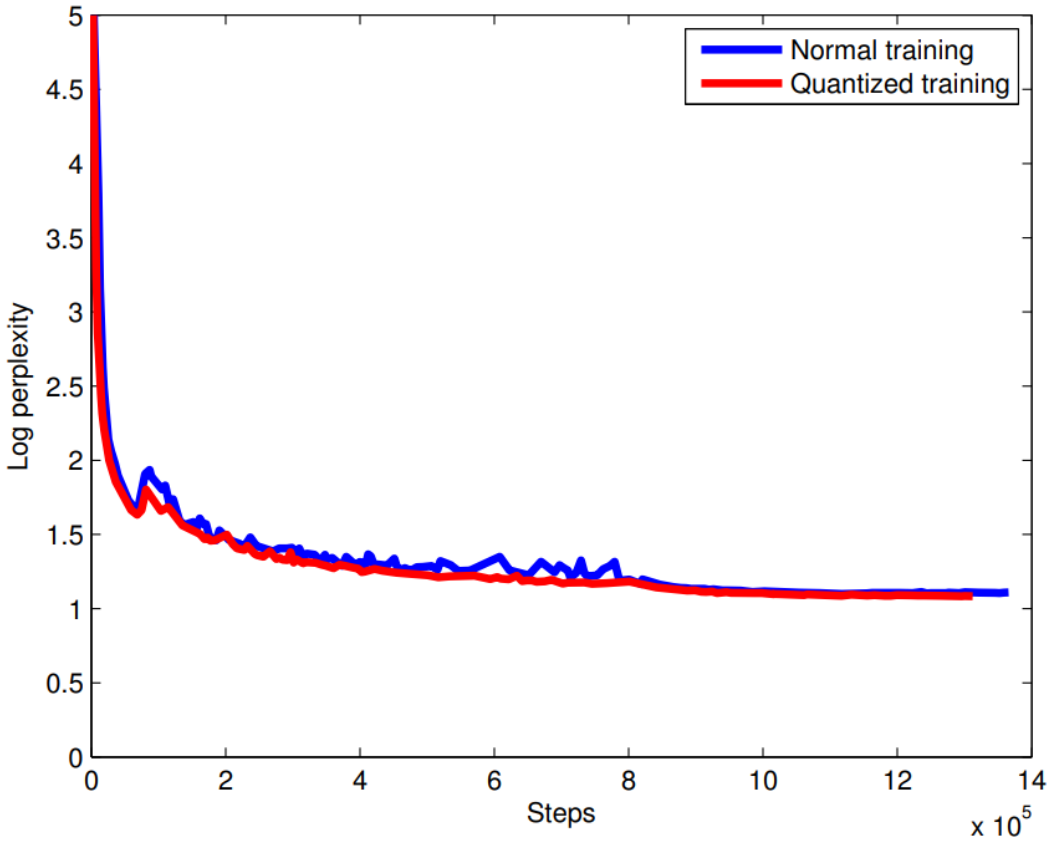
- Inference time을 늘리기 위한 전략: Quantization
- Deep LSTM with long sequences → Quantization Errors를 크게 줄일 수 있을 것인가?
- Proposed methodology: quantized arithmetic으로 inference 속도를 올리자! @TPU
- Quantization: 실수 값을 가지는 parameter를 정수 값으로 mapping 시켜서 표현
- 추론 시 모든 activation 및 hidden states를 quantize하기 위해 LSTM 모델에 제약을 추가

$$
\begin{aligned}
\mathbf{c}_t^{\prime i}, \mathbf{m}_t^i &= LSTM_i(\mathbf{c}_{t-1}^i, \mathbf{m}_{t-1}^i, \mathbf{x}_t^{i-1}; \mathbf{W}^i) \\
\mathbf{c}_t^i &= \max(-\delta, \min(\delta, \mathbf{c}_t^{\prime i})) \\
\mathbf{x}_t^{\prime i} &= \mathbf{m}_t^i + \mathbf{x}_t^{i-1} \\
\mathbf{x}_t^{\prime i} &= \max(-\delta, min(\delta, \mathbf{x}_t^{\prime i}) \\
\mathbf{c}_t^{i+1}, \mathbf{m}_t^{i+1} &= LSTM_{i+1}(\mathbf{c}_{t-1}^{i+1}, \mathbf{m}_{t-1}^{i+1}, \mathbf{x}_t^i; \mathbf{W}^{i+1}) \\
\mathbf{c}_t^{\prime i+1} &= \max(\delta, min(\delta, \mathbf{c}_t^{\prime i+1}))
\end{aligned}
\tag{10}
$$



Quantizable Model and Quantized Inference

- Quantization
  - Regularization 효과



Quantizable Model and Quantized Inference

- 학습이 끝난 모델의 weight matri를 8-bits 양자화하여 저장

$$\mathbf{s}_i = \max(\text{abs}(\mathbf{W}[i,:]))$$
$$\mathbf{WQ}[i,j] = \text{round}(\mathbf{W}[i,j]/\mathbf{s}_i \times 127.0)$$

- 이후 inference 시 거의 모든 연산은 quantized integer operation을 사용

- 8-bit 또는 16-bit의 fixed-point integer operation으로 표현 및 연산

$$\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{W}_6, \mathbf{W}_7, \mathbf{W}_8]$$
$$\mathbf{i}_t = \text{sigmoid}(\mathbf{W}_1\mathbf{x}_t + \mathbf{W}_2\mathbf{m}_t)$$
$$\mathbf{i'}_t = \tanh(\mathbf{W}_3\mathbf{x}_t + \mathbf{W}_4\mathbf{m}_t)$$
$$\mathbf{f}_t = \text{sigmoid}(\mathbf{W}_5\mathbf{x}_t + \mathbf{W}_6\mathbf{m}_t)$$
$$\mathbf{o}_t = \text{sigmoid}(\mathbf{W}_7\mathbf{x}_t + \mathbf{W}_8\mathbf{m}_t)$$
$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i'}_t \odot \mathbf{i}_t$$
$$\mathbf{m}_t = \mathbf{c}_t \odot \mathbf{o}_t$$

: 8-bit resolution

: 16-bit resolution

Quantizable Model and Quantized Inference

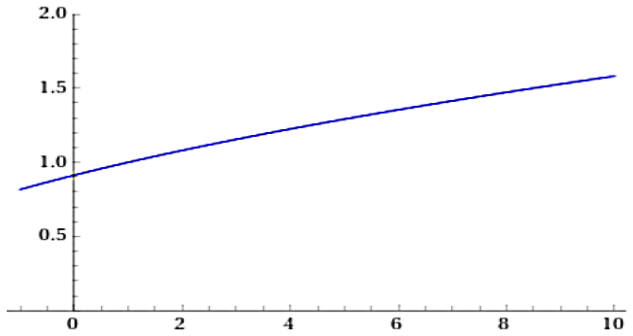
- 추론 속도 비교시, Quantization을 했을 때(TPU) loss function 값이 거의 변하지 않으면서 (+0.0072), 추론 시간은 현저히 빨라짐

	BLEU	Log Perplexity	Decoding time (s)
CPU	31.20	1.4553	1322
GPU	31.20	1.4553	3028
TPU	31.21	1.4626	384

Decoder

Decoder

- Beam Search
  - Max-probability based method
  - Refinement algorithm : coverage penalty + length normalization
  - Length normalization
    - $lp(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}}$
    - Beam Search는 보통 짧은 문장을 선호함. (왜? 각 step에서 log-probability가 더해지기에 긴 문장에 대해선 낮은 score를 가지게 됨)

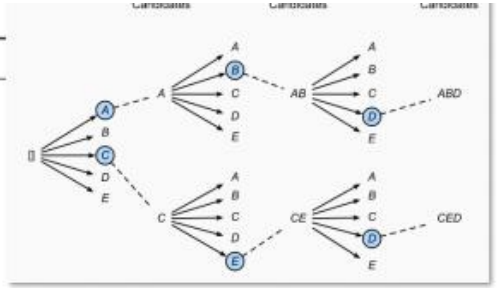


Time synchronous approximate search algorithm! (가설을 left-to-right로 build)

Algorithm 1 BeamSearch($\mathbf{x}, n \in \mathbb{N}_+$)

Input: $\mathbf{x}$: Source sentence, n: Beam size

$\mathcal{H}_{cur} \leftarrow \{(\epsilon, 0.0)\}$ {Initialize with empty translation prefix and zero score}
repeat
  $\mathcal{H}_{next} \leftarrow \emptyset$
  for all $(\mathbf{y}, p) \in \mathcal{H}_{cur}$ **do**
    if $y_{|\mathbf{y}|} = < /\mathbf{s} >$ **then** EOS 토큰 등장 시 다른 가설을 탐색하지 않음
      $\mathcal{H}_{next} \leftarrow \mathcal{H}_{next} \cup \{(\mathbf{y}, p)\}$ {Hypotheses ending with $< /\mathbf{s} >$ are not expanded}
    else
      $\mathcal{H}_{next} \leftarrow \mathcal{H}_{next} \cup \bigcup_{w \in \mathcal{T}} (\mathbf{y} \cdot w, p + \log P(w|\mathbf{x}, \mathbf{y}))$ {Add all possible continuations}
    end if
  end for
  모든 target word에 대해 가능한 continuation들을 탐색
  $\mathcal{H}_{cur} \leftarrow \{(\mathbf{y}, p) \in \mathcal{H}_{next} : |\{(\mathbf{y}', p') \in \mathcal{H}_{next} : p' > p\}| < n\}$ {Select n-best}
  Beam Size
  $(\tilde{\mathbf{y}}\ \tilde{p}) \leftarrow \arg\max_{(\mathbf{y}, p) \in \mathcal{H}_{cur}} p$
until $\tilde{y}_{|\tilde{\mathbf{y}}|} = < /\mathbf{s} >$
return $\tilde{\mathbf{y}}$



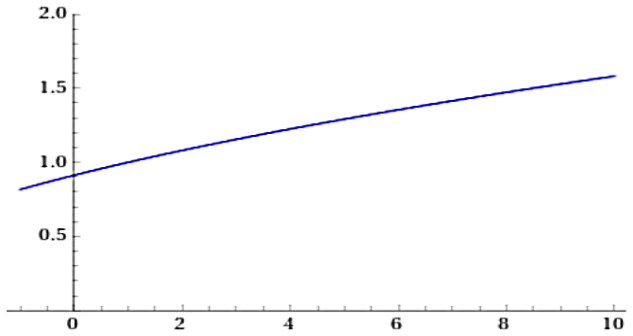
$\gamma \leq \log P(\hat{y}|x)$

$\tilde{p}_{beam\ search} := \gamma$

Note That〉 Beam Search Score는 Global Score의 Lower Bound!

Decoder

- Beam Search
  - Max-probability based method
  - Refinement algorithm : coverage penalty + length normalization
  - Length normalization
    - $lp(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}}$
    - Beam Search는 보통 짧은 문장을 선호함. (왜? 각 step에서 log-probability가 더해지기에 긴 문장에 대해선 낮은 score를 가지게 됨)
  - Coverage Penalty
    - $cp(X;Y) = \beta * \sum_{i=1}^{|X|} \log\left(\min\left(\sum_{j=1}^{|Y|} p_{i,j}, 1.0\right)\right)$



$$s(Y,X) = \log(P(Y|X))/lp(Y) + cp(X;Y)$$

$$lp(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}}$$

$$cp(X;Y) = \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0)),$$

Decoder

w/o RL

BLEU		α = 0.0	0.2	0.4	0.6	0.8	1.0
β	0.0	30.3	30.7	30.9	31.1	31.2	31.1
	0.2	31.4	31.4	31.4	31.3	30.8	30.3
	0.4	31.4	31.4	31.4	31.1	30.5	29.6
	0.6	31.4	31.4	31.3	30.9	30.1	28.9
	0.8	31.4	31.4	31.2	30.8	29.8	28.1
	1.0	31.4	31.3	31.2	30.6	29.4	27.2

with RL

BLEU		α = 0.0	0.2	0.4	0.6	0.8	1.0
β	0.0	0.320	0.321	0.322	0.322	0.322	0.322
	0.2	0.322	0.322	0.322	0.322	0.321	0.321
	0.4	0.322	0.322	0.322	0.321	0.321	0.316
	0.6	0.322	0.322	0.321	0.321	0.319	0.309
	0.8	0.322	0.322	0.321	0.321	0.316	0.302
	1.0	0.322	0.321	0.321	0.320	0.313	0.295

- Alpha=0.2, beta=0.2
- REINFORCE 적용이 더 큰 효과를 보였음

Experiments and Results

Experiments and Results

8.1 Datasets

We evaluate our model on the WMT En→Fr dataset, the WMT En→De dataset, as well as many Google-internal production datasets. On WMT En→Fr, the training set contains 36M sentence pairs. On WMT En→De, the training set contains 5M sentence pairs. In both cases, we use newstest2014 as the test sets to compare against previous work [31, 37, 45]. The combination of newstest2012 and newstest2013 is used as the development set.

In addition to WMT, we also evaluate our model on some Google-internal datasets representing a wider spectrum of languages with distinct linguistic properties: English ↔ French, English ↔ Spanish and English ↔ Chinese.

8.2 Evaluation Metrics

We evaluate our models using the standard BLEU score metric. To be comparable to previous work [41, 31, 45], we report tokenized BLEU score as computed by the `multi-bleu.pl` script, downloaded from the public implementation of Moses (on Github), which is also used in [31].

As is well-known, BLEU score does not fully capture the quality of a translation. For that reason we also carry out side-by-side (SxS) evaluations where we have human raters evaluate and compare the quality of two translations presented side by side for a given source sentence. Side-by-side scores range from 0 to 6, with a score of 0 meaning *"completely nonsense translation"*, and a score of 6 meaning *"perfect translation: the meaning of the translation is completely consistent with the source, and the grammar is correct"*. A translation is given a score of 4 if *"the sentence retains most of the meaning of the source sentence, but may have some grammar mistakes"*, and a translation is given a score of 2 if *"the sentence preserves some of the meaning of the source sentence but misses significant parts"*. These scores are generated by human raters who are fluent in both languages and hence often capture translation quality better than BLEU scores.

Experiments and Results

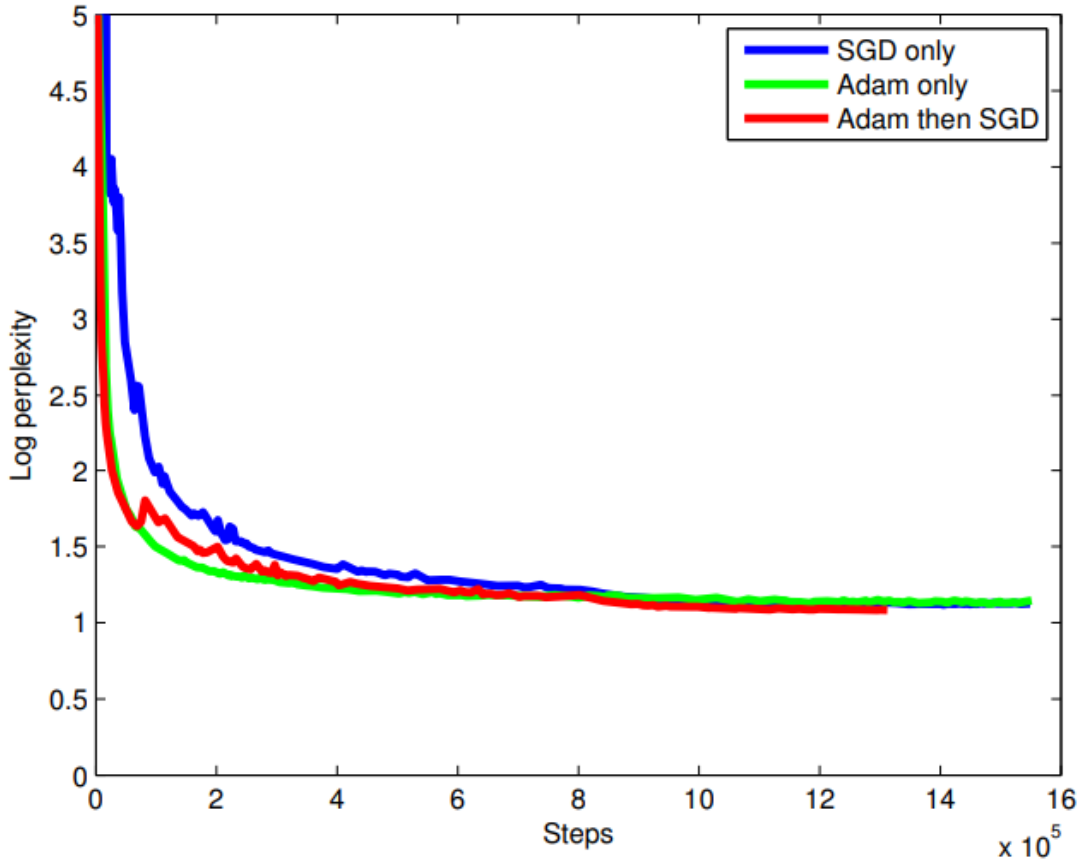


Figure 5: Log perplexity vs. steps for Adam, SGD and Adam-then-SGD on WMT En→Fr during maximum likelihood training. Adam converges much faster than SGD at the beginning. Towards the end, however, Adam-then-SGD is gradually better. Notice the bump in the red curve (Adam-then-SGD) at around 60k steps where we switch from Adam to SGD. We suspect that this bump occurs due to different optimization trajectories of Adam vs. SGD. When we switch from Adam to SGD, the model first suffers a little, but is able to quickly recover afterwards.

Experiments and Results

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	

Experiments and 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

Table 7: Model ensemble results on WMT En→Fr (newstest2014)

Model	BLEU
WPM-32K (8 models)	40.35
RL-refined WPM-32K (8 models)	41.16
LSTM (6 layers) [31]	35.6
LSTM (6 layers + PosUnk) [31]	37.5
Deep-Att + PosUnk (8 models) [45]	40.4

Table 8: Model ensemble results on WMT En→De (newstest2014). See Table 5 for a comparison against non-ensemble models.

Model	BLEU
WPM-32K (8 models)	26.20
RL-refined WPM-32K (8 models)	26.30

Experiments and Results

Table 9: Human side-by-side evaluation scores of WMT En→Fr models.

Model	BLEU	Side-by-side averaged score
PBMT [15]	37.0	3.87
NMT before RL	40.35	4.46
NMT after RL	41.16	4.44
Human		4.82

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%

Recent Advances in Google Translate