Productization of Neural Machine Translation System

Figure 1:

Productization

이전까지의 챕터들에서 우리는 자연어처리에 대한 다양한 이론과 실습 예제들을 다루었습니다. 이제 이러한 자연어처리 알고리즘들이 어떻게 실제 필드에서 결합되어 적용되고 있는지 살펴보고자 합니다. Google, Microsoft와 같은 여러 회사들도 자신들의 기술력을 자랑하기 위하여 앞다투어 자신들이 적용한 자연어처리(기계번역) 시스템에 대한 논문을 아주 상세하게 논문에 공개 합니다. 많은 회사들이 자신들의 상용화 경험을 토대로 성과를 자랑하는 논문을 써 내고 있습니다. 우리는 자연어처리 분야의 가장 성공적인 상용화분야인 기계번역의 상용화 사례에 대해서 다루어 보도록 하겠습니다. 이제 기계번역은 2014년도의 큰

충격 이후로부터 수많은 응용 방법들이 쏟아져나왔고, 어느정도 표준적인 방법 A to Z가 정립되어 가고 있는 시점입니다.

사실, 요즈음과 같이 새로운 기술에 대한 논문 및 소스코드 공개가 당연시되는 현상 아래에서는 각 기계번역 시스템마다의 수준의 편차가 그렇게 크지 않습니다. 따라서 누가 최신 기술을 구현했느냐보다, 상용화에 대한 경험과 연륜, 데이터의 양과 품질(정제 이슈)에 따라 성능이 결정될 수 있습니다.

아래는 현재 대표적인 기계번역 시스템의 현 주소를 살펴볼 수 있는 간단한 샘플 문장입니다. (2018년 04월 기준)

> 차를 마시러 공원에 가던 차 안에서 나는 그녀에게 차였다. - Google: I was kicking her in the car that went to the park for tea. - Microsoft: I was a car to her, in the car I had a car and went to the park. - Naver: I got dumped by her on the way to the park for tea. - Kakao: I was in the car going to the park for tea and I was in her car. - SK: I got dumped by her in the car that was going to the park for a cup of tea.

위와 같이 어려운 문장(문장구조는 쉬우나 중의성에 대해서 높은 난이도)에 대해서 어찌보면 갈 길이 멀기도 합니다. 따라서, 위와 같이 누군가 저런 어려운 문제에 대한 한단계 더 높은 기술력을 가졌다기보단, 기술의 수준은 비슷하다고 볼 수 있습니다.

Pipeline for Machine Translation

이번 섹션에서는 실제 기계번역 시스템을 구축하기 위한 절차와 시스템 Pipeline이 어떻게 구성되는지 살펴보도록 하겠습니다. 이 과정들은 대부분 기계번역 뿐만 아니라 기본적인 자연어처리 문제 전반에 적용 가능하다고 할 수 있습니다. 그리고 실제 Google 등에서 발표한 논문을 통해서 그들의 상용 번역 시스템의 실제 구성에 대해서 살펴보도록 하겠습니다.

통상적으로 번역시스템을 구축하면 아래와 같은 흐름을 가지게 됩니다.

준비과정

1. Corpus 수집, 정제
    - 병렬 말뭉치(parallel corpus)를 다양한 소스에서 수집합니다. WMT등 번역 시스템 평가를 위해 학술적으로 공개 된 데이터 셋도 있을 뿐더러,

뉴스 기사, 드라마/영화 자막, 위키피디아 등을 수집하여 번역 시스템에 사용 할 수 있습니다.
    - 수집된 데이터는 정제 과정을 거쳐야 합니다. 정제 과정에는 양 언어의 말뭉치에 대해서 문장 단위로 정렬을 시켜주는 작업부터, 특수문자 등의 noise를 제거해 주는 작업도 포함 됩니다.
2. Tokenization
    - 각 언어 별 POS tagger 또는 segmenter를 사용하여 띄어쓰기를 정제(normalization) 합니다. 영어의 경우에는 대소문자에 대한 정제 이슈가 있을 수도 있습니다. 한국어의 경우에는 한국어의 특성 상, 인터넷에 공개 되어 있는 corpus 들은 띄어쓰기가 제멋대로일 수 있습니다.
    - 한국어의 경우에는 Mecab과 같은 공개 되어 있는 파서(parser)들이 있습니다.
    - 띄어쓰기가 정제 된 이후에는 Byte Pair Encoding(BPE by Subword or Wordpiece)를 통해 추가적인 tokenization을 수행하고 어휘 목록을 구성합니다.
3. Batchfy
    - 전처리 작업이 끝난 corpus에 대해서 훈련을 시작하기 위해서 mini-batch로 만드는 작업이 필요합니다.
    - 여기서 중요한 점은 mini-batch 내의 문장들의 길이를 최대한 통일시켜 주는 것 입니다. 이렇게 하면 문장 길이가 다름으로 인해서 발생하는 훈련 시간 낭비를 최소화 할 수 있습니다.
    - 예를 들어 mini-batch 내에서 5단어 짜리 문장과 70단어 짜리 문장이 공존할 경우, 5단어 짜리 문장에 대해서는 불필요하게 65 time-step을 더 진행해야 하기 때문입니다. 따라서 5단어 짜리 문장끼리 모아서 mini-batch를 구성하면 해당 batch에 대해서는 훨씬 수행 시간을 줄일 수 있습니다.
    - 실제 훈련 할 때에는 이렇게 구성된 mini-batch들의 순서를 임의로 섞어(shuffling) 훈련하게 됩니다.
4. Training
    - 준비된 데이터셋을 사용하여 seq2seq 모델을 훈련 합니다.
5. Inference
    - 성능 평가(evaluation)를 위한 추론(inference)을 수행 합니다. 이때에는 준비된 테스트셋(훈련 데이터셋에 포함되어 있지 않은)을 사용하여 추론을 수행합니다.
6. De-tokenization
    - 추론 과정이 끝나더라도 tokenization이 되어 있어 아직 사람이 실제 사용하는 문장 구성과 형태가 다릅니다. 따라서 detokenization을

수행하면 실제 사용되는 문장의 형태로 반환 됩니다.

7. Evaluation
    - 이제 이렇게 얻어진 문장에 대해서 정량평가를 수행 합니다. 기계번역을 위한 정량평가 방법으로는 BLEU가 있습니다. 비교대상의 BLEU점수와 비교하여 어느 모델이 더 우월한지 알 수 있습니다.
    - 정량평가에서 기존 대비 또는 경쟁사 대비 더 우월한 성능을 갖추었음을 알게 되면, 정성평가를 수행 합니다. 번역의 경우에는 사람이 직접 비교대상들의 결과들과 비교하여 어느 모델이 우월한지 평가 합니다.
    - 정량평가와 정성평가 모두 성능이 개선되었음을 알게 되면 이제 서비스에 적용 가능 합니다.

서비스

1. API 호출 or 사용자로부터의 입력
    - 대부분 서비스 별 API 서버를 만들어 실제 프론트앤드(front-end)로부터 API 호출을 받아 프로세스를 시작 합니다.
    - 서비스의 스케일(scale)에 따라 load-balancer등을 두어 부하를 적절히 분배하기도 합니다.
    - 서비스의 형태나 성격에 따라서 실제 모델 추론(inference)을 수행하지 않고, 정해진 응답을 반환하기도 합니다.
2. Tokenization
    - 추론을 위해서 실제 모델에 훈련된 데이터셋과 동일한 형태로 tokenization을 수행 합니다.
    - 언어에 따라서 띄어쓰기 정제를 위한 tokenizer를 사용합니다. (예: 한국어의 경우에는 Mecab)
    - 띄어쓰기가 정제 된 이후에는 Byte Pair Encoding(BPE by Subword or Wordpiece)를 통해 subword 형태로 추가적인 tokenization을 해 줄 수 있습니다.
3. Inference
    - 정량/정성 평가를 통해 성능이 입증된 모델을 통해 추론을 수행 합니다.
4. De-tokenization
    - 다시 사람이 읽을 수 있는 형태로 detokenization을 수행 합니다.
5. API 결과 반환 or 사용자에게 결과 반환
    - 최종 결과물을 API 서버에서 반환하여 서비스 프로세스를 종료 합니다. # Google Neural Machine Translation (GNMT) (Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation)

Google은 2016년 논문([Wo at el.2016])을 발표하여 그들의 번역시스템에 대해서 상세히 소개하였습니다. 실제 시스템에 적용된 모델 구조(architecture)부터 훈련 방법까지 상세히 기술하였기 때문에, 실제 번역 시스템을 구성하고자 할 때에 좋은 참고자료(reference)가 될 수 있습니다. 또한 다른 논문들에서 실험 결과에 대해 설명할 때, GNMT를 baseline으로 참조하기도 합니다. 아래의 내용들은 그들의 논문에서 소개한 내용을 다루도록 하겠습니다.

Model Architecture

Google도 seq2seq 기반의 모델을 구성하였습니다. 다만, 구글은 훨씬 방대한 데이터셋을 가지고 있기 때문에 그에 맞는 깊은 모델을 구성하였습니다. 따라서 아래에 소개될 방법들이 깊은 모델들을 효율적으로 훈련 할 수 있도록 사용되었습니다.

Residual Connection

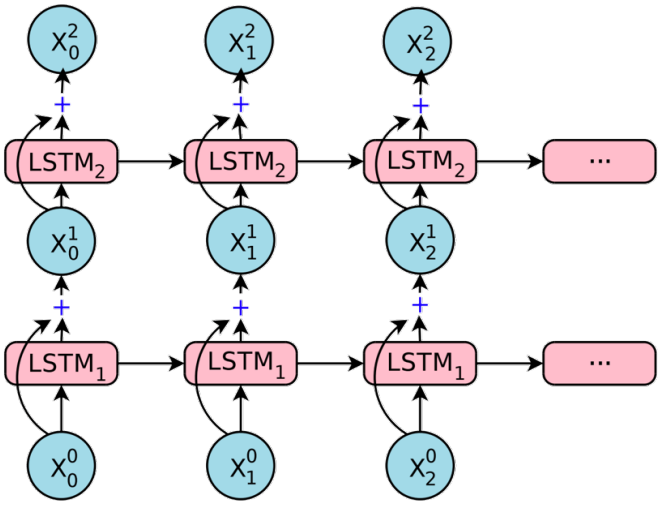

Figure 2:

보통 LSTM layer를 4개 이상 쌓기 시작하면 모델이 더욱 깊어(deeper)짐에 따라서 성능 효율이 저하되기 시작합니다. 따라서 Google은 깊은 모델은 효율적으로 훈련시키기 위하여 residual connection을 적용하였습니다.

Bi-directional Encoder for First Layer

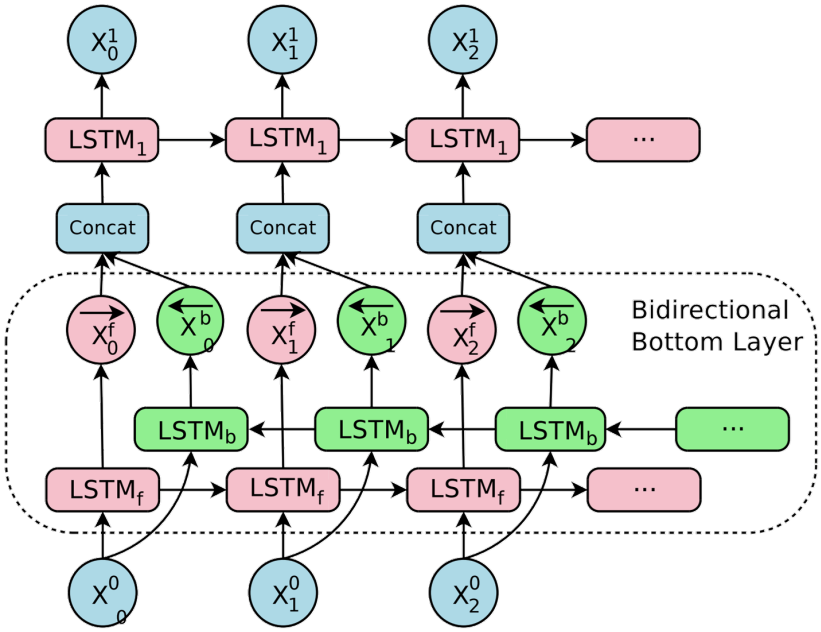

Figure 3:

또한, 모든 LSTM stack에 대해서 bi-directional LSTM을 적용하는 대신에, 첫번째 층에 대해서만 bi-directional LSTM을 적용하였습니다. 따라서 훈련(training) 및 추론(inference) 속도에 개선이 있었습니다.

Segmentation Approachs

Wordpiece Model

구글도 마찬가지로 BPE 모델을 사용하여 tokenization을 수행하였습니다. 그리고 그들은 그들의 tokenizer를 오픈소스로 공개하였습니다. - SentencePiece: https://github.com/google/sentencepiece 마찬가지로 아래와 같이 띄어쓰기는 underscore로 치환하고, 단어를 subword별로 통계에 따라 segmentation 합니다.

- original: 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

Training Criteria

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

Figure 4:

Google은 강화학습을 다룬 챕터에서 설명한 강화학습 기법을 사용하여 Maximum Likelihood Estimation (MLE)방식의 훈련된 모델에 fine-tuning을 수행하였습니다. 따라서 위의 테이블과 같은 추가적이 성능 개선을 얻어낼 수 있었습니다.

기존 MLE 방식의 목적함수(objective)를 아래와 같이 구성합니다. $Y^{*(i)}$은 최적(optimal)의 정답 데이터를 의미합니다.

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

여기에 추가로 RL방식의 목적함수(objective)를 추가하였는데 이 방식이 policy gradient 방식과 같습니다.

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

위의 수식도 Minimum Risk Training (MRT) 방식과 비슷합니다. $r(Y, Y^{*(i)})$ 또한 정답과 sampling 데이터 사이의 유사도(점수)를 의미합니다. 가장 큰 차이점은 기존에는 risk로 취급하여 최소화(minimize)하는 방향으로 훈련하였지만, 이번에는 reward로 취급하여 최대화(maximize)하는 방향으로 훈련하게 된다는 것 입니다.

이렇게 새롭게 추가된 목적함수(objective)를 아래와 같이 기존의 MLE방식의 목적함수와 선형 결합(linear combination)을 취하여 최종적인 목적함수가 완성됩니다.

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

이때에 α값은 주로 0.017로 셋팅하였습니다. 위와 같은 방법의 성능을 실험한 결과는 다음과 같습니다.

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

Figure 5:

$En \rightarrow De$의 경우에는 성능이 약간 하락함을 보였습니다. 하지만 이는 decoder의 length penalty, coverage penalty와 결합되었기 때문이고, 이 페널티(panalty)들이 없을 때에는 훨씬 큰 성능 향상이 있었다고 합니다.

Quantization

실제 인공신경망을 사용한 제품을 개발할 때에는 여러가지 어려움에 부딪히게 됩니다. 이때, Quantization을 도입함으로써 아래와 같은 여러가지 이점을 얻을 수 있습니다.

- 계산량을 줄여 자원의 효율적 사용과 응답시간의 감소를 얻을 수 있다.
- 모델의 실제 저장되는 크기를 줄여 deploy를 효율적으로 할 수 있다.
- 부가적으로 regularization의 효과를 볼 수 있다.

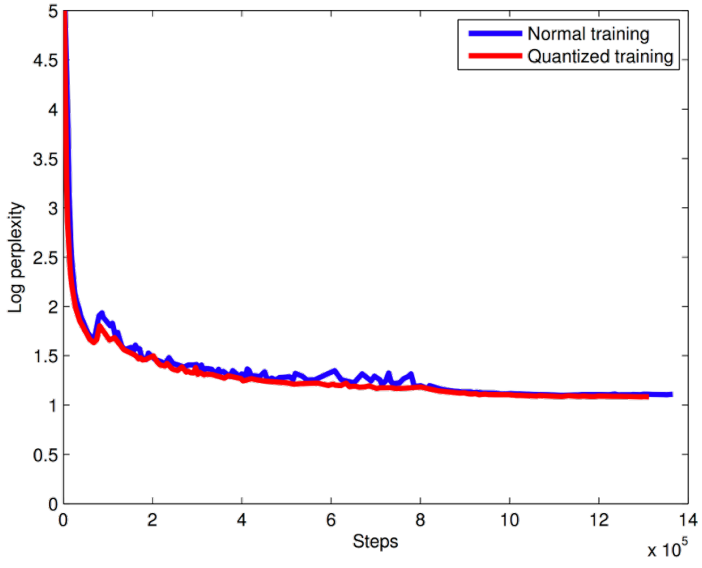

Figure 6:

위의 그래프를 보면 전체적으로 Quantized verion이 더 낮은 loss를 보여주는 것을 확인할 수 있습니다.

Search

Length Penalty and Coverage Penalty

Google은 기존에 소개한 Length Penalty에 추가로 Coverage Penalty를 사용하여 좀 더 성능을 끌어올렸습니다. Coverage penalty는 attention weight(probability)의 값의 분포에 따라서 매겨집니다. 이 페널티는 좀 더 attention이 고루 잘 퍼지게 하기 위함입니다.

$$
\begin{aligned}
s(Y,X) &= \log P(Y|X)/lp(Y) + cp(X;Y) \\
lp(Y) &= \frac{(5+|Y|)^\alpha}{(5+1)^\alpha} \\
cp(X;Y) &= \beta * \sum_{i=1}^{|X|} \log\left(\min\left(\sum_{j=1}^{|Y|} p_{i,j}, 1.0\right)\right)
\end{aligned}
$$

where $p_{i,j}$ is the attention weight of the j-th target word y_j on the i-th source word x_i.

Coverage penalty의 수식을 들여다보면, 각 source word x_i별로 attention weight의 합을 구하고, 그것의 평균(=합)을 내는 것을 볼 수 있습니다. 로그(log)를 취했기 때문에 그 중에 attention weight가 편중되어 있다면, 편중되지 않은 source word는 매우 작은 음수 값을 가질 것이기 때문에 좋은 점수를 받을 수 없을 겁니다.

실험에 의하면 α와 β는 각각 $0.6, 0.2$ 정도가 좋은것으로 밝혀졌습니다. 하지만, 상기한 강화학습 방식을 training criteria에 함께 이용하면 그다지 그 값은 중요하지 않다고 하였습니다.

Training Procedure

Google은 stochastic gradient descent (SGD)를 써서 훈련 시키는 것 보다, Adam과 섞어 사용하면 (epoch 1까지 Adam) 더 좋은 성능을 발휘하는 것을 확인하였습니다.

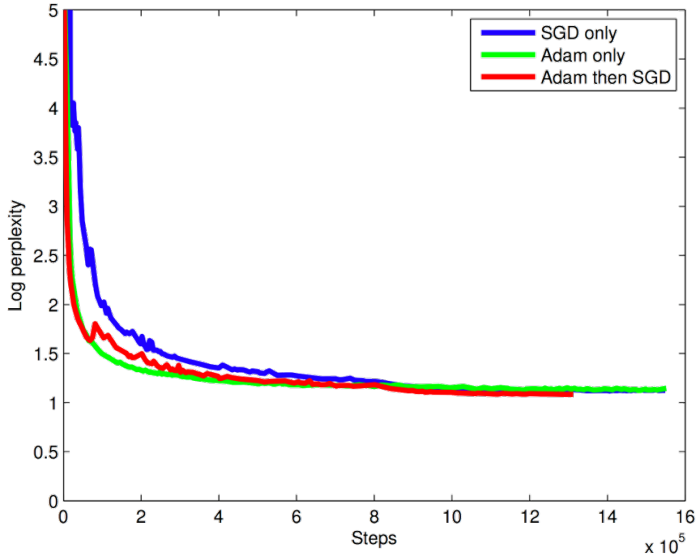

Figure 7:

Evaluation

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%

Figure 8:

실제 번역 품질을 측정하기 위하여 BLEU 이외에도 정성평가(implicit human evaluation)을 통하여 GNMT의 성능 개선의 정도를 측정하였습니다. 0(Poor)에서 6(Perfect)점 사이로 점수를 매겨 사람의 번역 결과 점수를 최대치로 가정하고 성능의 개선폭을 계산하였습니다. 실제 SMT 방식 대비 엄청난 천지개벽 수준의 성능 개선이 이루어진 것을 알 수 있고, 일부 언어쌍에 대해서는 거의 사람의 수준에 필적하는 성능을 보여주는 것을 알 수 있습니다.

The University of Edinburgh's Neural MT Systems for WMT17

사실 Google의 논문은 훌륭하지만 매우 스케일이 매우 큽니다. 저는 그래서 작은 스케일의 기계번역 시스템에 관한 논문은 이 논문[Sennrich at el.2017]을 높게 평가합니다. 이 논문도 기계번역 시스템을 구성할 때에 훌륭한 baseline이 될 수 있습니다. Edinburgh 대학의 Sennrich교수는 매년 열리는 WMT 대회에 참가하고 있고, 해당 대회에 참가하는 기계번역 시스템들은 이처럼 매년 자신들의 기술에 대한 논문을 제출합니다. 좋은 참고자료로 삼을 수 있습니다.

Subword Segmentation

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

r · → r·
l o → lo
lo w → low
e r· → er·

Figure 1: BPE merge operations learned from dictionary {'low', 'lowest', 'newer', 'wider'}.

[Sennrich at el.2016]

이 논문 또한 (그들이 처음으로 제안한 방식이기에) BPE 방식을 사용하여 tokenization을 수행하였습니다. 이제 우리는 subword 기반의 tokenization 방식이 하나의 정석이 되었음을 알 수 있습니다. 위의 code는 BPE algorithm에 대해서 간략하게 소개한 code 입니다. 전처리 챕터에서 소개했지만, subword 방식은 위와 같이 가장 많이 등장한 문자열(character sequence)에 대해서 합쳐주며 iteration을 반복하고, 원하는 어휘(vocabulary) 숫자가 채워질때가지 해당 iteration을 반복합니다.

Architecture

이 논문에서는 seq2seq를 기반으로 모델 구조(architecture)를 만들었는데, 다만 LSTM이 아닌 GRU를 사용하여 RNN stack을 구성하였습니다. Google과 마찬가지로 residual connection을 사용하여 stack을 구성하였고, encoder의 경우에는 4개층, decoder의 경우에는 8개 층을 쌓아 모델을 구성하였습니다. 실험 시에는 $hidden\ size = 1024$, $word\ vector\ dimension = 512$를 사용하였습니다. 또한, Google과는 다르게 순수하게 Adam만을 optimizer로 사용하여 훈련을 하였습니다.

Synthetic Data using Monolingual Data

이전 섹션에서 소개한 그들이 제안한 논문[Sennrich at el.2015]의 방식대로 back translation과 copied translation 방식을 사용하여 합성 병렬(pseudo parallel) corpus를 구성하여 훈련 데이터셋에 추가하였습니다. 이때에 비율은 실험결과에 따라서 $parallel : copied : back = 1 : 1 \sim 2 : 1 \sim 2$로 조절하여 사용하였습니다.

Ensemble

이 논문에서 그들은 2가지 앙상블(ensemble) 기법을 모두 사용하였습니다.

- checkpoint ensemble
  - 특정 epoch에서부터 다른 모델로 다시 훈련하여 ensemble을 구성합니다. 훈련 중간부터 다시 훈련하기 때문에 시간적으로 굉장히 효율적입니다.
- independent ensemble
  - 처음부터 다른 모델로 훈련하여 ensemble로 구성합니다. 처음부터 다시 훈련하므로 checkpoint 방식에 비해서 비효율적이지만, 다양성(diversity) 관점에서 낫습니다.
- Machine translation that makes sense: the Booking.com use case
- Machine Translation at Booking.com: Journey and Lessons Learned [Levin et al.2017] # Microsoft Machine Translation (Achieving Human Parity on Automatic Chinese to English News Translation)

2018년 3월에 나온 Microsoft의 기계번역 시스템에 대한 논문([Hassan et al.,2018])입니다. 2016년에 발표한 Google의 논문은 기계번역 자체의 기본 성능을 끌어올리는 모델 아키텍쳐와 훈련 방법 등의 내부 구조에 대해서 많은 설명을

할애하였던 것과 달리, Microsoft의 논문은 기술을 이미 끌어올려진 기술의 기반 위에서 더욱 그 성능을 견고히하는 방법에 대한 설명에 분량을 더 할애하였습니다.

이 논문은 중국어와 영어 간 기계번역 시스템을 다루고 있고, 뉴스 도메인(domain) 번역에 있어서 사람의 번역과 비슷한 성능에 도달하였다고 선언하고 있습니다. 다만, 제안한 방법에 의해 구성된 기계번역 시스템이 모든 언어쌍에 대해서, 모든 분야의 도메인에 대해서 같은 사람 번역 수준에 도달하지는 못할 수도 있다고 설명하고 있습니다.

Microsoft는 전통적인(?) RNN방식의 seq2seq 대신, Google의 Transformer 구조를 사용하여 seq2seq를 구현하였습니다. 이 논문에서 소개한 중점 기술은 아래와 같습니다.

- Back-translation과 Dual learning(Unsupervised and Supervised, both)을 통한 단방향(monolingual) corpora의 활용 극대화
- Auto-regressive 속성(이전 time-step의 예측(prediction)이 다음 time-step의 예측에 영향을 주는 것)의 단점을 보완하기 위한 Deliberation Networks([Xia et al.,2017])과 Kullback-Leibler (KL) divergence를 이용한 regularization
- NMT성능을 극대화 하기 위한 훈련 데이터 선택(selection)과 필터링(filtering)

위의 기술들에 대해서 한 항목씩 차례로 살펴보도록 하겠습니다.

Exploiting the Dual Nature of Translation

Dual Learning for NMT

이 논문에서는, 앞 챕터에서 설명한, duality를 활용하여 양방향(bilingual) corpus를 활용한 Dual Supervised Learning (DSL) [Xia et al.2017] 방식과, 단방향(monolingual) corpus를 marginal 분포(distribution)에 적용하여 활용한 Dual Unsupervised Learning (DUL) [Wang et al.2017] 방식을 모두 사용하였습니다.

Dual Unsupervised Learning (DUL)

$$\begin{aligned}\mathcal{L}(x;\theta_{x\to y}) &= E_{y\sim P(\cdot|x;\theta_{x\to y})}\{\log P(x|y;\theta_{y\to x})\} \\ &= \sum_{y} P(y|x;\theta_{x\to y}) \log P(x|y;\theta_{y\to x})\end{aligned}$$

$$\frac{\partial \mathcal{L}(x;\theta_{x\to y})}{\partial \theta_{x\to y}} = \sum_{y} \frac{\partial P(y|x;\theta_{x\to y})}{\partial \theta_{x\to y}} \log P(x|y;\theta_{y\to x})$$

Dual Supervised Learning (DSL)

Joint Training of Src2Tgt and Tgt2Src Models

Beyond the Left-to-Right Bias

Deliberation Networks

Microsoft는 [Xia et al.2017]에서 소개한 방식을 통해 번역 성능을 더욱 높이려 하였습니다. 이 방법은 사람이 번역을 하는 방법에서 영감을 얻은 것 입니다. 사람은 번역을 할 때에 source 문장에서 초안(draft) target 문장을 얻고, 그 초안으로부터 최종적인 target 문장을 번역 해 내곤 합니다. 따라서 신경망을 통해 같은 방법을 구현하고자 하였습니다.

예를 들어 기존의 번역의 수식은 deliberation networks를 통해 아래와 같이 바뀔 수 있습니다.

$$P(Y|X) = \prod P(y_i|X, y_{<i}) \longrightarrow P(Y|X) = \prod P(y_i|X, Y_{mid}, y_{<i})$$

Agreement Regularization of Left-to-Right and Right-to-Left Models

Data Selection and Filtering

Evaluation

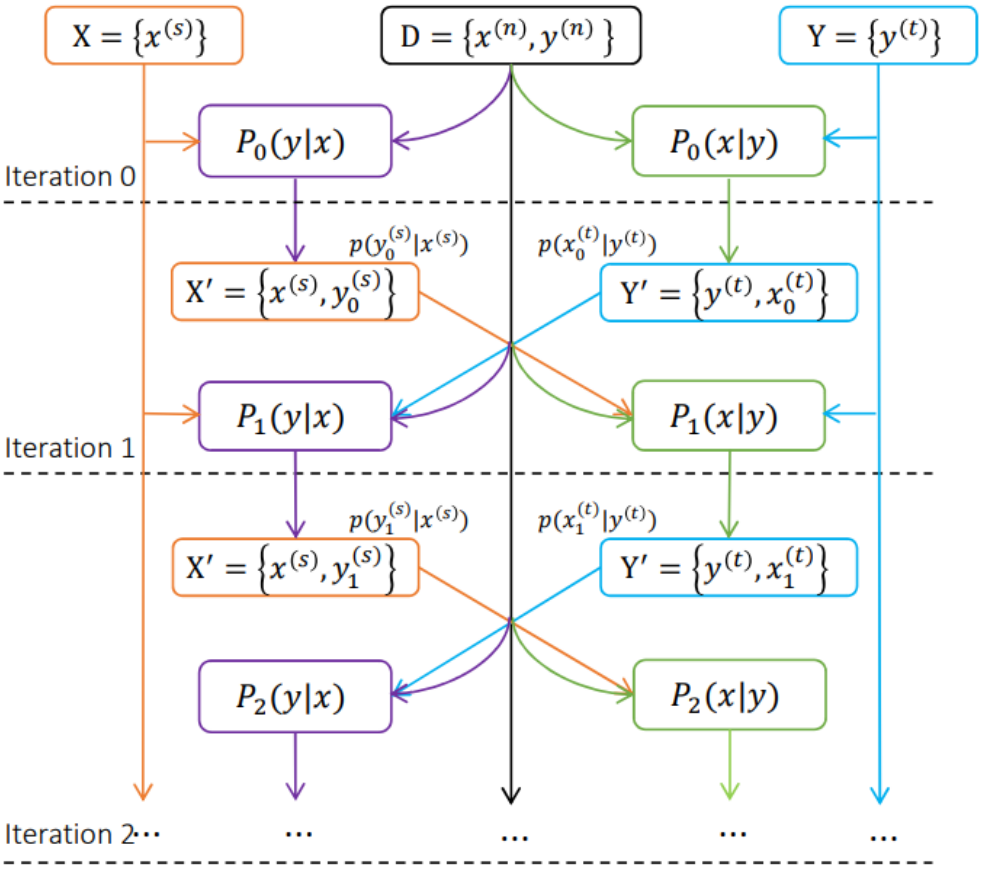

Figure 1: Illustration of joint training: S2T $p(\mathbf{y}|\mathbf{x})$ and T2S $p(\mathbf{x}|\mathbf{y})$

Figure 9:

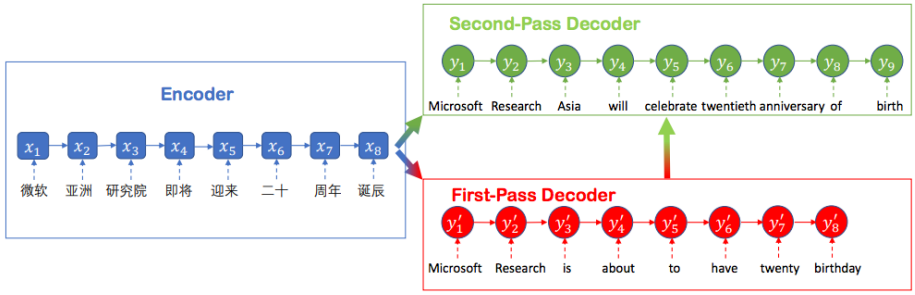

Figure 3: An example showing the decoding process of deliberation network.

Figure 10:

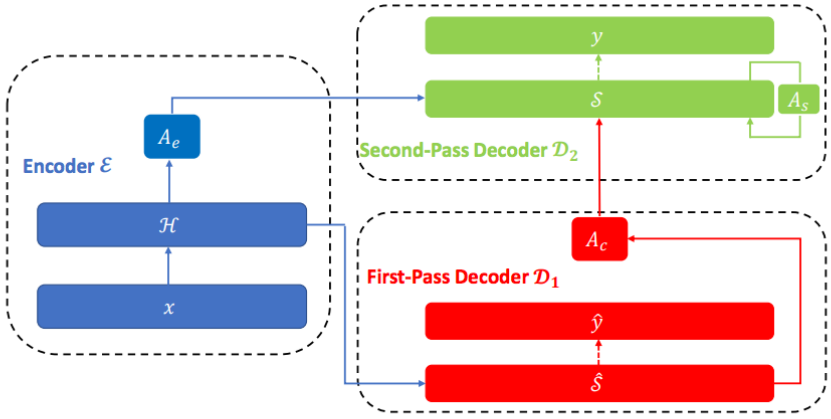

Figure 4: Deliberation network: Blue, red and green parts indicate encoder $\mathcal{E}$, first-pass decoder $\mathcal{D}_1$ and second-pass decoder $\mathcal{D}_2$ respectively. Solid lines represent the information flow via attention model. The self attention model within $\mathcal{E}$ and the $\mathcal{E}$-to-$\mathcal{D}_1$ attention model are omitted for readability.

Figure 11:

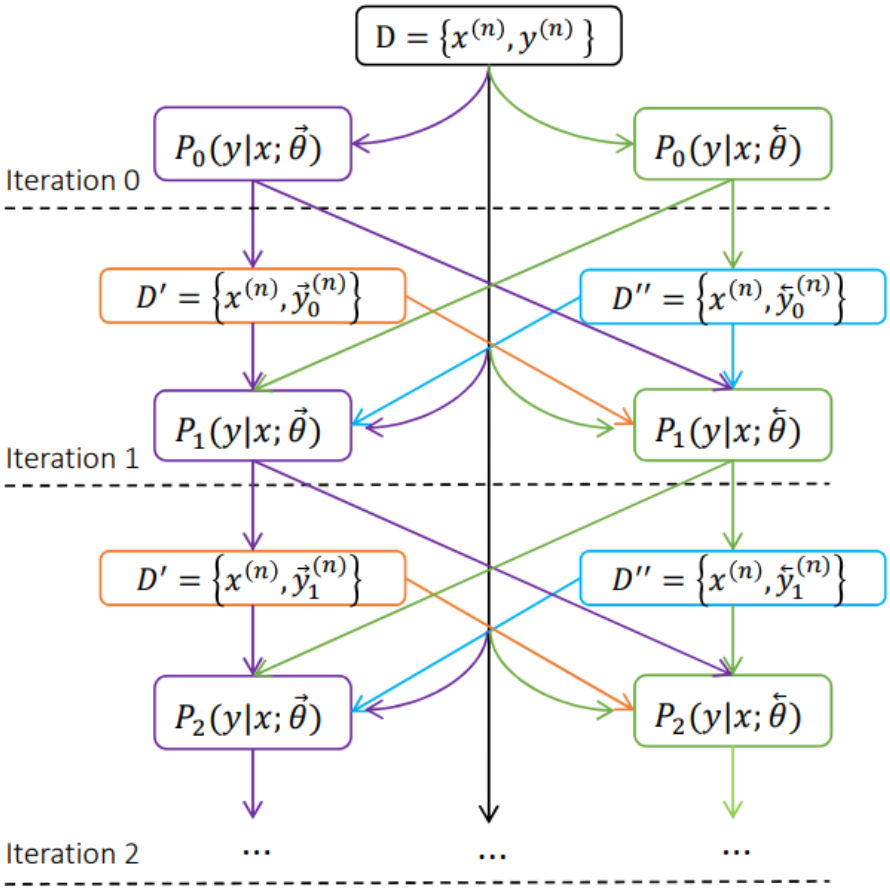

Figure 2: Illustration of agreement regularization: L2R $p(\mathbf{y}|\mathbf{x}; \overrightarrow{\theta})$ and R2L $p(\mathbf{y}|\mathbf{x}; \overleftarrow{\theta})$

Figure 12:

SystemID	Settings	BLEU
Sogou	WMT 2017 best result [42]	26.40
Base	Transformer Baseline	24.2
BT	+Back Translation	25.57
DL	BT + Dual Learning	26.51
DLDN	BT + Dual Learning + Deliberation Nets	27.40
DLDN2	DLDN without first decoder reranking	27.20
DLDN3	BT+ Dual Learning + R2L sampling	26.88
DLDN4	BT+ Dual Learning + Bi-NMT	27.16
AR	BT + Agreement Regularization	26.91
ARJT	BT + Agreement Regularization + Joint Training	27.38
ARJT2	ARJT + dropout=0.1	27.19
ARJT3	ARJT + dropout=0.05	27.07
ARJT4	ARJT + dropout=0.01	26.98

Table 1: Automatic (BLEU) evaluation results on the WMT 2017 Chinese-English test set

Figure 13: