Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation Cho. 2014

신성진

Overview

Abstract

Abstract

Key Points

RNN Encoder-Decoder

- RNN을 활용하여 다양한 길이의 문장 구절 학습
- RNN Encoder - Decoder 제안

기존의 Statistical MT의 성능을 개선

- Symentically & Synthetically

Introduction

RNN Encoder-Decoder

- Encoder는 유동적인 길이의 source sequence를 fixed-legnth vector로 변환함
- Decoder 는 vector representation을 유동적인 target sequence로 변환함

-> 주어진 source sequence에서 target sequence의 조건부 확률을 극대화 함

RNN Encoder-Decoder의 목적은 영어를 불어로 번역

- SMT를 보조하기 위해 활용

결국 RNN Encoder-Decoder 구조가 번역 성능을 높히고, Sementic과 Syntactic 구조도 잘 잡는 효과

Model

Model - RNN Basic

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

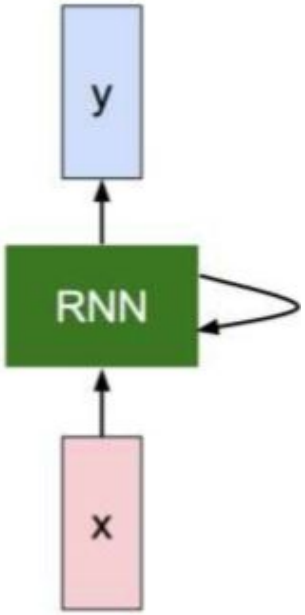
$$h_t = f_W(h_{t-1}, x_t)$$

new state

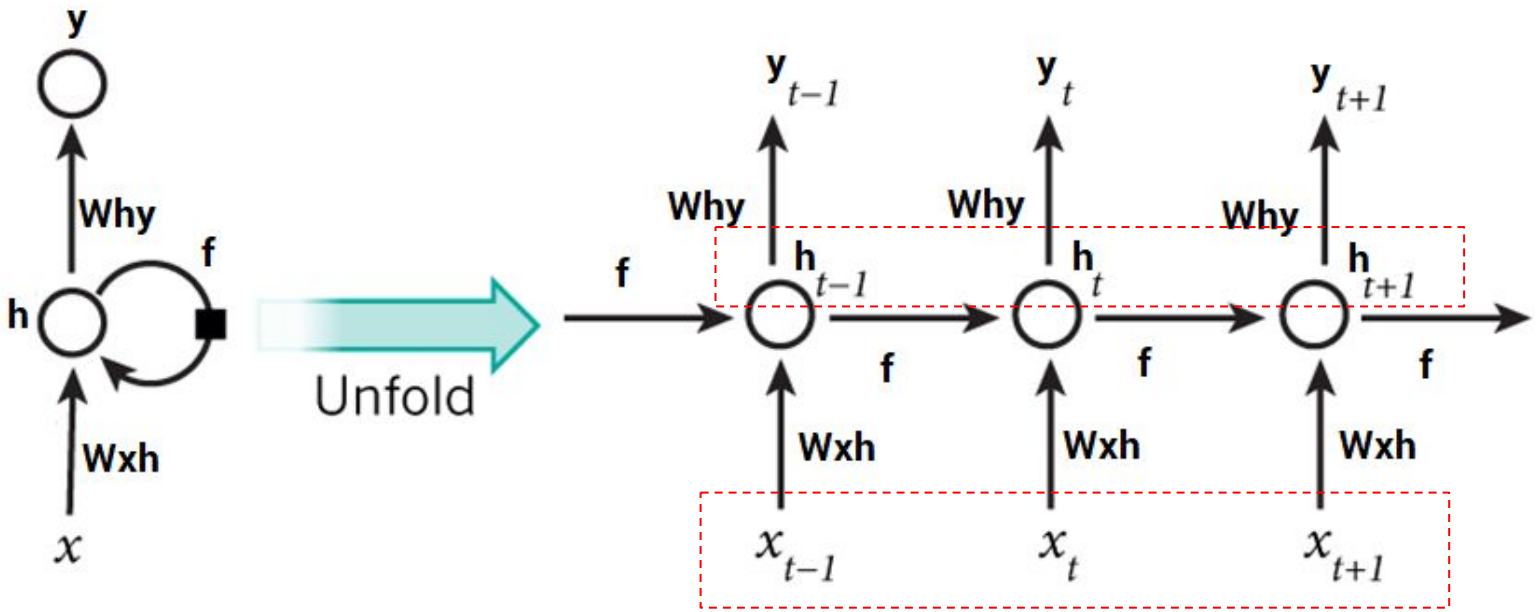
some function with parameters W

old state

input vector at some time step



Model - RNN Basic



$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right)$$

f는 non-linear activation function으로,
간단한 logistic sigmoid or 복잡한 LSTM

이전 Hidden State h와 현 input에
업데이트

Model - RNN Encoder Decoder

In General

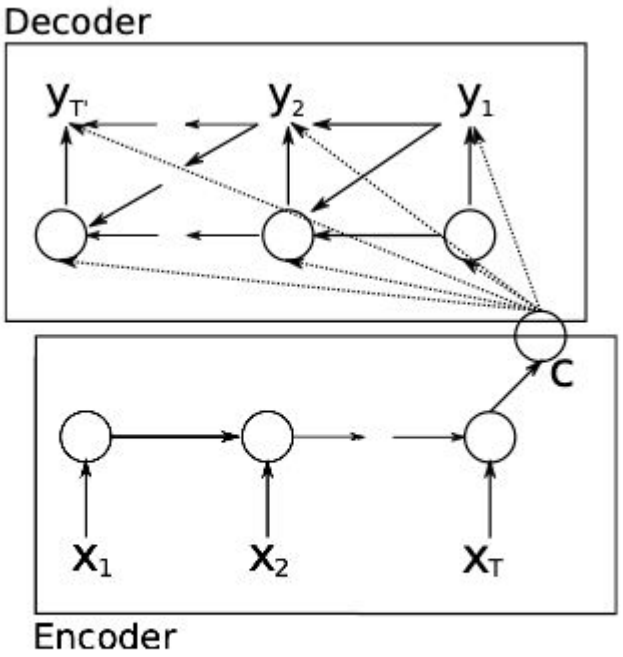


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

T'과 T는 다른 길이를 의미

$$p(y_1, \ldots, y_{T'} \mid x_1, \ldots, x_T)$$

입력값의 Seq.에 대한 decoder에 만들어진 Seq.의 조건부 분포를 학습

Model - RNN Encoder Decoder

RNN Encoder

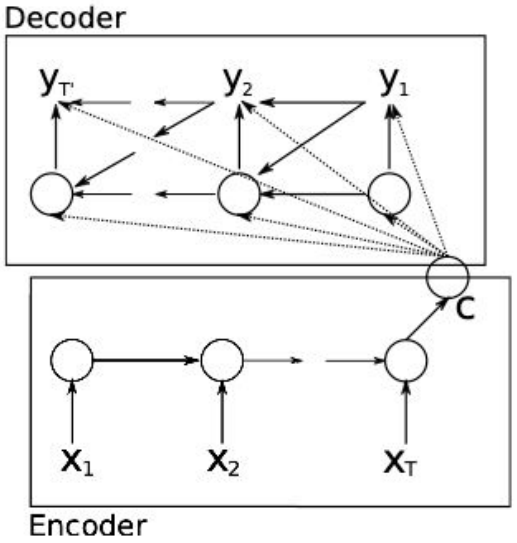


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right)$$

input Sequence X를 순서대로 읽고,
hidden state를 리턴

C: Encoder의 마지막 hidden state
전체 input을 요약

Model - RNN Encoder Decoder

RNN Decoder

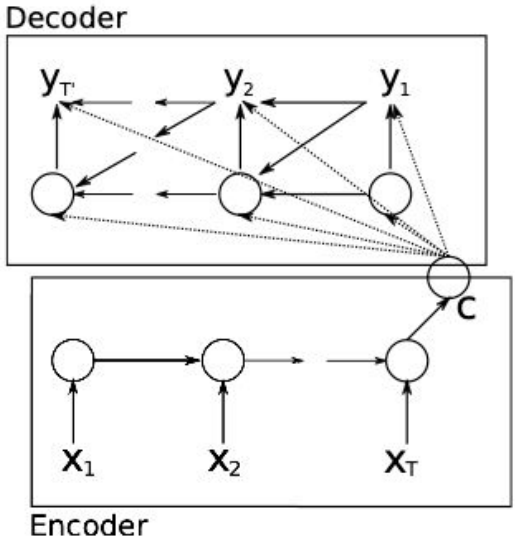


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c}\right)$$

주어진 h<t>를 활용하여, 다음 symbol인 yt 값을 예측하여 전체 output seq. 만듬

계산 시, yt−1 도 함께 사용됨

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g\left(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c}\right).$$

수식을 다음 symbol에 대한 조건부 분포로 나타냄

*f, g는 softmax와 같은 활성화 함수

Model - RNN Encoder Decoder

학습 및 활용

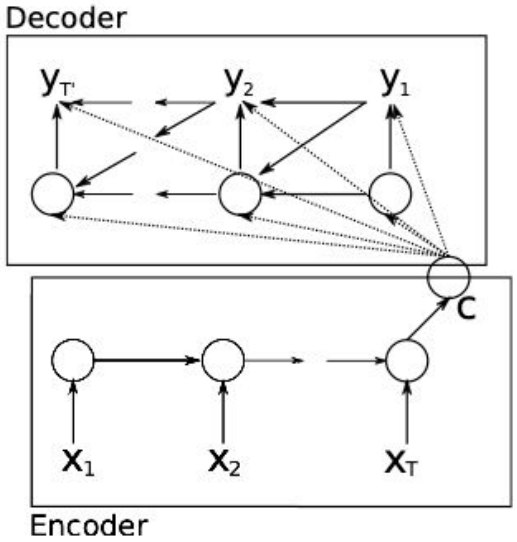


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y_n \mid x_n)$$

조건부 log-likelihood를 최대화 하여 학습

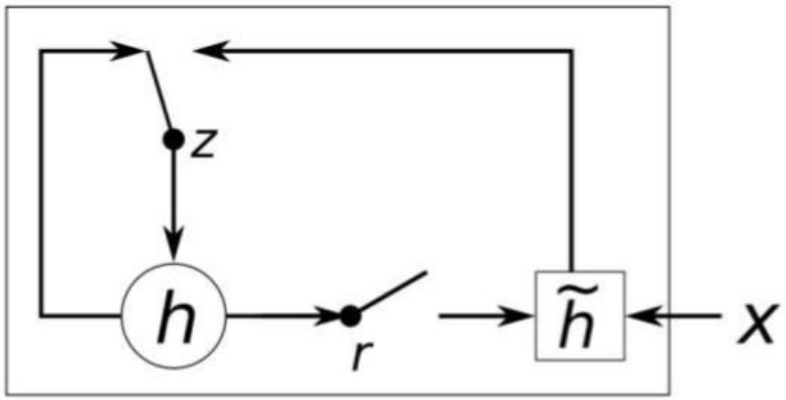
- theta는 모델의 파라메터
- y와 x는 학습 데이터의 입력과 출력 순서쌍

RNN Encoder-Decoder 학습 후 활용

- Input seq.가 주어졌을 때 target sent. 생성
- Pair input이 주어졌을때 주어진 확률을 활용한
  점수 Scoring

Model - Hidden Unit that Adaptively Remembers Forgets

새로운 Hidden Unit (A.k.a: GRU)



Reset Gate $$r_j = \sigma\left(\left[\mathbf{W}_r\mathbf{x}\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}\right]_j\right)$$

이전 Hidden을 제외

Update Gate $$z_j = \sigma\left(\left[\mathbf{W}_z\mathbf{x}\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right)$$

어떤 새로운 hidden state 업데이트 할 것인지

Model - Hidden Unit that Adaptively Remembers Forgets

새로운 Hidden Unit (A.k.a: GRU)

두개의 식으로 Hidden Unit을
계산

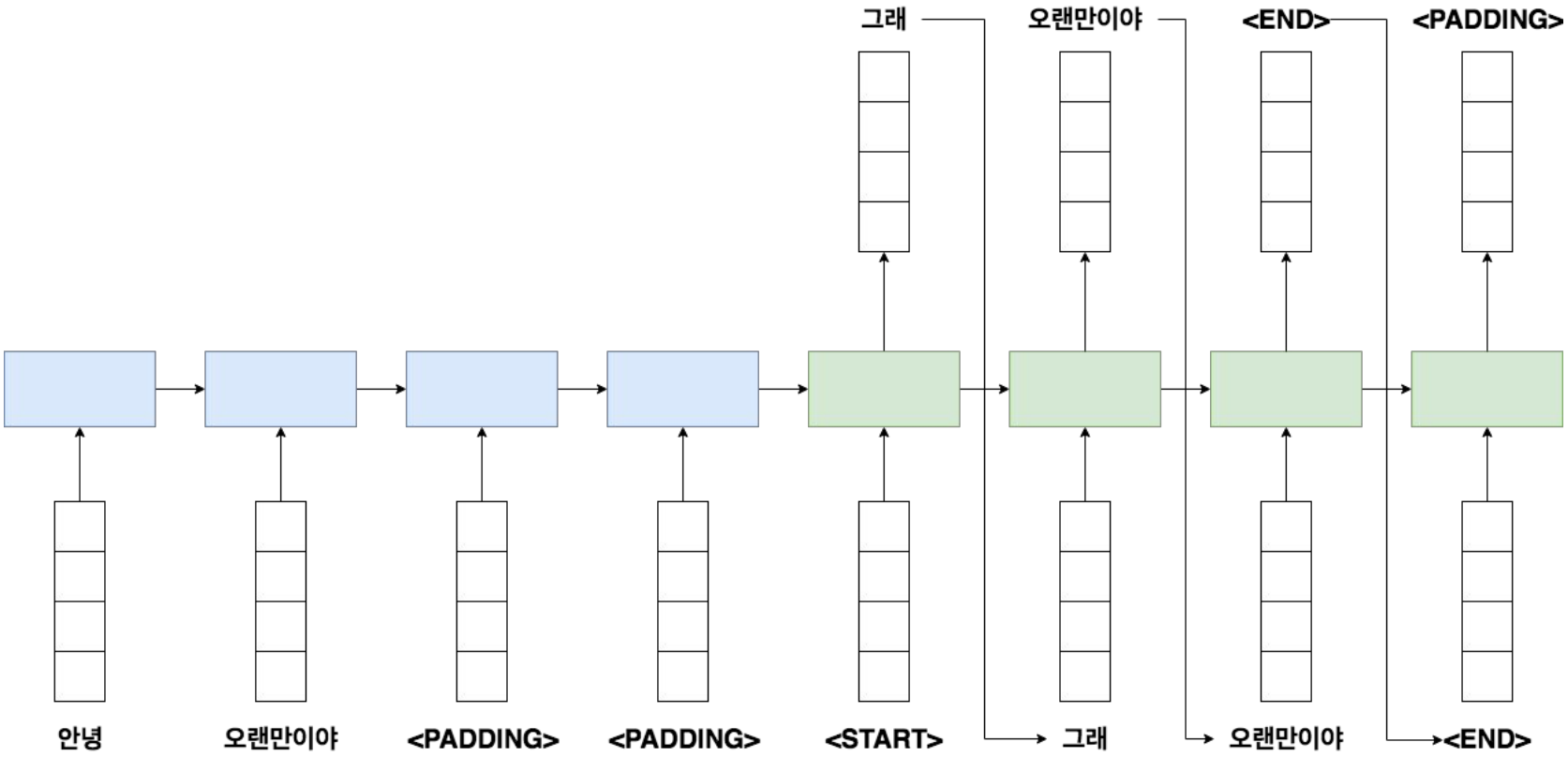
$$h_j^{<t>} = z_j h_j^{<t-1>} + (1 - z_j)\tilde{h}_j^{<t>}$$

$$\tilde{h}_j^{<t>} = \phi\left([\mathbf{W}\mathbf{x}]_j + [\mathbf{U}(\mathbf{r} \odot \mathbf{h}_{<t-1>})]_j\right)$$

각 hidden unit은 개별의 reset과 update garde를 가진다

Model - RNN Encoder Decoder

Example



Model - RNN Encoder Decoder

```
def gru(units):
    if tf.test.is_gpu_available():
        return tf.keras.layers.CuDNNGRU(units,
                                         return_sequences=True,
                                         return_state=True,
                                recurrent_initializer='glorot_uniform')
    else:
        return tf.keras.layers.GRU(units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_activation='sigmoid',
                                   recurrent_initializer='glorot_uniform')
```

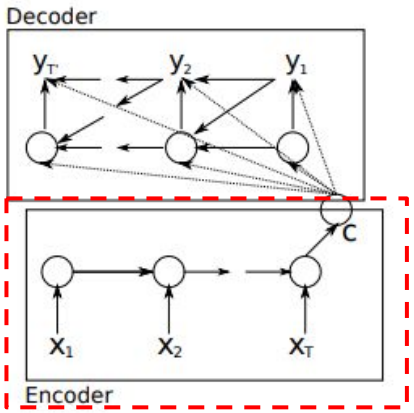
Model - RNN Encoder Decoder

```
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.enc_units)

    def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)

        return output, state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.enc_units))
```



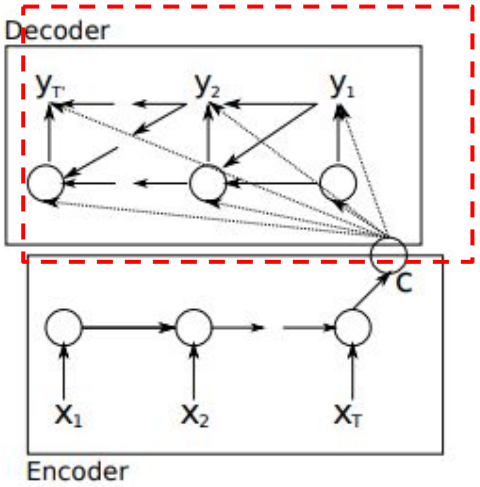
Model - RNN Encoder Decoder

```
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.dec_units)
        self.fc = tf.keras.layers.Dense(vocab_size)

    def call(self, x, hidden, enc_output):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)
        output = tf.reshape(output, (-1, output.shape[2]))
        x = self.fc(output)

        return x, state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.dec_units))
```



Model - RNN Encoder Decoder

```
encoder = Encoder(len(source2idx), embedding_dim, units, batch_size)
decoder = Decoder(len(target2idx), embedding_dim, units, batch_size)

EPOCHS = 100

for epoch in range(EPOCHS):
    hidden = encoder.initialize_hidden_state()
    total_loss = 0
    for i, (s_len, s_input, t_len, t_input, t_output) in enumerate(data):
        loss = 0
        with tf.GradientTape() as tape:
            enc_output, enc_hidden = encoder(s_input, hidden)
            dec_hidden = enc_hidden
            dec_input = tf.expand_dims([target2idx['<bos>']] * batch_size, 1)
            #Teacher Forcing: feeding the target as the next input
            for t in range(1, t_input.shape[1]):
                predictions, dec_hidden = decoder(dec_input, dec_hidden,
enc_output)
                loss += loss_function(t_input[:, t], predictions)
                dec_input = tf.expand_dims(t_input[:, t], 1) #using teacher forcing
```

Model - RNN Encoder Decoder

```
        batch_loss = (loss / int(t_input.shape[1]))
        total_loss += batch_loss
        variables = encoder.variables + decoder.variables
        gradient = tape.gradient(loss, variables)
        optimizer.apply_gradients(zip(gradient, variables))

    if epoch % 10 == 0:
        #save model every 10 epoch
        print('Epoch {} Loss {:.4f} Batch Loss {:.4f}'.format(epoch,
                                            total_loss / n_batch,
                                            batch_loss.numpy()))
        checkpoint.save(file_prefix = checkpoint_prefix)
```

Epoch 0 Loss 0.0307 Batch Loss 0.7687
Epoch 10 Loss 0.0297 Batch Loss 0.7414
Epoch 20 Loss 0.0267 Batch Loss 0.6676
Epoch 30 Loss 0.0237 Batch Loss 0.5925
Epoch 40 Loss 0.0159 Batch Loss 0.3964
Epoch 50 Loss 0.0131 Batch Loss 0.3271
Epoch 60 Loss 0.0100 Batch Loss 0.2498
Epoch 70 Loss 0.0075 Batch Loss 0.1874
Epoch 80 Loss 0.0051 Batch Loss 0.1283
Epoch 90 Loss 0.0031 Batch Loss 0.0763

Model - RNN Encoder Decoder

I feel hungry

W/O Teacher Forcing

	X	Y predict
1	[bos]	a
.		
2	[bos], a	?
.		

Teacher Forcing

1	[bos]	?
.		
2	[bos], I	?
.		
3	[bos], I feel	?
.		

Model - Statistical Machine Translation

주어진 문장 e에 대해 translation func.인 f를 찾는 과정

$$p(\mathbf{f}|\mathbf{e}) \propto p(\mathbf{e}|\mathbf{f})p(\mathbf{f})$$

위의 식을 최대화 하는
과정

Model - Statistical Machine Translation

실제로는 대부분의 SMT 시스템은 아래 식을 모델링

$$\log p\left(f \mid e\right) = \sum_{n=1}^{N} w_n f_n(f, e) + \log Z(e)$$

Z(e): 일반화 상수

n번째 feature와 weights

Weight는 BLEU값을 최대화 하기 위해 학습 됨
RNN Encoder-Decoder를 학습시킨 후 additional feature활용

Experiments

Experiments - Data & Baseline

실험 데이터? English/French '14 Workshop 번역 task

- Europarl(61M words), news commentary (5.5M words), UN(421M words) 90M, 780M의 crawled 한 copora 포함
- French를 위해 712M 단어 학습

Statistical 모델 학습 때는

- 여러 데이터를 concat하는 것이 항상 최적의 성능을 보이지 않고
- 큰 모델일수록 다루기 어려움

Task에 집중하기 위해 Data selection 활용

- Language Model을 위해 418M 단어 추출
- 학습을 위해 348M

RNN Encoder-Decoder에서는 전체의 93% 포함되는 15,000개의 단어로 하고 나머지는 [UNK]로 맵핑함

실험에 사용된 RNN Encoder- Decoder 구조

1. 1000개 hidden unit (GRU)
2. Hidden unit 계산 때는 활성화 함수로 tanh활용
3. 입력 symbol에서 hidden unit으로 갈 때 사용되는 input matrix는 2 lower-rank approx. matrix사용. output matrix도 같은 방식
4. Adadelta / SGD
5. 매 업데이트 마다 64 개 임의 추출된 phase pair 사용
6. 영어 불어에서 가장 많이 활용되는 15,000개 단어 사용

Experiments - Data & Baseline

English/French '14 Workshop

Models	BLEU	
	dev	test
Baseline	30.64	33.30
RNN	31.20	33.87
CSLM + RNN	31.48	34.64
CSLM + RNN + WP	31.50	34.54

Table 1: BLEU scores computed on the development and test sets using different combinations of approaches. WP denotes a *word penalty*, where we penalizes the number of unknown words to neural networks.

RNN비교를 위해서

Target lang.을 학습하기

위한 방법 전통 CSLM 추가

+

Word Penalty

BLEU (Bilingual Evaluation Understudy)

BLEU (Bilingual Evaluation Understudy)

기계 번역의 품질을 측정하는데 사용하는 지표로써,
실제 사람이 한 번역과 기계 번역의 유사성을 계산하는 방식

$$BLEU = min(1, \frac{output\ length(\text{예측 문장})}{reference\ length(\text{실제 문장})})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

n-gram을 통한 순서쌍들이 얼마나 겹치는지 측정(precision)

문장길이에 대한 과적합 보정 (Brevity Penalty)

같은 단어가 연속적으로 나올때 과적합 되는 것을 보정(Clipping)

https://donghwa-kim.github.io/BLEU.html

BLEU (Bilingual Evaluation Understudy)

1. n-gram(1~4)을 통한 순서쌍들이 얼마나 겹치는지 측정(precision)

- `예측된 sentence` : 빛이 쐬는 노인은 완벽한 어두운곳에서 잠든 사람과 비교할 때 강박증이 심해질 기회가 훨씬 높았다
- `true sentence` : 빛이 쐬는 사람은 완벽한 어둠에서 잠든 사람과 비교할 때 우울증이 심해질 가능성이 훨씬 높았다

- 1-gram precision: $\frac{\text{일치하는}1-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}1-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{10}{14}$
- 2-gram precision: $\frac{\text{일치하는}2-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}2-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{5}{13}$
- 3-gram precision: $\frac{\text{일치하는}3-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}3-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{2}{12}$
- 4-gram precision: $\frac{\text{일치하는}4-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}4-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{1}{11}$

$$(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}} = (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

https://donghwa-kim.github.io/BLEU.html

BLEU (Bilingual Evaluation Understudy)

2. 같은 단어가 연속적으로 나올때 과적합 되는 것을 보정(Clipping)

> 위 예제에서 단어 단위로 n-gram을 할 경우 보정할 것이 없지만, 영어의 한 예제에서 1-gram precision를 구하면, 예측된 문장에 중복된 단어들(`the`:3, `more`:2)이 있다. 이를 보정하기 위해 `true sentence`에 있는 중복되는 단어의 max count(`the`:2, `more`:1)를 고려하게 된다. (Clipping). 다른 n-gram도 같은 방식으로 처리하면 된다.

- `예측된 sentence` : The more decomposition the more flavor the food has
- `true sentence` : The more the merrier I always say

- 1-gram precision: $\frac{\text{일치하는} 1-gram\text{의 수(예측된 } sentence\text{중에서)}}{\text{모든} 1-gram\text{쌍 (예측된 } sentence\text{중에서)}} = \frac{5}{9}$
- (보정 후) 1-gram precision: $\frac{\text{일치하는} 1-gram\text{의 수(예측된 } sentence\text{중에서)}}{\text{모든} 1-gram\text{쌍 (예측된 } sentence\text{중에서)}} = \frac{3}{9}$

https://donghwa-kim.github.io/BLEU.html

BLEU (Bilingual Evaluation Understudy)

3. 문장길이에 대한 과적합 보정 (Brevity Penalty)

> 같은 예제에 문장길이에 대한 보정계수를 구하면 다음과 같다.

- `예측된 sentence` : 빛이 쐬는 노인은 완벽한 어두운곳에서 잠듬
- `true sentence` : 빛이 쐬는 사람은 완벽한 어둠에서 잠든 사람과 비교할 때 우울증이 심해질 가능성이 훨씬 높았다

$$min(1, \frac{\text{예측된 } sentence\text{의 길이(단어의 갯수)}}{true\ sentence\text{의 길이(단어의 갯수)}}) = min(1, \frac{6}{14}) = \frac{3}{7}$$

https://donghwa-kim.github.io/BLEU.html

BLEU (Bilingual Evaluation Understudy)

BLEU score

- 위에서 구한 값들을 종합해서 아래 결과의 BLEU score를 계산하면 다음과 같다.
- `예측된 sentence` : 빛이 쐬는 노인은 완벽한 어두운곳에서 잠든 사람과 비교할 때 강박증이 심해질 기회가 훨씬 높았다
- `true sentence` : 빛이 쐬는 사람은 완벽한 어둠에서 잠든 사람과 비교할 때 우울증이 심해질 가능성이 훨씬 높았다

$$BLEU = min(1, \frac{output\ length(\text{예측 문장})}{reference\ length(\text{실제 문장})})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

$$= min(1, \frac{6}{14}) \times (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

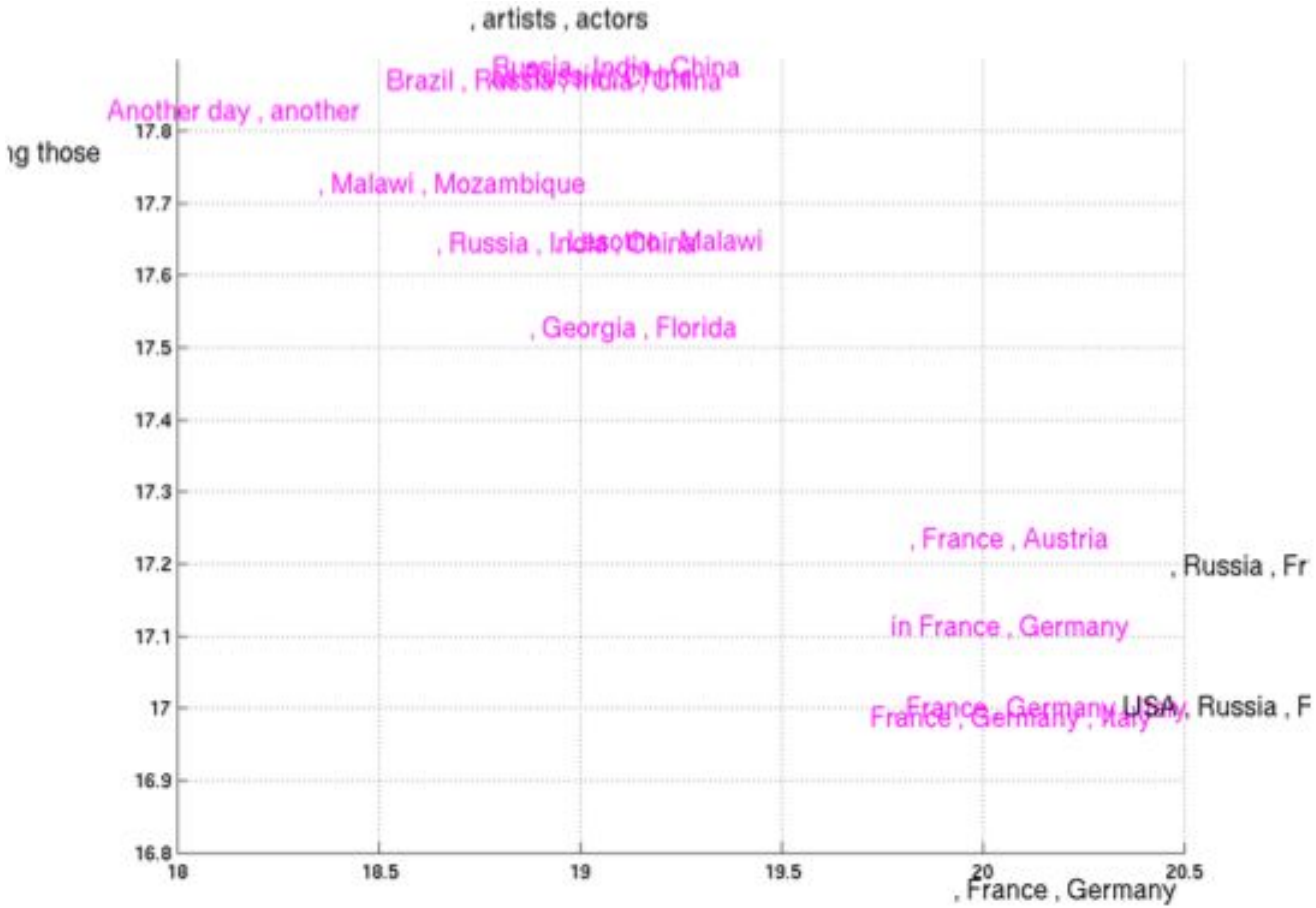
https://donghwa-kim.github.io/BLEU.html

Model - Qualitative Analysis

Semantic & Syntatic 구조 다 잘보더라

Model - Qualitative Analysis

Semantic & Syntatic 구조 다 잘보더라



성능이 잘 나오는 이유?

- **Frequency 정보없이 훈련되어, 통계적으로 도출된 Score가 아님 (언어적 규칙)**
- **출현 빈도가 많지 않은 표현들에 대해서 보완이 가능함**
- **RNN Encoder-Decoder가 짧은 표현을 선호함**
  - **일반적으로 BLEU 값은 짧은 표현에서 높게 나옴**

Conclusion

Conclusion

- **RNN Encoder-Decoder 모델 제시**
  - 임의의 길이 sequence에도 잘 working하더라
- **Purpose of RNN Encoder-Decoder**
  - Score a pair of sequences (Conditional Prob.)
    -> 언어학적 규칙성
  - Generate a target sequence a given source sequence
    -> Well-Formed Target Phases
- **새로운 Hidden units with a reset gate + update gate**
- **BLEU Score를 기반으로 한 번역 성능의 향상**
- **범용적인 Phase Representation**
  - 의미와 문법 모두를 담아냄

Reference

https://www.youtube.com/watch?v=_Dp8u97_rQ0&t=747s
https://reniew.github.io/31/