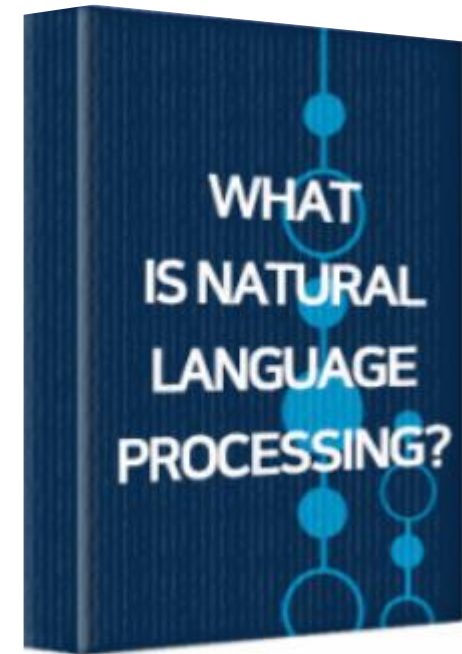


Tensorflow를 활용한 딥러닝 자연어 처리 입문. 6강

앞으로 배우게 될 내용

- Text preprocessing for NLP & Language Model
- Basic Tensorflow & Vectorization
- Word Embedding (Word2Vec, FastText, GloVe)
- Text Classification (using RNN & CNN)
- Chatbot with Deep Learning
- **Sequence to Sequence**
- Attention Mechanism
- Transformer & BERT



참고 자료 : <https://wikidocs.net/book/2155>

Language Generation

Natural Language Generation

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Generation

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Generation이 필요한 분야?

Natural Language Generation

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Generation이 필요한 분야?
Image Captioning

Natural Language Generation

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Generation이 필요한 분야?

Image Captioning

Chatbot

Natural Language Generation

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Generation이 필요한 분야?

Image Captioning

Chatbot

AI writer(GPT-2)

가짜뉴스, 수필까지 척척...사람처럼 글쓰는 AI 등장

[JTBC] 입력 2019-02-19 21:44 | 수정 2019-02-20 17:40

안내 ▶ JTBC 뉴스는 여러분의 생생한 제보를 기다리고 있습니다.

[앵커]

인공지능이 사람처럼 글을 스스로 써내는 기술이 나왔습니다. 단어를 제시하면 수려한 문장을 만들어 냅니다. 미국의 한 연구소가 개발한 것인데, 가짜뉴스를 실감나게 만들어내는 사례도 소개했습니다. 이 연구소는 인공지능이 악용될 수 있다면서 관련기술을 공개하지 않기로 했습니다.

Natural Language Generation

- 자연어 처리는 크게 자연어 이해(NLU)와 자연어 생성(NLG)의 영역이 있다.
- 자연어 생성은 기계가 텍스트를 스스로 생성하는 영역을 말한다.

$$\text{NLP} = \text{NLU} + \text{NLG}$$

Natural Language Generation이 필요한 분야?
Image Captioning

Chatbot

Neural Machine Translation

AI writer(GPT-2)

가짜뉴스, 수필까지 척척...사람처럼 글쓰는 AI 등장

[JTBC] 입력 2019-02-19 21:44 | 수정 2019-02-20 17:40

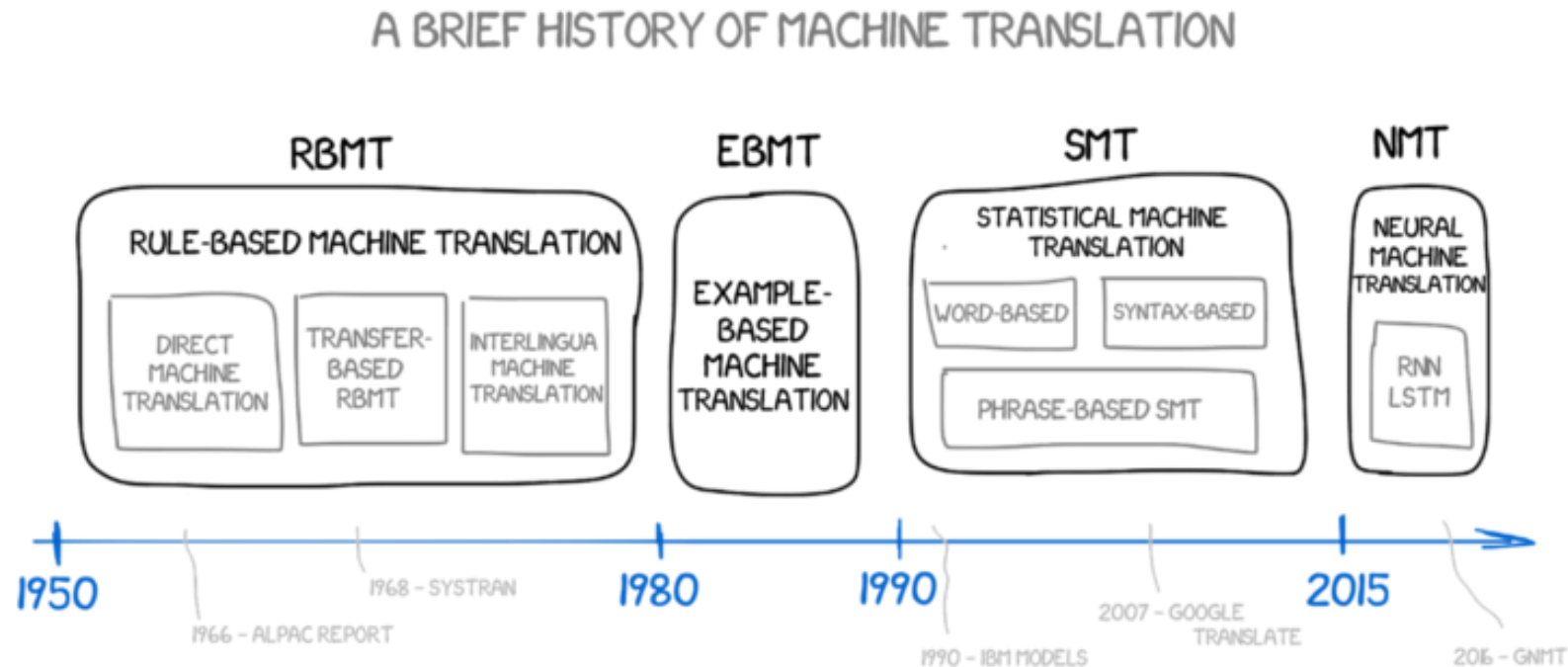
안녕하세요. JTBC 뉴스는 여러분의 생생한 제보를 기다리고 있습니다.

[앵커]

인공지능이 사람처럼 글을 스스로 써내는 기술이 나왔습니다. 단어를 제시하면 수려한 문장을 만들어 냅니다. 미국의 한 연구소가 개발한 것인데, 가짜뉴스를 실감나게 만들어내는 사례도 소개했습니다. 이 연구소는 인공지능이 악용될 수 있다면서 관련기술을 공개하지 않기로 했습니다.

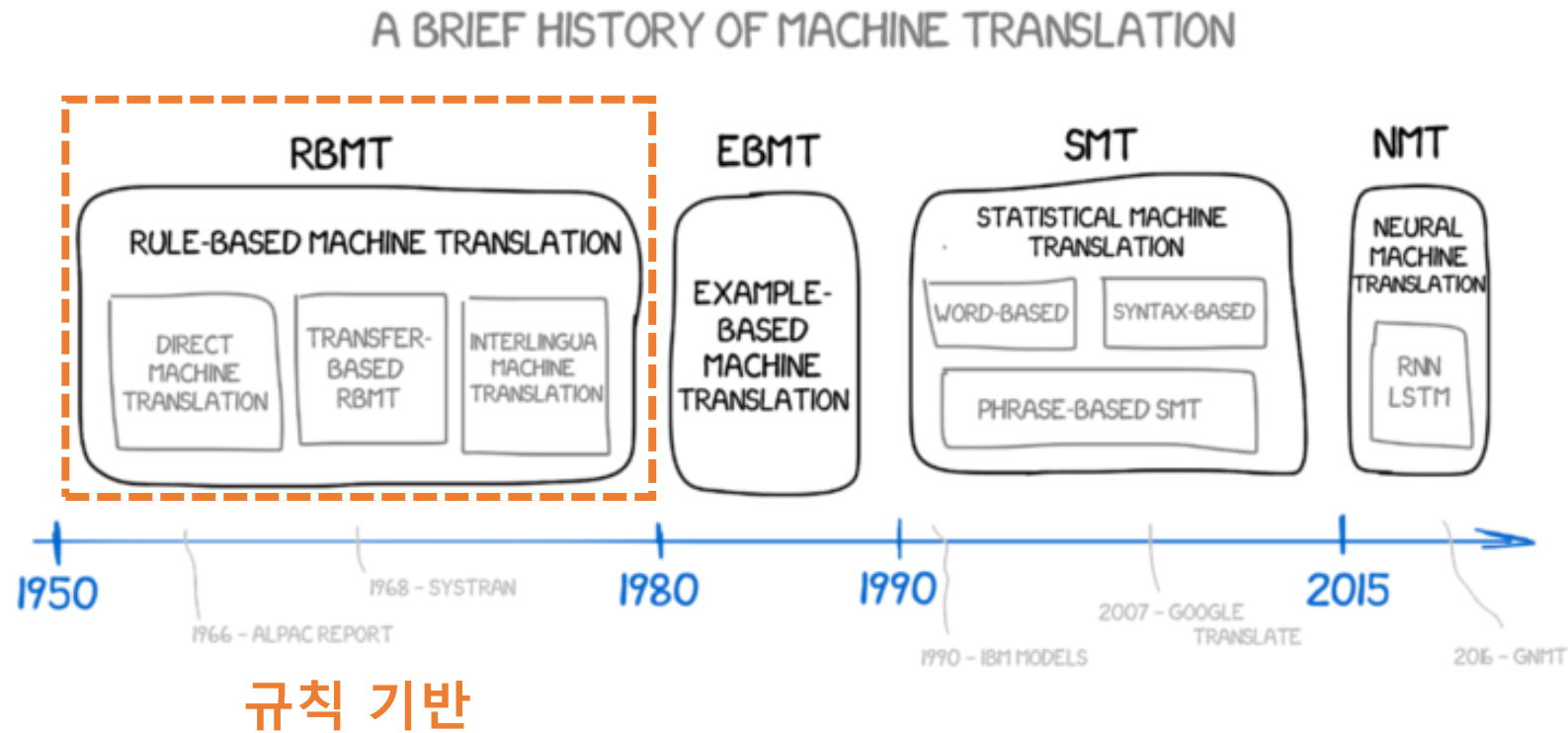
Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.



Machine Translation

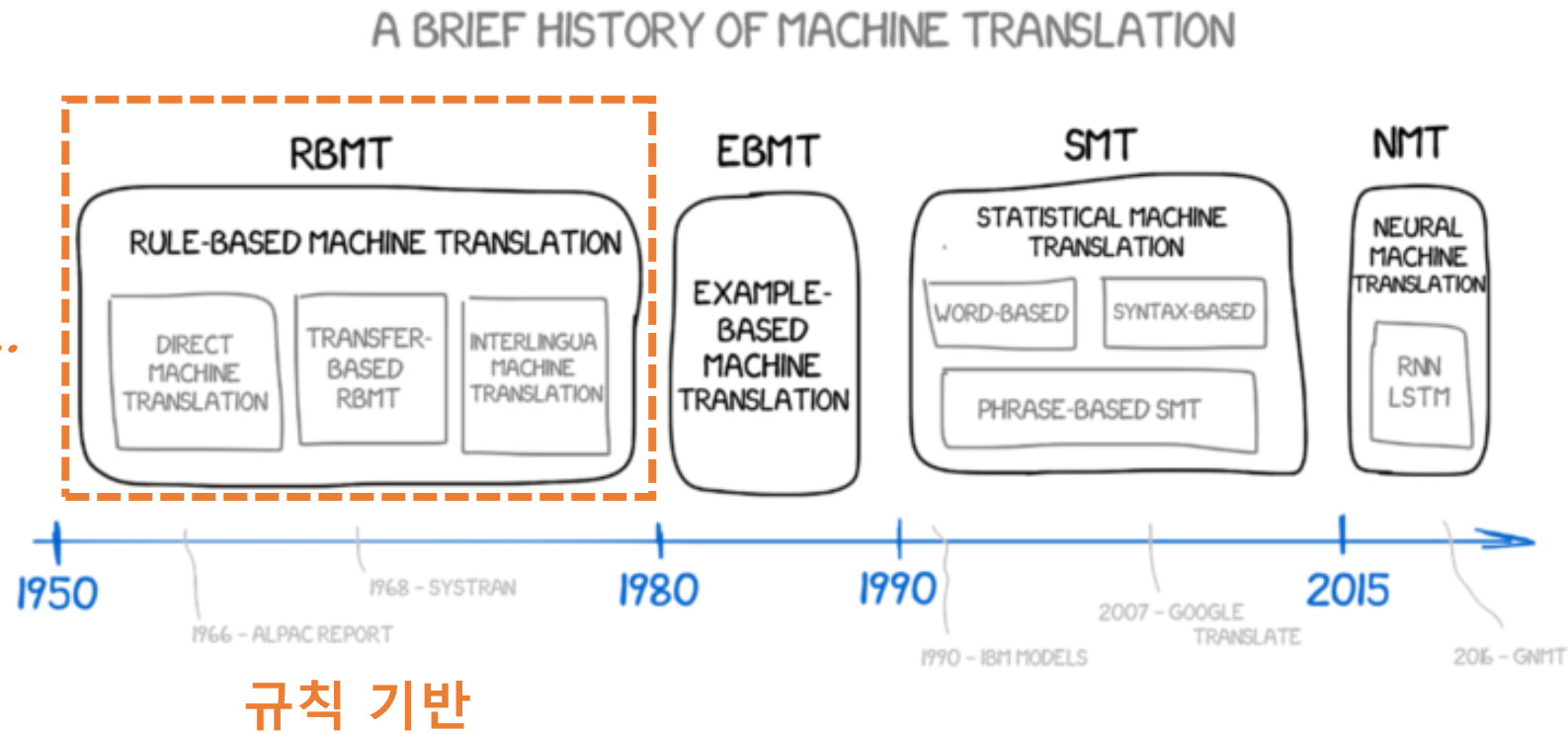
- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.



Machine Translation

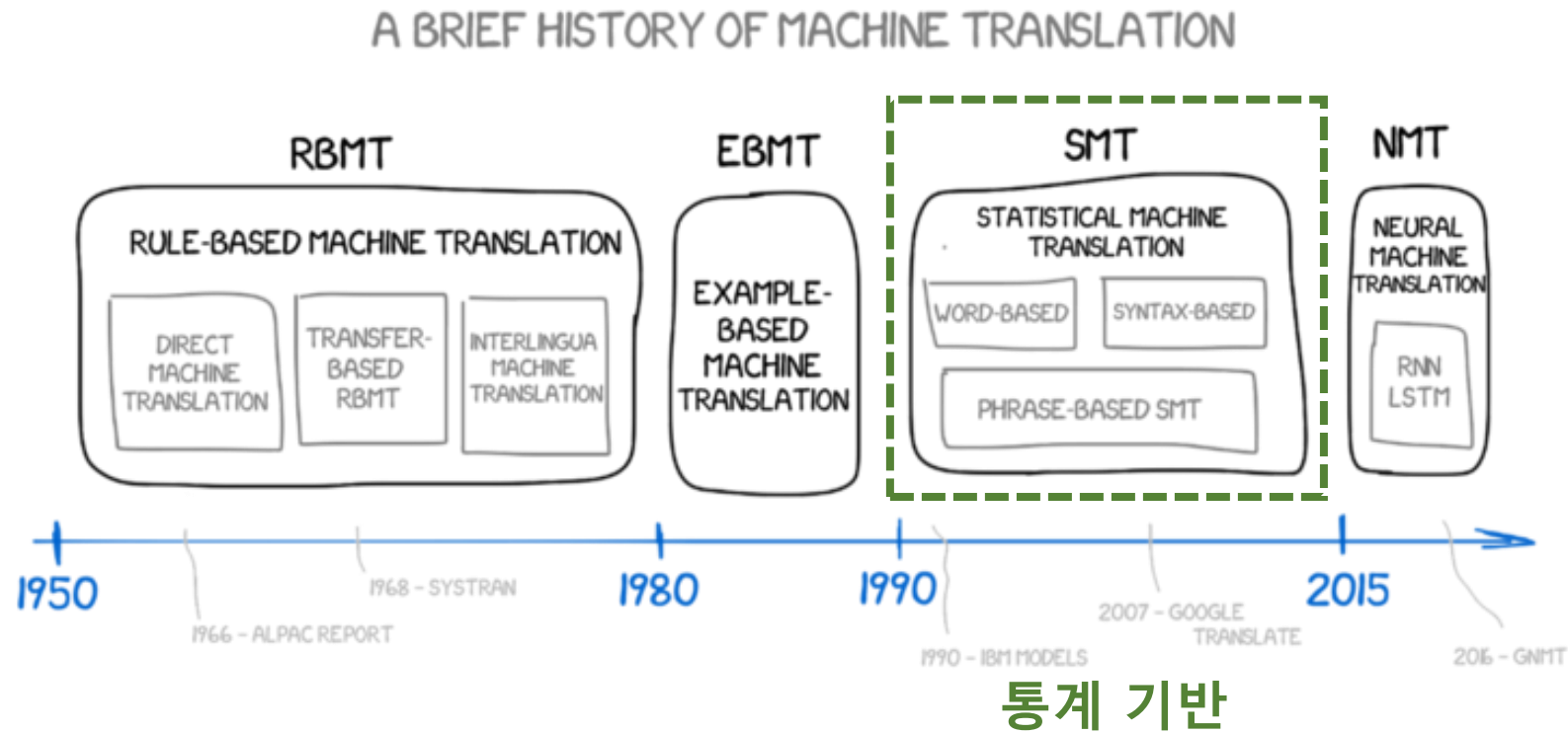
- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

*If... else...
제대로 될리가...*



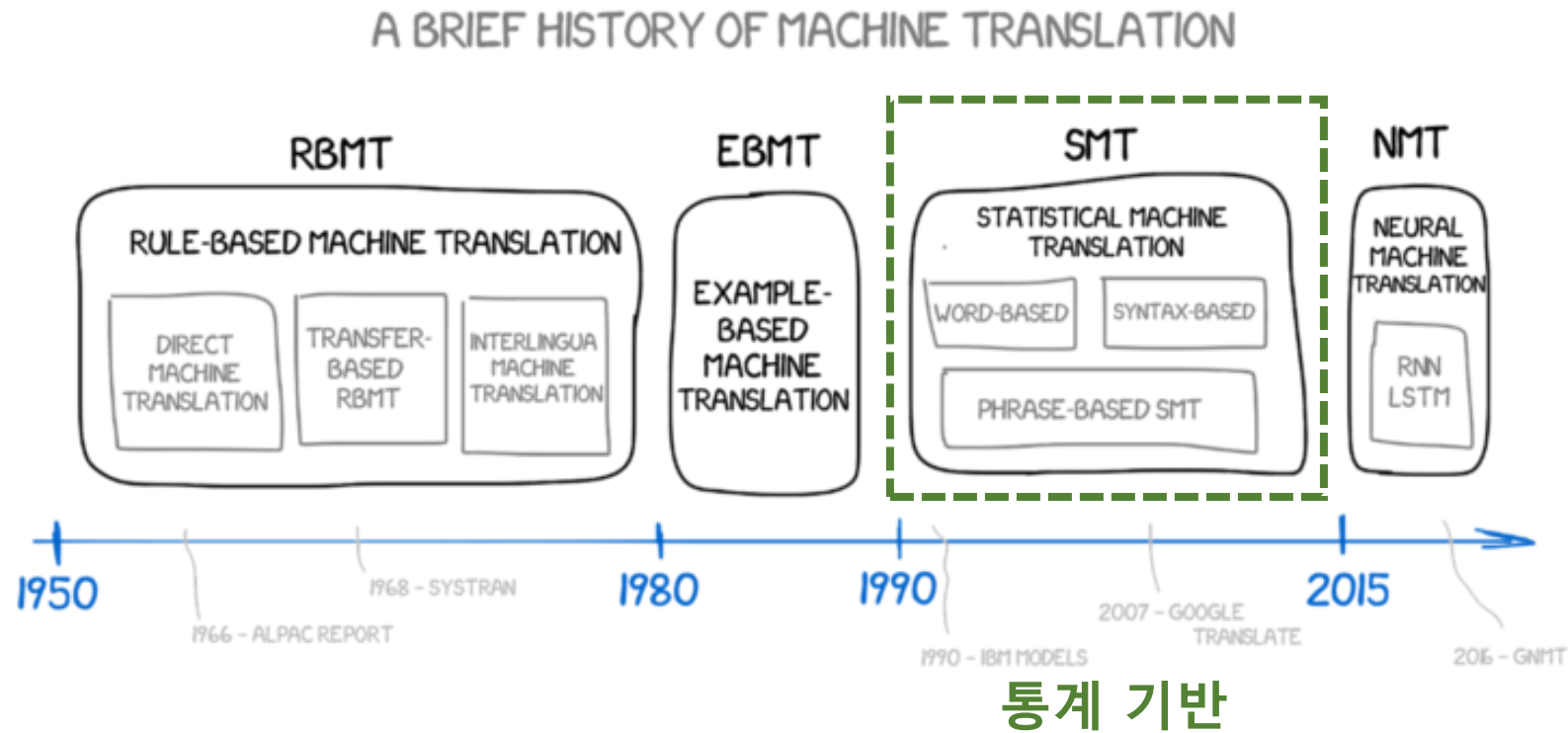
Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.



Machine Translation

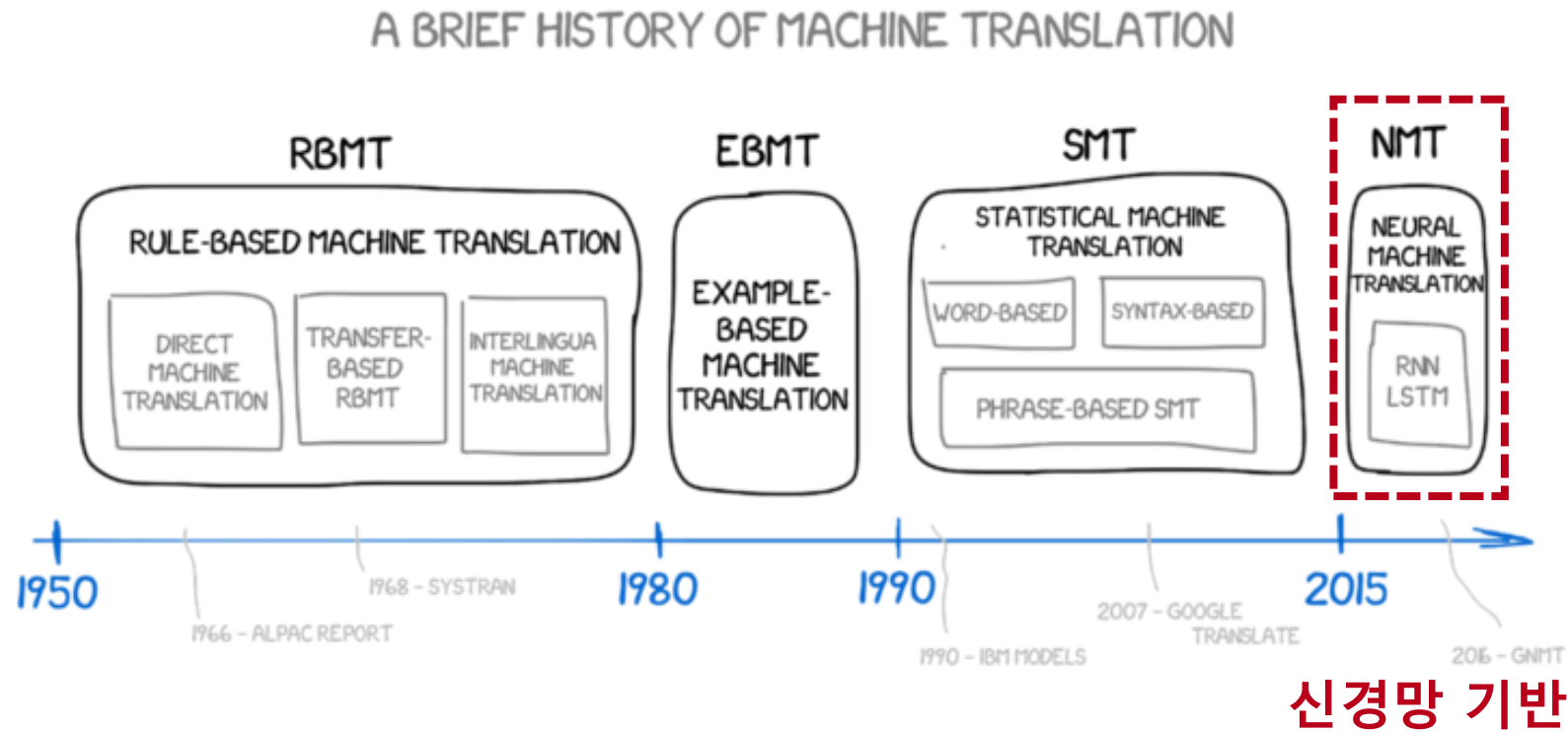
- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.



그럭저럭
여전히 아쉽...

Machine Translation

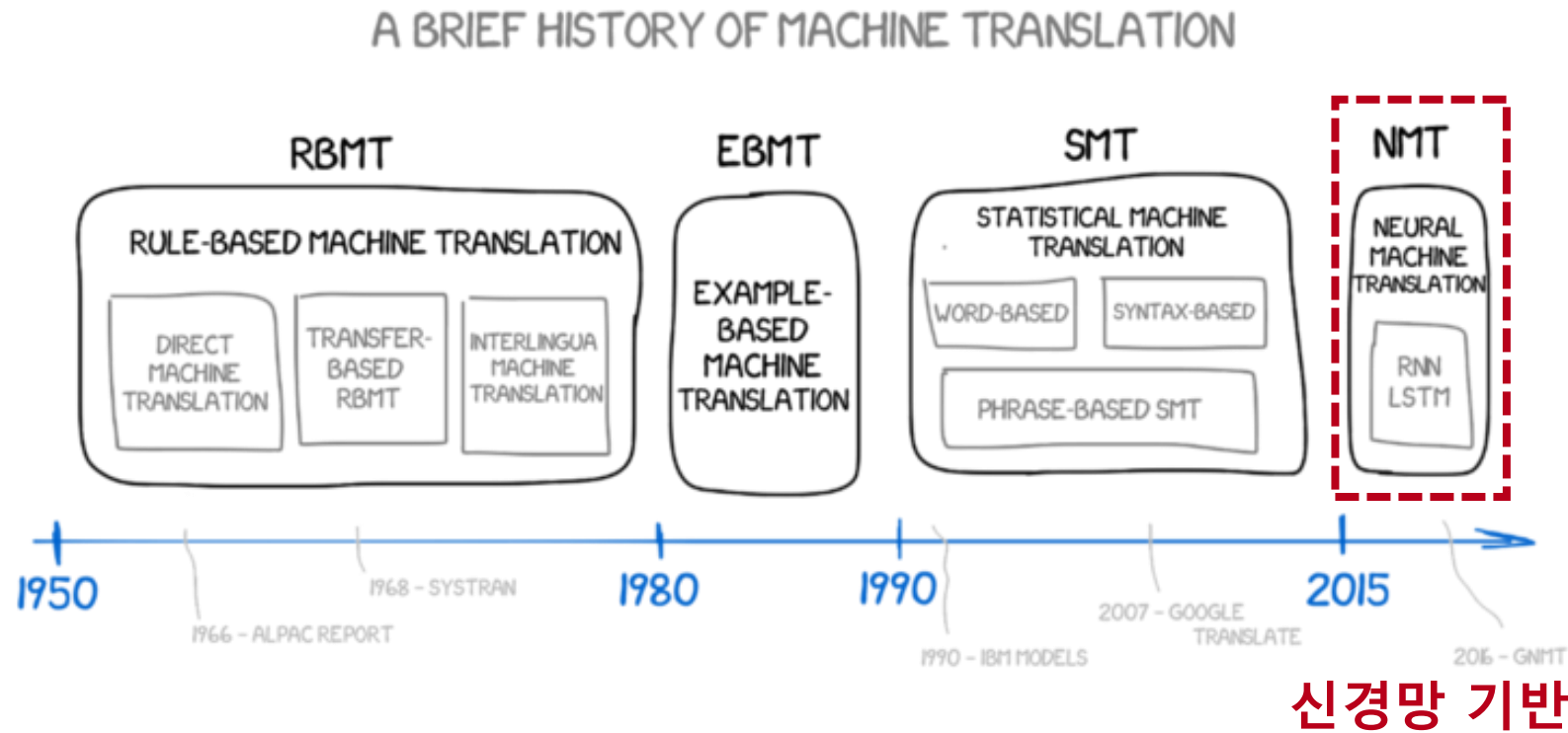
- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.



Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

Neural Machine Translation 시대의 시작.



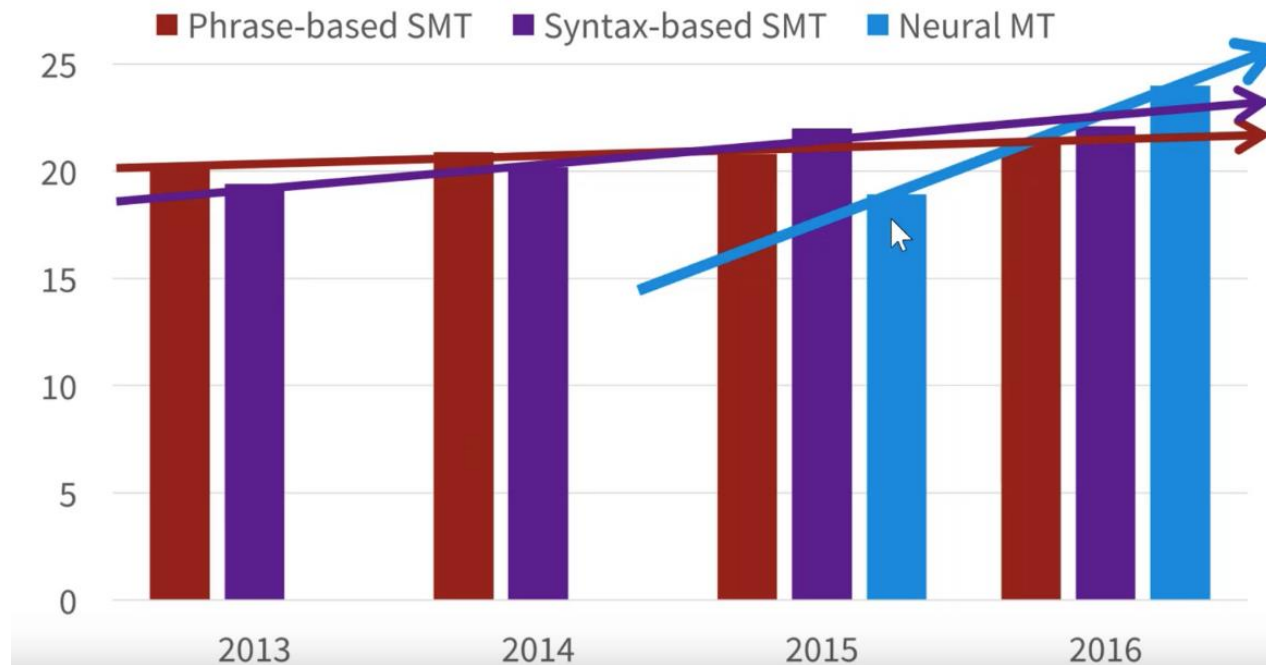
Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

Neural Machine Translation 시대의 시작.

Progress in Machine Translation

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



Neural Machine Translation

- 입력 문장(source text)을 번역한 출력 문장(target text)을 생성해내는 Task.

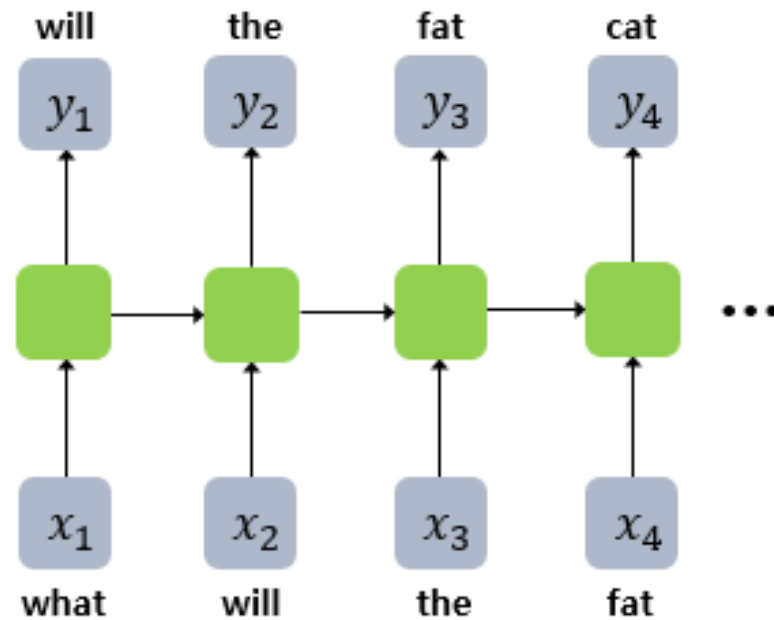
Neural Machine Translation 시대의 시작.

HOW?

1. Word Embedding으로 인한 Continuous Representation의 힘.
2. 기존 SMT가 여러 모듈이 결합된 결과였다면 이제는 End-to-End 모델의 시대.
3. Attention으로 인해 길이가 긴 문장 또한 좋은 성능을 보이기 시작!

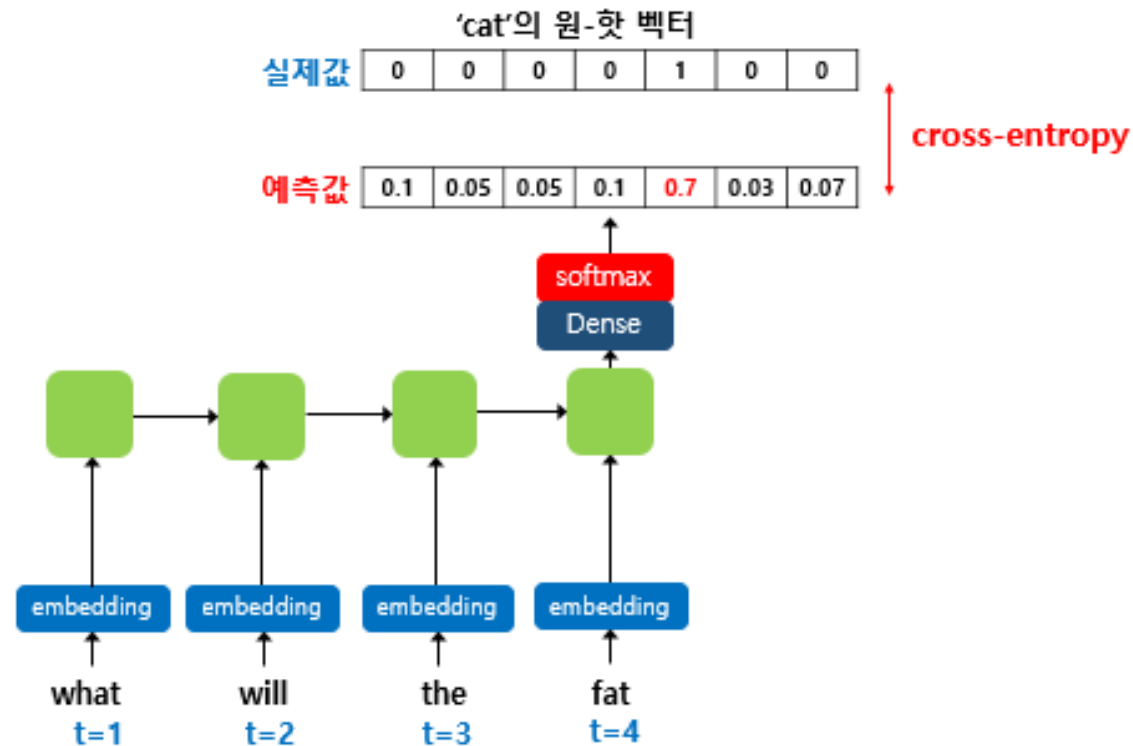
RNN Language Model

- RNN을 사용하여 언어 모델을 구현
- **입력의 길이를 자유롭게 할 수 있으면서** 임베딩층을 사용하여 워드 임베딩의 이점을 가진다.
- 현재 시점의 출력이 다음 시점의 입력으로 사용되며, 이를 마지막 시점까지 반복한다.



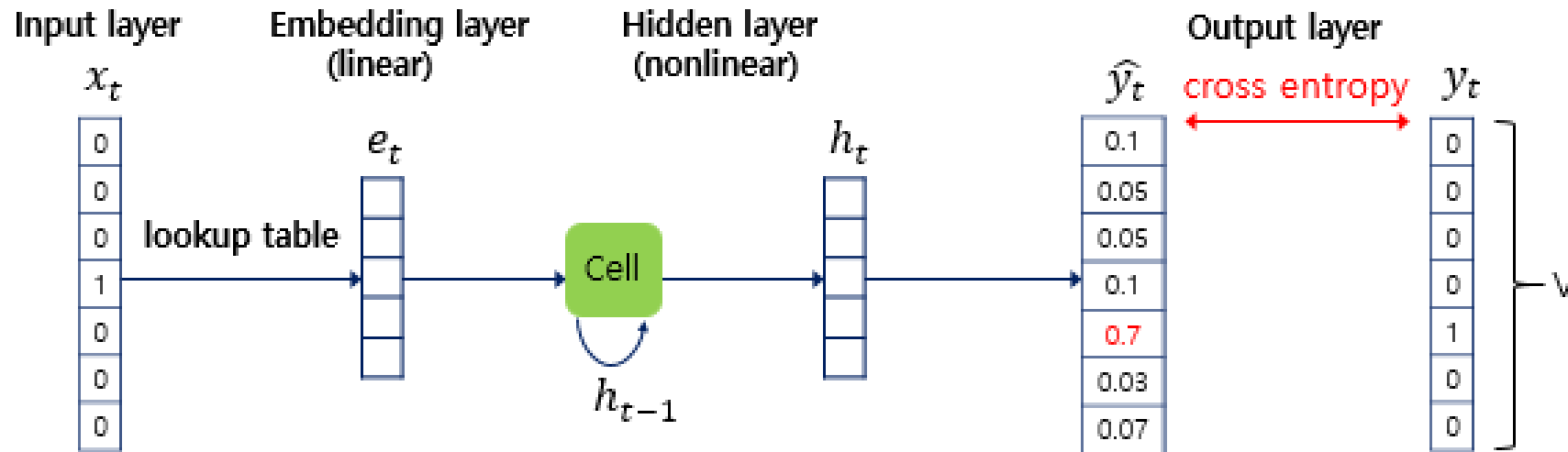
RNN Language Model (Output Layer)

- RNN을 사용하여 언어 모델을 구현
- **입력의 길이를 자유롭게 할 수 있으면서** 임베딩층을 사용하여 워드 임베딩의 이점을 가진다.
- 현재 시점의 출력이 다음 시점의 입력으로 사용되며, 이를 마지막 시점까지 반복한다.



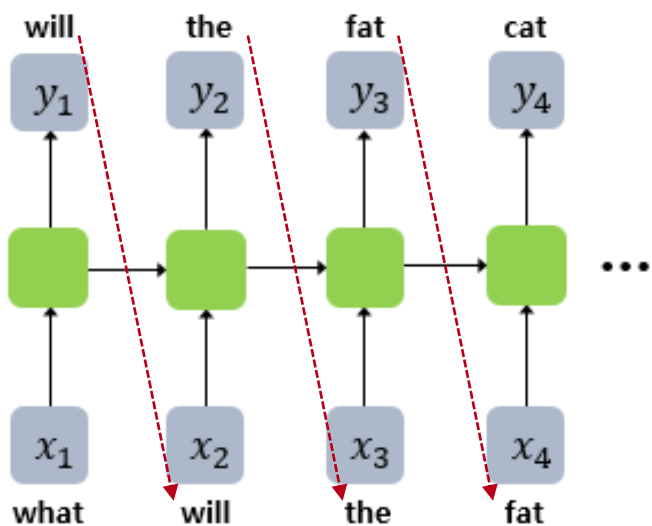
RNN Language Model

- RNN을 사용하여 언어 모델을 구현
- **입력의 길이를 자유롭게 할 수 있으면서** 임베딩층을 사용하여 워드 임베딩의 이점을 가진다.
- 현재 시점의 출력이 다음 시점의 입력으로 사용되며, 이를 마지막 시점까지 반복한다.

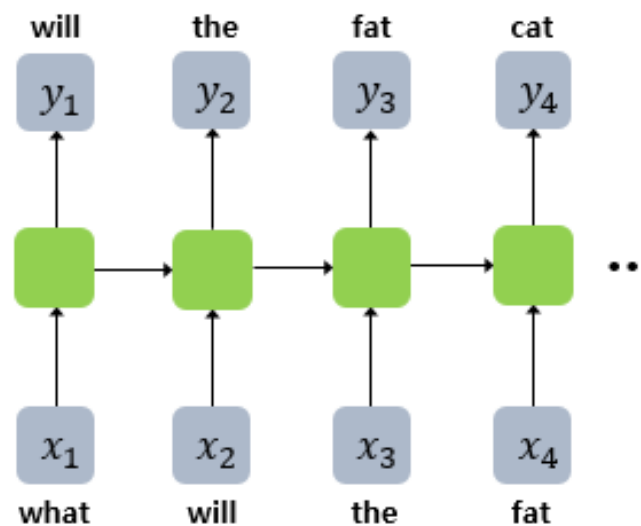


Teacher forcing

- RNN 언어 모델의 훈련 기법.
- 테스트 단계에서는 현재 시점의 예측값이 다음 시점의 입력으로 사용된다.
- 하지만 이를 훈련 단계에서 사용하면 훈련 속도가 매우 느려질 수 있다.
- 훈련 단계에서는 실제값으로 훈련하여 빠르게 훈련한다.



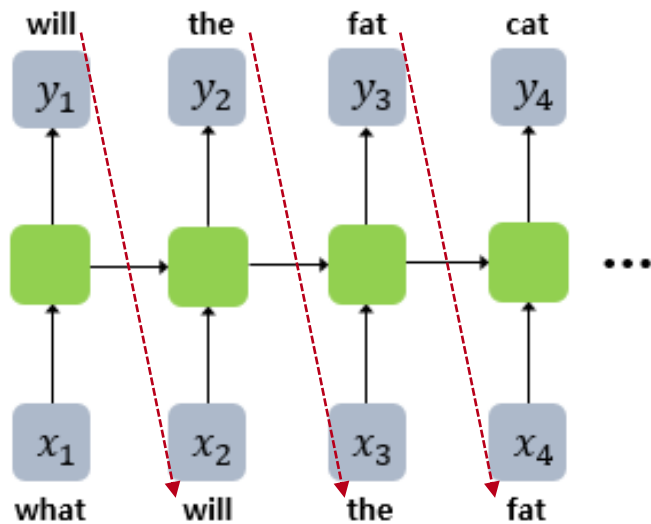
Teacher Forcing 미사용



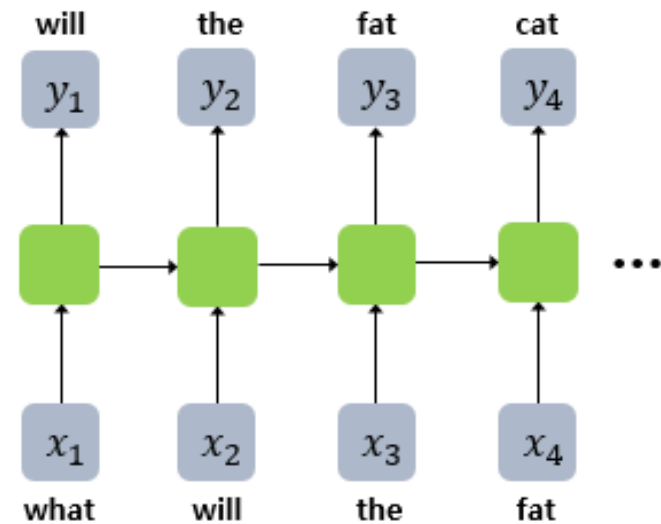
Teacher Forcing 사용

RNN Language Model

- Quiz : RNN LM을 Bidirectional RNN으로 구현이 가능할까요?



Teacher Forcing 미사용

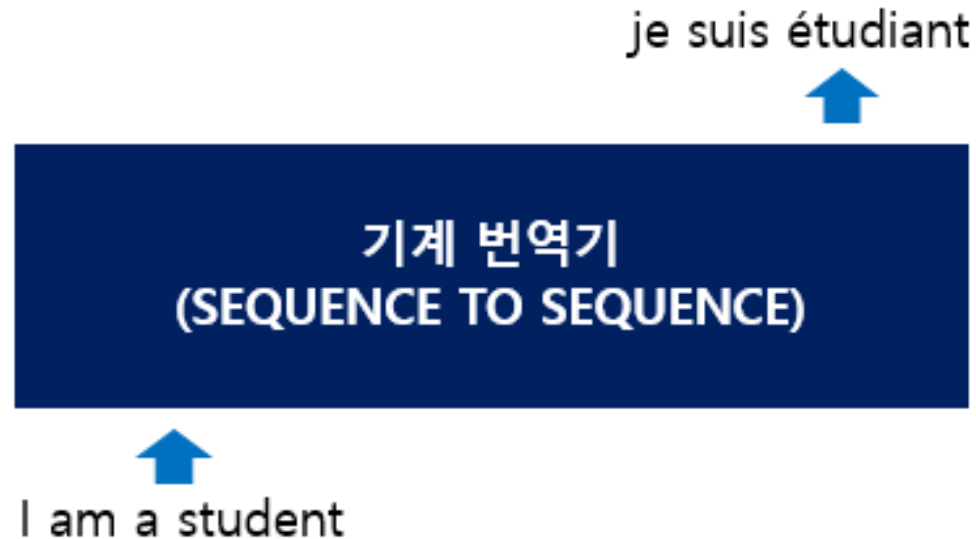


Teacher Forcing 사용

Sequence-to-Sequence

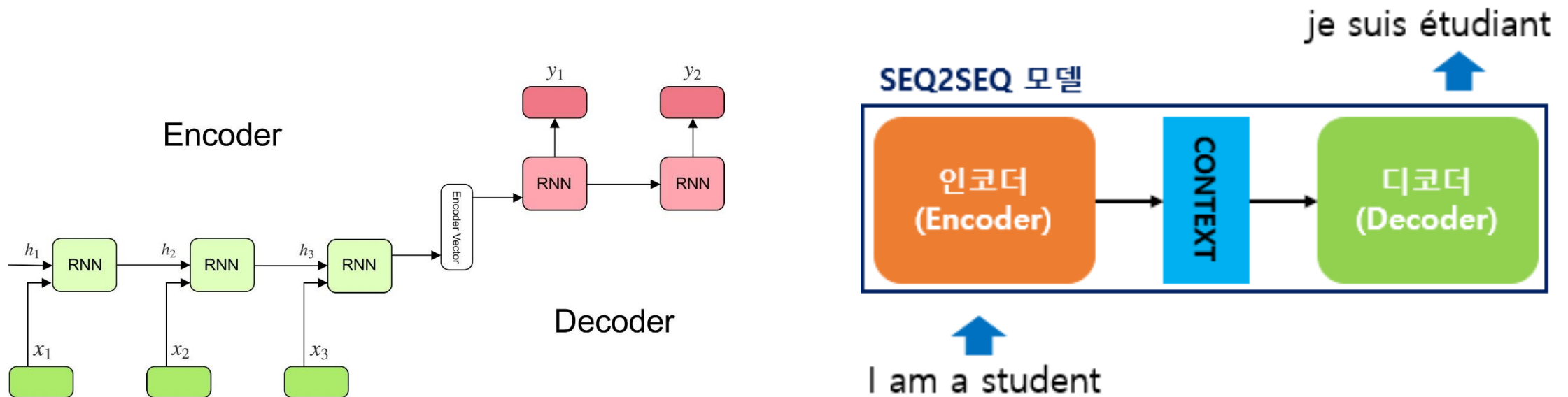
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력한다.
- Ex) 챗봇(Chatbot), 기계 번역(Machine Translation), 텍스트 요약(Text Summarization) 등..



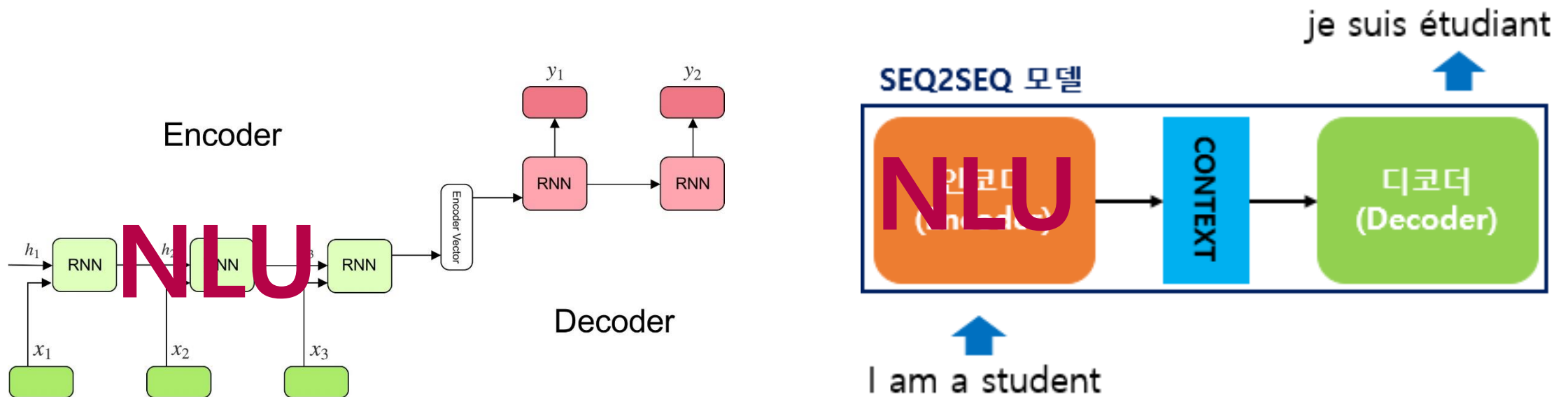
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력한다.
- Ex) 챗봇(Chatbot), 기계 번역(Machine Translation), 텍스트 요약(Text Summarization) 등..
- seq2seq는 내부적으로 인코더와 디코더 구조를 가지고 있다.



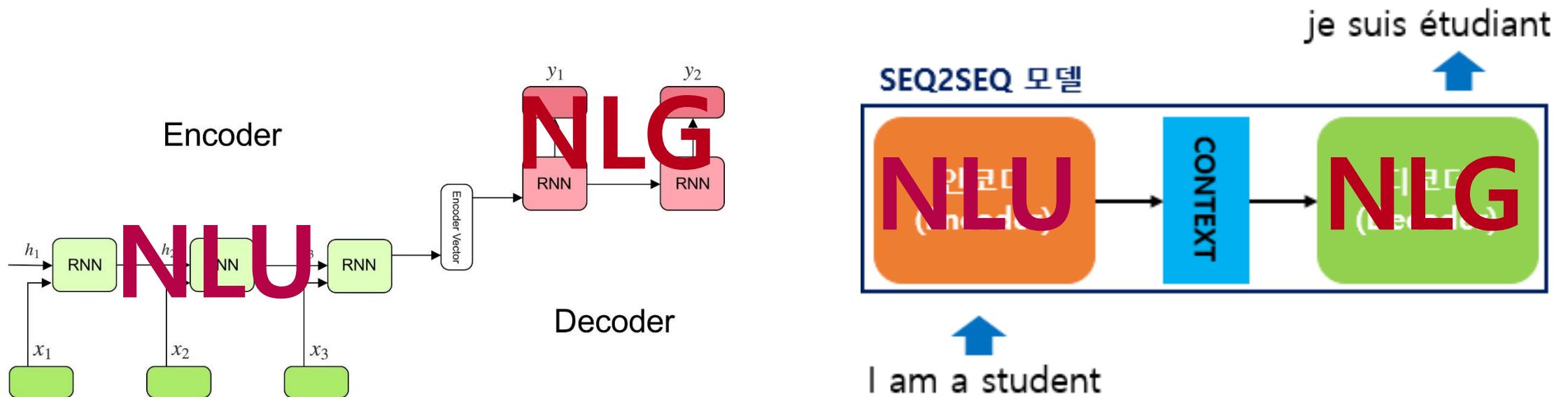
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력한다.
- Ex) 챗봇(Chatbot), 기계 번역(Machine Translation), 텍스트 요약(Text Summarization) 등..
- seq2seq는 내부적으로 인코더와 디코더 구조를 가지고 있다.



Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력한다.
- Ex) 챗봇(Chatbot), 기계 번역(Machine Translation), 텍스트 요약(Text Summarization) 등..
- seq2seq는 내부적으로 인코더와 디코더 구조를 가지고 있다.



Sequence-To-Sequence, Seq2Seq

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

Kyunghyun Cho

Bart van Merriënboer Caglar Gulcehre
Université de Montréal

firstname.lastname@umontreal.ca

Dzmitry Bahdanau

Jacobs University, Germany

d.bahdanau@jacobs-university.de

Fethi Bougares Holger Schwenk

Université du Maine, France Université de Montréal, CIFAR Senior Fellow

firstname.lastname@lium.univ-lemans.fr

Yoshua Bengio

find.me@on.the.web

Abstract

In this paper, we propose a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the RNN Encoder–Decoder as an additional feature in the existing log-linear model. Qualitatively, we show that the proposed model learns a semantically and syntactically meaningful representation of linguistic phrases.

Along this line of research on using neural networks for SMT, this paper focuses on a novel neural network architecture that can be used as a part of the conventional phrase-based SMT system. The proposed neural network architecture, which we will refer to as an *RNN Encoder–Decoder*, consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence. The two networks are trained jointly to maximize the conditional probability of the target sequence given a source sequence. Additionally, we propose to use a rather sophisticated hidden unit in order to improve both the memory capacity and the ease of training.

The proposed RNN Encoder–Decoder with a novel hidden unit is empirically evaluated on the task of translating from English to French. We train the model to learn the translation probability of an English phrase to a corresponding French phrase. The model is then used as a part of a stan-

- 인코더와 디코더 구조를 제시한 논문
- 통계 번역기에 이를 적용함.

Sequence-To-Sequence, Seq2Seq

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT-14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous state of the art. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

1 Introduction

Deep Neural Networks (DNNs) are extremely powerful machine learning models that achieve excellent performance on difficult problems such as speech recognition [13, 7] and visual object recognition [19, 6, 21, 20]. DNNs are powerful because they can perform arbitrary parallel computation for a modest number of steps. A surprising example of the power of DNNs is their ability to sort N N -bit numbers using only 2 hidden layers of quadratic size [27]. So, while neural networks are related to conventional statistical models, they learn an intricate computation. Furthermore, large DNNs can be trained with supervised backpropagation whenever the labeled training set has enough information to specify the network's parameters. Thus, if there exists a parameter setting of a large DNN that achieves good results (for example, because humans can solve the task very rapidly), supervised backpropagation will find these parameters and solve the problem.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori. For example, speech recognition and machine translation are sequential problems. Likewise, question answering can also be seen as mapping a sequence of words representing the question to a

- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 RNN 아키텍처를 인코더-디코더로 사용

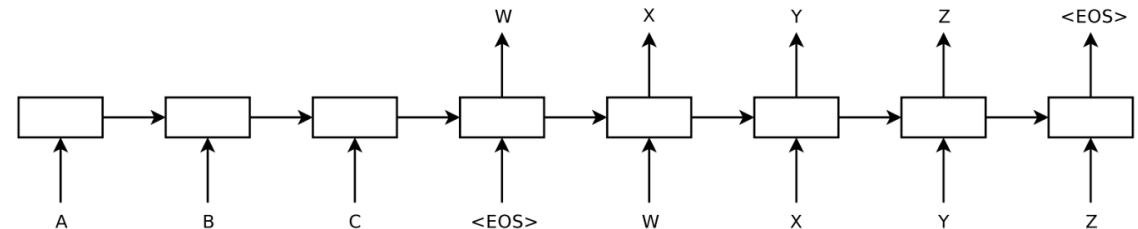


Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Sequence-To-Sequence, Seq2Seq

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT-14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous state of the art. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

1 Introduction

Deep Neural Networks (DNNs) are extremely powerful machine learning models that achieve excellent performance on difficult problems such as speech recognition [13, 7] and visual object recognition [19, 6, 21, 20]. DNNs are powerful because they can perform arbitrary parallel computation for a modest number of steps. A surprising example of the power of DNNs is their ability to sort N N -bit numbers using only 2 hidden layers of quadratic size [27]. So, while neural networks are related to conventional statistical models, they learn an intricate computation. Furthermore, large DNNs can be trained with supervised backpropagation whenever the labeled training set has enough information to specify the network's parameters. Thus, if there exists a parameter setting of a large DNN that achieves good results (for example, because humans can solve the task very rapidly), supervised backpropagation will find these parameters and solve the problem.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori. For example, speech recognition and machine translation are sequential problems. Likewise, question answering can also be seen as mapping a sequence of words representing the question to a

- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 RNN 아키텍처를 인코더-디코더로 사용

이 논문을 좀 더 자세히 보겠습니다.

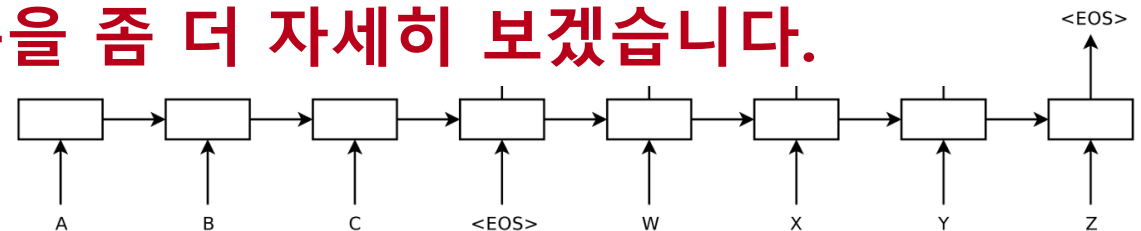


Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 인코더-디코더로 사용

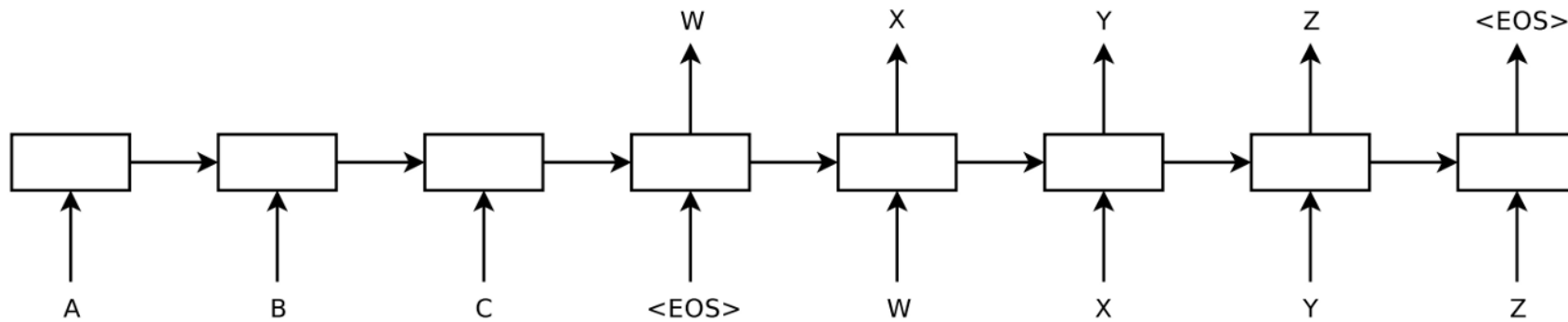
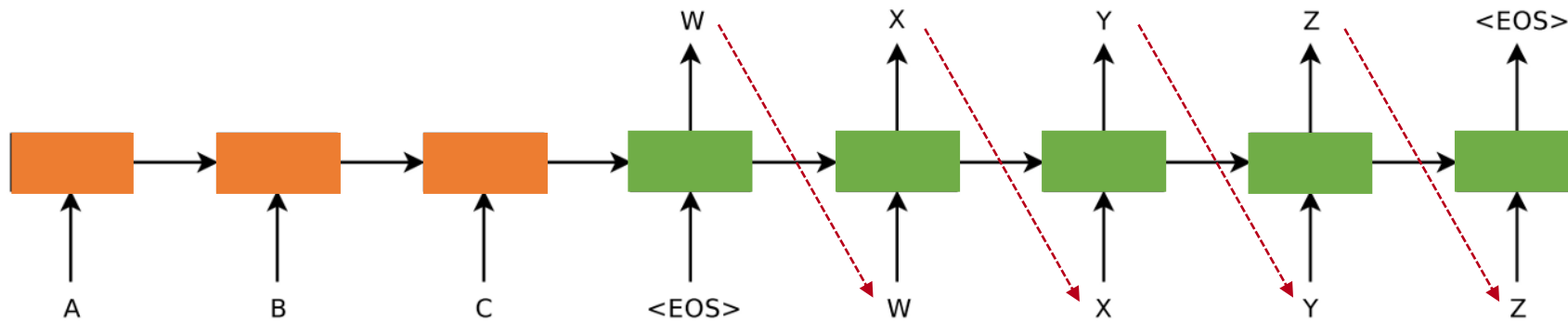


Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

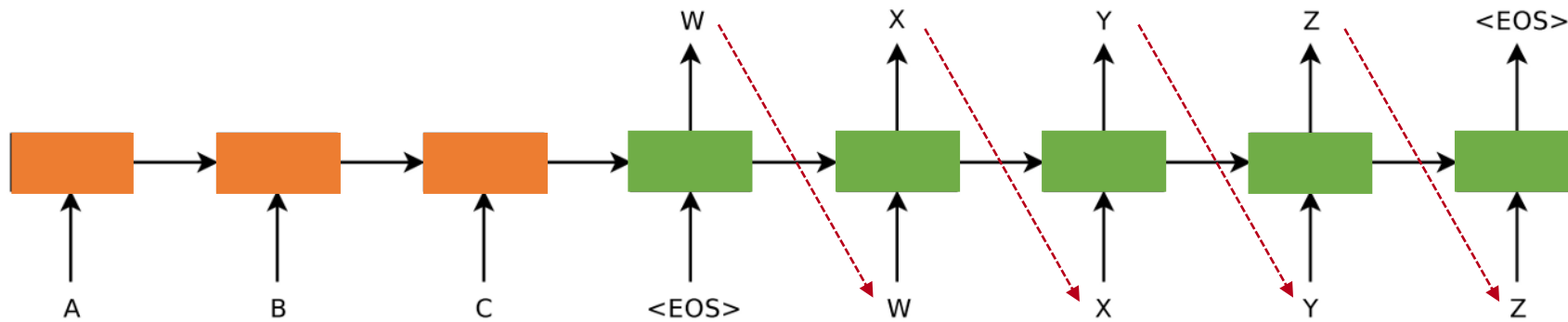
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



구체적인 동작 과정을 볼까요?

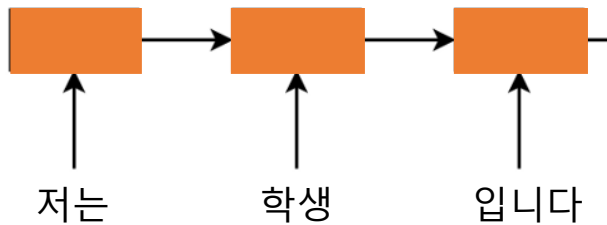
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



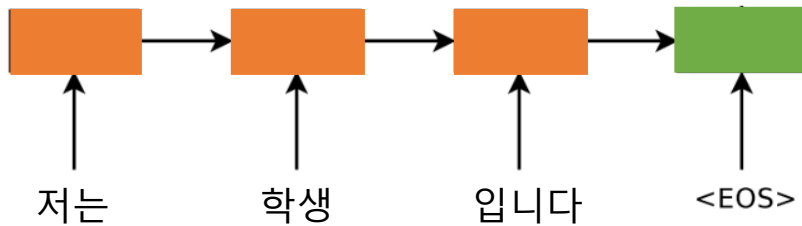
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



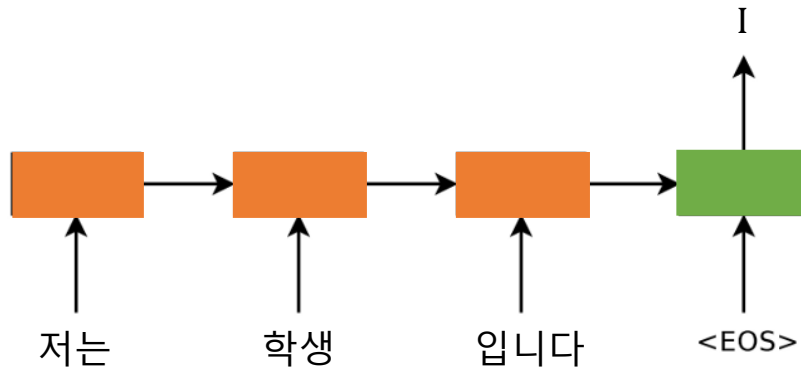
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



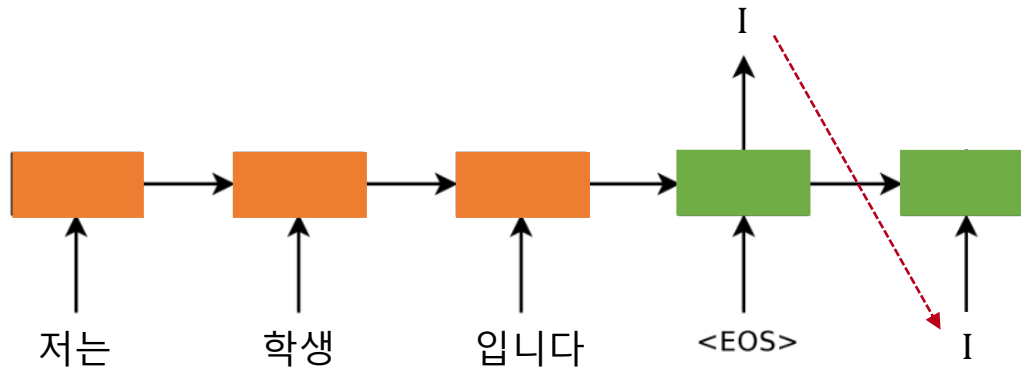
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



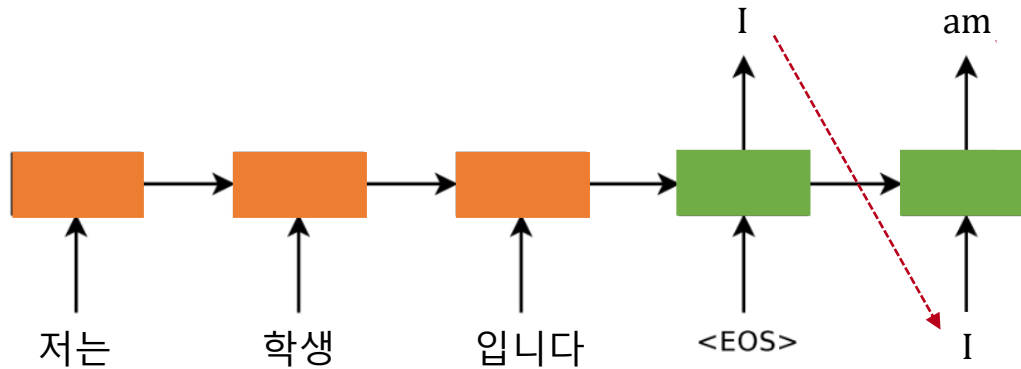
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



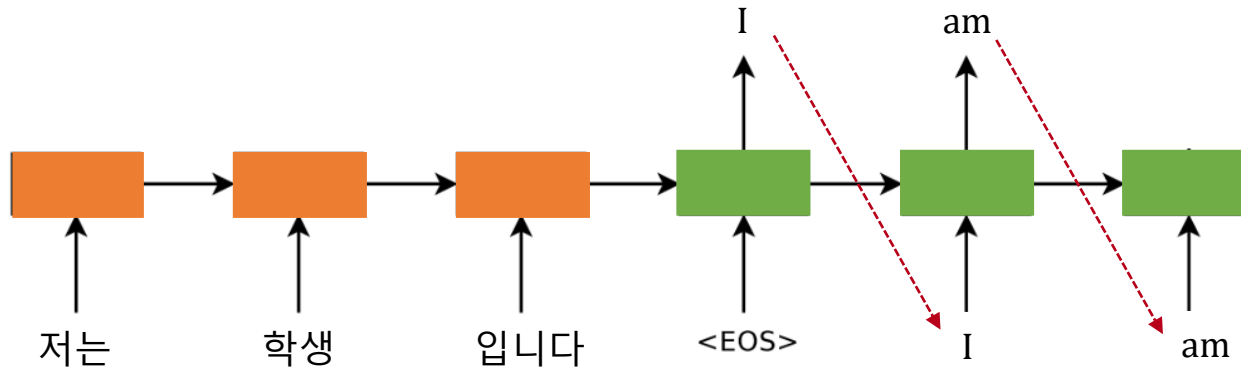
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



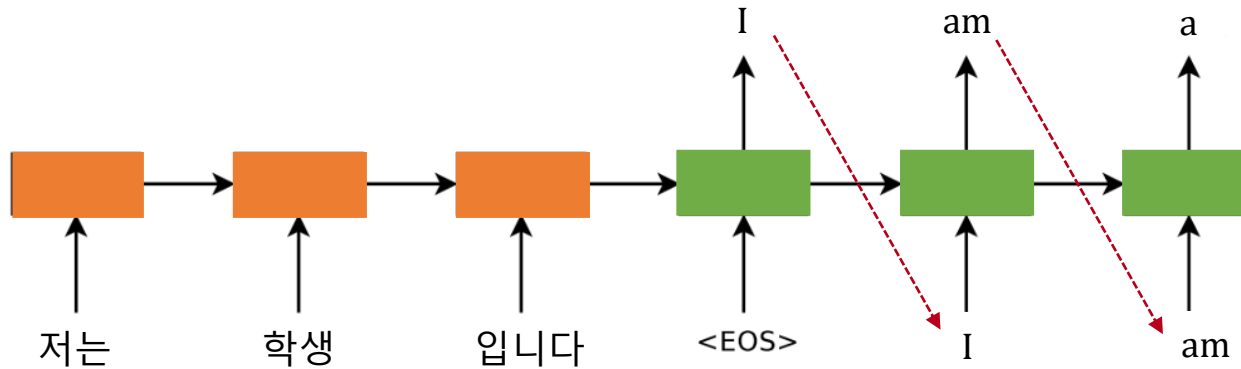
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



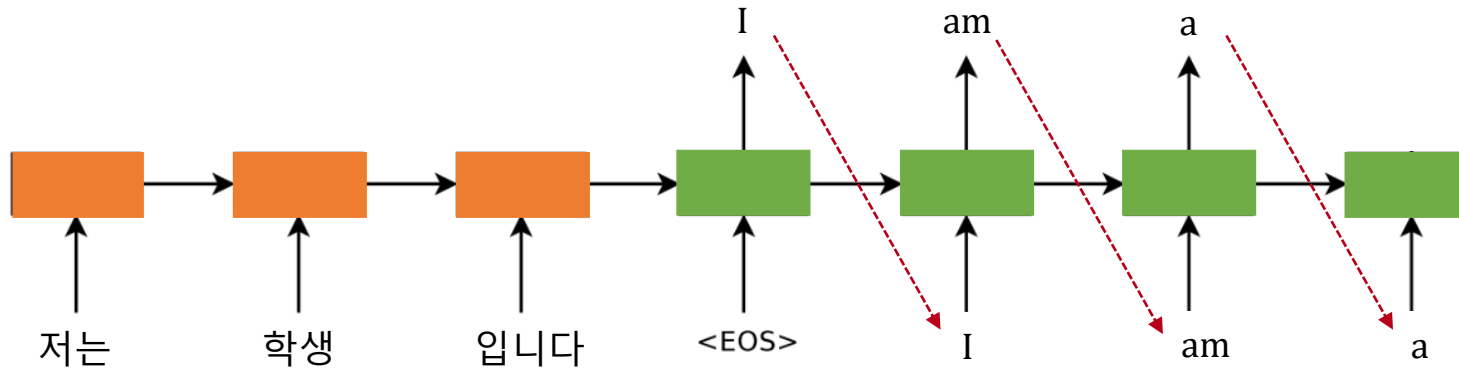
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



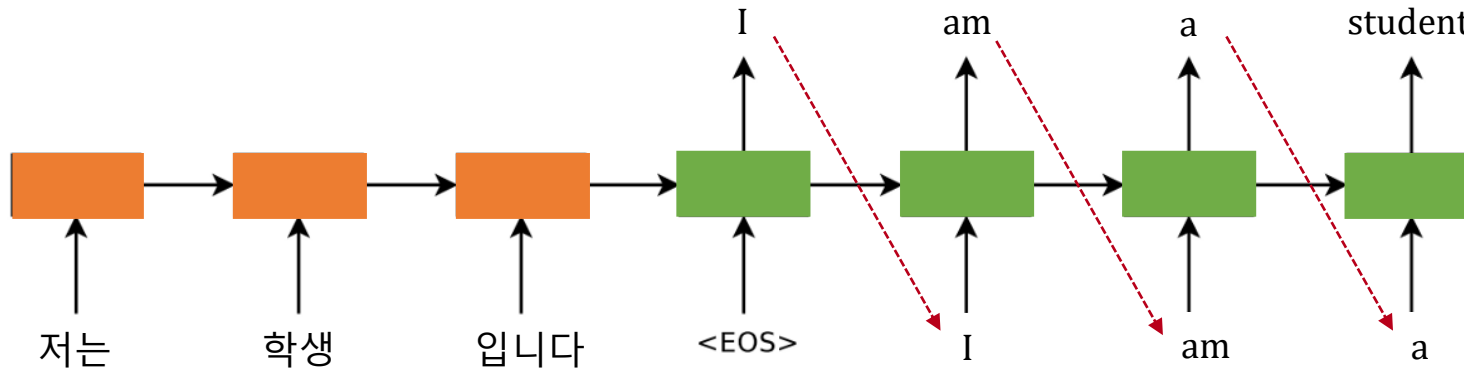
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



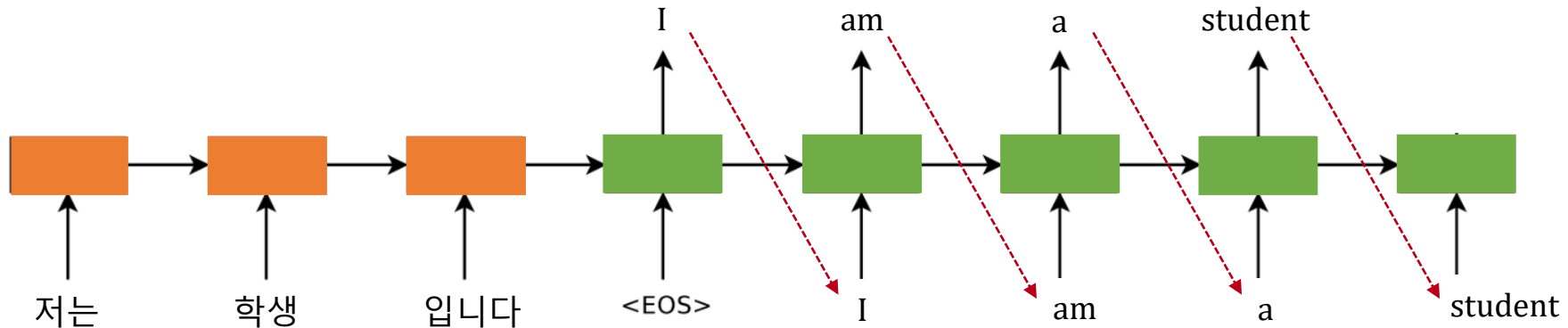
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



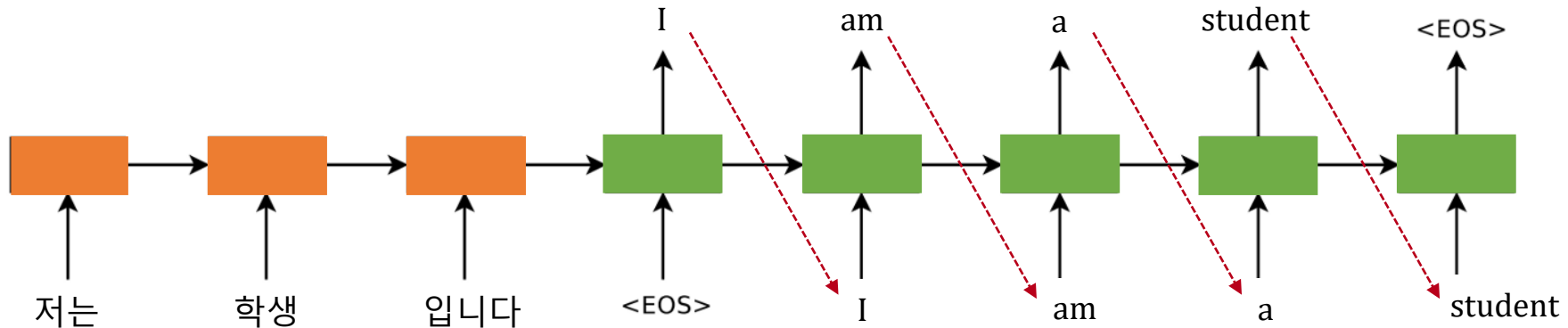
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



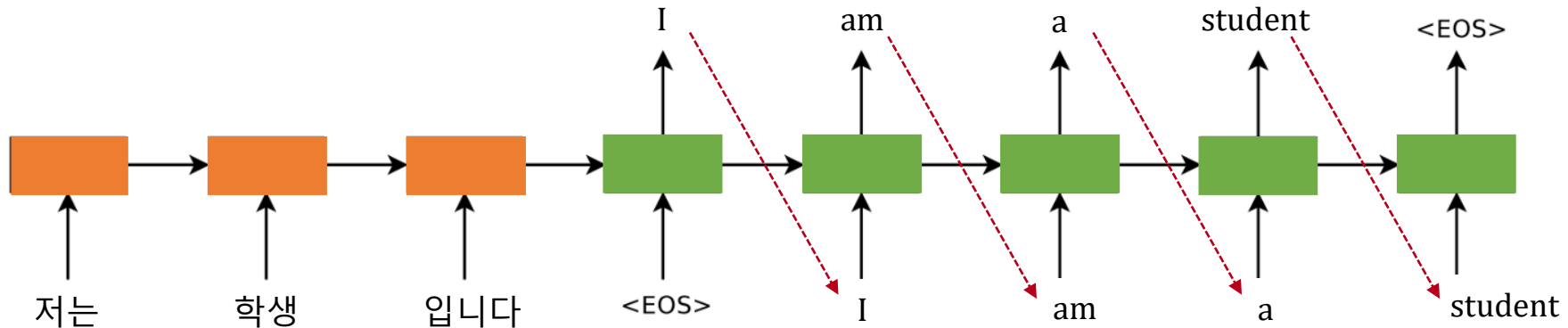
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



Sequence-To-Sequence, Seq2Seq

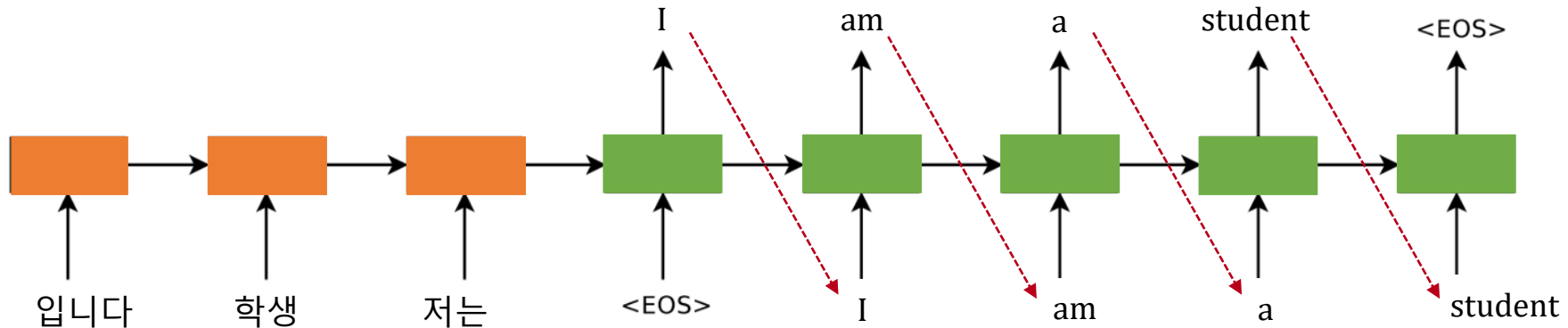
- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



인코더의 입력 시퀀스를 반대 순서로 넣는テクニック도 제안!

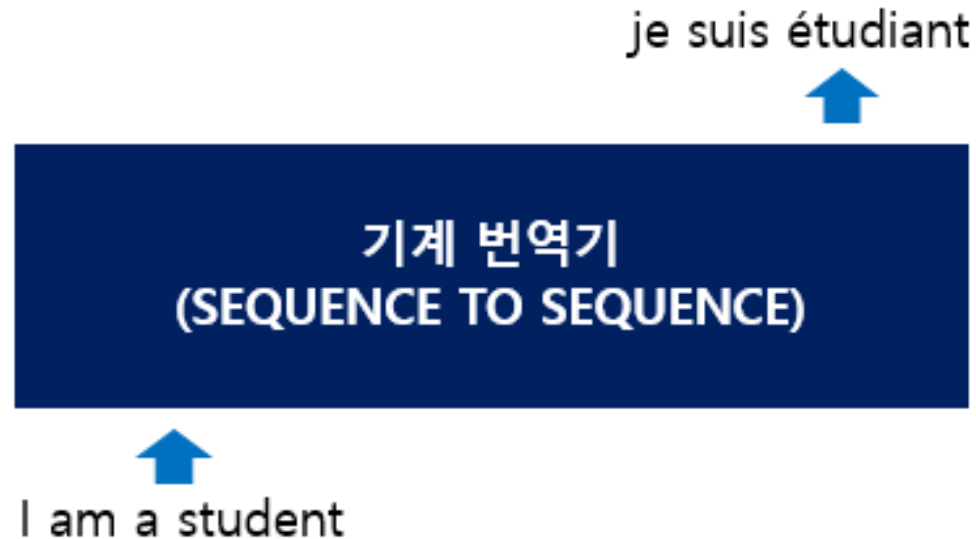
Sequence-To-Sequence, Seq2Seq

- Sequence to Sequence Learning with Neural Networks 논문
- 본격적인 신경망 기계 번역기를 제시
- 서로 다른 두 개의 LSTM 아키텍처를 각각 **인코더**-**디코더**로 사용



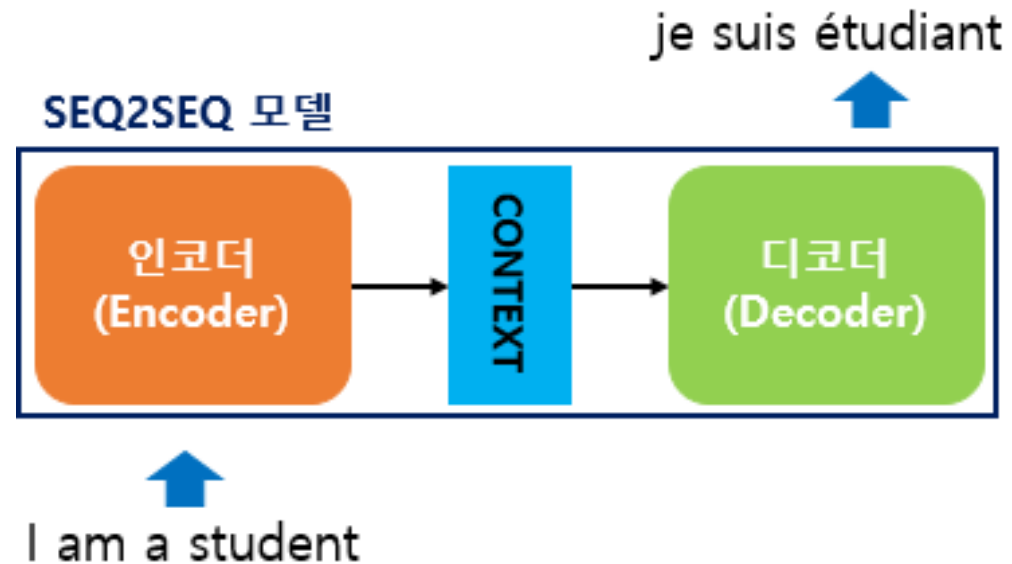
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력한다.
- Ex) 챗봇(Chatbot), 기계 번역(Machine Translation), 텍스트 요약(Text Summarization) 등..



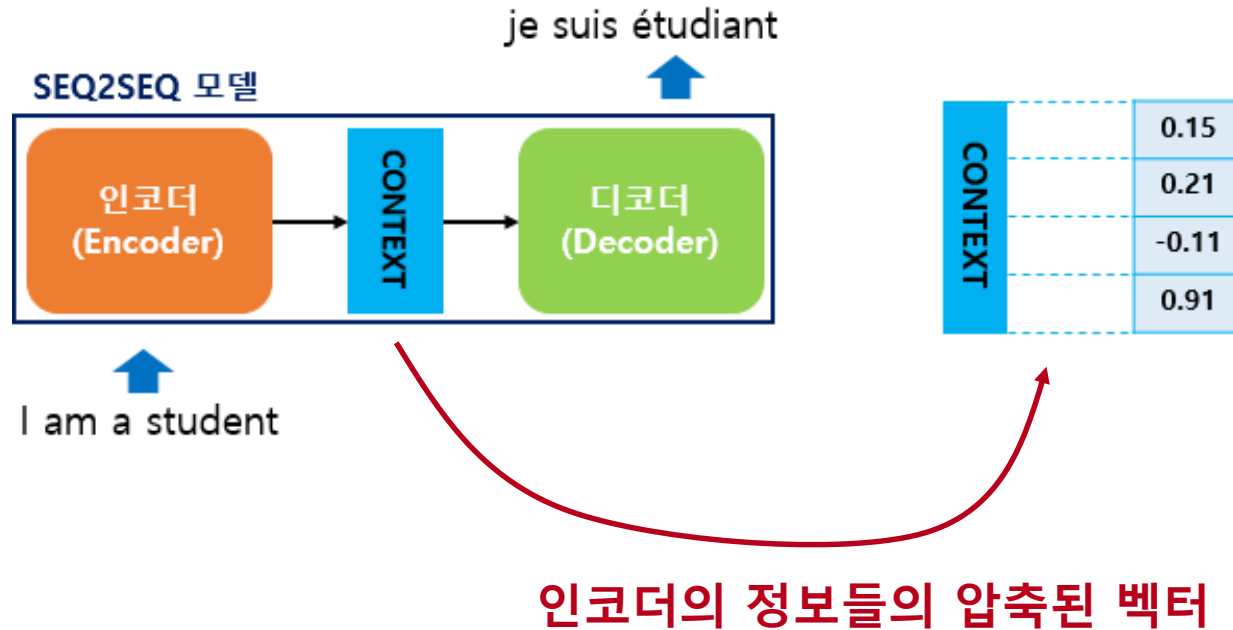
Sequence-To-Sequence, Seq2Seq

- Sequence-to-Sequence는 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력한다.
- Ex) 챗봇(Chatbot), 기계 번역(Machine Translation), 텍스트 요약(Text Summarization) 등..
- seq2seq는 내부적으로 인코더와 디코더 구조를 가지고 있다.



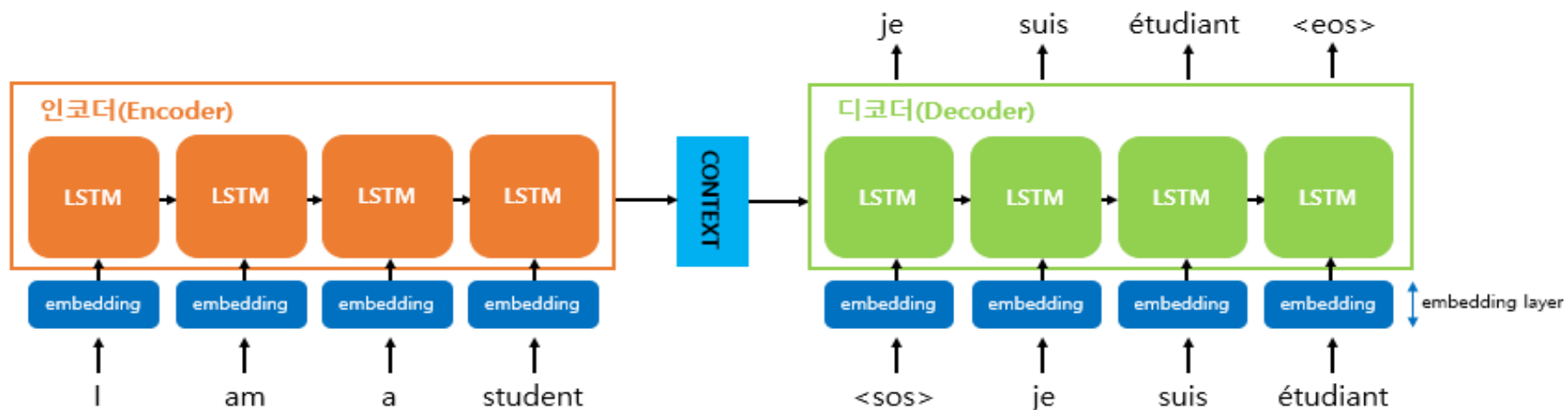
Sequence-To-Sequence, Seq2Seq

- 인코더는 입력 문장의 모든 단어들을 순차적으로 입력받은 뒤에 마지막에 이 모든 단어 정보들을 압축해서 컨텍스트 벡터(context vector)를 만든다.
- 디코더는 컨텍스트 벡터를 받아서 번역된 단어를 한 개씩 순차적으로 출력한다.



Sequence-To-Sequence, Seq2Seq

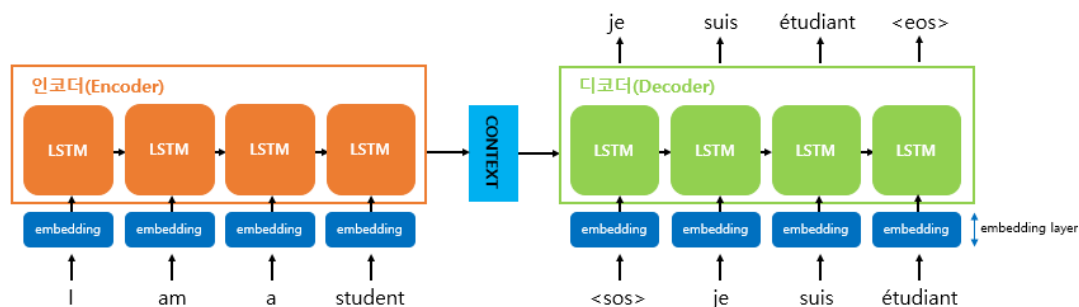
- 인코더의 마지막 은닉 상태를 컨텍스트 벡터라 부른다.
- 컨텍스트 벡터는 디코더 RNN 셀의 첫번째 은닉 상태로 사용된다.
- 인코더와 디코더의 각 시점의 입력은 기본적으로 임베딩 벡터.



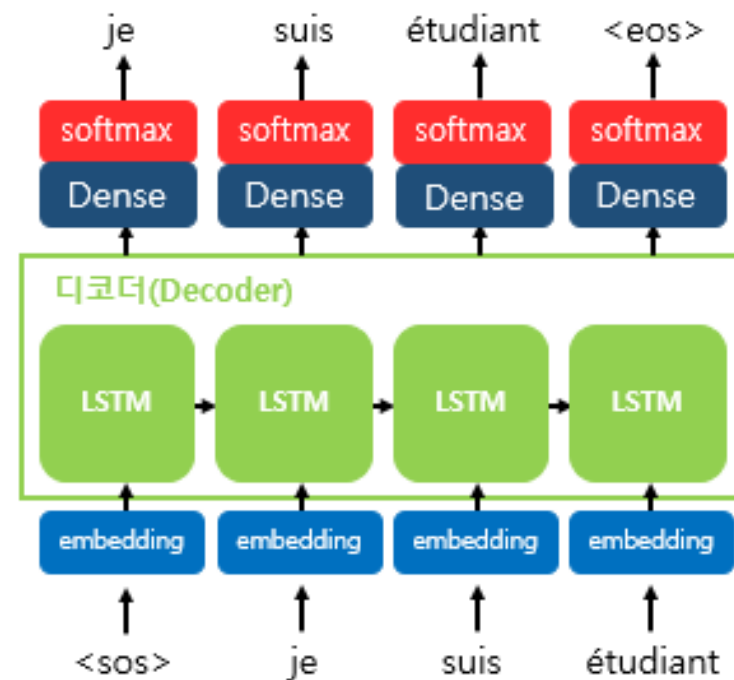
I		0.157	am		0.78	a		0.75	student		0.88
		-0.25			0.29			-0.81			-0.17
		0.478			-0.96			0.96			0.29
		-0.78			0.52			0.12			0.48

Sequence-To-Sequence, Seq2Seq

- 디코더는 RNN 언어 모델. (Teacher Forcing 사용!)
- <sos>는 시작 심볼이며, <eos>는 종료 심볼이다.



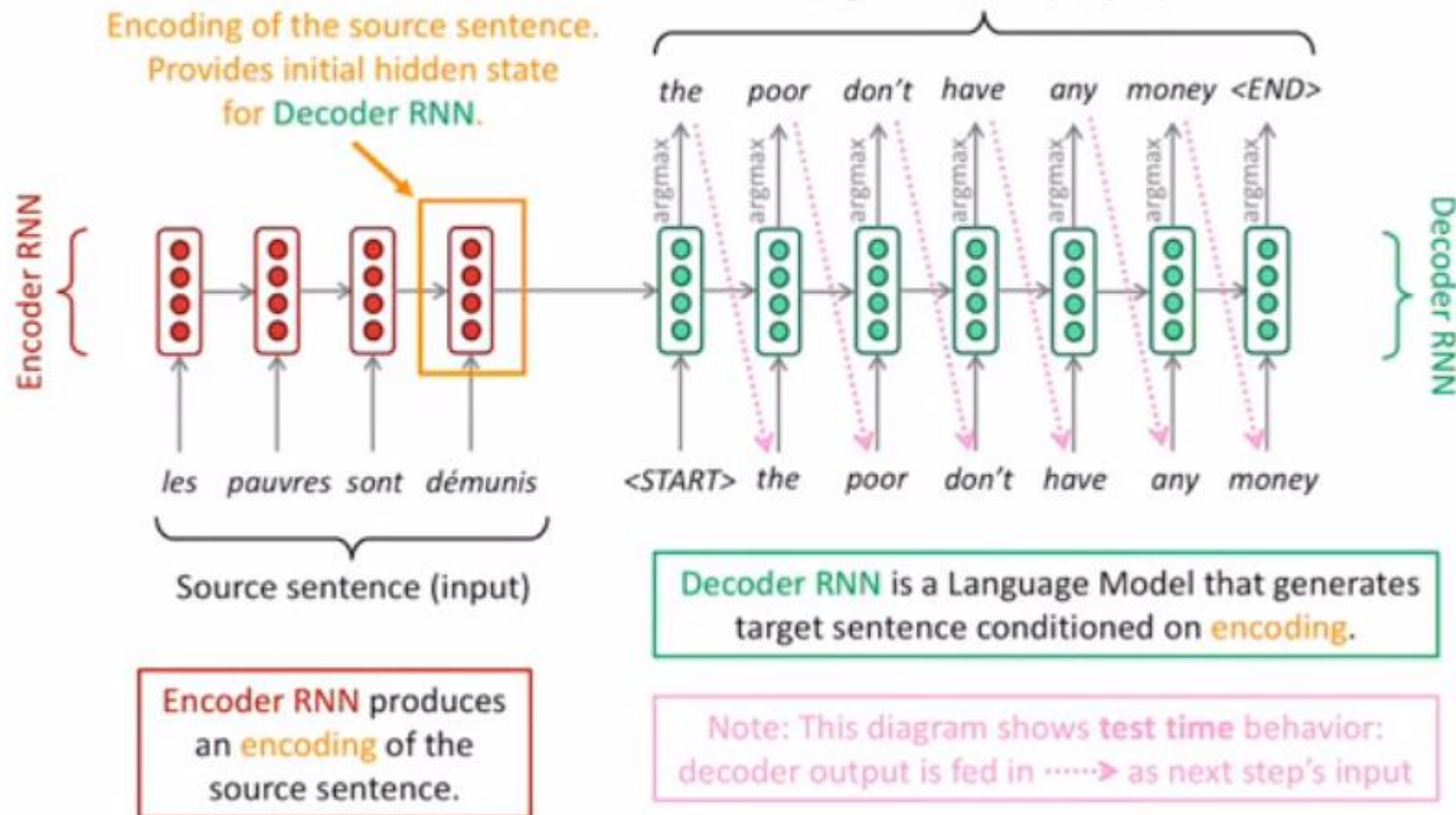
I		0.157
		-0.25
		0.478
		-0.78
am		0.78
		0.29
		-0.96
		0.52
a		0.75
		-0.81
		0.96
		0.12
student		0.88
		-0.17
		0.29
		0.48



Sequence-To-Sequence, Seq2Seq

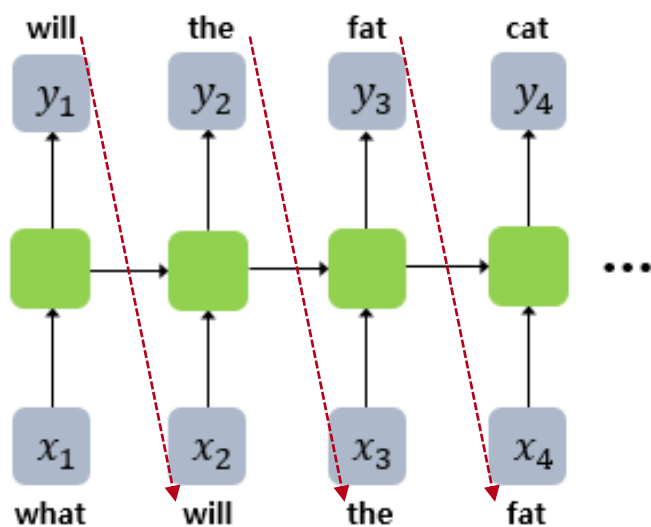
Neural Machine Translation (NMT)

The sequence-to-sequence model

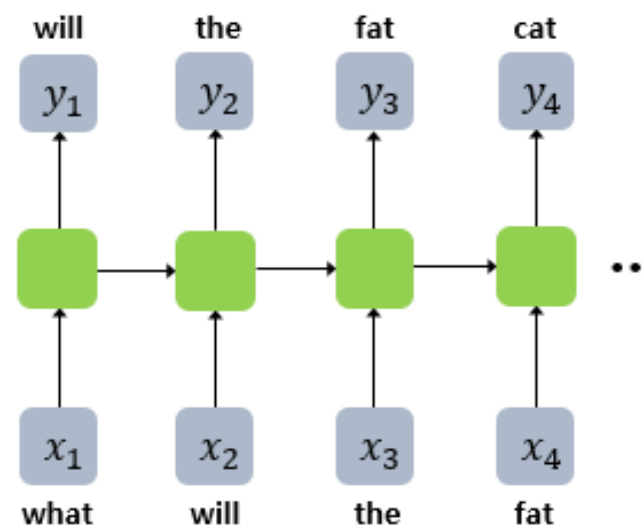


Teacher forcing

- 훈련 단계에서 Teacher Forcing을 무조건 하는 것이 아니라 비율을 정해서 수행. 이 비율은 하이퍼파라미터.
- 이 비율을 높게 설정할수록 빠른 학습이 가능해지지만
학습 데이터에 과적합(Overfitting)되어 테스트 단계에서 악영향을 줄 수 있음.
- 테스트 단계에서는 Teacher Forcing을 사용하지 않으며 현 시점의 출력을 다음 시점의 입력으로 사용.



Teacher Forcing 미사용

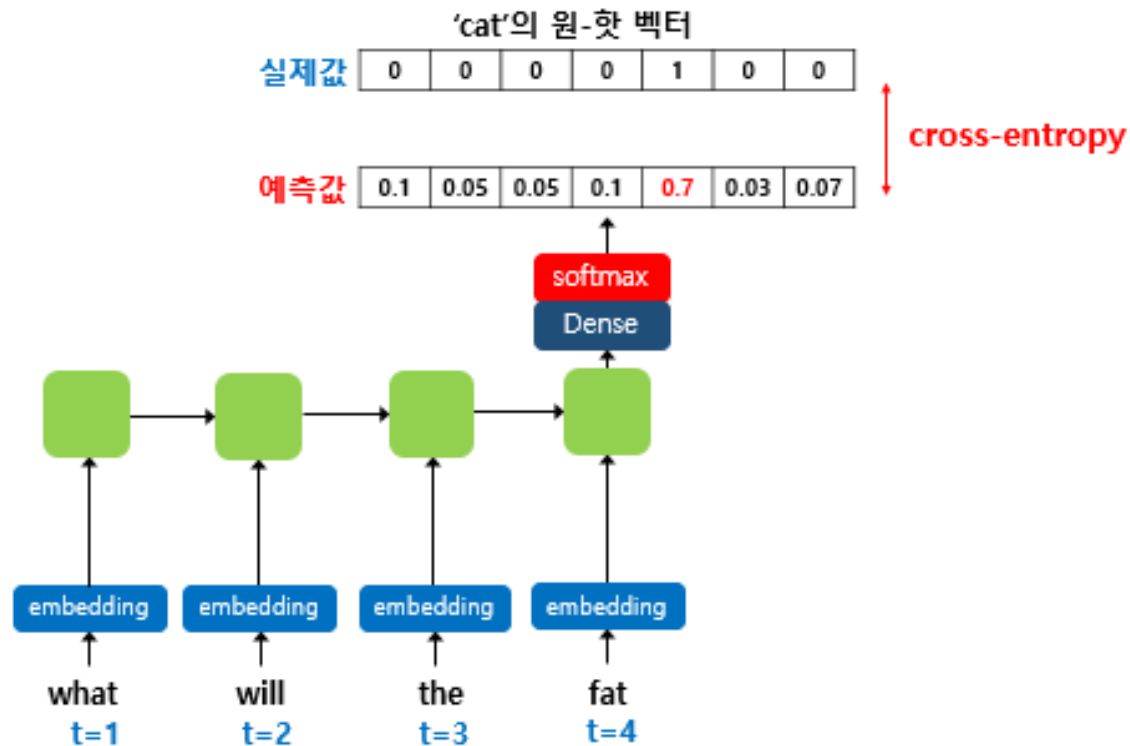


Teacher Forcing 사용

Beam Search

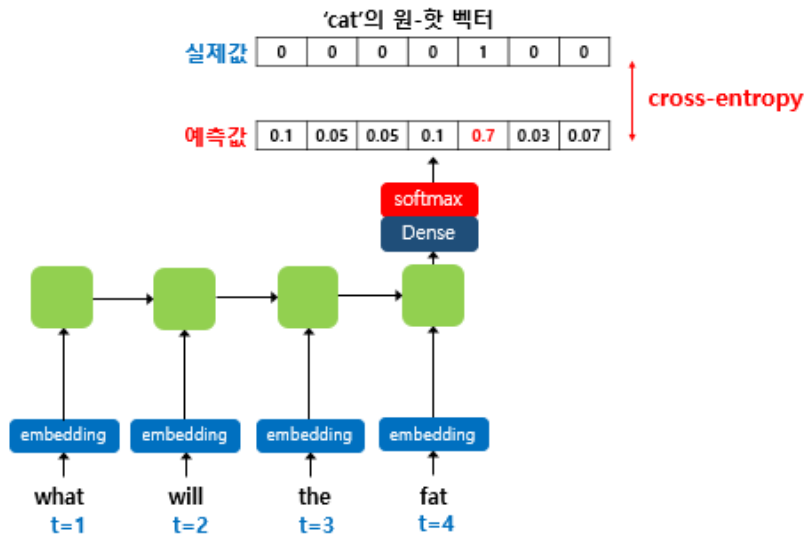
Greedy Decoding

- seq2seq의 디코더는 기본적으로는 RNN 언어 모델이다.
- 디코더(RNN 언어 모델)는 매 시점마다 가장 높은 확률을 가지는 단어를 선택한다.
- 이를 Greedy Decoding이라고 한다.



Greedy Decoding

- Greedy Decoding은 매 순간에서의 최적의 선택을 한다.
- 하지만 전체적으로 봤을 때는 그 순간의 선택이 최적의 선택이 아닐 수 있다.
- Greedy Decoding은 순간 잘못된 선택을 했더라도 그 결정을 취소할 수 없다.



인코더 입력 문장 :

les pauvres sont démunis

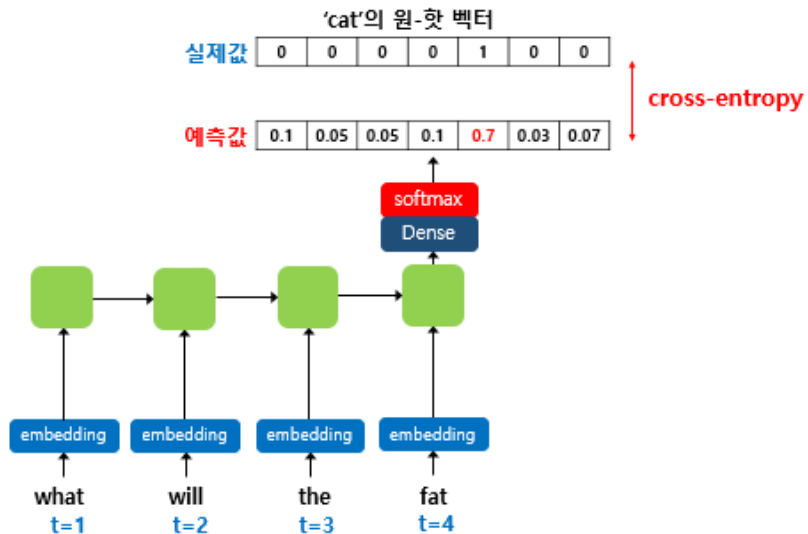
(실제 정답 : the poor don't have any money)

디코더 생성 문장 :

timestep 1: the

Greedy Decoding

- Greedy Decoding은 매 순간에서의 최적의 선택을 한다.
- 하지만 전체적으로 봤을 때는 그 순간의 선택이 최적의 선택이 아닐 수 있다.
- Greedy Decoding은 순간 잘못된 선택을 했더라도 그 결정을 취소할 수 없다.



인코더 입력 문장 :

les pauvres sont démunis

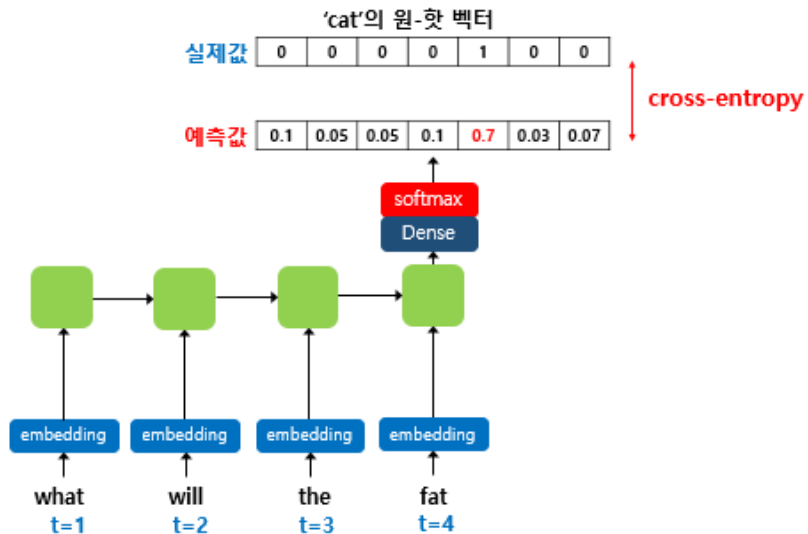
(실제 정답 : the poor don't have any money)

디코더 생성 문장 :

timestep 1: the
timestep 2: the poor

Greedy Decoding

- Greedy Decoding은 매 순간에서의 최적의 선택을 한다.
- 하지만 전체적으로 봤을 때는 그 순간의 선택이 최적의 선택이 아닐 수 있다.
- Greedy Decoding은 순간 잘못된 선택을 했더라도 그 결정을 취소할 수 없다.



인코더 입력 문장 :

les pauvres sont démunis

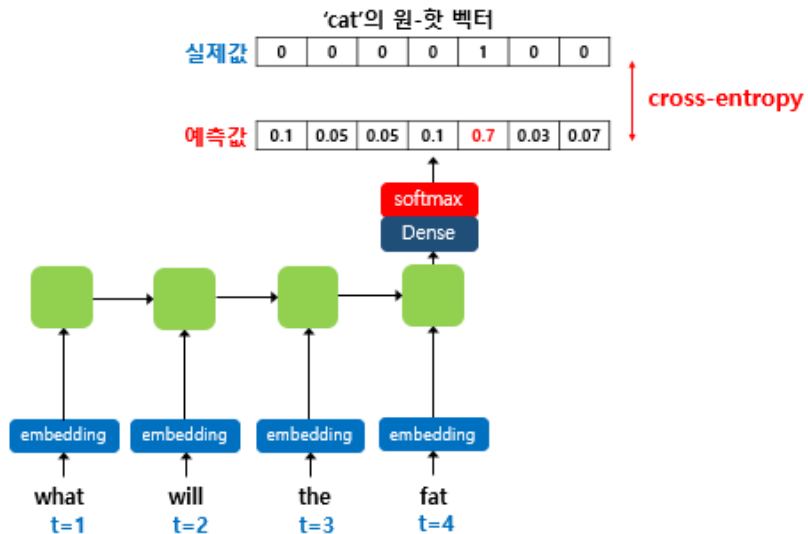
(실제 정답 : the poor don't have any money)

디코더 생성 문장 :

timestep 1: the
timestep 2: the poor
timestep 3: the poor **are**

Greedy Decoding

- Greedy Decoding은 매 순간에서의 최적의 선택을 한다.
- 하지만 전체적으로 봤을 때는 그 순간의 선택이 최적의 선택이 아닐 수 있다.
- Greedy Decoding은 순간 잘못된 선택을 했더라도 그 결정을 취소할 수 없다.



인코더 입력 문장 :

les pauvres sont démunis

(실제 정답 : the poor don't have any money)

디코더 생성 문장 :

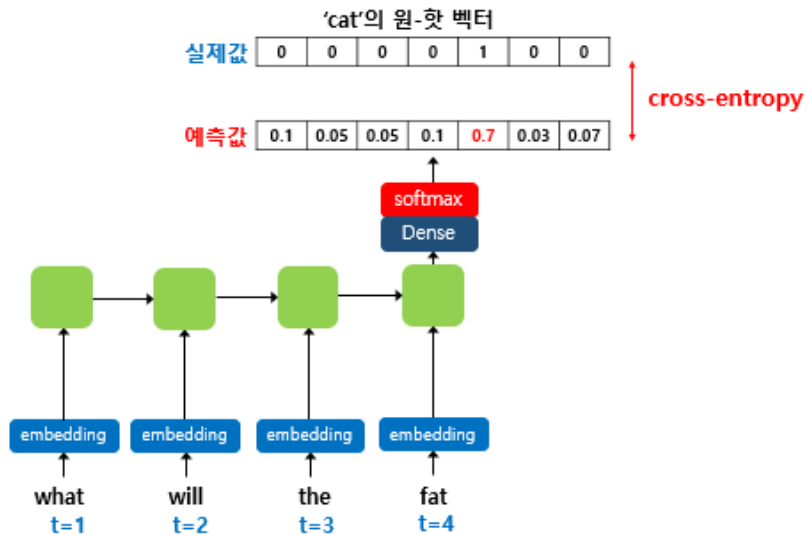
timestep 1: the

timestep 2: the poor

timestep 3: the poor are ← 뒤로 영향을 주는 잘못된 선택

Greedy Decoding

- Greedy Decoding은 매 순간에서의 최적의 선택을 한다.
- 하지만 전체적으로 봤을 때는 그 순간의 선택이 최적의 선택이 아닐 수 있다.
- Greedy Decoding은 순간 잘못된 선택을 했더라도 그 결정을 취소할 수 없다.



인코더 입력 문장 :

les pauvres sont démunis

(실제 정답 : the poor don't have any money)

디코더 생성 문장 :

timestep 1: the

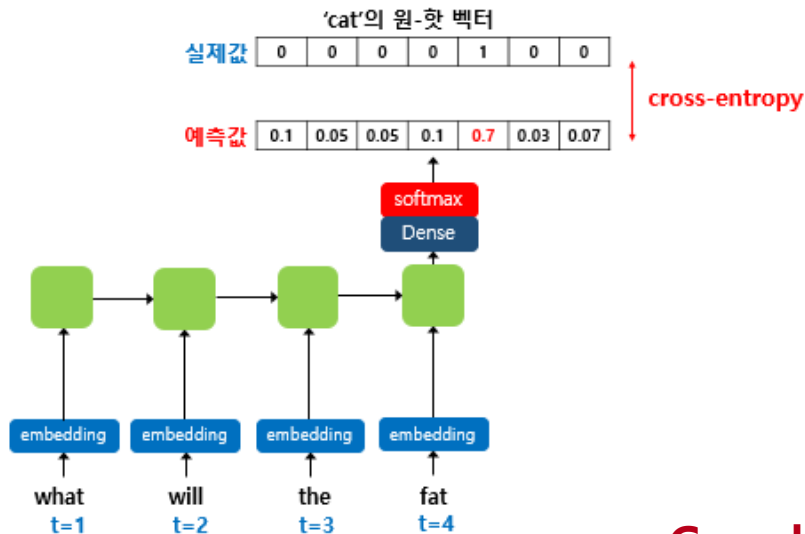
timestep 2: the poor

timestep 3: the poor are ← 뒤로 영향을 주는 잘못된 선택

timestep 4: the poor are ????

Greedy Decoding

- Greedy Decoding은 매 순간에서의 최적의 선택을 한다.
- 하지만 전체적으로 봤을 때는 그 순간의 선택이 최적의 선택이 아닐 수 있다.
- Greedy Decoding은 순간 잘못된 선택을 했더라도 그 결정을 취소할 수 없다.



인코더 입력 문장 :

les pauvres sont démunis

(실제 정답 : the poor don't have any money)

디코더 생성 문장 :

timestep 1: the

timestep 2: the poor

timestep 3: the poor are ← 뒤로 영향을 주는 잘못된 선택

timestep 4: the poor are ? ? ? ?

Greedy decoding 대신 Beam Search라는 알고리즘을 사용할 수 있다!
(몇 가지 가설을 세우고 최적의 선택지를 고른다.)

Searching Algorithm for Decoding

- **Greedy Search Decoding**

- : 매 순간에서의 최적의 선택을 한다.
- : 최적의 해를 보장할 수 없다.

- **Exhaustive Search Decoding**

- : 모든 가능한 조합의 경우를 전부 생성해보고 가장 확률이 높은 문장을 선택한다.
- : 너무 많은 리소스가 소요된다.

- **Beam Search Decoding**

- : 매 시점마다 가장 확률이 높은 k개의 다음 단어를 선택 후, 다음 시점 단어들의 확률 예측.
- : $k * \text{Vocab-size}$ 개의 후보군 중 다시 확률이 높은 k개의 후보군만 선택하고 나머지 단어는 제거
- : 최종적으로 $k*k$ 개의 후보군 중에서 가장 확률이 높은 k개의 후보군만을 유지
- : Beam Search Decoding 또한 항상 최적해를 보장하지는 않지만 Exhaustive search보다 효율적.
- : Greedy Search Decoding이 놓칠 수 있는 더 나은 후보군을 유지할 수 있음.

Searching Algorithm for Decoding

- **Greedy Search Decoding**

- : 매 순간에서의 최적의 선택을 한다.
- : 최적의 해를 보장할 수 없다.

- **Exhaustive Search Decoding**

- : 모든 가능한 조합의 경우를 전부 생성해서 그 가장 확률이 높은 조합을 선택한다.
 - : 너무 많은 리소스가 소모된다.
- k=3인 경우의 예제를 보겠습니다.**

- **Beam Search Decoding**

- : 매 시점마다 가장 확률이 높은 k개의 다음 단어를 선택 후, 다음 시점 단어들의 확률 예측.
- : $k * \text{Vocab-size}$ 개의 후보군 중 다시 확률이 높은 k개의 후보군만 선택하고 나머지 단어는 제거
- : 최종적으로 $k*k$ 개의 후보군 중에서 가장 확률이 높은 k개의 후보군만을 유지
- : Beam Search Decoding 또한 항상 최적해를 보장하지는 않지만 Exhaustive search보다 효율적.
- : Greedy Search Decoding이 놓칠 수 있는 더 나은 후보군을 유지할 수 있음.

Beam Search Decoding (k=3)

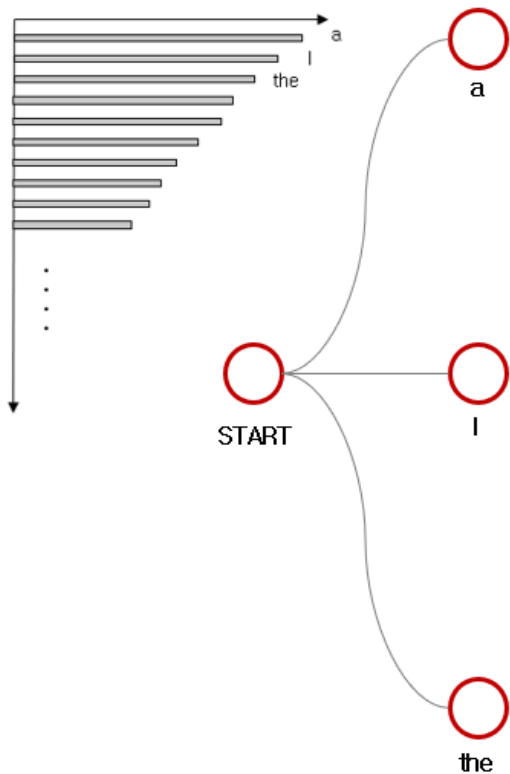
- 디코더의 매 스텝마다, 정답일 확률이 높은 k개의 선택지를 추적한다. (이를 가설이라 한다.)
- 여기서 k는 보통 5 ~ 10의 크기를 사용한다. (10 이상의 k는 성능 개선의 효과가 거의 없다.)



디코더의 첫번째 시점에 <START> 토큰이 입력된다.
(그 외 <SOS>, <S>, <GO>라고도 불리기도 한다.)

Beam Search Decoding (k=3)

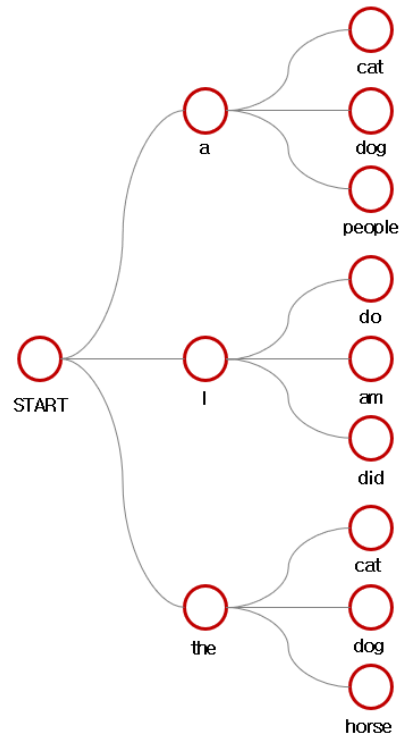
- 디코더의 매 스텝마다, 정답일 확률이 높은 k개의 선택지를 추적한다. (이를 가설이라 한다.)
- 여기서 k는 보통 5 ~ 10의 크기를 사용한다. (10 이상의 k는 성능 개선의 효과가 거의 없다.)



<START> 토큰이 입력되면 디코더는 출력층에서 가장 확률이 높은 k개의 단어를 고른다.

Beam Search Decoding (k=3)

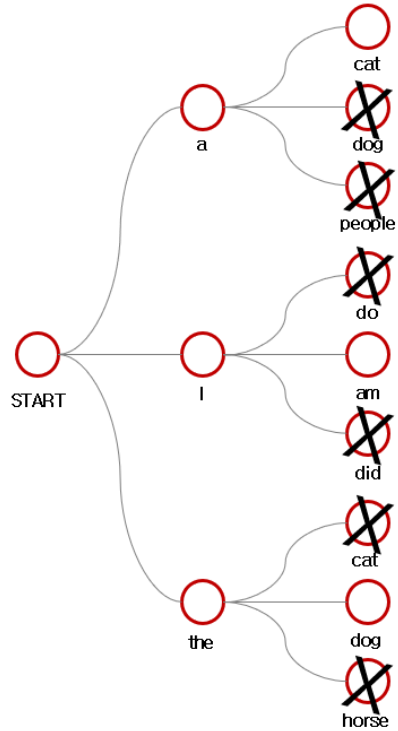
- 디코더의 매 스텝마다, 정답일 확률이 높은 k개의 선택지를 추적한다. (이를 가설이라 한다.)
- 여기서 k는 보통 5 ~ 10의 크기를 사용한다. (10 이상의 k는 성능 개선의 효과가 거의 없다.)



선택된 k개의 단어 각각에 대해서 다음 시점에서
또 다시 가장 높은 확률의 k개의 단어를 고른다.

Beam Search Decoding (k=3)

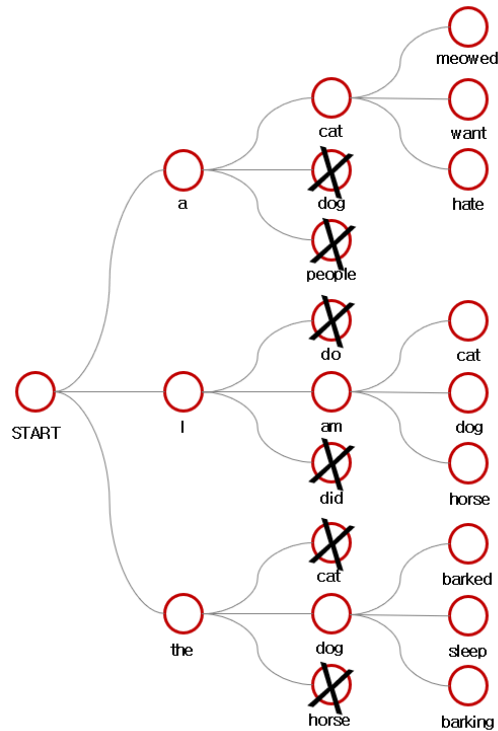
- 디코더의 매 스텝마다, 정답일 확률이 높은 k개의 선택지를 추적한다. (이를 가설이라 한다.)
- 여기서 k는 보통 5 ~ 10의 크기를 사용한다. (10 이상의 k는 성능 개선의 효과가 거의 없다.)



선택된 k개의 단어 각각에 대해서 다음 시점에서
또 다시 가장 높은 확률의 k개의 단어를 고른다.
해당 시점에서 **누적 확률** 순으로 상위 k개를 선택한다.

Beam Search Decoding (k=3)

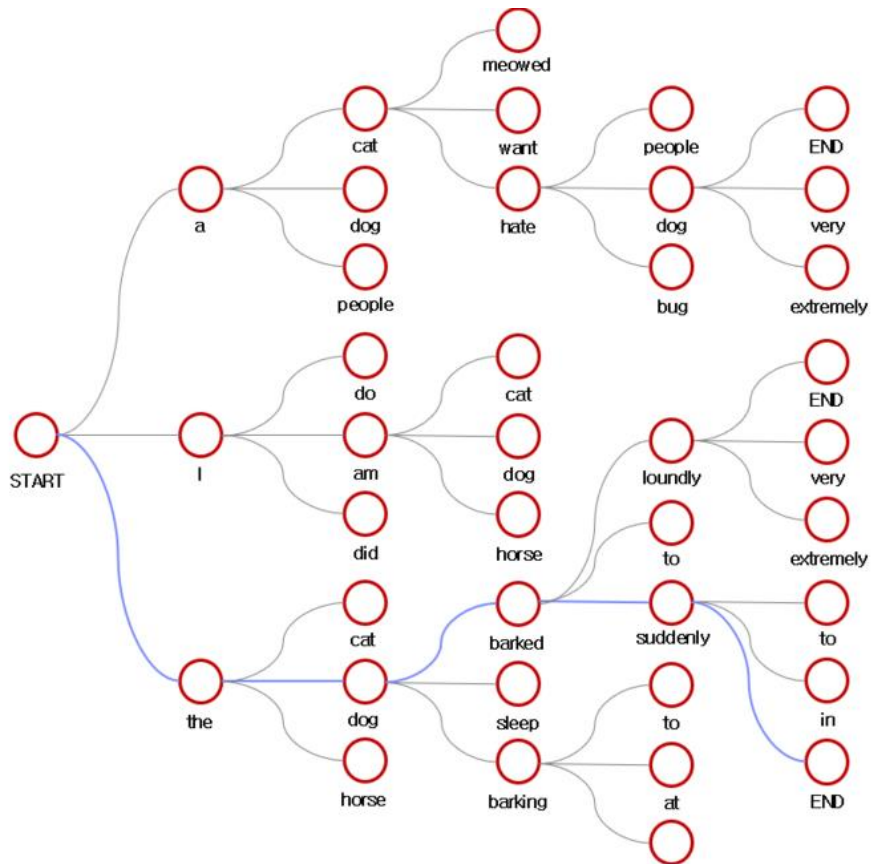
- 디코더의 매 스텝마다, 정답일 확률이 높은 k개의 선택지를 추적한다. (이를 가설이라 한다.)
- 여기서 k는 보통 5 ~ 10의 크기를 사용한다. (10 이상의 k는 성능 개선의 효과가 거의 없다.)



선택된 k개의 단어 각각에 대해서 다음 시점에서
또 다시 가장 높은 확률의 k개의 단어를 고른다.

Beam Search Decoding (k=3)

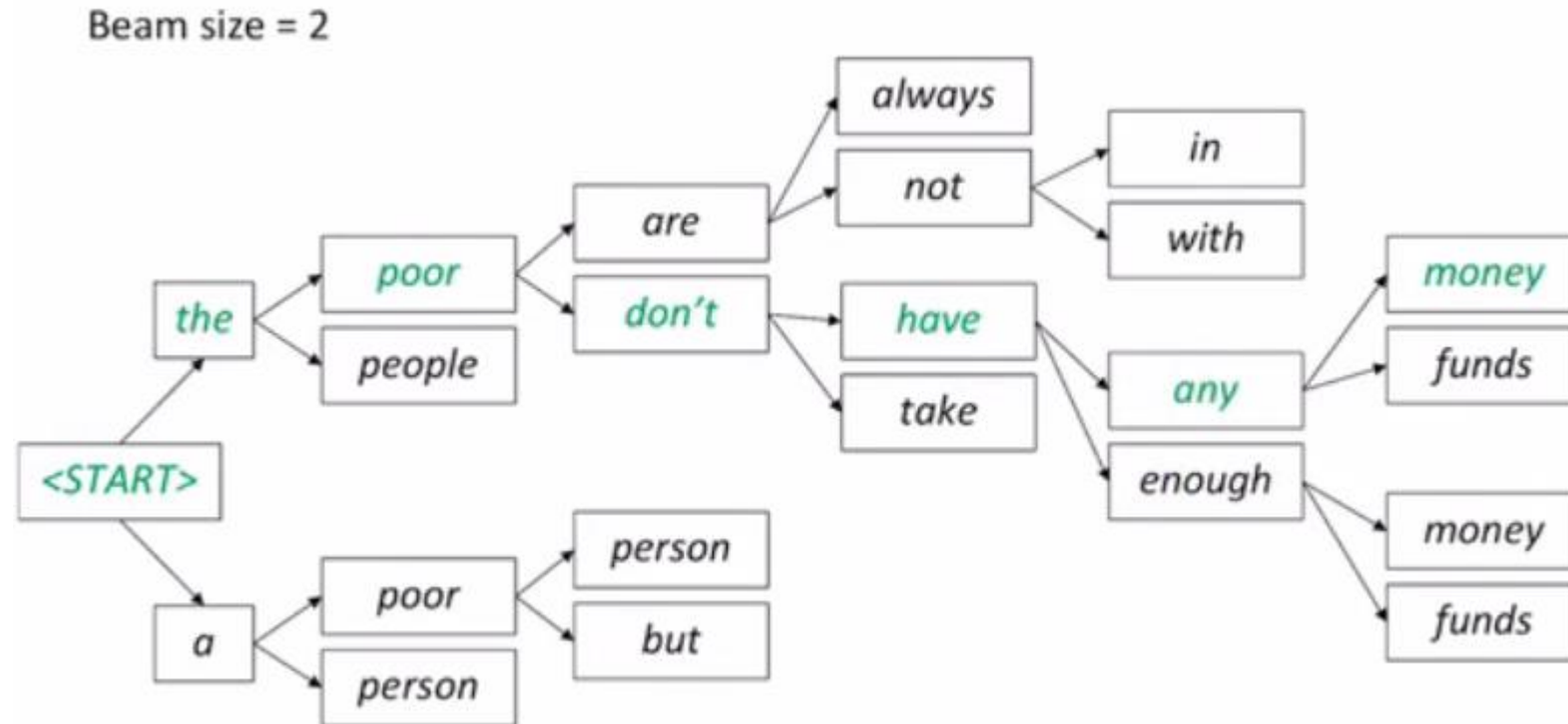
- 디코더의 매 스텝마다, 정답일 확률이 높은 k개의 선택지를 추적한다. (이를 가설이라 한다.)
- 여기서 k는 보통 5 ~ 10의 크기를 사용한다. (10 이상의 k는 성능 개선의 효과가 거의 없다.)



**<eos>를 만난 경우가 k개가 될 때까지 이를 반복한다.
<eos>가 선택되는 경우가 생기면
이는 최종 선택 후보가 된다.**

Beam Search Decoding (k=2)

- 다음은 k가 2인 경우의 또 다른 예제를 보여준다.



Subword Tokenization

Subword Tokenization

- 기계 번역기의 단어 집합의 크기는 30,000 ~ 50,000의 한정된 크기를 가지지만 현실에서의 단어 수는 이보다 훨씬 많다.
- 이로 인해 단어 집합(Vocabulary)에 없는 (Out-Of-Vocabulary, OOV) 문제가 발생한다.
- 이 논문은 Byte Pair Encoding이라는 서브워드 단위(Subword unit)의 인코딩을 제안한다.

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and Barry Haddow and Alexandra Birch

School of Informatics, University of Edinburgh

{rico.sennrich,a.birch}@ed.ac.uk,bhaddow@inf.ed.ac.uk

Abstract

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is based on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). We discuss the suitability of different word segmentation techniques, including simple character n -gram models and a segmentation based on the *byte pair encoding* compression algorithm, and empirically show that subword models improve over a back-off dictionary baseline for the WMT 15 translation tasks English→German and English→Russian by up to 1.1 and 1.3 BLEU, respectively.

Introduction

Neural machine translation has recently shown impressive results (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). However, the translation of rare words is an open problem. The vocabulary of neural models is typically limited to 30 000–50 000 words, but translation is an open-vocabulary prob-

lem, and especially for languages with productive word formation processes such as agglutination and compounding, translation models require mechanisms that go below the word level. As an example, consider compounds such as the German *Abwasserbehandlungsanlage* ‘sewage water treatment plant’, for which a segmented, variable-length representation is intuitively more appealing than encoding the word as a fixed-length vector.

For word-level NMT models, the translation of out-of-vocabulary words has been addressed through a back-off to a dictionary look-up (Jean et al., 2015; Luong et al., 2015b). We note that such techniques make assumptions that often do not hold true in practice. For instance, there is not always a 1-to-1 correspondence between source and target words because of variance in the degree of morphological synthesis between languages, like in our introductory compounding example. Also, word-level models are unable to translate or generate unseen words. Copying unknown words into the target text, as done by (Jean et al., 2015; Luong et al., 2015b), is a reasonable strategy for names, but morphological changes and transliteration is often required, especially if alphabets differ.

We investigate NMT models that operate on the level of subword units. Our main goal is to model open-vocabulary translation in the NMT network itself, without requiring a back-off model for rare words. In addition to making the translation process simpler, we also find that the subword models achieve better accuracy for the translation of rare words than large-vocabulary models and back-off dictionaries, and are able to productively generate new words that were not seen at training time. Our analysis shows that the neural networks are able to learn compounding and transliteration from subword representations.

This paper has two main contributions:

- We show that open-vocabulary neural ma-

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. and Samsung R&D Institute Poland.

Subword Tokenization

- 기계 번역기의 단어 집합의 크기는 30,000 ~ 50,000의 한정된 크기를 가지지만 현실에서의 단어 수는 이보다 훨씬 많다.
 - 이로 인해 단어 집합(Vocabulary)에 없는 (Out-Of-Vocabulary, OOV) 문제가 발생한다.
 - 이 논문은 Byte Pair Encoding이라는 서브워드 단위(Subword unit)의 인코딩을 제안한다.
- + 이후
- Byte Pair Encoding의 여러 변형 Subword Tokenizing 알고리즘들이 이어서 제안되었다.
 - 기계 번역기에서 **Subword Tokenizing 알고리즘**의 사용은 이제 기본이다.

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and Barry Haddow and Alexandra Birch
School of Informatics, University of Edinburgh

{rico.sennrich,a.birch}@ed.ac.uk,bhaddow@inf.ed.ac.uk

Abstract

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is based on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). We discuss the suitability of different word segmentation techniques, including simple character n -gram models and a segmentation based on the *byte pair encoding* compression algorithm, and empirically show that subword models improve over a back-off dictionary baseline for the WMT 15 translation tasks English→German and English→Russian by up to 1.1 and 1.3 BLEU, respectively.

Introduction

Neural machine translation has recently shown impressive results (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). However, the translation of rare words is an open problem. The vocabulary of neural models is typically limited to 30 000–50 000 words, but translation is an open-vocabulary problem,

and especially for languages with productive word formation processes such as agglutination and compounding, translation models require mechanisms that go below the word level. As an example, consider compounds such as the German *Abwasserbehandlungsanlage* 'sewage water treatment plant', for which a segmented, variable-length representation is intuitively more appealing than encoding the word as a fixed-length vector.

For word-level NMT models, the translation of out-of-vocabulary words has been addressed through a back-off to a dictionary look-up (Jean et al., 2015; Luong et al., 2015b). We note that such techniques make assumptions that often do not hold true in practice. For instance, there is not always a 1-to-1 correspondence between source and target words because of variance in the degree of morphological synthesis between languages, like in our introductory compounding example. Also, word-level models are unable to translate or generate unseen words. Copying unknown words into the target text, as done by (Jean et al., 2015; Luong et al., 2015b), is a reasonable strategy for names, but morphological changes and transliteration is often required, especially if alphabets differ.

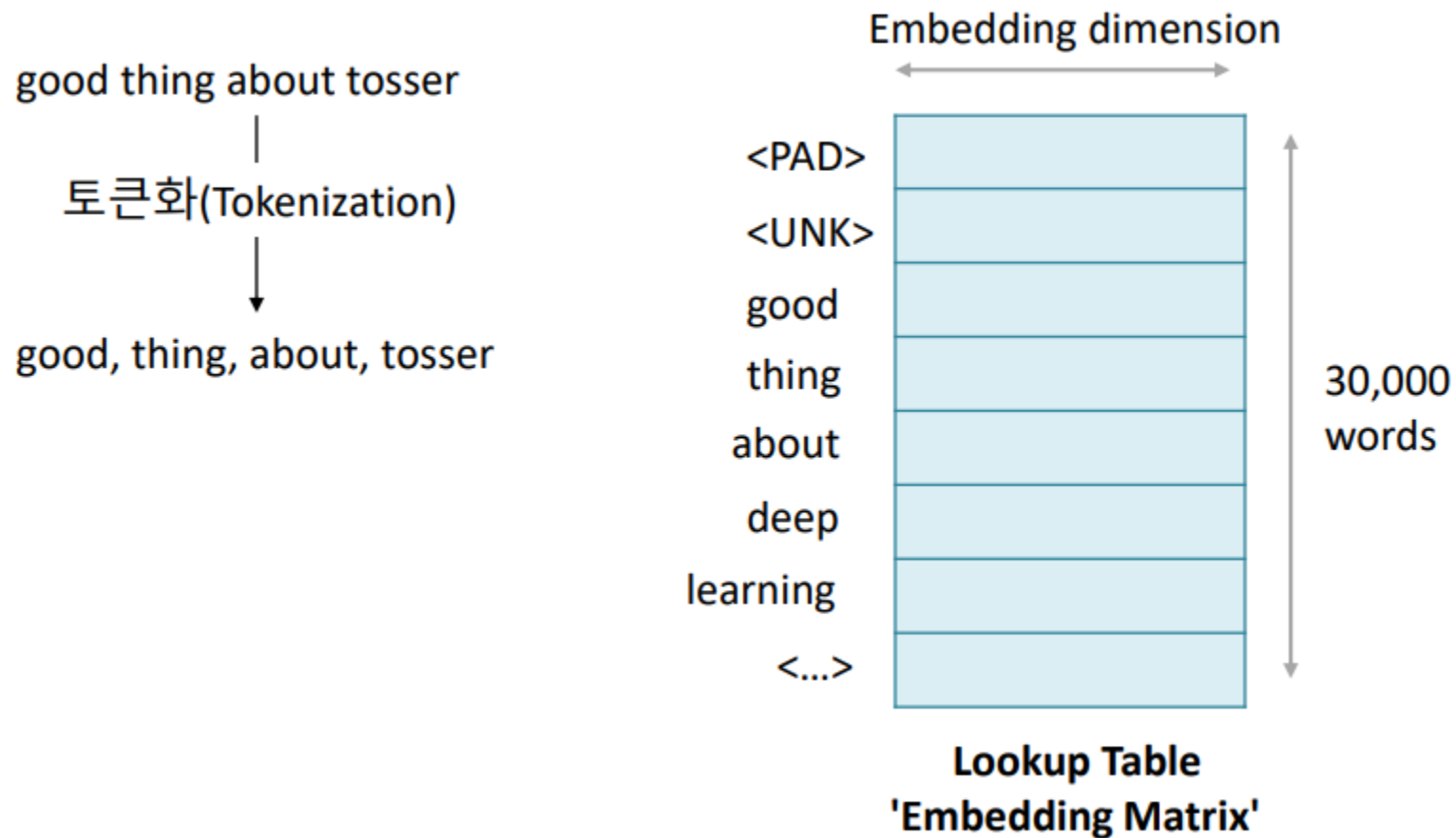
We investigate NMT models that operate on the level of subword units. Our main goal is to model open-vocabulary translation in the NMT network itself, without requiring a back-off model for rare words. In addition to making the translation process simpler, we also find that the subword models achieve better accuracy for the translation of rare words than large-vocabulary models and back-off dictionaries, and are able to productively generate new words that were not seen at training time. Our analysis shows that the neural networks are able to learn compounding and transliteration from subword representations.

This paper has two main contributions:

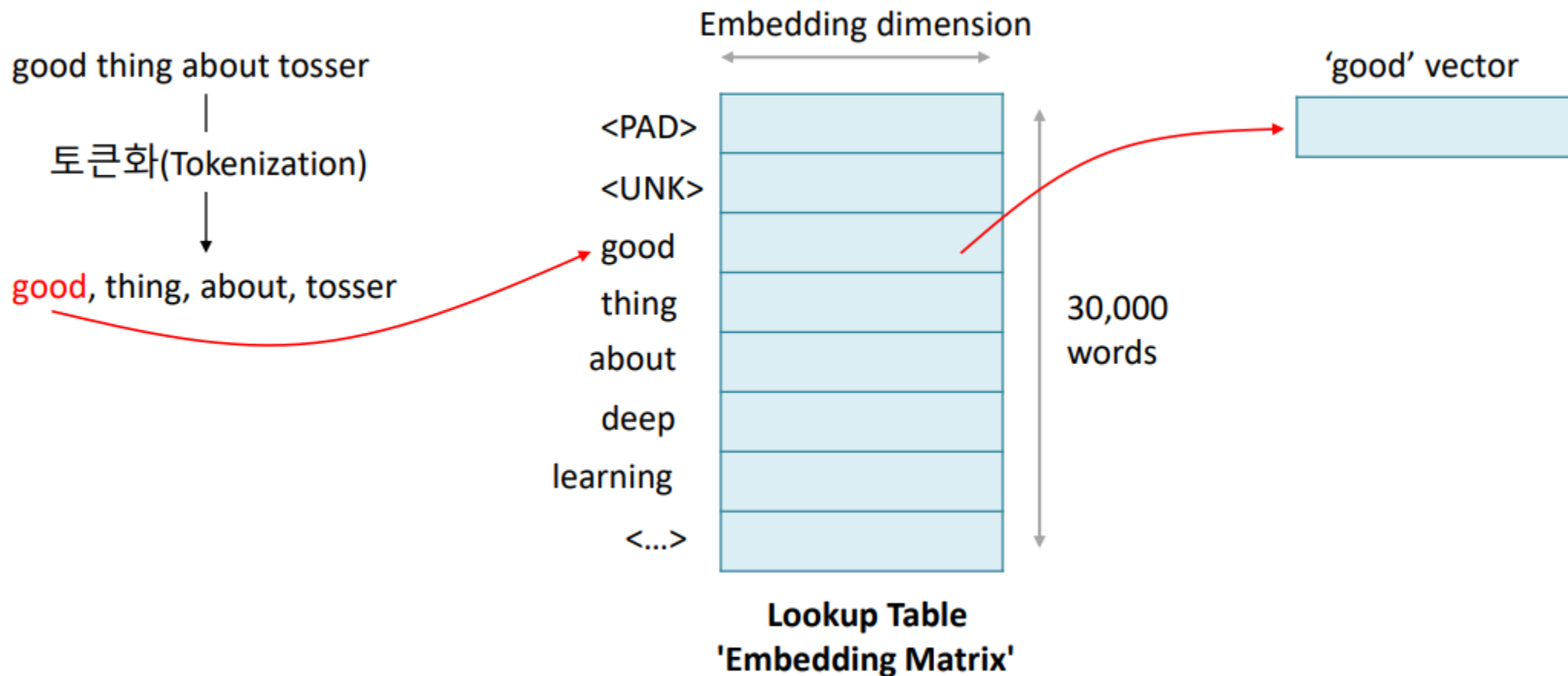
- We show that open-vocabulary neural ma-

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland.

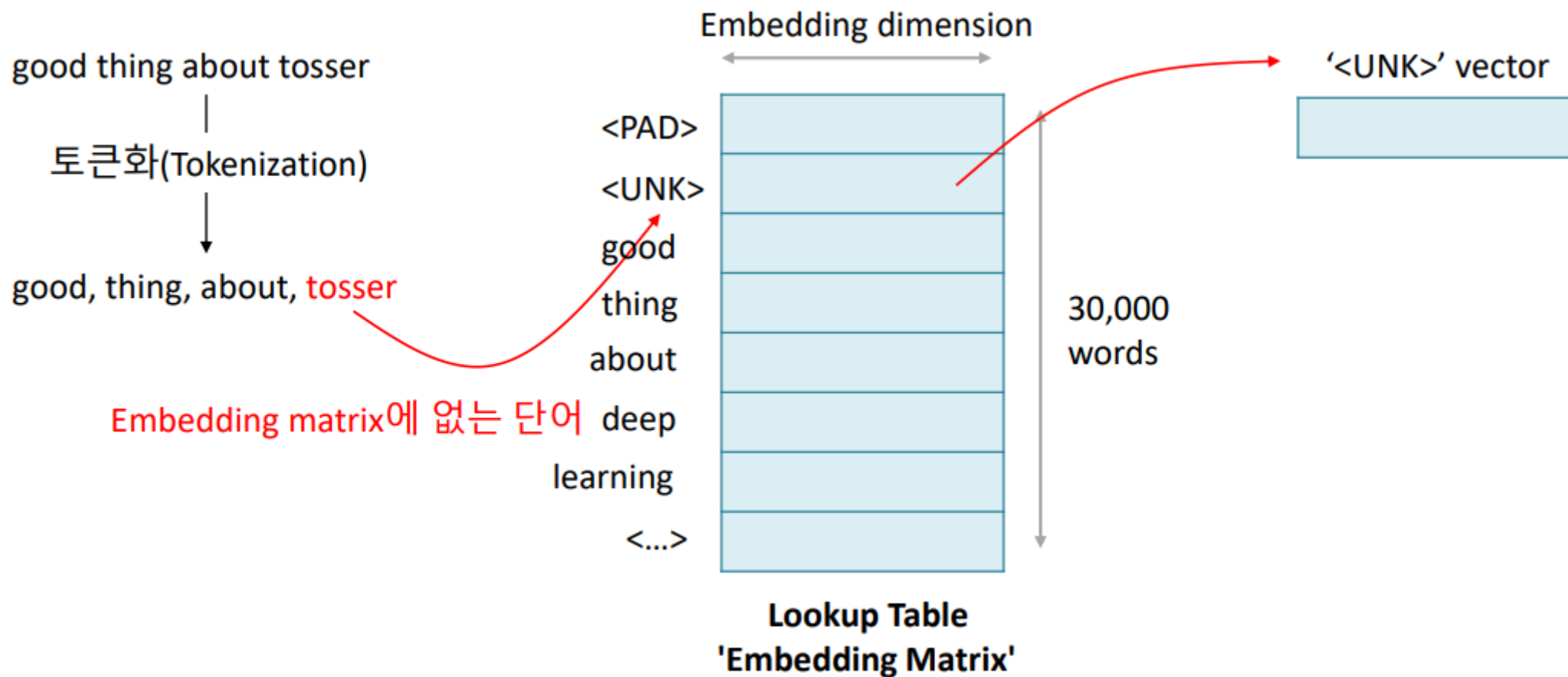
OOV(Out-Of-Vocabulary) 문제



OOV(Out-Of-Vocabulary) 문제



OOV(Out-Of-Vocabulary) 문제




Byte Pair Encoding 개요

- BPE 자체는 1994년에 제안된 데이터 압축 알고리즘. (NLP 알고리즘이 아님.)
- 자주 등장하는 Byte Pair는 새로운 하나의 Byte가 된다.
- 이를 단어 분리(Word segmentation)에 도입한다. (NLP 알고리즘이 됨.)
- Bottom-up 방식의 클러스터링
- 데이터의 모든 글자(character) 단위의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 바이그램을 유니그램으로 통합한다.
- 모든 바이그램이 선택되거나 정해진 단어 집합의 크기에 도달할 때까지 반복!

Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)

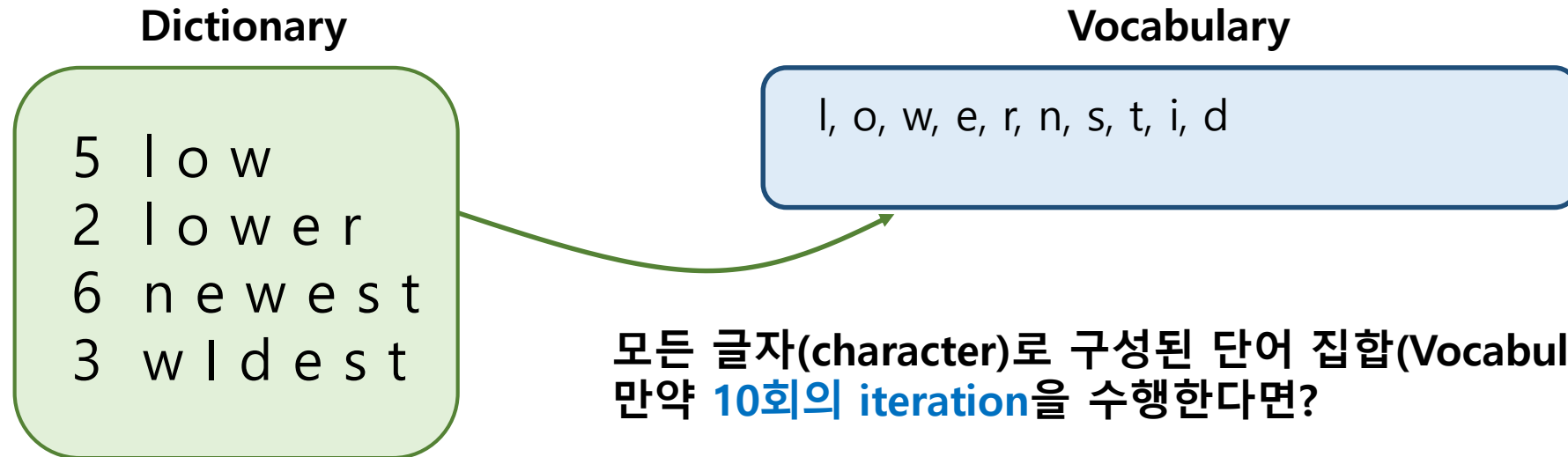
Dictionary



5 low
2 lower
6 newest
3 widest

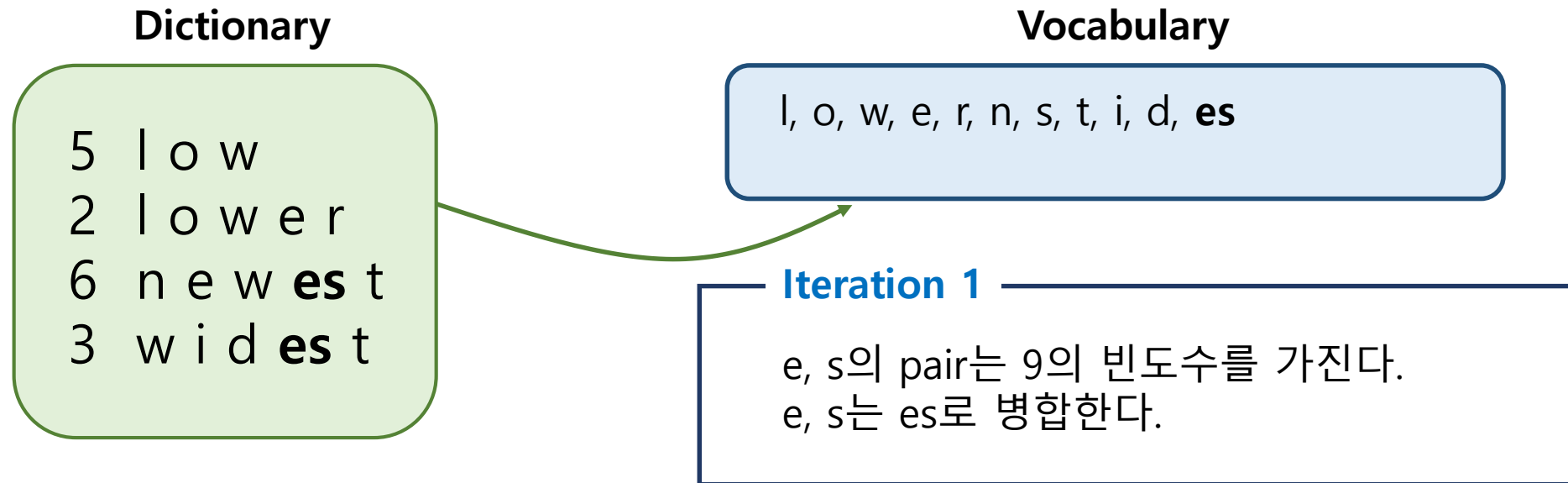
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



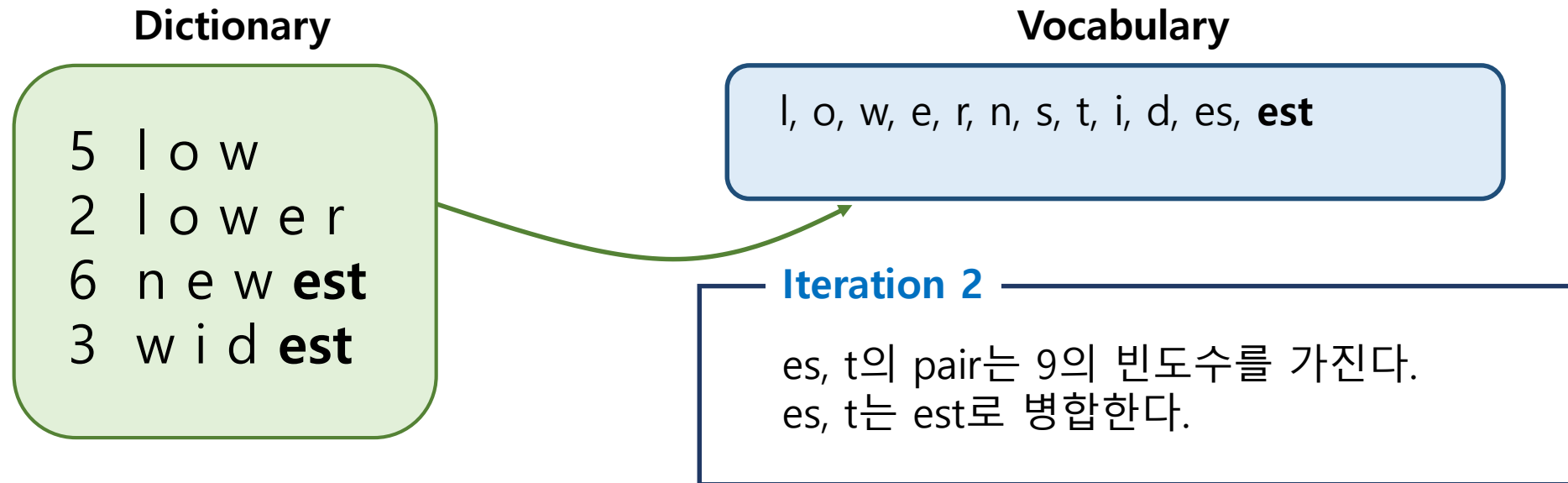
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



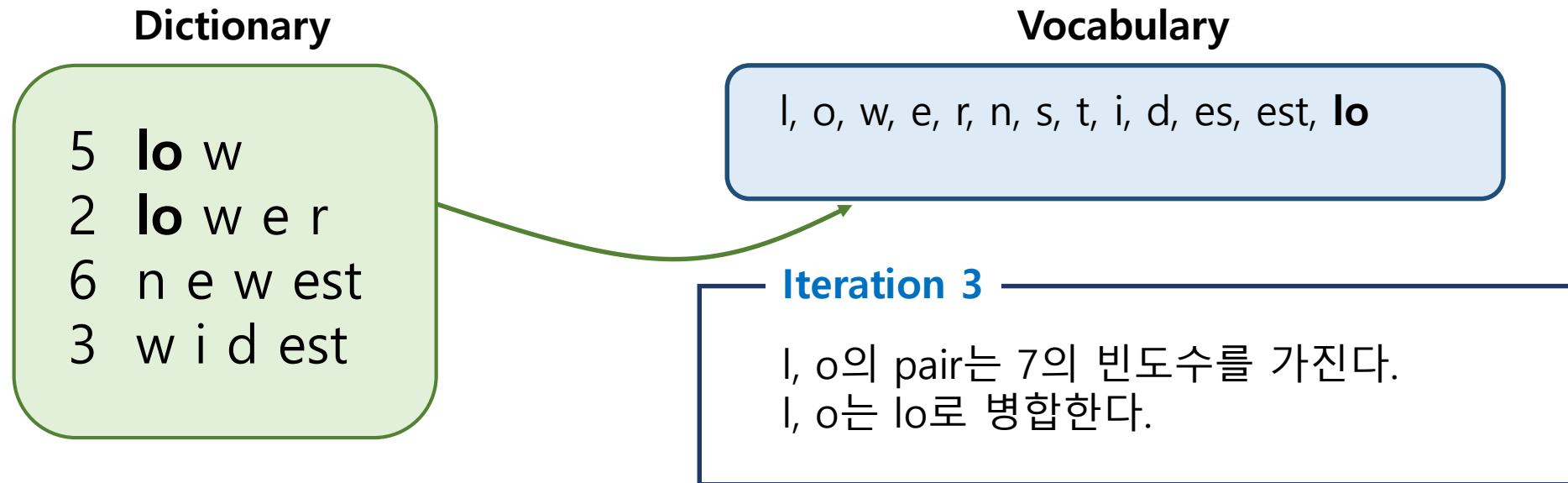
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



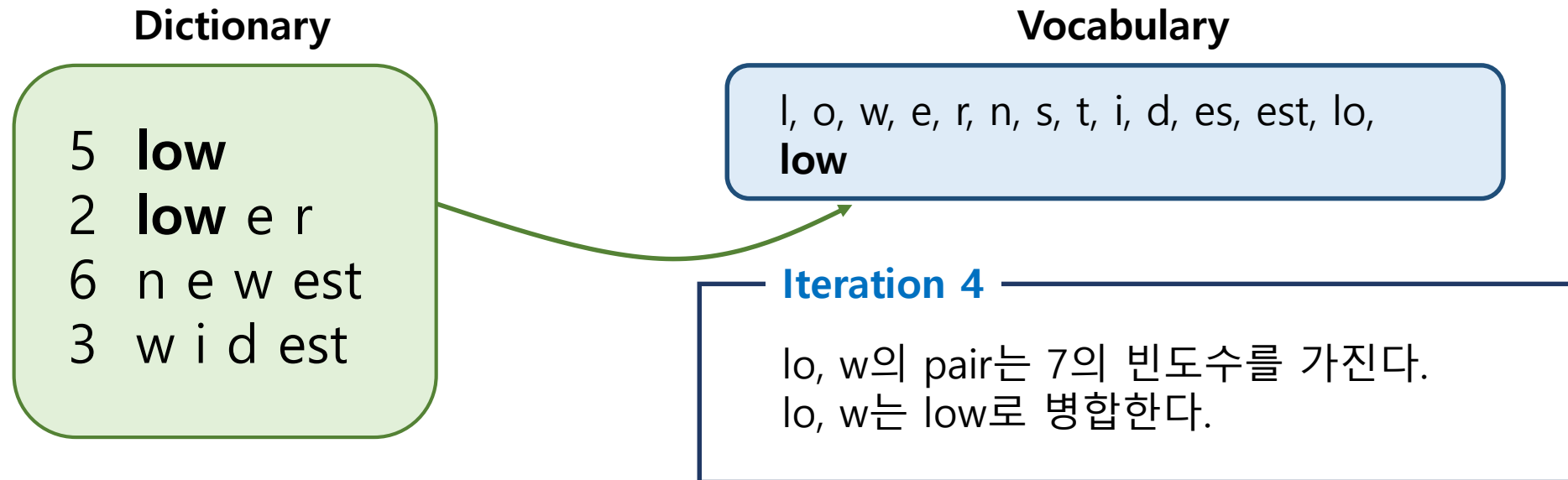
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



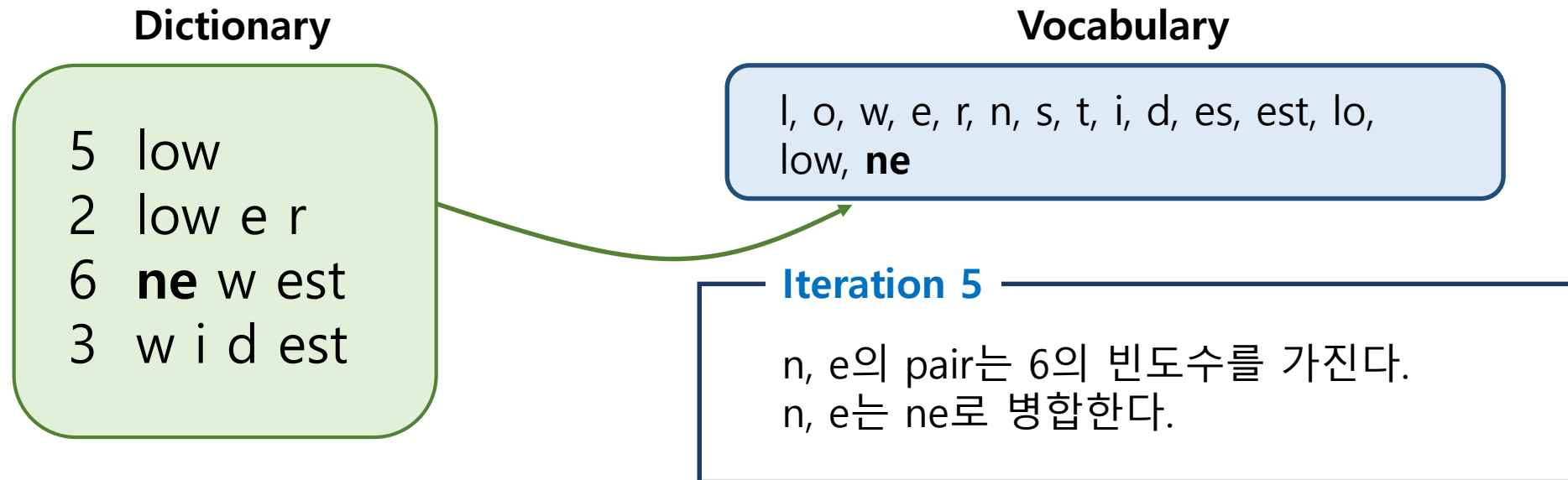
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



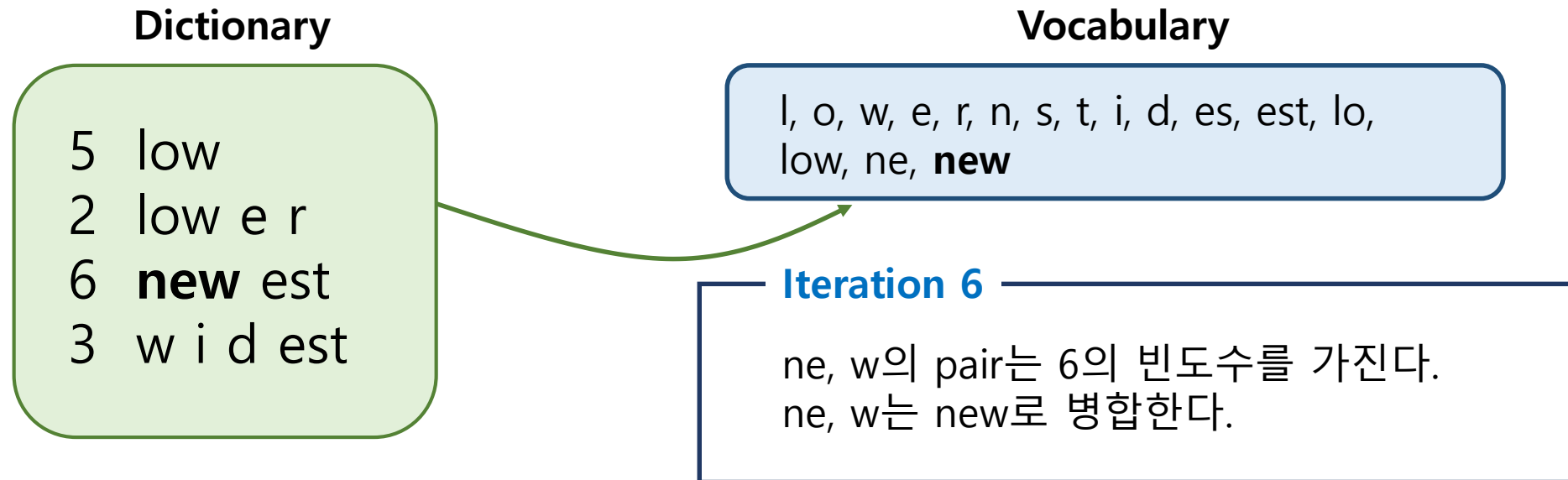
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



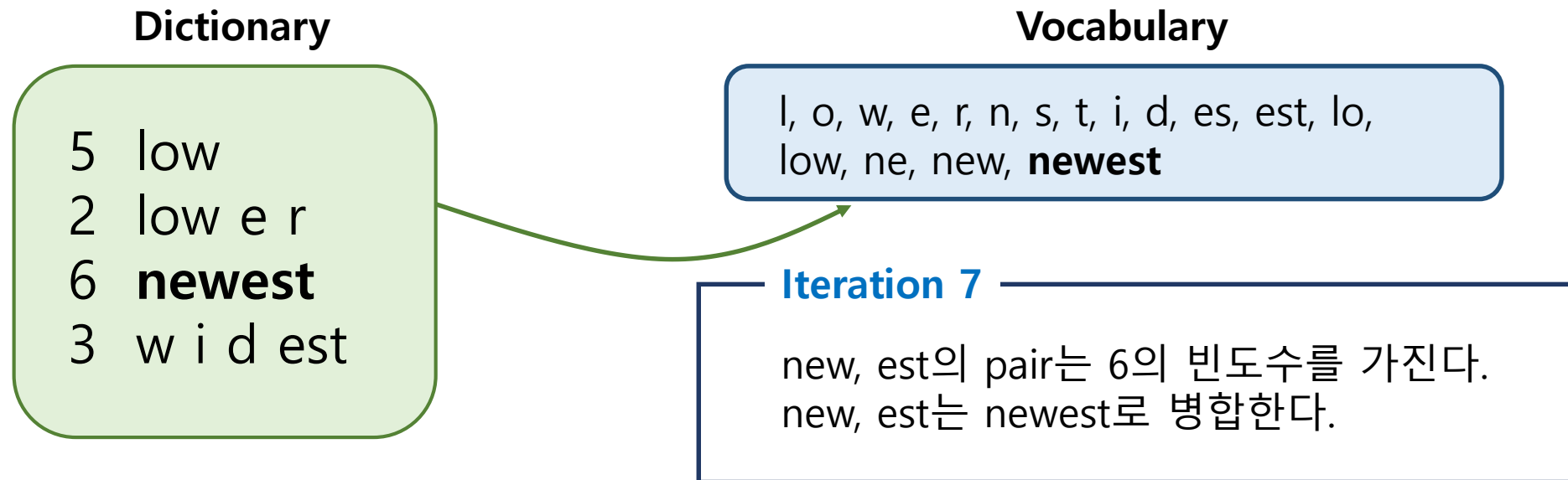
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



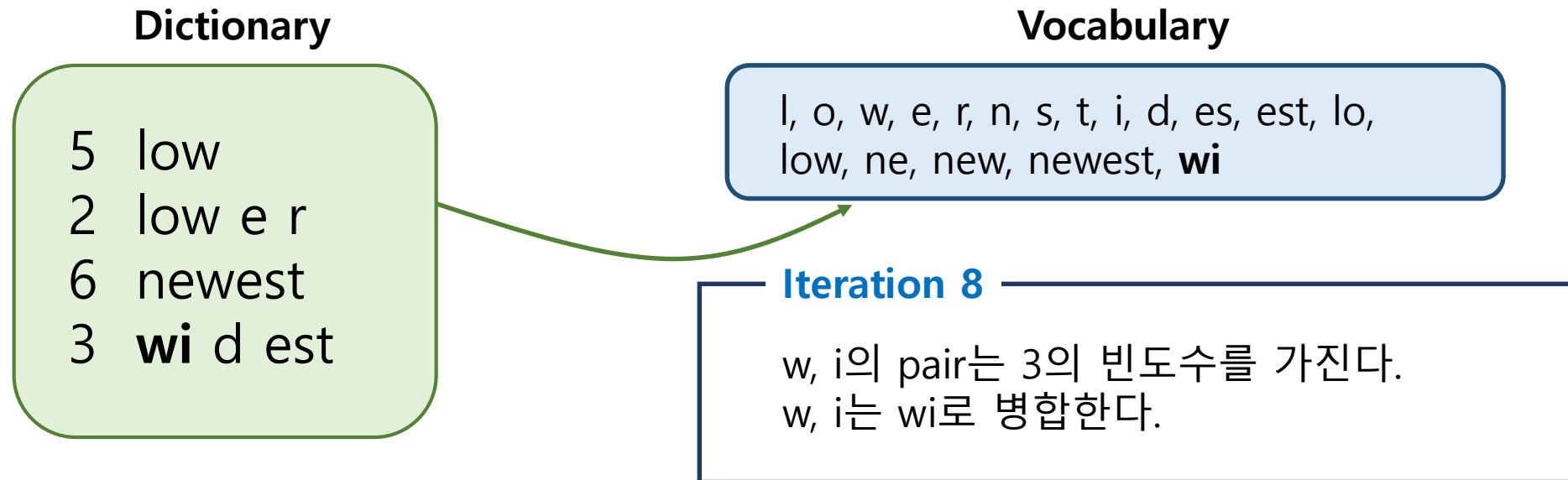
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



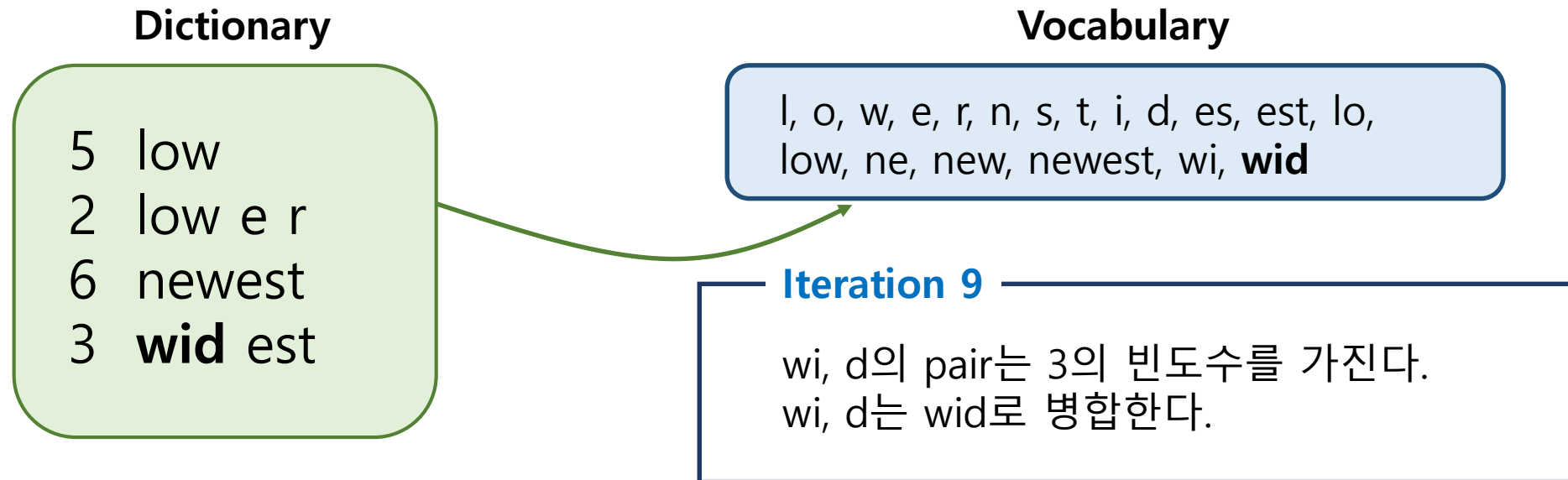
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



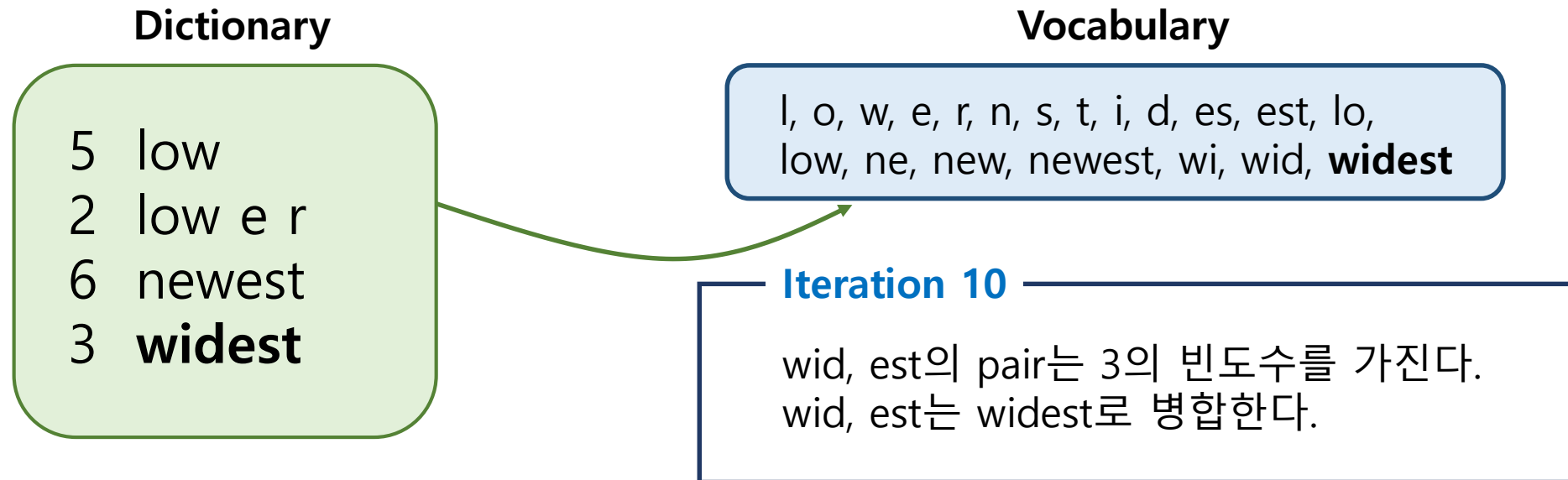
Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



Byte Pair Encoding 개요

- 데이터의 모든 글자(character)의 유니그램 단어 사전에서 시작한다.
- 자주 등장하는 Byte Pair는 새로운 Byte가 된다.
- (자주 등장하는 바이그램을 하나의 유니그램으로 병합한다.)



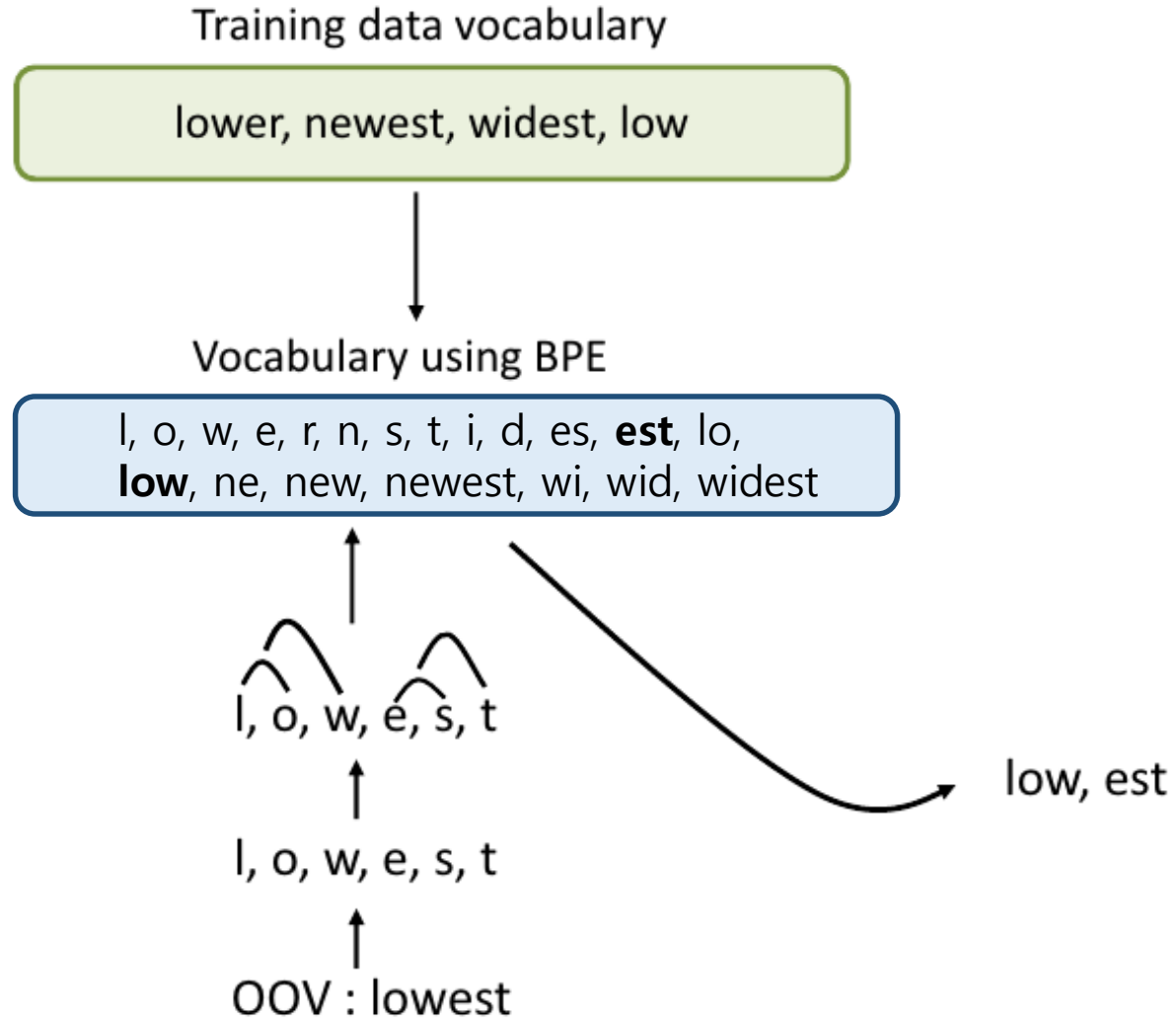
Byte Pair Encoding 개요

- iteration을 할수록 단어 집합의 크기가 커진다.
- 한국어에도 적용 가능!
- 자주 등장하는 바이그램은 그 자체가 단어로 취급된다.
- 희귀 단어, OOV(Out-Of-Vocabulary)에 강건해진다.

Vocabulary

l, o, w, e, r, n, s, t, i, d, es, est, lo,
low, ne, new, newest, wi, wid, widest

OOV가 들어왔을 때 BPE의 동작 과정



Byte Pair Encoding Implementation

```
[1] import re, collections

def get_stats(vocab):
    """Compute frequencies of adjacent pairs of symbols."""
    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

[2] from IPython.display import display, Markdown, Latex

train_data = {'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}

bpe_codes = {}
bpe_codes_reverse = {}

num_merges = 10

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(train_data)
    best = max(pairs, key=pairs.get)
    train_data = merge_vocab(best, train_data)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("train data: {}".format(train_data))
```

'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

Byte Pair Encoding Implementation

```
[1] import re, collections

def get_stats(vocab):
    """Compute frequencies of adjacent pairs of symbols."""
    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

[2] from IPython.display import display, Markdown, Latex

train_data = {'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}

bpe_codes = {}
bpe_codes_reverse = {}
num_merges = 10

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(train_data)
    best = max(pairs, key=pairs.get)
    train_data = merge_vocab(best, train_data)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("train data: {}".format(train_data))
```

chracter 단위로 분할

끝을 의미하는 특수기호

빈도수

'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

1 iteration: ('e', 's')
2 iteration: ('es', 't')
3 iteration: ('est', '</w>')
4 iteration: ('l', 'o')
5 iteration: ('lo', 'w')
6 iteration: ('n', 'e')
7 iteration: ('ne', 'w')
8 iteration: ('new', 'est</w>')
9 iteration: ('low', '</w>')
10 iteration: ('w', 'i')

'low</w>': 5, 'l o w e r </w>': 2,
'newest</w>': 6, 'w i d e s t </w>': 3

Byte Pair Encoding Implementation

```
[1] import re, collections

def get_stats(vocab):
    """Compute frequencies of adjacent pairs of symbols."""
    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'(?!\wS)' + bigram + r'(!\wS)')
    for word in v_in:
        w_out = p.sub(' '.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

[2] from IPython.display import display, Markdown, Latex

train_data = {'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}

bpe_codes = {}
bpe_codes_reverse = {}

num_merges = 10

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(train_data)
    best = max(pairs, key=pairs.get)
    train_data = merge_vocab(best, train_data)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("train data: {}".format(train_data))
```

'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



Subword Unit

es, est, est</w>, lo, low, ne, new,
newest</w>, low</w>, wi

Byte Pair Encoding Implementation

```
[1] import re, collections

def get_stats(vocab):
    """Compute frequencies of adjacent pairs of symbols."""
    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

[2] from IPython.display import display, Markdown, Latex

train_data = {'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}

bpe_codes = {}
bpe_codes_reverse = {}

num_merges = 10

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(train_data)
    best = max(pairs, key=pairs.get)
    train_data = merge_vocab(best, train_data)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("train data: {}".format(train_data))
```

'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



Subword Unit

es, est, est</w>, lo, low, ne, new,
newest</w>, low</w>, wi

참고) 위의 Subword Unit에서
단일 알파벳은 별도 표기하지 않았음.

Byte Pair Encoding Implementation

Training data vocabulary

lower, newest, widest, low

Subword Unit

es, est, est</w>, lo, low, ne, new,
newest</w>, low</w>, wi

Byte Pair Encoding Implementation

Training data vocabulary

lower, newest, widest, low

Subword Unit

es, est, est</w>, lo, low, ne, new,
newest</w>, low</w>, wi

OOV : 'lowest' ?

Byte Pair Encoding Implementation

```
[1] import re, collections

def get_stats(vocab):
    """Compute frequencies of adjacent pairs of symbols."""
    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'(?!\wS)' + bigram + r'(!\wS)')
    for word in v_in:
        w_out = p.sub(' '.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

[2] from IPython.display import display, Markdown, Latex

train_data = {'low </w>': 5, 'lower </w>': 2, 'newest </w>': 6, 'widest </w>': 3}

bpe_codes = {}
bpe_codes_reverse = {}

num_merges = 10

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(train_data)
    best = max(pairs, key=pairs.get)
    train_data = merge_vocab(best, train_data)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("train data: {}".format(train_data))
```

OOV : 'lowest'

`encode("lowest")`

l o w e s t </w>

1 iteration: ('l', 'o', 'w', 'es', 't', '<\w>')
2 iteration: ('l', 'o', 'w', 'est', '<\w>')
3 iteration: ('l', 'o', 'w', 'est<\w>')
4 iteration: ('lo', 'w', 'est<\w>')
5 iteration: ('low', 'est<\w>')
Result : ('low', 'est')

Subword Unit

es, est, est</w>,
lo, low, ne, new,
low</w>, wi

실습 : <https://colab.research.google.com/drive/1G9vRvOThc5We0ji-x-aNU4CoeOVu3fV-?usp=sharing>

Byte Pair Encoding Implementation

```
[1] import re, collections

def get_stats(vocab):
    """Compute frequencies of adjacent pairs of symbols."""
    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'(?!\wS)' + bigram + r'(!\wS)')
    for word in v_in:
        w_out = p.sub(' '.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

[2] from IPython.display import display, Markdown, Latex

train_data = {'low </w>': 5, 'lower </w>': 2, 'newest </w>': 6, 'widest </w>': 3}

bpe_codes = {}
bpe_codes_reverse = {}

num_merges = 10

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(train_data)
    best = max(pairs, key=pairs.get)
    train_data = merge_vocab(best, train_data)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("train data: {}".format(train_data))
```

OOV : 'lowest'

`encode("lowest")`

l o w e s t </w>

1 iteration: ('l', 'o', 'w', 'es', 't', '<\w>')
2 iteration: ('l', 'o', 'w', 'est', '<\w>')
3 iteration: ('l', 'o', 'w', 'est<\w>')
4 iteration: ('lo', 'w', 'est<\w>')
5 iteration: ('low', 'est<\w>')
Result : ('low', 'est')

Subword Unit

es, est, est</w>,
lo, low, ne, new,
newest</w>,
low</w>, wi

실습을 통해서 이해해봅시다.

실습 : <https://colab.research.google.com/drive/1G9vRvOThc5We0ji-x-aNU4CoeOVu3fV-?usp=sharing>

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 1. Replace the most frequent pair of characters with a new unit.
(Record this “merge” operation.)
 2. Repeat until the desired number of merge operations is reached.

Current vocabulary

The new merge

lower lowest newer widest

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 1. Replace the most frequent pair of characters with a new unit.
(Record this “merge” operation.)
 2. Repeat until the desired number of merge operations is reached.

Current vocabulary

low**er** low**est** new**er** widest

The new merge

we → we

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 1. Replace the most frequent pair of characters with a new unit.
(Record this “merge” operation.)
 2. Repeat until the desired number of merge operations is reached.

Current vocabulary

lower lowest newer widest

lower lowest newer widest

The new merge

we → we

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 1. Replace the most frequent pair of characters with a `new unit`.
(Record this “merge” operation.)
 2. Repeat until the desired number of merge operations is reached.

Current vocabulary	The new merge
low er low est new er widest	we → <code>we</code>
lo we r lo we st ne we r widest	<code>we</code> r → <code>wer</code>

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 - Replace the most frequent pair of characters with a `[new unit]`.
(Record this “merge” operation.)
 - Repeat until the desired number of merge operations is reached.

Current vocabulary

low**er** low**est** new**er** widest

lo**[we]****r** lo**[we]**st ne**[we]****r** widest

lo**[we]****r** lo**[we]**st ne**[we]****r** widest

The new merge

we → `[we]`

`[we]`r → `[we]``r`

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 1. Replace the most frequent pair of characters with a `[new unit]`.
(Record this “merge” operation.)
 2. Repeat until the desired number of merge operations is reached.

Current vocabulary	The new merge
low er low est new er widest	we → [we]
lo[we] r lo[we]st ne[we] r widest	[we]r → [we r]
lo[we r] lo[we] st ne[we r] widest st	st → [st]

Byte Pair Encoding Example

- Given a dictionary of token types and frequencies.
 - Replace the most frequent pair of characters with a `[new unit]`. (Record this “merge” operation.)
 - Repeat until the desired number of merge operations is reached.

Current vocabulary	The new merge
low er low est new er widest	we → [we]
lo[we] r lo[we]st ne[we] r widest	[we]r → [we]r
lo[we]r lo[we] st ne[we]r widest	st → [st]

- New input: Apply the **recorded sequence** of merges:
newest → ne[we]st → ne[we][st] ⇒ n@@ e@@ we@@ st
- Ensures that vocabulary size = alphabet + merge ops.

여러가지 변형 구현체들

- **Byte Pair Encoding**
 - : <https://github.com/rsennrich/subword-nmt/>
- **Wordpiece Model (NMT, BERT)**
 - : BPE의 변형 알고리즘
 - : 구글에서 번역기와 BERT 훈련 시 사용. 내부적으로 사용되며 코드는 미공개.
- **Unigram Language Model Tokenizer**
 - : BPE의 변형 알고리즘
- **SubwordTextEncoder in Tensor2tensor**
 - : Tensorflow 2.0에서 제공. Wordpiece를 통해 단어를 분리.
- **SentencePiece (별도 설치 필요)**
 - : Best! 가장 많이 사용되는 패키지
 - : <https://github.com/google/sentencepiece>

여러가지 변형 구현체들

→ 알고리즘

- **Byte Pair Encoding**

- : <https://github.com/rsennrich/subword-nmt/>

- **Wordpiece Model (NMT, BERT)**

- : BPE의 변형 알고리즘

- : 구글에서 번역기와 BERT 훈련 시 사용. 내부적으로 사용되며 코드는 미공개.

- **Unigram Language Model Tokenizer**

- : BPE의 변형 알고리즘

- **SubwordTextEncoder in Tensor2tensor**

- : Tensorflow 2.0에서 제공. Wordpiece를 통해 단어를 분리.

- **SentencePiece (별도 설치 필요)**

- : Best! 가장 많이 사용되는 패키지. **BPE**와 **Unigram**을 구현하여 제공.

- : <https://github.com/google/sentencepiece>

→ 실제 사용 가능한 패키지. (위키독스 내 실습 챕터 존재)

여러가지 변형 구현체들

- 그 외 **SetencePiece** 대신 사용할 수 있는 다른 대안
 - : Huggingface의 서브워드 토큰나이저
 - : <https://keep-steady.tistory.com/37>
 - : <https://huffon.github.io/2020/07/05/tokenizers/>

한국어에서는? SentencePiece Vs. Mecab

- 채팅 데이터에 대해서 실험
- 채팅 데이터는 띄어쓰기 오류, 오타, 신조어가 많아서 OOV가 많이 발생.
- 또한 실시간 채팅이라함은 처리 속도가 매우 민감!!
- 단어 집합의 크기를 각각 30000, 50000, 100000로 제한하여 비교.

Tokenizer	Out-of-Vocabulary	Tokenizing Speed (sample/sec)
SentencePiece-30000	0.12 %	132632.42
SentencePiece-50000	0.13 %	131353.65
SentencePiece-100000	0.14 %	129670.69
Mecab-30000	0.62 %	47035.73
Mecab-50000	0.20 %	46795.35
Mecab-100000	0.04 %	46085.86
Kharii	-	4444.43

한국어에서는? SentencePiece Vs. Mecab

- 1 ~ 4번 문장은 띄어쓰기가 되어있지 않은 문장.
- Mecab의 경우 형태소 분석을 통해 형태소 단위를 잘 잡아낸다.
- SentencePiece는 Mecab에 비해 상대적으로 잘못된 Tokenize를 수행.
- Mecab은 피스타치오를 (사전 등록) 잡아내지만 SentencePiece는 전부 분리.

#	Origin Sentence	SentencePiece Tokenizer	Mecab Tokenizer
1	엄청 빨리끝나는거같네	엄청 V 빨리 끝나 V 는 거 같네	엄청 V 빨리 V 끝나 V 는 V 거 V 같 V 네
2	차량운행할때 불편하겠다 조심해요	차량 V 운 V 행 V 할 때 V 불편하겠다 V 조심해요	차량 V 운행 V 할 V 때 V 불편 V 하 V 겠 V 다 V 조 심 V 해요
3	출석부르기전에 들어가면 되지않을까	출석 V 부르 V 기 전에 V 들어가면 V 되 지않을까	출석 V 부르 V 기 V 전 V 에 V 들어가 V 면 V 되 V 지 V 않 V 을까
4	잘못될까무섭고	잘못 V 될 V 까 V 무 V 겁 V 고	잘못 V 될까 V 무섭 V 고
5	피스타치오향 약간 새로운맛이야	피 V 스타 V 치 V 오 V 향 V 약간 V 새로운 V 맛이야	피스타치오 V 향 V 약간 V 새로운 V 맛 V 이 V 야
6	후리스따뜻해?	후리스 V 따뜻해 V ?	후 V 리스 V 따뜻 V 해 V ?
7	응 당직은 칼퇴	응 V 당직 V 은 V 칼퇴	응 V 당직 V 은 V 칼 V 퇴
8	자꾸자꾸 심쿵하네	자꾸자꾸 V 심쿵 V 하네	자꾸 V 자꾸 V 심 V 쿵 V 하 V 네
9	친구들이랑 가죠 뭐	친구들이랑 V 가죠 V 뭐	친 V 구 V 들 V 이랑 V 가 V 죠 V 뭐

한국어에서는? SentencePiece Vs. Mecab

- SentencePiece:

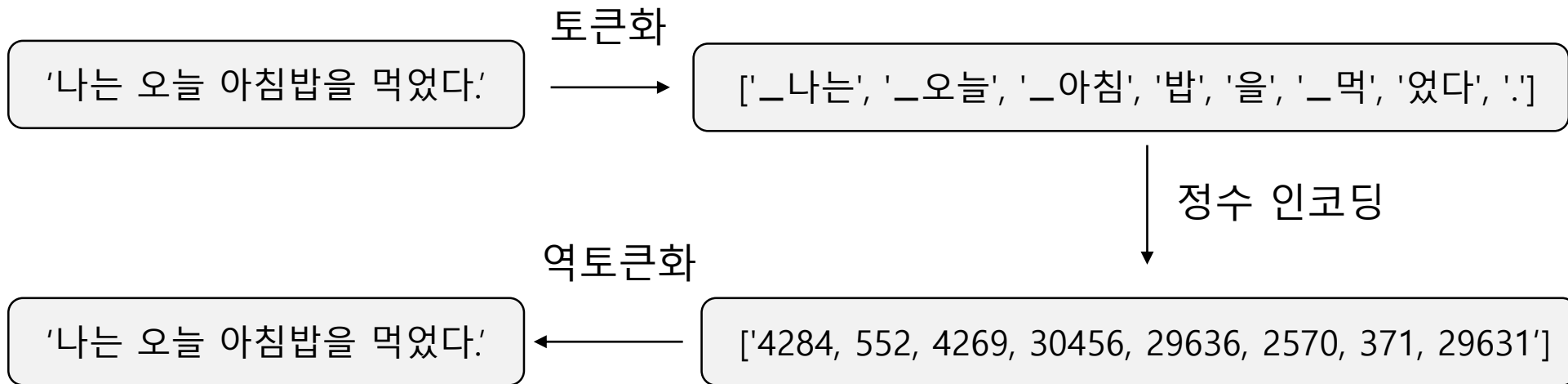
- 아주 빠름
- Subword의 빈도에 기반하기에 unknown token 처리 가능
- 의미 단위로 잘 나뉘지지 않을 수 있음

- Mecab:

- SentencePiece만큼은 아니지만 충분히 빠름
- 형태소 분석을 기반으로 하기 때문에 의미 단위를 잘 포착
- 신조어나 오타에 취약

SentencePiece의 Detokenization

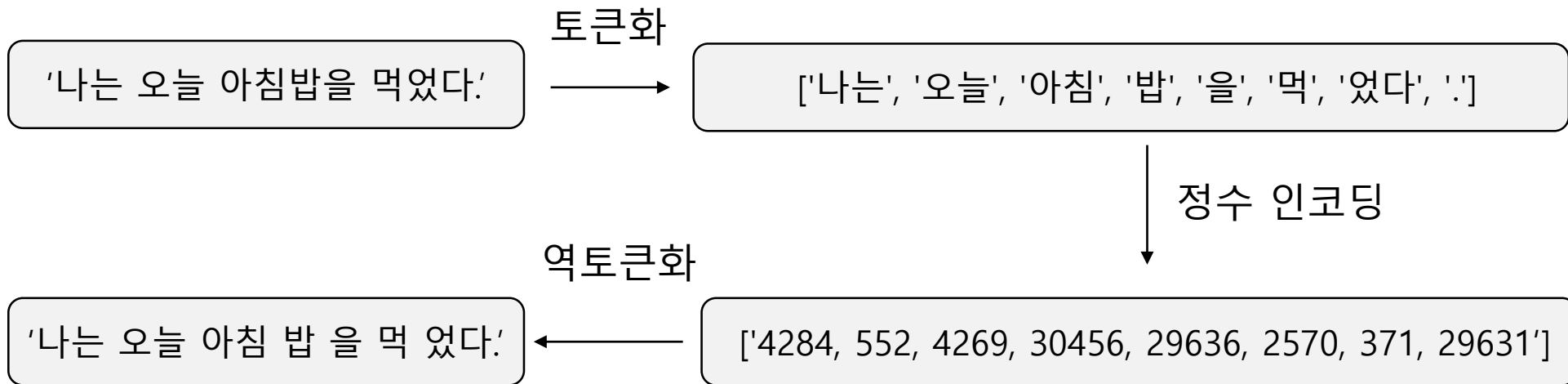
- 서브워드 토크나이저의 경우, 분리 시에 서브워드로 분리하므로 이게 기존의 단어인지 서브워드 인지를 표시하면서 분리하는데, 단어의 시작에 _를 붙이는 것이다.
- 서브워드로 분리된 것을 다시 원문으로 복원하는 것을 Detokenization이라 한다. 단어의 시작에 _를 붙였으므로 이를 참고하여 기존의 원문으로 자연스럽게 다시 복원한다.



SentencePiece의 Detokenization

- 서브워드 토크나이저의 경우, 분리 시에 서브워드로 분리하므로 이게 기존의 단어인지 서브워드 인지를 표시하면서 분리하는데, 단어의 시작에 _를 붙이는 것이다.
- 서브워드로 분리된 것을 다시 원문으로 복원하는 것을 Detokenization이라 한다. 단어의 시작에 _를 붙였으므로 이를 참고하여 기존의 원문으로 자연스럽게 다시 복원한다.

만약 표시해주지 않았다면?



BLEU

BLEU score

- 한 개의 문장에도 다양한 번역이 나올 수 있다.
- 번역의 품질을 어떻게 평가할 수 있을까?

트와이스는 데뷔 첫 빌보드 메인차트 진입에 성공했다.

TWICE succeeded in entering the first billboard main chart.

TWICE succeeded in entering the billboard main chart for the first time.

After making their debut, Twice succeeded in entering the billboard main chart for the first time.

TWICE also succeeded in entering the Billboard main chart for the first time since its debut.

BLEU score

- 기계 번역의 성능 측정을 위한 대표적인 방법인 BLEU(Bilingual Evaluation Understudy)
- 참고해야할 논문 : BLEU: a Method for Automatic Evaluation of Machine Translation
- 언어에 구애받지 않고 사용할 수 있으며, 계산 속도가 빠르다!


BLEU score

- 기계 번역의 성능 측정을 위한 대표적인 방법인 BLEU(Bilingual Evaluation Understudy)
- 참고해야할 논문 : BLEU: a Method for Automatic Evaluation of Machine Translation
- 언어에 구애받지 않고 사용할 수 있으며, 계산 속도가 빠르다!

$$BLEU = BP \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$


BLEU score

- 기계 번역의 성능 측정을 위한 대표적인 방법인 BLEU(Bilingual Evaluation Understudy)
- 참고해야할 논문 : BLEU: a Method for Automatic Evaluation of Machine Translation
- 언어에 구애받지 않고 사용할 수 있으며, 계산 속도가 빠르다!

$$BLEU = \underline{BP} \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$
$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$


BLEU score

- 기계 번역의 성능 측정을 위한 대표적인 방법인 BLEU(Bilingual Evaluation Understudy)
- 참고해야할 논문 : BLEU: a Method for Automatic Evaluation of Machine Translation
- 언어에 구애받지 않고 사용할 수 있으며, 계산 속도가 빠르다!

$$BLEU = \underline{BP} \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$
$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$


이 식을 차근, 차근 도출해보자!

What is Unigram Precision?

- 두 개의 기계 번역기가 존재한다고 하고, 번역된 문장을 각각 Candidate1, 2라고 해보자.
- 세 명의 사람에게 영작을 시켜서 세 개의 번역 문장을 만들었다고 해보자. 이를 Reference1, 2, 3이라고 하자.

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

What is Unigram Precision?

- 두 개의 기계 번역기가 존재한다고 하고, 번역된 문장을 각각 Candidate1, 2라고 해보자.
- 세 명의 사람에게 영작을 시켜서 세 개의 번역 문장을 만들었다고 해보자. 이를 Reference1, 2, 3이라고 하자.

번역기가 번역한 문장

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

사람이 번역한 문장

Unigram Precision

- 두 개의 기계 번역기가 존재한다고 하고, 번역된 문장을 각각 Candidate1, 2라고 해보자.
- 세 명의 사람에게 영작을 시켜서 세 개의 번역 문장을 만들었다고 해보자. 이를 Reference1, 2, 3이라고 하자.

편의상 앞으로 Candidate를 Ca, Reference를 Ref로 명명함.

번역기가 번역한 문장 앞으로 Ca들의 정확도를 비교할 예정!

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

사람이 번역한 문장 (실제 정답)

Unigram Precision

- Ref 1, 2, 3 중 어느 한 문장에서라도 등장한 Ca의 단어의 개수를 카운트하여 분모.
 - Ca의 모든 단어의 카운트의 합. 즉, Ca에서의 총 단어의 수를 카운트하여 분자.
- 이를 **유니그램 정밀도(Unigram Precision)**라고 한다.

Unigram Precision =

Unigram Precision

- Ref 1, 2, 3 중 어느 한 문장에서라도 등장한 Ca의 단어의 개수를 카운트하여 분모.
 - Ca의 모든 단어의 카운트의 합. 즉, Ca에서의 총 단어의 수를 카운트하여 분자.
- 이를 **유니그램 정밀도(Unigram Precision)**라고 한다.

$$\text{Unigram Precision} = \frac{\text{the number of Ca words(unigrams) which occur in any Ref}}{\text{the total number of words in the Ca}}$$

Unigram Precision

- Ref 1, 2, 3 중 어느 한 문장에서라도 등장한 Ca의 단어의 개수를 카운트하여 분모.
 - Ca의 모든 단어의 카운트의 합. 즉, Ca에서의 총 단어의 수를 카운트하여 분자.
- 이를 **유니그램 정밀도(Unigram Precision)**라고 한다.

Unigram Precision =

the number of Ca words(unigrams) which occur in any Ref
the total number of words in the Ca

$$= \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}}$$

Unigram Precision

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}}$$

$$\text{Ca1 Unigram Precision} = \frac{17}{18}$$

$$\text{Ca2 Unigram Precision} = \frac{8}{14}$$

Unigram Precision

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}}$$

$$\text{Ca1 Unigram Precision} = \frac{17}{18}$$



$$\text{Ca2 Unigram Precision} = \frac{8}{14}$$

Ca1에 있는 단어들은 Ref1, Ref2, Ref3에 전반적으로 등장하지만
Ca2에 있는 단어들은 그렇지 않다. Ca1의 승리!

Unigram Precision

Ref1, 2, 3 어디에도 등장하지 않는다.

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}}$$

$$\text{Ca1 Unigram Precision} = \frac{17}{18}$$



$$\text{Ca2 Unigram Precision} = \frac{8}{14}$$

Ca1에 있는 단어들은 Ref1, Ref2, Ref3에 전반적으로 등장하지만
Ca2에 있는 단어들은 그렇지 않다. Ca1의 승리!

Unigram Precision

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}}$$

$$\text{Ca1 Unigram Precision} = \frac{17}{18}$$



$$\text{Ca2 Unigram Precision} = \frac{8}{14}$$

다른 예제를 보자!

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}} = ?$$

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

한 번 직접 계산해보세요.

Reference2 : there is a cat on the mat

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}} = ?$$

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}} = \frac{7}{7} = 1$$

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

말도 안 되는 번역인데...
정밀도는 최고 성능?

이를 어떻게 해결할 수 있을까?

$$\text{Unigram Precision} = \frac{\text{Ref들 중에서 존재하는 Ca의 단어의 수}}{\text{Ca의 총 단어 수}} = \frac{7}{7} = 1$$

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

말도 안 되는 번역인데...
정밀도는 최고 성능?

Reference2 : there is a cat on the mat

이를 어떻게 해결할 수 있을까?
중복을 제거하여 보정하자!

$$\text{Unigram Precision} = \frac{\text{Ref들과 Ca를 고려한 새로운 카운트 방법이 필요!}}{\text{Ca의 총 유니그램 수}}$$

Unigram Precision

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

말도 안 되는 번역인데...
정밀도는 최고 성능?

Reference2 : there is a cat on the mat

이를 어떻게 해결할 수 있을까?
중복을 제거하여 보정하자!

$$\text{Unigram Precision} = \frac{\text{Ref들과 Ca를 고려한 새로운 카운트 방법이 필요!}}{\text{Ca의 총 유니그램 수}}$$

~~Unigram Precision~~

Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

말도 안 되는 번역인데...
정밀도는 최고 성능?

Reference2 : there is a cat on the mat

이를 어떻게 해결할 수 있을까?
중복을 제거하여 보정하자!

$$\text{Unigram Precision} = \frac{\text{Ref들과 Ca를 고려한 새로운 카운트 방법이 필요!}}{\text{Ca의 총 유니그램 수}}$$

~~Unigram Precision~~

Modified Unigram Precision!

Modified Unigram Precision

- 유니그램이 하나의 Ref에서 최대 몇 번 등장했는지를 카운트 : Max_Ref_Count

$$Count_{clip} = \min(Count, Max_Ref_Count)$$

$$\text{Modified Unigram Precision} = \frac{\sum_{unigram \in Candidate} Count_{clip}(unigram)}{\sum_{unigram \in Candidate} Count(unigram)}$$

Modified Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

Modified Unigram Precision = ?

Modified Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

$$\text{Modified Unigram Precision} = \frac{2}{7}$$

Modified Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

$$\text{Modified Unigram Precision} = \frac{2}{7}$$

단순 Count값은 7이지만, Max_Ref_Count는 2이므로
단어 the에 대한 최종 카운트인 $\text{Count_clip} = \min(7, 2) = 2$

Modified Unigram Precision

Candidate : the the the the the the the

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

중복 단어에 대한 문제는 해결!
이제 문제가 없을까? 다른 예제를 보자!

$$\text{Modified Unigram Precision} = \frac{2}{7}$$

단순 Count값은 7이지만, Max_Ref_Count는 2이므로
단어 the에 대한 최종 카운트인 $\text{Count_clip} = \min(7, 2) = 2$

Modified Unigram Precision이 여전히 가지는 한계

- 앞서 본 예제에서 Ca1의 모든 단어의 순서를 랜덤으로 뒤바꾼 Ca3을 추가.

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Candidate3 : the that military a is It guide ensures which to commands the of action obeys always party the.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

Modified Unigram Precision이 여전히 가지는 한계

- 앞서 본 예제에서 Ca1의 모든 단어의 순서를 랜덤으로 뒤바꾼 Ca3을 추가.

Modifed Unigram Precision으로 하면 누가 더 성능이 좋다고 나올까요?

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Candidate3 : the that military a is It guide ensures which to commands the of action obeys always party the.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

Modified Unigram Precision이 여전히 가지는 한계

- 앞서 본 예제에서 Ca1의 모든 단어의 순서를 랜덤으로 뒤바꾼 Ca3을 추가.
- Ca1과 Ca3의 Unigram Precision을 계산하면 두 값은 동일.
- Unigram Precision은 단어의 순서를 고려하지 않는다!

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Candidate3 : the that military a is It guide ensures which to commands the of action obeys always party the.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

N-gram Precision (N=2)

- 이제는 Unigram이 아니라 N-gram으로 Precision을 구한다.
- 이제는 N=1인 Unigram이 아니라 N의 개수는 1보다 클 수 있다.
- Ca2에 대해서 N=2인 Bigram의 Count와 Count_clip을 구한 표는 아래와 같다.

Candidate1 : the the the the the the the

Candidate2 : the cat the cat on the mat

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

바이그램	the cat	cat the	cat on	on the	the mat	SUM
<i>Count</i>	2	1	1	1	1	6
<i>Count_{clip}</i>	1	0	1	1	1	4

N-gram Precision (N=2)

- 이제는 Unigram이 아니라 N-gram으로 Precision을 구한다.
- 이제는 N=1인 Unigram이 아니라 N의 개수는 1보다 클 수 있다.
- Ca2에 대해서 N=2인 Bigram의 Count와 Count_clip을 구한 표는 아래와 같다.

Candidate1 : the the the the the the the

Candidate2 : the cat the cat on the mat

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

바이그램	the cat	cat the	cat on	on the	the mat	SUM
$Count$	2	1	1	1	1	6
$Count_{clip}$	1	0	1	1	1	4

Ca1의 Modified Bigram Precision : ?

Ca20 Modified Bigram Precision : ?

N-gram Precision (N=2)

- 이제는 Unigram이 아니라 N-gram으로 Precision을 구한다.
- 이제는 N=1인 Unigram이 아니라 N의 개수는 1보다 클 수 있다.
- Ca2에 대해서 N=2인 Bigram의 Count와 Count_clip을 구한 표는 아래와 같다.

Candidate1 : the the the the the the the

Candidate2 : the cat the cat on the mat

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

바이그램	the cat	cat the	cat on	on the	the mat	SUM
$Count$	2	1	1	1	1	6
$Count_{clip}$	1	0	1	1	1	4

Ca1의 Modified Bigram Precision : ?

Ca2의 Modified Bigram Precision : $\frac{4}{6}$

N-gram Precision (N=2)

- 이제는 Unigram이 아니라 N-gram으로 Precision을 구한다.
- 이제는 $N=1$ 인 Unigram이 아니라 N 의 개수는 1보다 클 수 있다.
- Ca2에 대해서 $N=2$ 인 Bigram의 Count와 Count_clip을 구한 표는 아래와 같다.

Candidate1 : the the the the the the the

Candidate2 : the cat the cat on the mat

Reference1 : the cat is on the mat

Reference2 : there is a cat on the mat

바이그램	the cat	cat the	cat on	on the	the mat	SUM
$Count$	2	1	1	1	1	6
$Count_{clip}$	1	0	1	1	1	4

Ca1의 Modified Bigram Precision : 0

Ca2의 Modified Bigram Precision : $\frac{4}{6}$

N-gram Precision (N=2)

나중에 따로 풀어보세요!

Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference3 : It is the practical guide for the army always to heed the directions of the party.

Ca1의 Modified Bigram Precision : $\frac{10}{17}$

Ca2의 Modified Bigram Precision : $\frac{1}{13}$

N-gram Precision

- BLEU는 보통 Modified 1-Gram, 2-Gram, 3-Gram, 4-Gram을 사용한다.
- 이들을 각각 p_1, p_2, p_3, p_4 라고 하였을 때 각각에 대한 가중치를 달리하여 합산한 뒤에 BLEU를 계산한다.

$$BLEU = \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$

p_n : 각 gram의 보정된 정밀도입니다.

N : n-gram에서 n 의 최대 숫자입니다. 보통은 4의 값을 가집니다. N 이 4라는 것은 p_1, p_2, p_3, p_4 를 사용한다는 것을 의미합니다.

w_n : 각 gram의 보정된 정밀도에 서로 다른 가중치를 줄 수 있습니다. 이 가중치의 합은 1로 합니다. 예를 들어 N 이 4라고 하였을 때, p_1, p_2, p_3, p_4 에 대해서 동일한 가중치를 주고자한다면 모두 0.25를 적용할 수 있습니다.

N-gram Precision

- BLEU는 보통 Modified 1-Gram, 2-Gram, 3-Gram, 4-Gram을 사용한다.
- 이들을 각각 p_1, p_2, p_3, p_4 라고 하였을 때 각각에 대한 가중치를 달리하여 합산한 뒤에 BLEU를 계산한다.

$\overbrace{\quad}^N$
마지막으로 한 가지 보정이 더 들어갑니다!

p_n : 각 gram의 보정된 정밀도입니다.

N : n-gram에서 n 의 최대 숫자입니다. 보통은 4의 값을 가집니다. N 이 4라는 것은 p_1, p_2, p_3, p_4 를 사용한다는 것을 의미합니다.

w_n : 각 gram의 보정된 정밀도에 서로 다른 가중치를 줄 수 있습니다. 이 가중치의 합은 1로 합니다. 예를 들어 N 이 4라고 하였을 때, p_1, p_2, p_3, p_4 에 대해서 동일한 가중치를 주고자한다면 모두 0.25를 적용할 수 있습니다.


BLEU score

- 기계 번역의 성능 측정을 위한 대표적인 방법인 BLEU(Bilingual Evaluation Understudy)
- 참고해야할 논문 : BLEU: a Method for Automatic Evaluation of Machine Translation
- 언어에 구애받지 않고 사용할 수 있으며, 계산 속도가 빠르다!

$$BLEU = BP \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$

BLEU score

- 기계 번역의 성능 측정을 위한 대표적인 방법인 BLEU(Bilingual Evaluation Understudy)
- 참고해야할 논문 : BLEU: a Method for Automatic Evaluation of Machine Translation
- 언어에 구애받지 않고 사용할 수 있으며, 계산 속도가 빠르다!

$$BLEU = \underline{BP} \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$
$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$


짧은 문장 길이에 대한 패널티(Brevity Penalty)

- Ca의 길이에 BLEU의 점수가 과한 영향을 받을 수 있다.
- 이전의 예제에 다음과 같은 Candidate 3을 추가해보자.

▫ Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.

▫ Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

▫ Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

▫ Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

▫ Reference3 : It is the practical guide for the army always to heed the directions of the party.

짧은 문장 길이에 대한 패널티(Brevity Penalty)

- Ca의 길이에 BLEU의 점수가 과한 영향을 받을 수 있다.
- 이전의 예제에 다음과 같은 Candidate 3을 추가해보자.

Candidate3 : it is

- Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.
- Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.
- Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.
- Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.
- Reference3 : It is the practical guide for the army always to heed the directions of the party.

짧은 문장 길이에 대한 패널티(Brevity Penalty)

- Ca의 길이에 BLEU의 점수가 과한 영향을 받을 수 있다.
- 이전의 예제에 다음과 같은 Candidate 3을 추가해보자.

Candidate3 : it is ← 유니그램 정밀도, 바이그램 정밀도를 계산하면?

- Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.
- Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.
- Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.
- Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.
- Reference3 : It is the practical guide for the army always to heed the directions of the party.

짧은 문장 길이에 대한 패널티(Brevity Penalty)

- Ca의 길이에 BLEU의 점수가 과한 영향을 받을 수 있다.
- 이전의 예제에 다음과 같은 Candidate 3을 추가해보자.

Candidate3 : it is ← 유니그램 정밀도, 바이그램 정밀도가 둘 다 1.

- Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.
- Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.
- Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.
- Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.
- Reference3 : It is the practical guide for the army always to heed the directions of the party.

짧은 문장 길이에 대한 패널티(Brevity Penalty)

- Ca의 길이에 BLEU의 점수가 과한 영향을 받을 수 있다.
- 이전의 예제에 다음과 같은 Candidate 3을 추가해보자.
- 이와 같이 짧은 문장이 점수를 과하게 받는 경우에 패널티가 필요.

Candidate3 : it is ← 유니그램 정밀도, 바이그램 정밀도가 둘 다 1.

- Candidate1 : It is a guide to action which ensures that the military always obeys the commands of the party.
- Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.
- Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.
- Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.
- Reference3 : It is the practical guide for the army always to heed the directions of the party.

짧은 문장 길이에 대한 패널티(Brevity Penalty)

- Ca의 길이에 BLEU의 점수가 과한 영향을 받을 수 있다.
- 이전의 예제에 다음과 같은 Candidate 3을 추가해보자.
- 이와 같이 짧은 문장이 점수를 과하게 받는 경우에 패널티가 필요.

Candidate3 : it is ← 유니그램 정밀도, 바이그램 정밀도가 둘 다 1.

· Candidate1 : It is **긴 문장에는 패널티가 필요 없을까요?**

· Candidate2 : It is to insure the troops forever hearing the activity guidebook that party direct.

· Reference1 : It is a guide to action that ensures that the military will forever heed Party commands.

· Reference2 : It is the guiding principle which guarantees the military forces always being under the command of the Party.

· Reference3 : It is the practical guide for the army always to heed the directions of the party.

긴 문장에서의 패널티

- Ca1은 Ca2보다 좋지 않은 번역.
- Ref의 단어들을 가장 많이 사용했다고 해서 좋은 번역이 아니다.
- 2-gram, 3-gram, 4-gram 정밀도 계산 과정에서 패널티를 자연스럽게 받는다.

Candidate 1: I always invariably perpetually do.

Candidate 2: I always do.

Reference 1: I always do.


Reference 2: I invariably do.

Reference 3: I perpetually do.

BLEU score

- 짧은 문장에 대한 패널티 BP는 앞서 배운 식에 곱하므로써 적용된다.
- Ca가 모든 Ref보다 길이가 크다면 패널티는 없다!
- Ca와 길이가 정확히 동일한 Ref가 있다면 길이 차이가 0인 최고 수준의 매치!
- 서로 다른 길이의 Ref이지만 Ca와 길이 차이가 동일한 경우에는 더 작은 길이의 Ref를 택한다.
- Ca가 길이가 10인데, Ref 1, 2가 각각 9와 11이라면 9를 택한다.

$$BLEU = \underline{BP} \times \exp\left(\sum_{n=1}^N w_n \log p_n\right)$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$


c : Candidate의 길이

r : Candidate와 가장 길이 차이가 작은 Reference의 길이

BLEU의 한계

- BLEU score가 높다는 것이 사람이 보기에 좋은 번역임을 의미하지는 않는다.
- 결국 서비스를 할 정도의 성능인지 판단하려면 사람의 평가가 필수불가결하다.
- BLEU는 문장의 의미를 고려한 평가가 아니다.
- BLEU는 문장의 구조를 고려하지 않는다.

기계 번역기의 서비스 과정

- 데이터 수집 : 병렬 데이터 구매 또는 크롤링
- 데이터 정제(Cleaning) : 크롤링 된 데이터에는 반드시 노이즈가 있다.
문장 단위로 제대로 정렬이 되었는지 확인해주고, 특수문자 등의 노이즈도 제거해준다.
- 토큰나이저 사용 : Mecab 또는 SentencePiece 사용 권장.
- 데이터의 분리 : 학습 데이터, 검증 데이터, 테스트 데이터 분리.
- 모델 선정 : seq2seq with Attention, Transformer 계열 등의 모델을 선정.
- 학습 : 배치 크기, learning rate 등과 같은 하이퍼파라미터 선정 후 GPU를 통해 학습.
- 추론(테스트) : Beam Search를 이용하여 테스트.
- 역토큰화(Detokenization) : 번역된 문장을 역토큰화하여 자연스러운 문장 형태로 변환.
- 성능 평가 : BLEU, TER 등을 사용하여 모델을 평가 후 개선.
- 모델 배포 : 서버와 웹 서비스를 사용하여 실제 번역기를 서비스화.

번역기를 정말 만들고 싶다면?

- seq2seq with Attention이나 Transformer를 직접 개발하는 것보다는 이미 구현된 오픈소스(OpenNMT, fairseq)를 사용하시는 게 좋습니다.
- 참고 링크 : <https://youtu.be/kuAE9AI-HLk> (삼성 개발자들의 번역기 개발 과정)
- 토큰ON세미나 : <https://youtu.be/3WvA-sFil6w> (기계 번역 개발 과정)
- OpenNMT Tutorial : <https://github.com/Parkchanjun/OpenNMT-Colab-Tutorial>
- 오픈되어져 있는 한-영 번역 데이터(문어체 110만개) : <https://aihub.or.kr/>
- 물론 위 데이터(110만개)로는 부족합니다. 서비스를 위해서는 최소 200만. 좋은 번역기를 위해서는 300만 이상, 다다익선의 데이터가 필요하고, 추가적으로는 데이터 구매나 크롤링을 고려하셔야 합니다.
ex) 플리토 등에서 병렬 데이터 팝니다.
- 위 AI 허브 데이터 + OpenNMT로 만든 번역기 : <https://github.com/jeongwonkwak/OpenNMT-Project>