부스트캠프 AI Tech 2기

boostcamp ai tech

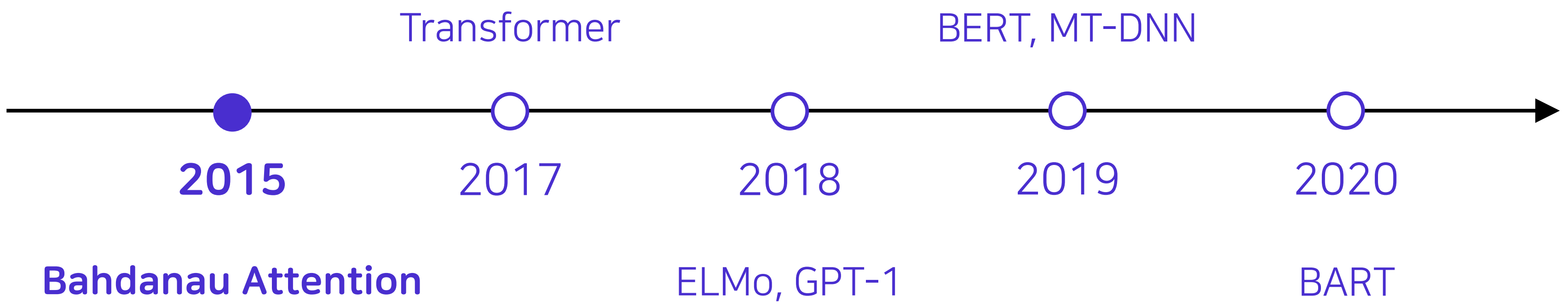
Bahdanau Attention

Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015

Email : jinmang2@gmail.com
GitHub : github.com/jinmang2
Huggingface Hub: huggingface.co/jinmang2

진명훈_T2216

Boostcamp AI Tech 2 NLP 논문 모임에 오신 여러분 환영합니다!



Quick Research History

Why attention? Be someone's shoes!

Experiment and Results

Is enough?

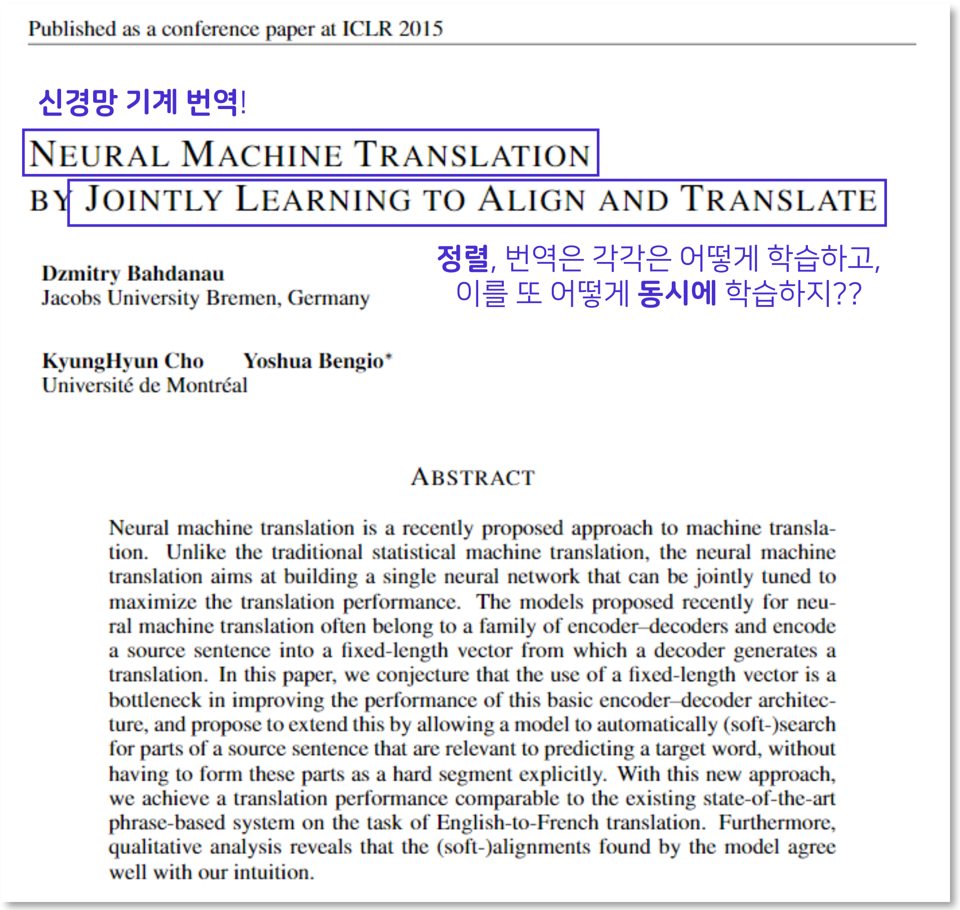
Code Review

00. 논문 및 저자 소개

Dzmitry Bahdanau
AI / ML / DL
Element AI
http://rizar.github.io/

Kyunghyun Cho
Machine Learning / Deep Learning
New York University
https://kyunghyuncho.me/

Yoshua Bengio
ML / DL / AI
University of Montreal, Mila, IVADO, CIFAR
https://yoshuabengio.org/



Published as a conference paper at ICLR 2015

신경망 기계 번역!

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

정렬, 번역은 각각은 어떻게 학습하고,
이를 또 어떻게 동시에 학습하지??

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho Yoshua Bengio*
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

- ✓ 기존 NMT는 번역 성능의 최대화에 초점을 맞추고 **보통 Enc-Dec 구조로 구현**
- ✓ Encoder-Decoder 구조에서 **fixed-sized vector가 문제**인 듯 보임(conjecture)
- ✓ Target word 예측에 연관된 Source sentence part를 자동적으로 찾아주는 (**automatically soft-search**) attention module 추가했더니 SOTA!
- ✓ Qualitative 분석도 해봤는데 우리 직관이랑 맞던데? 우리 짱이지?

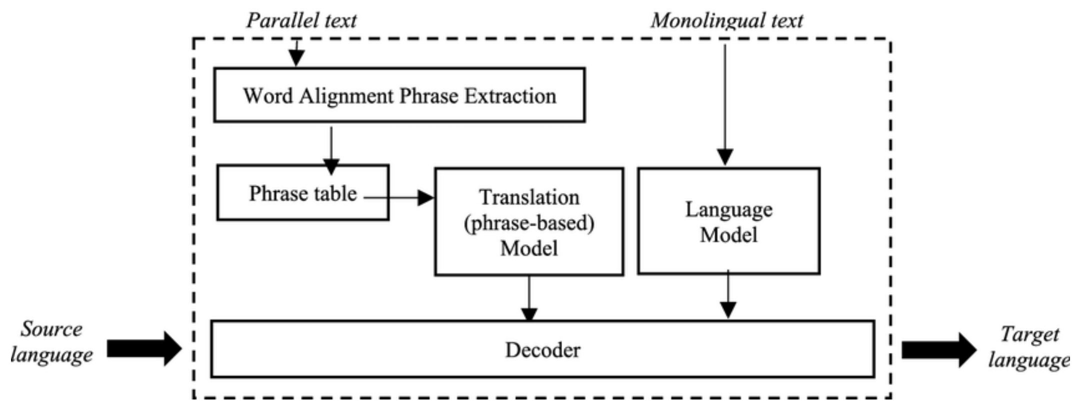
01. Quick Research History

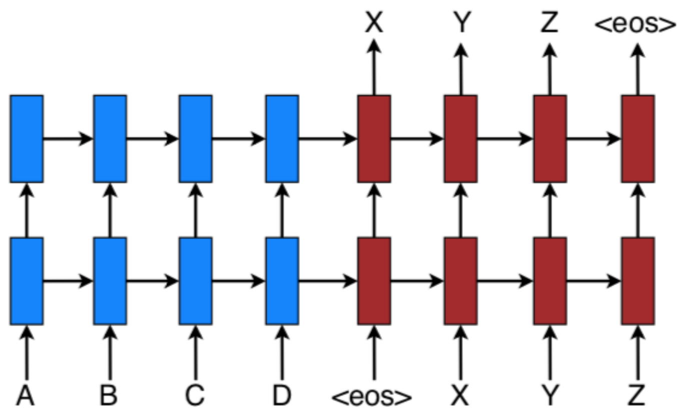
Old: Phrase-based translation

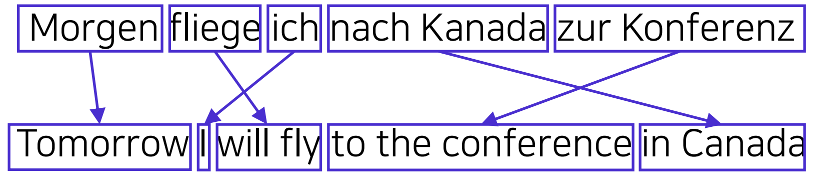
- Lots of individual pieces
- Optimized somewhat independently
- Choice design problem

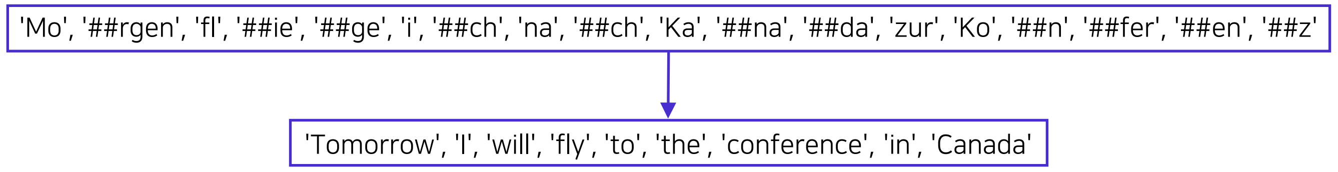
New: Neural Machine Translation

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





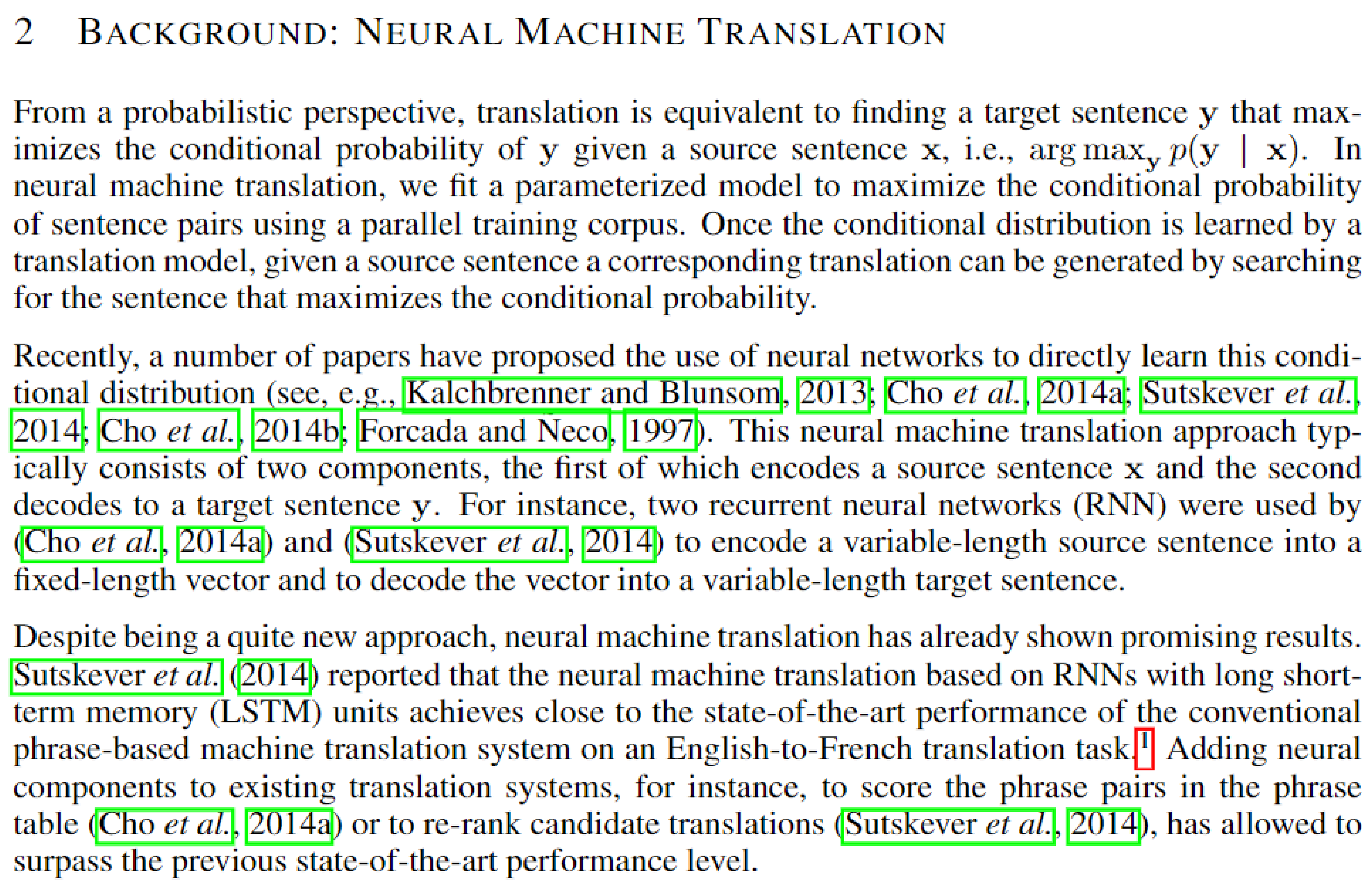



'Mo', '##rgen', 'fl', '##ie', '##ge', 'i', '##ch', 'na', '##ch', 'Ka', '##na', '##da', 'zur', 'Ko', '##n', '##fer', '##en', '##z'

'Tomorrow', 'I', 'will', 'fly', 'to', 'the', 'conference', 'in', 'Canada'

https://www.researchgate.net/figure/Phrase-based-statistical-machine-translation-PBSMT_fig4_343662791
https://www.slideshare.net/ChesterChen/moving-to-neural-machine-translation-at-google-gopromeetup?from_action=save
https://www.researchgate.net/figure/Architecture-of-the-NMT-system-equipped-with-an-attention-mechanism_fig1_308152598

01. Quick Research History



2 Background: Neural Machine Translation

From a probabilistic perspective, translation is equivalent to finding a target sentence $\mathbf{y}$ that maximizes the conditional probability of $\mathbf{y}$ given a source sentence $\mathbf{x}$, i.e., $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$. In neural machine translation, we fit a parameterized model to maximize the conditional probability of sentence pairs using a parallel training corpus. Once the conditional distribution is learned by a translation model, given a source sentence a corresponding translation can be generated by searching for the sentence that maximizes the conditional probability.

Recently, a number of papers have proposed the use of neural networks to directly learn this conditional distribution (see, e.g., Kalchbrenner and Blunsom, 2013; Cho *et al.*, 2014a; Sutskever *et al.*, 2014; Cho *et al.*, 2014b; Forcada and Ñeco, 1997). This neural machine translation approach typically consists of two components, the first of which encodes a source sentence $\mathbf{x}$ and the second decodes to a target sentence $\mathbf{y}$. For instance, two recurrent neural networks (RNN) were used by (Cho *et al.*, 2014a) and (Sutskever *et al.*, 2014) to encode a variable-length source sentence into a fixed-length vector and to decode the vector into a variable-length target sentence.

Despite being a quite new approach, neural machine translation has already shown promising results. Sutskever *et al.* (2014) reported that the neural machine translation based on RNNs with long short-term memory (LSTM) units achieves close to the state-of-the-art performance of the conventional phrase-based machine translation system on an English-to-French translation task.[1] Adding neural components to existing translation systems, for instance, to score the phrase pairs in the phrase table (Cho *et al.*, 2014a) or to re-rank candidate translations (Sutskever *et al.*, 2014), has allowed to surpass the previous state-of-the-art performance level.

- ✓ Maximize conditional probability p(y|x)
- ✓ Fit parameterized model using a parallel training corpus
- ✓ 학습된 모델로 조건부 확률을 최대로 만드는 번역본을 생성

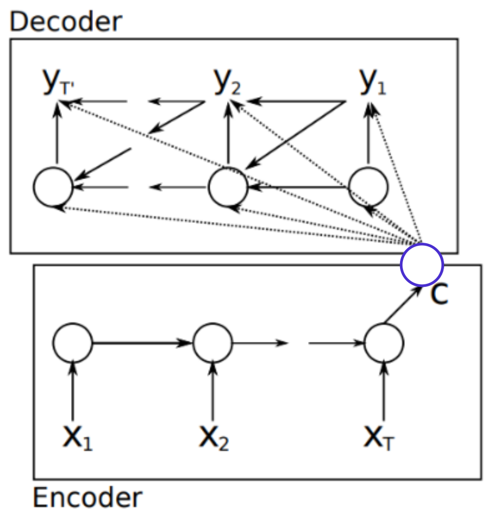
- ✓ NMT의 two component: Encoder and Decoder
- ✓ Variable length src -> fixed vector -> variable length tgt

- ✓ PBSMT와 비슷한 성적을 거두는데 성공!
- ✓ 기존 방식에 Neural Component를 섞으니 성능이 더 올라가더라!

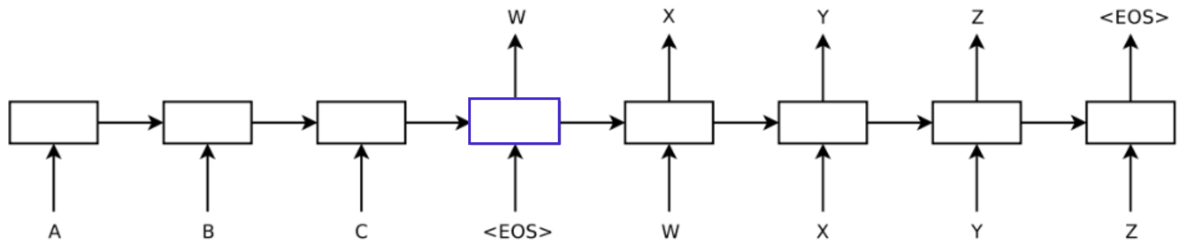
02. Why attention? Be someone's shoes!

Sequence-To-Sequence Models

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



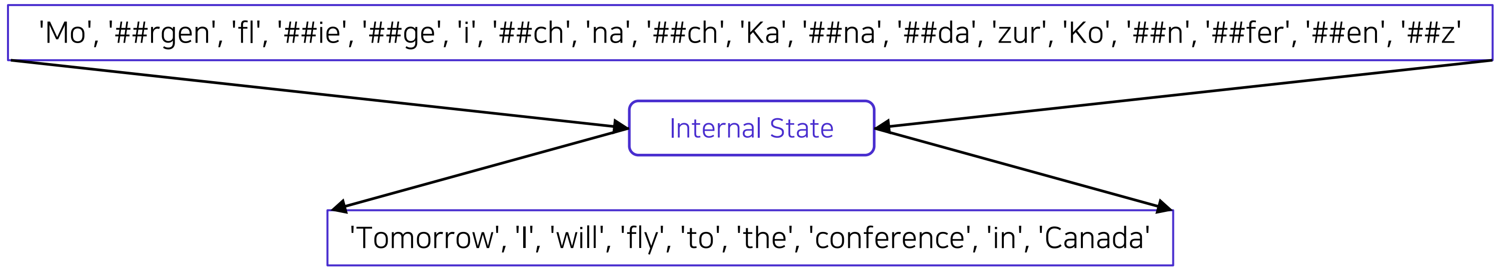
RNN Encoder−decoder
Kyunghyun Cho et al., 2014



Seq2Seq
Ilya Sutskever et al., 2014

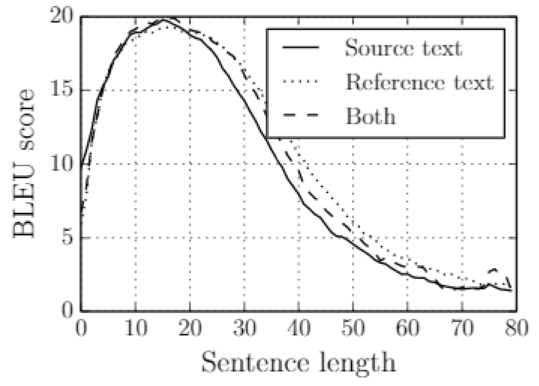
02. Why attention? Be someone's shoes!

A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. Cho *et al.* (2014b) showed that indeed the performance of a basic encoder–decoder deteriorates rapidly as the length of an input sentence increases.

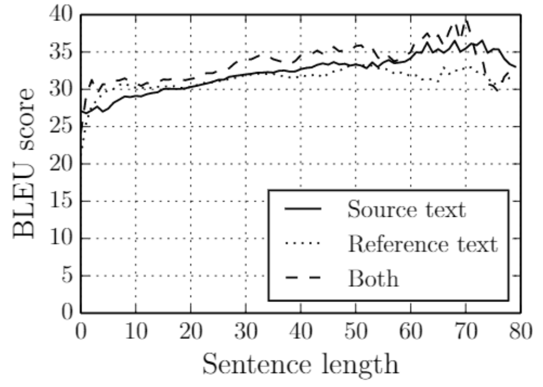


'Mo', '##rgen', 'fl', '##ie', '##ge', 'i', '##ch', 'na', '##ch', 'Ka', '##na', '##da', 'zur', 'Ko', '##n', '##fer', '##en', '##z'
Internal State
'Tomorrow', 'I', 'will', 'fly', 'to', 'the', 'conference', 'in', 'Canada'

NMT (RNNenc)



PBSMT (Moses)



	Model	Development	Test
All	RNNenc	13.15	13.92
	grConv	9.97	9.97
	Moses	30.64	33.30
	Moses+RNNenc⋆	31.48	34.64
	Moses+LSTM°	32	35.65
No UNK	RNNenc	21.01	23.45
	grConv	17.19	18.22
	Moses	32.77	35.63

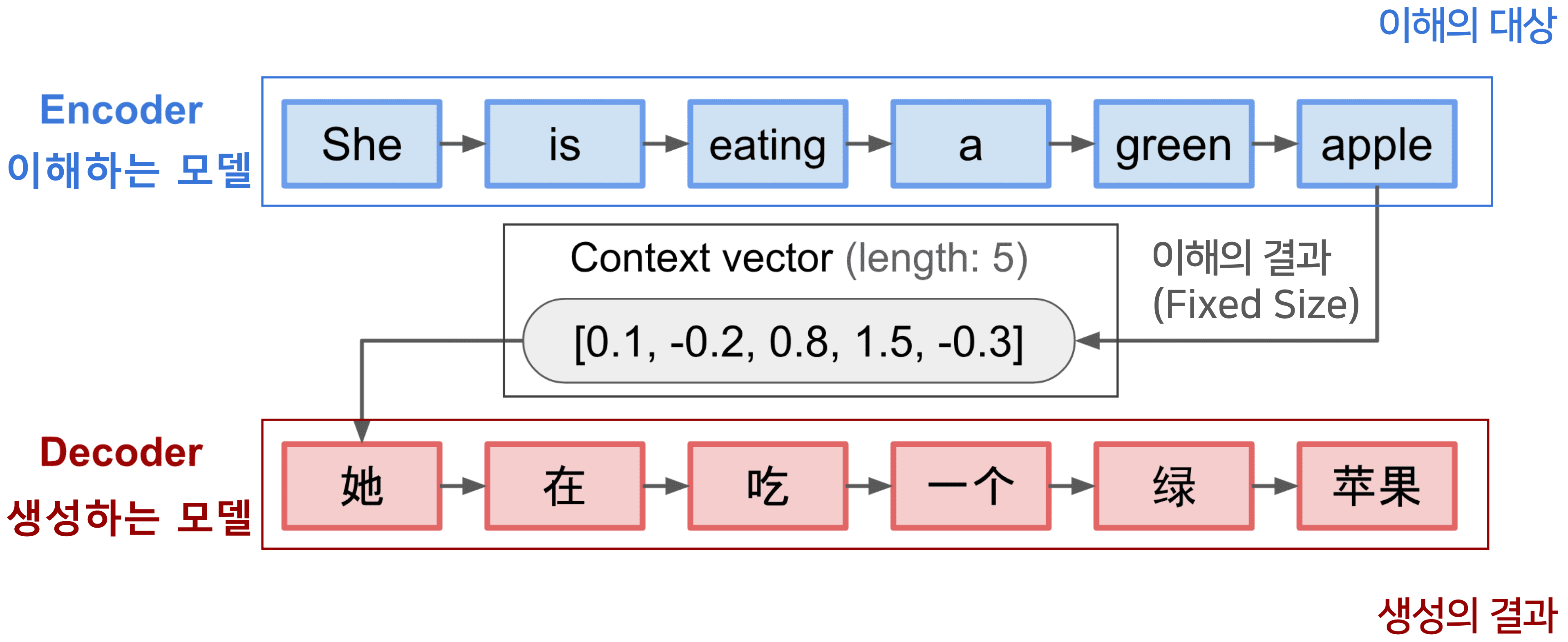
(a) All Lengths

	Model	Development	Test
All	RNNenc	19.12	20.99
	grConv	16.60	17.50
	Moses	28.92	32.00
No UNK	RNNenc	24.73	27.03
	grConv	21.74	22.94
	Moses	32.20	35.40

(b) 10–20 Words

Table 1: BLEU scores computed on the development and test sets. The top three rows show the scores on all the sentences, and the bottom three rows on the sentences having no unknown words. (⋆) The result reported in (Cho et al., 2014) where the RNNenc was used to score phrase pairs in the phrase table. (○) The result reported in (Sutskever et al., 2014) where an encoder–decoder with LSTM units was used to re-rank the n-best list generated by Moses.

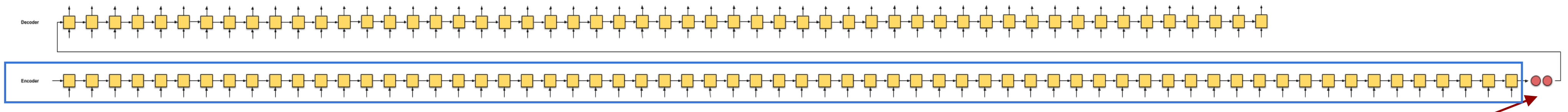
02. Why attention? Be someone's shoes!



Seq2Seq: Input Sequence를 입력으로 받고 Output Sequence를 출력하는 모델 !! (AutoEncoder)

https://youtu.be/KQfvEg-fGMw

02. Why attention? Be someone's shoes!



입력이 길 때, 이해한 모든 내용을 Fixed Size 벡터에 완벽히 표현 가능?

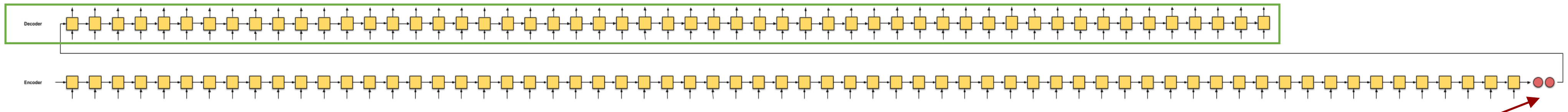
(Encoder) : 엥?? 그런 건 당연히 불가능 하죠…

근데 뭐.. 어쨌든 제가 100페이지 전부 읽어보고 개발자가 지정한 사이즈 (5페이지) 분량으로

Decoder한테 넘겨 드릴게요

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!



입력이 길 때, 이해한 모든 내용을 Fixed Size 벡터에 완벽히 표현 가능?

(Decoder) : 100페이지나 읽어 놓고 딸랑 5페이지 분량만 넘겨준다고?!

아니 겨우 5페이지 분량만 보고 어떻게 100페이지 전체의 내용을 커버해서 생성하죠?

(Context Vector의 고정된 사이즈 때문에 반드시 정보의 손실이 생긴다 !!)

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!

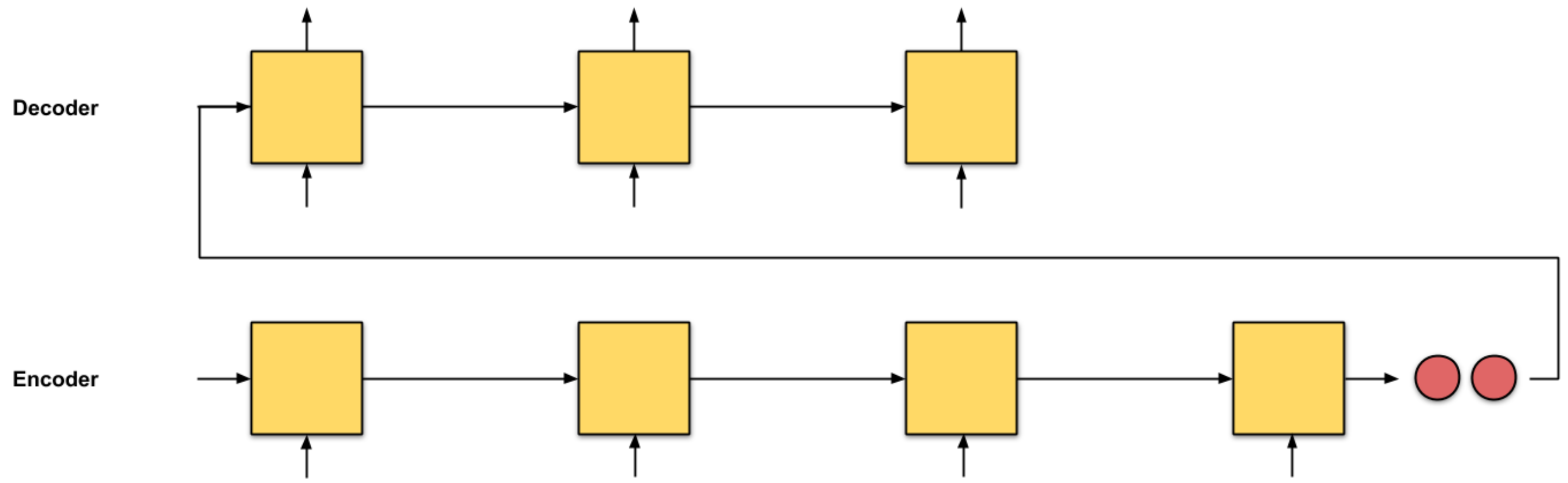
(Decoder) : 저 이렇게는 일 못해요. 아까 한번에 5페이지씩만 넘겨줄 수 있다면서요.

그럼 내용을 한번에 정리해서 넘겨주지 말고 Encoder가 이해한 내용을 "단어(토큰)별"로

5페이지씩 정리해서 보내주세요. 제가 직접 보고나서 중요한 단어만 뽑아서 쓸게요.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!

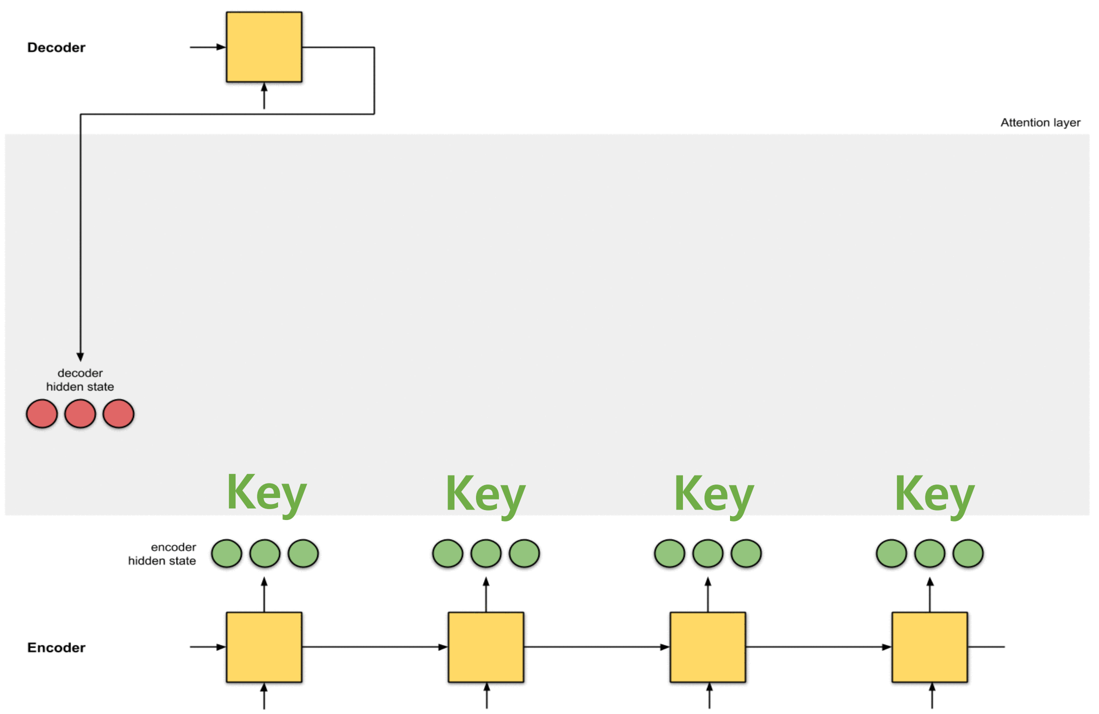


(Decoder) : 그러니까, 이렇게 한꺼번에 모아서 보내주지 마시구요.

(작은 공간에 너무 심하게 압축되어 있으니까 제가 보기 어려워요.)

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!

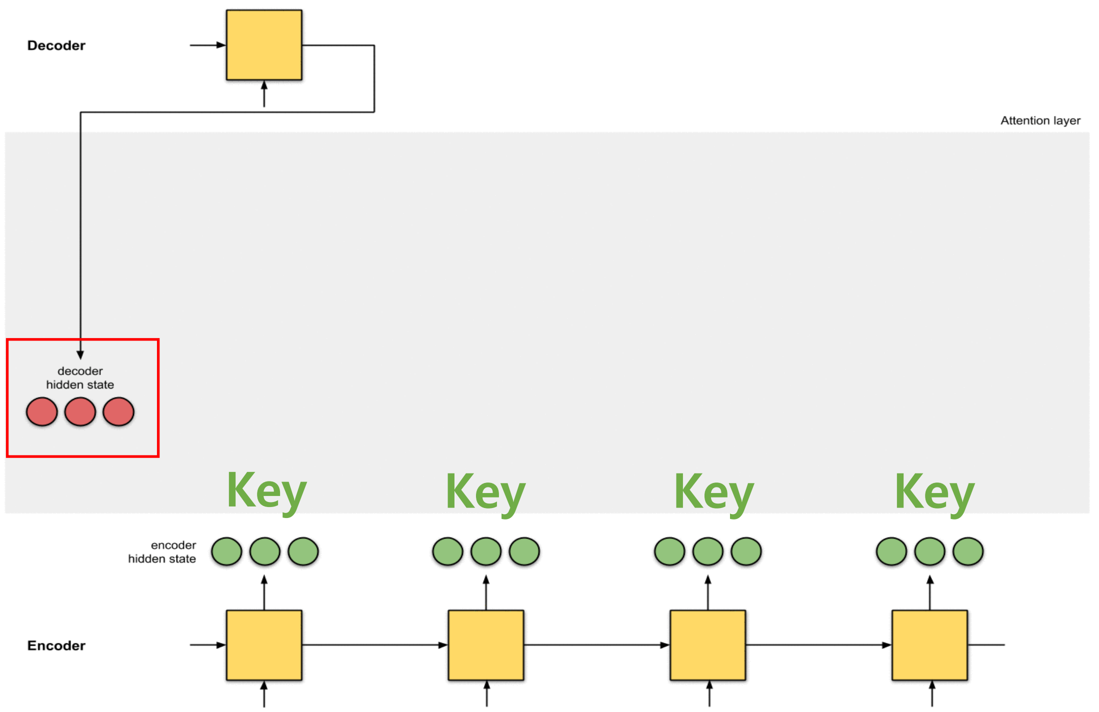


(Decoder) : 이렇게 단어(토큰) 한 개당 하나씩 정리해서 넘겨주세요.

그리고 저는 이 것들(인코더가 토큰마다 이해한 내용)을 Key라고 부를 거에요.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

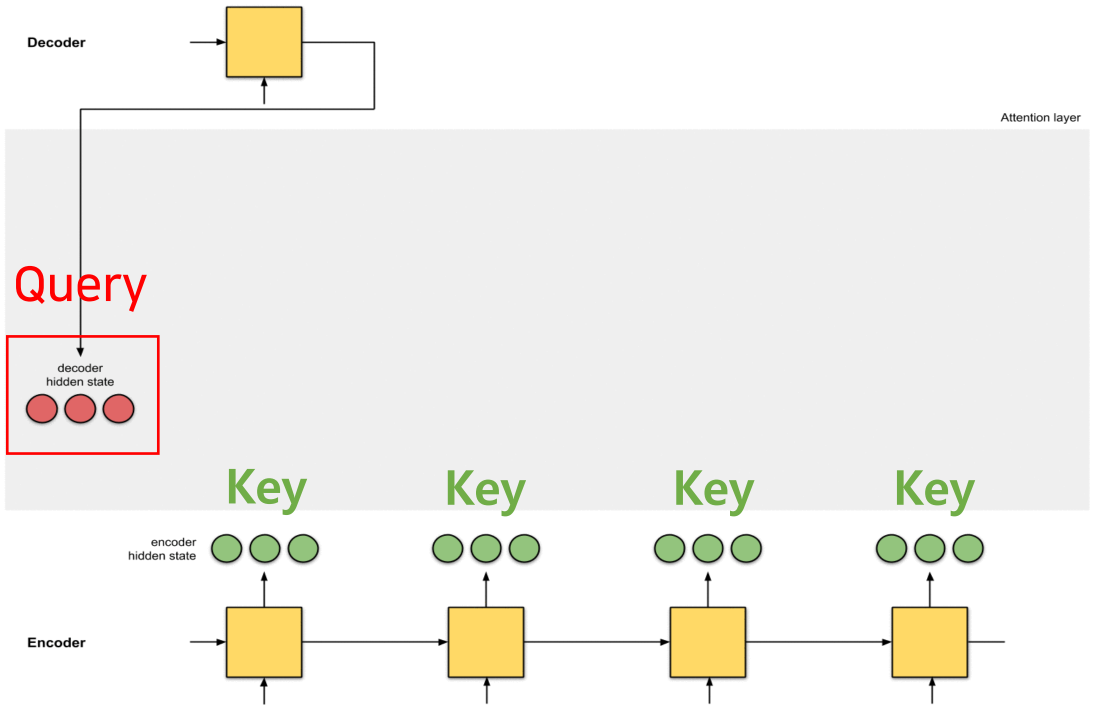
02. Why attention? Be someone's shoes!



(Decoder) : 그 다음에 이건 제가 지금 타임스텝까지 계속 문장을 생성해오면서 고민했던 것들을 정리한 내용이에요.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!

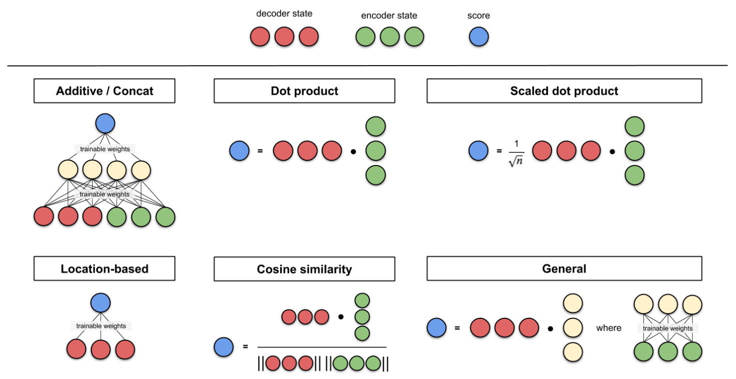


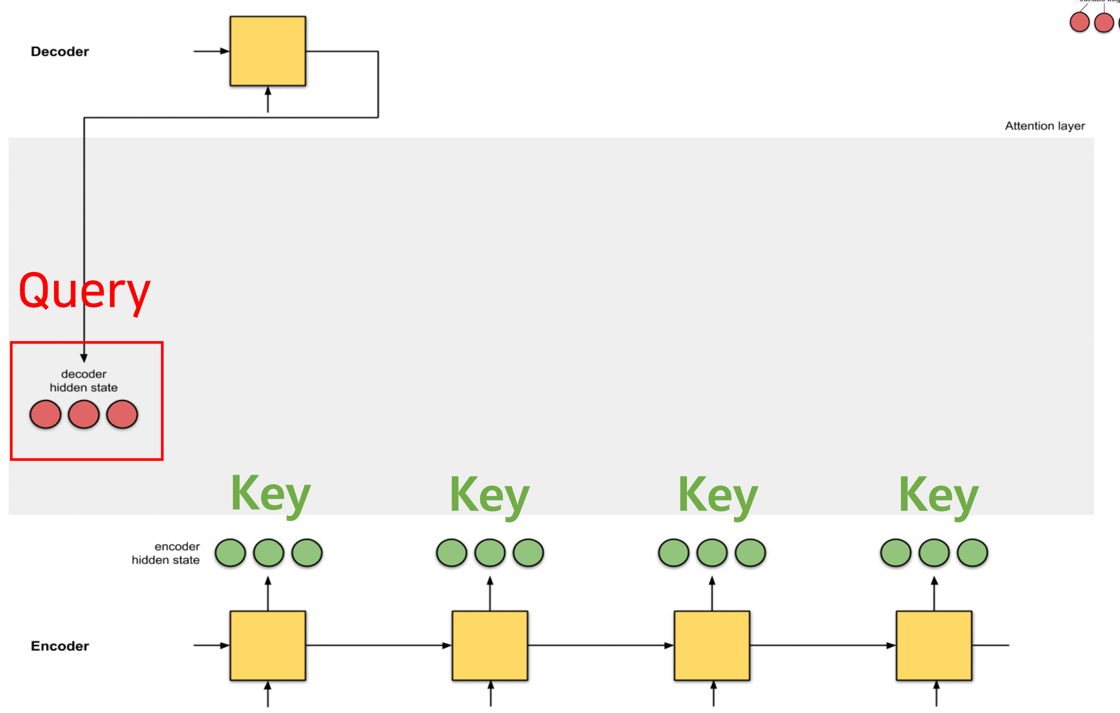
(Decoder) : 저는 이걸 Query라고 부를게요. 흔히 검색 Query라는 말을 쓰기도 하죠.

이 Qeury를 가지고 중요한 Key를 찾을 것이 거든요. 검색 Query 처럼요! (이름의 이유)

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!

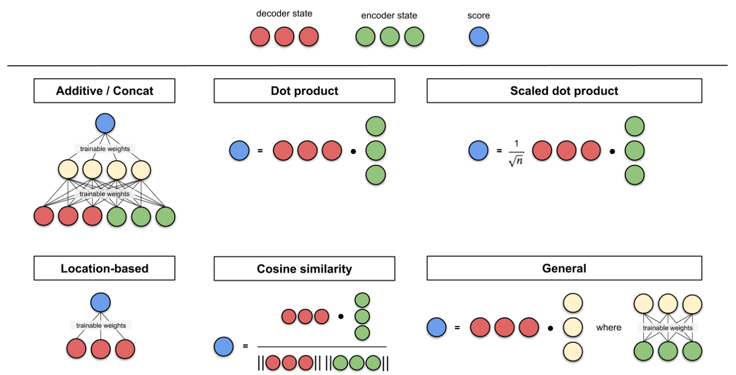


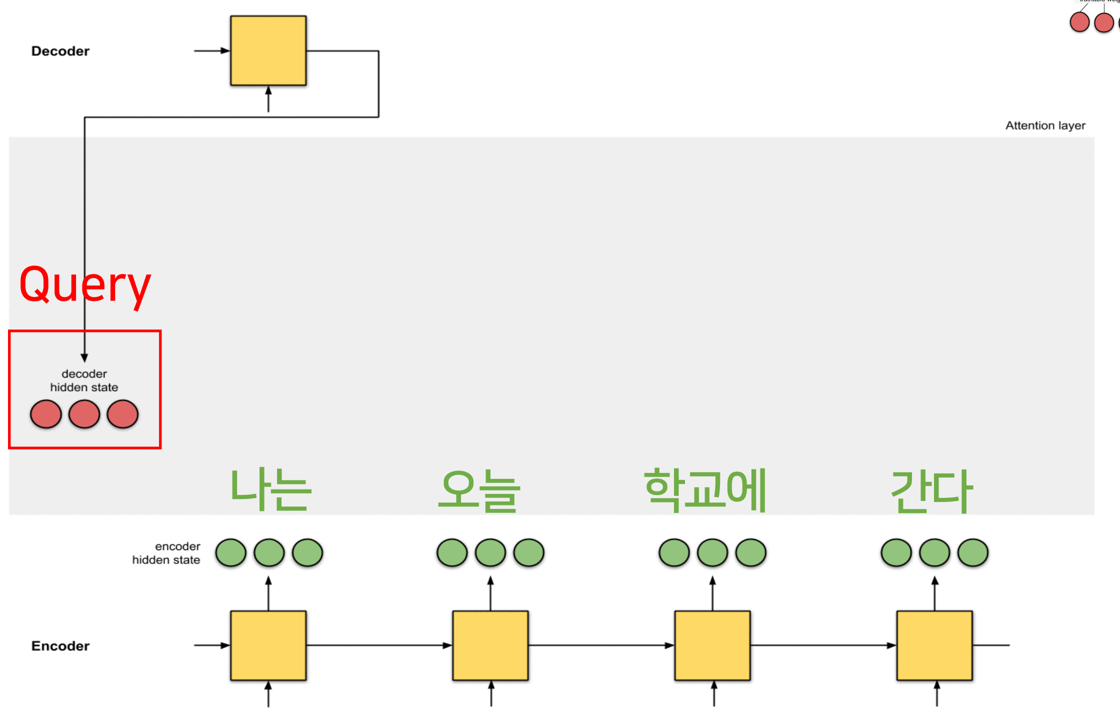


(Decoder) : 이제 이 Query를 가지고 중요한 Key를 찾아볼 거에요. Score function을 이용해서 찾을 건데 사이즈가 1(스칼라)인 유사도(score)가 돼요.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

02. Why attention? Be someone's shoes!



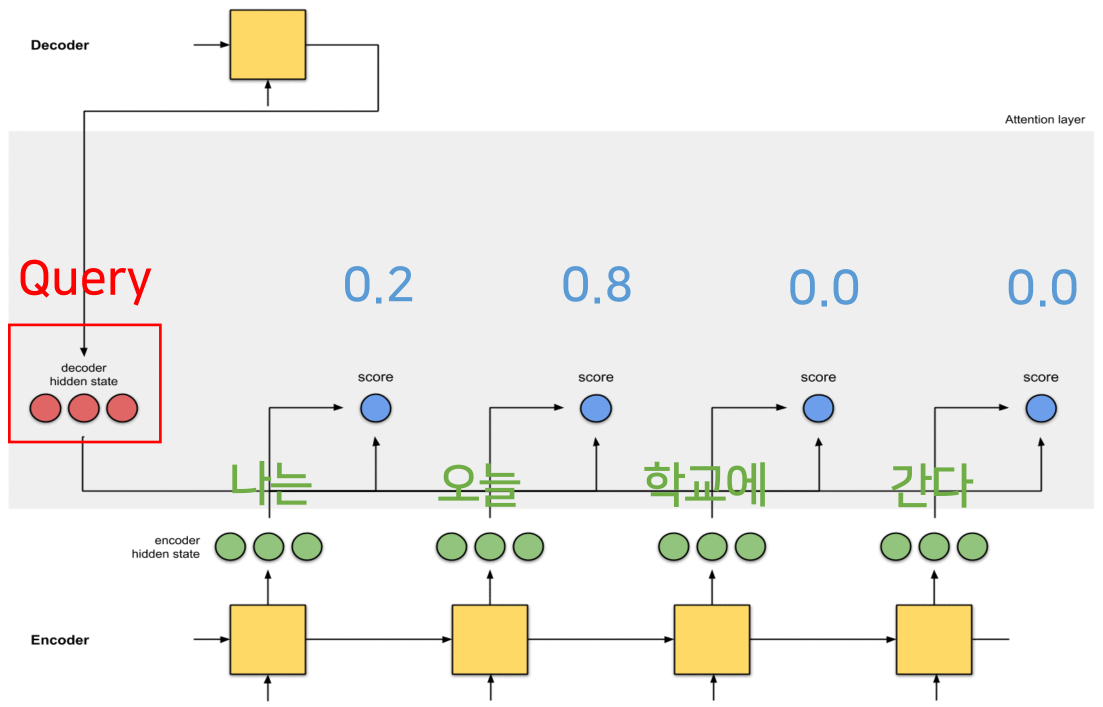


(Decoder) : 가령 입력 시퀀스 = ["나는", "오늘", "학교에", "간다 "]라면 Key 1, 2, 3, 4는 각각 인코더가 "나는", "오늘", "학교를", "간다 " 를 보고 나서 정리한 벡터가 되고, 이들과 현재 Query를 주어진 Score function으로 유사도를 구합니다.

이해하기 쉽게 Dot Product라고 가정해볼게요!

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

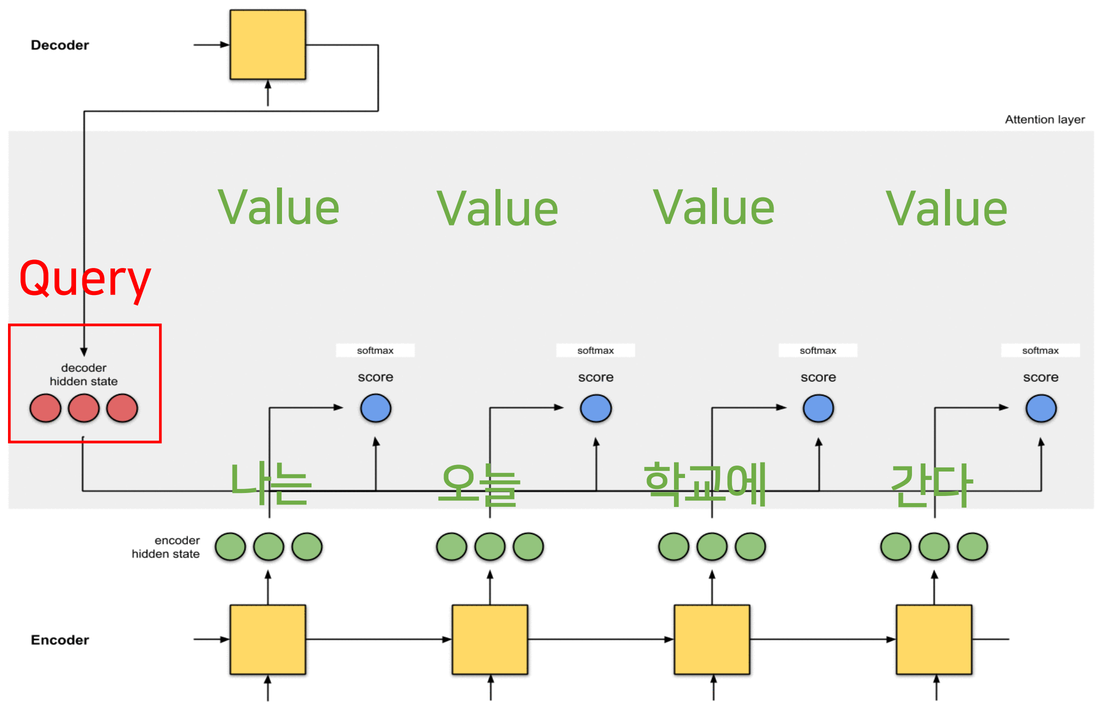
02. Why attention? Be someone's shoes!



(Decoder) : 그리고 나서 이 값들에 Softmax를 취하면 0 ~ 1 사이의 값이 되죠. 가령 현재까지 제가 번역해오면서 정리한 결과인 Query와 "나는 " 의 유사도는 0.2, "오늘 " 과의 유사도는 0.8, 나머지는 0이였다고 해보죠.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

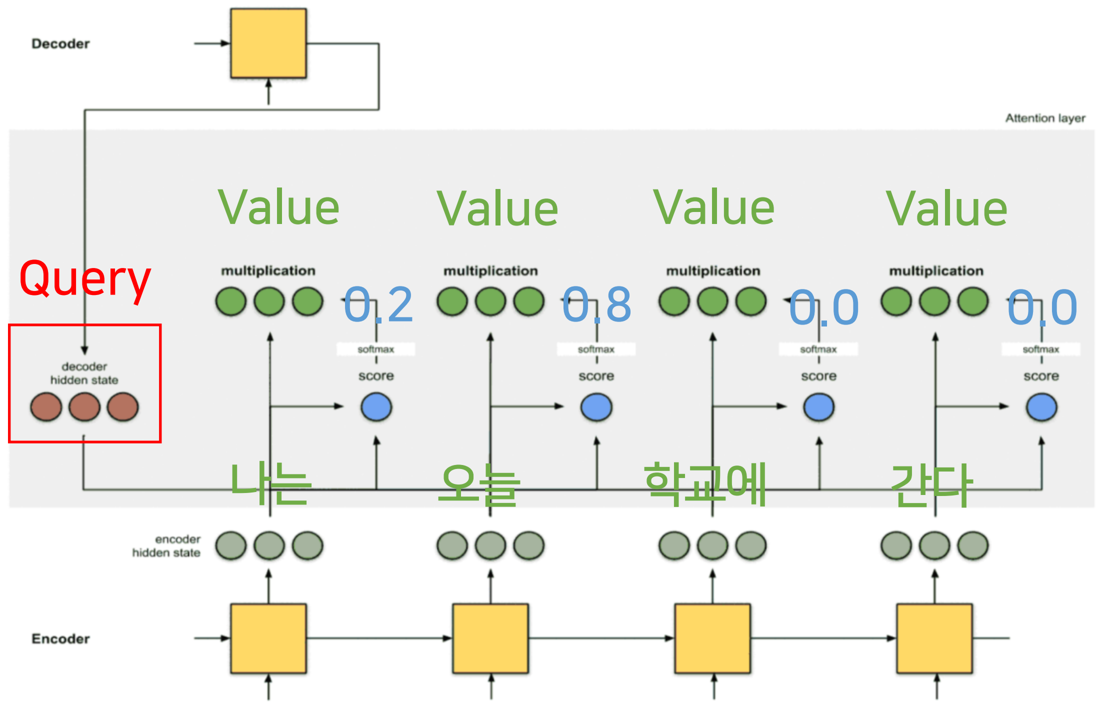
02. Why attention? Be someone's shoes!



(Decoder) : 이제 이 score를 기존의 Key 벡터에 곱할 거에요. 이걸 저는 Value라고 부를 거에요. 왜 Value냐고 하면 Dictionary 처럼 각 Key는 score와 곱해진 각각의 Value값을 가지고 있게 되기 때문이에요 (이름의 이유)

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

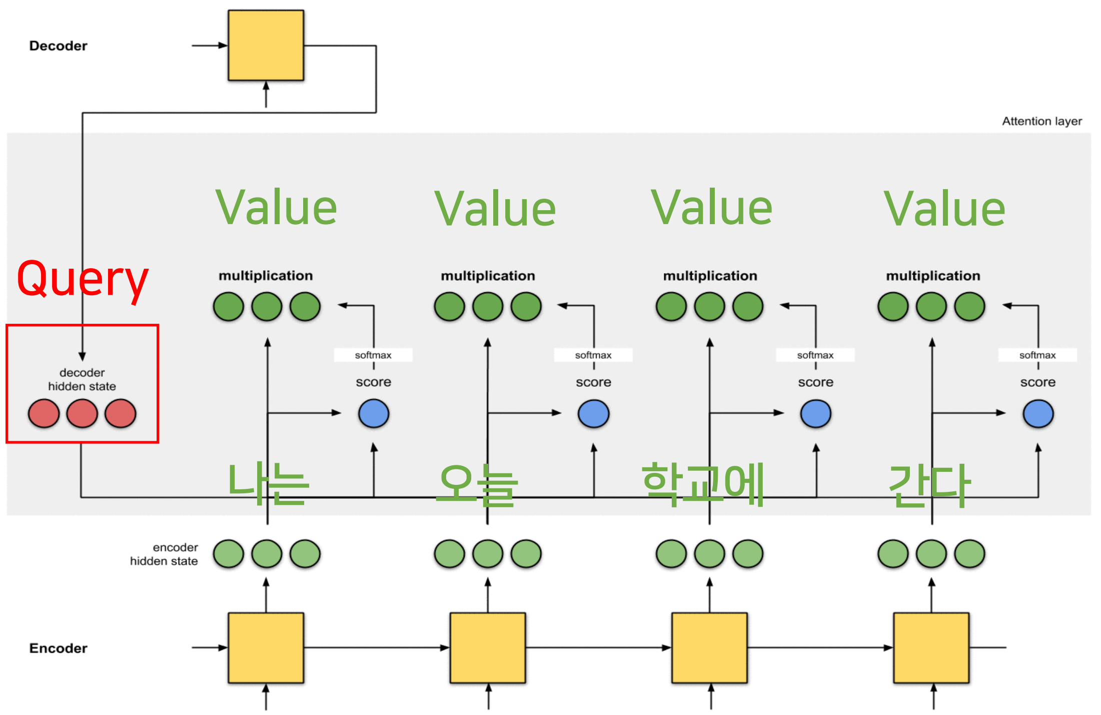
02. Why attention? Be someone’s shoes!



(Decoder) : 기존의 각 Key 벡터에 유사도 값을 곱하면 어떤 Key 벡터는 희미해지고, 어떤 Key 벡터는 비교적 값을 잃지 않겠죠? 가령 “나는“의 key 벡터가 [1, 2, 9]였다면 0.2를 곱해서 value 벡터는 [0.2, 0.4, 1.8]가 될 거에요.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

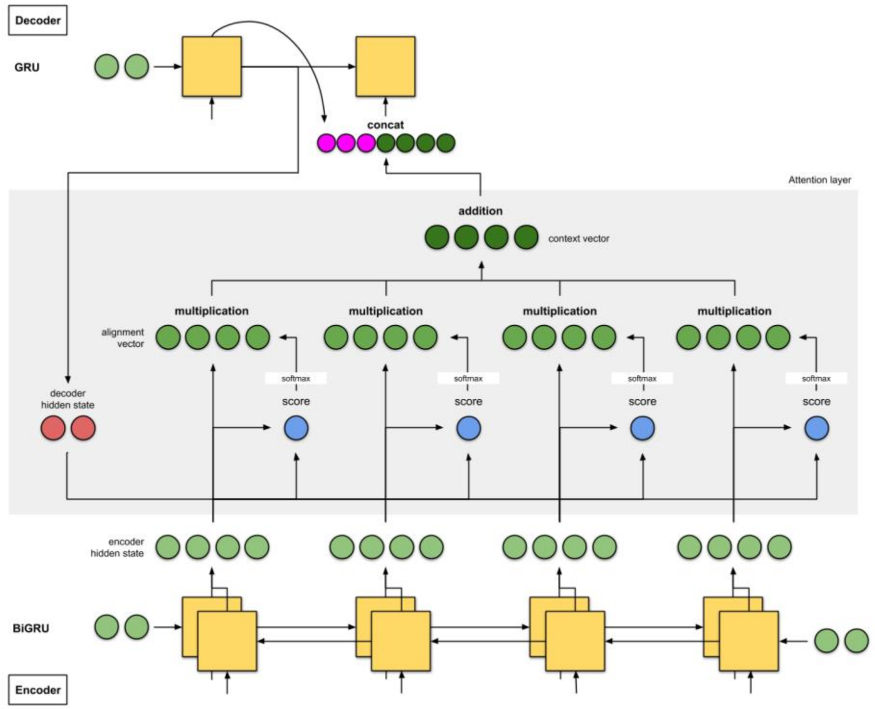
02. Why attention? Be someone's shoes!



(Decoder) : 이제 모든 value 벡터를 더 해서 입력으로 쓸 거에요.. "나는 " 의 [0.2, 0.4, 1.8] + "오늘"의 [7.2, 0.8, 0.8]이 더해져서 [8.0, 1.2, 2.6]이 최종 value 벡터가 될거에요. ("오늘 " 의 key 벡터가 많이 반영되었죠?)

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

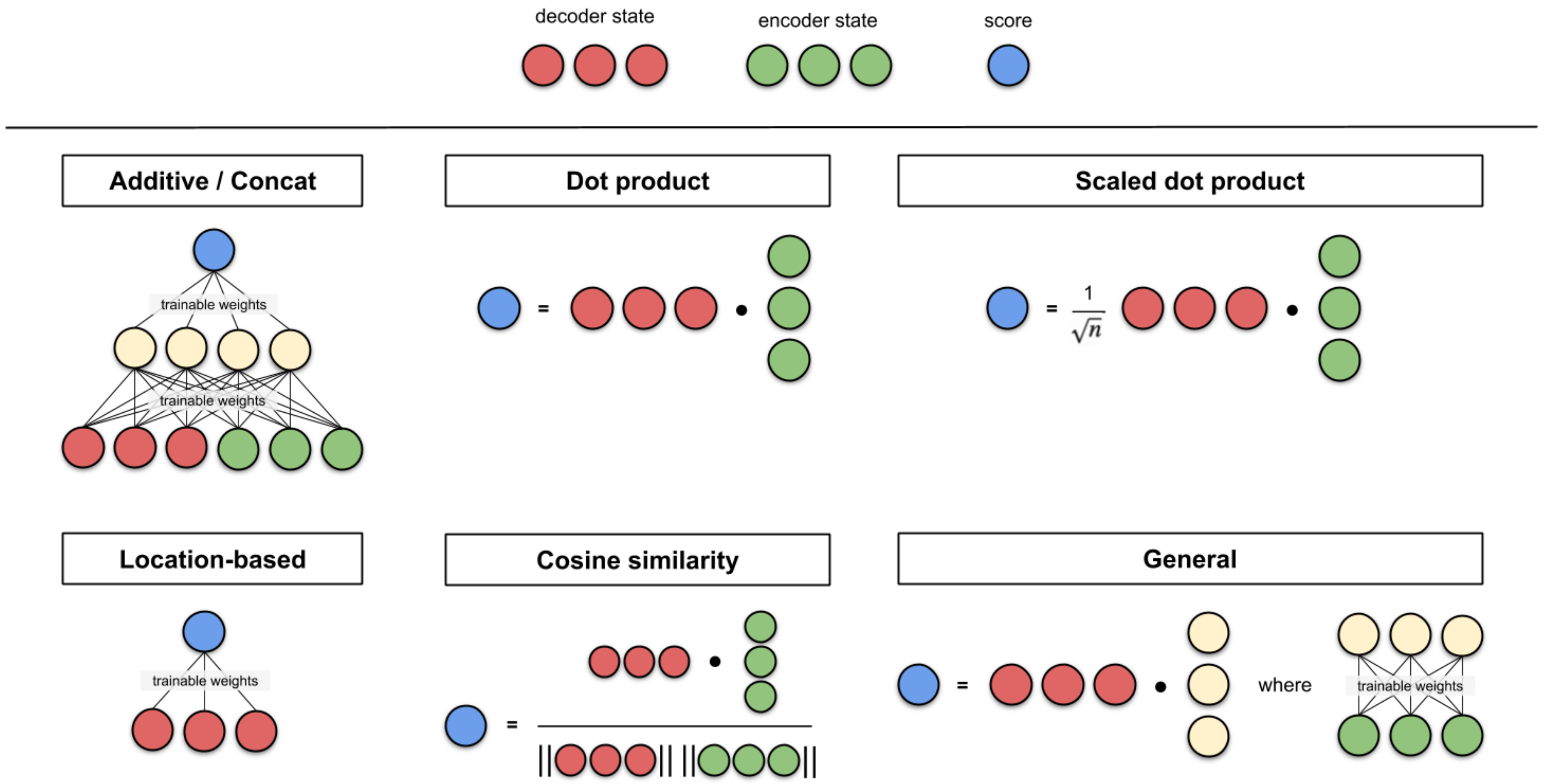
02. Why attention? Be someone's shoes!



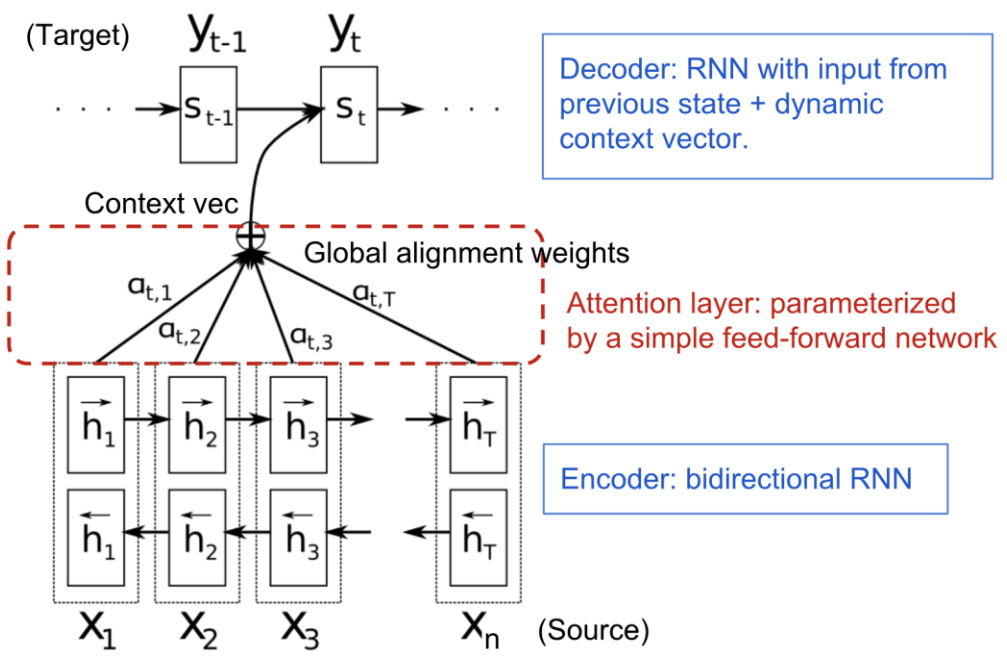
(Decoder) : 이 최종 value 벡터와 이전 스텝까지 디코더가 정리한 벡터를 concat해서 다음 입력으로 사용할 거에요.

어텐션 메커니즘은 인코더가 단어(토큰) 단위로 정리한 벡터 중에서 디코더가 중요한 벡터를 골라서 쓰는 방법입니다.

https://youtu.be/KQfvEg-fGMw
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3


decoder state
encoder state
score
Additive / Concat
trainable weights
trainable weights
Dot product
=
•
Scaled dot product
= $\frac{1}{\sqrt{n}}$
•
Location-based
trainable weights
Cosine similarity
=
•
|| || || ||
General
=
•
where
trainable weights

02. Why attention? Be someone's shoes!


(Target)
y_{t-1}
y_t
s_{t-1}
s_t
Decoder: RNN with input from previous state + dynamic context vector.
Context vec
Global alignment weights
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
Attention layer: parameterized by a simple feed-forward network
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
Encoder: bidirectional RNN
x_1 x_2 x_3 x_n
(Source)

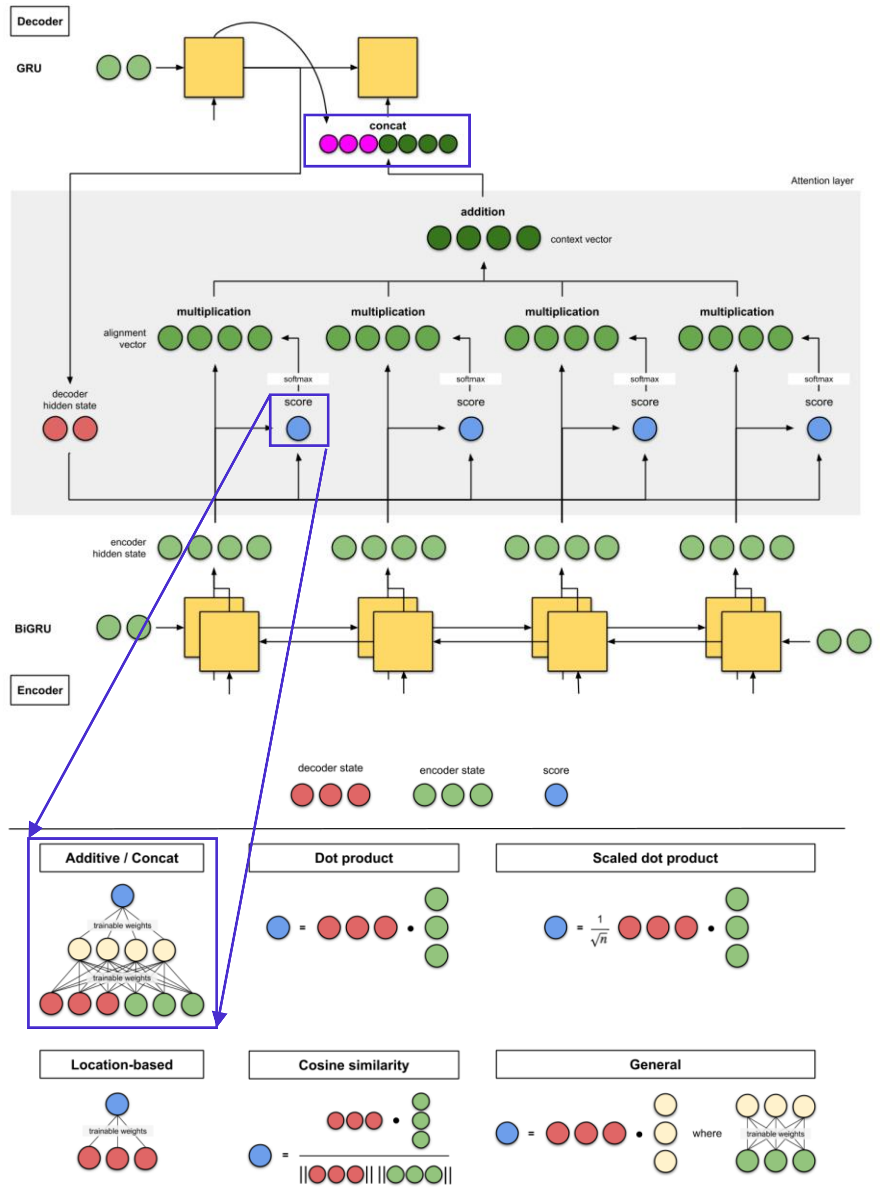
Additive Attention

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i \qquad \text{; Context vector for output } y_t$$

$$\alpha_{t,i} = \text{align}(y_t, x_i) \qquad \text{; How well two words } y_t \text{ and } x_i \text{ are aligned.}$$

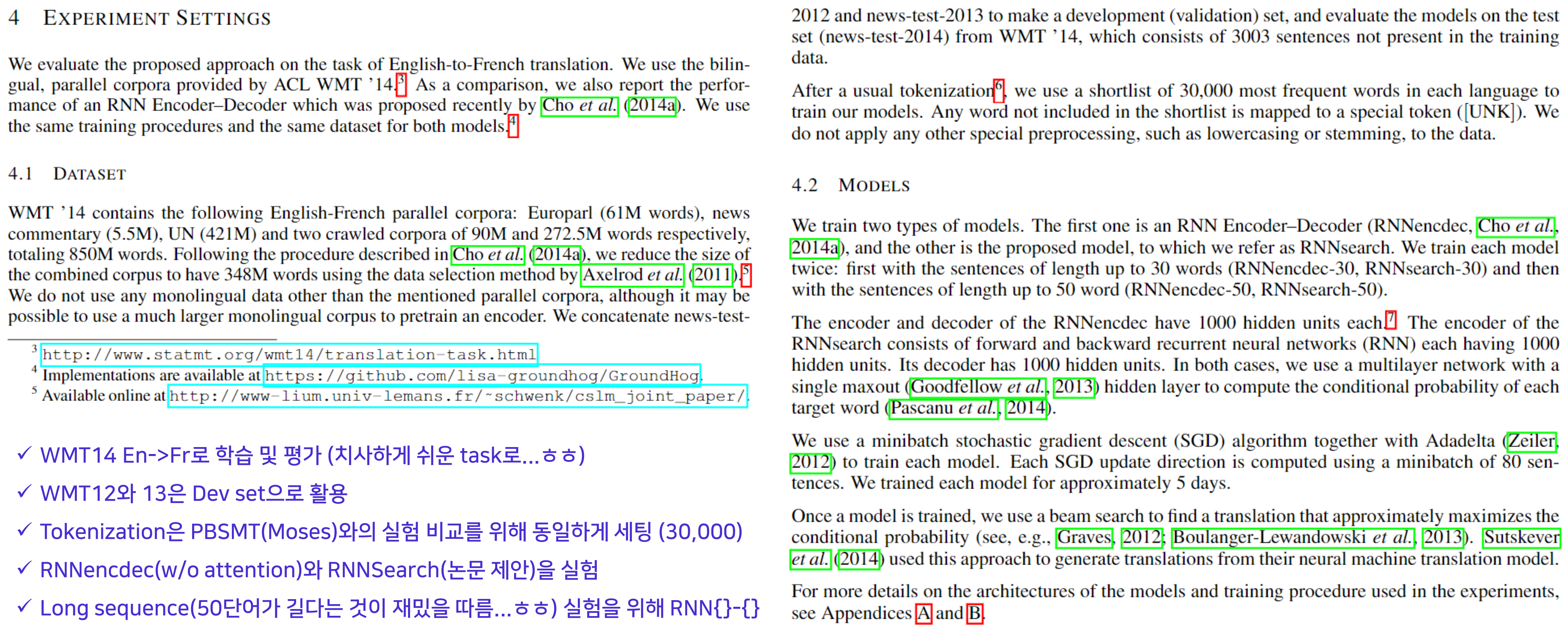
$$= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} \qquad \text{; Softmax of some predefined alignment score..}$$

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3
https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

03. Experiment and Results



4 Experiment Settings

We evaluate the proposed approach on the task of English-to-French translation. We use the bilingual, parallel corpora provided by ACL WMT '14.[3] As a comparison, we also report the performance of an RNN Encoder–Decoder which was proposed recently by Cho *et al.* (2014a). We use the same training procedures and the same dataset for both models.[4]

4.1 Dataset

WMT '14 contains the following English-French parallel corpora: Europarl (61M words), news commentary (5.5M), UN (421M) and two crawled corpora of 90M and 272.5M words respectively, totaling 850M words. Following the procedure described in Cho *et al.* (2014a), we reduce the size of the combined corpus to have 348M words using the data selection method by Axelrod *et al.* (2011).[5] We do not use any monolingual data other than the mentioned parallel corpora, although it may be possible to use a much larger monolingual corpus to pretrain an encoder. We concatenate news-test-2012 and news-test-2013 to make a development (validation) set, and evaluate the models on the test set (news-test-2014) from WMT '14, which consists of 3003 sentences not present in the training data.

After a usual tokenization[6], we use a shortlist of 30,000 most frequent words in each language to train our models. Any word not included in the shortlist is mapped to a special token ([UNK]). We do not apply any other special preprocessing, such as lowercasing or stemming, to the data.

4.2 Models

We train two types of models. The first one is an RNN Encoder–Decoder (RNNencdec, Cho *et al.*, 2014a), and the other is the proposed model, to which we refer as RNNsearch. We train each model twice: first with the sentences of length up to 30 words (RNNencdec-30, RNNsearch-30) and then with the sentences of length up to 50 word (RNNencdec-50, RNNsearch-50).

The encoder and decoder of the RNNencdec have 1000 hidden units each.[7] The encoder of the RNNsearch consists of forward and backward recurrent neural networks (RNN) each having 1000 hidden units. Its decoder has 1000 hidden units. In both cases, we use a multilayer network with a single maxout (Goodfellow *et al.*, 2013) hidden layer to compute the conditional probability of each target word (Pascanu *et al.*, 2014).

We use a minibatch stochastic gradient descent (SGD) algorithm together with Adadelta (Zeiler, 2012) to train each model. Each SGD update direction is computed using a minibatch of 80 sentences. We trained each model for approximately 5 days.

Once a model is trained, we use a beam search to find a translation that approximately maximizes the conditional probability (see, e.g., Graves, 2012; Boulanger-Lewandowski *et al.*, 2013). Sutskever *et al.* (2014) used this approach to generate translations from their neural machine translation model.

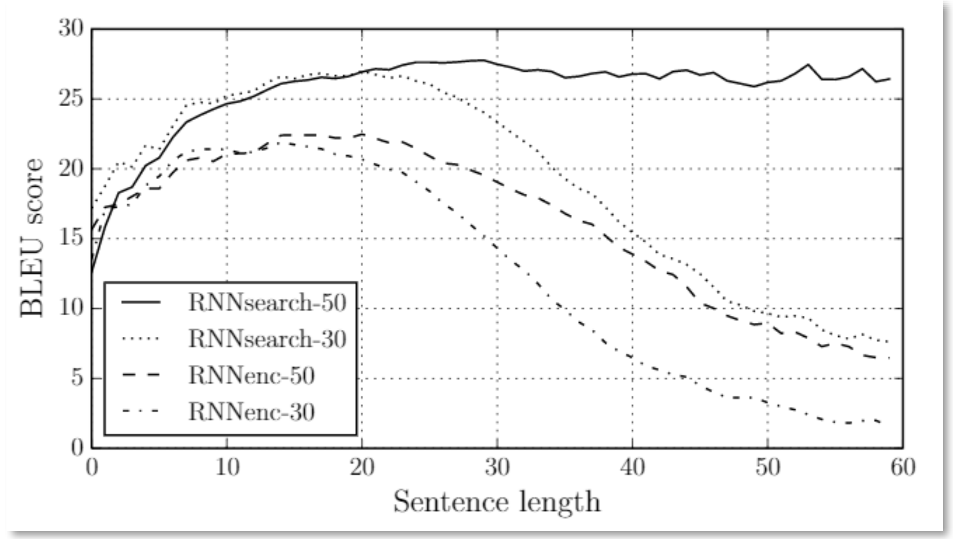
For more details on the architectures of the models and training procedure used in the experiments, see Appendices A and B.

[3] `http://www.statmt.org/wmt14/translation-task.html`
[4] Implementations are available at `https://github.com/lisa-groundhog/GroundHog`.
[5] Available online at `http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/`.

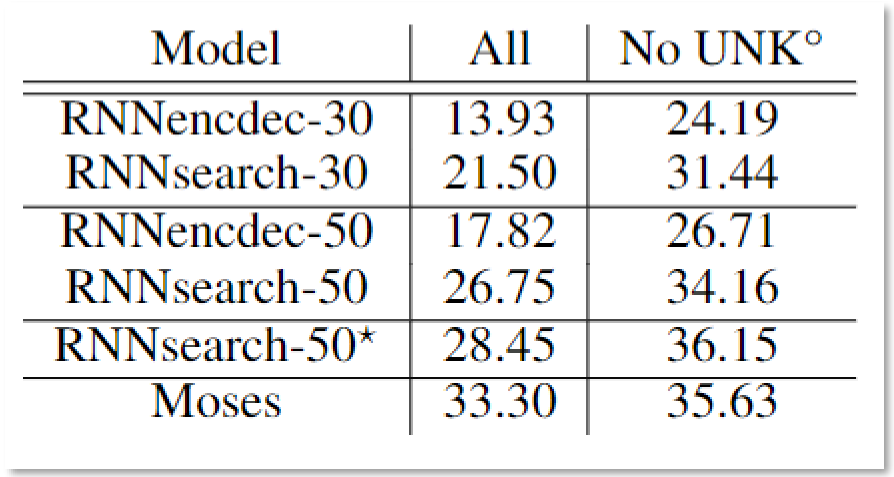
- ✓ WMT14 En->Fr로 학습 및 평가 (치사하게 쉬운 task로…ㅎㅎ)
- ✓ WMT12와 13은 Dev set으로 활용
- ✓ Tokenization은 PBSMT(Moses)와의 실험 비교를 위해 동일하게 세팅 (30,000)
- ✓ RNNencdec(w/o attention)와 RNNSearch(논문 제안)을 실험
- ✓ Long sequence(50단어가 길다는 것이 재밌을 따름…ㅎㅎ) 실험을 위해 RNN{}-{} 뒤의 forma에 sequence length를 달아줌
- ✓ hidden_size = 1,000 maxout activation function
- ✓ Adadelta, mini-batch 80, 5일 동안 학습
- ✓ Beam Search로 inference

03. Experiment and Results

AFTER

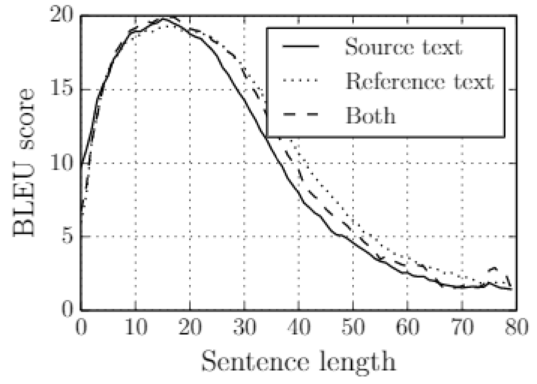


Test set

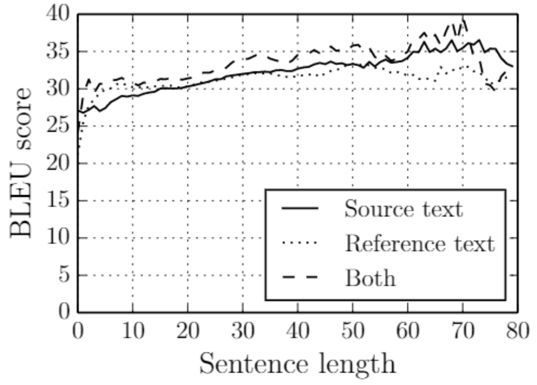


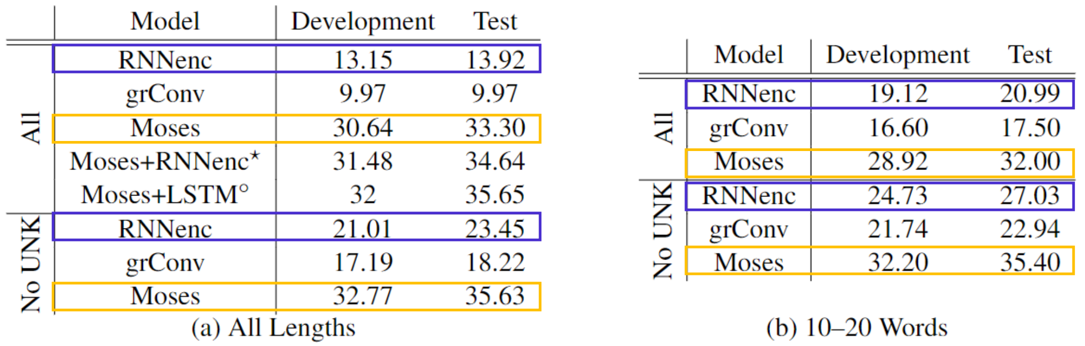
Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

NMT (RNNenc)



PBSMT (Moses)





	Model	Development	Test
All	RNNenc	13.15	13.92
	grConv	9.97	9.97
	Moses	30.64	33.30
	Moses+RNNenc*	31.48	34.64
	Moses+LSTM°	32	35.65
No UNK	RNNenc	21.01	23.45
	grConv	17.19	18.22
	Moses	32.77	35.63

(a) All Lengths

	Model	Development	Test
All	RNNenc	19.12	20.99
	grConv	16.60	17.50
	Moses	28.92	32.00
No UNK	RNNenc	24.73	27.03
	grConv	21.74	22.94
	Moses	32.20	35.40

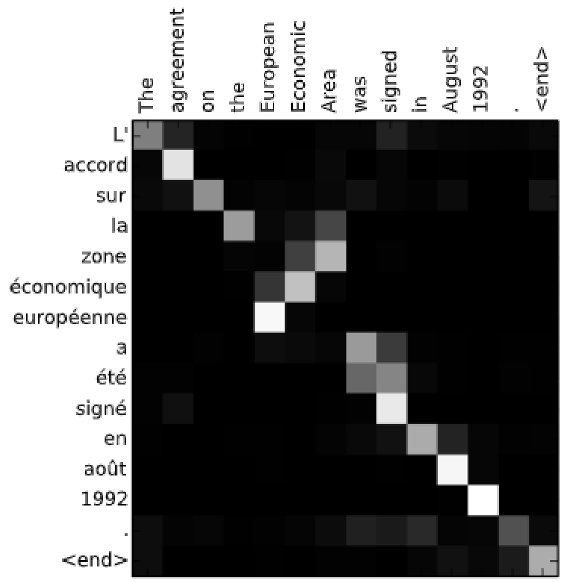
(b) 10–20 Words

Table 1: BLEU scores computed on the development and test sets. The top three rows show the scores on all the sentences, and the bottom three rows on the sentences having no unknown words. (⋆) The result reported in (Cho et al., 2014) where the RNNenc was used to score phrase pairs in the phrase table. (∘) The result reported in (Sutskever et al., 2014) where an encoder–decoder with LSTM units was used to re-rank the n-best list generated by Moses.

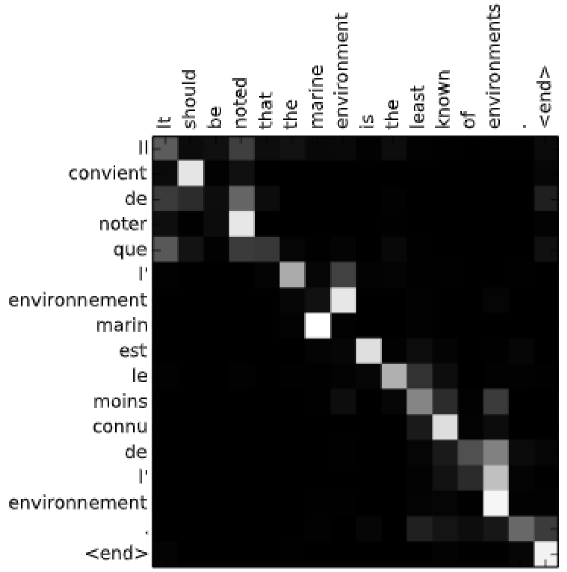
https://arxiv.org/pdf/1409.1259.pdf

BEFORE

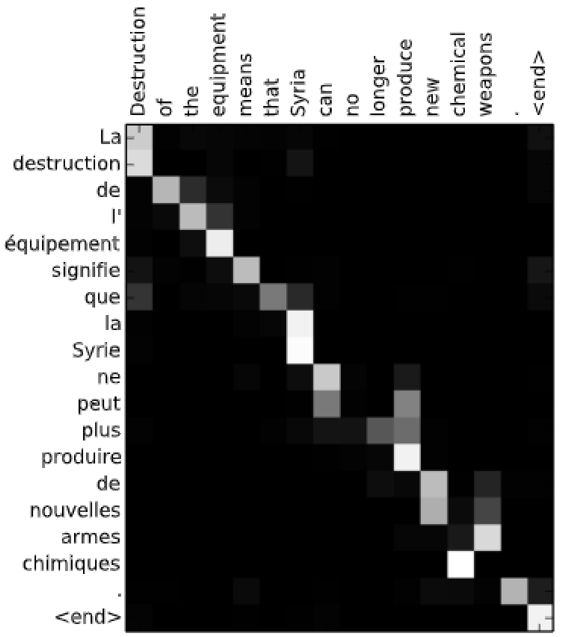
03. Experiment and Results



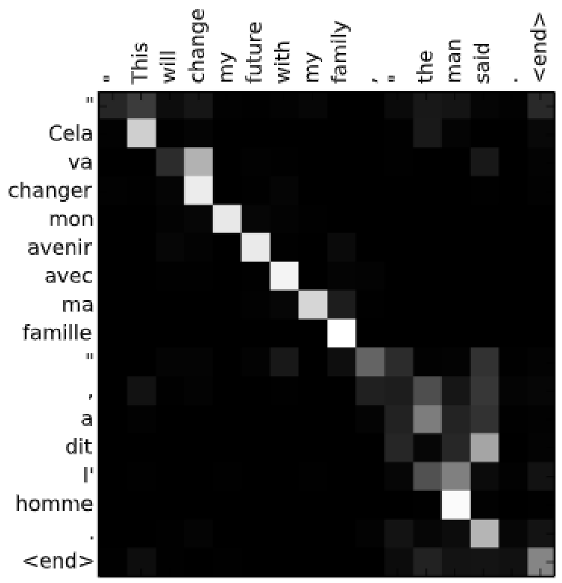
(a)



(b)



(c)



(d)

5.2 QUALITATIVE ANALYSIS

5.2.1 ALIGNMENT

The proposed approach provides an intuitive way to inspect the (soft-)alignment between the words in a generated translation and those in a source sentence. This is done by visualizing the annotation weights α_{ij} from Eq. (6), as in Fig. 3. Each row of a matrix in each plot indicates the weights associated with the annotations. From this we see which positions in the source sentence were considered more important when generating the target word.

We can see from the alignments in Fig. 3 that the alignment of words between English and French is largely monotonic. We see strong weights along the diagonal of each matrix. However, we also observe a number of non-trivial, non-monotonic alignments. Adjectives and nouns are typically ordered differently between French and English, and we see an example in Fig. 3 (a). From this figure, we see that the model correctly translates a phrase [European Economic Area] into [zone économique européen]. The RNNsearch was able to correctly align [zone] with [Area], jumping over the two words ([European] and [Economic]), and then looked one word back at a time to complete the whole phrase [zone économique européenne].

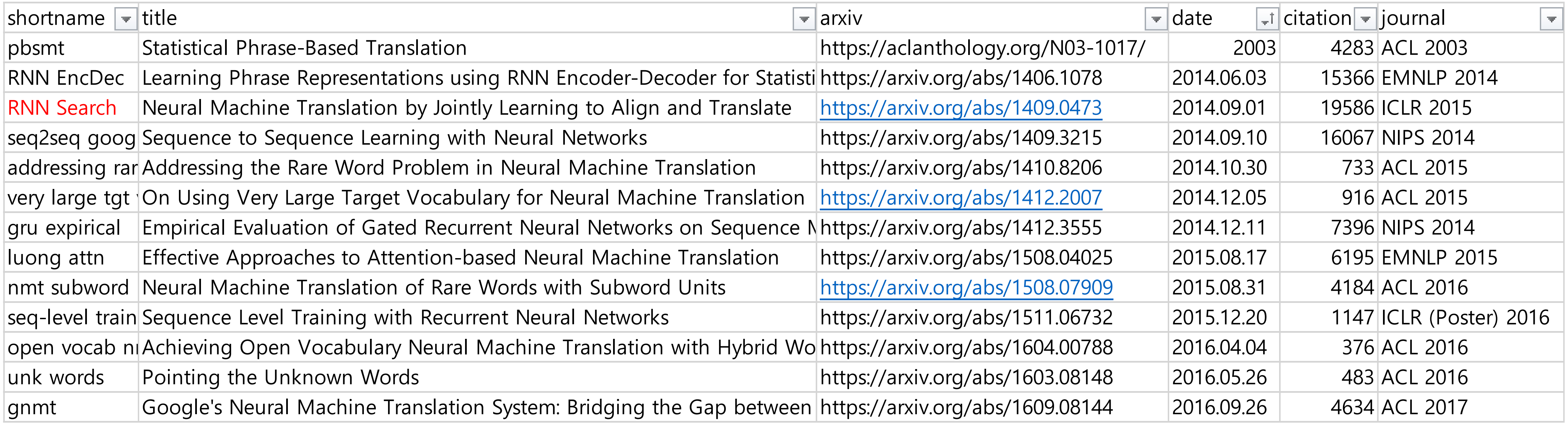
The strength of the soft-alignment, opposed to a hard-alignment, is evident, for instance, from Fig. 3 (d). Consider the source phrase [the man] which was translated into [l' homme]. Any hard alignment will map [the] to [l'] and [man] to [homme]. This is not helpful for translation, as one must consider the word following [the] to determine whether it should be translated into [le], [la], [les] or [l']. Our soft-alignment solves this issue naturally by letting the model look at both [the] and [man], and in this example, we see that the model was able to correctly translate [the] into [l']. We observe similar behaviors in all the presented cases in Fig. 3. An additional benefit of the soft alignment is that it naturally deals with source and target phrases of different lengths, without requiring a counter-intuitive way of mapping some words to or from nowhere ([NULL]) (see, e.g., Chapters 4 and 5 of Koehn, 2010).

04. Is enough?

Nope! 아직도 PBSMT 대비 준수한 성능을 보일 뿐...

- 이 후 여러 논문에서 성능 향상을 위한 연구들 + Attention 기법에 대한 연구들이 등장
- 성능 향상 계열로는 rare word 처리, subword information, stacking, gru cell, large target vocabulary 등이 있었고
- Attention 연구 계열로는 soft deterministic attn, multi-hop attn, luong attn, hierarchical attn, embed encode attend and predict, intra-attention 등 다양합니다.
- GNMT 논문에서 잘 정리해서 읽어보시는 것을 권장드립니다!

- **하지만 끝판 왕은 Transformer!!! 내일 같이 볼거에요!**

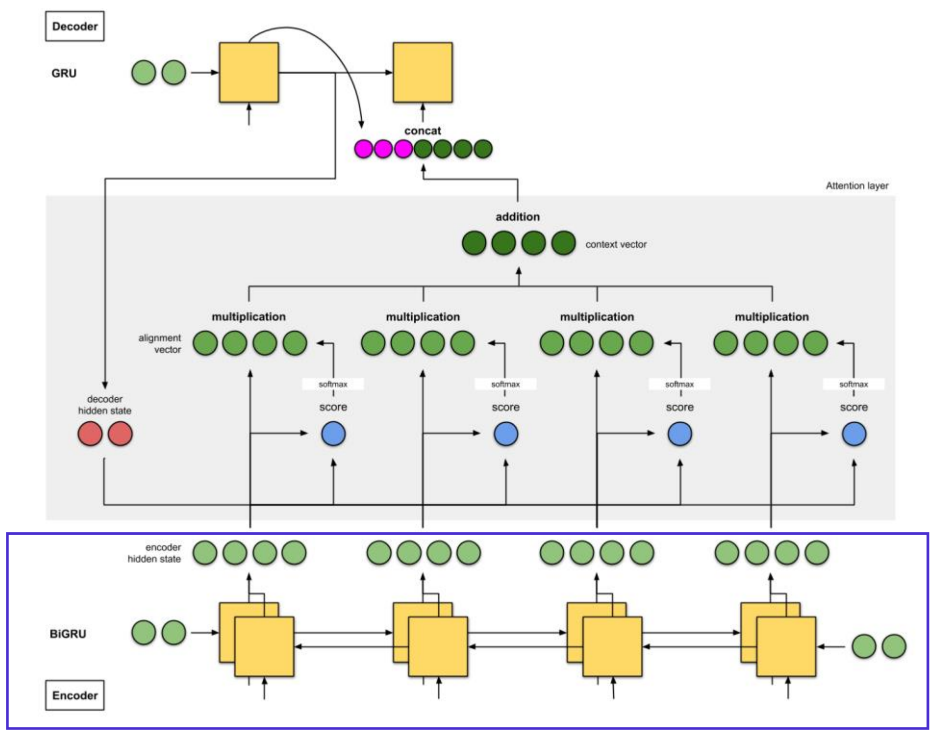


shortname	title	arxiv	date	citation	journal
pbsmt	Statistical Phrase-Based Translation	https://aclanthology.org/N03-1017/	2003	4283	ACL 2003
RNN EncDec	Learning Phrase Representations using RNN Encoder-Decoder for Statisti	https://arxiv.org/abs/1406.1078	2014.06.03	15366	EMNLP 2014
RNN Search	Neural Machine Translation by Jointly Learning to Align and Translate	https://arxiv.org/abs/1409.0473	2014.09.01	19586	ICLR 2015
seq2seq goog	Sequence to Sequence Learning with Neural Networks	https://arxiv.org/abs/1409.3215	2014.09.10	16067	NIPS 2014
addressing rar	Addressing the Rare Word Problem in Neural Machine Translation	https://arxiv.org/abs/1410.8206	2014.10.30	733	ACL 2015
very large tgt	On Using Very Large Target Vocabulary for Neural Machine Translation	https://arxiv.org/abs/1412.2007	2014.12.05	916	ACL 2015
gru expirical	Empirical Evaluation of Gated Recurrent Neural Networks on Sequence M	https://arxiv.org/abs/1412.3555	2014.12.11	7396	NIPS 2014
luong attn	Effective Approaches to Attention-based Neural Machine Translation	https://arxiv.org/abs/1508.04025	2015.08.17	6195	EMNLP 2015
nmt subword	Neural Machine Translation of Rare Words with Subword Units	https://arxiv.org/abs/1508.07909	2015.08.31	4184	ACL 2016
seq-level train	Sequence Level Training with Recurrent Neural Networks	https://arxiv.org/abs/1511.06732	2015.12.20	1147	ICLR (Poster) 2016
open vocab n	Achieving Open Vocabulary Neural Machine Translation with Hybrid Wo	https://arxiv.org/abs/1604.00788	2016.04.04	376	ACL 2016
unk words	Pointing the Unknown Words	https://arxiv.org/abs/1603.08148	2016.05.26	483	ACL 2016
gnmt	Google's Neural Machine Translation System: Bridging the Gap between	https://arxiv.org/abs/1609.08144	2016.09.26	4634	ACL 2017

https://devopedia.org/attention-mechanism-in-neural-networks

04. Code Review

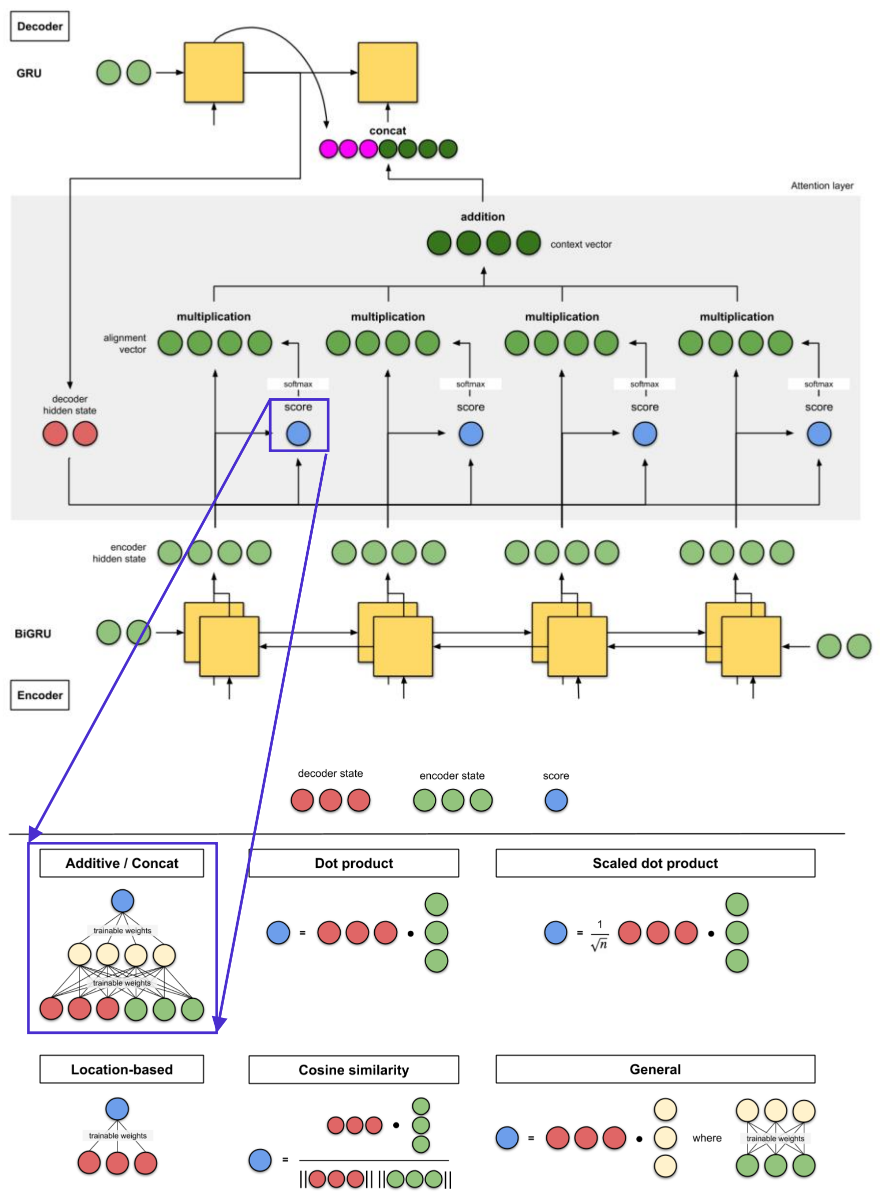
```python
import torch
import torch.nn as nn
from typing import Tuple


class Encoder(nn.Module):
    """
    seq2seq encoder module for bahdanau attention
    https://github.com/bentrevett/pytorch-seq2seq
    """

    def __init__(self, input_dim: int, emb_dim: int,
                 enc_hid_dim: int, dec_hid_dim: int, dropout: float):
        super().__init__()
        self.embedding = nn.Embedding(input_dim, emb_dim)
        self.rnn = nn.GRU(emb_dim, enc_hid_dim, bidirectional = True)
        self.fc = nn.Linear(enc_hid_dim * 2, dec_hid_dim)
        self.dropout = nn.Dropout(dropout)

    def forward(self, src: torch.Tensor) -> Tuple[torch.Tensor]:
        # src = [src len, batch size]
        embedded = self.dropout(self.embedding(src))
        # embedded = [src len, batch size, emb dim]
        outputs, hidden = self.rnn(embedded)
        # outputs = [src len, batch size, hid dim * num directions]
        # hidden = [n layers * num directions, batch size, hid dim]
        # hidden is stacked [forward_1, backward_1, forward_2, backward_2, ...]
        # outputs are always from the last layer
        # hidden [-2, :, : ] is the last of the forwards RNN
        # hidden [-1, :, : ] is the last of the backwards RNN
        # initial decoder hidden is final hidden state of the forwards and backwards
        #   encoder RNNs fed through a linear layer
        hidden = torch.tanh(self.fc(torch.cat((hidden[-2,:,:], hidden[-1,:,:]), dim=1)))
        # outputs = [src len, batch size, enc hid dim * 2]
        # hidden = [batch size, dec hid dim]
        return outputs, hidden
```



https://github.com/bentrevett/pytorch-seq2seq

04. Code Review

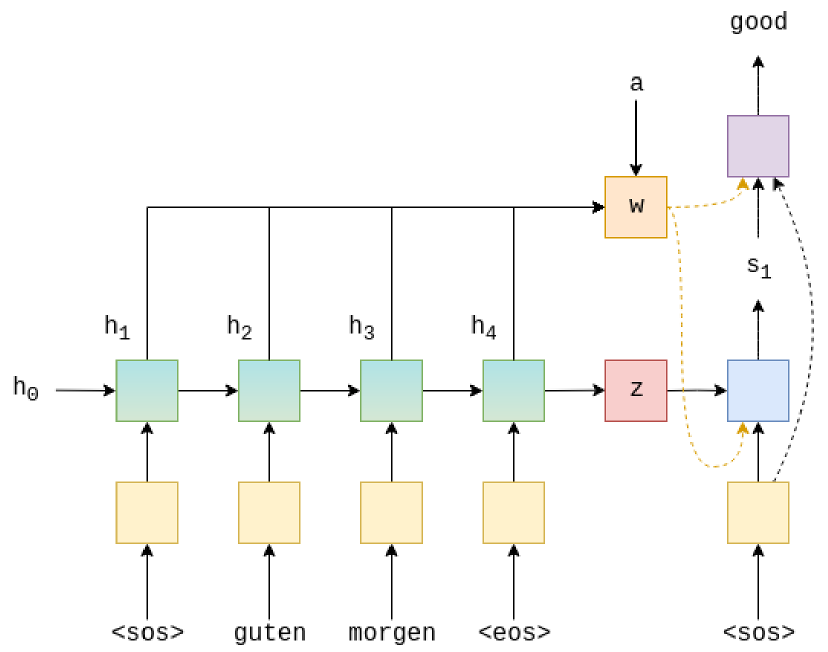
```python
class Attention(nn.Module):
    """
    Bahdanau Attention Module
    https://github.com/bentrevett/pytorch-seq2seq
    """

    def __init__(self, enc_hid_dim: int, dec_hid_dim: int):
        super().__init__()
        self.attn = nn.Linear((enc_hid_dim * 2) + dec_hid_dim, dec_hid_dim)
        self.v = nn.Linear(dec_hid_dim, 1, bias=False)

    def forward(self, hidden: torch.Tensor, encoder_outputs: torch.Tensor):
        # hidden = [batch size, dec hid dim]
        # encoder_outputs = [src len, batch size, enc hid dim * 2]
        batch_size = encoder_outputs.shape[1]
        src_len = encoder_outputs.shape[0]

        # repeat decoder hidden state src_len times
        # hidden = [batch size, src len, dec hid dim]
        # encoder_outputs = [batch size, src len, enc hid dim * 2]
        hidden = hidden.unsqueeze(1).repeat(1, src_len, 1)
        encoder_outputs = encoder_outputs.permute(1, 0, 2)

        # energy = [batch size, src len, dec hid dim]
        energy = torch.tanh(self.attn(torch.cat((hidden, encoder_outputs), dim = 2)))

        # attention= [batch size, src len]
        attention = self.v(energy).squeeze(2)
        return torch.softmax(attention, dim=1)
```



https://github.com/bentrevett/pytorch-seq2seq

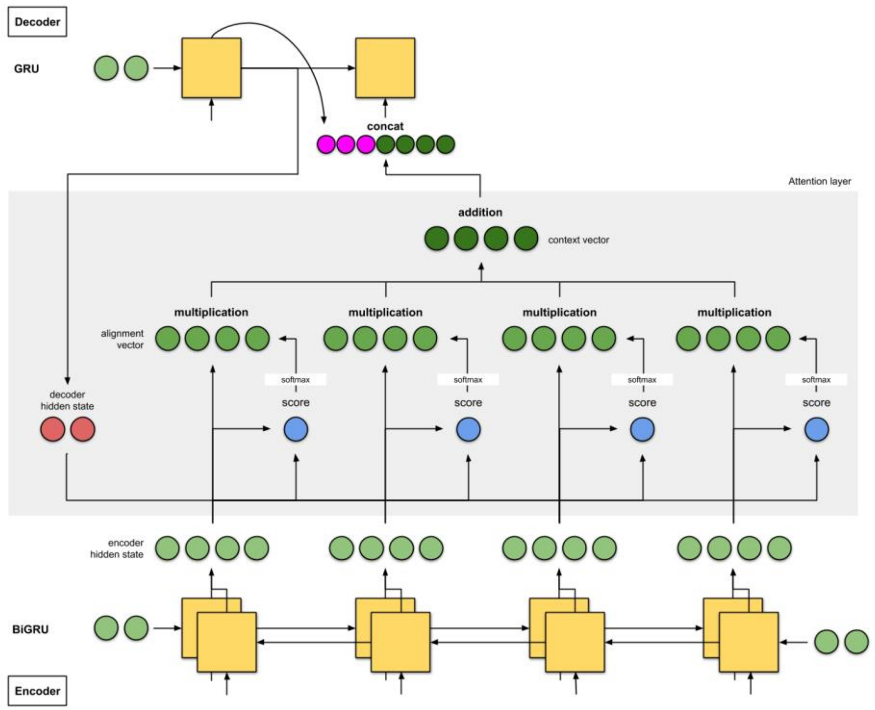
04. Code Review

```python
class Decoder(nn.Module):
    """
    seq2seq decoder class for bahdanau attention
    https://github.com/bentrevett/pytorch-seq2seq
    """

    def __init__(self, output_dim: int, emb_dim: int,
                 enc_hid_dim: int, dec_hid_dim: int, dropout: float, attention: nn.Module):
        super().__init__()
        self.output_dim = output_dim
        self.attention = attention
        self.embedding = nn.Embedding(output_dim, emb_dim)
        self.rnn = nn.GRU((enc_hid_dim * 2) + emb_dim, dec_hid_dim)
        self.fc_out = nn.Linear((enc_hid_dim * 2) + dec_hid_dim + emb_dim, output_dim)
        self.dropout = nn.Dropout(dropout)

    def forward(self, input, hidden, encoder_outputs):
        # input = [batch size]
        # hidden = [batch size, dec hid dim]
        # encoder_outputs = [src len, batch size, enc hid dim * 2]

        # input = [1, batch size]
        input = input.unsqueeze(0)

        # embedded = [1, batch size, emb dim]
        embedded = self.dropout(self.embedding(input))

        # a = [batch size, 1, src len]
        a = self.attention(hidden, encoder_outputs)
        a = a.unsqueeze(1)

        # encoder_outputs = [batch size, src len, enc hid dim * 2]
        encoder_outputs = encoder_outputs.permute(1, 0, 2)

        # weighted = [batch size, 1, enc hid dim * 2]
        weighted = torch.bmm(a, encoder_outputs)
        # weighted = [1, batch size, enc hid dim * 2]
        weighted = weighted.permute(1, 0, 2)

        # rnn_input = [1, batch size, (enc hid dim * 2) + emb dim]
        rnn_input = torch.cat((embedded, weighted), dim = 2)

        # output = [seq len, batch size, dec hid dim * n directions]
        # hidden = [n layers * n directions, batch size, dec hid dim]
        output, hidden = self.rnn(rnn_input, hidden.unsqueeze(0))

        # seq len, n layers and n directions will always be 1 in this decoder, therefore:
        # output = [1, batch size, dec hid dim]
        # hidden = [1, batch size, dec hid dim]
        # this also means that output == hidden
        assert (output == hidden).all()

        embedded = embedded.squeeze(0)
        output = output.squeeze(0)
        weighted = weighted.squeeze(0)

        # prediction = [batch size, output dim]
        prediction = self.fc_out(torch.cat((output, weighted, embedded), dim = 1))
        return prediction, hidden.squeeze(0)
```



https://github.com/bentrevett/pytorch-seq2seq

04. Code Review



```python
class Seq2Seq(nn.Module):

    def __init__(self, encoder: nn.Module, decoder: nn.Module, device: str):
        super().__init__()

        self.encoder = encoder
        self.decoder = decoder
        self.device = device

    def forward(
        self,
        src: torch.Tensor,
        trg: torch.Tensor,
        teacher_forcing_ratio: float = 0.5
    ) -> torch.Tensor:
        # src = [src len, batch size]
        # trg = [trg len, batch size]
        # teacher_forcing_ratio is probability to use teacher forcing
        # e.g. if teacher_forcing_ratio is 0.75 we use teacher forcing 75% of the time

        batch_size = src.shape[1]
        trg_len = trg.shape[0]
        trg_vocab_size = self.decoder.output_dim

        # tensor to store decoder outputs
        outputs = torch.zeros(trg_len, batch_size, trg_vocab_size).to(self.device)

        # encoder_outputs is all hidden states of the input sequence, back and forwards
        # hidden is the final forward and backward hidden states, passed through a linear layer
        encoder_outputs, hidden = self.encoder(src)

        # first input to the decoder is the <sos> tokens
        input = trg[0,:]

        for t in range(1, trg_len):
            # insert input token embedding, previous hidden state and all encoder hidden states
            # receive output tensor (predictions) and new hidden state
            output, hidden = self.decoder(input, hidden, encoder_outputs)

            # place predictions in a tensor holding predictions for each token
            outputs[t] = output

            # decide if we are going to use teacher forcing or not
            teacher_force = random.random() < teacher_forcing_ratio

            # get the highest predicted token from our predictions
            top1 = output.argmax(1)

            # if teacher forcing, use actual next token as next input
            # if not, use predicted token
            input = trg[t] if teacher_force else top1
        return outputs
```

https://github.com/bentrevett/pytorch-seq2seq

부스트캠프 AI Tech 2기

boostcamp ai tech

Discussion

Email : jinmang2@gmail.com

GitHub : github.com/jinmang2

Huggingface Hub: huggingface.co/jinmang2

진명훈_T2216