Seq2seq

한양대학교 AILAB 박소영

01. Translator

번역기를 만들어보자

나는 널 사랑해

I love you

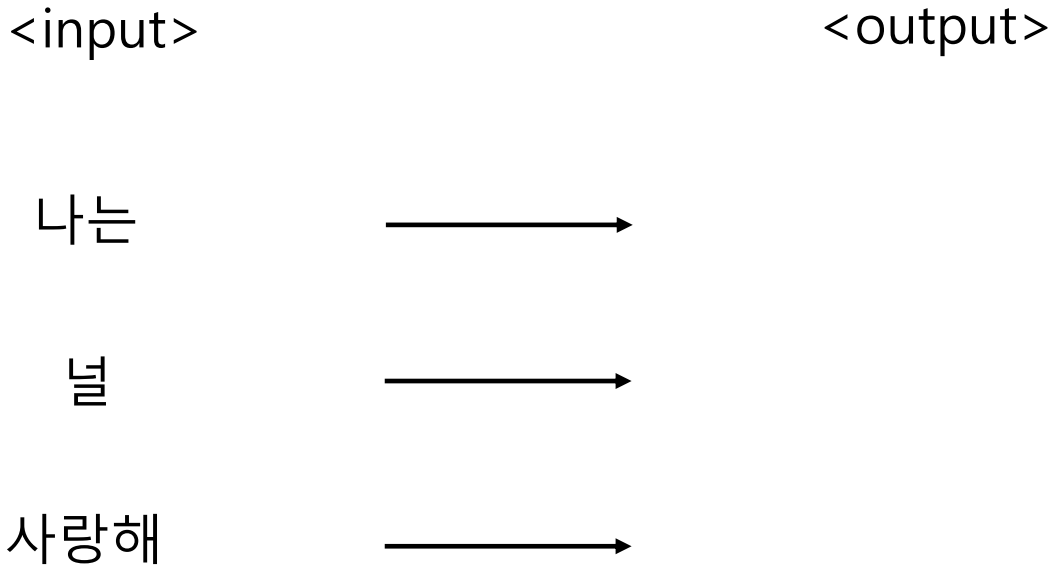
我愛你

愛してる

je t'aime

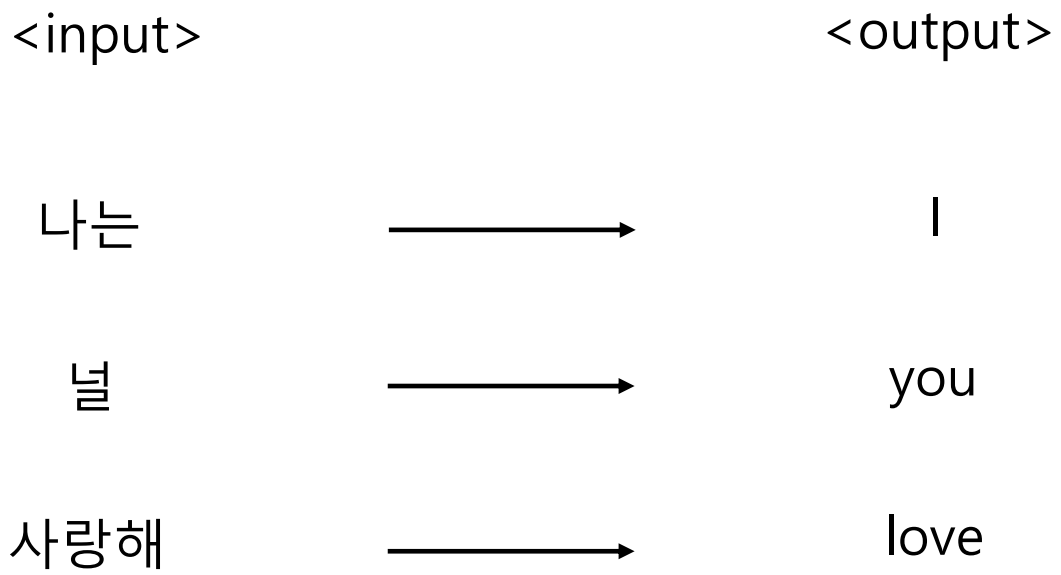
01. Translator

번역기를 만들어보자



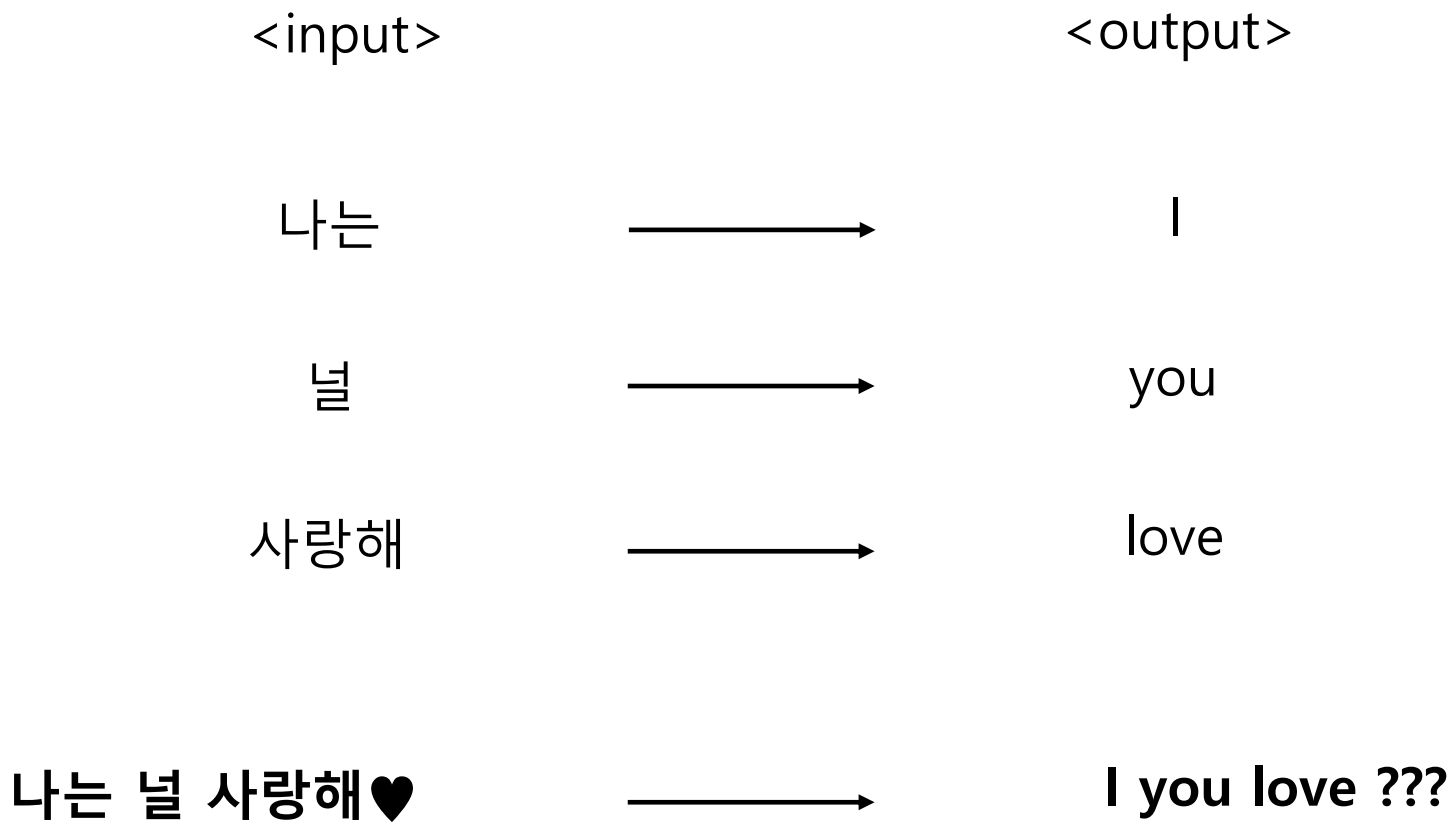
01. Translator

번역기를 만들어보자



01. Translator

번역기를 만들어보자

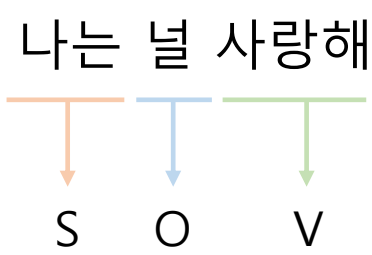


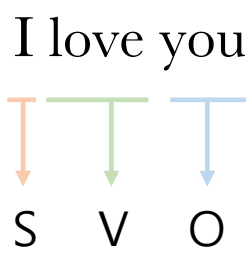
01. Translator

번역기를 만들어 보자 – 문제01

- 한국어, 일본어 : 주어 목적어 서술어 순으로 구성 (S-O-V)
- 영어, 중국어 : 주어 서술어 목적어로 구성 (S-V-O)

단어 별로 번역을 하면 **어순이 엇갈리는 문제**가 생긴다

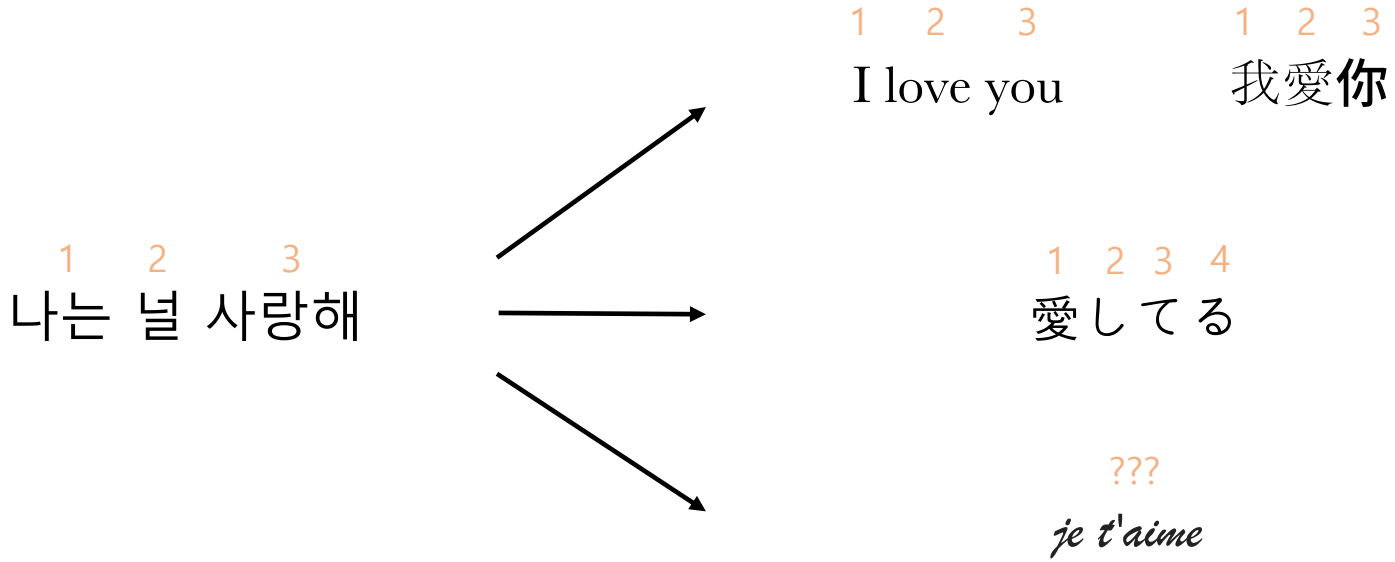




01. Translator

번역기를 만들어 보자 – 문제02

단어가 1:1로 수가 맞아 번역이 되는 경우가 이는 한편 **단어의 수가 바뀔 수도 있다**



02. seq2seq

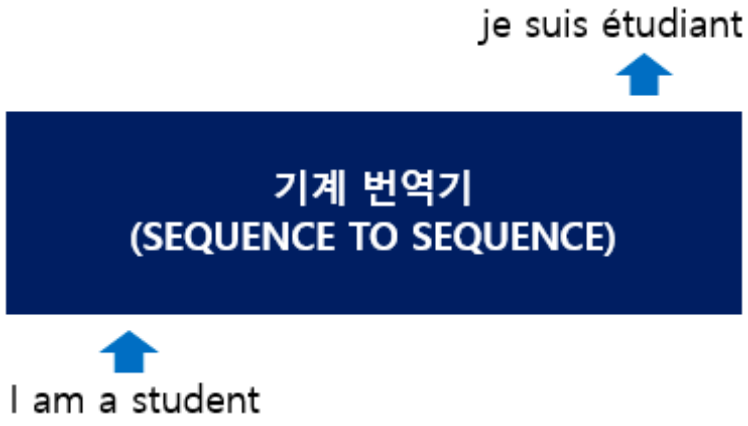
Sequence-to-sequence

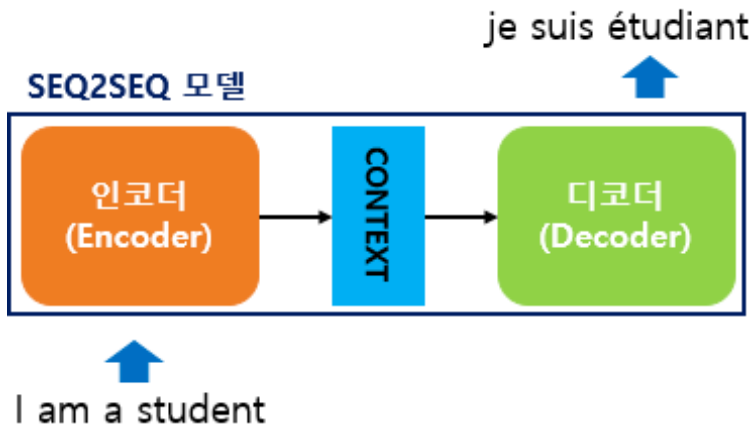
- 불어-영어 번역을 위해 2014년 처음 사용되었다
- **한 domain에서 다른 domain으로 시퀀스를 변환**하는 모델 학습

  ex) chatbot, machine translation, text summarization, STT, ...

  chatbot : 입력(질문), 출력(대답)

  machine translation : 입력(문장), 출력(번역 문장)

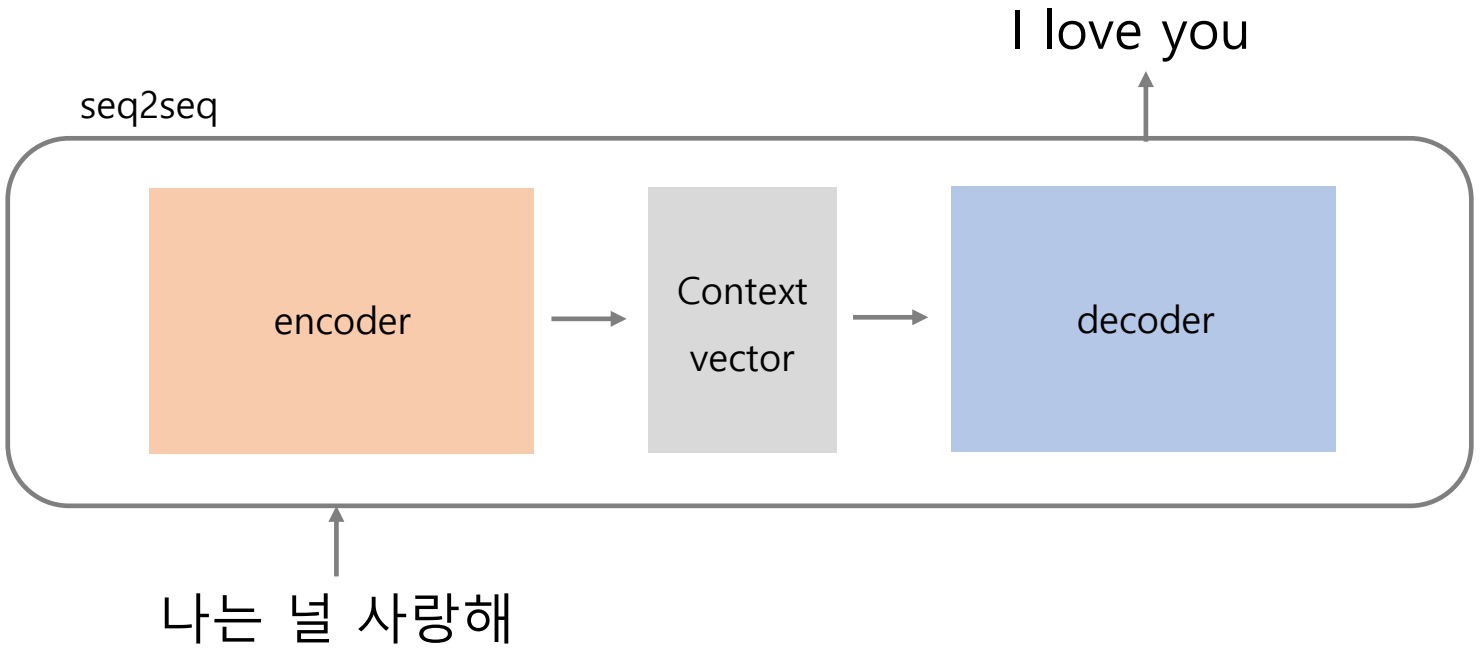




03. architecture

Seq2seq의 구조를 알아보자

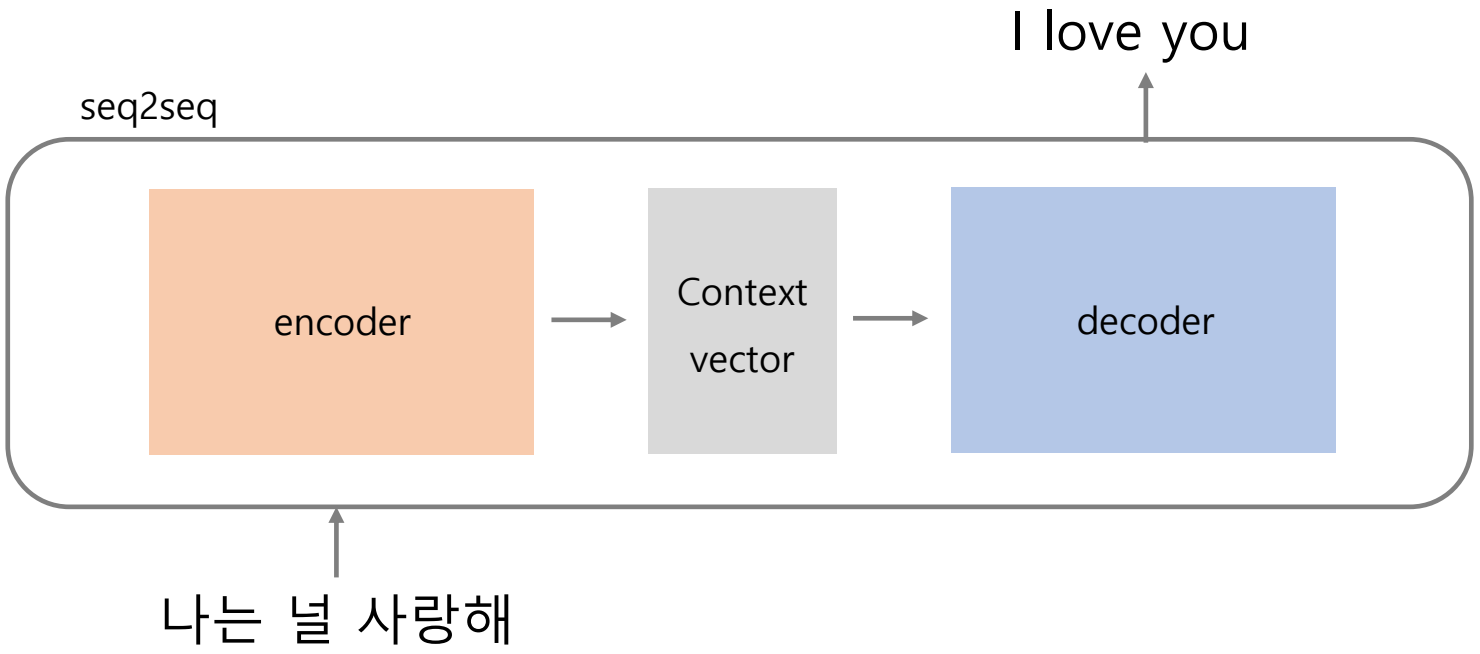
- **Encoder**
  - 입력 단어를 순차적으로 받음으로써 context vector 생성
  - 입력 문장의 정보가 하나의 context vector로 압축되면 decoder로 전송
- **Decoder**
  - context vector를 받아서 번역된 단어를 한 개씩 순차적으로 출력



03. architecture

Seq2seq의 구조를 알아보자

- **Context vector**
  - Encoder에서의 input에 대한 정보를 함축
- 입력 시퀀스, 출력 시퀀스의 단어 순서가 달라도 학습을 함으로써 **문법, 문맥 고려 가능**



03. architecture

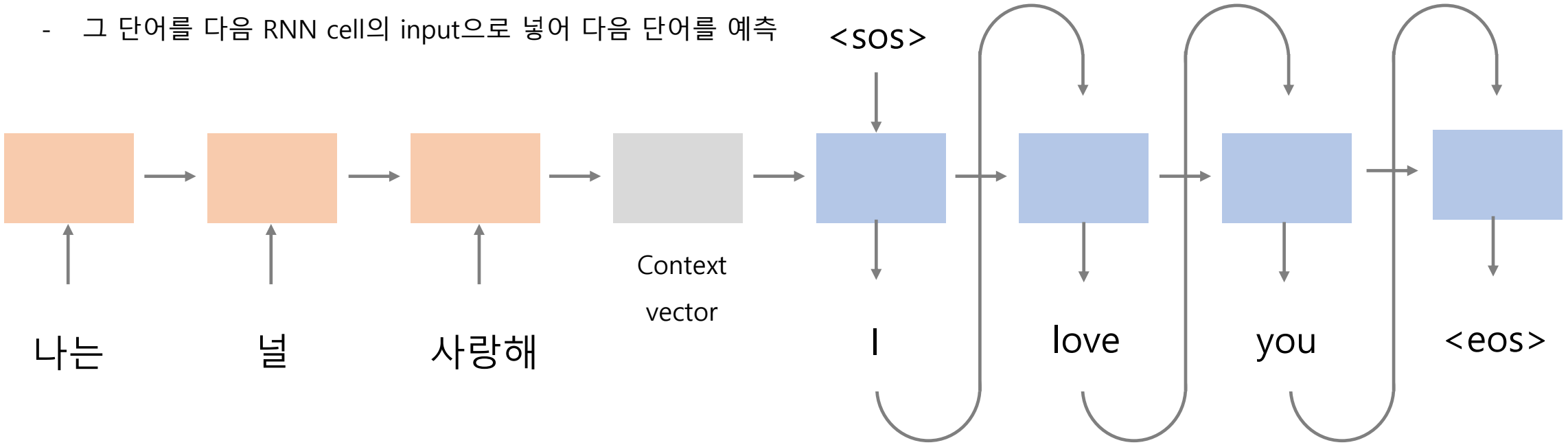
Seq2seq의 Train, test 때의 작동 방식

Train

- context vector, GT <sos> I love you 두 개를 받았을 때 I love you <eos>가 와야 한다고 정답을 알려주면서 학습

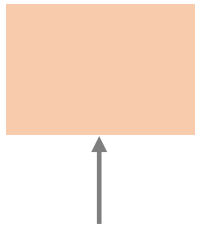
Test

- Decoder는 context vector, <sos>만을 input으로 받고 다음에 올 단어를 예측.
- 그 단어를 다음 RNN cell의 input으로 넣어 다음 단어를 예측



03. architecture

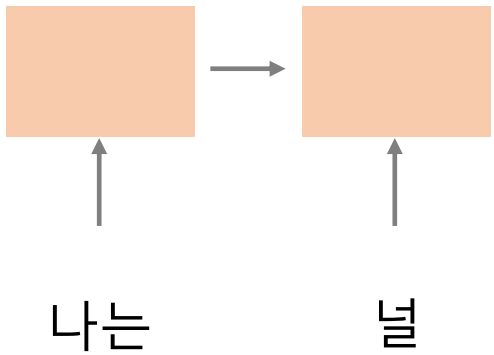
Seq2seq의



나는

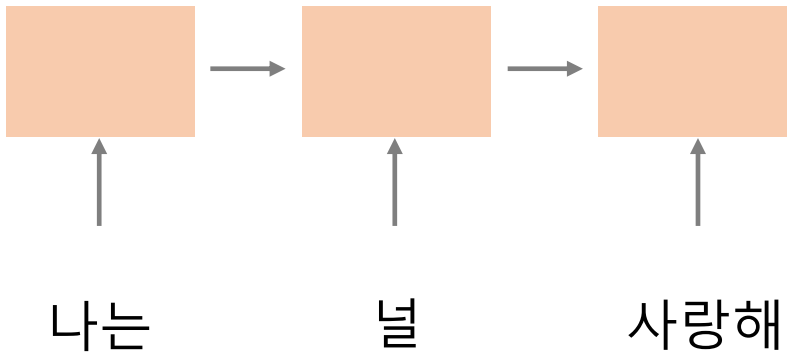
03. architecture

Seq2seq의 구조를 알아보자



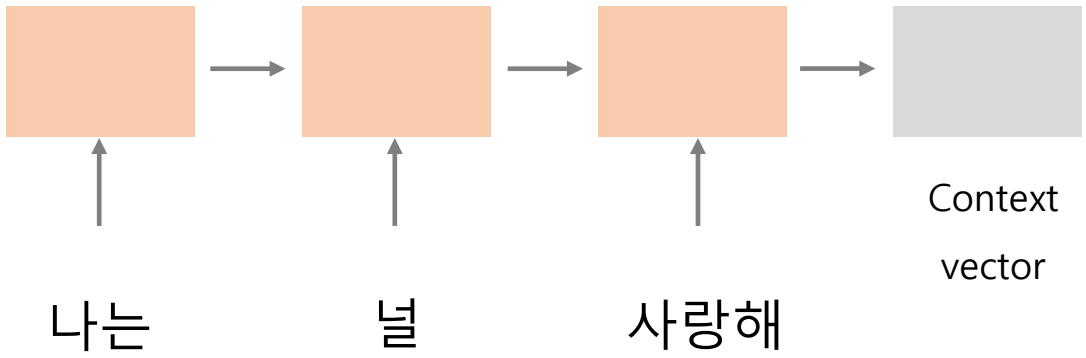
03. architecture

Seq2seq의 구조를 알아보자



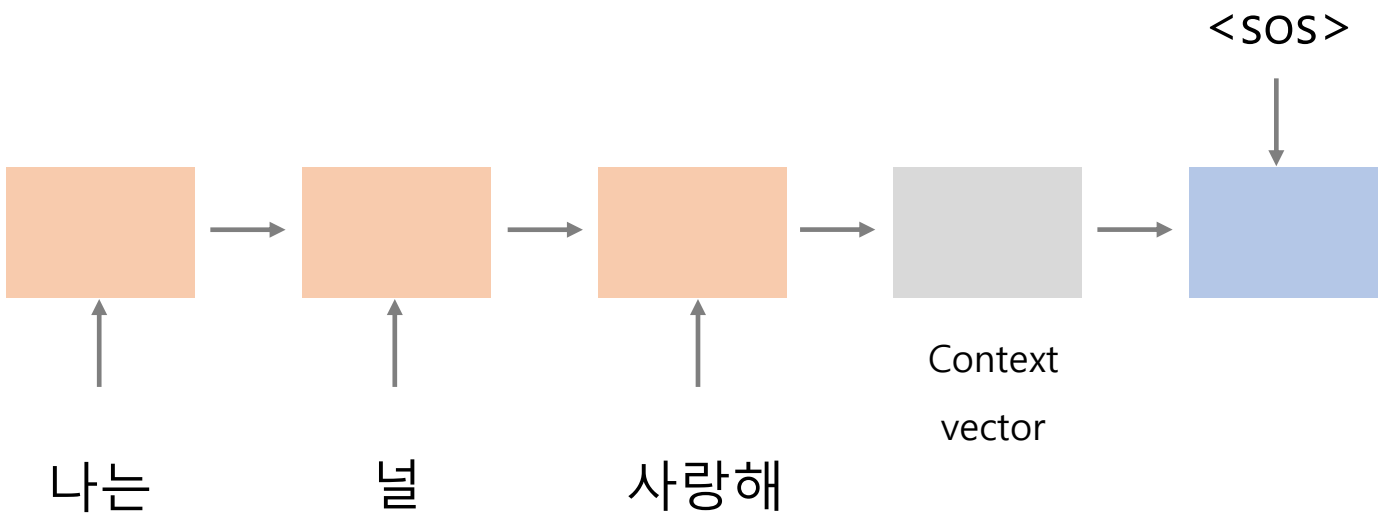
03. architecture

Seq2seq의 구조를 알아보자



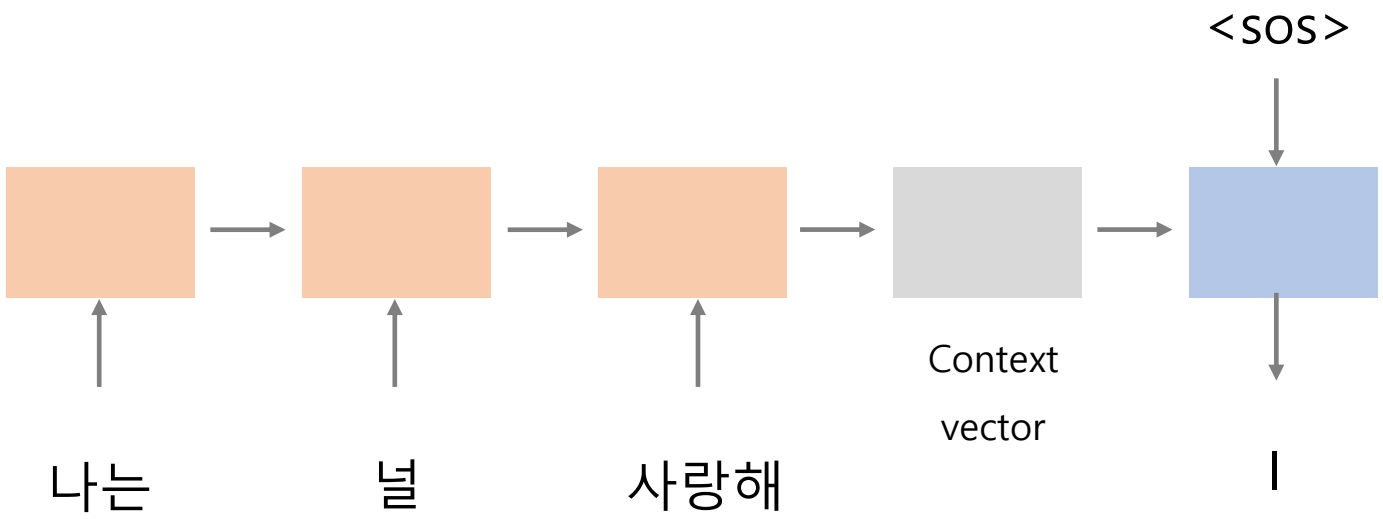
03. architecture

Seq2seq의 구조를 알아보자



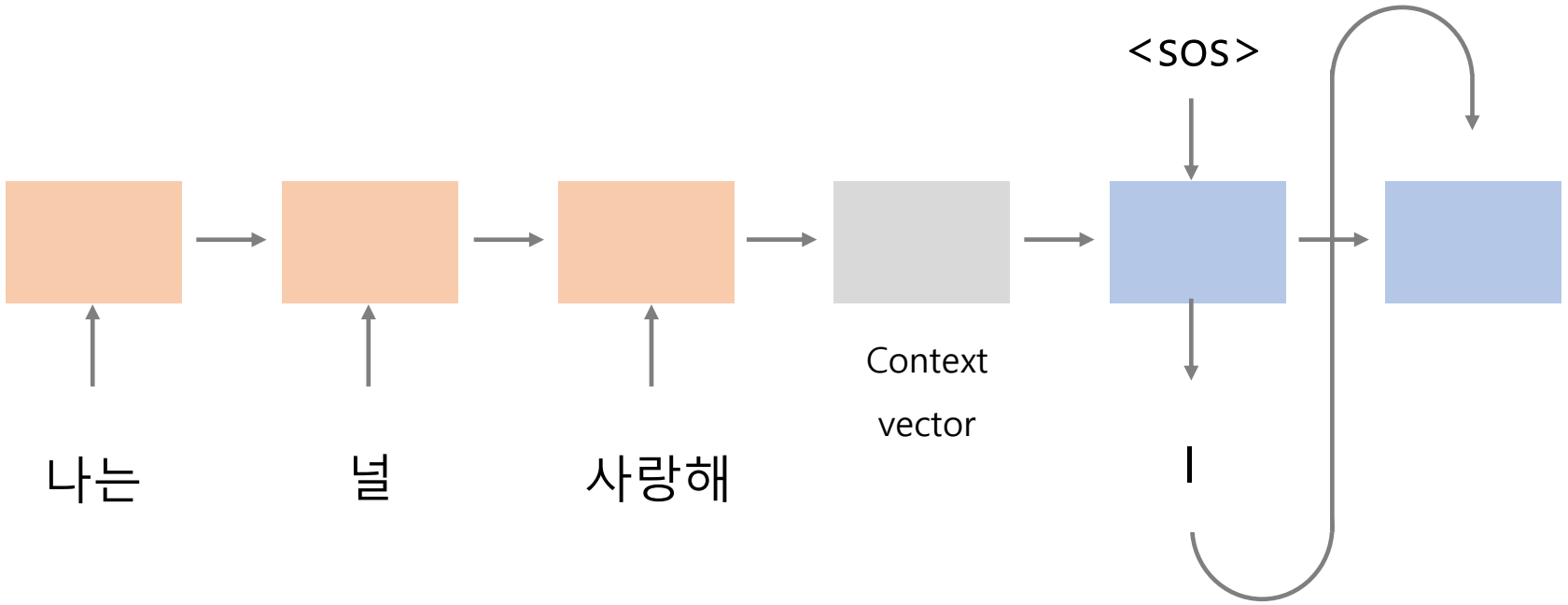
03. architecture

Seq2seq의 구조를 알아보자



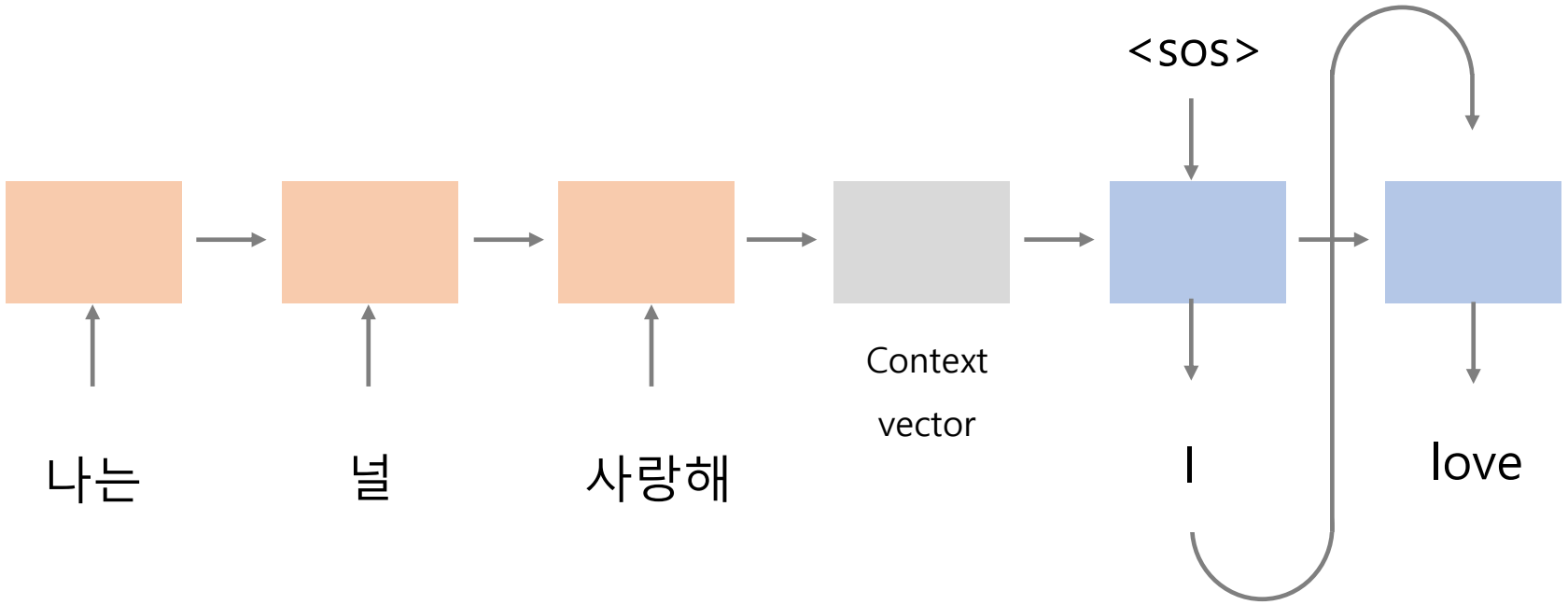
03. architecture

Seq2seq의 구조를 알아보자



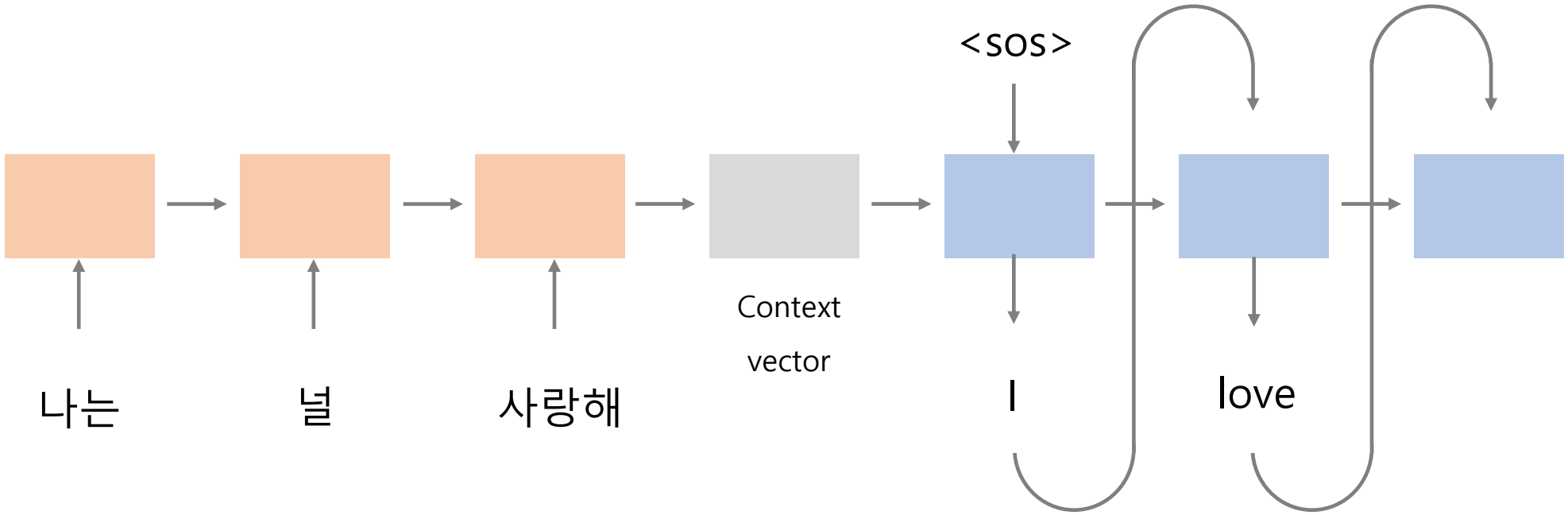
03. architecture

Seq2seq의 구조를 알아보자



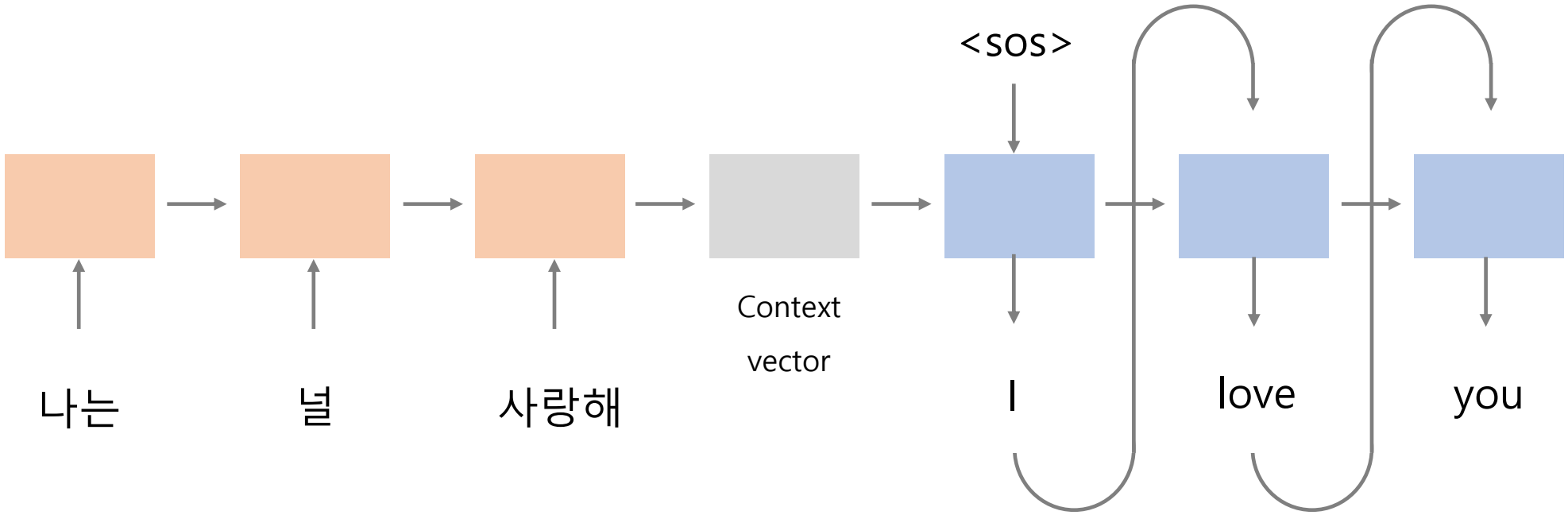
03. architecture

Seq2seq의 구조를 알아보자



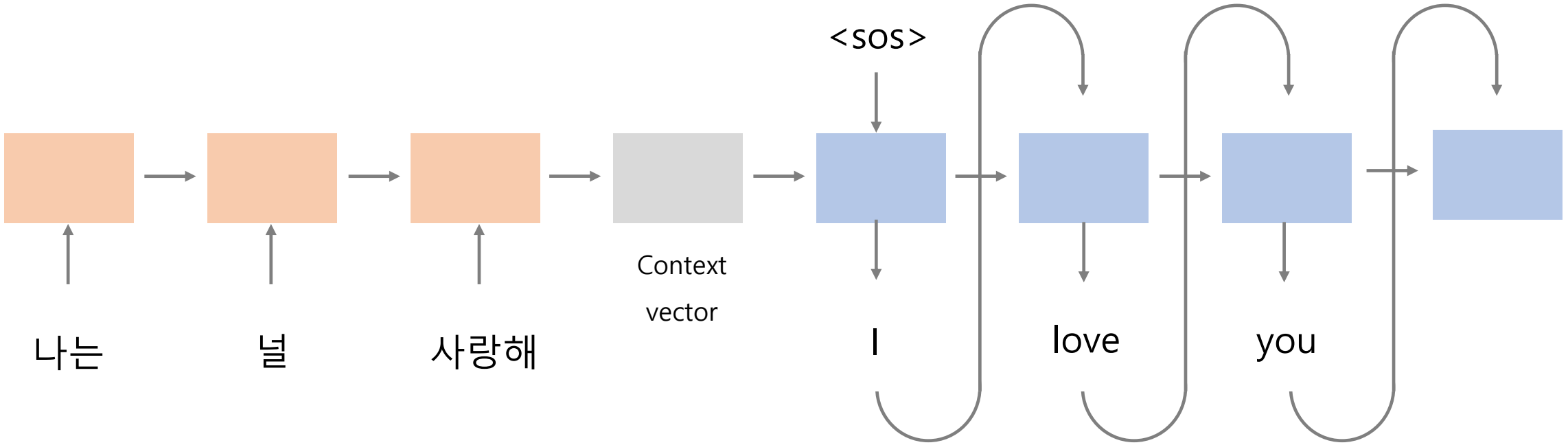
03. architecture

Seq2seq의 구조를 알아보자



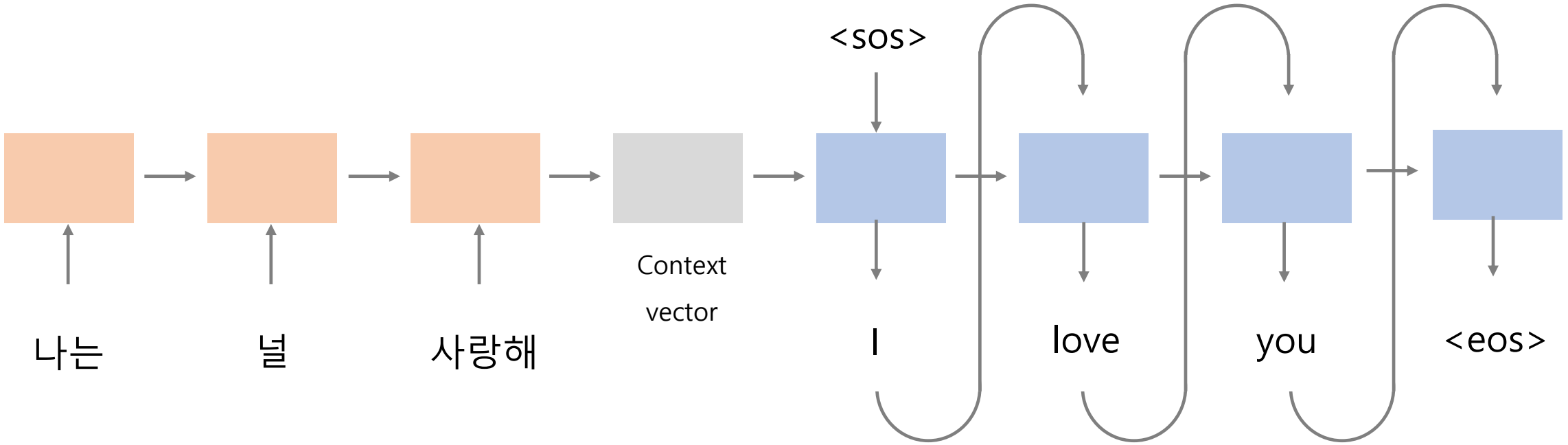
03. architecture

Seq2seq의 구조를 알아보자



03. architecture

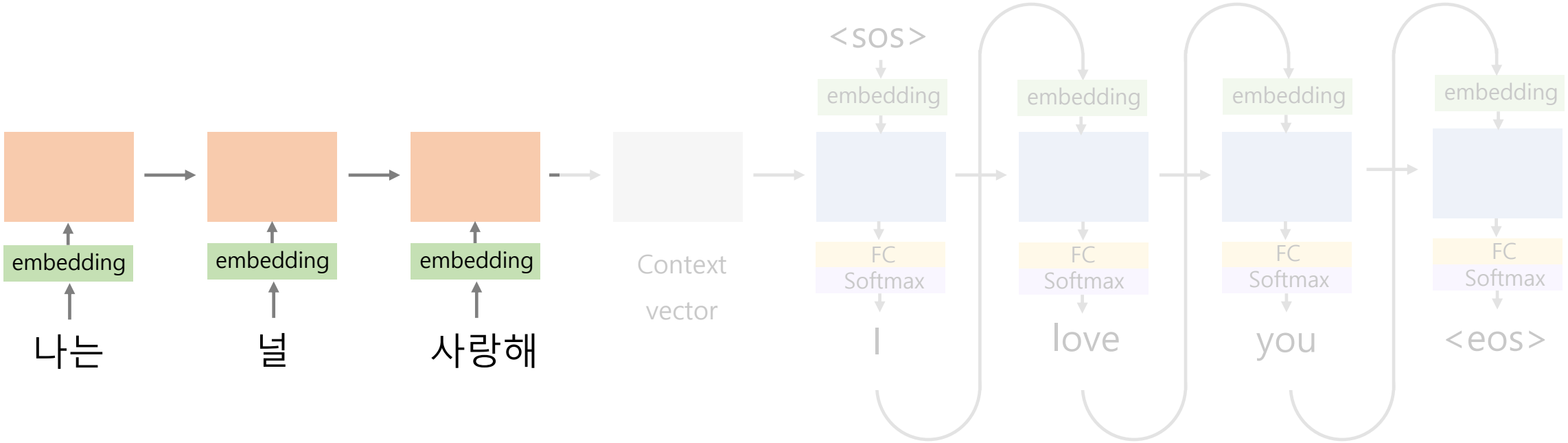
Seq2seq의 구조를 알아보자



04. Layers

Seq2seq의 embedding, dense, softmax layers

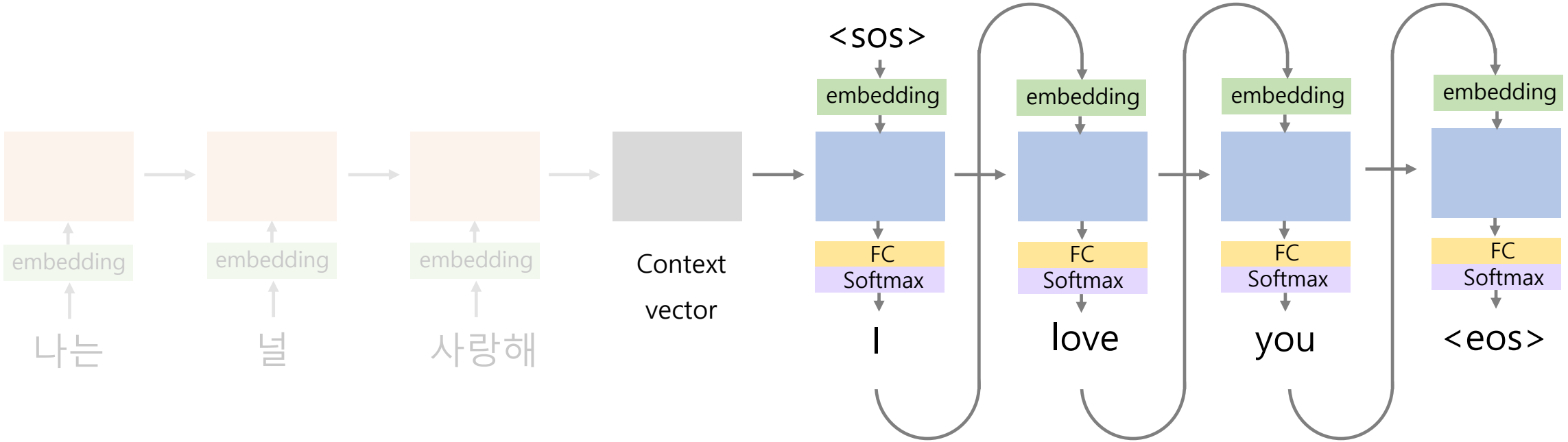
Seq2seq에서 사용되는 모든 단어들은 word embedding을 통해 **embedding vector**로 표현된다



04. Layers

Seq2seq의 embedding, fc, softmax layers

- RNN cell에서 어떠한 '단어'가 나올지 선택
- 각 시점에서 RNN cell으로부터 어떠한 output이 나오면 softmax를 통해 각 단어별 확률 값을 반환

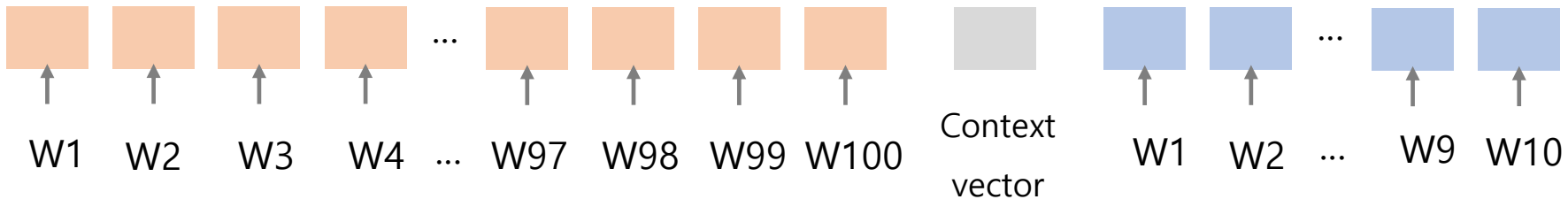


05. drawback

단순 seq2seq의 단점

Context vector의 사이즈는 input과 상관없이 고정

만약 문장이 길어질 경우 **context vector가 충분히 크지 않으면** 모든 내용을 함축할 만큼을 담을 수 없다

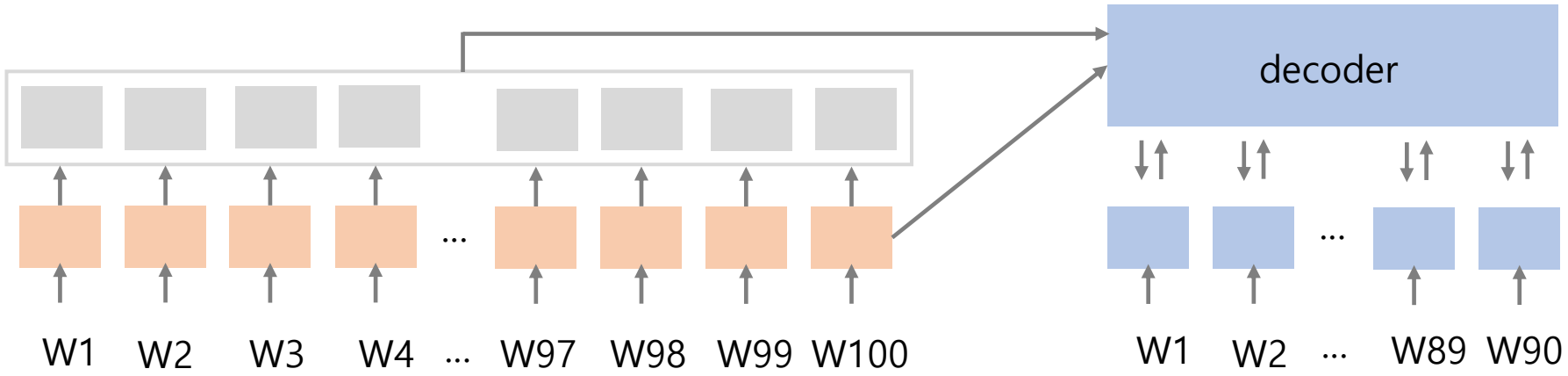


06. attention

Attention seq2seq

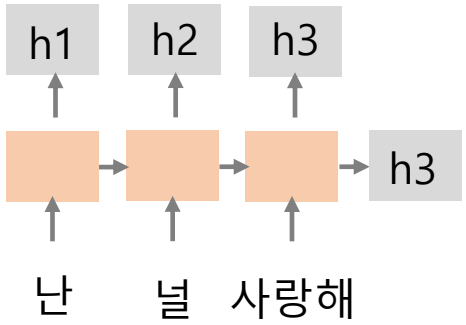
Attention 사용해 해결

- 기존엔 context vector만 사용 -> Encoder의 모든 state를 사용하면 context vector의 길이 제약에서 벗어날 수 있다
  - 각각의 state vector를 가지고 새로운 context vector 생성
  - encoder에 있었던 **모든 state 중에서 핵심 단어에만 집중하는 매커니즘**



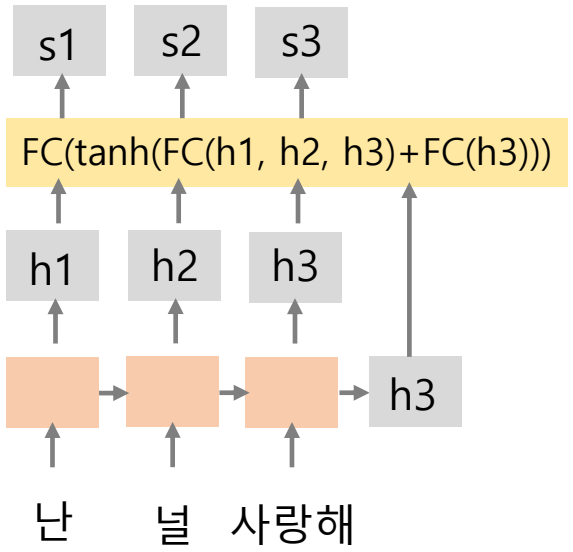
06. attention

Attention seq2seq



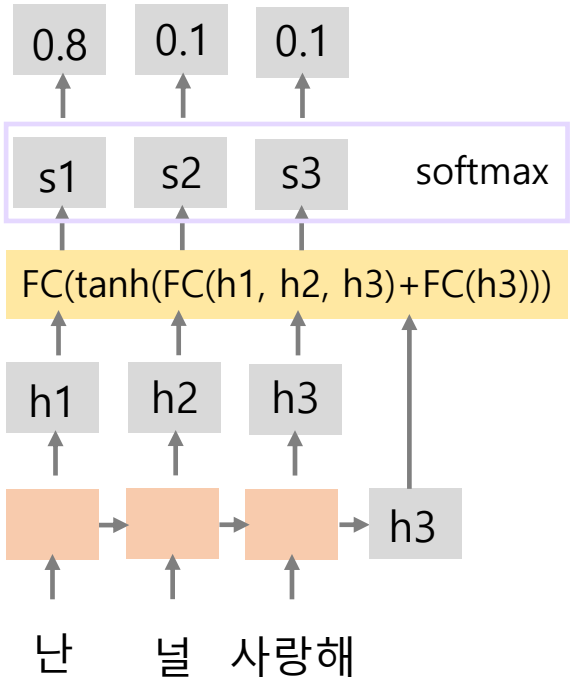
06. attention

Attention seq2seq



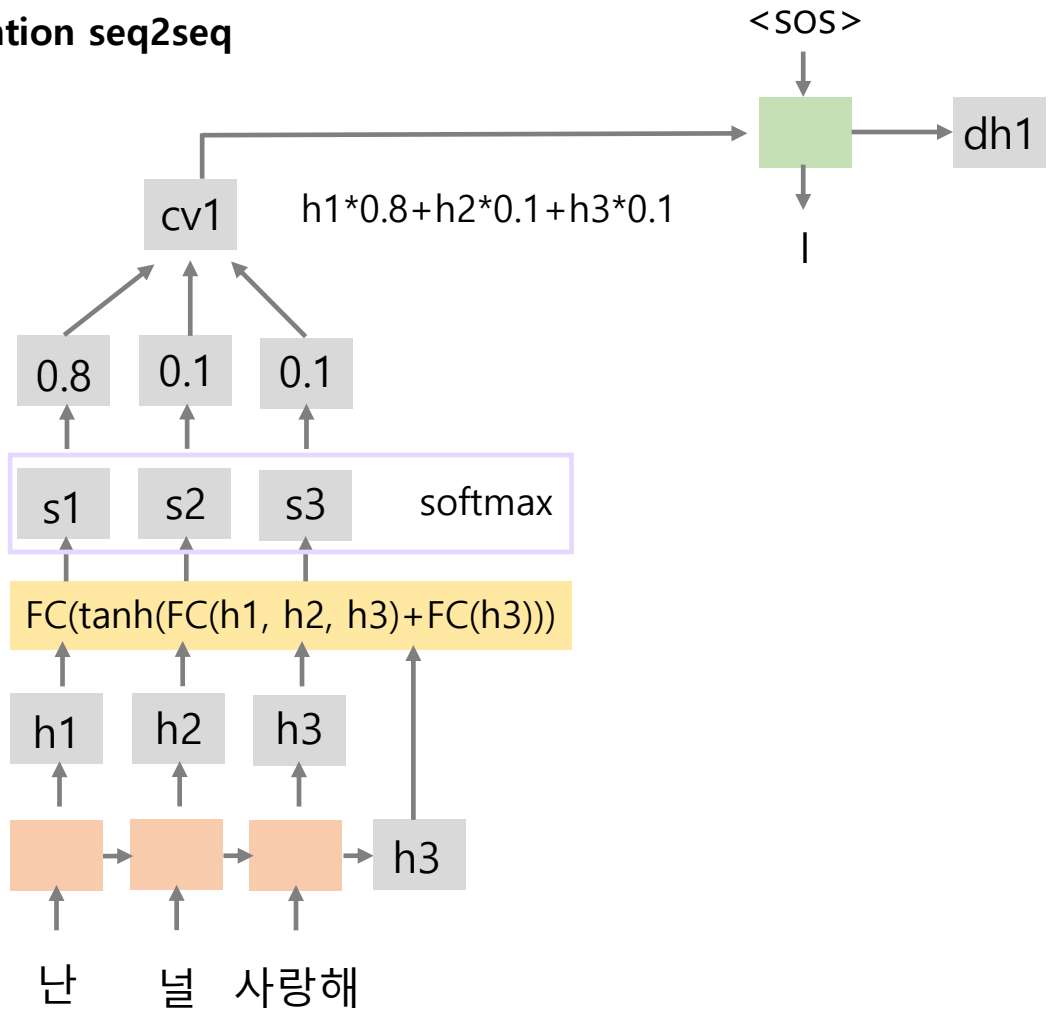
06. attention

Attention seq2seq



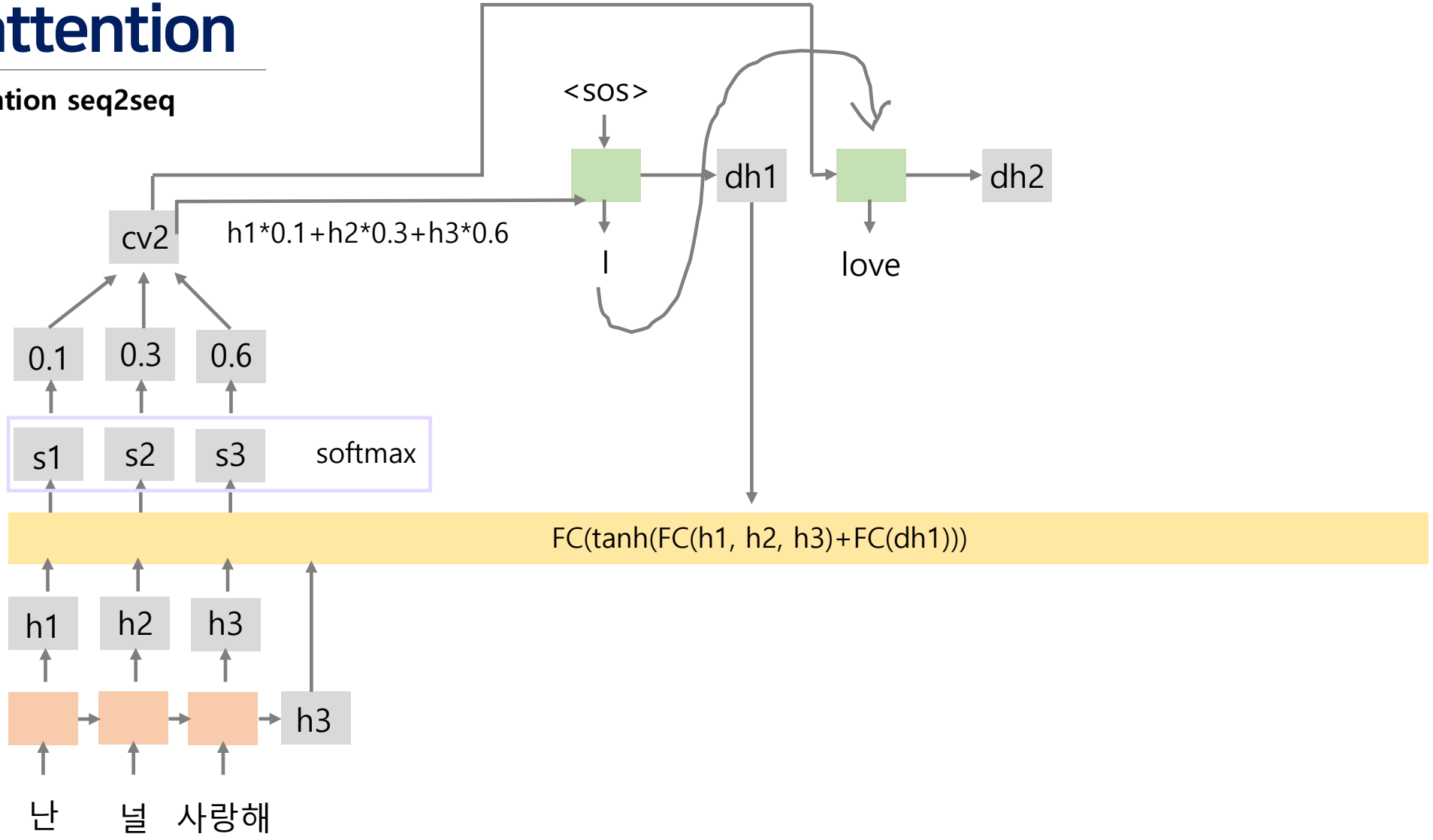
06. attention

Attention seq2seq



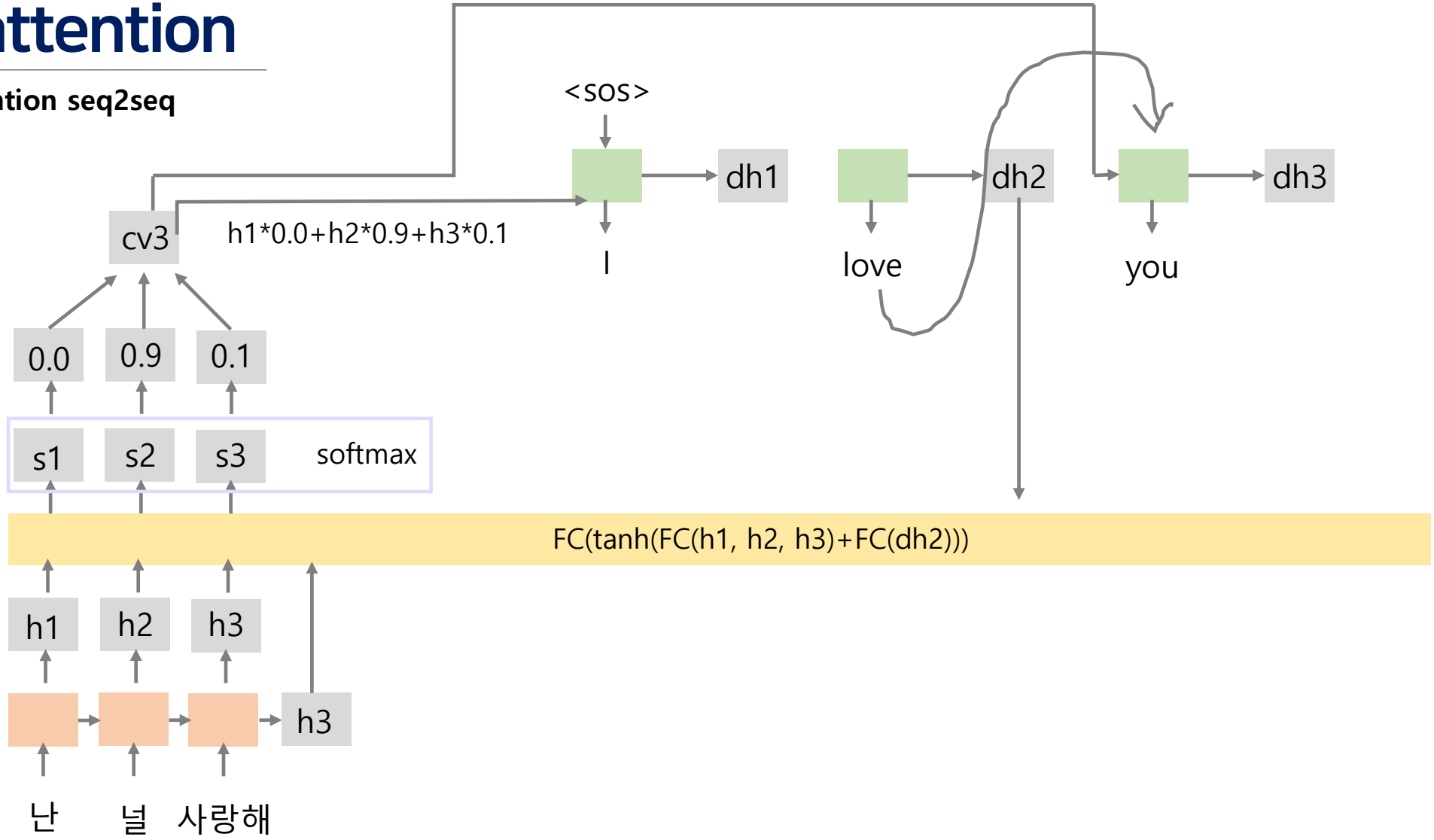
06. attention

Attention seq2seq



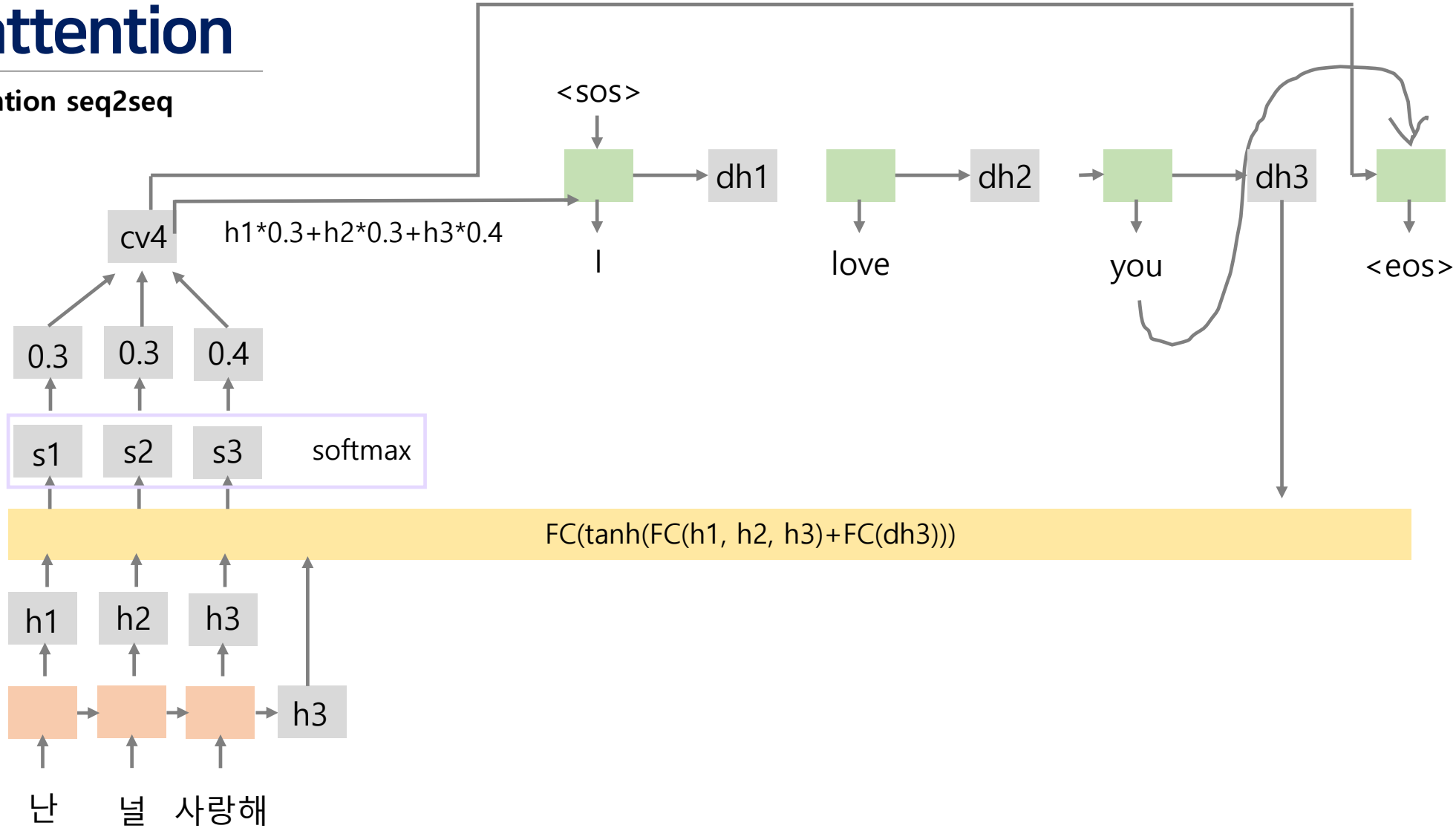
06. attention

Attention seq2seq



06. attention

Attention seq2seq



07. etc

구현 방법

- Context vector를 decoder의 초기 hidden state로만 사용
- Context vector를 decoder가 단어를 예측하는 매 시점마다 하나의 input으로 사용
- Attention mechanism을 통해 새로운 context vector를 구하여 매 시점마다 하나의 input으로 사용