Deep Learning Language Models

정윤경
성균관대학교

Outline

- CNN
- Seq2seq
- Attention

Reference

- Speech and Language Processing (3rd ed. draft). Dan Jurafsky and James H. Martin. Ch. 9: Sequence Processing with Recurrent Networks (https://web.stanford.edu/~jurafsky/slp3/9.pdf) and Encoder-Decoder Models, Attention, and Contextual Embeddings(https://web.stanford.edu/~jurafsky/slp3/10.pdf)
- 밑바닥부터 시작하는 딥러닝 2
- NLP in Action
- 딥 러닝을 이용한 자연어 처리 입문 (https://wikidocs.net/book/2155)

CNN (convolutional neural network) for NLP

EMNLP 2014

- Use CNN instead of RNN

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

Abstract

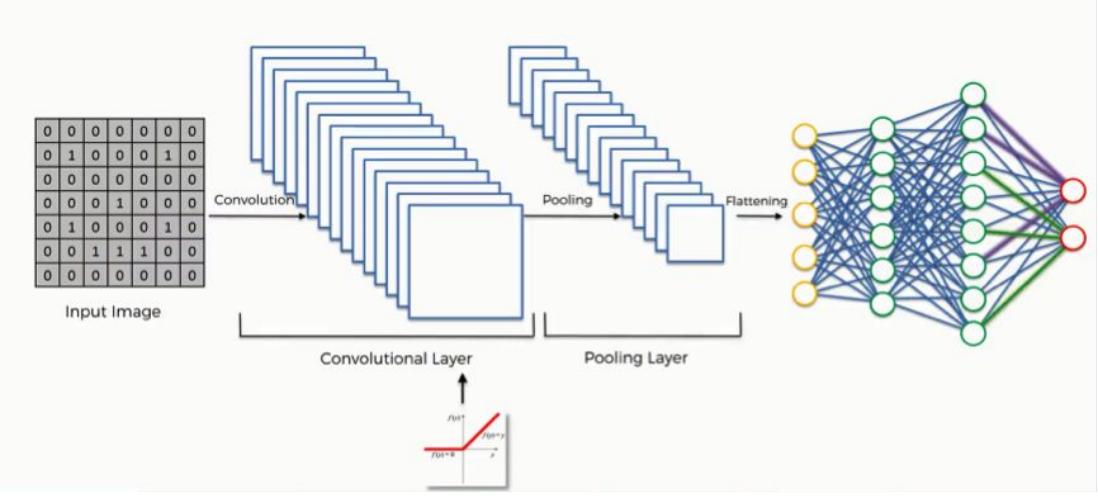
We report on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. We show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. We additionally propose a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed herein improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

local features (LeCun et al., 1998). Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing (Yih et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011).

In the present work, we train a simple CNN with one layer of convolution on top of word vectors obtained from an unsupervised neural language model. These vectors were trained by Mikolov et al. (2013) on 100 billion words of Google News, and are publicly available.[1] We initially keep the word vectors static and learn only the other parameters of the model. Despite little tuning of hyperparameters, this simple model achieves excellent results on multiple benchmarks, suggesting that the pre-trained vectors are 'universal' feature ex-

CNN

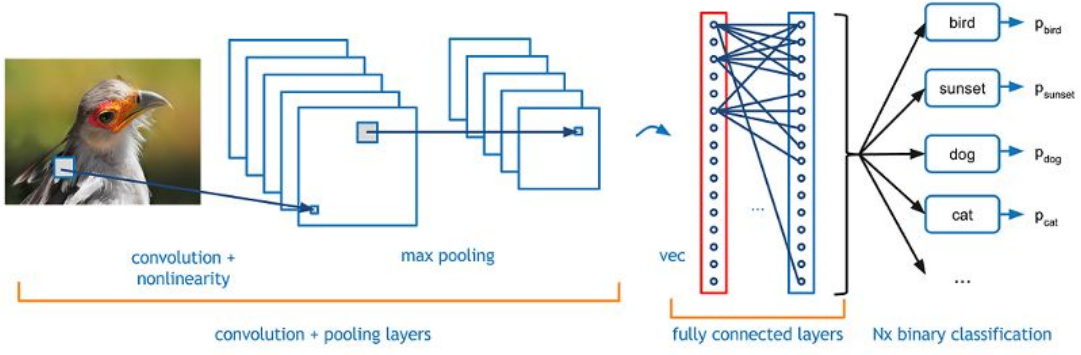
- CNN is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers
- Appy kernels/filters to an image to extract features
- Compress the features using Max Pooling or Average Pooling
- 데이터의 2차원 형상을 보존하여 공간적 정보 고려



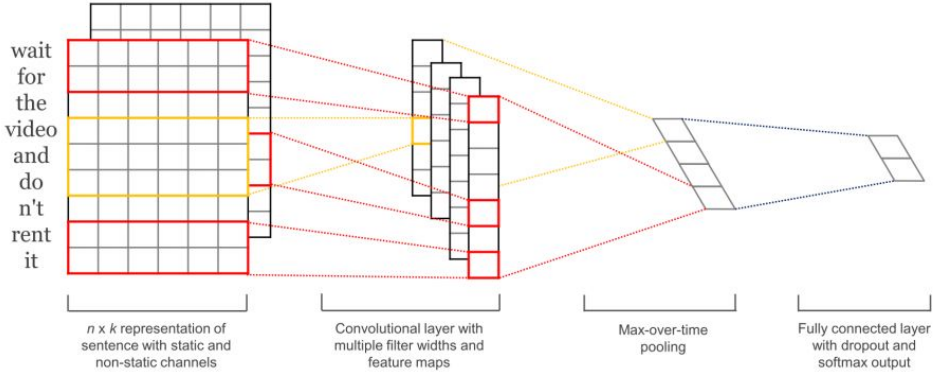
https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-summary/

CNN (convolutional neural network)

- For image classification

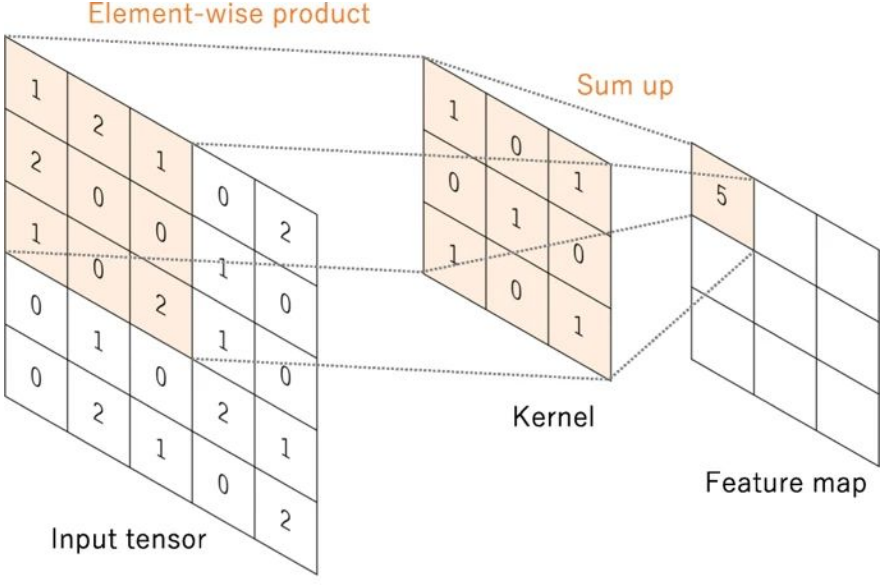


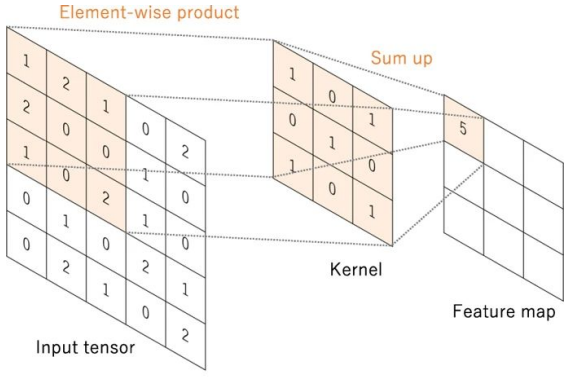
- For text classification

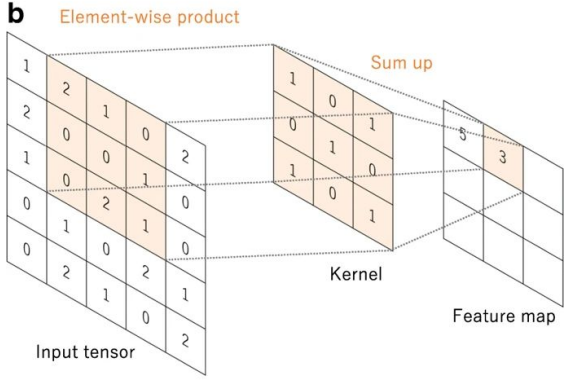


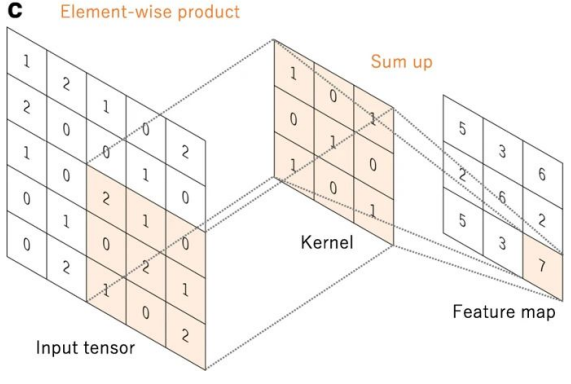
Convolution

- Element-wise multiplication and summation (fused multiply-add, FMA)





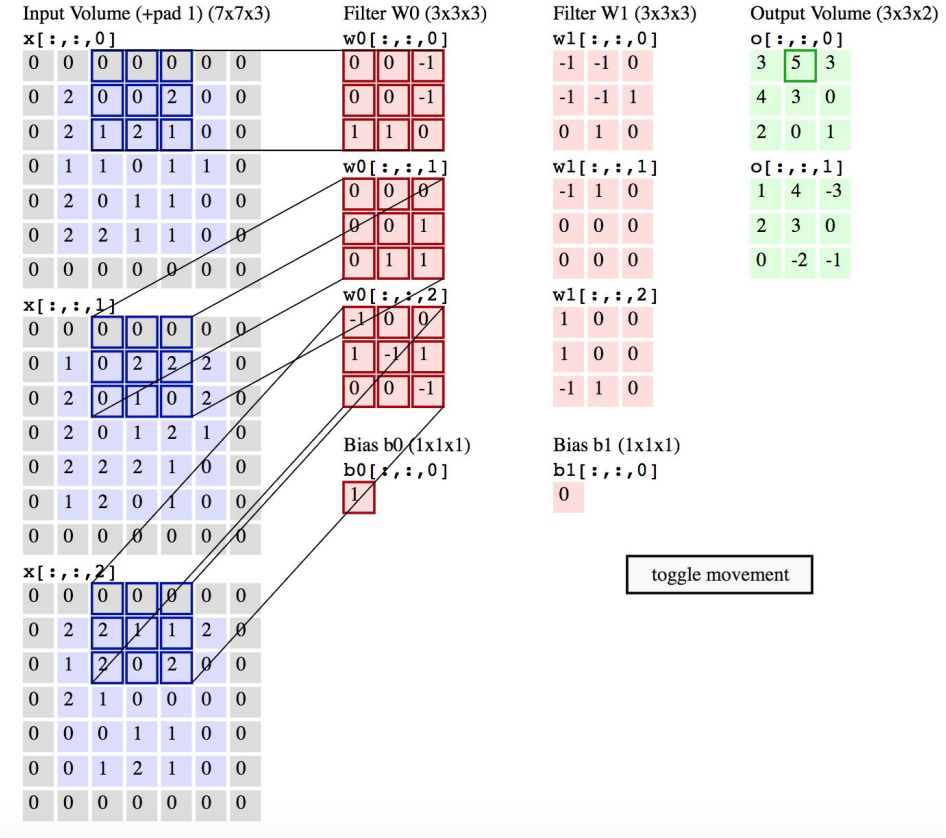




https://link.springer.com/article/10.1007/s13244-018-0639-9#:~:text=CNN%20is%20a%20type%20of,%2D%20to%20high%2Dlevel%20patterns.

CNN

- **Padding**: adding enough data to the input's outer edges
- **Stride (step size)**: the distance each convolution "travels"



http://cs231n.github.io/convolutional-networks/

1D Convnet for text processing

1 × 3 Filter

The cat and dog went to the bodega together.

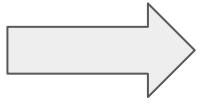
1 × 3 Filter

The cat and dog went to the bodega together.

1 × 3 Filter

The cat and dog went to the bodega together.

Slide in one dimension, left to right.



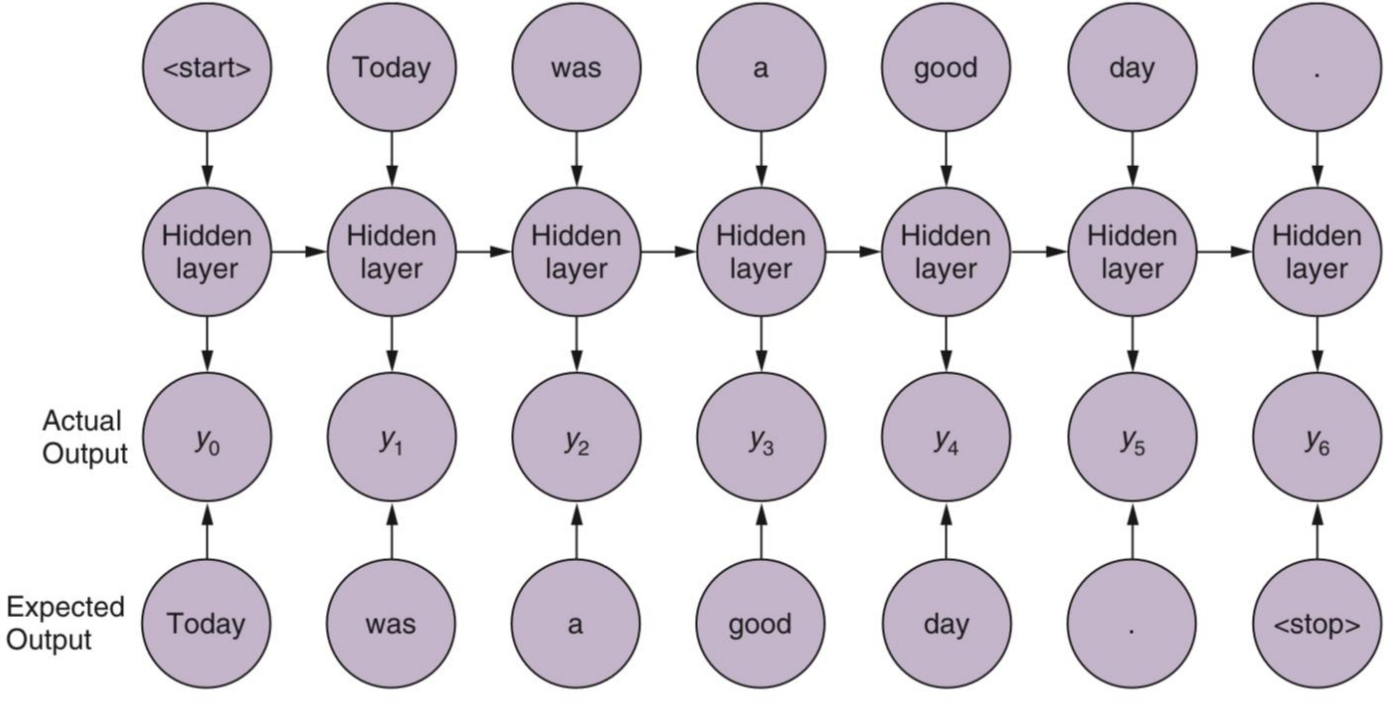
Classification of IMDB Reviews

https://wikidocs.net/80783

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Dropout, Conv1D,
GlobalMaxPooling1D, Dense

embedding_dim = 256
batch_size = 256

model = Sequential()
model.add(Embedding(vocab_size, 256))
model.add(Dropout(0.3))
model.add(Conv1D(256, 3, padding='valid', activation='relu')) # kernel number and size
model.add(GlobalMaxPooling1D())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss = 'binary_crossentropy', metrics = ['acc'])
history = model.fit(X_train, y_train, epochs = 20, validation_data = (X_test, y_test))
```

seq2seq

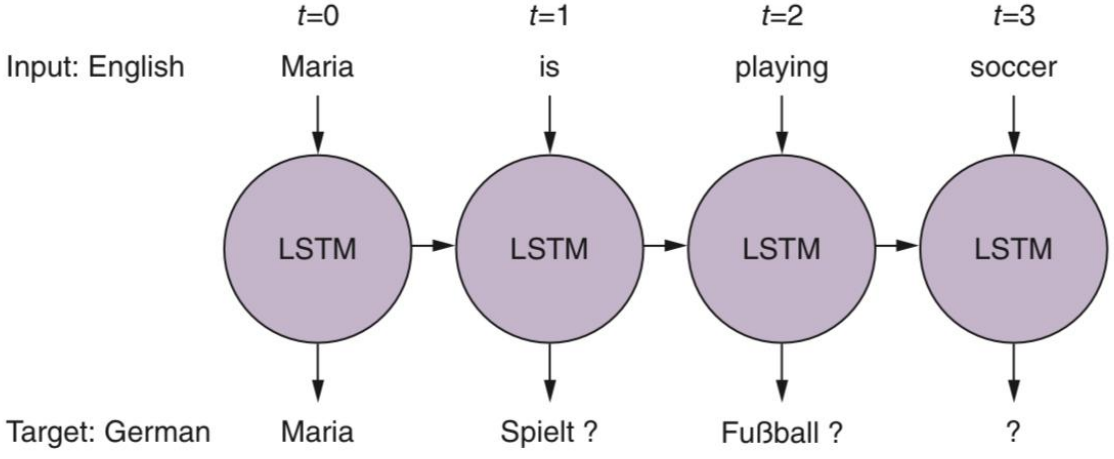
RNN for text generation



Expected output is the next token in the sample. Shown here on word level.

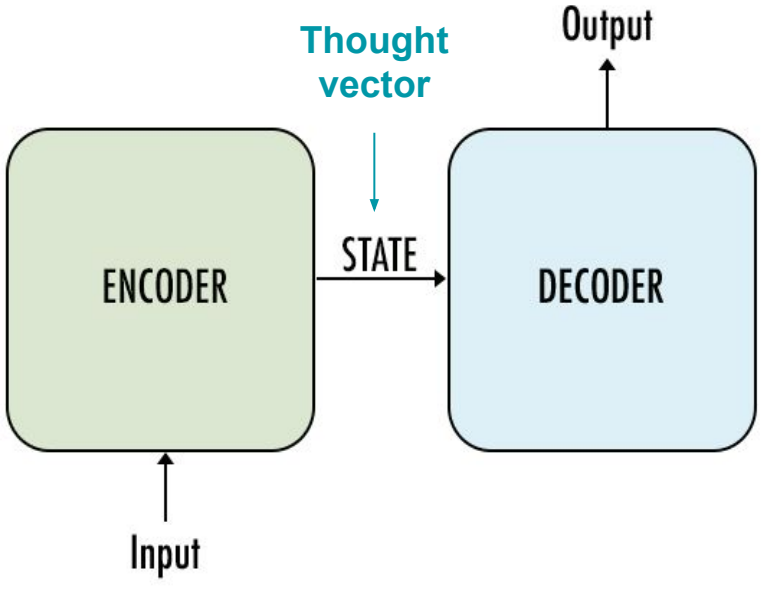
Limitations of RNN

- We can predict a sequence element at time step *t* based on the previous element at time step *t-1*
- But directly using an LSTM for translation is problematic. For a single LSTM to work, we need input and output sequences to have the same sequence lengths, and for translation they rarely do
- The English and the German sentence have different lengths - “is playing” is translated to the German “spielt.” But “spielt” here would need to be predicted solely on the input of “is;” you haven’t gotten to “playing” yet at that time step.



Encoder-Decoder Architecture

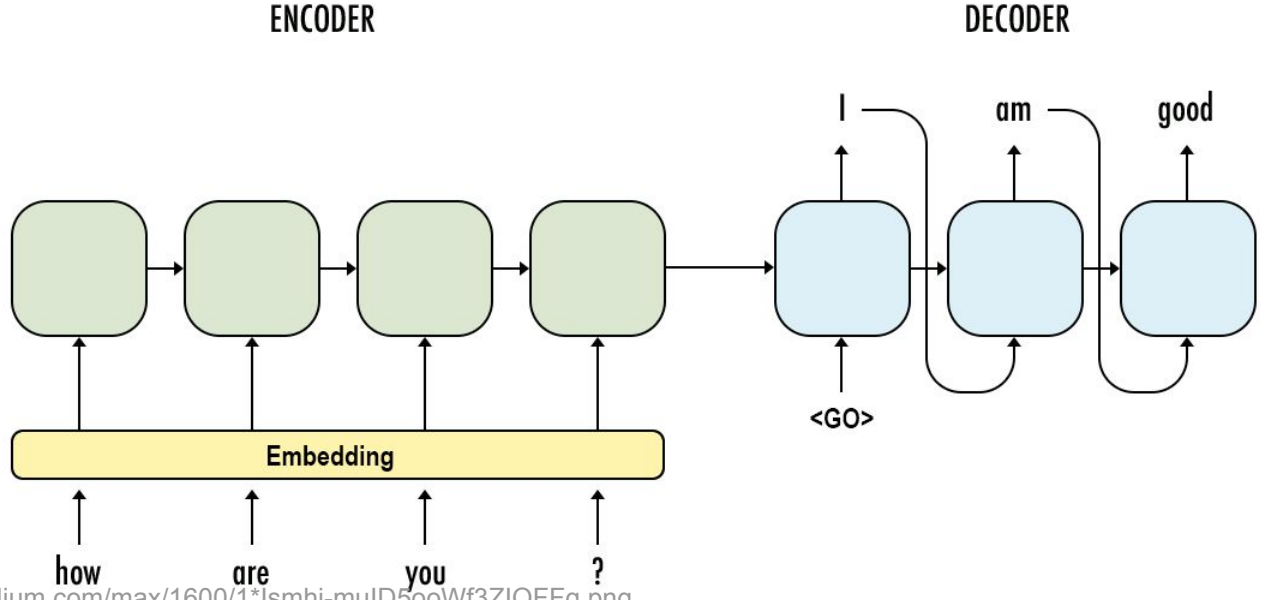
- The generated state (**thought vector**) will serve as the initial state of the decoder network
- The second network then uses that initial state and a start token. Primed with that information, the second network has to learn to generate the first element of the target sequence
- Trained end-to-end this way, the decoder will turn a thought vector into a fully decoded response to the initial input sequence



https://cdn-images-1.medium.com/max/1600/1*Ismhi-muID5ooWf3ZIQFFg.png

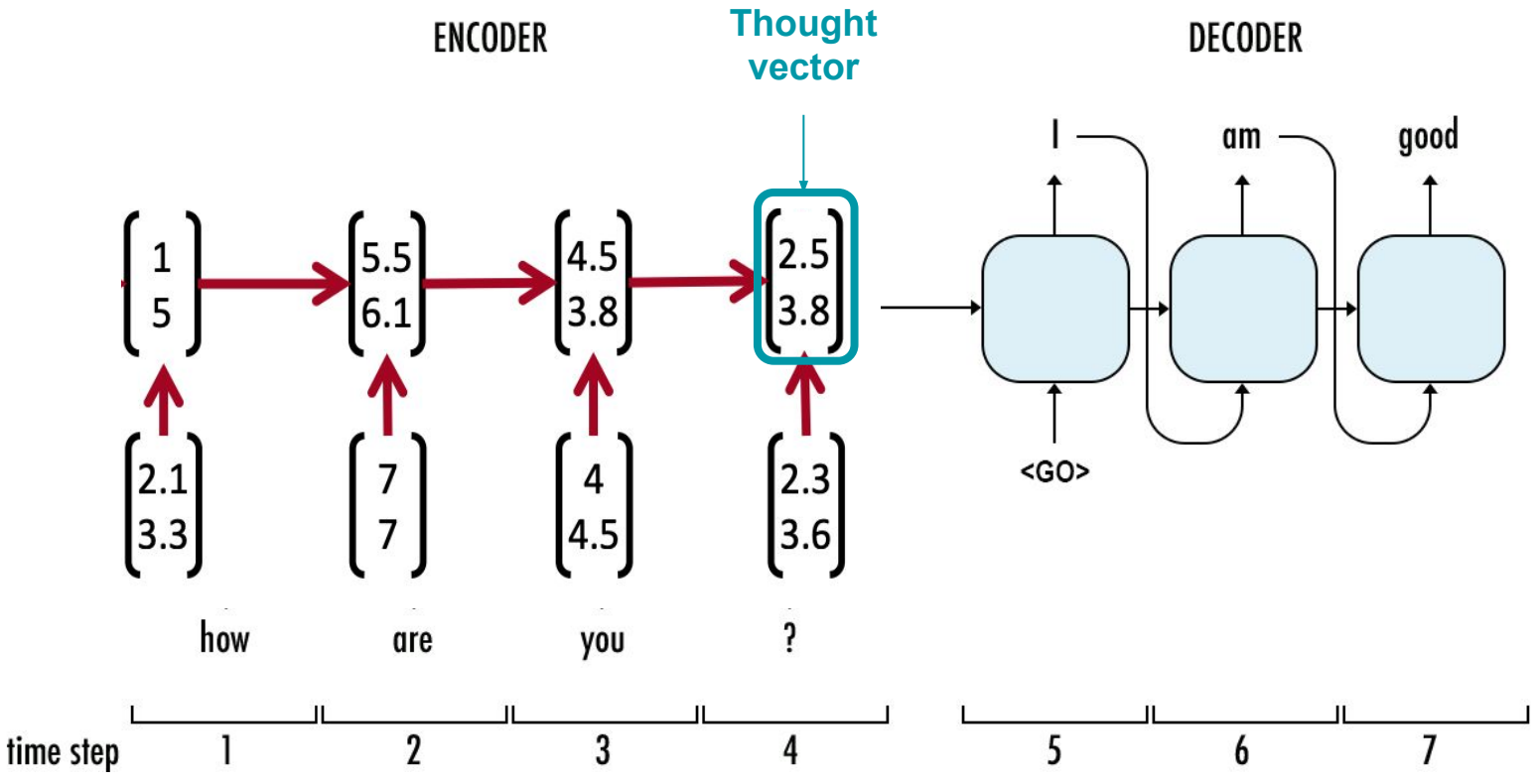
seq2seq

- During training, pass the starting text to the encoder and the expected text as the input to the decoder
- The first direct input to the decoder will be the start token; the second input should be the first expected or predicted token, which should in turn prompt the network to produce the second expected token



https://cdn-images-1.medium.com/max/1600/1*lsmhi-muID5ooWf3ZIQFFg.png

seq2seq



https://cdn-images-1.medium.com/max/1600/1*Ismhi-muID5ooWf3ZIQFFg.png

Preparing your dataset for the sequence-to-sequence training

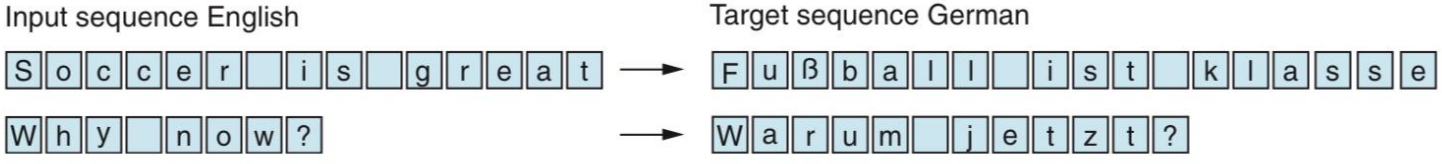


Figure 10.5 Input and target sequence before preprocessing

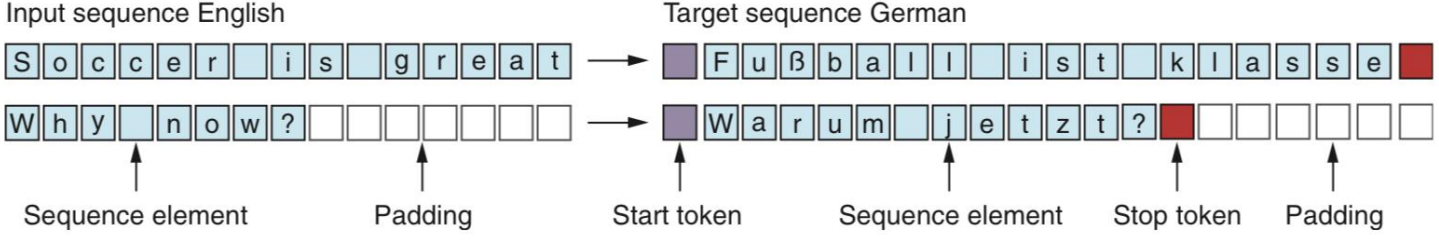
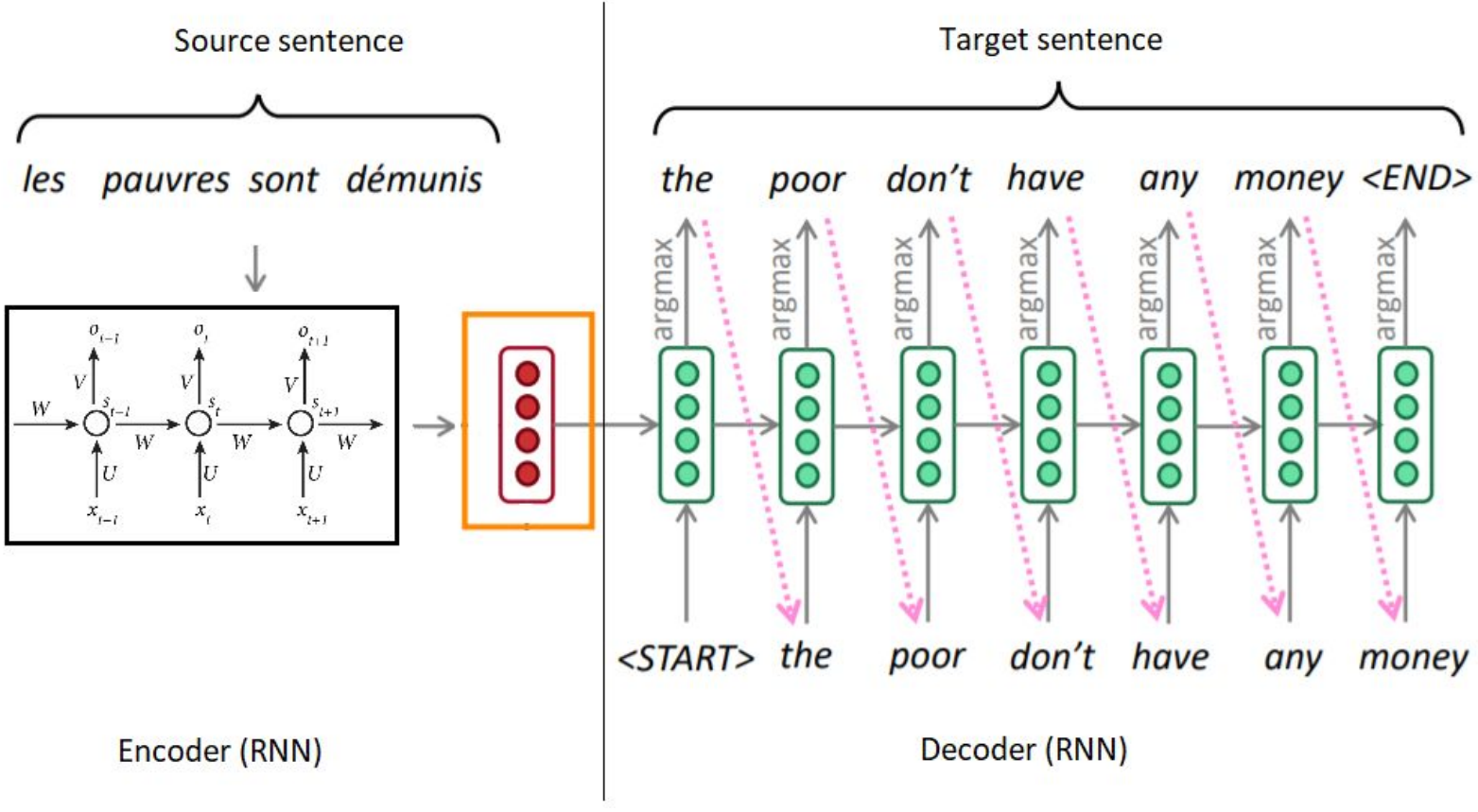


Figure 10.6 Input and target sequence after preprocessing

Key seq2seq applications

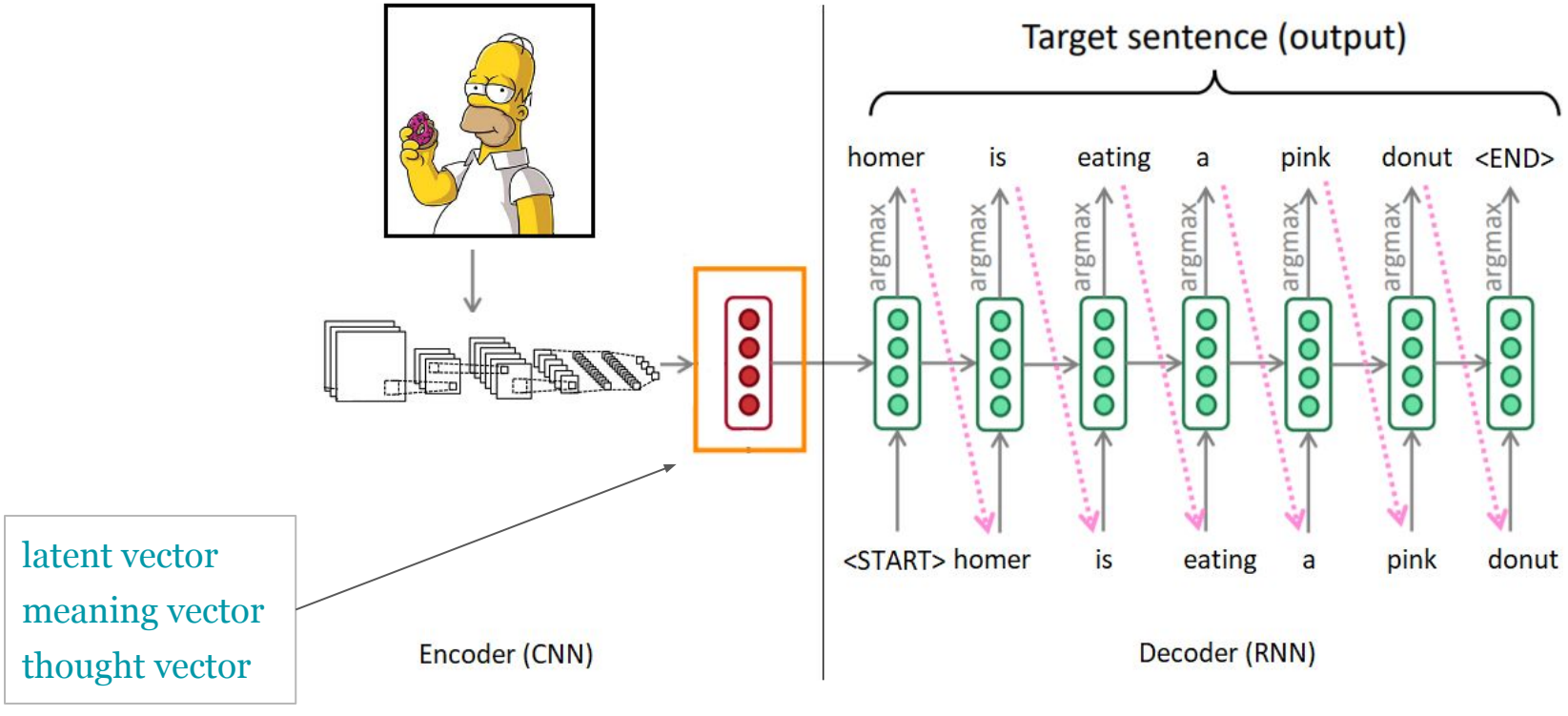
- Chatbot conversations
- Question answering
- Machine translation
- Email reply
- Image captioning
- Visual question answering
- Document summarization
- 자동 프로그래밍: 소스코드와 출력

Translation



https://medium.com/phrasee/neural-text-generation-generating-text-using-conditional-language-models-a37b69c7cd4b

Image captioning



https://medium.com/phrasee/neural-text-generation-generating-text-using-conditional-language-models-a37b69c7cd4b

Machine Translation code using Keras

- Seq2seq training for English-French
- https://tykimos.github.io/2018/09/14/ten-minute_introduction_to_sequence-to-sequence_learning_in_Keras/
- (original source) https://keras.io/examples/lstm_seq2seq/

```
from keras.models import Model
from keras.layers import Input, LSTM, Dense

encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# `encoder_outputs`는 버리고 상태(`state_h, state_c`)는 유지
encoder_states = [state_h, state_c]

# `encoder_states`를 초기 상태로 사용해 decoder를 설정
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# 전체 출력 시퀀스를 반환하고 내부 상태도 반환하도록 decoder를 설정.
# 학습 모델에서 상태를 반환하도록 하진 않지만, inference에서 사용할 예정.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
model.fit([encoder_input_data, decoder_input_data], decoder_target_data,
          batch_size=batch_size,  epochs=epochs,validation_split=0.2)
```

Attention

ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

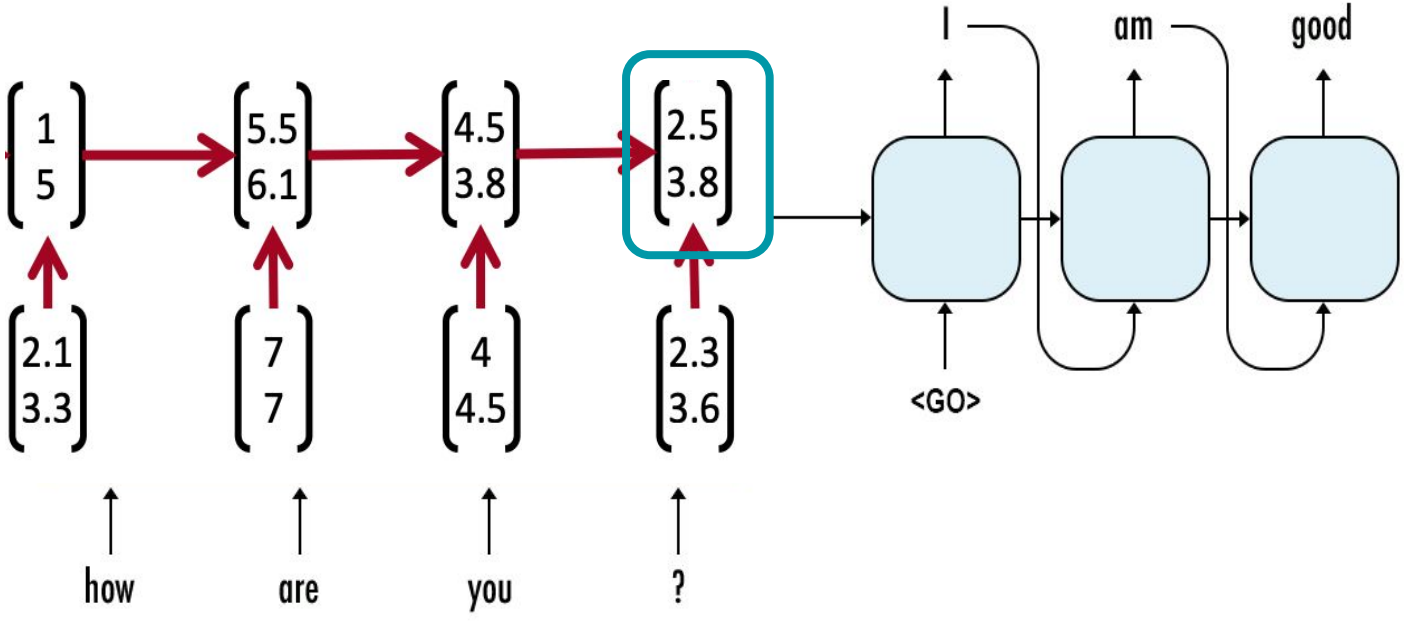
KyungHyun Cho **Yoshua Bengio***
Université de Montréal

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.

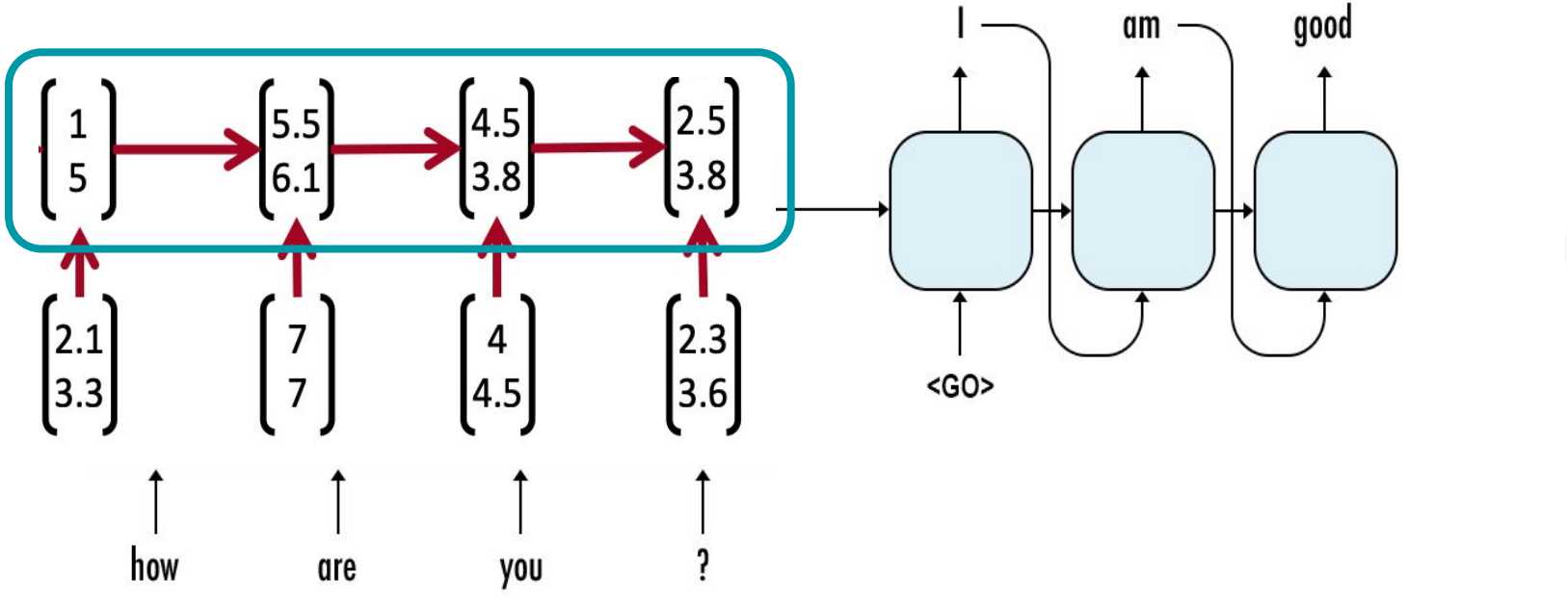
seq2seq의 문제

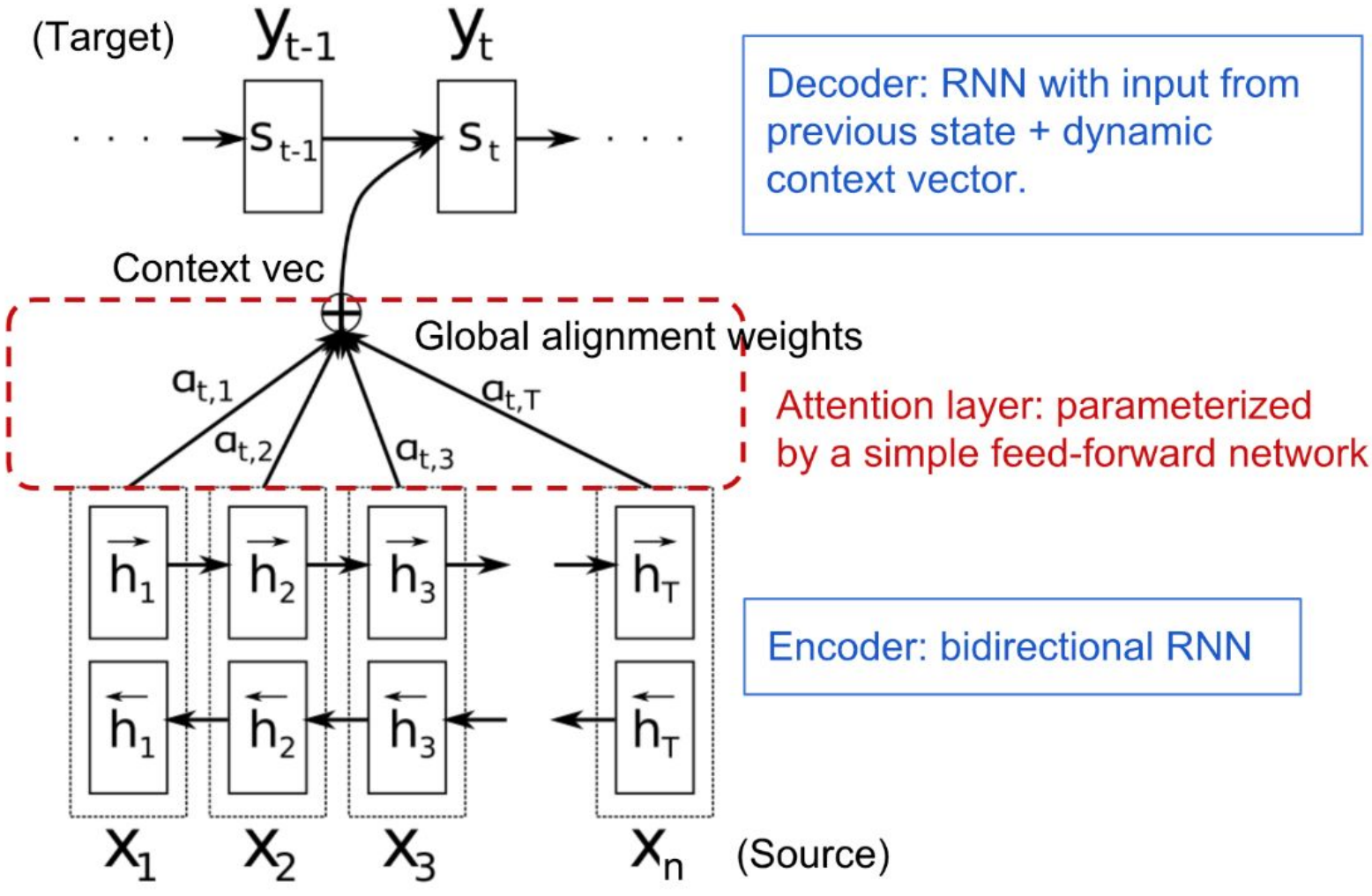
- 고정 길이 벡터에 입력 문장을 밀어 넣으므로 필요한 정보가 다 담기지 못한다



Solution

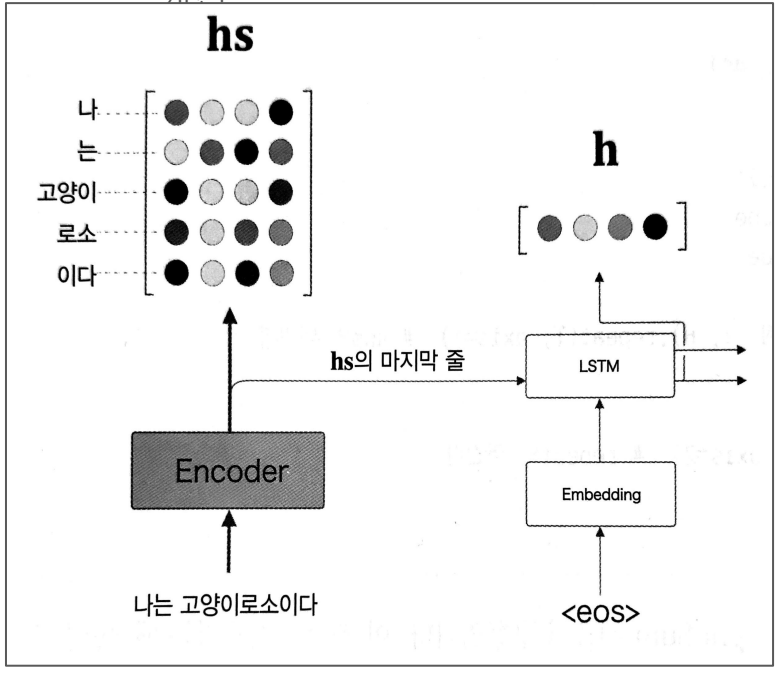
- Encoder는 각 시각에서 계산한 모든 h 값을 보낸다
- Decoder는 alignment를 이용하여 모든 h중 **필요한 정보에 주목**하여 변환을 수행

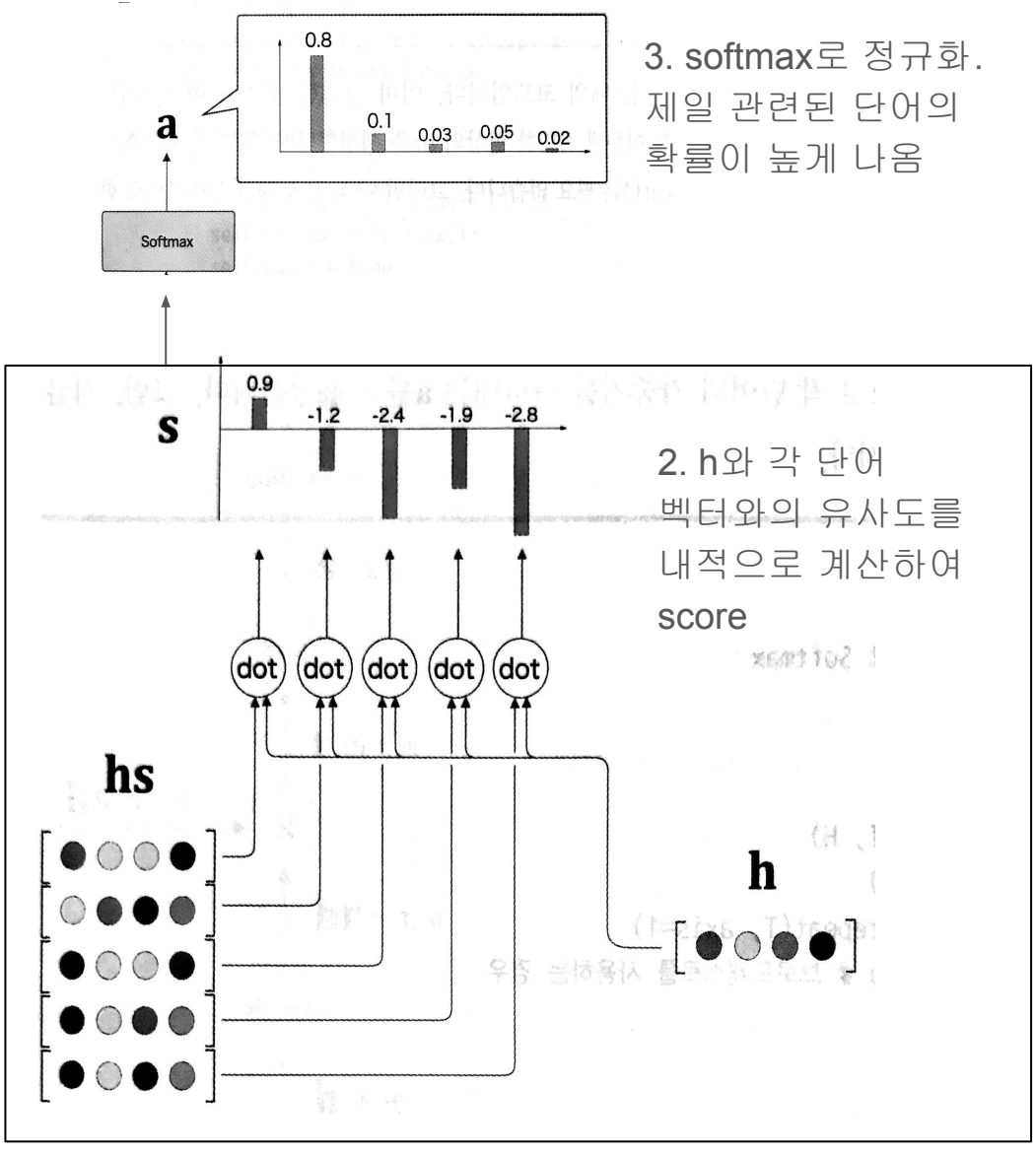



(Target)
y_{t-1}
y_t
s_{t-1}
s_t
Context vec
Global alignment weights
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$
$\overrightarrow{h}_2$
$\overrightarrow{h}_3$
$\overrightarrow{h}_T$
$\overleftarrow{h}_1$
$\overleftarrow{h}_2$
$\overleftarrow{h}_3$
$\overleftarrow{h}_T$
x_1
x_2
x_3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

Alignment 계산

1. Decoder의 첫번째 h값 계산



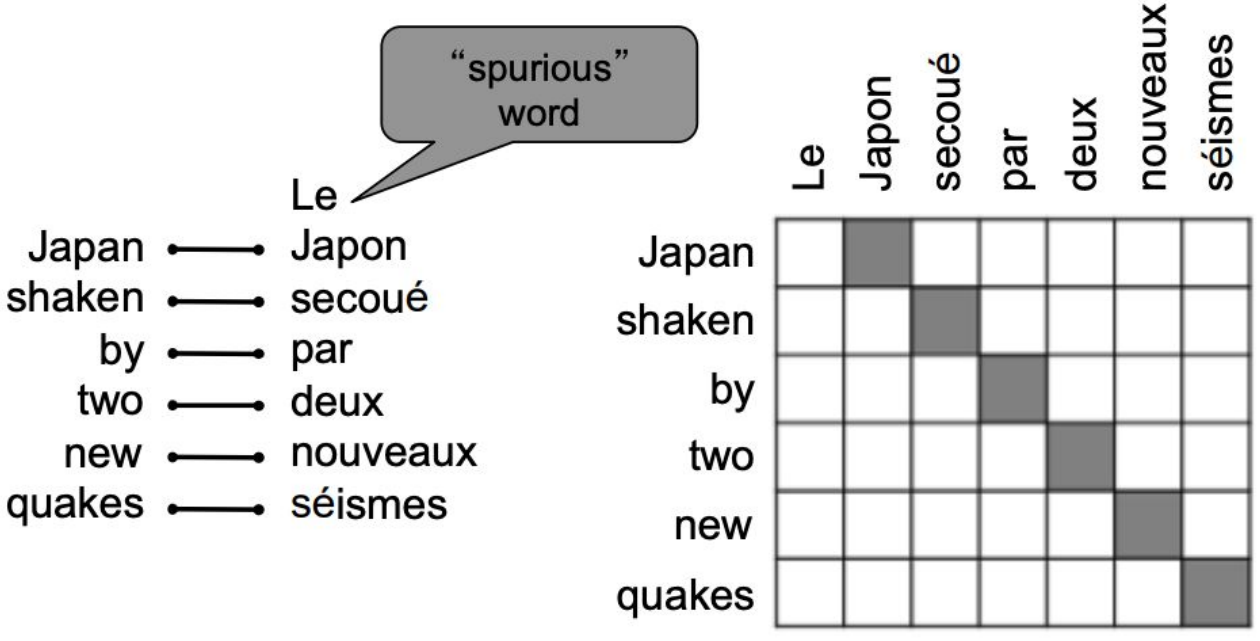


3. softmax로 정규화. 제일 관련된 단어의 확률이 높게 나옴

2. h와 각 단어 벡터와의 유사도를 내적으로 계산하여 score

Alignment

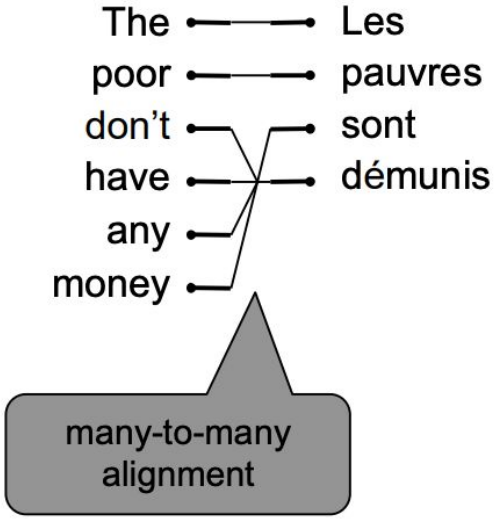
- 번역 문장 쌍에서 단어간 대응 관계

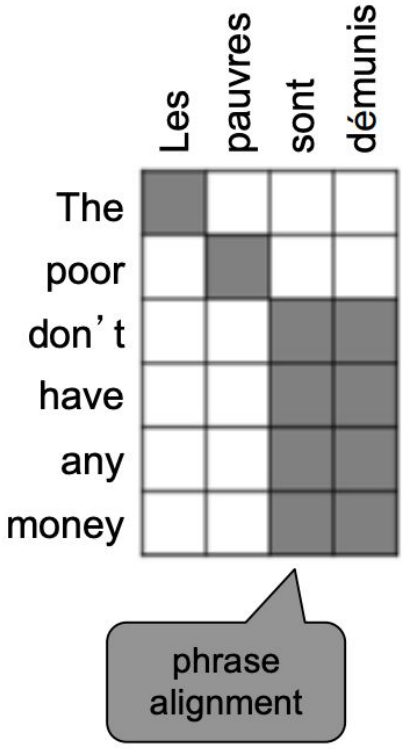


http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Alignment

- 일대일 대응이 아닌 관계도 있어 복잡





Alignment visualization

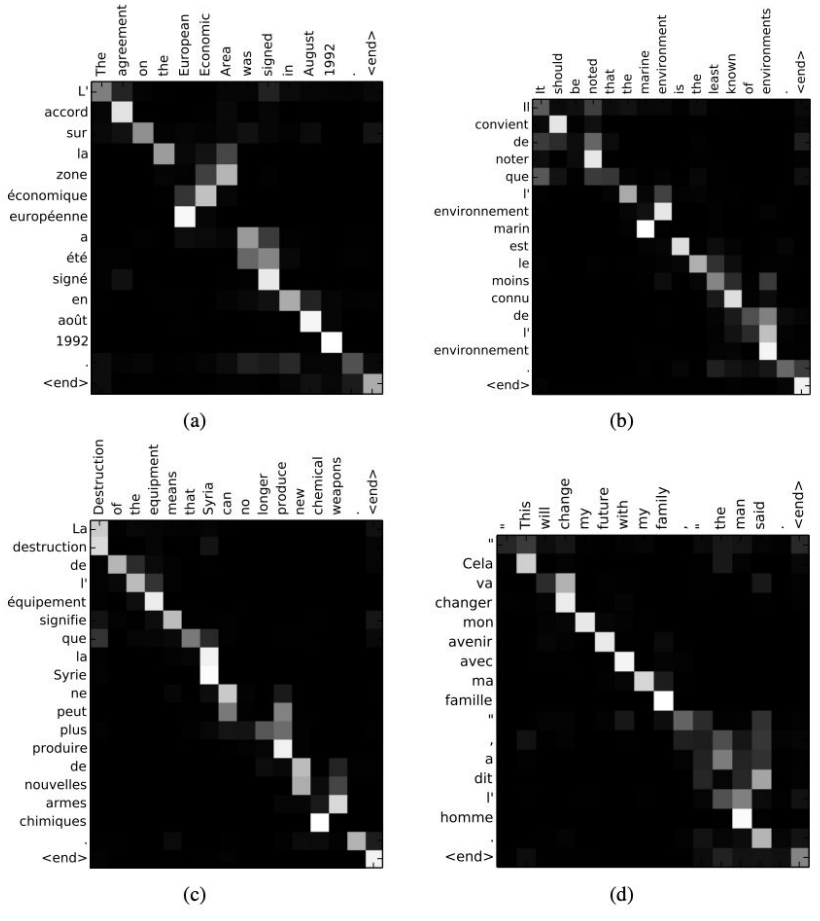


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

Summarization using attention

https://wikidocs.net/72820

Summary of Attention

- Use the hidden values of all time-steps
- Attention to the most important *h*
- Alignment weight based on similarity between the current *h* of decoder and each *h* of encoder
- Weighted sum as context vector
- Use the context vector for decoding

Evaluation

Evaluation of Translation and Generation

- Translation
- Generation
- Summarization

BLEU(Bilingual Evaluation Understudy)

Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.

BLEU: a Method for Automatic Evaluation of Machine Translation

Kishore Papineni, Salim Roukos, Todd Ward, and **Wei-Jing Zhu**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{papineni,roukos,toddward,weijing}@us.ibm.com

Abstract

Human evaluations of machine translation are extensive but expensive. Human evaluations can take months to finish and involve human labor that can not be reused. We propose a method of automatic machine translation evaluation that is quick, inexpensive, and language-independent, that correlates highly with human evaluation, and that has little marginal cost per run. We present this method as an automated understudy to skilled human judges which substitutes for them when there is need for quick or frequent evaluations.[1]

the evaluation bottleneck. Developers would benefit from an inexpensive automatic evaluation that is quick, language-independent, and correlates highly with human evaluation. We propose such an evaluation method in this paper.

1.2 Viewpoint

How does one measure translation performance? *The closer a machine translation is to a professional human translation, the better it is.* This is the central idea behind our proposal. To judge the quality of a machine translation, one measures its closeness to one or more reference human translations according to a numerical metric. Thus, our MT evaluation system requires two ingredients:

BLEU Measures *precision*

- Candidate와 reference의 n-gram이 일치하는 수치 (단어의 순서는 무관)
- Candidate 1이 Candidate 2 보다 더 좋은 번역이라고 평가

예측 문장

Candidate 1: **It is a guide to action which ensures that the military always obeys the commands of the party.**

Candidate 2: **It is to insure the troops forever hearing the activity guidebook that party direct.**

Reference Translations (참고 문장)

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

BLEU limitations

- 예측 문장 길이가 너무 짧거나 같은 단어가 반복해서 나올 경우 Precision이 높아짐
- Clipping: reference중복되는 단어의 max count 기준으로 점수를 제한

예측 문장
Candidate: **the the the the the the the**

Reference Translations(번역 평가를 위한 참고 문장)
Reference: **The cat is on the mat**

- Standard unigram precision : **7/7**
- Unified unigram precision : **2/7**

ROUGE (Lin, 2004)

- Originally developed for evaluating summaries
- 정밀도(precision): 찾은 것 중에 정답의 비율
- **재현율(recall): 찾아야 하는 것을 얼마나 찾았는가**
- ROUGE-N: Overlap of N-grams
- ROUGE-L: Longest Common Subsequence (LCS) based statistics
- ROUGE-W: weighted LCS-based statistics that favors consecutive LCSes
- ROUGE-S: Skip-bigram based co-occurrence statistics

ROUGE-N : N-gram Co-Occurrence Statistics

- ROUGE-N refers to the overlap of 1-gram between the system and reference summaries
  - ROUGE-1 refers to the overlap of 1-gram
  - ROUGE-2 refers to the overlap of bigrams
- ***Prediction: the cat was found under the bed***
- ***Reference: the cat was under the bed***
  - ROUGE-1 = 1
  - ROUGE-2 = 0.8

```
the cat,
cat was,
was found,
found under,
under the,
the bed
```

```
the cat,
cat was,
was under,
under the,
the bed
```

ROUGE-L : Longest Common Subsequence

- Measures longest matching sequence of words using LCS
- An advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order
- Since it automatically includes longest in-sequence common n-grams, we don’t need a predefined n-gram length.

ROUGE-S : Skip-Bigram Co-Occurrence Statistics

- Any pair of word in a sentence in order, allowing for arbitrary gaps
- For example, skip-bigram measures the overlap of word pairs that can have a maximum of two gaps in between words
- E.g., “cat in the hat” the skip-bigrams would be “cat in, cat the, cat hat, in the, in hat, the hat”

METEOR (Metric for Evaluation of Translation with Explicit ORdering)

- **Harmonic mean** of unigram precision and recall, with recall weighted higher than precision
- **Stemming and synonymy** matching, along with the standard exact word matching

Limitations

- **Only assess content selection** and do not account for other quality aspects, such as fluency, grammaticality, coherence, etc.
- **Rely on lexical overlap**, although an abstractive summary could express they same content as a reference without any lexical overlap
- Given the subjectiveness of summarization and the correspondingly low agreement between annotators, the metrics were designed to be used with multiple reference summaries per input. However, recent datasets such as CNN/DailyMail and Gigaword provide **only a single reference**
- Therefore, tracking progress and claiming state-of-the-art based only on these metrics is questionable. Most papers carry out additional manual comparisons of alternative summaries. Unfortunately, such experiments are difficult to compare across papers

http://nlpprogress.com/english/summarization.html

Summary

- CNN for text processing
- Seq2seq
- Attention
- Evaluation of generation, translation, and summarization