영어-프랑스어 번역기

2017108065 김동욱

CONTENTS

개요/필요성

NLP(Natural Language Processing, 자연어 처리)는 인공지능의 한 분야로서 머신러닝을 사용하여 텍스트와 데이터를 처리하고 해석합니다. 자연어 인식 및 자연어 생성이 NLP의 유형입니다.

1. 번역 및 맞춤법 검사
2. 인공지능 기반 음성인식 스피커
3. 사용자 정서분석

과제	벤치마크	예제	
감정 분석 Sentiment Analysis	SST, IMDb, NSMC	**입력**	**문장:** 신만이 이 영화를 용서할수 있다
		출력	긍정 · **부정**
유사도 예측 Similarity Prediction	STS, MRPC, QQP, PAWS-X	**입력**	**문장1:** 파이썬과 자바 중 뭐부터 배워야 하나요? **문장2:** Java나 Python 중 하나를 배워야 한다면, 뭐부터 시작해야 할까요?
		출력	**유사** · 비유사
자연어 추론 Natural Language Inference	SNLI, MNLI, XNLI, aNLI	**입력**	**전제:** 회색 개가 숲에서 쓰러진 나무를 핥고 있다. **가설:** 개가 밖에 있다.
		출력	**참** · 거짓 · 모름
언어적 용인 가능성 Linguistic Acceptability	CoLA	**입력**	**문장:** 그는 한 번도 프랑스에 갔다.
		출력	가능 · **불가능**
기계 독해 Reading Comprehension	SQuAD, RACE, MS MARCO, KorQuAD	**입력**	**지문:** 시카고 대학의 학자들은 경제학, 사회학, 법학 분석, 문학 비평, 신학, 행동주의 정치학 등 다양한 학문 발전에 중요한 역할을 담당해왔다. […] 이 대학은 미국 최대 대학 출판사인 시카고 대학 프레스(University Press of Chicago Press)의 본거지이기도 하다. 오는 2020년 완공될 버락 오바마 대통령 센터에는 오바마 대통령 도서관과 오바마 재단 사무소가 세워질 예정이다. **질문:** 버락 오바마 대통령 센터의 완공 예정연도는?
		출력	**2020년**
의도 분류 Intent Classification	ATIS, SNIPS, AskUbuntu	**입력**	**질문:** 현재 위치 주변에 있는 맛집 두 곳만 알려줘
		출력	**장소 검색** (식당, 가까운, 평점 좋은, 2)

관련 내용/ 연구

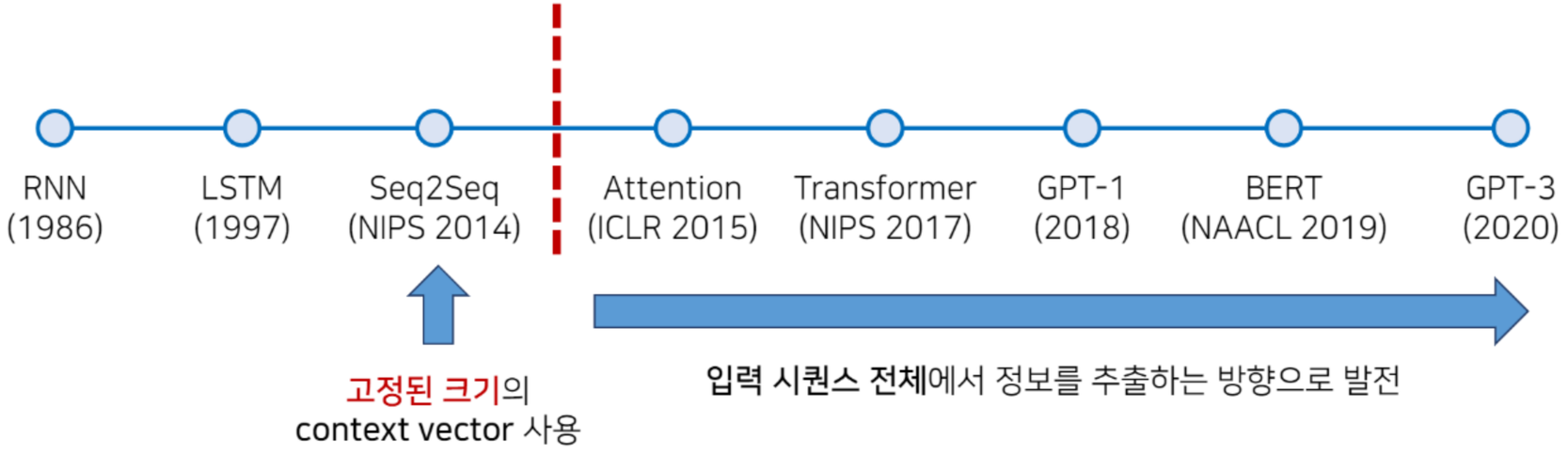
기계번역 발전과정

attention 기법을 활용한 2015년

이후 모델부터 성능이 크게 향상됨

딥러닝 기반의 기계 번역 발전 과정

- 2021년 기준으로 최신 고성능 모델들은 Transformer 아키텍처를 기반으로 하고 있습니다.
  - **GPT**: Transformer의 **디코더(Decoder)** 아키텍처를 활용
  - **BERT**: Transformer의 **인코더(Encoder)** 아키텍처를 활용



Sequence to Sequence Learning with Neural Networks (NIPS 2014)

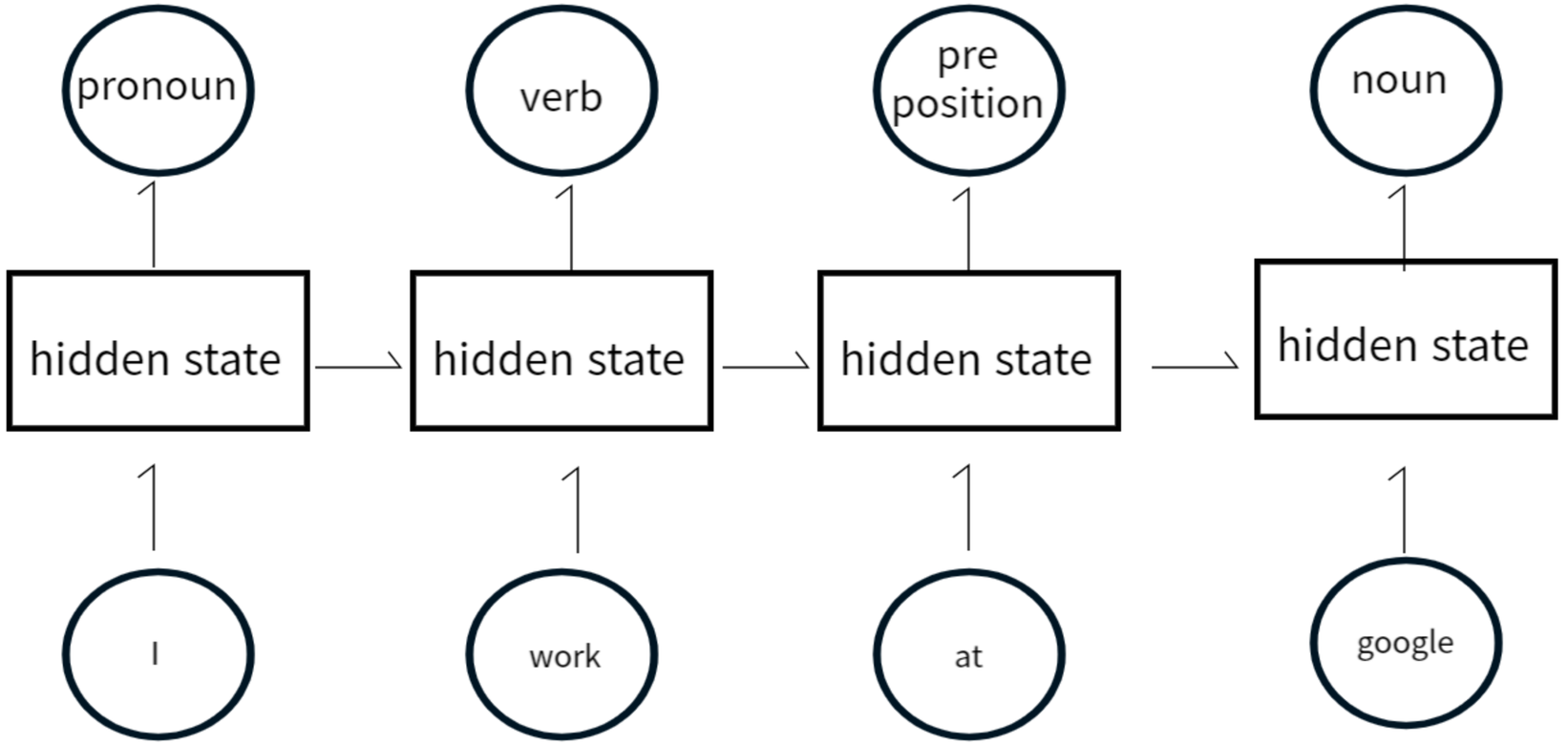
나동빈

RNN(Recurrent Neural Network)

순환 신경망

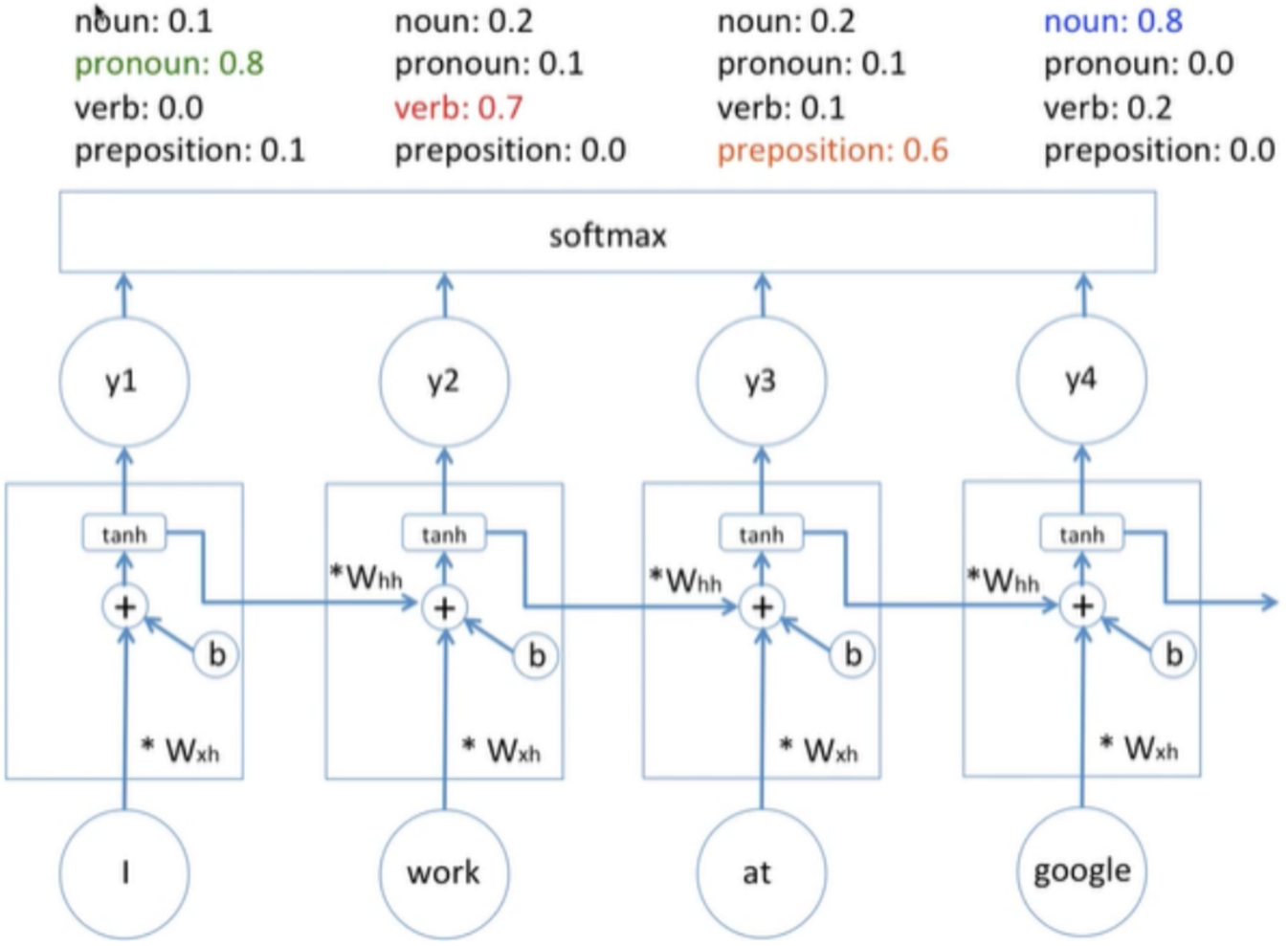
I google at work >> 일할때 난 구글 써

I work at google >> 난 구글에서 일해



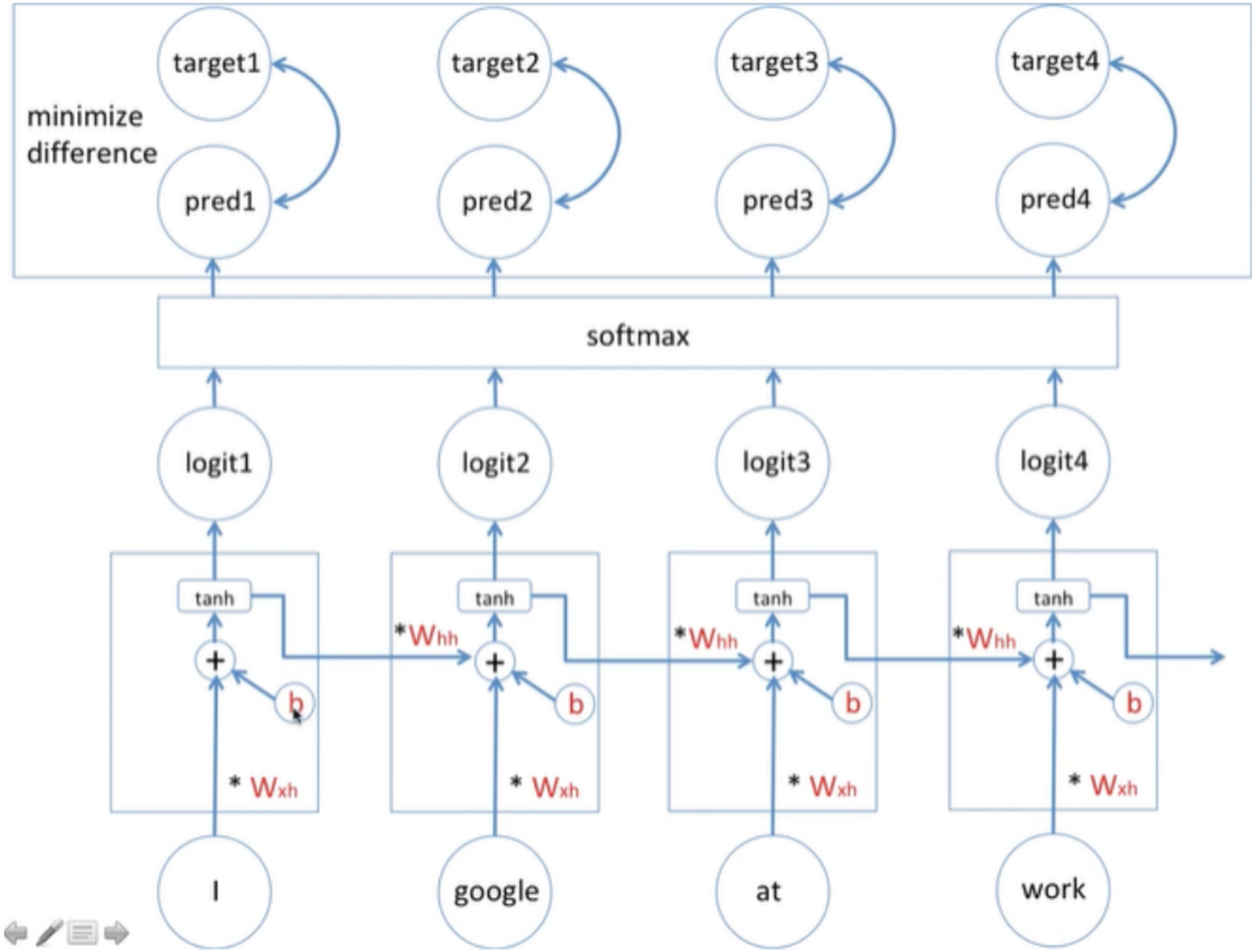
RNN(Recurrent Neural Network)

순환 신경망



RNN(Recurrent Neural Network)

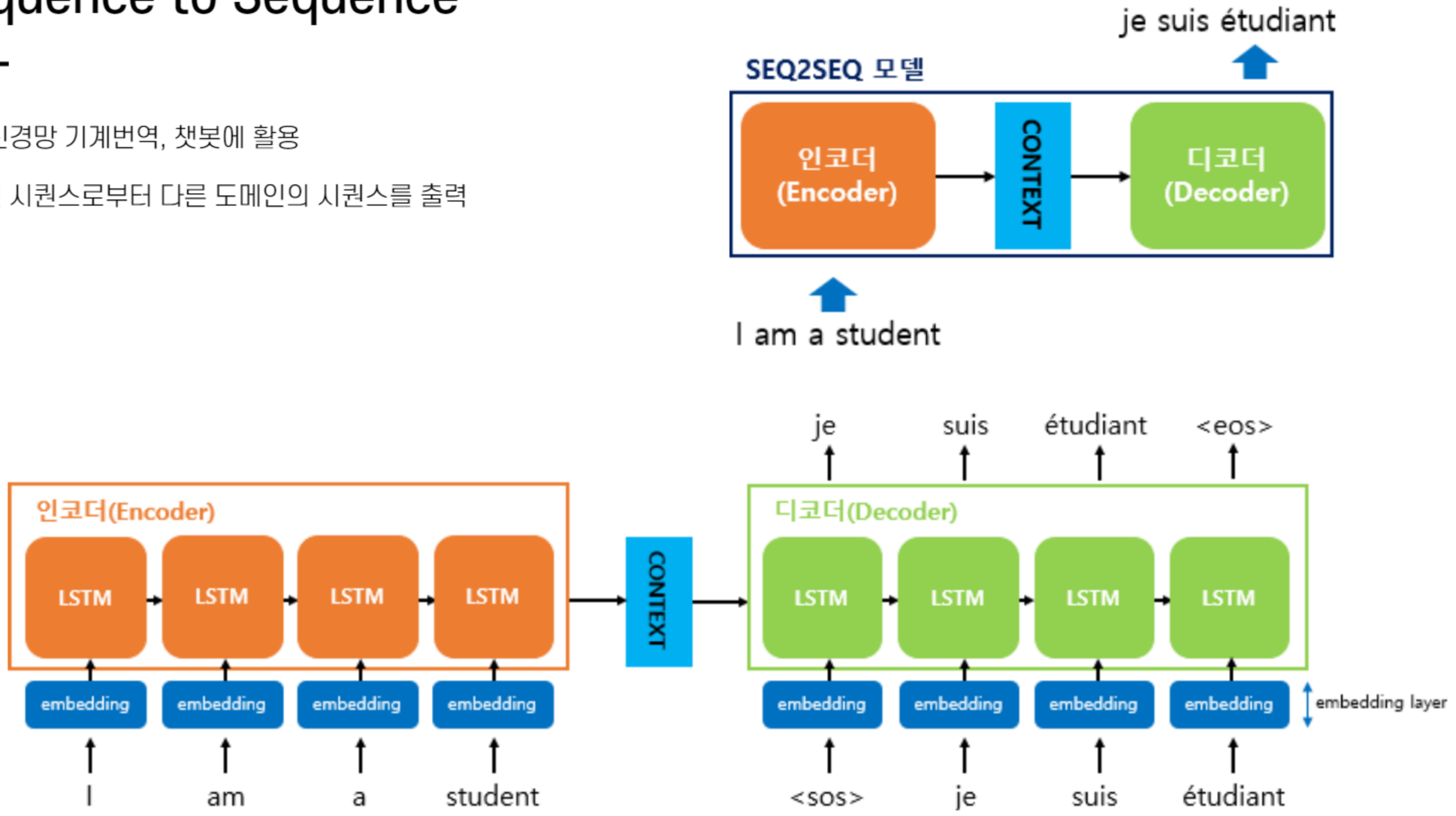
순환 신경망



Sequence to Sequence

인공신경망 기계번역, 챗봇에 활용

입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력



01

In [ ]:

```
#import all libraries
import pandas as pd
import numpy as np
```

In [ ]:

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model, Sequential
from keras.layers import GRU, Input, Dense, TimeDistributed, Activation, RepeatVector, Bidirec
tional,LSTM, Dropout
from keras.layers.embeddings import Embedding
from keras.optimizers import Adam
from keras.losses import sparse_categorical_crossentropy
from keras.callbacks import ModelCheckpoint
```

Libraries

—

주요 라이브러리

numpy

pandas

keras

datasets

csv파일의 형태로 데이터를 받음

02

In [ ]:
```
df  =pd.read_csv('/content/eng_-french.csv')
df.head()
```

Out[ ]:

	English words/sentences	French words/sentences
0	Hi.	Salut!
1	Run!	Cours!
2	Run!	Courez!
3	Who?	Qui ?
4	Wow!	Ça alors!

Separating the English and French data.

In [ ]:
```
eng = df['English words/sentences']
fr = df['French words/sentences']
```

03

In [ ]:

```
english_words_counter = collections.Counter([word for sentence in eng for word in sentence.spl
it()])
french_words_counter = collections.Counter([word for sentence in fr for word in sentence.split
()])

print('{} English words.'.format(len([word for sentence in eng for word in sentence.split
()])))
print('{} unique English words.'.format(len(english_words_counter)))
print('10 Most common words in the English dataset:')
print('"' + '" "'.join(list(zip(*english_words_counter.most_common(10)))[0]) + '"')
print()
print('{} French words.'.format(len([word for sentence in fr for word in sentence.split()])))
print('{} unique French words.'.format(len(french_words_counter)))
print('10 Most common words in the French dataset:')
print('"' + '" "'.join(list(zip(*french_words_counter.most_common(10)))[0]) + '"')
```

```
1082098 English words.

27393 unique English words.

10 Most common words in the English dataset:

"I" "to" "you" "the" "a" "is" "Tom" "of" "in" "have"



1177832 French words.

44918 unique French words.

10 Most common words in the French dataset:

"de" "Je" "?" "pas" "que" "à" "ne" "la" "le" "Tl"
```

데이터 분석

단어의 수

중복 단어 제외한 단어 수

많이 사용된 10개의 단어

토큰화

04

In [ ]:
```
def tokenize(x):

    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(x)
    return tokenizer.texts_to_sequences(x), tokenizer
```

We also have to pad each text as all the neural networks needs to have the inputs that should be in similar shape and size.

In [ ]:
```
def pad(x, length=None):
    if length is None:
        length = max([len(sentence) for sentence in x])
    return pad_sequences(x, maxlen = 55, padding = 'post')
```

05

In [ ]:
```
def preprocess(x, y):

    preprocess_x, x_tk = tokenize(x)
    preprocess_y, y_tk = tokenize(y)

    preprocess_x = pad(preprocess_x)
    preprocess_y = pad(preprocess_y)

    # Keras's sparse_categorical_crossentropy function requires the labels to be in 3 dimensions
    preprocess_y = preprocess_y.reshape(*preprocess_y.shape, 1)

    return preprocess_x, preprocess_y, x_tk, y_tk
```

In [ ]:
```
preproc_english_sentences, preproc_french_sentences, english_tokenizer, french_tokenizer = pre
process(eng, fr)
```

In [ ]:
```
max_english_sequence_length = preproc_english_sentences.shape[1]
max_french_sequence_length = preproc_french_sentences.shape[1]
english_vocab_size = len(english_tokenizer.word_index)
french_vocab_size = len(french_tokenizer.word_index)

print("Max English sentence length:", max_english_sequence_length)
print("Max French sentence length:", max_french_sequence_length)
print("English vocabulary size:", english_vocab_size)
print("French vocabulary size:", french_vocab_size)
```

```
Max English sentence length: 55

Max French sentence length: 55

English vocabulary size: 14531

French vocabulary size: 30660
```

토큰화, 패딩

—

RNN model

06

In [ ]:
```
def logits_to_text(logits, tokenizer):

    index_to_words = {id: word for word, id in tokenizer.word_index.items()}
    index_to_words[0] = '<PAD>'

    return ' '.join([index_to_words[prediction] for prediction in np.argmax(logits, 1)])
```

We will now create our create our Bidirectional RNN model with Embeddings.

In [ ]:
```
def bd_model(input_shape, output_sequence_length, english_vocab_size, french_vocab_size):

    learning_rate = 0.003

    # Build the layers
    model = Sequential()
    model.add(Embedding(french_vocab_size, 256, input_length=input_shape[1], input_shape=input
_shape[1:]))
    model.add(Bidirectional(GRU(256, return_sequences=True)))
    model.add(TimeDistributed(Dense(1024, activation='relu')))
    model.add(Dropout(0.5))
    model.add(TimeDistributed(Dense(english_vocab_size, activation='softmax')))

    # Compile model
    model.compile(loss=sparse_categorical_crossentropy,
                  optimizer=Adam(learning_rate),
                  metrics=['accuracy'])
    return model
```

In [ ]:
```
tmp_x.shape
```

Out[ ]:
```
(175621, 55)
```

RNN model

최종

loss : 0.2015

accuracy : 0.9557

06

In [ ]:

```
# Reshape the input
tmp_x = pad(preproc_french_sentences, preproc_french_sentences.shape[1])
tmp_x = tmp_x.reshape((-1, preproc_french_sentences.shape[-2]))

# Train
model = bd_model(
    tmp_x.shape,
    preproc_english_sentences.shape[1],
    len(english_tokenizer.word_index)+1,
    len(french_tokenizer.word_index)+1)

model.summary()

model.fit(tmp_x, preproc_english_sentences, batch_size=64, epochs=5, validation_split=0.2)
```

```
Epoch 1/5

2196/2196 [==============================] - 487s 222ms/step - loss: 0.4459 - accuracy: 0.9
324 - val_loss: 0.9272 - val_accuracy: 0.8682

Epoch 2/5

2196/2196 [==============================] - 488s 222ms/step - loss: 0.2934 - accuracy: 0.9
456 - val_loss: 0.8925 - val_accuracy: 0.8726

Epoch 3/5

2196/2196 [==============================] - 486s 221ms/step - loss: 0.2471 - accuracy: 0.9
503 - val_loss: 0.8896 - val_accuracy: 0.8748

Epoch 4/5

2196/2196 [==============================] - 484s 220ms/step - loss: 0.2201 - accuracy: 0.9
534 - val_loss: 0.9097 - val_accuracy: 0.8757

Epoch 5/5

2196/2196 [==============================] - 484s 220ms/step - loss: 0.2015 - accuracy: 0.9
557 - val_loss: 0.9014 - val_accuracy: 0.8753
```

05

In [ ]:

```
i=100076


print("Prediction:")
print(logits_to_text(model.predict(tmp_x[[i]])[0], english_tokenizer))

print("\nCorrect Translation:")
print(eng[i])

print("\nOriginal text:")
print(fr[i])
```

```
Prediction:

we won't have time to do that <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <
PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <P
AD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PA
D> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>



Correct Translation:

We won't have time to do that.



Original text:

Nous n'aurons pas le temps de faire ça.
```

결과

—

실제 정답과 가깝게 문장을 만들어 내는 것을 확인가능

결론 / 소감

이번 기회로 NLP에 대해 알아보기위해 기초부터 차근차근 쌓아올린 지식이 아니라 중간중간 건너띄면서 학습하였기 때문에 기초적인 부분에서도 모르는내용이 많아 어려움을 겪었다.

그래도 새로운 내용을 많이 알아갈수 있었고 이후 계속해서 발표된 학습법 등에 대해서도 궁금증이 생겨났다.

그래서 앞으로 시퀀스 투 시퀀스 이후 발표된 모델에 대해서도 공부해보고 , 중간중간 넘어갔던 지식에 공백도 채워넣어볼 생각이다.

THANK

YOU EVERYONE