자연어처리팀

2팀

한준호

윤세인

김나현

윤여원

권능주

INDEX

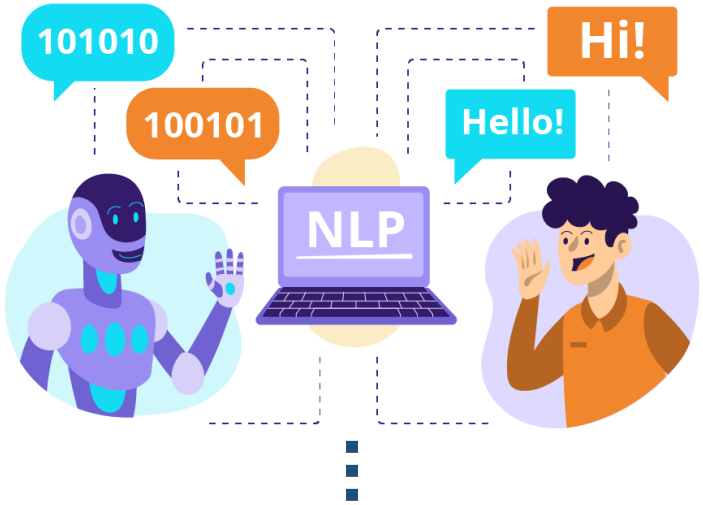
1

자연어

자연어

자연어

의식적인 계획이나
계획 없이 사용, 반복, 변화의 과정을 거쳐
인간 공동체에서 발생한 모든 언어



자연어 처리의 목표는 컴퓨터가 문서 내 언어의
문맥적 뉘앙스를 포함해 문서 내용을 이해할 수 있도록 하는 것

1 자연어

자연어의 특징

순차성

자연어의 가장 대표적인 특징으로
순서가 달라지는 경우 의미가 손상됨

문맥의 중요성

언어가 가진 다의성으로
하나의 단어나 문장이 상황에 따라 다양한 의미로 활용

문장 내 정보 부족

언어는 효율성을 극대화 하기 위해
많은 정보를 생략하는 경우가 빈번하게 발생

자연어의 특징

순차성

자연어의 가장 대표적인 특징으로
순서가 달라지는 경우 의미가 손상됨

문맥의 중요성

나는 밥을 먹었다.

밥이 나를 먹었다.

→ 단어의 순서가 바뀜으로 문장의 의미가 달라짐

자연어의 특징

문맥의 중요성

언어가 가진 다의성으로

하나의 단어나 문장이 상황에 따라 다양한 의미로 활용

순차성

배가 달달하니 맛있네!

과일의 한 종류 배 | 교통수단의 일종인 배

→ 뉘앙스를 통해 뜻을 파악하기 위해 문맥 해석이 중요

자연어의 특징

문장 내 정보 부족

언어는 효율성을 극대화 하기 위해

많은 정보를 생략하는 경우가 빈번하게 발생

문맥의 중요성

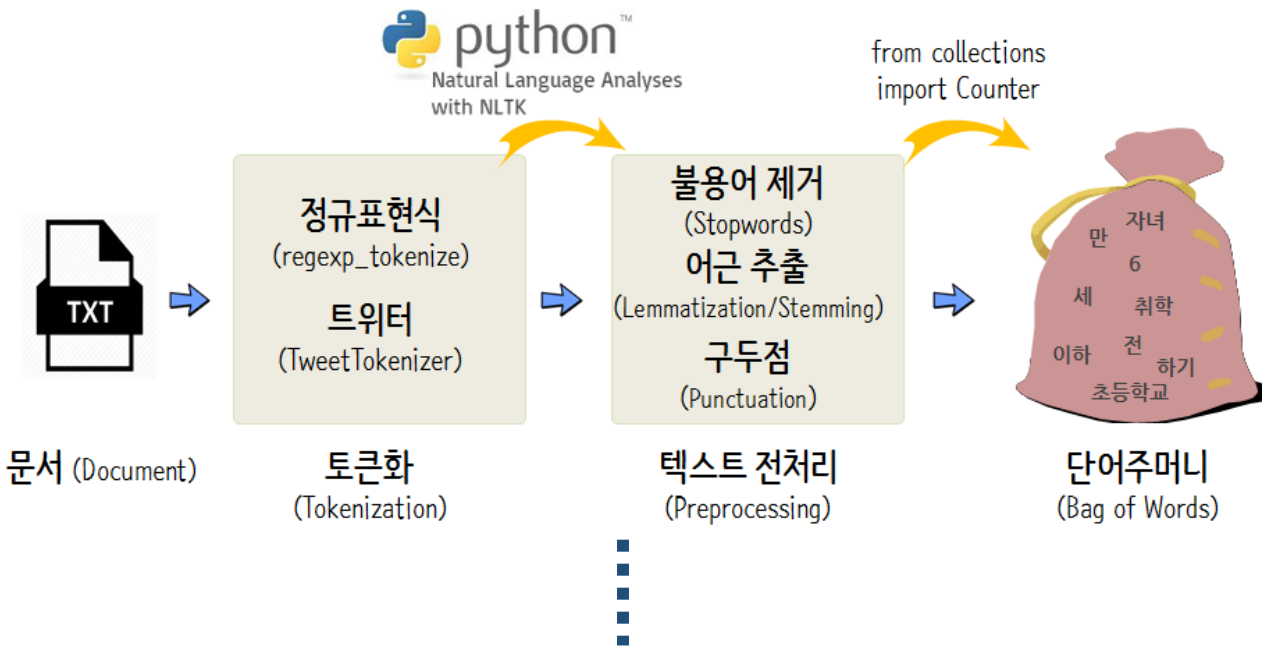
내가 한거 아니야!

무엇을 하지 않았는지 생략

→ 정보의 생략이 많을수록 자연어 처리가 어려워짐

자연어 전처리

자연어 처리의 절차



텍스트 데이터를 모델의 입력으로 사용하기 위해

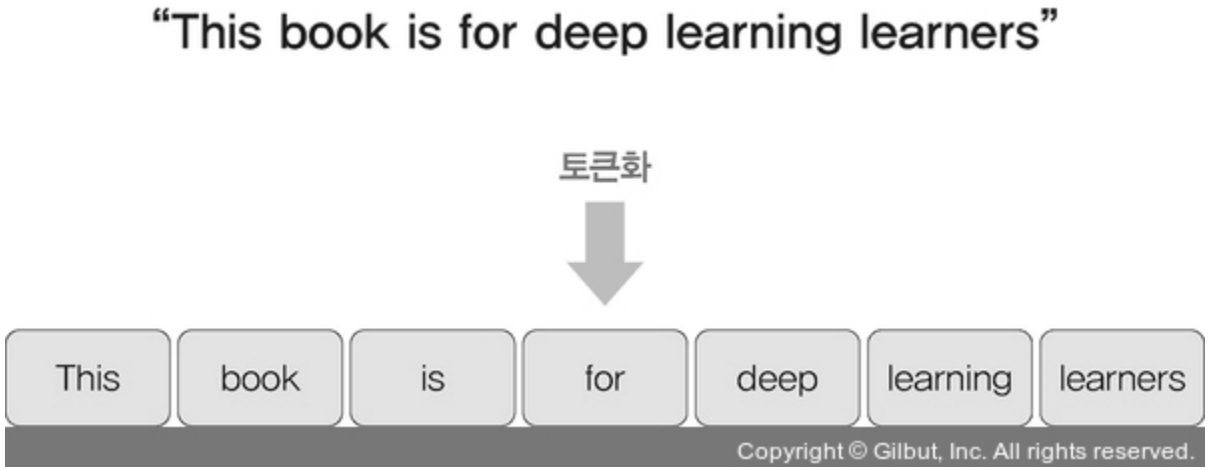
자연어 특성을 반영할 수 있는 변환방법이 선행

자연어 전처리 : 토큰화

토큰화

주어진 말뭉치(corpus)에서
토큰이라 불리는 단위로 나누는 작업

의미 있는 단위로 토큰을 정의



자연어 전처리 : 토큰화

토큰화

주어진 말뭉치(corpus)에서

토큰이라 불리는 단위로 나누는 작업

의미 있는 단위로 토큰을 정의

가장 기본적인 토큰화는 단어 토큰화

단어 외에도 단어구, 문장 토큰화도 가능

명확한 선택 기준이 없어 필요에 따라 적절한 토큰화 선택

자연어 전처리 : 토큰화 방법

“Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp”

세가지 토큰화 방법 사용

공백 기준으로 분리

```
['Hey', 'Amazon', '-', 'my', 'package', 'never', 'arrived',
'https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first',
'PLEASE', 'FIX', 'ASAP!', '@amazonhelp']
```

→ 단순히 공백을 기준으로 분리

자연어 전처리 : 토큰화 방법

“Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp”

세가지 토큰화 방법 사용

keras의 text_to_word_sequence 함수

```
['hey', 'amazon', 'my', 'package', 'never', 'arrived', 'https', 'www',
'amazon', 'com', 'gp', 'css', 'order', 'history', 'ref', 'nav', 'orders',
'first', 'please', 'fix', 'asap', 'amazonhelp']
```

→ 문장 부호와 특수문자를 제거하고 대문자를 소문자로 변환

자연어 전처리 : 토큰화 방법

"Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp"

세가지 토큰화 방법 사용


nltk의 word_tokenize 함수

```
['Hey', 'Amazon', '-', 'my', 'package', 'never', 'arrived', 'https', ':',
'//www.amazon.com/gp/css/order-history', '?', 'ref_=nav_orders_first',
'PLEASE', 'FIX', 'ASAP', '!', '@', 'amazonhelp']
```

→ 문장 부호와 특수문자 토큰화를 포함해 세세하게 분리

자연어 전처리 : 정제

정제

말뭉치(corpus)에서 불필요한
노이즈 데이터를 제거하는 것

- 등장 빈도가 적은 단어 제거
- 길이가 짧은 단어 제거
- 불용어 제거

토큰화 작업 전, 후에 텍스트 데이터를 용도에 맞게
정제 및 정규화 하는 작업을 진행

자연어 전처리 : 정제

“Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp”

등장 빈도가 적은 단어 제거

```
['Hey', 'Amazon', '-', 'my', 'package', 'never', 'arrived', 'PLEASE', 'FIX',
'ASAP', '!', '@', 'amazonhelp']
```

→ 웹사이트 주소는 등장 빈도가 적고, 유의미한 정보가 없어서 제거

자연어 전처리 : 정제

“Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp”

길이가 짧은 단어 제거

```
['Hey', 'Amazon', 'my', 'package', 'never', 'arrived', 'PLEASE', 'FIX', 'ASAP',
'amazonhelp']
```

→ 길이가 짧은 문장 부호와 특수 문자를 제거

1 자연어

자연어 전처리 : 정제

불용어

"Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp"

조사나 접미사처럼 문장 내에서 **자주 등장**하지만

큰 의미를 갖지 않는 단어

✌ 불용어 제거

```
['Hey', 'Amazon', 'my', 'package', 'never', 'arrived', 'PLEASE', 'FIX',
'ASAP', '!', '@amazonhelp']
```

패키지에서 제공하는 **불용어 목록**에 추가로

중요하지 않다고 판단되는 단어를 지정해서 사용

→ "my"를 불용어로 판단해 제거

자연어 전처리 : 정제

“Hey Amazon - my package never arrived https://www.amazon.com/gp/css/order-history?ref_=nav_orders_first PLEASE FIX ASAP! @amazonhelp”

✌ 불용어 제거

```
['hey', 'amazon', 'package', 'never', 'arrived', 'please', 'fix', 'asap',
'amazonhelp']
```

nltk 패키지의 stopwords 목록 사용

→ “my”를 불용어로 판단해 제거

1 자연어

자연어 전처리 : 정규화

정규화

표현 방법이 다른 단어들을 통합해서
같은 단어로 만들어주는 것

대소문자 통일

```
['hey', 'amazon', 'package', 'never', 'arrived', 'please', 'fix', 'asap',
'amazonhelp']
```

→ 대문자와 소문자는 의미 차이를 만들지 않는 경우 통일해서 사용

해당 예시의 경우 Amazon은 회사 명칭을 의미하지만
amazon은 다른 의미를 갖기 때문에 주의

자연어 전처리 : 정규화

표제어 추출

```
['hey', 'amazon', 'package', 'never', 'arrive', 'please', 'fix', 'asap',
'amazonhelp']
```

nltk 패키지의 WordNetLemmatizer 사용

→ 표제어 추출을 통해 arrived -> arrive로 변환

어간 추출

```
['hey', 'amazon', 'packag', 'never', 'arriv', 'pleas', 'fix', 'asap',
'amazonhelp']
```

nltk 패키지의 PorterStemmer 사용

→ 어간 추출을 통해 arrive -> arriv로 변환

자연어 전처리 : 정규화

정제와 정규화

토큰화 -> 정제 -> 정규화 과정을 거쳐

쓸만한 토큰을 생성하고

필요에 따라 **특정 부분 생략** 가능

✌ 표제어 추출

```
['hey', 'amazon', 'packag', 'never', 'arriv', 'pleas', 'fix', 'asap',
'amazonhelp']
```

nltk 패키지의 WordNetLemmatizer 사용

→ 표제어 추출을 통해 arrived -> arrive로 변환

✌ 어간 추출

```
['hey', 'amazon', 'packag', 'never', 'arriv', 'pleas', 'fix', 'asap',
'amazonhelp']
```

nltk 패키지의 PorterStemmer 사용

→ 어간 추출을 통해 arrive -> arriv로 변환

정제 과정은 토큰화 이전과 이후에 지속적으로 진행

하지만 완벽한 정제가 불가능하기 때문에

적절한 합의점 선택

2

단어 표현 방법

2 단어 표현 방법

희소 표현(Sparse representation)

희소 표현(Sparse representation)

벡터 또는 행렬의 값이 대부분 0으로 표현되는 방법

이산적으로 단어의 의미 표현

Book과 Books같은 단어의 변형 형태도 다른 단어로 간주

2 단어 표현 방법

One-Hot Encoding

One-Hot vector

N개의 단어를 N차원의 벡터로 나타낸 것

John likes to watch movies. Mary likes movies too.

John	1	0	0	0	0	0	0
likes	0	1	0	0	0	0	0
to	0	0	1	0	0	0	0
watch	0	0	0	1	0	0	0
movies	0	0	0	0	1	0	0
mary	0	0	0	0	0	1	0
too	0	0	0	0	0	0	1

단어의 존재 여부에 따라 1 또는 0으로 값이 정해짐

2 단어 표현 방법

One-Hot Encoding

One-Hot vector

N개의 단어를 N차원의 벡터로 나타낸 것

John likes to watch movies. Mary likes movies too.

John	1	0	0	0	0	0	0
likes	0	1	0	0	0	0	0
to	0	0	1	0	0	0	0
watch	0	0	0	1	0	0	0
movies	0	0	0	0	1	0	0
mary	0	0	0	0	0	1	0
too	0	0	0	0	0	0	1

단어 집합에 포함되는 단어가 많아질수록 차원이 기하급수적으로 커짐

→ 매우 심한 메모리 낭비

단어의 존재 여부에 따라 1 또는 0으로 값이 정해짐

2 단어 표현 방법

Bag of Words(BoW)

Bag of Words(BoW)

문서 내 단어들에 대해 각 단어의 등장 빈도를 세는 방법

단어들의 순서를 전혀 고려하지 않음

```
John likes to watch movies. Mary likes movies too.
```

```
{"John":0, "likes":1, "to":2, "watch":3, "movies":4, "Mary":5, "too":6}
```

```
Word to Frequencies: [1, 2, 1, 1, 2, 1, 1]
```

단어에 인덱스 부여 후 각 단어의 빈도 수 계산

2 단어 표현 방법

Document-Term Matrix(DTM)

Document-Term Matrix(DTM)

문서가 여러 개일 때, 각 문서마다 단어 빈도를 행렬로 나타내는 방법

```
Document1: John likes to watch movies. Mary likes movies too.

Document2: I think that movie was awful, but John likes that movie.

Document3: John likes Mary.
```

	John	likes	to	watch	movies	Mary	too	I	think	that	movie	was	awful
Doc1	1	2	1	1	2	1	1	0	0	0	0	0	0
Doc2	1	1	0	0	0	0	0	2	1	2	2	1	1
Doc3	1	1	0	0	0	1	0	0	0	0	0	0	0

위의 예시처럼 문서 별 빈도 수 행렬을 만들 수 있음

2 단어 표현 방법

Document-Term Matrix (DTM)

Document-Term Matrix (DTM)

문서가 여러 개일 때, 각 문서마다의 단어 빈도를 행렬로 나타내는 방법

```
Document1: John likes to watch movies. Mary likes movies too.
Document2: I think that movie was awful, but John likes that movie.
Document3: John likes Mary.
```

3개의 문서

	John	likes	to	watch	movies	Mary	too	I	think	that	movie	was	awful
Doc1	1	2	1	1	2	1	1	0	0	0	0	0	0
Doc2	1	1	0	0	0	0	0	1	1	1	2	1	1
Doc3	1	1	0	0	0	1	0	0	0	0	0	0	0

문서 별 빈도수 행렬

DTM의 문제점

불용어처럼 자주 나오는 단어라면,
단어의 중요도에 상관없이 높은 빈도수 가짐

중요도가 낮은 단어들에 대해
또 다른 조치 취해주는 것이 옳은 선택

2 단어 표현 방법

TF-IDF(Term Frequency-Inverse Document Frequency)

TF-IDF(Term Frequency-Inverse Document Frequency)

단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취함)를 곱하는 방식

↳ DTM 내 각 단어들마다 중요한 정도를 가중치로 줄 수 있음

역 문서 빈도 (IDF)

$$idf(t) = log\frac{n}{1 + df(t)}$$

t : 문서 내의 단어 수

n : 전체 문서 수

....

분모가 0이 되지 않기 위해 1을 더함

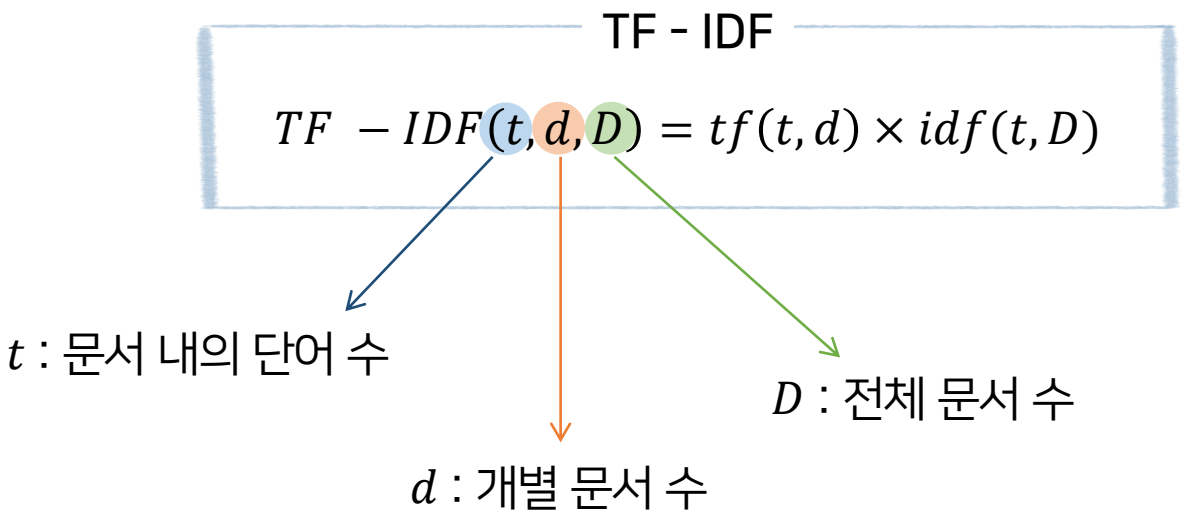
과도한 가중치를 주는 것을 방지하기 위해 로그를 취함

2 단어 표현 방법

TF-IDF(Term Frequency-Inverse Document Frequency)

TF-IDF(Term Frequency-Inverse Document Frequency)

단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취함)를 곱하는 방식



2 단어 표현 방법

희소 단어 표현의 문제점

고차원 문제

희소 표현은 전체 단어 집합 크기만큼의 차원을 가지는 벡터 생성

많은 단어가 존재하는 경우, 매우 고차원의 벡터 생성

→ 계산 비용 기하급수적 증가 & 차원의 저주로 효율적 학습 X

공간의 낭비(Sparse Matrix)

희소 표현 방법에서는 대부분의 값이 0이 됨

즉, 대부분은 정보를 가지고 있지 않음

→ 텍스트 데이터 정제와 불용어 처리 과정에서 어느 정도 보완 가능

but 여전히 심한 공간의 낭비는 불가피

2 단어 표현 방법

희소 단어 표현의 문제점

단어 간 의미 관계 부재

One-Hot Encoding & Bow 방법 : 단어 존재 유무, 단어 빈도 수를 정수형으로 나타냄

임의로 부여한 숫자로, 단어 간의 의미적 관계가 포함되어 있지 않음

→ 모델이 단어 간의 의미적 유사성을 학습하는 데 제약 발생

2 단어 표현 방법

밀집 표현(Dense Representation)

밀집 표현(Dense Representation)

단어 벡터의 차원을 단어 집합의 크기로 상정하지 않고,

사용자가 설정한 값으로 모든 단어의 벡터 표현의 차원을 맞추는 방식

밀집 표현 과정에서 실수 값을 가지게 됨

강아지 = [0.2 1.8 1.1 -2.1 1.1 2.8 ... 중략 ···]

↓

의미 부여 가능한 128 차원의 벡터로 표현

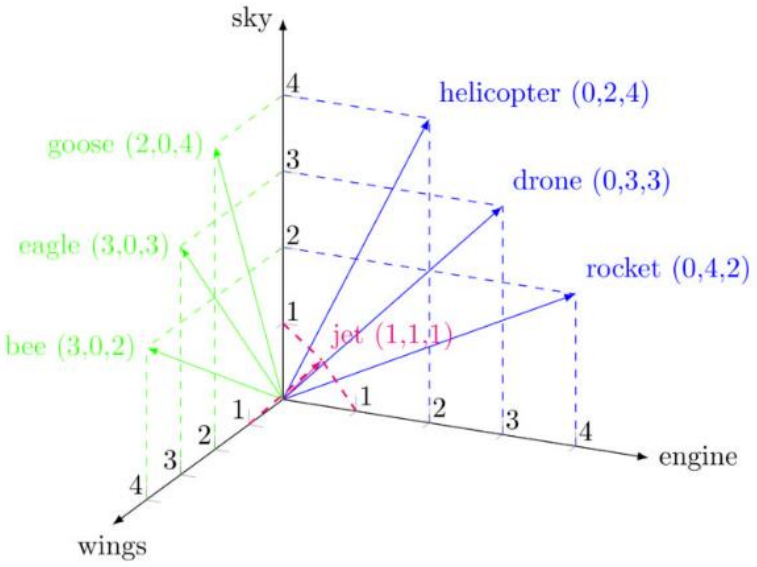
2 단어 표현 방법

모델 기반 Word Embedding

모델 기반 Word Embedding

각 단어를 인공 신경망 학습을 통해 벡터화하는 Word Embedding 방법

Word Embedding은 단어를 밀집 벡터의 형태로 표현하는 것



단어들을 저차원 공간의 실수 값의 점으로

간단히 표현 & 의미 유지 가능

2 단어 표현 방법

모델 기반 Word Embedding

Word Embedding은 어떻게 계산할까?

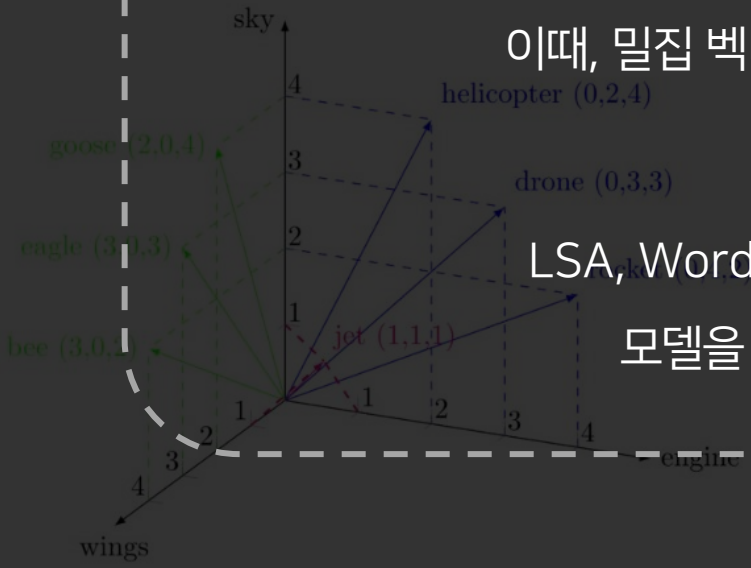
단어를 랜덤한 값을 가지는 밀집 벡터로 변환한 후

인공 신경망의 가중치를 학습하는 것과 같은 방식으로 단어 벡터 학습

이때, 밀집 벡터의 차원은 사용자가 설정

LSA, Word2Vec, FastText, Glove

모델을 통해 학습된 값 사용



wings, engine, sky를 축으로 하는

3차원의 공간에 한 점으로 단어들을 표현

즉, Word Embedding의 차원이 3인 결과

단어들을 저차원 공간의 실수 값의 점으로

간단히 표현 & 의미 유지 가능

2 단어 표현 방법

Word2Vec

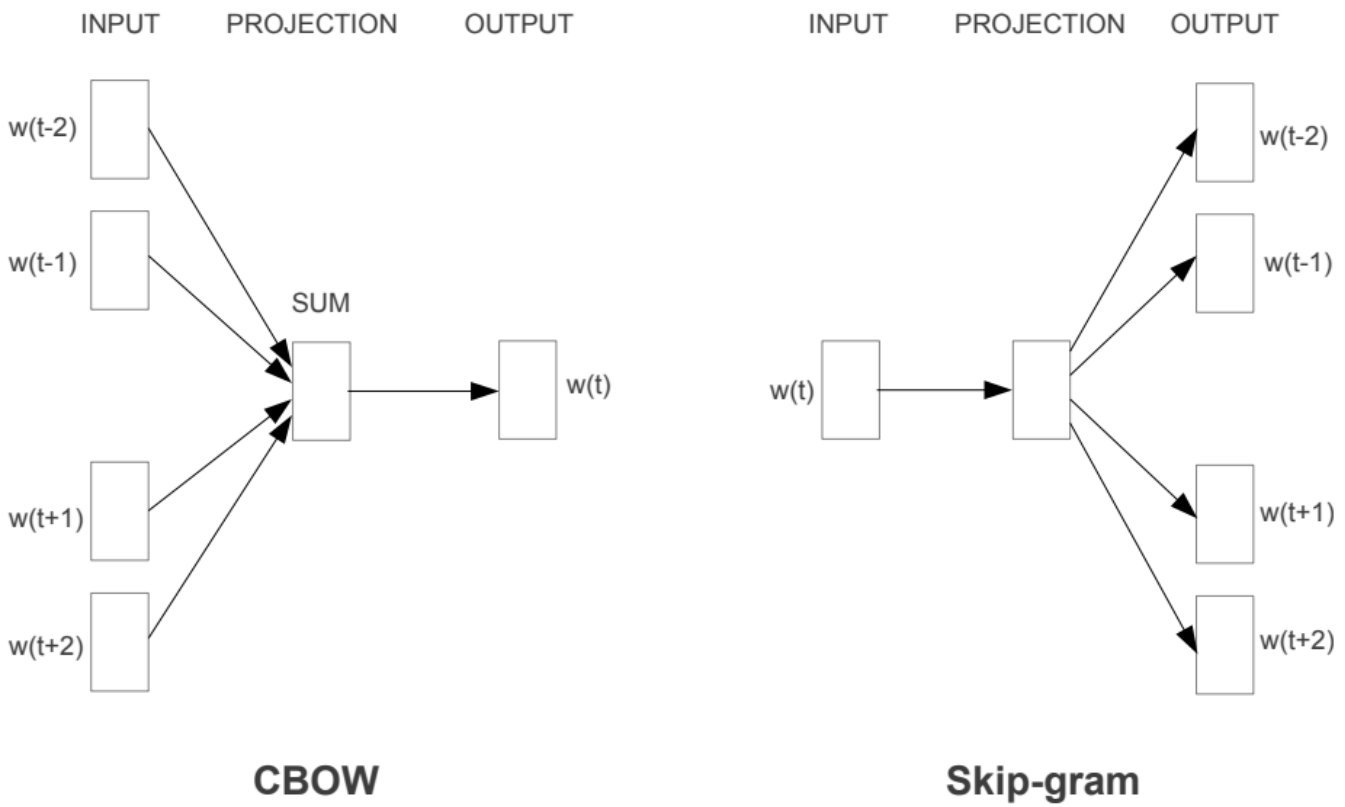
Word2Vec

단어를 밀집(dense) 벡터로 표현하는 방법 중 하나로,
임베딩 벡터를 학습하기 위해 주변 단어의 문맥을 활용

자연어 처리에서 단어의 의미를 효과적으로 포착하기 위해 사용
주변 문맥을 어디까지 활용할 것인지 사용자가 지정해야 함

2 단어 표현 방법

Word2Vec



주변 문맥 학습 방법에 따라 CBOW 모델과 Skip-gram 모델로 나뉨

2 단어 표현 방법

Word2Vec

CBOW : 주변 단어 문맥으로 중심 단어를 예측하는 방식

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log p(w_t|w_{t+j})$$

Skip-gram : 중심 단어에서 주변 단어를 예측하는 방식

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log p(w_{t+j}|w_t)$$

→ 각각의 확률을 최대로 하는 단어를 찾도록 학습

일반적으로 CBOW 모델에 비해,

Skip-gram 모델로 학습한 임베딩의 성능이 더 좋은 것으로 알려짐

2 단어 표현 방법

Word2Vec

CBOW : 주변 단어 문맥으로 중심 단어를 예측하는 방식

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log p(w_t|w_{t+j})$$

학습을 위한 데이터셋은 무엇을 사용하는 것이 좋을까?

BoW 방법을 응용해

같이 나오는 단어들의 빈도 수를 저장한 행렬인

co-occurrence matrix를 데이터셋으로 하여 학습

→ 각각의 확률을 최대로 하는 단어를 찾도록 학습

Skip-gram : 중심 단어에서 주변 단어를 예측하는 방식

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log p(w_{t+j}|w_t)$$

일반적으로 CBOW 모델에 비해,

Skip-gram 모델로 학습한 임베딩의 성능이 더 좋은 것으로 알려짐

2 단어 표현 방법

Co-occurrence matrix 예시

Document1: Apples are green and red.

Document2: Red apples are sweet.

Document3: Green oranges are sour.

Document4: Monkeys eat red apples.

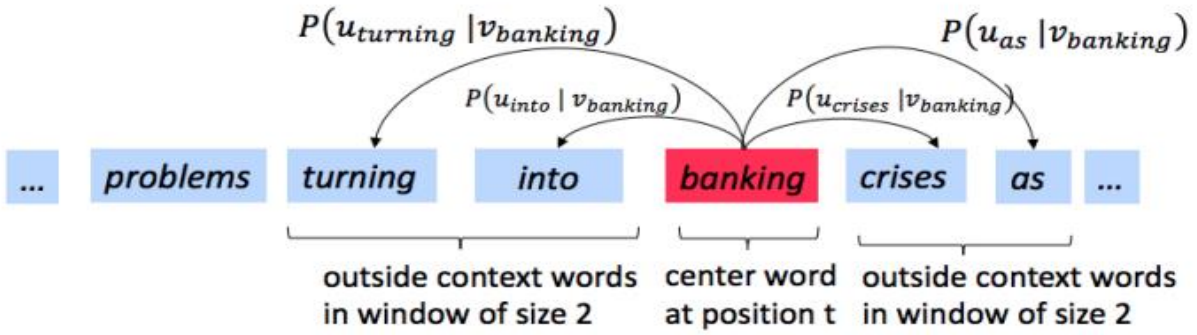
-	apples	are	green	and	red	sweet	oranges	sour	monkeys	eat
apples	3	2	1	1	3	1	0	0	1	1
are	2	3	2	1	2	1	1	1	0	0
green	1	2	2	1	1	0	1	1	0	0
and	1	1	1	1	1	0	0	0	0	0
red	3	2	1	1	3	1	0	0	1	1
sweet	1	1	0	0	1	1	0	0	0	0
oranges	0	1	1	0	0	0	1	1	0	0
sour	0	1	1	0	0	0	1	1	0	0
monkeys	1	0	0	0	1	0	0	0	1	1
eat	1	0	0	0	1	0	0	0	1	1

Skip-gram

Skip-gram

중심 단어를 기준으로 주변의 단어 예측

앞 뒤로 볼 단어의 범위를 윈도우라고 함



Skip-gram (window size = 2)

2 단어 표현 방법

Skip-gram

주변 단어들의 확률 분포

$$P(O = o \mid C = c) = \frac{\exp(\boldsymbol{u}_0^T \mathbf{v}_c)}{\sum_{w \in Vocab} \exp(\boldsymbol{u}_W^T \mathbf{v}_c)}$$

→ 이 확률을 최대화하는 방향으로 학습

$$\mathbf{J}(\mathbf{v}_c, o, \mathbf{U}) = -logP(O = o \mid C = c)$$

-1을 곱하여 손실을 양의 값으로 바꾸고, 최대화 하는 방향으로 학습 가능

SGNS(Skip-gram with Negative Sampling)

SGNS(Skip-gram with Negative Sampling)

좀 더 빠른 계산을 위해 Skip-gram에 Negative Sampling 기법을 추가한 모델

Negative Sampling

Word2Vec이 학습 과정에서 전체 단어 집합이 아니라
일부 단어 집합에만 집중할 수 있도록 하는 방법

2 단어 표현 방법

Negative Sampling

$$P(O = o \mid C = c) = \frac{\exp(\boldsymbol{u_0^T v_c})}{\sum_{w \in Vocab} \exp(\boldsymbol{u_w^T v_c})}$$

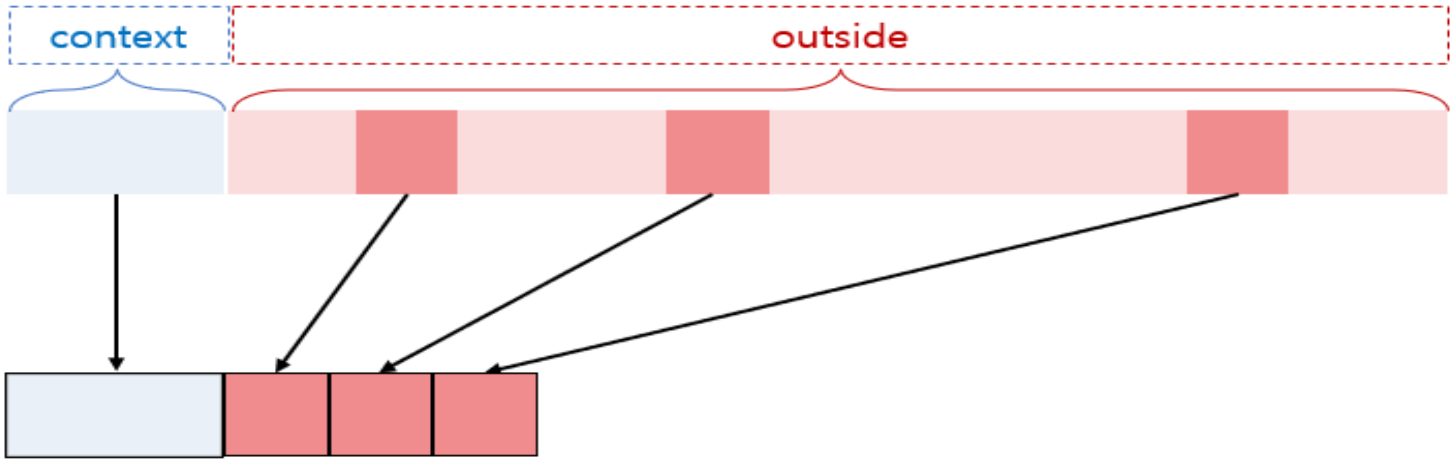
중심 벡터의 전체 벡터에 대한 dot product 계산

일반적인 경우, 전체 단어 집합에 대해 계산하므로

단어 집합이 커질수록 연산 양이 훨씬 더 많아짐

2 단어 표현 방법

Negative Sampling



전체 단어 집합에는 속하지만 주변 단어에는 속하지 않는 단어 중,

랜덤하게 샘플링해 부분 단어 집합을 만듦

전체 집합 대신 부분 집합에 대해 계산

2 단어 표현 방법

SGNS(Skip-gram with Negative Sampling)

손실 함수

$$\mathbf{J}(\mathbf{v_c}, \mathrm{o}, \mathbf{U}) = -log(\sigma(\boldsymbol{u_0^T}\mathbf{v_c}) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u_{W_s}^T}\mathbf{v_c}))$$

내적 계산 시 -1을 곱하여 outside vector를 포함하지 않는 방향으로 학습함

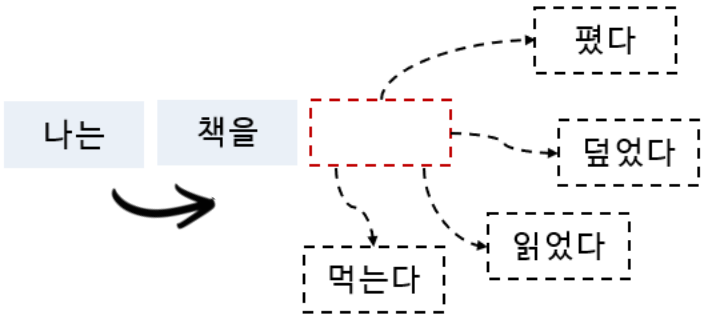
3

언어 모델링

언어 모델링

언어 모델링(Language Modeling)

언어라는 현상을 모델링하고자 문장(단어 시퀀스)에 확률을 할당하는 모델



단어 시퀀스에 확률을 할당하기 위해 이전 단어들로부터 다음 단어 예측

3 언어 모델링

언어 모델링

언어 모델링(Language Modeling)

언어라는 현상을 모델링하고자 **문장(단어 시퀀스)에 확률을 할당**하는 모델

즉, 이전 단어들을 입력으로 **조건부 확률**을 생성하는 것

$$P(Y) = P(y_1, \dots, y_t) = \prod_{t=1}^{\tau} P(y_t | y_{t-1}, \dots, y_1)$$

- 통계적 언어 모델(SML)
- 인공 신경망 기반 모델(NML)

3 언어 모델링

통계적 언어 모델(SML)

카운트 기반 접근 모델

$x^{(t+1)}$은 오직 이전 t개의 항에 의존한다는 마르코프 가정에 기반함

N-gram

전체를 보는 것이 아닌 이전의 (N-1)개의 단어만 보는 방법

통계적 언어 모델(SML) – 카운트 기반 접근 모델

카운트 기반 접근 모델

$x^{(t+1)}$은 오직 이전 t개의 항에 의존한다는 마르코프 가정에 기반함

N-gram

전체를 보는 것이 아닌 이전의 (N-1)개의 단어만 보는 방법

Ex) P (나는 밥을 먹고 다시 학교로 가서 공부를 했다)

= P (나는) X P (밥을 | 나는) X …

X P (했다 | 나는 밥을 먹고 다시 학교로 가서 공부를)

계산

$$\frac{\text{count (나는 밥을 먹고 다시 학교로 가서 공부를 했다)}}{\text{count (나는 밥을 먹고 다시 학교로 가서 공부를)}}$$

3 언어 모델링

통계적 언어 모델(SML) – 카운트 기반 접근 모델

카운트 기반 접근 모델

$x^{(t+1)}$은 오직 이전 t개의 word에 의존한다는 **마르코프 가정**에 기반함

N-gram

"만약 카운트가 0이 된다면?"

분모가 0인 경우: 분모에 매우 작은 수(ε)를 더함

Ex) 분자가 0인 경우: 더할 수 없으므로, 했다)

훈련 데이터에서 실제 count가 없다면 확률은 0

X P (했다 | 나는 밥을 먹고 다시 학교로 가서 공부를)

계산

count (나는 밥을 먹고 다시 학교로 가서 공부를)

데이터가 매우 드물게 분포되어 있어 모델 학습에 어려움을 겪는 **희소 문제** 발생

통계적 언어 모델(SML) – N-gram

카운트 기반 접근 모델

$x^{(t+1)}$은 오직 이전 t개의 항에 의존한다는 **마르코프 가정**에 기반함

N-gram

전체를 보는 것이 아닌 **이전의 (N-1)개의 단어만 보는 방법**

~~나는 밥을 먹고~~ 다시 학교로 가서 공부를 ____

무시! 4개의 단어만 고려!

N-gram(n = 5)의 예시

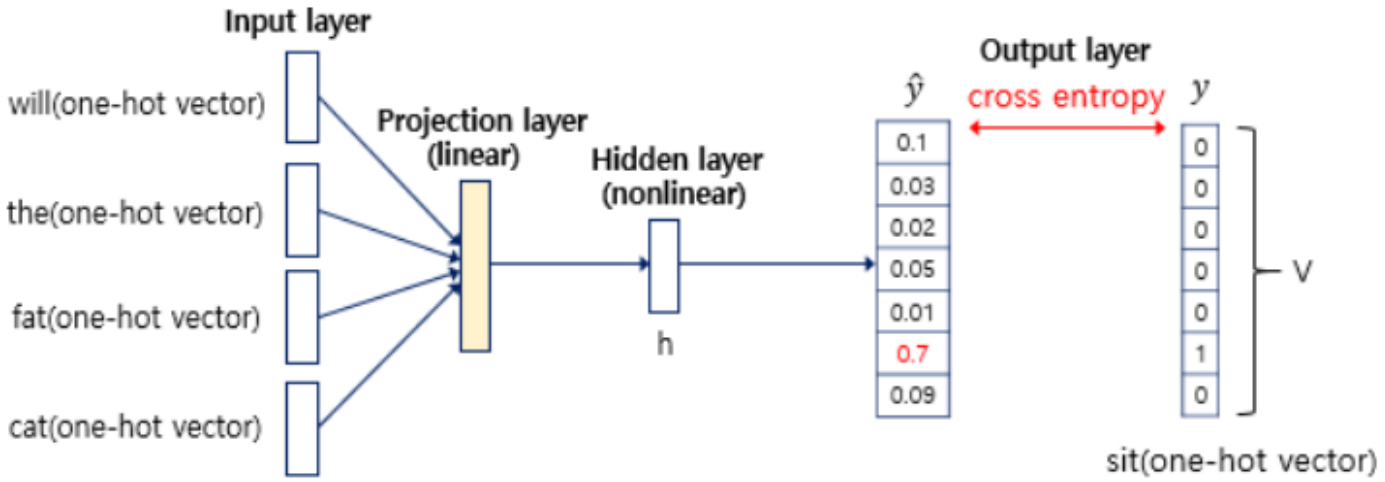
이 역시 **희소 문제**를 피할 수 없었음

3 언어 모델링

인공 신경망 기반 언어 모델(NML)

일반적인 인공 신경망(Feed-forward)

밀집 표현인 임베딩을 학습함으로써 해결함



3 언어 모델링

인공 신경망 기반 언어 모델(NML)

카운트 기반 접근 모델

$x^{(t+1)}$은 오직 이전 t개의 word에 의존한다는 마르코프 가정에 기반함

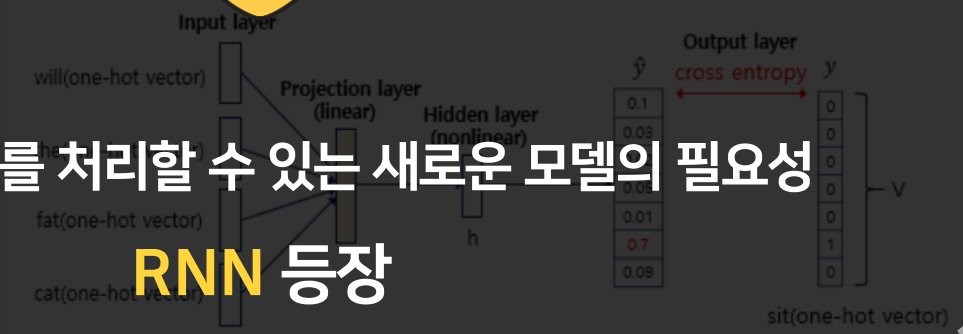
N-gram

전체를 보는 것이 아닌 이전의 (N-1)개의 단어만 보는 방법

Feed-forward 신경망의 문제점

고정된 길이의 단어 시퀀스만을 입력으로 받기 때문에
자연어 처리를 위한 방법으로는 한계점이 존재함

일반적인 인공 신경망 (Feed-forward)



밀집 표현인 임베딩을
학습함으로써 해결함

가변적인 길이를 처리할 수 있는 새로운 모델의 필요성

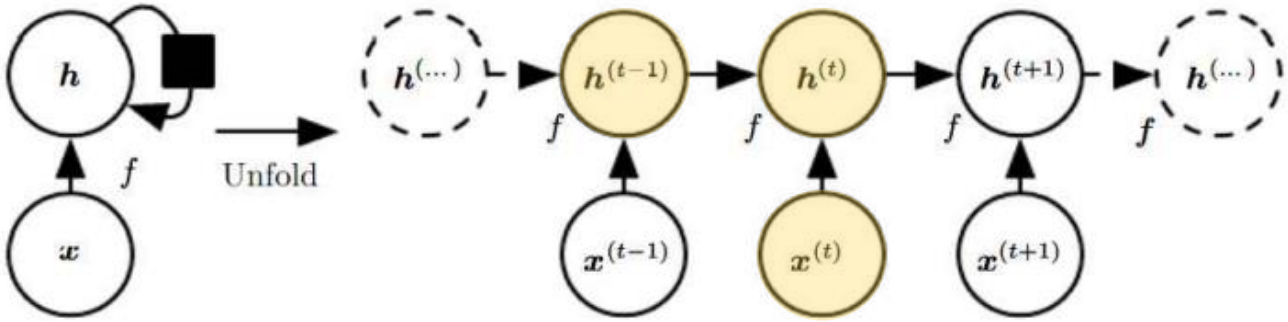
RNN 등장

3 언어 모델링

Vanilla RNN

일반적인 신경망과 RNN의 차이점

모델의 Output이 Input뿐만 아니라 이전의 Output에도 영향을 받음



왼: 모델의 순환성 / 오: 시점에 따른 학습으로 펼친 모습

t 시점에는 t번째 인풋, t-1시점의 아웃풋을 입력으로 받음

기존 인공 신경망과 달리 가변적 길이의 단어 시퀀스를 입력으로 받음

3 언어 모델링

RNN의 연산

3개의 가중치 행렬

③ V (Hidden-to-Output)

t 시점 은닉층 값과의 연산을 통해

t 시점의 아웃풋이 만들어짐

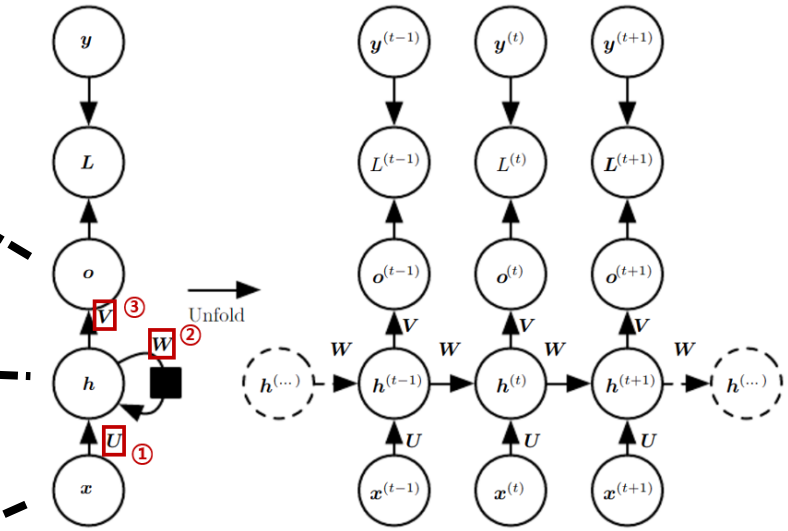
② W (Hidden-to-Hidden)

이전 시점에서 다음 시점으로 은닉층의 가중치 전달

즉, **매 시점마다 동일한 가중치 행렬 사용**

① U (Input-to-Hidden)

인풋과 연산한 후 은닉층으로 전달



RNN의 연산

연산식

1) $\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}$

2) $\mathbf{h}^{(t)} = tanh(\mathbf{a}^{(t)})$

3) $\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$

4) $\hat{\mathbf{y}}^{(t)} = softmax(\mathbf{o}^{(t)})$

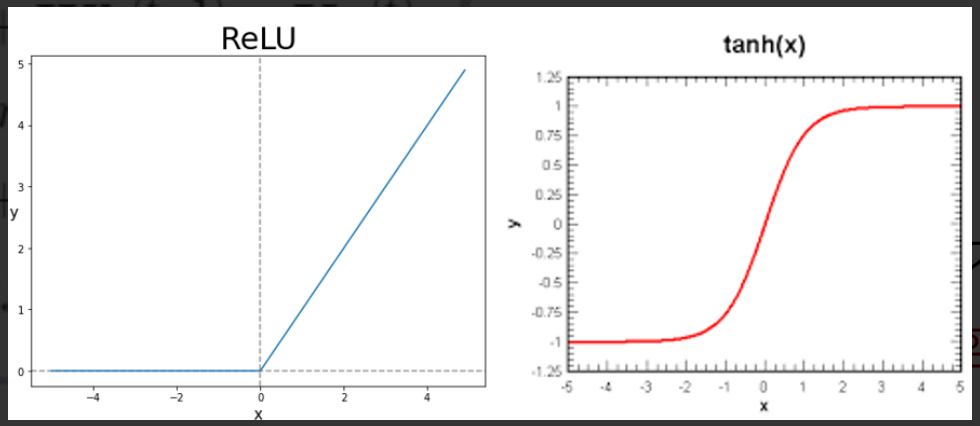
은닉층 가중치 / 아웃풋으로 바꿔주기 위해 활성화 함수 사용

매개변수들이 시점 t에 의존하고 있지 않음

즉, 순전파 과정에서 매개변수는 모든 시점에서 공유되며 변하지 않음

RNN의 연산

RNN에서 ReLU 함수를 사용하지 않는 이유



/ 아웃풋으로

기 위해

함수 사용

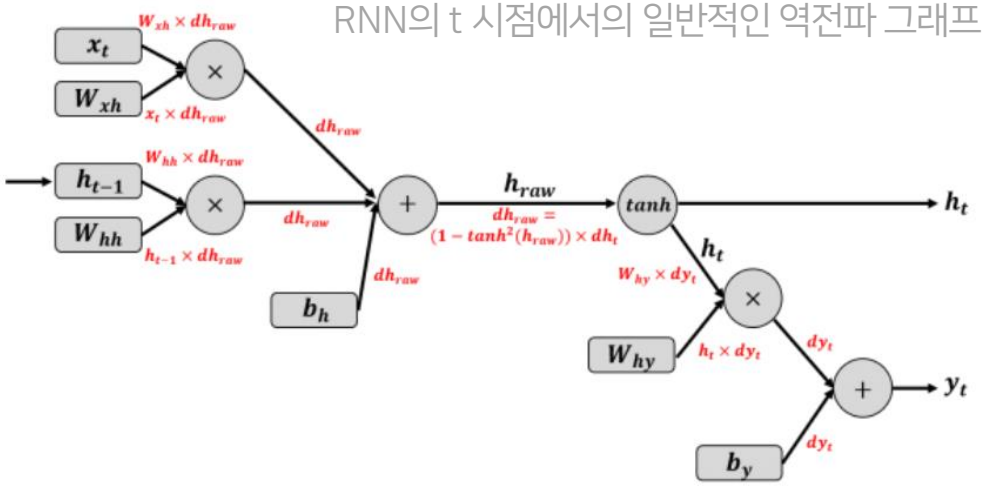
ReLU는 절댓값 1 이상의 값을 가질 수 있음

순환적인 RNN의 특성 상 **발산의 가능성**을 만들어주기에 사용하지 않음

3 언어 모델링

RNN 역전파

RNN의 t 시점에서의 일반적인 역전파 그래프



RNN은 시점에 따라 다른 출력이 존재

순전파 과정이 끝나 모든 출력이 만들어진 다음

마지막 출력부터 역전파가 일어나야 함

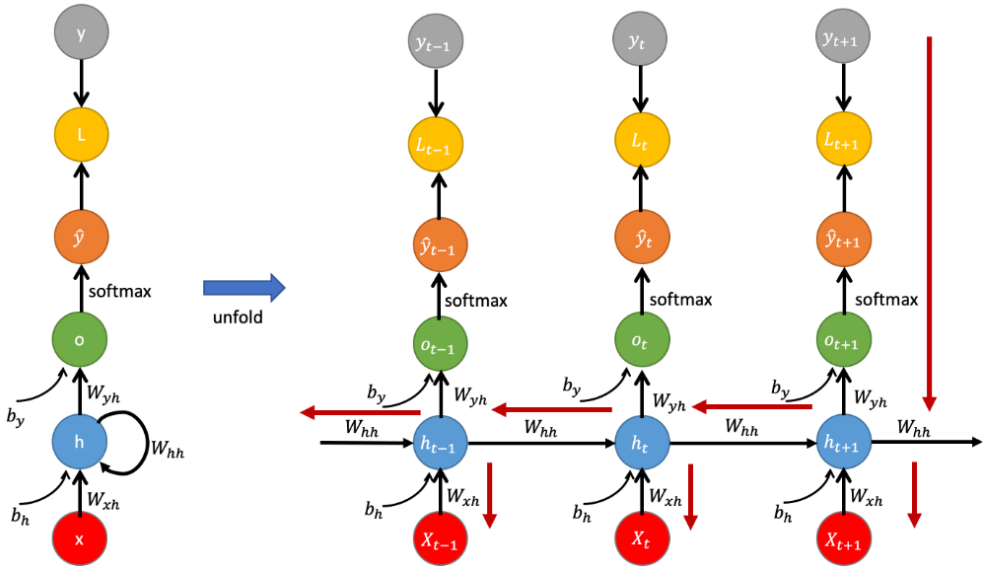
각 시점마다 은닉층 상태를 메모리에 저장하고 있어야 함

BPTT 사용

BPTT(Back Propagation through Time)

BPTT (Back Propagation through Time)

기울기가 매 시점마다 계산되어 손실 값 또한 매 시점 계산되는 형태의 역전파



모델을 시점 단위로 펼쳐서 마지막 시점부터 첫 번째 시점까지 업데이트

Truncated BPTT

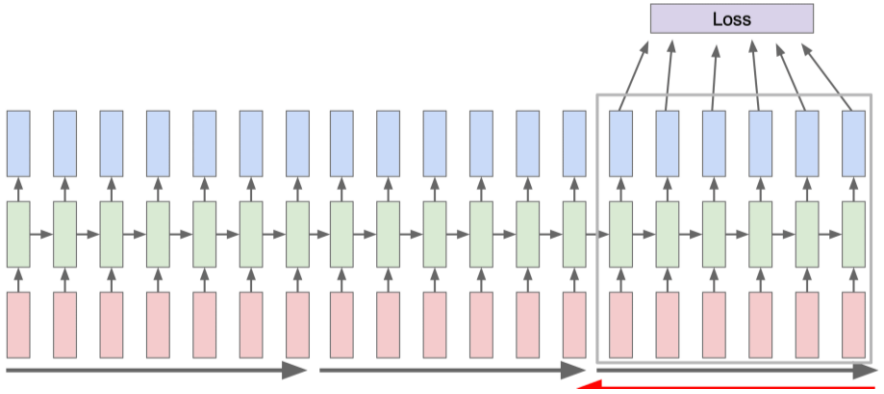
Sequential Data의 길이가 길어지면, BPTT는 더 오래 걸림

값이 1보다 크게/작게 되면 첫 시점으로 갈수록 미분 값이 커짐/작아짐

즉, 과거 시점에 대한 예측이 거의 불가능함

Truncated BPTT

특정 시점 단위로 끊어서 기울기를 계산하는 형태의 역전파



RNN의 한계점 – 1) 매우 긴 학습 시간

출력 결과가 이전의 계산 결과에 의존함

즉, 이전의 계산 결과가 끝나야지만 다음 단어 시퀀스에 계산이 가능하므로

loop를 통해 계산할 수밖에 없음

Ex) 나는 밥을 먹고 다시 학교로 가서 공부를 했다

단어 8개 = 8번의 loop

매우 느린 학습 시간 초래

RNN의 한계점

BPTT를 통해 미분값을 계산할 때의 문제점

시퀀스가 길어짐에 따라

기울기 < 1

: 매우 작은 기울기 값이 됨

기울기 > 1

: 훨씬 더 큰 기울기 값이 됨

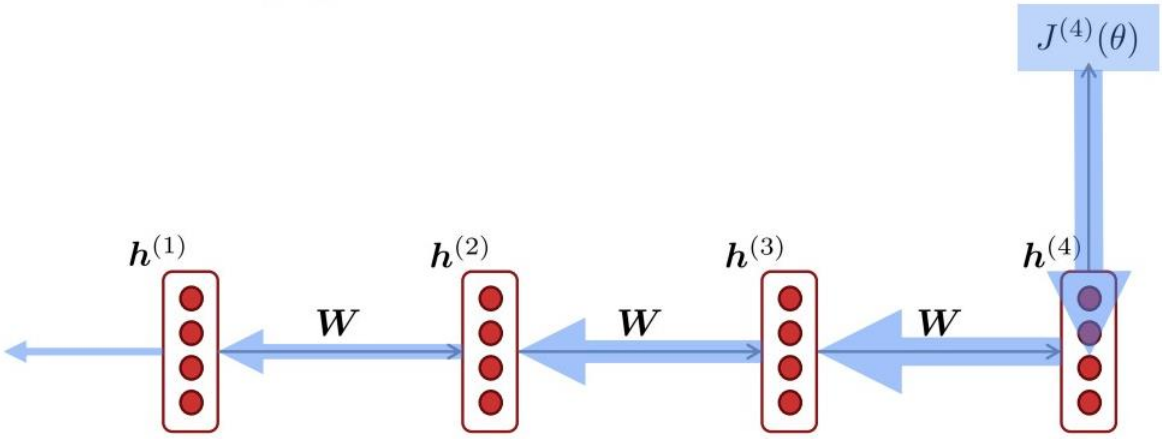
시점이 멀어짐에 따라 정보가 제대로 반영되지 못 함

즉, **장기 의존성 문제**가 발생함

RNN의 한계점 – 2) 장기 의존성 문제

장기 의존성 문제

은닉층의 과거의 정보가 마지막까지 제대로 전달되지 못 하는 현상



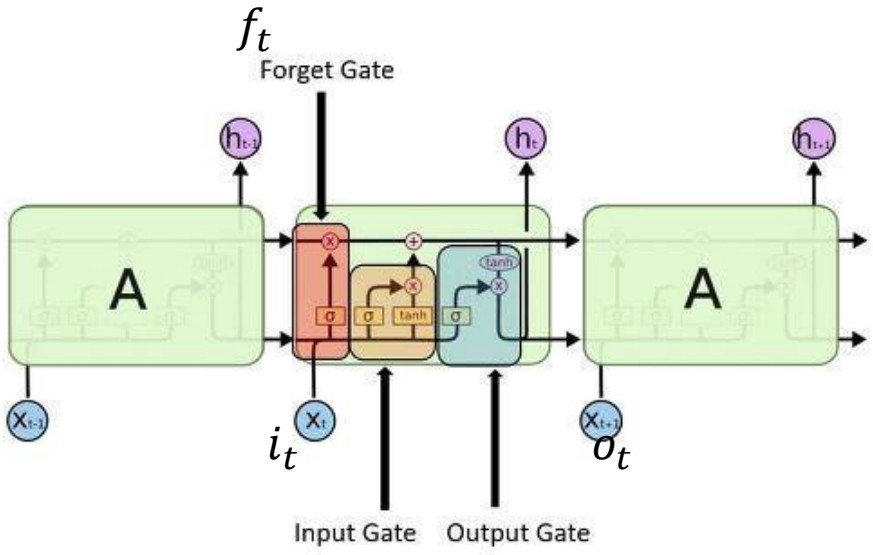
가까운 시점의 데이터 위주로 학습이 진행됨

LSTM (Long Short-Term Memory)

LSTM

단기 기억을 h_t(hidden state), 장기 기억을 c_t(cell state)

각각 담당해 **두 개의 hidden vector**를 갖는 모델



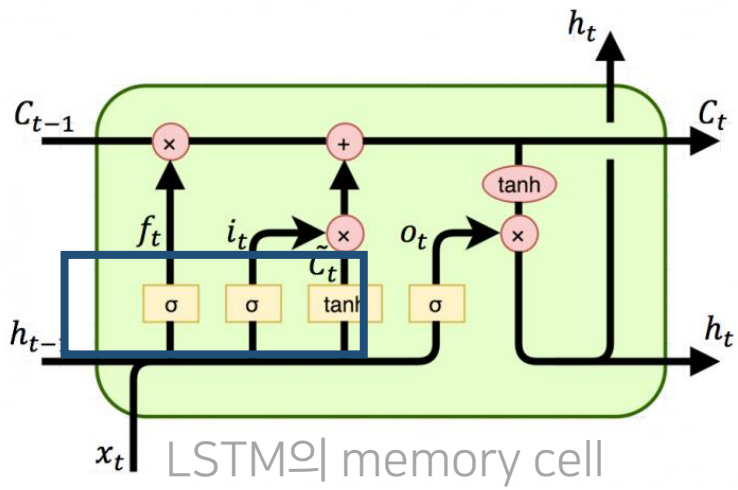
LSTM의 구조

LSTM (Long Short-Term Memory)

LSTM

단기 기억을 h_t(hidden state), 장기 기억을 c_t(cell state)

각각 담당해 **두 개의 hidden vector**를 갖는 모델



LSTM의 memory cell

3개의 게이트를 통해 어떤 정보를 **읽고, 지우고, 쓸 것인지 결정함**

LSTM의 연산 – 1) 게이트 연산

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$
$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$
$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

RNN의 연산과 동일하며

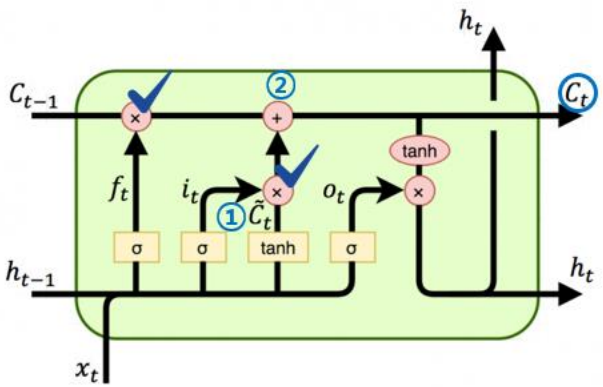
가중치는 시점과 무관하게 존재함

0과 1 사이의 값을 갖는 Sigmoid 함수를 적용함으로써

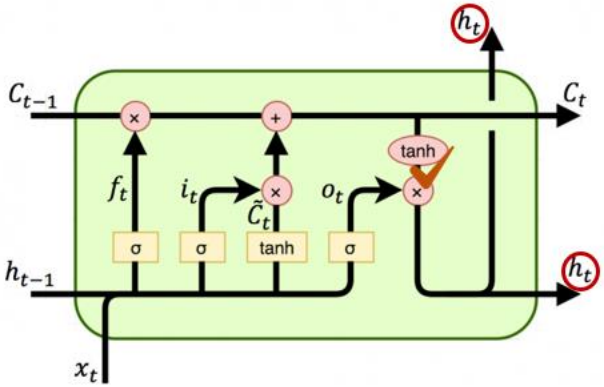
정보를 얼마나 반영하는지에 대한 가중치로 작용함

LSTM의 연산 – 2) 셀 상태 & 은닉 상태 연산

$$① \vec{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$
$$② \boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \vec{\boldsymbol{c}}^{(t)}$$
$$③ \boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{c}^{(t)})$$



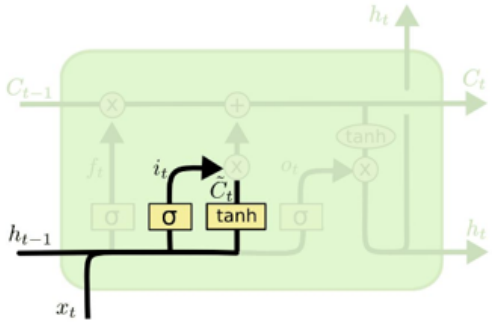
Cell State



Hidden State

LSTM의 연산 – 2) 셀 상태 & 은닉 상태 연산

New Cell content

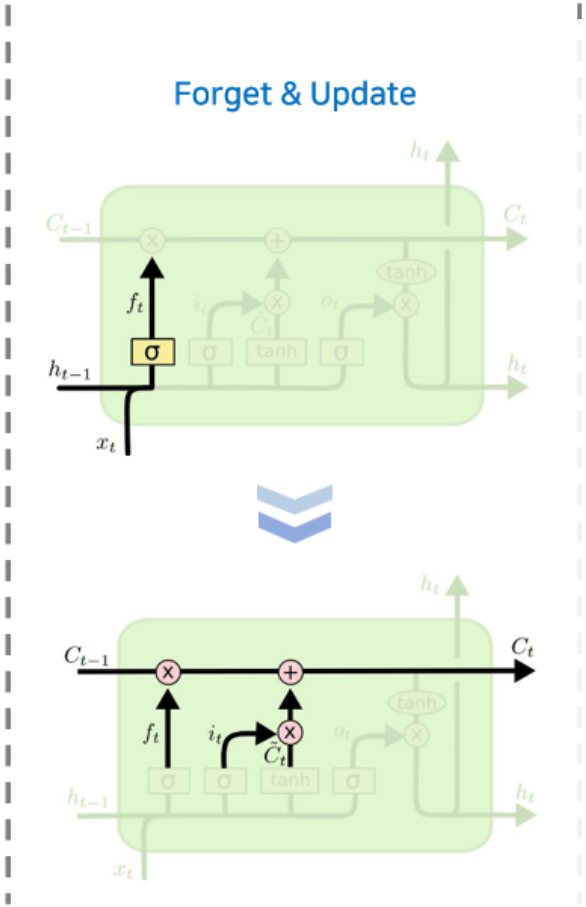


① 새로운 cell content를 만드는 과정

이전 시점의 hidden state와 현재 시점의 입력으로 연산

LSTM의 연산 – 2) 셀 상태 & 은닉 상태 연산

② 현재 시점의 cell state를 만드는 과정



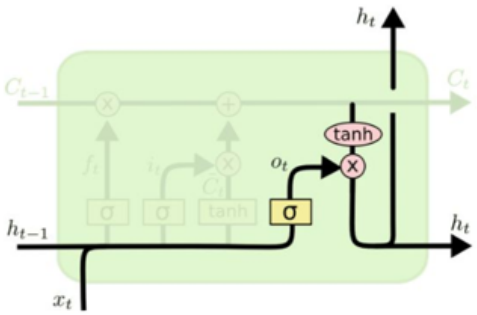
이전 상태의 정보를
얼마나 잊고
현재 상태의 정보를
얼마나 반영할지 결정

LSTM의 연산 – 2) 셀 상태 & 은닉 상태 연산

New Cell content

Forget & Update

Output gate



③ 새로운 hidden state를 만들어내는 과정

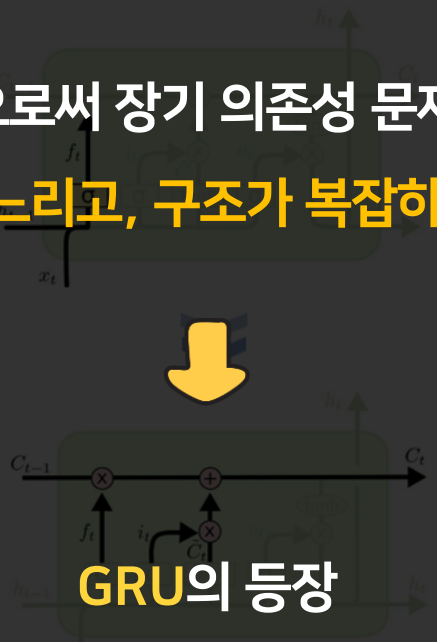
Output gate의 결과물과

현재 시점의 cell state를 곱함

LSTM의 연산

LSTM의 문제점

New Cell content | Forget & Update | Output gate



Memory cell을 사용함으로써 장기 의존성 문제를 어느 정도 해결했으나

여전히 연산 속도가 느리고, 구조가 복잡하다는 단점이 존재함

GRU의 등장

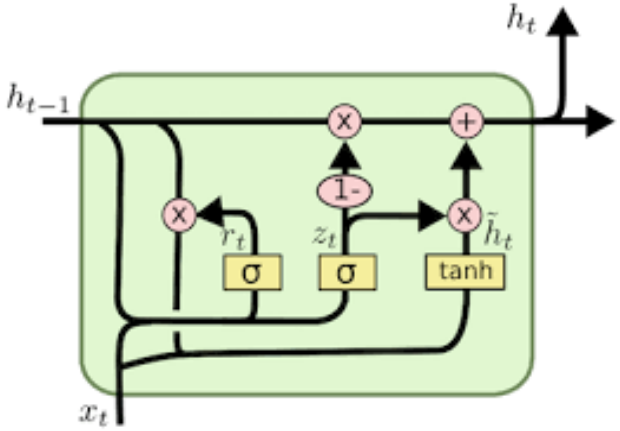
GRU (Gate Recurrent Unit)

✓ **LSTM vs GRU**

LSTM: 3개의 게이트(input, forget, output)

↓

GRU: 2개의 게이트(update, reset)로 줄어들어 구조가 간단해짐



일반적으로 GRU의 학습 속도가 더 빠르다고 알려져 있으나, 성능은 비슷함

Other RNNs

Bidirectional RNN

이전 시점의 정보만을 활용하는 Vanilla RNN과
달리 양방향으로 RNN을 학습시키는 구조

Multi-layer RNN

Neural Network를 여러 층으로 쌓아
RNN에 더 복잡한 구조를 학습 시키는 방법

Other RNNs

Bidirectional RNN

이전 시점의 정보만을 활용하는 Vanilla RNN과
달리 양방향으로 RNN을 학습시키는 구조

RNN 대신 LSTM, GRU로 대체 가능

The movie was terribly exciting!

exciting이 terribly의 의미를 부정에서 긍정으로 전환

단방향 RNN의 경우 문맥에 따른 의미 변화를 파악하기 어려워
terribly를 부정적 의미로만 파악

Other RNNs

Bidirectional RNN

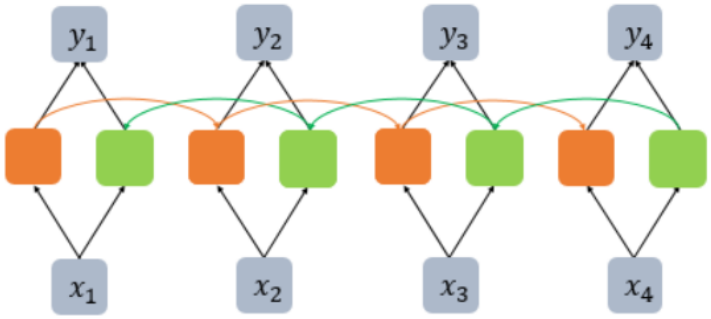
이전 시점의 정보만을 활용하는 Vanilla RNN과
달리 양방향으로 RNN을 학습시키는 구조

RNN 대신 LSTM, GRU로 대체 가능

The movie was terribly exciting!

exciting이 terribly의 의미를 부정에서 긍정으로 전환

→ 양방향 학습을 통해 의미 변화를 포착



Other RNNs

Bidirectional RNN

이전 시점의 정보만을 활용하는 Vanilla RNN과
달리 양방향으로 RNN을 학습시키는 구조

RNN 대신 LSTM, GRU로 대체 가능

Forward RNN

$$\vec{h}^{(t)} = RNN_{FW}(\vec{h}^{(t-1)}, x^{(t)})$$

Backward RNN

$$\overleftarrow{h}^{(t)} = RNN_{BW}(\overleftarrow{h}^{(t-1)}, x^{(t)})$$

Concatenated RNN

$$h^{(t)} = [\vec{h}^{(t)}; \overleftarrow{h}^{(t)}]$$

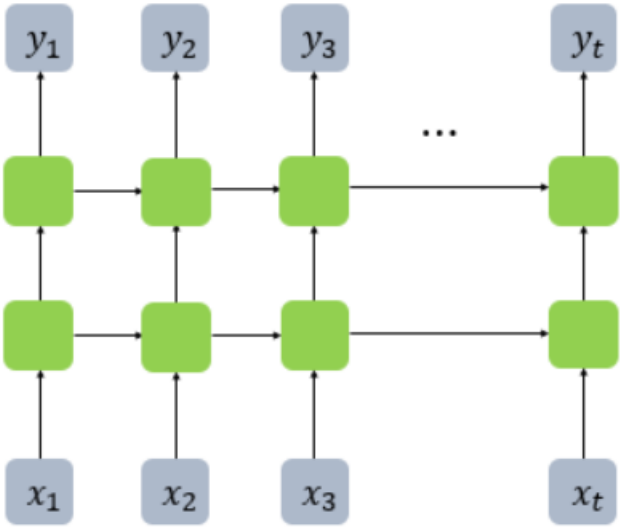
Other RNNs

Multi-layer RNN

Neural Network를 여러 층으로 쌓아

RNN에 더 복잡한 구조를 학습 시키는 방법

RNN 대신 LSTM, GRU로 대체 가능



3 언어 모델링

Other RNNs

언어 모델링의 성능 평가는 어떻게 할까?

언어 모델링에 다양한 모델이 존재

해당 모델들을 비교하기 위해서 평가지표가 필요

회귀문제에서 사용하는 RMSE나

분류문제에서 사용하는 정확도와 같은

평가 방법인 PPL 사용

3 언어 모델링

Perplexity(PPL)

Perplexity(PPL)

언어 모델링에서 모델 자체적으로 자신의 성능을 수치화하기 위해 사용하는 지표

문장의 길이로 정규화 된 문장 확률의 역수

'헷갈리는 정도'를 의미

PPL이 낮을수록 언어 모델의 성능이 좋음

3 언어 모델링

Perplexity(PPL)

PPL

$$PPL(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}} = \sqrt[N]{\frac{1}{\pi_{i=1}^{N} P(w_i | w_1, w_2, \dots, w_N)}}$$

N-gram 적용 (N = 5)

$$PPL(W) = \sqrt[N]{\frac{1}{\pi_{i=5}^{N} P(w_i | w_{i-4}, w_{i-3}, w_{i-2}, w_{i-1})}}$$

Perplexity(PPL)

PPL

문장 W의 길이가 N일 때,

$$PPL(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}} = \sqrt[N]{\frac{1}{\pi_{i=1}^{N} P(w_i | w_1, w_2, \dots, w_N)}}$$

문장 내에서 평균적으로 몇 개의 선택지를 가지고 고민하는지를 의미

$$PPL(W) = \sqrt[N]{\frac{1}{\pi_{i=5}^{N} P(w_i | w_{i-4}, w_{i-3}, w_{i-2}, w_{i-1})}}$$

PPL 계산 예시

$$A : PPL = 20 \ / \ B : PPL = 30$$

⋮

문장의 모든 시점마다

모델 A는 20개, 모델 B는 30개의

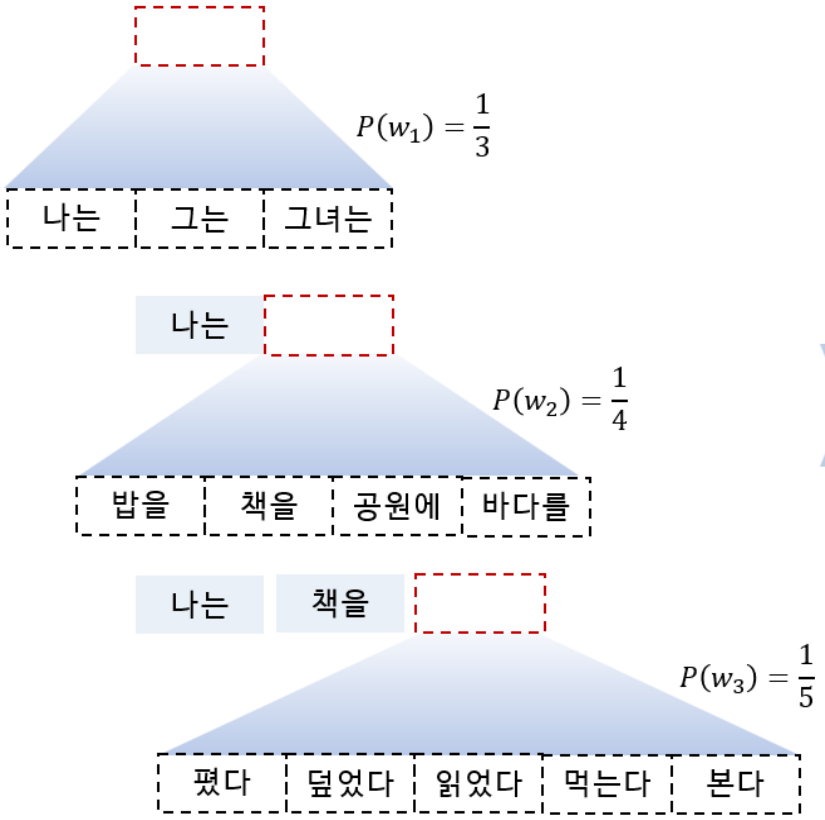
선택지 고민

↓

모델 A가 B에 비해 덜 헷갈려 하므로

A가 더 성능이 좋음

PPL 계산 예시



실제 정답 : “나는 책을 읽었다 "

$$PPL = 4$$

4

기계 번역

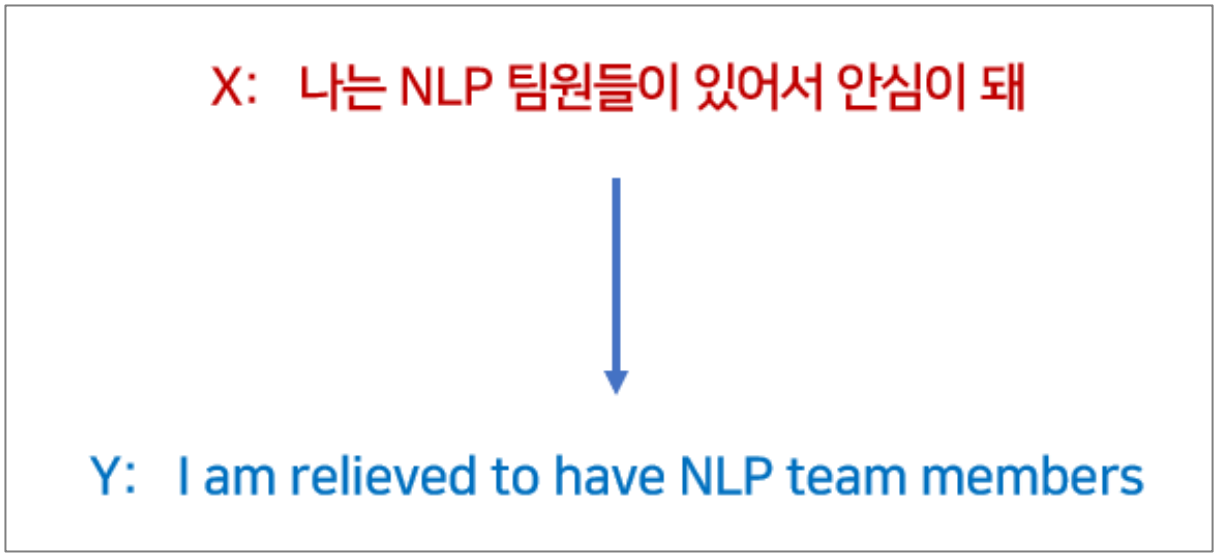
4 기계 번역(Machine Translation)

기계 번역

기계 번역

어떤 문장 X를 다른 문장 Y로 번역하는 것

언어 모델링과 더불어 자연어 처리 분야에서 대표적인 과제 중 하나 !



4 기계 번역(Machine Translation)

기계 번역의 주요 이슈

대응되는 단어의 부재

대응되는 단어가 없는 경우

→ 한국어에는 조사가 있지만, 영어에는 그렇지 않음

나는 NLP 팀원들이 있어서 안심이 돼!

는, 이 등과 같은 조사는 영어에 존재하지 않음

4 기계 번역(Machine Translation)

기계 번역의 주요 이슈

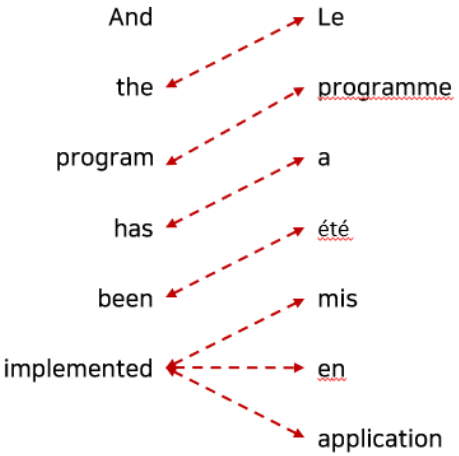
정렬(Alignment)의 문제점

대응 관계에서 정렬에 대한 문제점 →

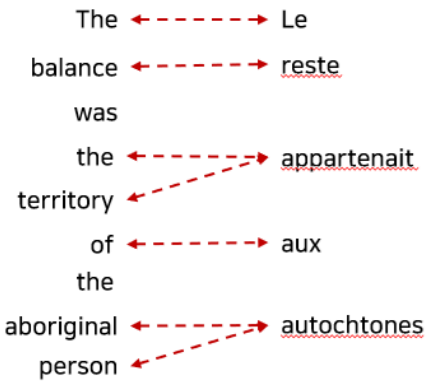
어떤 단어는 하나의 단어에만 대응 (one-to-one)

어떤 단어는 여러 개에 대응 (one-to-many)

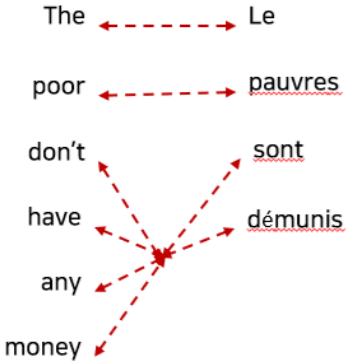
여러 개의 단어가 여러 개에 대응 (many-to-many)



<one-to-many>



<many-to-one>

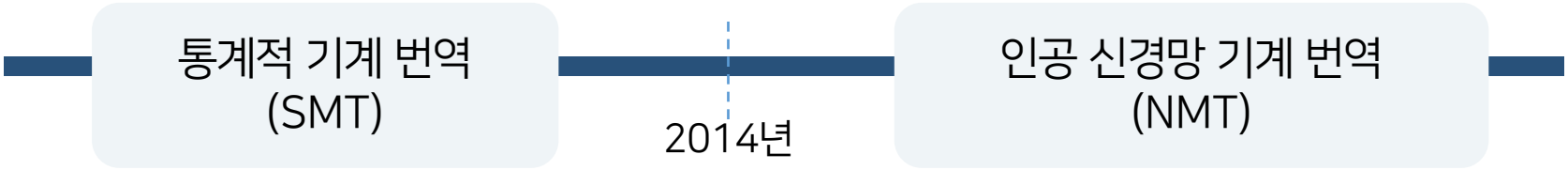


<many-to-many>

4 기계 번역(Machine Translation)

기계 번역의 종류

기계 번역에는 통계적 기계 번역과 인공 신경망 기계 번역이 있음



4 기계 번역(Machine Translation)

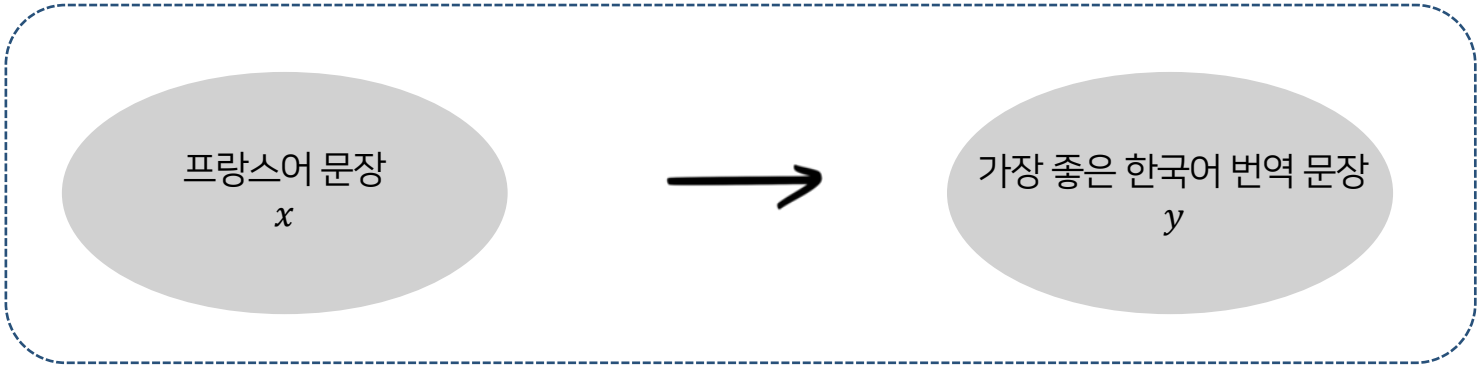
SMT(Statistical Machine Translation)

SMT(Statistical Machine Translation)

데이터로부터 확률적 모델을 학습하는

통계적 접근 방법

"프랑스어를 한국어로 번역해보자!"



4 기계 번역(Machine Translation)

SMT(Statistical Machine Translation)

이때, 베이지안 법칙에 기반하여 두 가지 문제로 쪼갤 수 있음

$$argmax_y P(y|x) = argmax_y (P(x|y)P(y))$$

$P(x|y)$: 한국어 문장이 주어졌을 때, 프랑스어 문장으로 번역하는 방법에 대해 학습하는 번역 모델(MT)

$P(y)$: 좋은(유창한) 한국어 문장을 만들어내는 것을 목표로 하는 언어 모델(LM)

그러나, SMT는 큰 성과를 거두지 못 함.

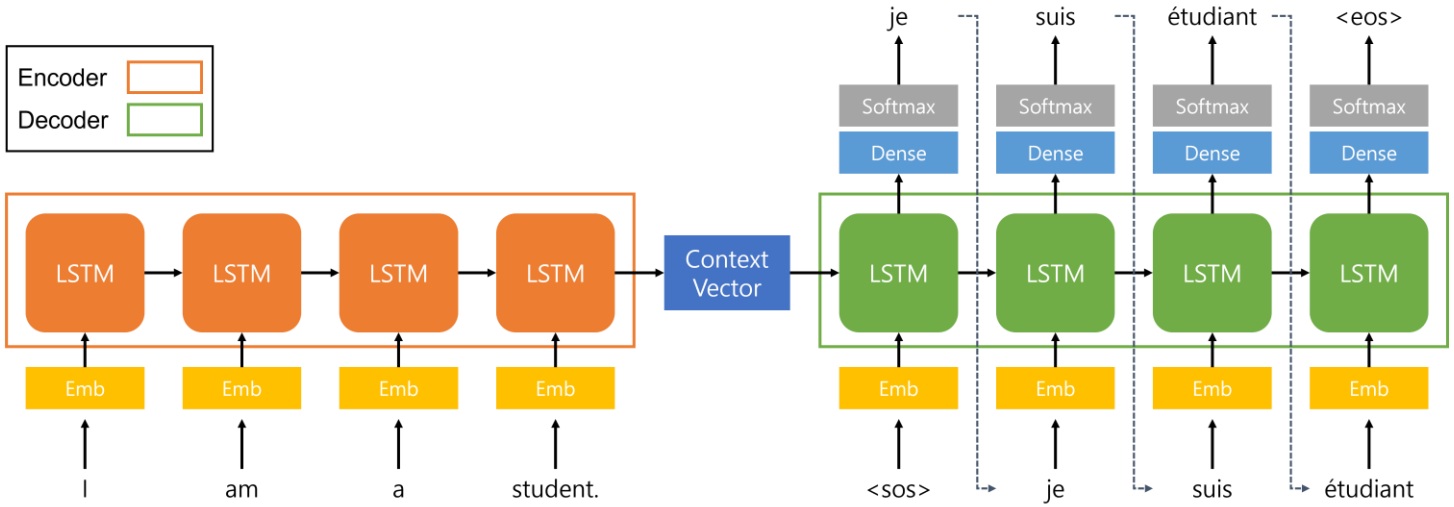
→ 2014년, 하나의 end-to-end 모델로 기계번역을 하는 NMT(Neural Machine Translation)이 등장

4 기계 번역(Machine Translation)

NMT(Neural Machine Translation)

NMT(Neural Machine Translation)

인공 신경망을 사용하는 기계 번역 접근 방법

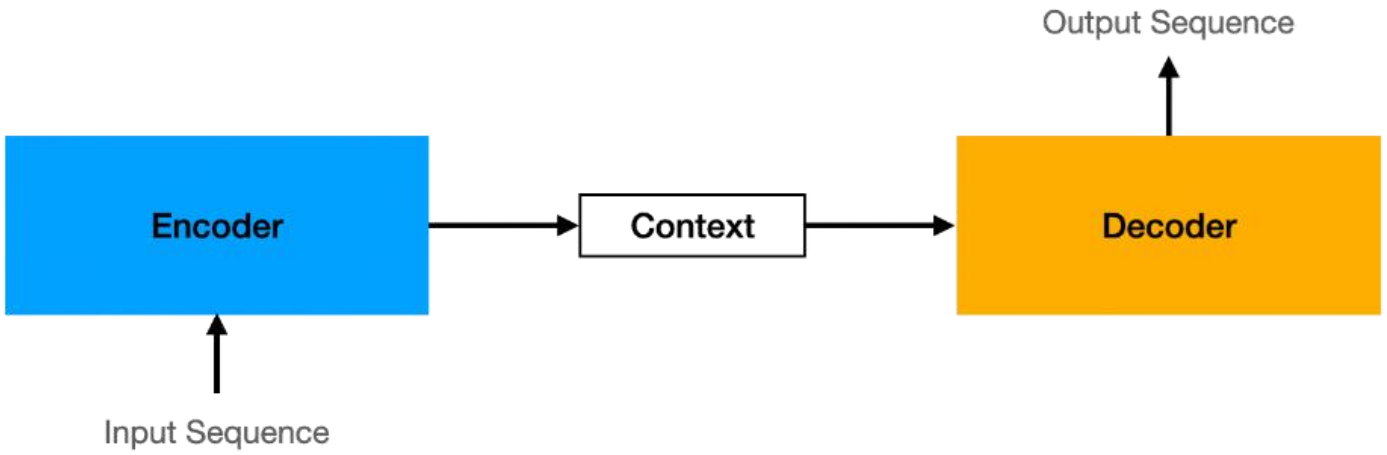


하나의 end-to-end 모델로 기계 번역이 가능함

: 입력에서 출력까지 신경망으로 한 번에 처리

4 기계 번역(Machine Translation)

Encoder-Decoder



원래의 문장을 압축하여

하나의 벡터로 만듦

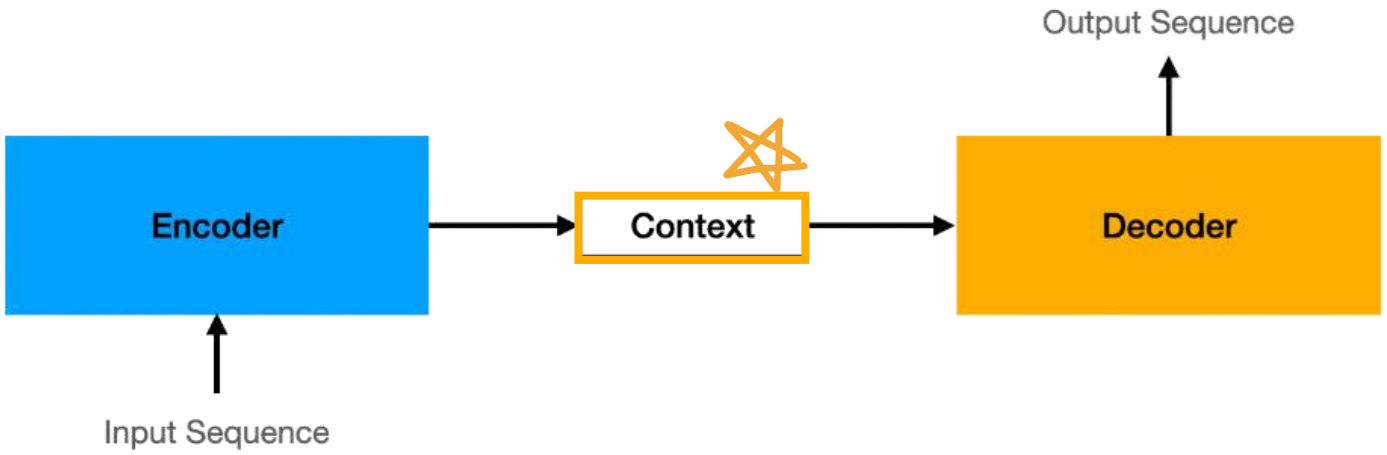
context vector

만들어진 벡터를 바탕으로

새로운 문장으로 복원

4 기계 번역(Machine Translation)

Encoder-Decoder



Context vector는 Encoder의 마지막 hidden state 값이 됨

이 context vector를 받아서

Decoder의 첫 번째 hidden state로 사용

4 기계 번역(Machine Translation)

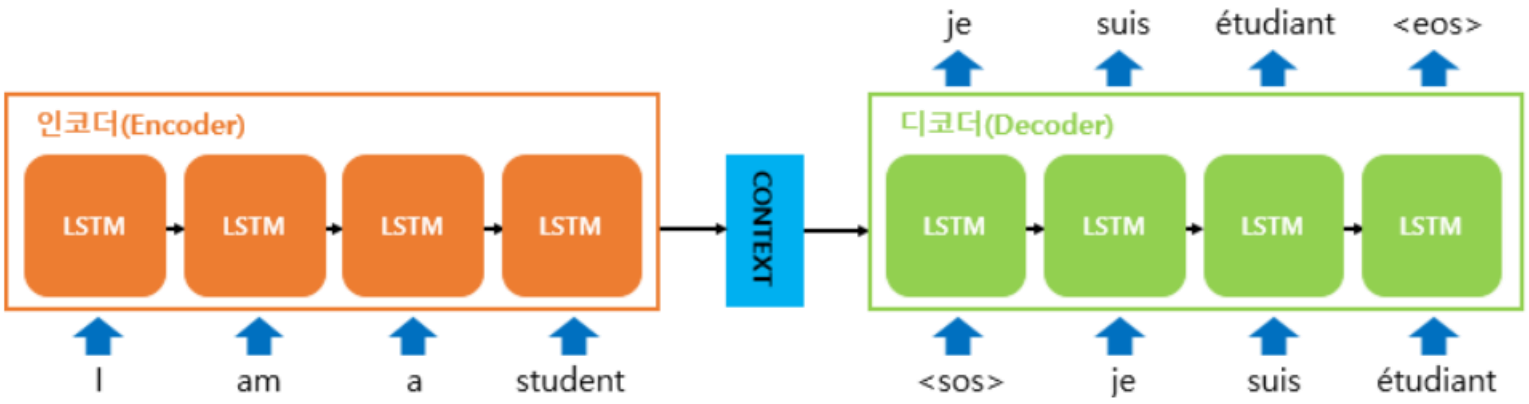
seq2seq

sequence-to-sequence model

NMT에서 대표적으로 사용되는 Encoder-Decoder 모델

Encoder와 Decoder에 각각 RNN 계열의 모델을 사용

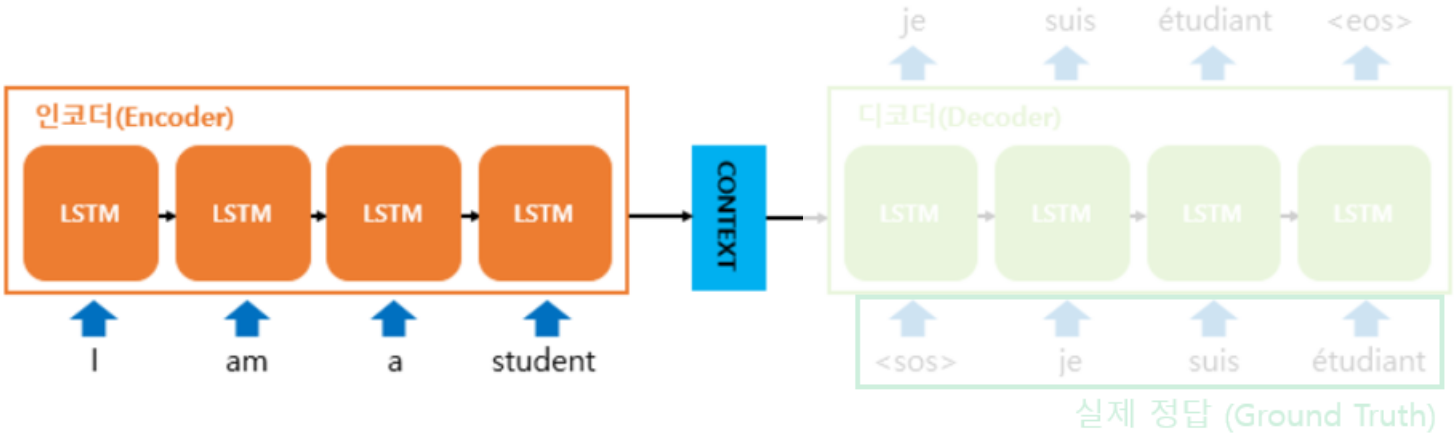
특히 내부 cell로 LSTM 을 사용하는 것이 학습 성능이 좋음



4 기계 번역(Machine Translation)

seq2seq에서 Encoder-Decoder의 학습

Encoder 학습



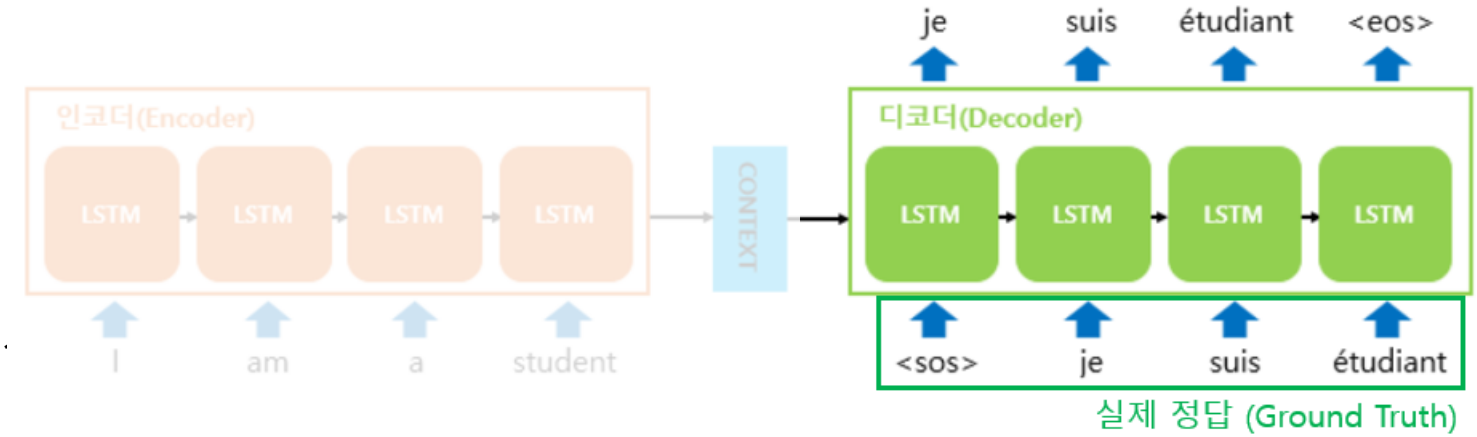
입력 문장을 받아서 하나의 context vector로 압축하고

입력 문장의 문맥을 학습함

4 기계 번역(Machine Translation)

seq2seq에서 Encoder-Decoder의 학습

Decoder 학습



Decoder는 context vector와 새로운 입력에 의존함

⋮

테스트 과정 – 학습 과정에서의 작동 방식이 다름

4 기계 번역(Machine Translation)

seq2seq에서 Encoder-Decoder의 학습

Decoder 학습

<sos>*: Start-Of-String, 문장의 시작을 알리는 토큰

테스트 과정

첫 번째 셀은 context vector와 <sos>*를 바탕으로 je를 생성

두 번째 셀은 첫 번째 셀의 예측인 je를 바탕으로 suis를 생성

:

이 과정을 문장이 끝날 때까지 반복 !

학습 과정

예측 대신 실제 정답(ground truth)을 입력 받은 후,

context vector를 바탕으로 실제 정답을 만들어내도록 학습하는

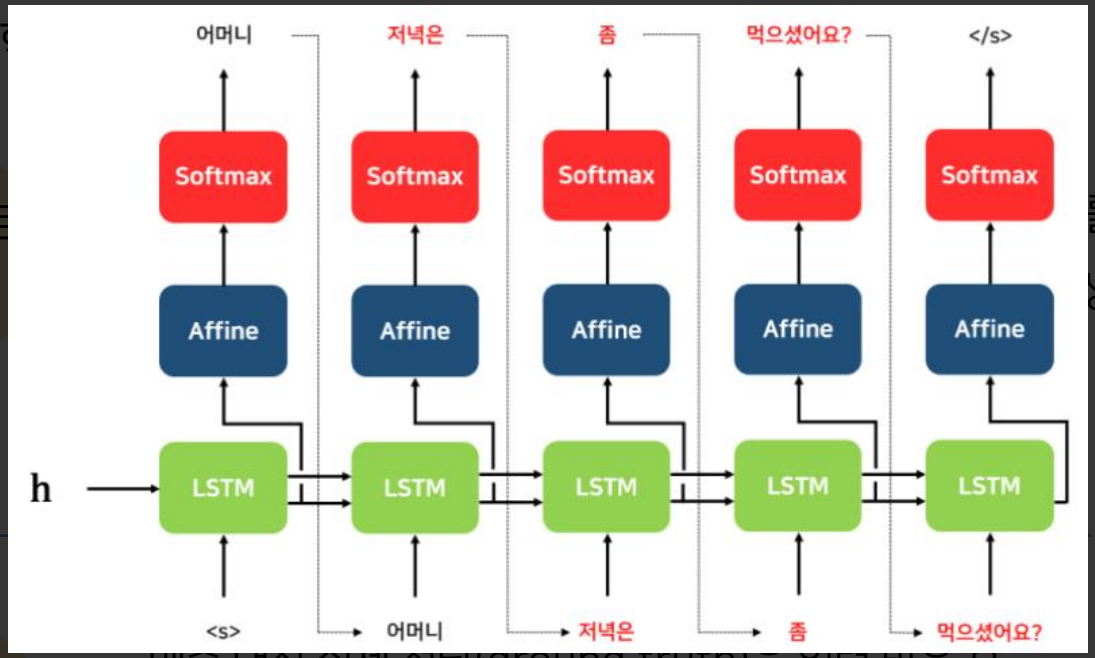
교사강요 적용

4 기계 번역(Machine Transition)

Decoder 학습에서 교사 강요를 사용하는 이유

seq2seq에서 Encoder-Decoder의 학습

Decoder



문장의 시작을 알리는 토큰

테스트 과정

를 생성

생성

학습 과정

예측 내신 실제 정답(ground truth)을 입력 받은 후,

context vector를 바탕으로 실제 정답을 만들어내도록 학습

: 교사 강요

Teacher Forcing을 **사용하지 않을 경우**

: t번째 예측을 만들어내기 위해 t-1번째 디코더의 예측값을 사용

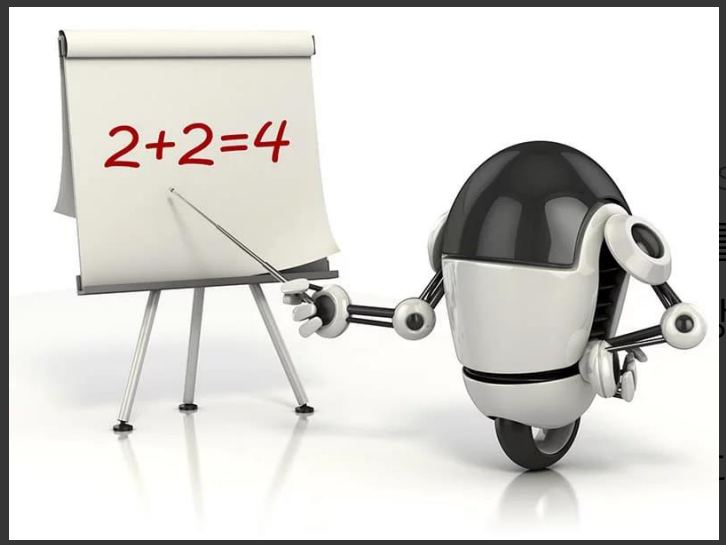
만약 예측이 올바르지 않다면, **이전 디코더의 예측 값**을 사용하는 것은 큰 문제

4 기계 번역(Machine Transition)

Decoder 학습에서 교사 강요를 사용하는 이유

seq2seq에서 Encoder-Decoder의 학습

Decoder 학습



테스트 과정

Start-Of-String, 문장의 시작을 알리는 토큰

를 바탕으로 je를 생성

으로 suis를 생성

까지 반복 !

학습 과정

장점

예측 대신 실제 정답(ground truth)을 입력 받은 후,

context vector를 바탕으로 실제 정답을 만들어내도록 학습하는 교

Teacher Forcing은 학습 과정에서 실제 정답을

입력 받음으로써 **더 빨리 학습 가능**

단점

학습과 추론 사이의 차이가 생기는

노출 편향 문제(Exposure Bias Problem)가

존재하나 성능에 큰 영향을 미치지 않음

4 기계 번역(Machine Translation)

seq2seq의 한계점

고정된 길이의 context vector

입력 문장의 모든 정보를 하나의 벡터에 저장 → 정보의 손실 발생

길이가 매우 길다면 고정된 길이의 context vector에 이를 다 반영하는 것은 어려움

기울기 소실 문제

이전의 은닉 상태(h_{t-1})와 현재의 입력(x_t)을 바탕으로 현재의 출력을 예측

→ 시간의 흐름에 따른 연속적인 구조로 기울기 소실 문제 존재

4 기계 번역(Machine Translation)

BLEU Score

Bilingual Evaluation Understudy

기계 번역 평가 방법으로

"실제 전문가의 번역과 가까울수록 더 좋은 번역이다"라는 내용을 함의

기계 번역 결과
Candidate Translation

유사도 측정

사람이 직접 번역한 결과
Reference Translation

4 기계 번역(Machine Translation)

BLEU Score

BLEU Score의 장점

① 빠르게 점수를 구할 수 있고 계산 비용이 저렴함
② 결과를 비교적 이해하기 쉬움
③ 언어에 구애 받지 않고 사용할 수 있음
④ 사람의 번역과 크게 연관시킬 수 있음
⑤ 폭 넓게 적용 가능

4 기계 번역(Machine Translation)

BLEU Score

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

BP : Brevity Penalty,

만들어낸 문장의 길이가 짧을 경우 페널티를 줌

$\exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$: 보정된 N-gram 정밀도 기하평균

4 기계 번역(Machine Translation)

BLEU Score

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

BP : Brevity Penalty,
만들어낸 문장의 길이가 짧을 경우 페널티를 줌

$\exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$: 보정된 N-gram 정밀도 기하평균

해당 식이 어떻게 나오게 되는지 알아보자!

4 기계 번역(Machine Translation)

Unigram 정밀도

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.*

두 개의 Candidate Translation (Machine Translation)

Reference 1: *It is a guide to action that ensures that the military will forever heed Party commands.*

Reference 2: *It is the guiding principle which guarantees the military forces always to heed the Party command of the Party*

Reference 3: *It is the practical guide for the army always to heed the directions of the party.*

세 개의 Reference Translation (Human Translation)

Candidate 1 – ($P = \frac{17}{18}$)

C1	It	is	a	guide	to	action	which	ensures	that	the	military	always	obeys	the	commands	of	the	party
R1	0	0	0	0	0	0	X	0	0	0	0	X	X	0	0	X	0	0
R2	0	0	X	X	X	X	0	X	X	0	0	0	X	0	0	0	0	0
R3	0	0	X	0	0	X	X	X	X	0	X	0	X	0	X	0	0	0

Candidate 2 – ($P = \frac{8}{14}$)

C2	It	is	to	insure	the	troops	forever	hearing	the	activity	guidebook	that	party	direct
R1	0	0	0	X	0	X	0	X	0	X	X	0	0	X
R2	0	0	X	X	0	X	X	X	0	X	X	X	0	X
R3	0	0	0	X	0	X	X	X	0	X	X	X	0	X

이전 단어를 고려하지 않고 현재 단어 하나만을

보기 때문에 단어를 단순히 카운트하는 방법

4 기계 번역(Machine Translation)

Unigram 정밀도의 문제점

중복된 단어 처리

Machine Translation : *the the the the the the the*

Human Translation : ① *The cat is on the mat*

② *There is a cat on the mat*

기계 번역이 the로만 이루어져 있음

→ 단어를 단순히 카운트하는 Unigram 방식에서는 결과가 $\frac{7}{7} = 1$이 나옴

즉, 좋은 번역이 아니어도 좋은 결과가 나올 수 있음

4 기계 번역(Machine Translation)

Unigram 정밀도의 문제점

순서를 고려하지 않음

Unigram 방식은 단어 각각에 대해 독립적으로 카운트하는 방법

그러나, 자연어는 기본적으로 순차 데이터

→ 순서를 고려해줄 방법 필요 !

보정된 N-gram 정밀도
(Modified N-gram
precision)

4 기계 번역(Machine Translation)

보정된 N-gram 정밀도

중복된 단어 처리 문제 해결

Machine Translation : *the the the the the the the*

Human Translation : ① *The cat is on the mat*

② *There is a cat on the mat*

참조 문장에서는 the가 2번 밖에 안 나왔는데 이를 반영하지 못함

$$Count_{clip} = \min(Count, Max\_Ref\_Count)$$

↓

참조 문장에서 동일한 단어 등장 횟수(최대 참조 횟수)

단순 카운트와 최대 참조 횟수를 동시에 고려함

→ 정밀도는 $\frac{2}{7}$ 가 나옴

4 기계 번역(Machine Translation)

보정된 N-gram 정밀도

순서 고려

Candidate 1 – $(P = \frac{10}{17})$

C1	It is	is a	a guide	guide to	to action	action which	which ensures	ensures that	that the
R1	0	0	0	0	0	X	X	0	0
R2	0	X	X	X	X	X	X	X	X
R3	0	X	X	X	X	X	X	X	X
C1	the military	military always	always obeys	obeys the	the commands	commands of	of the	the party	
R1	0	X	X	X	X	X	X	X	
R2	0	X	X	X	X	X	0	0	
R3	X	X	X	X	X	X	0	0	

Candidate 2 – $(P = \frac{1}{13})$

C2	it is	is to	to insure	insure the	the troops	troops forever	forever hearing
R1	0	X	X	X	X	X	X
R2	0	X	X	X	X	X	X
R3	0	X	X	X	X	X	X
C2	hearing the	the activity	activity guidebook	guidebook that	that party	party direct	
R1	X	X	X	X	X	X	
R2	X	X	X	X	X	X	
R3	X	X	X	X	X	X	

이전 단어를 포함해 순서를 고려하는 N-gram으로 Unigram의 단점 보완

4 기계 번역(Machine Translation)

Brevity Penalty (BP)

Brevity Penalty

짧은 문장에 대해 페널티를 줌

아래와 같은 Reference 문장과 Candidate 문장에 대해
보정된 2-gram 정밀도로 계산해보자.

Machine Translation : ① *That cat sat on the soft mat.*
② *Black and white cat sat on mat in corner of room.*

Human Translation : *The black and white that cat sat peacefully on the soft mat in the corner of the room.*

4 기계 번역(Machine Translation)

Brevity Penalty (BP)

Candidate1 – $(P = \frac{5}{6})$

C1	that cat	cat sat	sat on	on the	the soft	soft mat
R	0	0	X	0	0	0

That cat sat on the soft mat.

Candidate 2 – $(P = \frac{5}{10})$

C2	black and	and white	white cat	cat sat	sat on	on mat	mat in	in corner	corner of	of room
R	0	0	X	0	X	X	0	X	0	X

Black and white cat sat on mat in corner of room.

Candidate 1은 많은 부분이 생략되어 좋은 문장은 아니지만,

2-gram 정밀도로 계산하면 Candidate 2보다 좋다고 판단

→ 이를 완화하기 위해 BP factor 부여

4 기계 번역(Machine Translation)

Brevity Penalty (BP)

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

Machine Translation : ① *That cat sat on the soft mat.*
② *Black and white cat sat on mat in corner of room.*

Human Translation : *The black and white that cat sat peacefully on the soft mat in the corner of the room.*

→ BP = 0.2077

→ BP = 0.5292

문장의 길이가 짧을수록 작은 수를 부여하여

문장 길이에 대한 페널티 적용

Brevity Penalty (BP)

이러한 BP와 보정된 N-gram 정밀도를 종합하여

$BLEU = BP \times \exp(\sum_{n=1}^{N} w_n \log p_n)$ 라는 식을 만듦

(일반적으로 N=1,2,3,4)

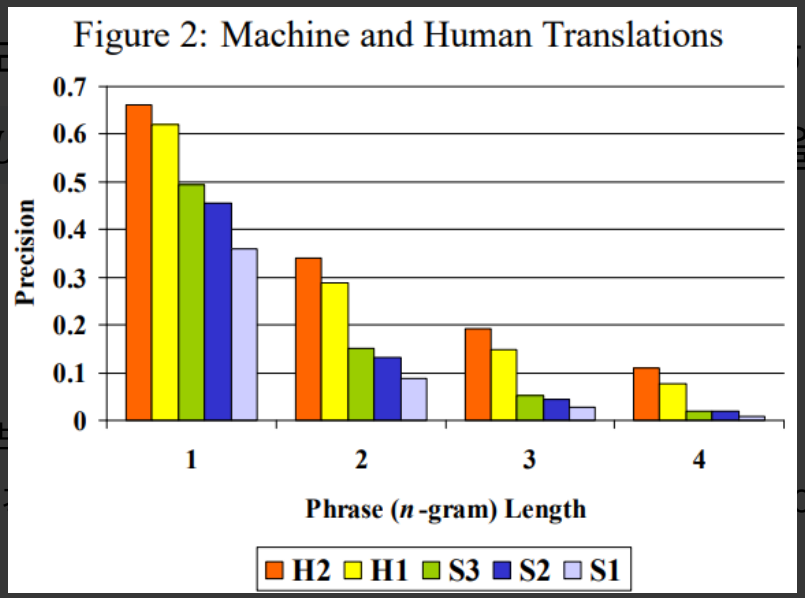
* p_n: 보정된 N-gram 정밀도
* w_n: 각 N-gram에 대한 가중치, 보통 모두 같은 가중치를 부여

4 기계 번역(Machine Transition)

Brevity Penalty (BP)

$\log p_n$ 을 한 다음 더해주는 이유가 무엇일까?

N-gram 정밀도의 N=1,2,3,4까지의 결과



N이 증가할수록 정밀도는 지수적으로 감소함

이를 보정해주기 위해 log를 취해줌

다음 주 예고

1. Attention
2. Transformer
3. Pretrained Model

감사합니다