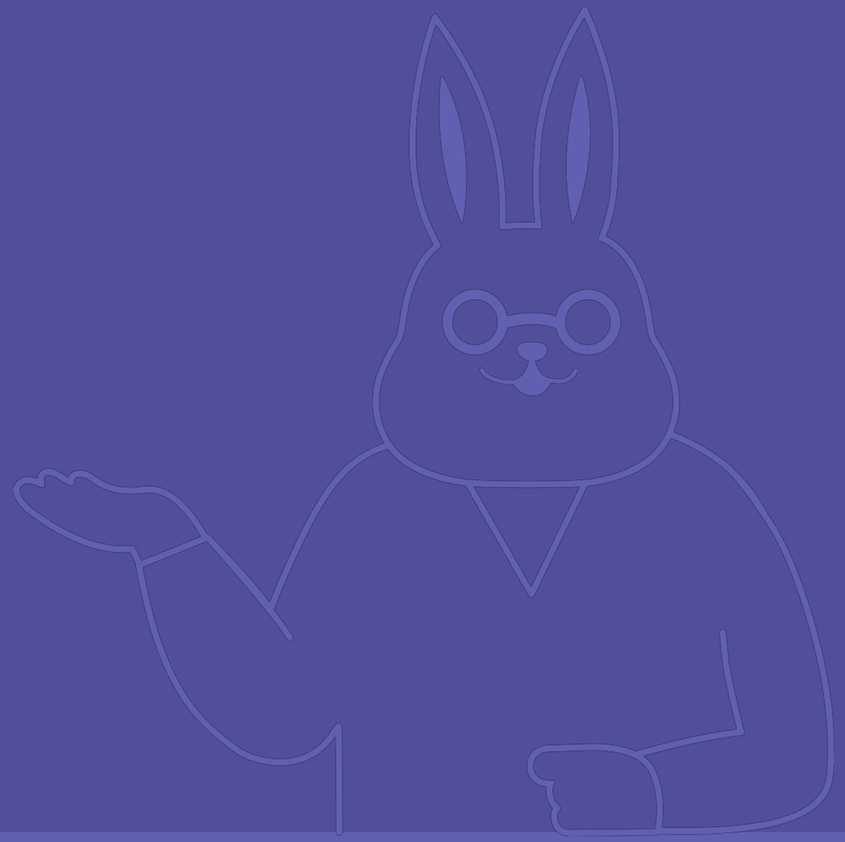
딥러닝을 이용한 자연어 처리

04 문서 유사도 및 언어 모델

목차

01

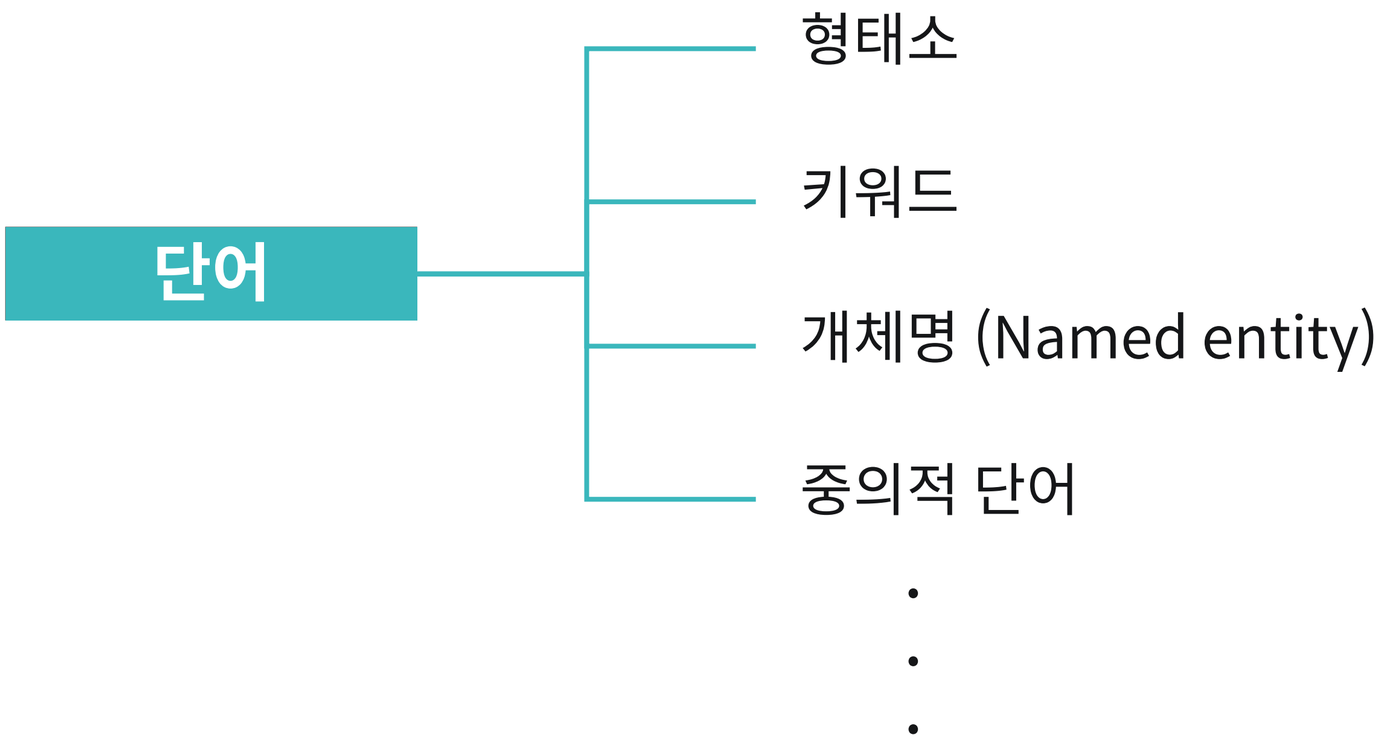
문서 유사도 측정



문서

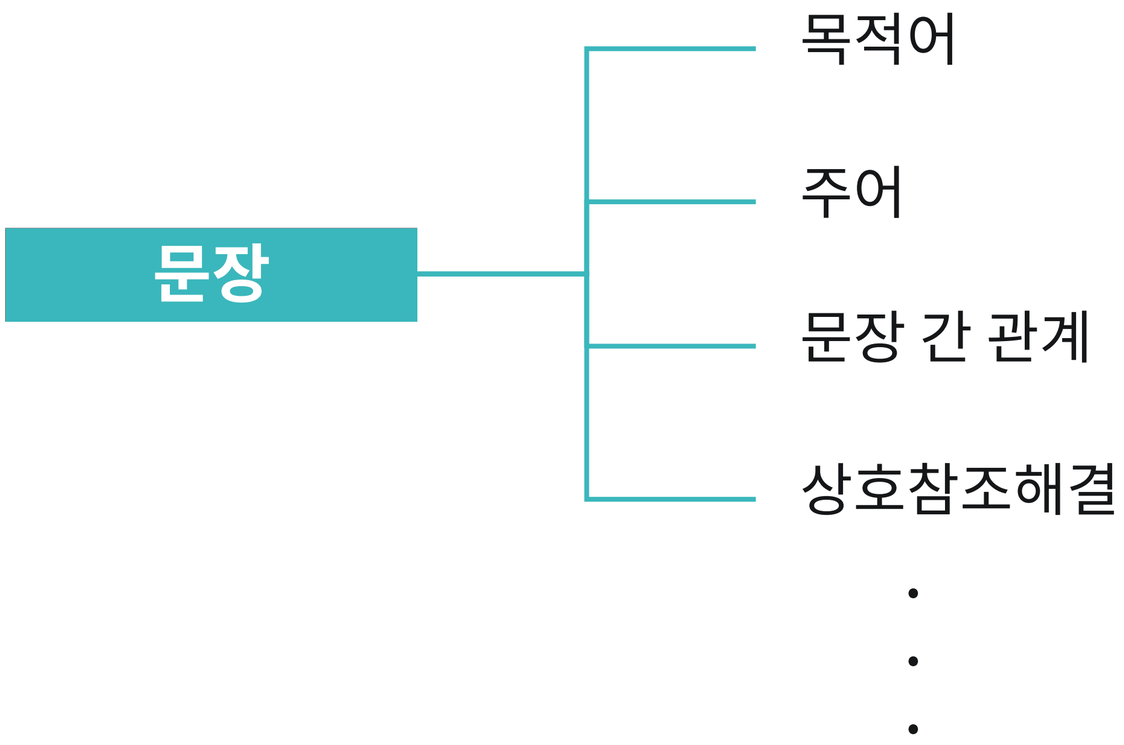
문서는 **다양한 요소**와 이들의 **상호작용**으로 구성

문서



가장 기본 단위인 단어 조차 문서와 관련된 **다양한 정보**를 포함

문서



상위 개념인 문장 또한 추가적인 **정보**를 제공

문서

문서의 가장 기본 단위인 **단어**를 활용하여 문서를 표현

문서 유사도

[문서 1] : [0, 2, 93, 2, ⋯]

[문서 2] : [1, 3, 0, 38, ⋯]

.
.
.

문서 유사도를 측정하기 위해 단어 기준으로 생성한 **문서 벡터** 간의 코사인 유사도를 사용

문서 유사도

[문서 1] : [0, 1, 0, 0, ⋯]

[문서 2] : [8, 0, 0, 0, ⋯]

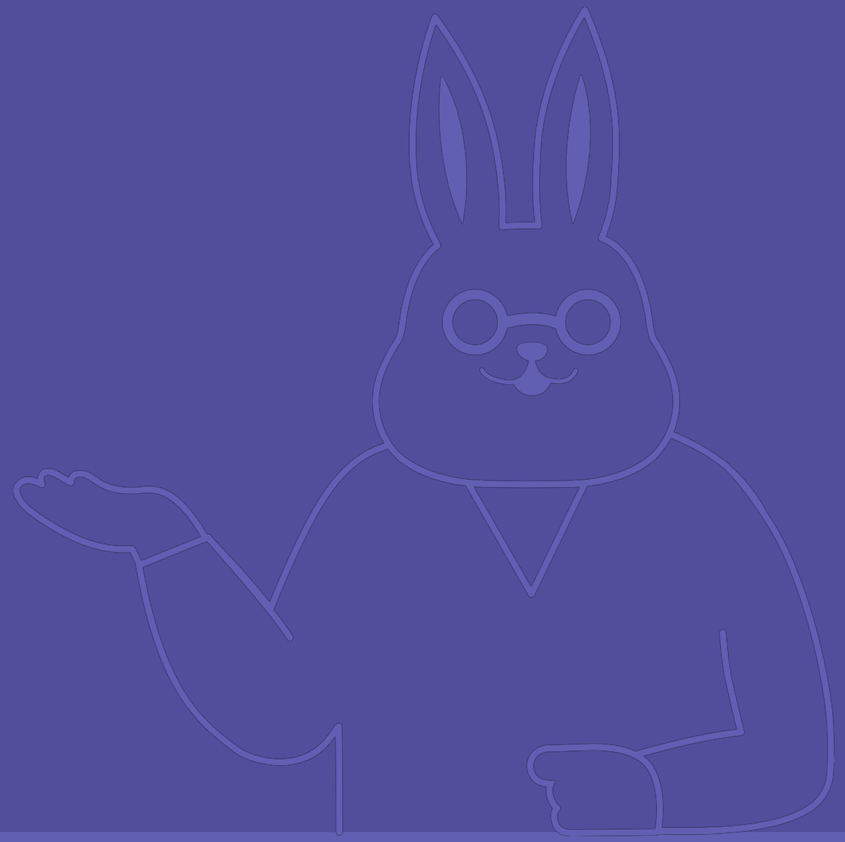
[문서 1] : [0.29, 0.372, 0.93, 0.21, ⋯]

[문서 2] : [0.11, 0.223, 0.001, 1.38, ⋯]

정확한 문서 유사도 측정을 위해 문서의 특징을 잘 보존하는 **벡터 표현 방식**이 중요

02

Bag of words



Bag of words

[문서 1] : these | are | five | IT | companies …

[문서 2] : these | five | great | singers …

	these	are	five	IT	companies	great	singers	a
문서 1	3	7	1	6	1	0	0	0
문서 2	2	4	1	0	0	1	4	0

문서 내 **단어의 빈도수**를 기준으로 문서 벡터를 생성

Bag of words

[문서 1] : these | are | five | IT | companies …

[문서 2] : these | five | great | singers …

	these	are	five	IT	companies	great	singers	a
문서 1	3	7	1	6	1	0	0	0
문서 2	2	4	1	0	0	1	4	0

자주 발생하는 단어가 문서의 특징을 나타낸다는 것을 가정

Bag of words

[문서 1] : these | are | five | IT | companies …

[문서 2] : these | five | great | singers …

	these	are	…	…	a	about	…	zebra
문서 1	3	7	…	…	12	0	…	0
문서 2	2	4	…	…	14	1	…	0

Bag of words 문서 벡터의 차원은 **데이터** 내 발생하는 **모든 단어의 개수**와 동일

Bag of words

[문서 3] : … | don’t | log | off …

[문서 4] : … | this | log | is | great …

	these	are	…	…	log	off	…	zebra
문서 1	1	3	…	…	1	1	…	0
문서 2	5	2	…	…	1	0	…	0

Bag of words 문서 벡터는 **합성어**를 독립적인 단어로 개별 처리

Bag of N-grams

[문서 1] (unigram) : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망되며 …

N = 2 (bi-gram) : 포근한 봄 | 봄 날씨가 | 날씨가 이어질 | 이어질 것으로 | …

N = 3 (tri-gram) : 포근한 봄 날씨가 | 봄 날씨가 이어질 | 날씨가 이어질 것으로 | …

N-gram은 연속된 N개의 단어를 기준으로 텍스트 분석을 수행

Bag of N-grams

[문서 1] : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망되며 …

	포근한 봄	봄 날씨가	날씨가 이어질	이어질 것으로	것으로 전망되며	내일 하루	…
문서 1	1	1	1	6	1	0	…

Bag of N-grams은 n-gram의 발생 빈도를 기준으로 문서 벡터를 표현

Bag of N-grams

[문서 1] : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망되며 …

	포근한	포근한 봄	봄	봄 날씨가	날씨가	날씨가 이어질	…
문서 1	1	1	1	1	3	0	…

Bag of N-grams은 여러 n-gram을 합쳐서 발생 빈도를 기준으로 문서 벡터를 표현

TF-IDF

	그리고	그러나	잘	봄	컴퓨터	오늘	만약	…
문서 1	32	23	17	4	0	20	12	…
문서 2	27	12	11	0	7	10	8	…

자주 발생하는 단어가 문서의 주요 내용 및 특징을 항상 **효과적으로 표현하지는 않음**

TF-IDF

[문서 1] : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망되며 …

문서 1에서 단어 "봄"의 TF-IDF 점수 =

$$\frac{\text{문서 1 내 "봄"의 빈도수}}{\text{문서1 내 모든 단어의 빈도수}} \times \log\left(\frac{\text{데이터 내 총 문서의 개수}}{\text{데이터 내 "봄"이 들어간 문서의 개수}}\right)$$

TF-IDF(term frequency– inverse document frequency)는 문서 내
상대적으로 자주 발생하는 단어가 더 중요하다는 점을 반영

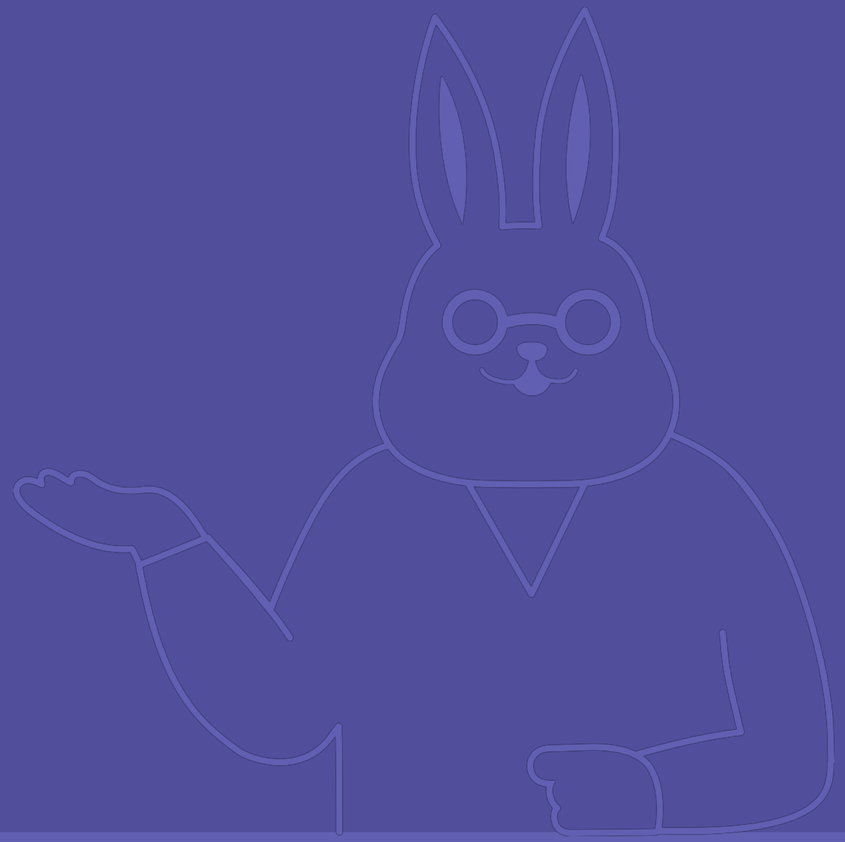
TF-IDF

	그리고	그러나	잘	봄	컴퓨터	오늘	만약	…
문서 1	0.003	0.023	0.03	0.537	0	0.02	0.12	…
문서 2	0.001	0.021	0.01	0	0.783	0.05	0.22	…

TF-IDF 기반의 bag of words 문서 벡터는 **단어의 상대적 중요성**을 반영

03

doc2vec

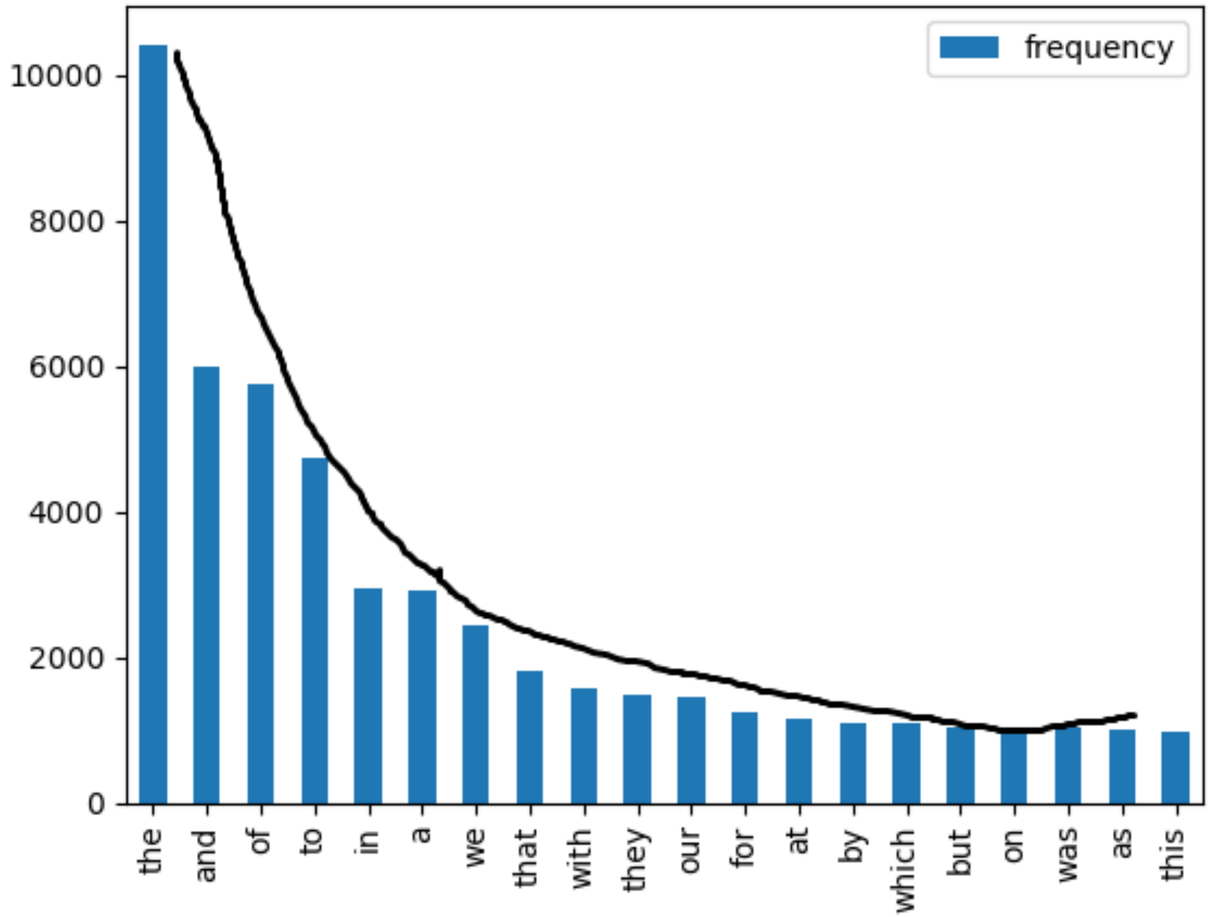


Bag of words 기반 문서 벡터의 장점

	그리고	그러나	잘	봄	컴퓨터	오늘	만약	…
문서 1	0	0	0	0.537	0	0	0	…
문서 2	0	0	0	0	0.783	0	0	…

벡터의 구성 요소가 **직관적**인 것은 bag of words 기반 기법의 큰 장점

Bag of words 기반 문서 벡터의 단점



텍스트 데이터의 양이 증가하면, 문서 벡터의 **차원 증가**

Bag of words 기반 문서 벡터의 단점

	그리고	그러나	잘	봄	컴퓨터	오늘	…	흐미
문서 1	0	0	0	0.537	0	0	…	0
문서 2	0	0	0	0	0.783	0	…	0

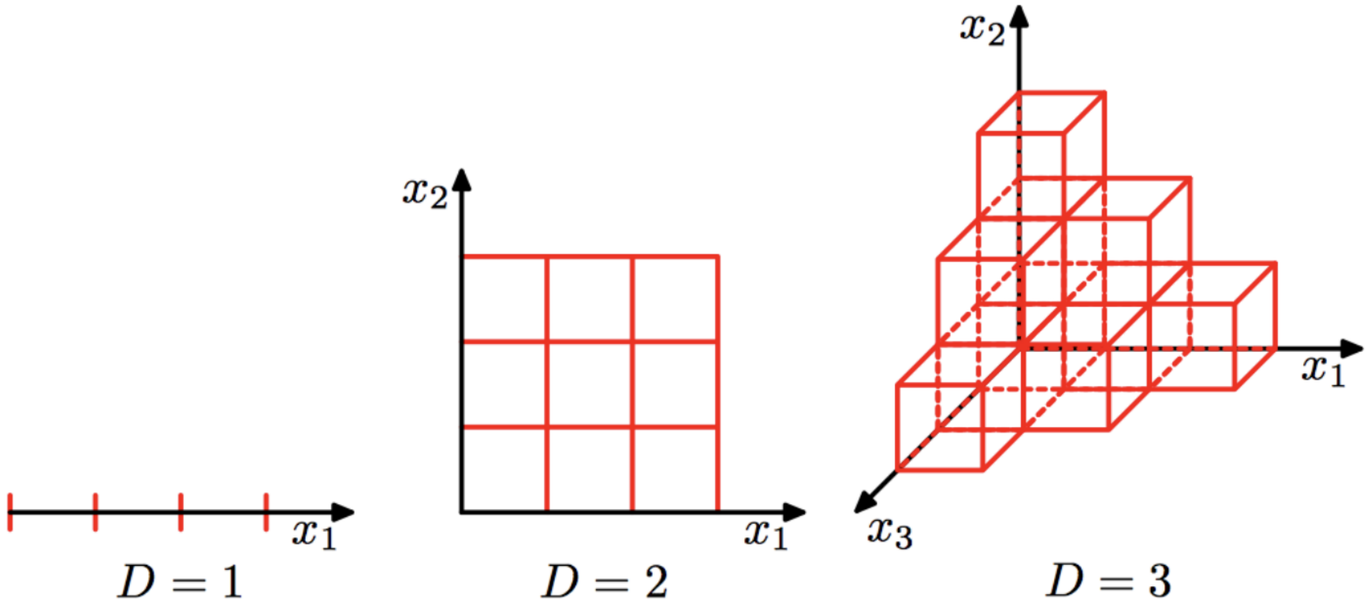
대부분 단어의 빈도수가 0인 **희소(sparse) 벡터**가 생성

Bag of words 기반 문서 벡터의 단점

```
---------------------------------------------------------------------------
MemoryError                                 Traceback (most recent call l
<ipython-input-11-1be1bfb42645> in <module>()
----> 1 np.random.uniform(low=0,high=1.0, size=(100000,1000000))

mtrand.pyx in mtrand.RandomState.uniform()

mtrand.pyx in mtrand.cont2_array_sc()

MemoryError:
```

문서 벡터의 차원 증가에 따른 **메모리 제약 및 비효율성** 발생

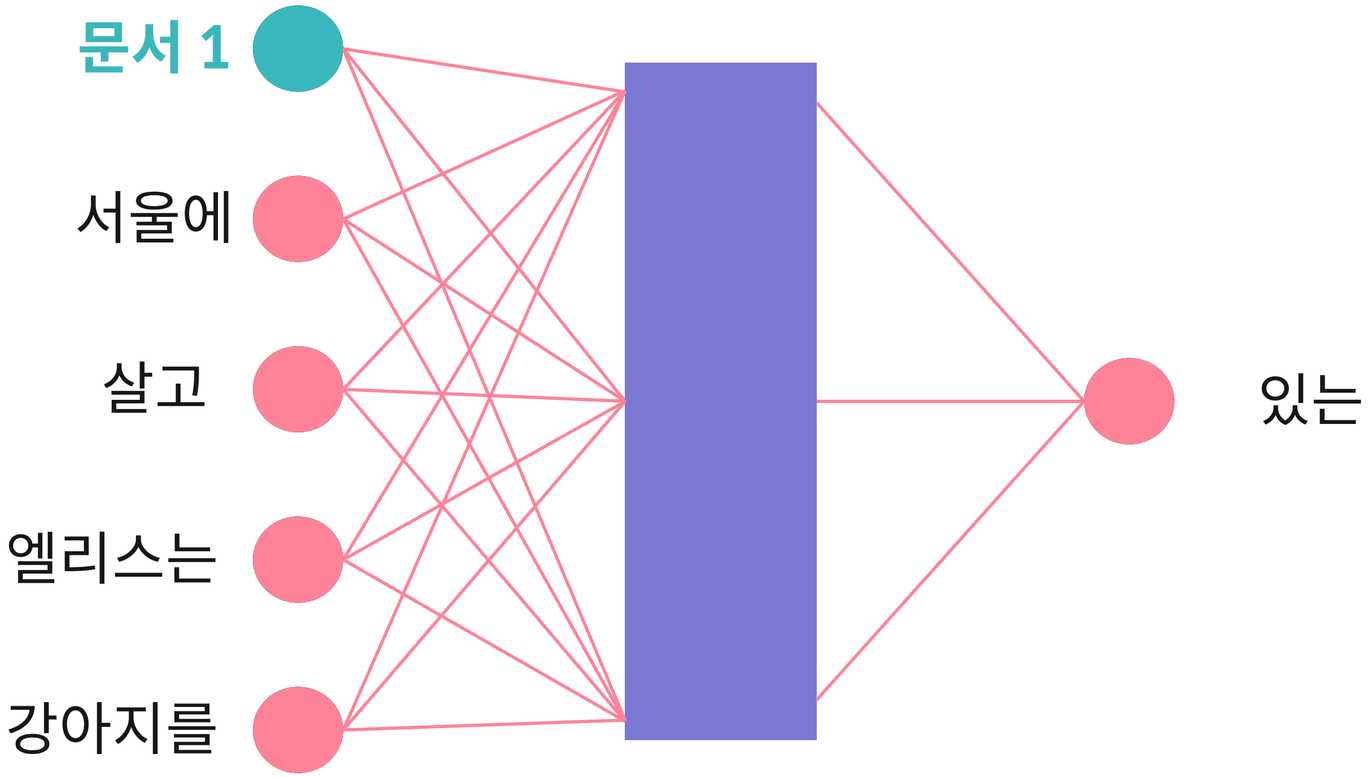
Bag of words 기반 문서 벡터의 단점



문서 벡터의 차원 증가에 따른 **차원의 저주** 발생

doc2vec

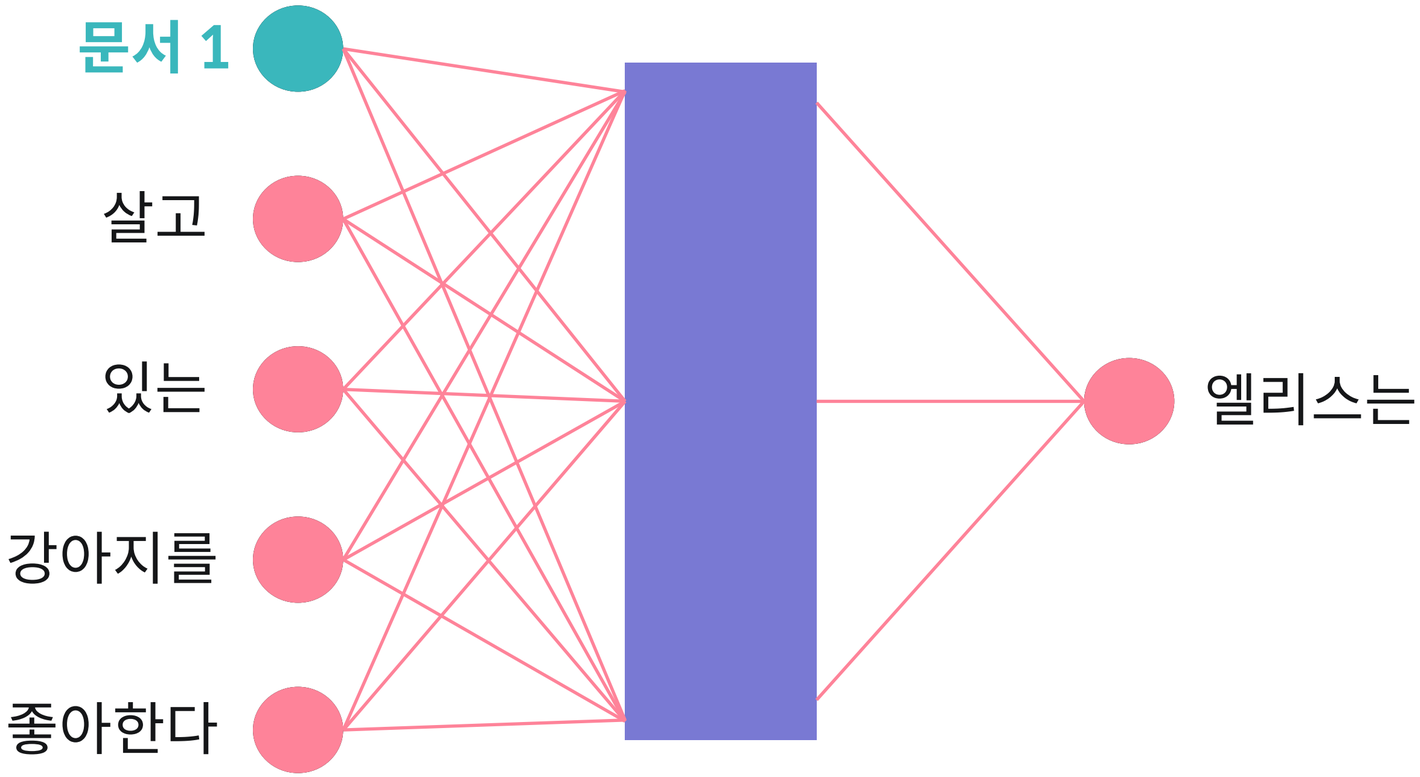
[문서 1] : 서울에 살고 있는 엘리스는 강아지를 좋아한다.



doc2vec은 문서 내 단어 간 문맥적 유사도를 기반으로 문서 벡터를 임베딩

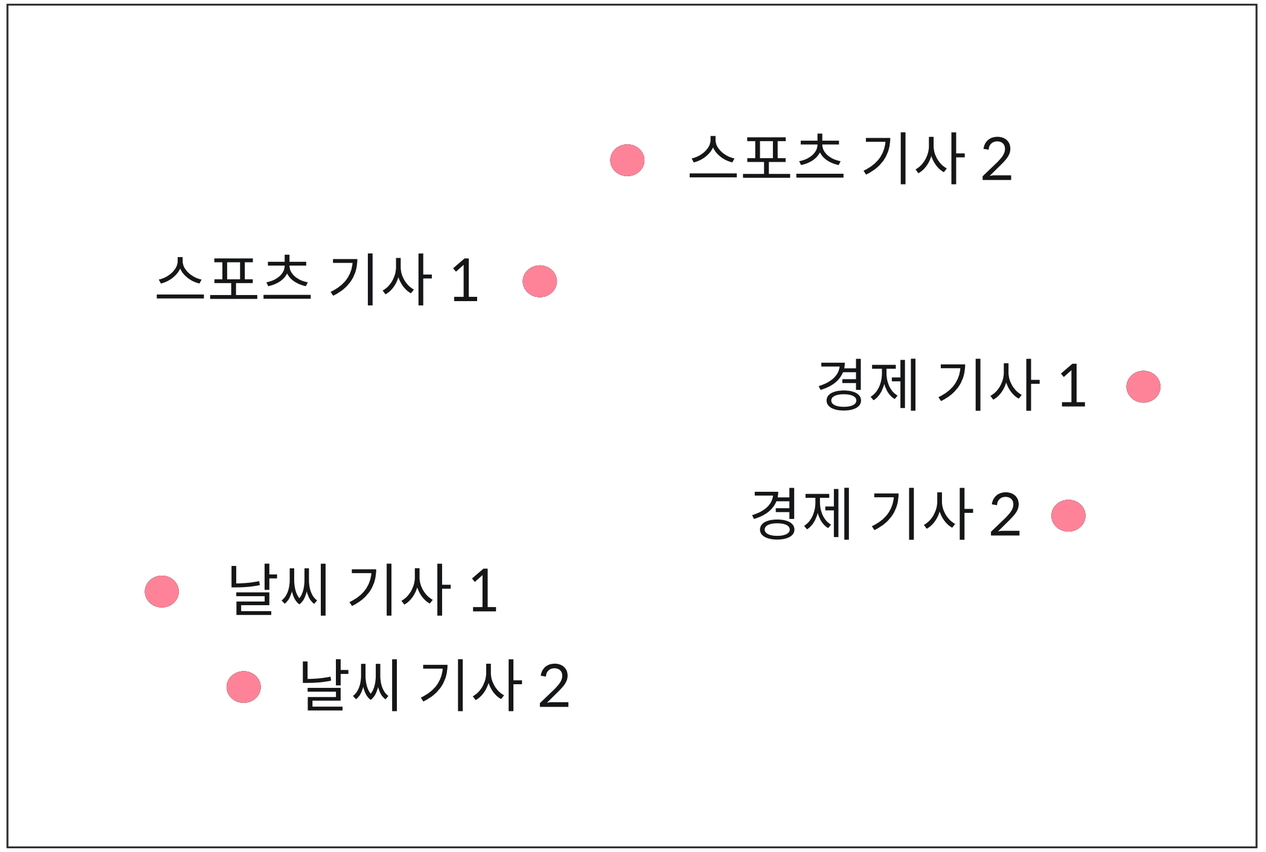
doc2vec

[문서 1] : 서울에 살고 있는 엘리스는 강아지를 좋아한다.



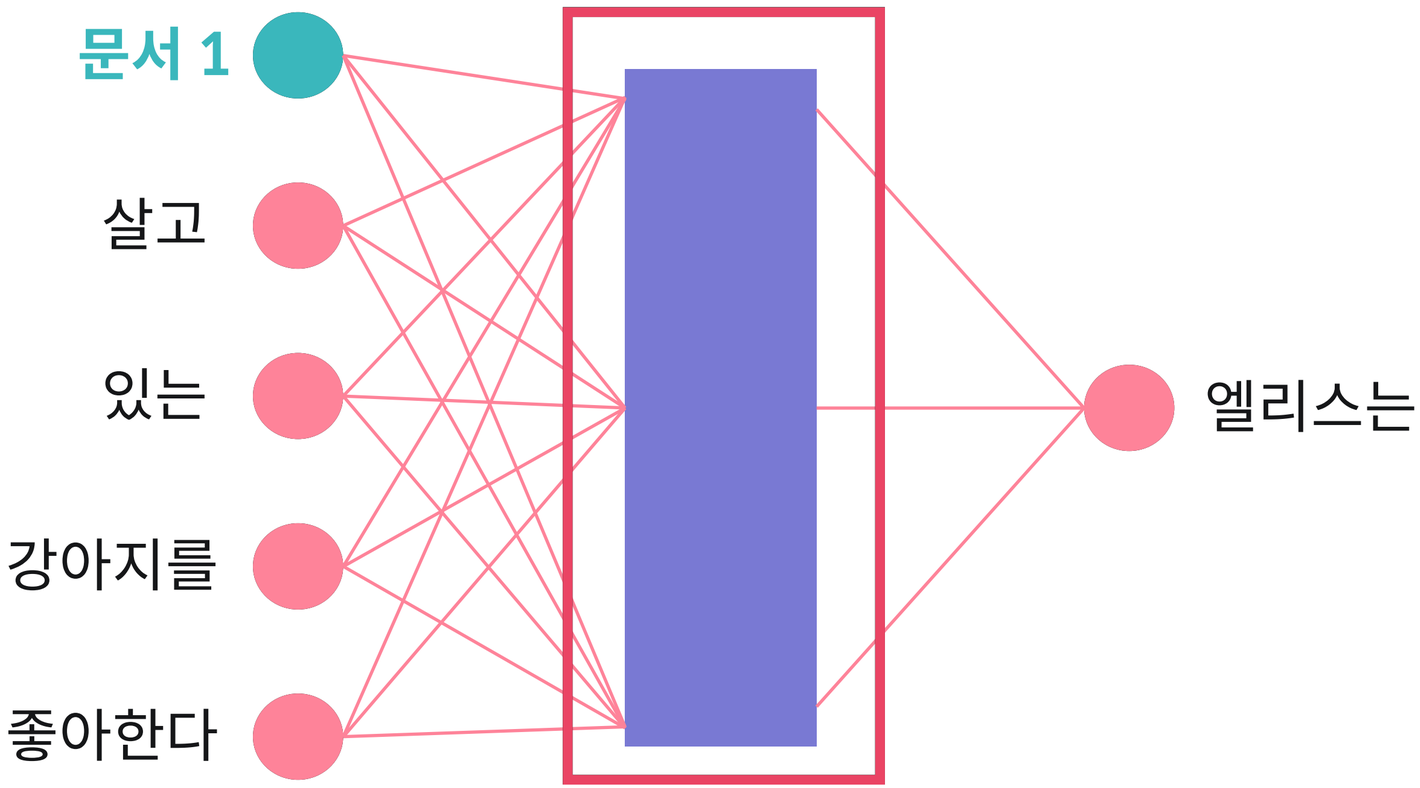
문서 내 단어의 임베딩 벡터를 학습하면서 문서의 임베딩 또한 **지속적**으로 학습

doc2vec



유사한 문맥의 문서 임베딩 벡터는 **인접한 공간**에 위치

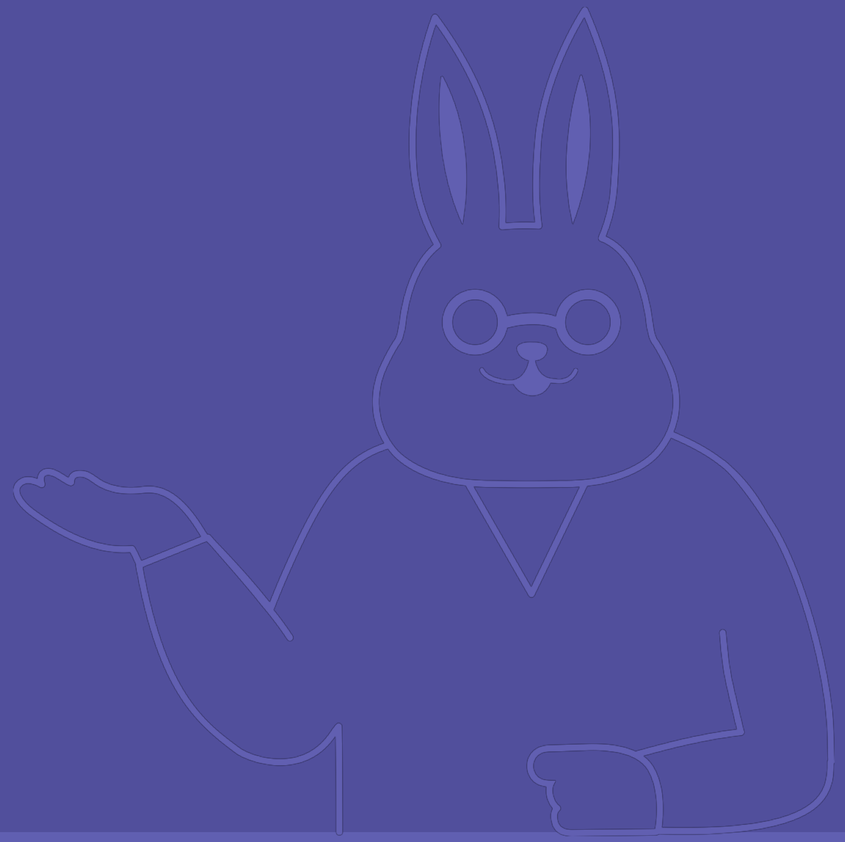
doc2vec



doc2vec은 상대적으로 **저차원**의 공간에서 문서 벡터를 생성

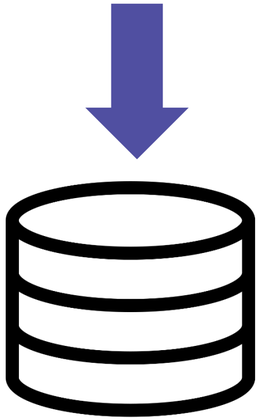
04

N-gram 기반 언어 모델



언어 모델

문장 1 : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망됩니다.

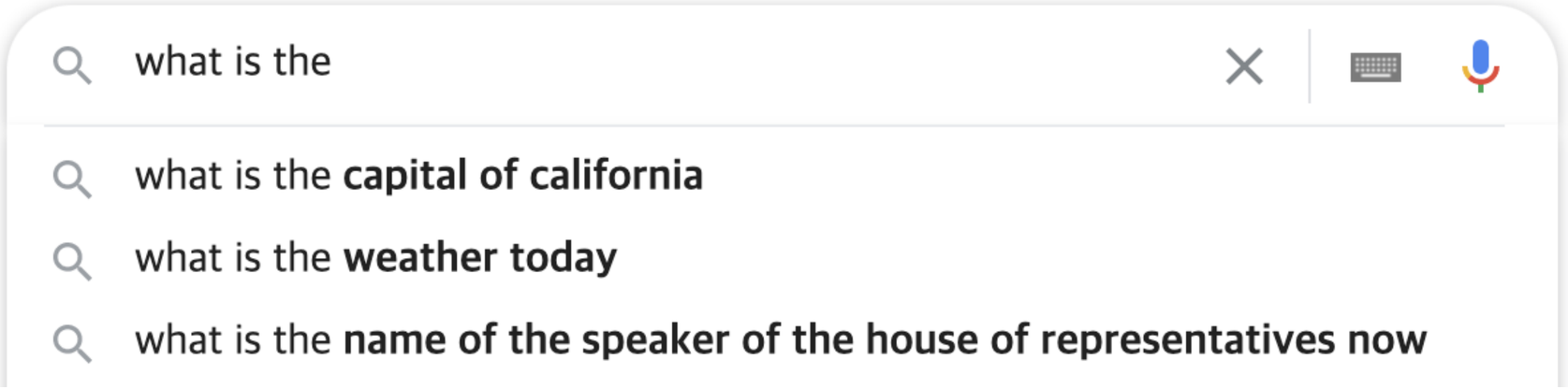


텍스트 데이터

$$P(\text{문장 } 1) = 0.233$$

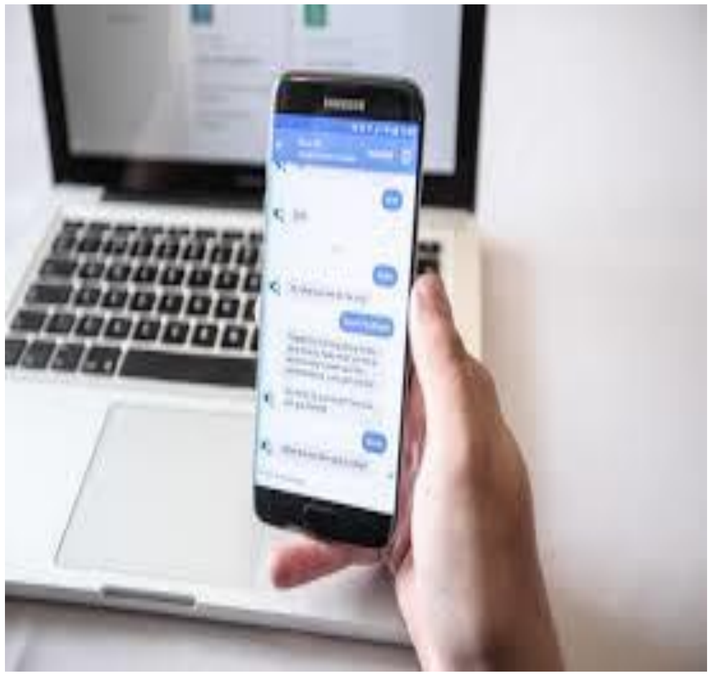
언어 모델이란 주어진 문장이 텍스트 데이터에서 발생할 확률을 계산하는 모델

언어 모델



언어 모델을 통해 자동 문장 생성이 가능

언어 모델



챗봇 내 핵심 요소 중 하나

언어 모델

문장 1 : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망됩니다.

$$P(\text{문장 1}) = P(\text{포근한}) \times P(\text{봄} \mid \text{포근한}) \times P(\text{날씨가} \mid \text{포근한, 봄}) \times$$
$$P(\text{이어질} \mid \text{포근한, 봄, 날씨가}) \times \ldots \times P(\text{전망됩니다} \mid \text{포근한, 봄, 날씨가, 이어질, 것으로})$$

문장의 발생 확률은 **단어가 발생할 조건부 확률의 곱으로** 계산

N-gram 기반 언어 모델

문장 1 : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망됩니다.

Tri-gram 기준 $P(\text{문장 1}) \approx P(\text{날씨가} \mid \text{포근한, 봄}) \times P(\text{이어질} \mid \text{봄, 날씨가}) \times \ldots \times P(\text{전망됩니다} \mid \text{이어질, 것으로})$

N-gram을 사용하여 **단어의 조건부 확률**을 근사

N-gram 기반 언어 모델

$$P(\text{날씨가} \mid \text{포근한, 봄}) = \frac{\text{전체 데이터 내 "포근한 봄 날씨가" 의 빈도수}}{\text{전체 데이터에서 "포근한 봄"의 빈도수}}$$

각 N-gram 기반 조건부 확률은 데이터 내 **각 n-gram의 빈도수**로 계산

N-gram 기반 언어 모델

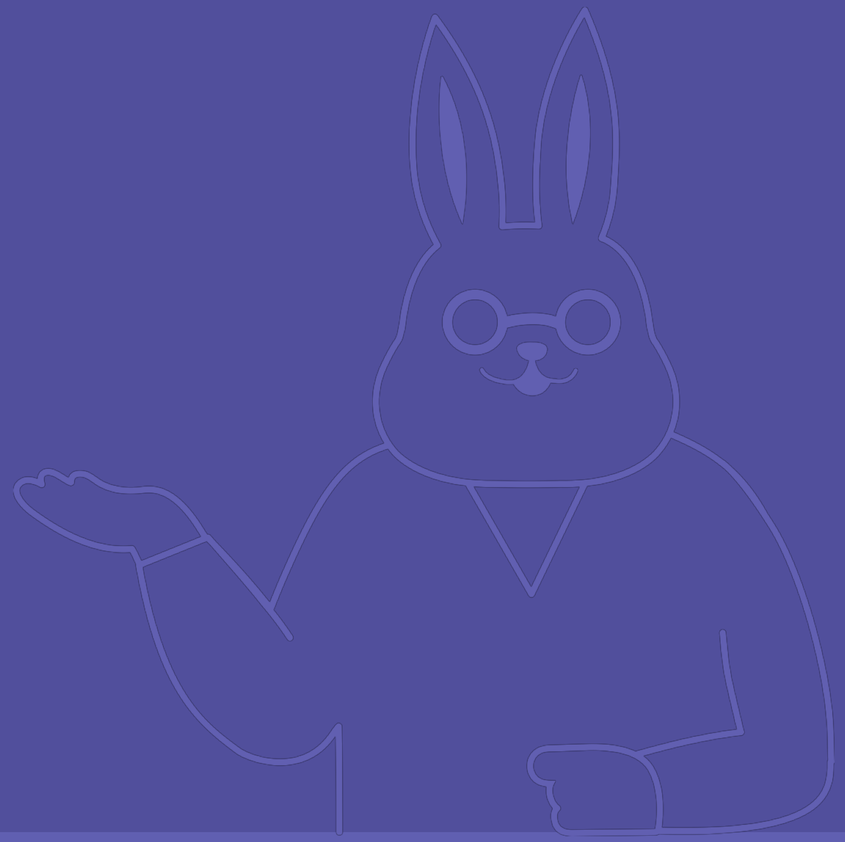
생성되는 문장 : 무더운 | 여름 | ?

$$P(\text{엘리스} \mid \text{여름}) = 0.02$$
$$P(\text{여름} \mid \text{바다}) = 0.5$$
$$P(\text{날씨} \mid \text{무더운, 여름}) = 0.87$$
$$\cdots$$

문장 생성 시, 주어진 단어 기준 **최대 조건부 확률**의 단어를 다음 단어로 생성

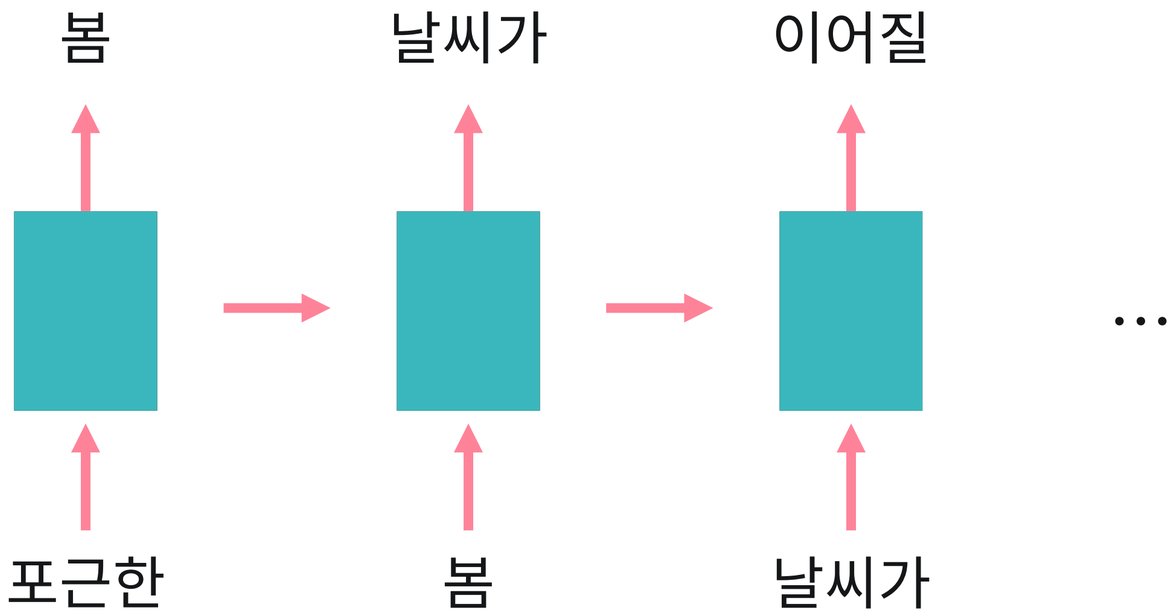
05

RNN 기반 언어 모델



RNN 기반 언어 모델

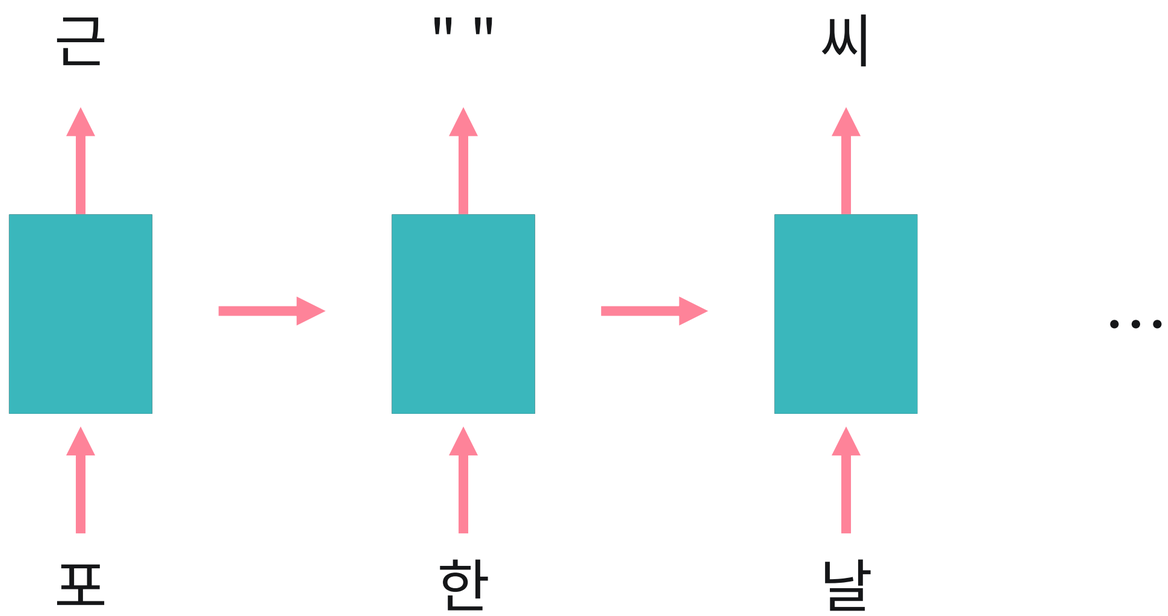
문장 1 : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망됩니다.



RNN으로 문장의 각 단어가 주어졌을 때 **다음 단어를 예측하는 문제**로 언어 모델 학습

RNN 기반 언어 모델

문장 1 : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망됩니다.



문자 단위 언어 모델로 학습 데이터 내 존재하지 않았던 단어 처리 및 생성 가능

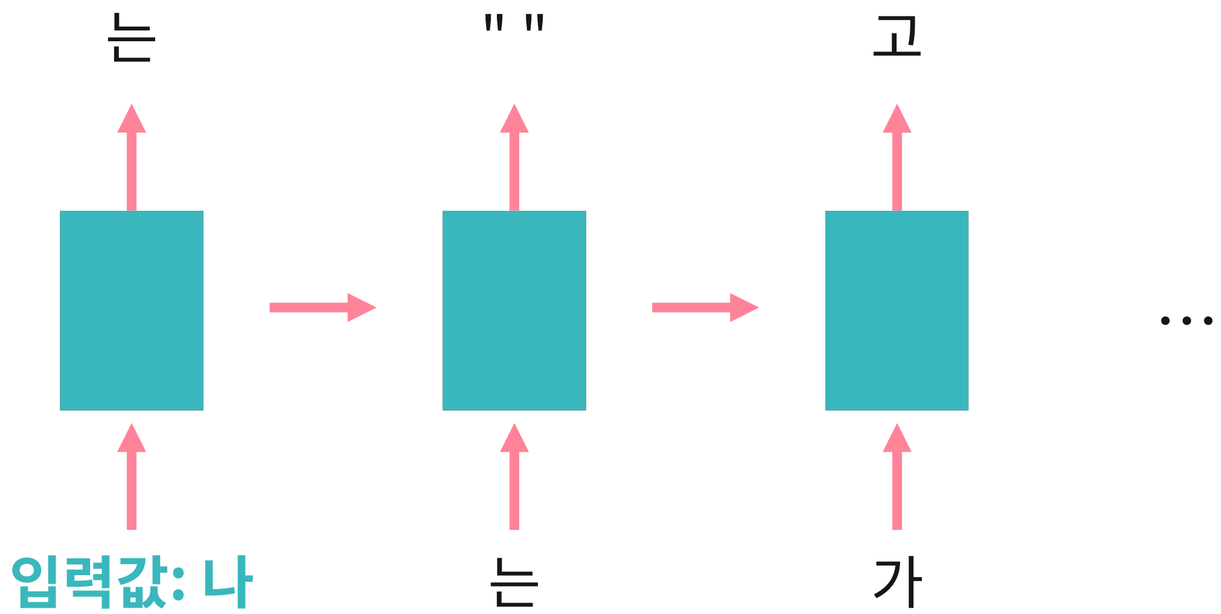
RNN 기반 언어 모델

문장 1 : 포근한 | 봄 | 날씨가 | 이어질 | 것으로 | 전망됩니다.

학습 데이터 내 문장 1 : **<Start>** | 포근한 | 봄 | 날씨가 | … | 입니다 | **<End>**

모델 학습 시, 문장의 시작과 종료를 의미하는 **태그(tag)** 추가

RNN 기반 언어 모델



문장 생성 시, 주어진 입력값부터 **순차적**으로 예측 단어 및 문자를 생성

딥러닝 기반 언어 모델

OpenAI

GPT-3, an autoregressive language model with 175 billion parameters

고성능 언어 모델은 **대용량 데이터**와 이를 학습할 수 있는 **하드웨어**가 필수