

딥러님을 이용한 자연어 처리

01 텍스트 전처리 및 단어 임베딩

Confidential all rights reserved

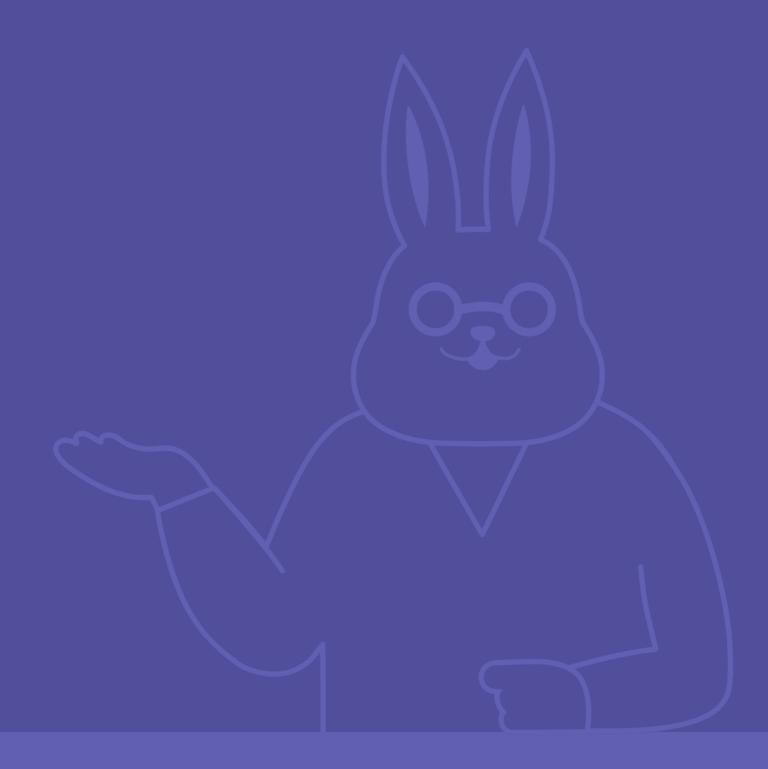
/* elice */



- 01. 자연어 처리
- 02. 텍스트 전처리
- 03. 단어 임베딩
- 04. word2vec
- 05. fastText

01

자연어 처리

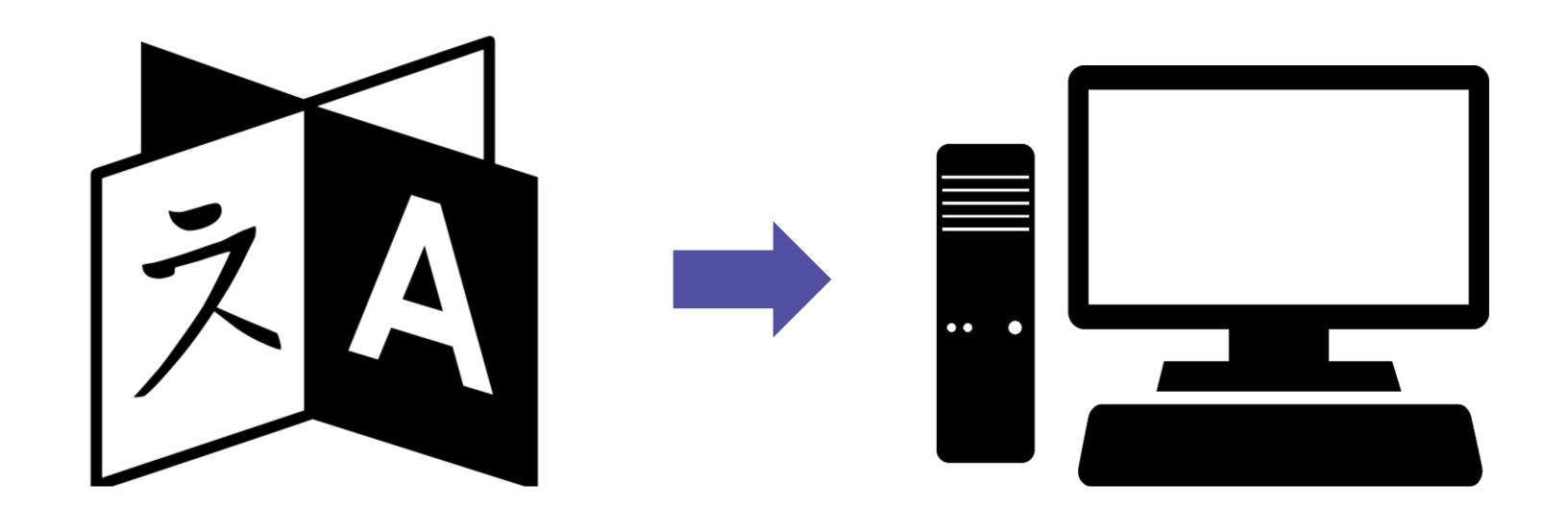


Confidential all rights reserved

/* elice */

01 자연어 처리 /* elice */

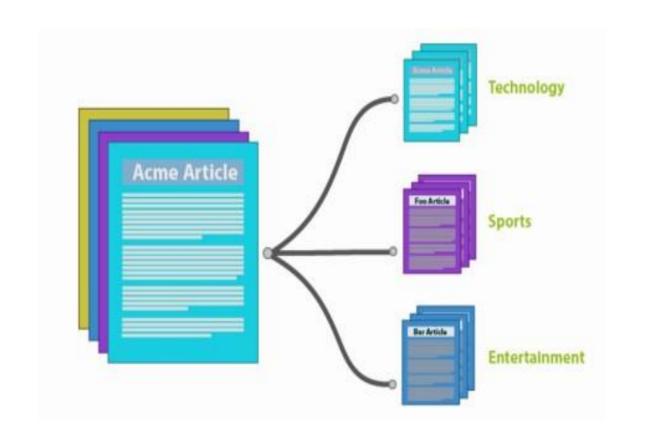
❷ 자연어 처리란

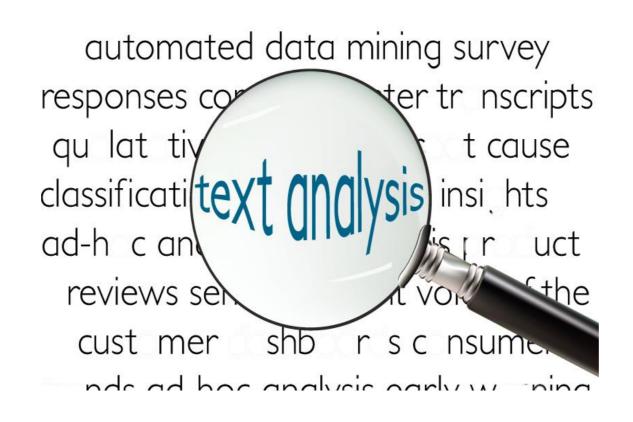


자연어 처리(Natural Language Processing, NLP)는 컴퓨터를 통해 인간의 언어를 분석 및 처리하는 인공지능의 한 분야

01 자연어 처리

☑ 자연어 처리의 적용 사례







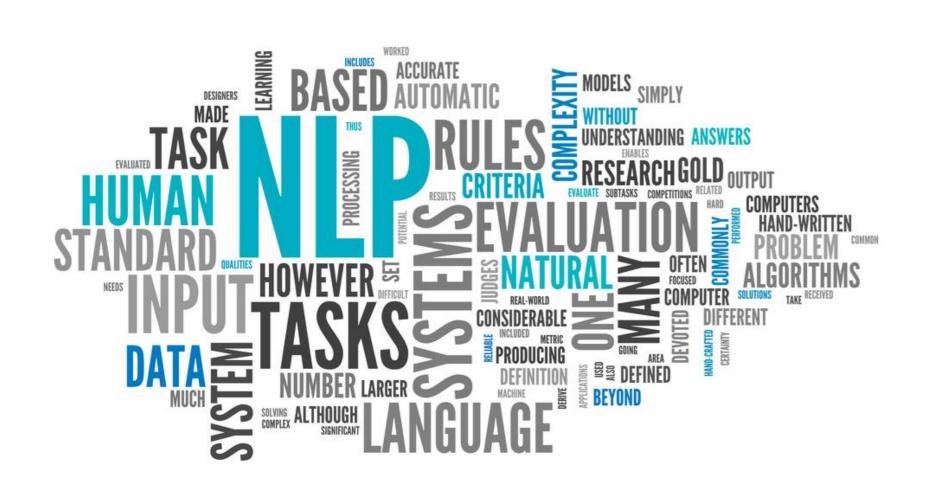
문서 분류

키워드 추출

감정 분석

01 자연어 처리 /* elice */

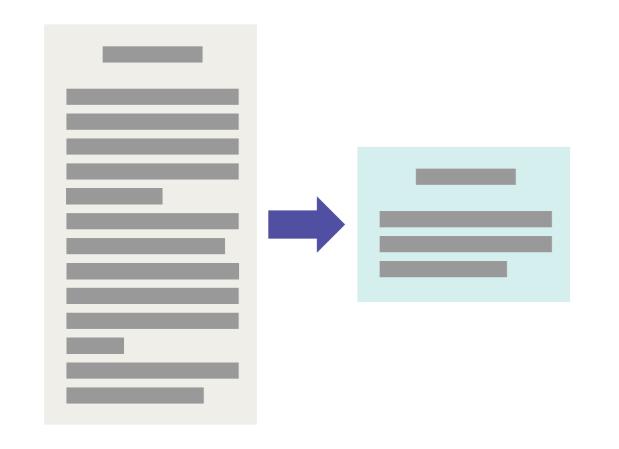
☑ 자연어 처리 + 머신러닝

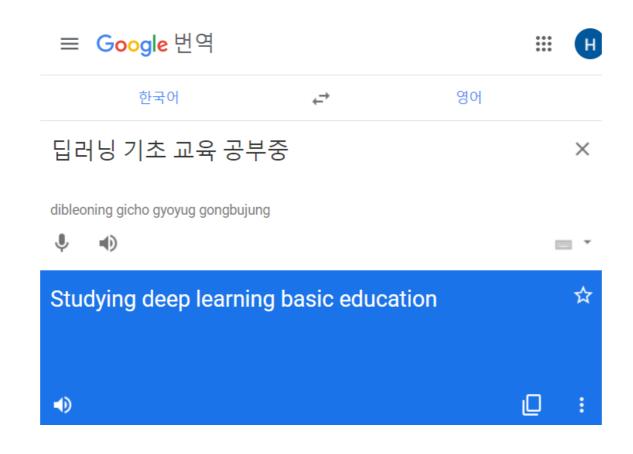




학습 가능한 데이터양의 증가 및 연산 처리 속도의 발전으로 자연어 처리 또한 더욱 복잡한 머신러닝 알고리즘 적용 가능 01 자연어 처리

❷ 머신러닝 기반 자연어 처리의 적용 사례



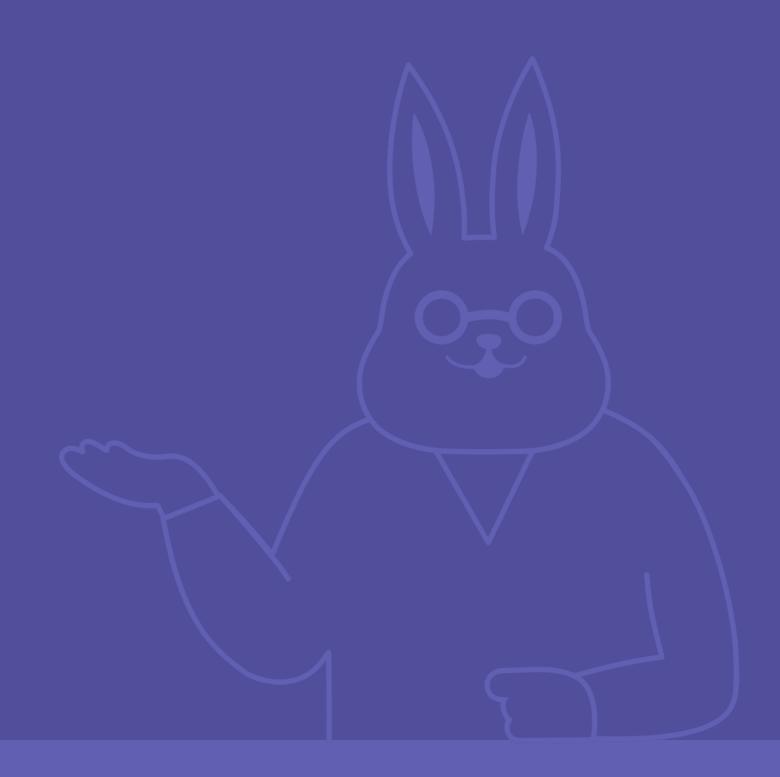




문서 요약

기계 번역

Chat bot



Confidential all rights reserved

☑ 모델링을 위한 데이터 탐색 및 전처리

데이터 탐색

데이터 통계치 변수별 특징

데이터 전처리

이상치 제거 정규화 (normalization)

❷ 모델링을 위한 데이터 탐색 및 전처리

데이터 탐색

단어의 개수 **단어**별 빈도수

• • •

데이터 전처리

특수기호 제거 단어 정규화

• • •

Tokenization

텍스트 1: Hello, my name is Elice! What is your name?

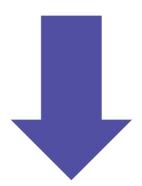


텍스트 1: Hello, my name is … name?

토큰화(tokenization)는 주어진 텍스트를 각 단어 기준으로 분리하는 것을 의미

Tokenization

텍스트 1: Hello, my name is Elice! What is your name?

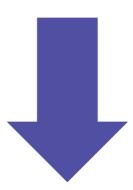


텍스트 1: Hello, my name is … name?

가장 기본적인 토큰화의 기준은 공백

Tokenization

텍스트 1: Hello, my name is Elice! What is your name?



텍스트 1: hello my name is … name

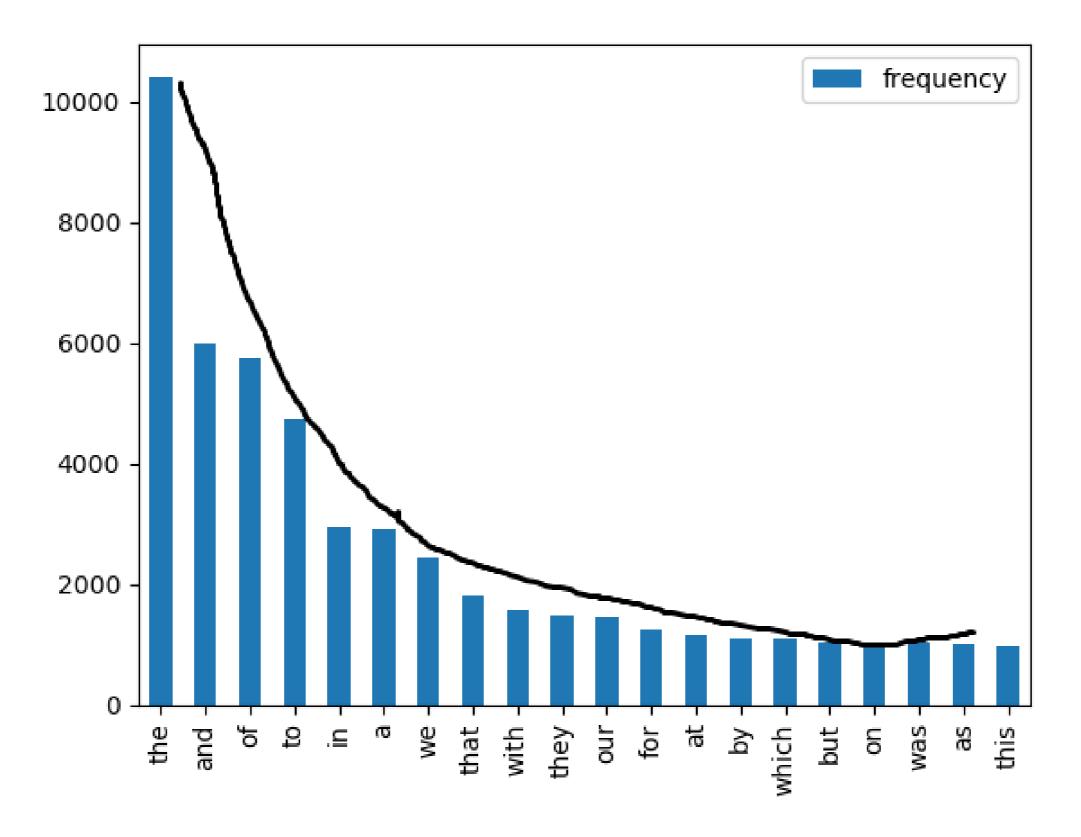
소문자 처리 및 특수기호 제거를 통해 동일한 의미의 토큰은 동일한 형태로 변환

❷ 단어의 개수 및 빈도수 확인

Example

```
counter = dict()
with open(파일명, 'r') as f:
    for line in f:
        for word in line.rstrip().split():
            if word not in word_counter:
                word_counter[word] = 1
            else:
                word_counter[word] += 1
```

☑ 자연어 처리 + 머신러닝



대부분 단어 빈도수의 분포는 지프의 법칙(Zipf's law)을 따름

☑ 전처리 I: 특수 기호 제거

Example

```
import re
word = "123hello993 $!%eli$@ce^"
regex = re.compile('[^a-z A-Z]')
print(regex.sub('', word))
# hello elice
```

re: 정규표현식 라이브러리

정규표현식: 정의하는 규칙을 가진 문자열 집합

☑ 전처리 II: Stopword 제거

[ir, 'me', 'my', 'myself', 'we', 'our', 'ourse, 'ourselves', 'you', "you're", "you've", "you'd", 'your', 'yours, 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'itself', 'they', 'them', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "haven', "haven', "haven't", 'isn', "isn't", 'ma', 'mightn't", 'mustn't", 'mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wouldn't"]

문법적인 기능을 지닌 단어 및 불필요하게 자주 발생하는 단어를 제거

☑ 전처리 II: Stopword 제거

Example

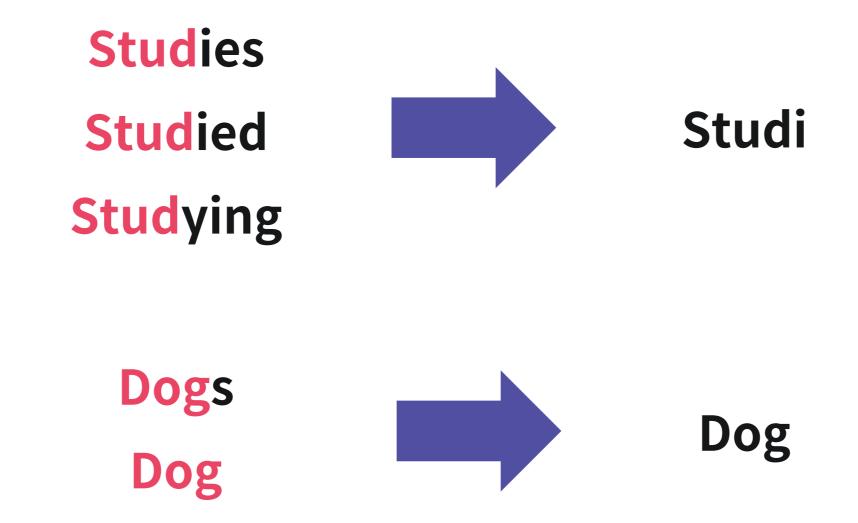
```
import nltk
from nltk.corpus import stopwords
sentence = ["the", "green", "egg", "and", "ham", "a", "an"]
stopwords = stopwords.words('english') # 리스트를 반환
new_sentence = [word for word in sentence if word not in stopwords]
print(new_sentence)
# ["green", "egg", "ham"]
```

☑ 전처리 II: Stopword 제거

Example

```
import nltk
from nltk.corpus import stopwords
new_stopwords = ["none", "는", "가"] # 신규 stopword
stopwords = stopwords.words('English') # 리스트를 반환
stopwords += new_stopwords
```

☑ 전처리 II: Stemming



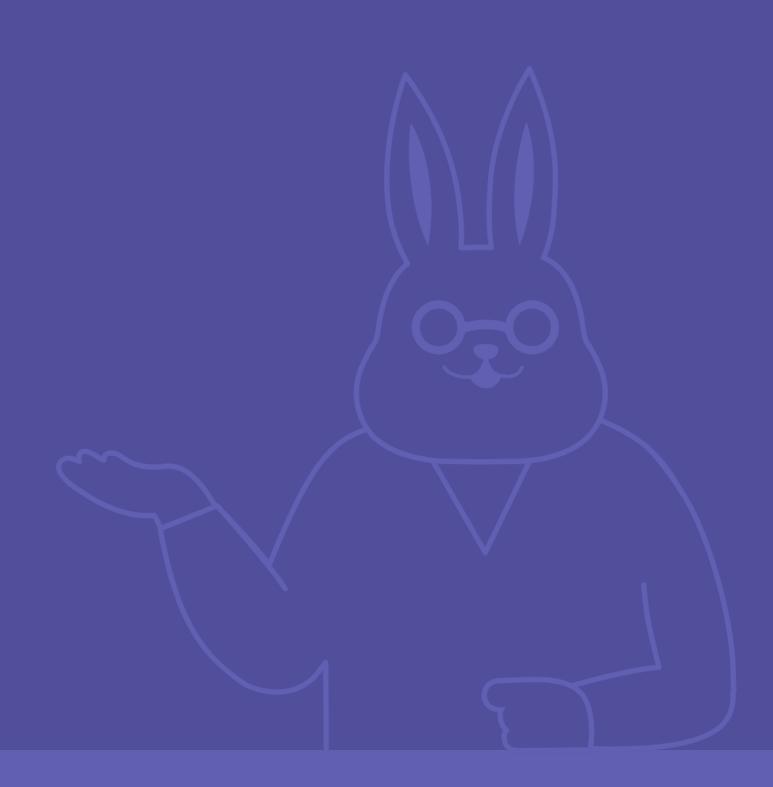
동일한 의미의 단어이지만, 문법적인 이유 등 표현 방식이 다양한 단어를 공통된 형태로 변환

☑ 전처리 II: Stemming

Example

```
import nltk
from nltk.stem import PorterStemmer
words = ["studies", "studied", "studying", "dogs", "dog"]
stemmer = PorterStemmer()
for word in words:
   print(stemmer.stem(word)) # studi, studi, studi, dog, dog
```

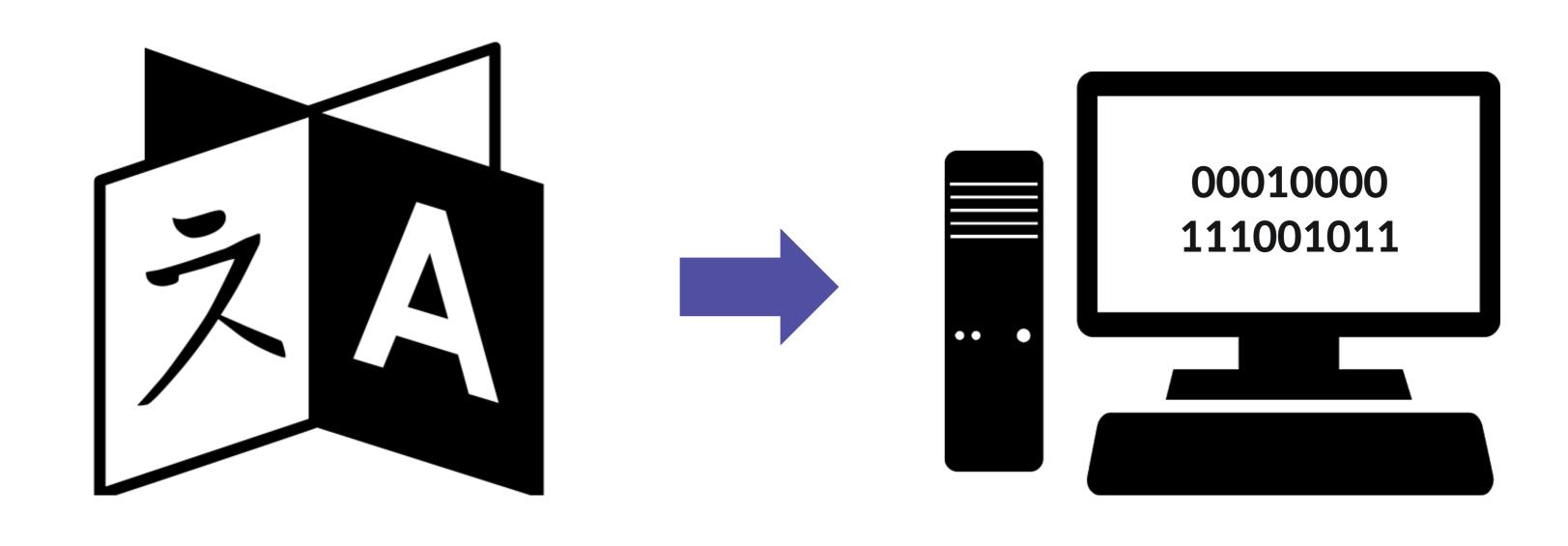
단어 임베딩



Confidential all rights reserved

03 단어임베딩

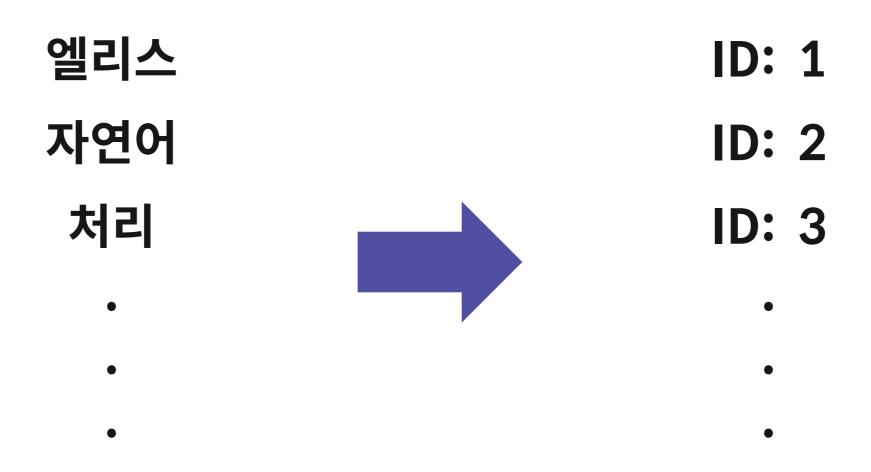
❷ 컴퓨터 - 언어



컴퓨터는 텍스트를 포함하여 모든 데이터를 0과 1로 처리

03 단어 임베딩 /* elice */

☑ 컴퓨터 - 언어

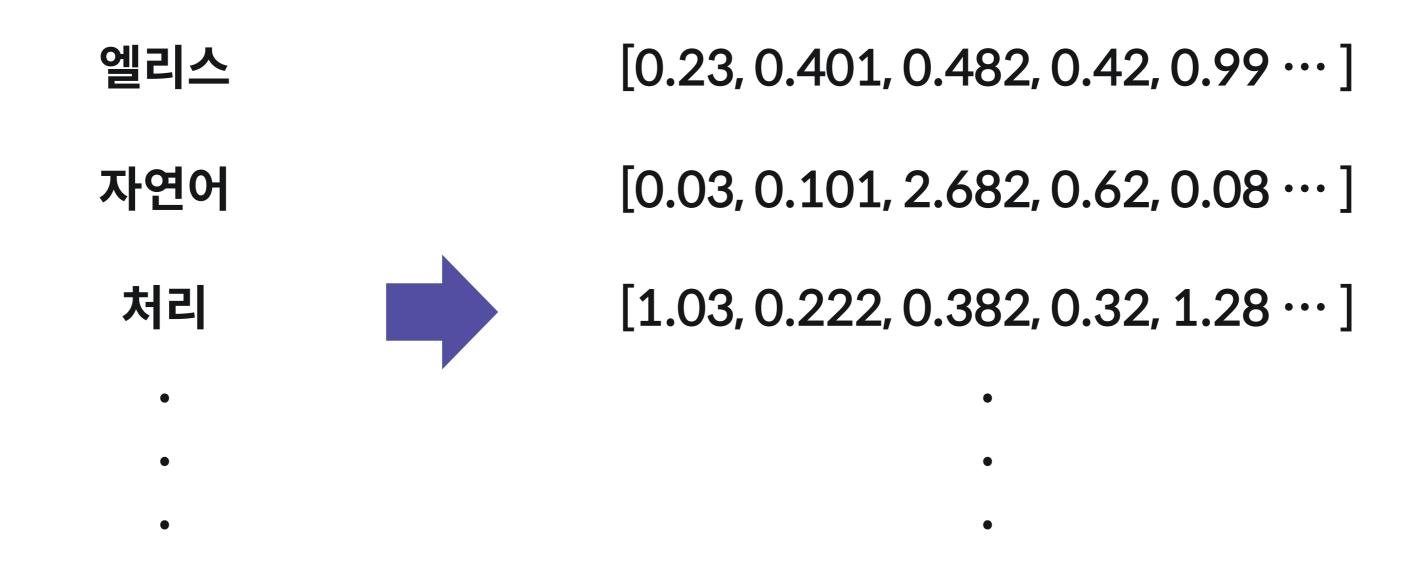


자연어의 기본 단위인 단어를 수치형 데이터로 표현하는 것이 중요

03 단어임베딩

/*elice*/

♥ 단어 임베딩



단어 임베딩이란 각 단어를 연속형 벡터로 표현하는 방법을 의미



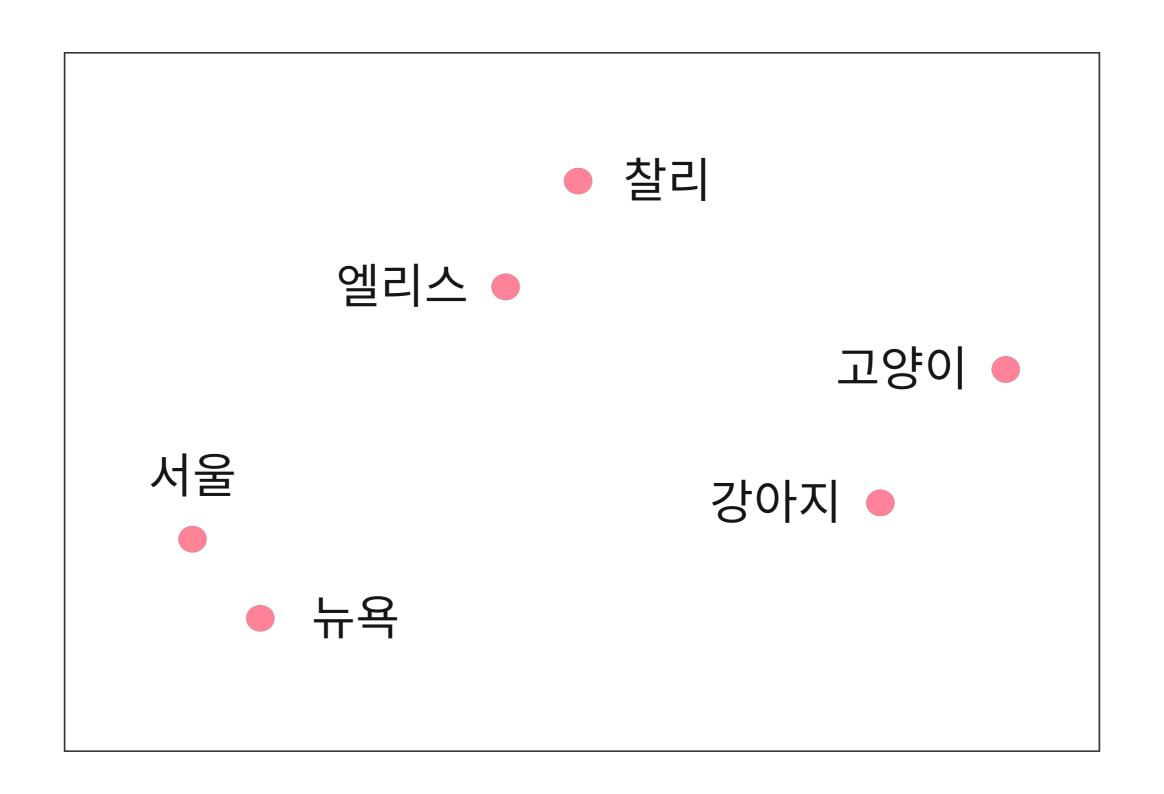
서울에 살고 있는 엘리스는 강아지를 좋아한다.

뉴욕에 살고 있는 찰리는 고양이를 좋아한다.

비슷한 문맥에서 발생하는 단어는 유사한 의미를 지님

03 단어임베딩

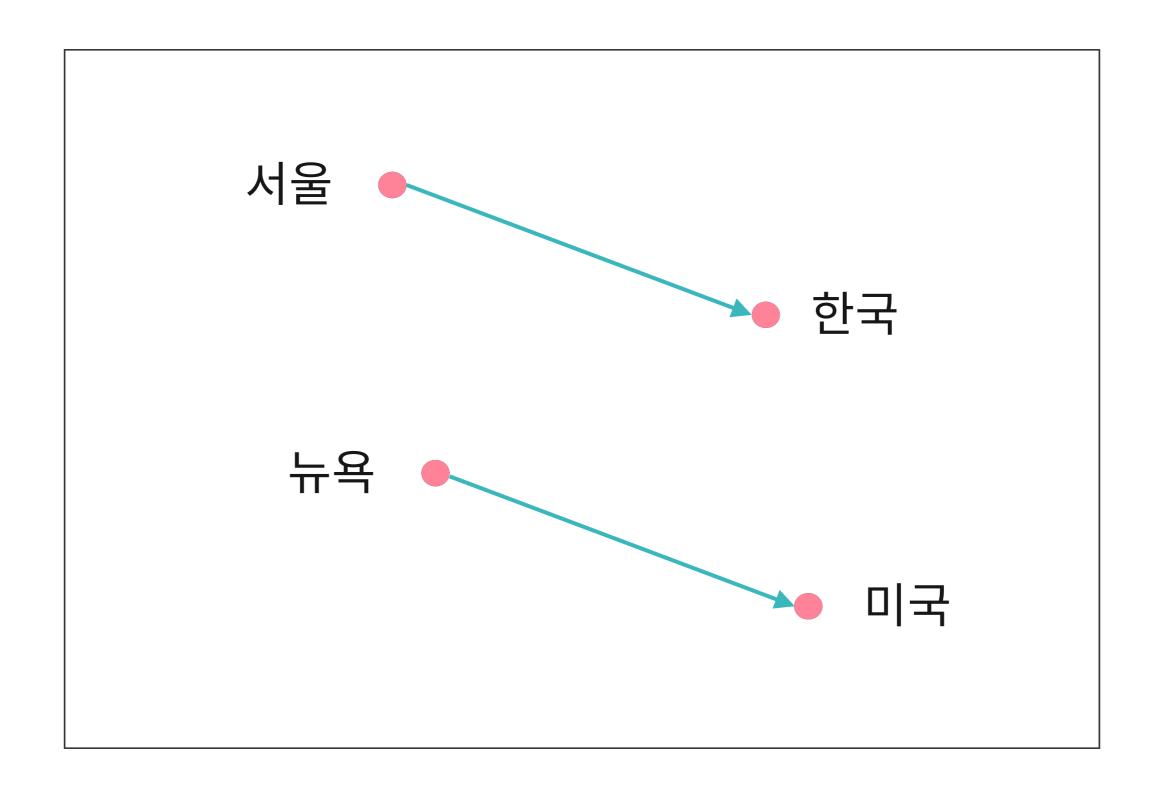
❷ 단어 임베딩



유사한 단어의 임베딩 벡터는 인접한 공간에 위치

03 단어임베딩

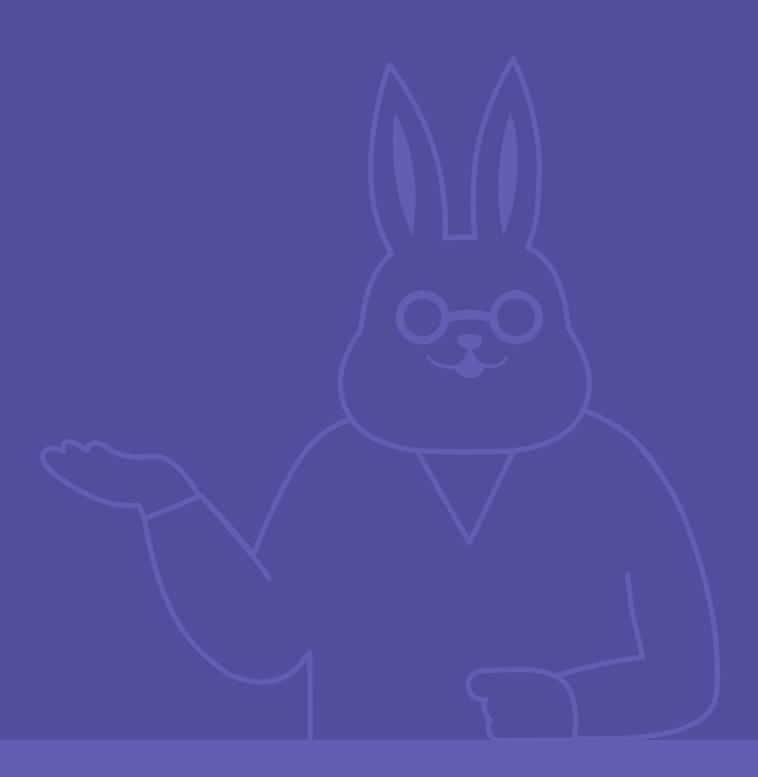
❷ 단어 임베딩



임베딩 벡터 간 합과 차로 단어의 의미적 특징을 활용 가능

04

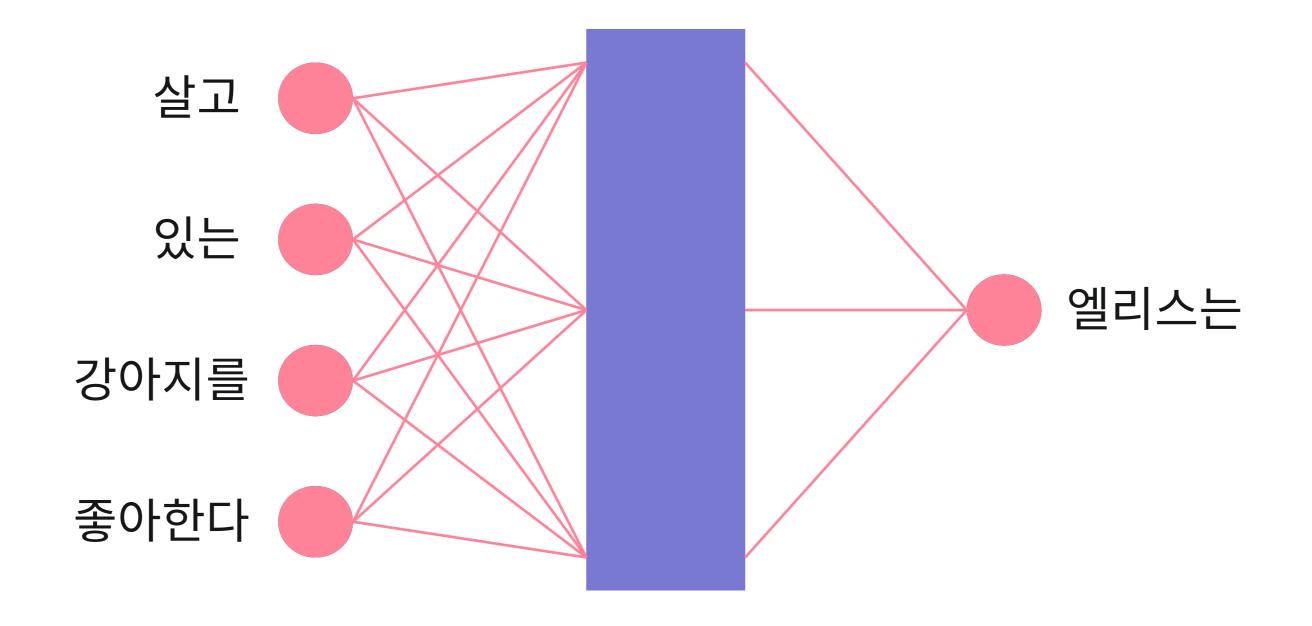
word2vec



Confidential all rights reserved

word2vec

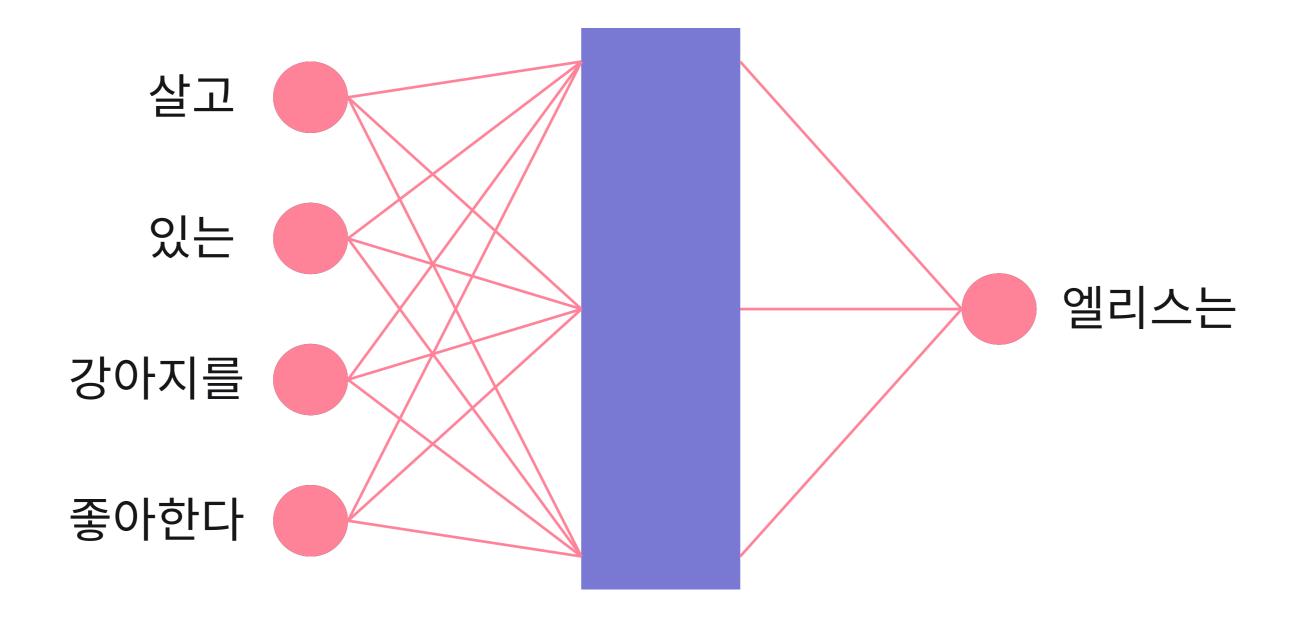
서울에 살고 있는 엘리스는 강아지를 좋아한다.



신경망(Neural Network)을 통해 단어 임베딩 벡터를 학습

word2vec

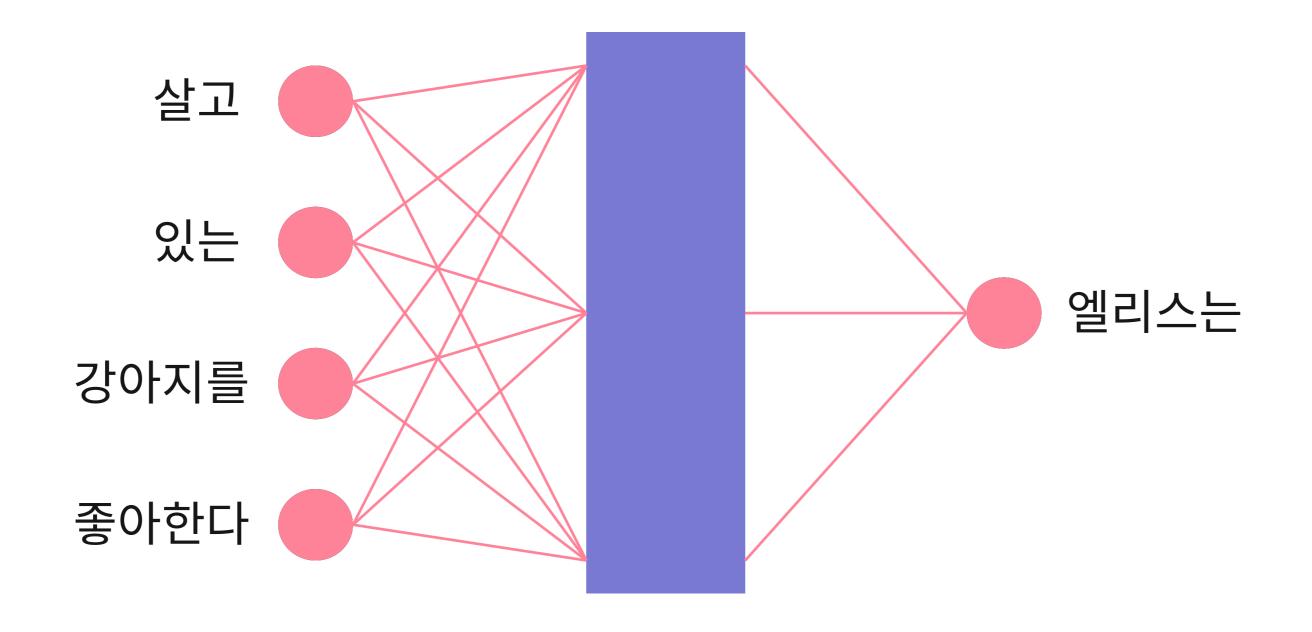
서울에 살고 있는 엘리스는 강아지를 좋아한다.



주어진 문맥에서 발생하는 단어를 예측하는 문제로 통해 단어 임베딩 벡터를 학습

word2vec

서울에 살고 있는 엘리스는 강아지를 좋아한다.



각 단어의 벡터는 해당 단어가 입력으로 주어졌을 때 계산되는 은닉층의 값을 사용

gensim

Example

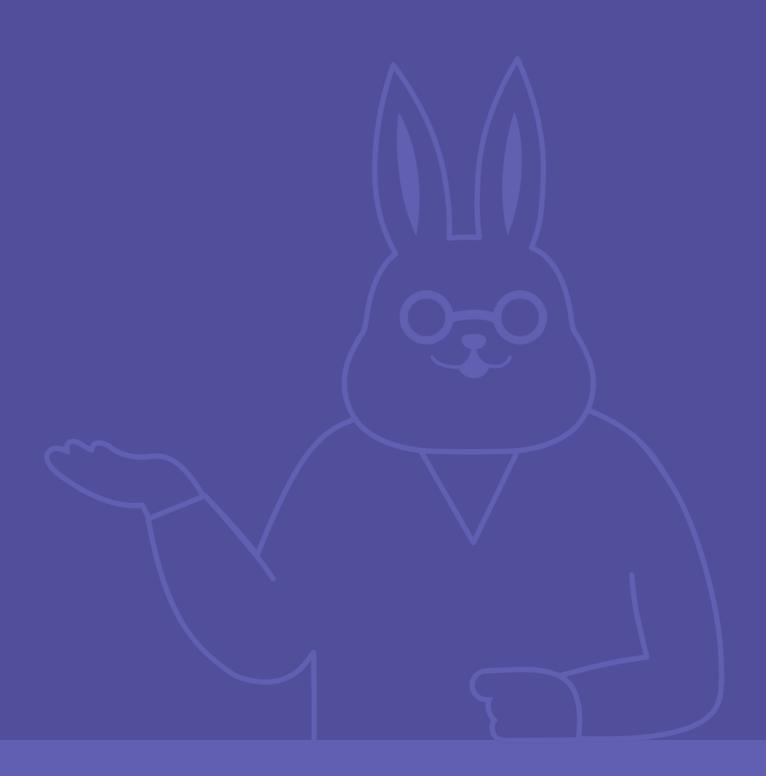
```
from gensim.models import Word2Vec
doc = [["서울에", "살고", "있는", "엘리스는", "강아지를", "좋아한다"]]
w2v_model = Word2Vec(min_count=1, window=2, vector_size=300)
w2v_model.build_vocab(doc)
w2v_model.train(doc, total_examples=w2v_model.corpus_count, epochs=20)
```

gensim

Example

```
similar_word = w2v_model.wv.most_similar("엘리스는")
print(similar_word)
# [('있는', 0.05005083233118057), ('좋아한다', 0.03316839784383774),
   ('강아지를', 0.025744464248418808), ('서울에', 0.013042463921010494),
   ('살고', -0.0342760793864727)]
score = w2v_model.wv.similarity("엘리스는", "좋아한다")
print(score)
# 0.03316839784383774
```

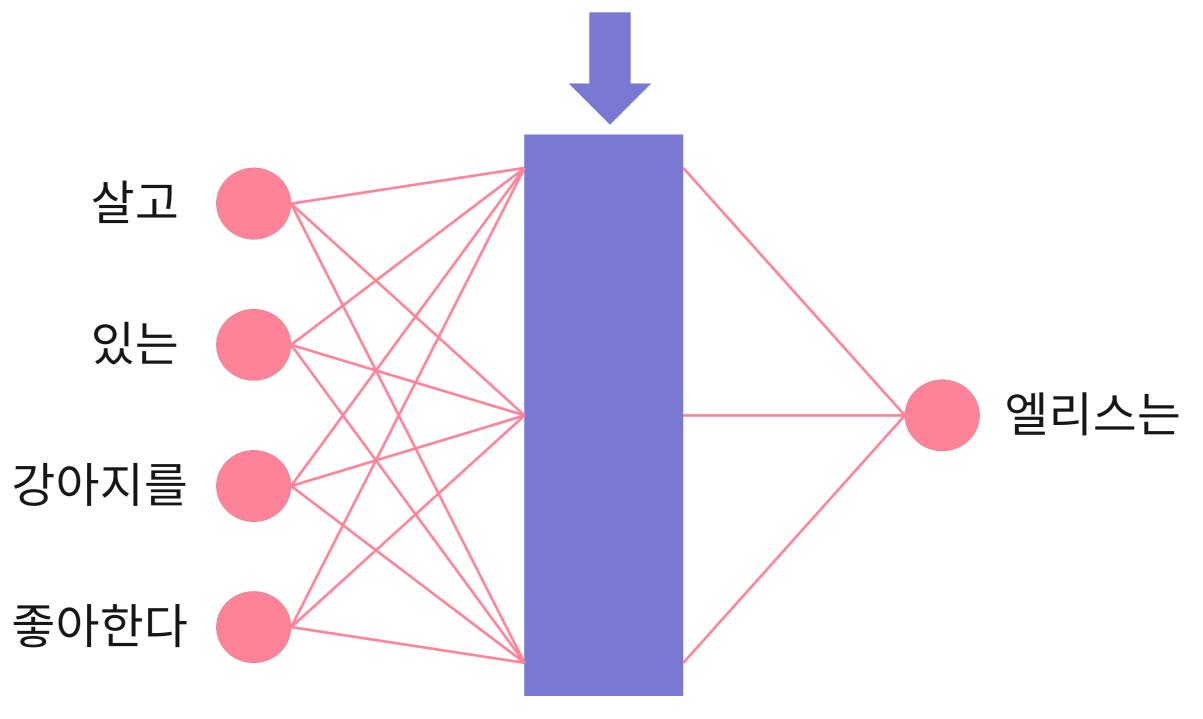
os fastText



Confidential all rights reserved

fastText

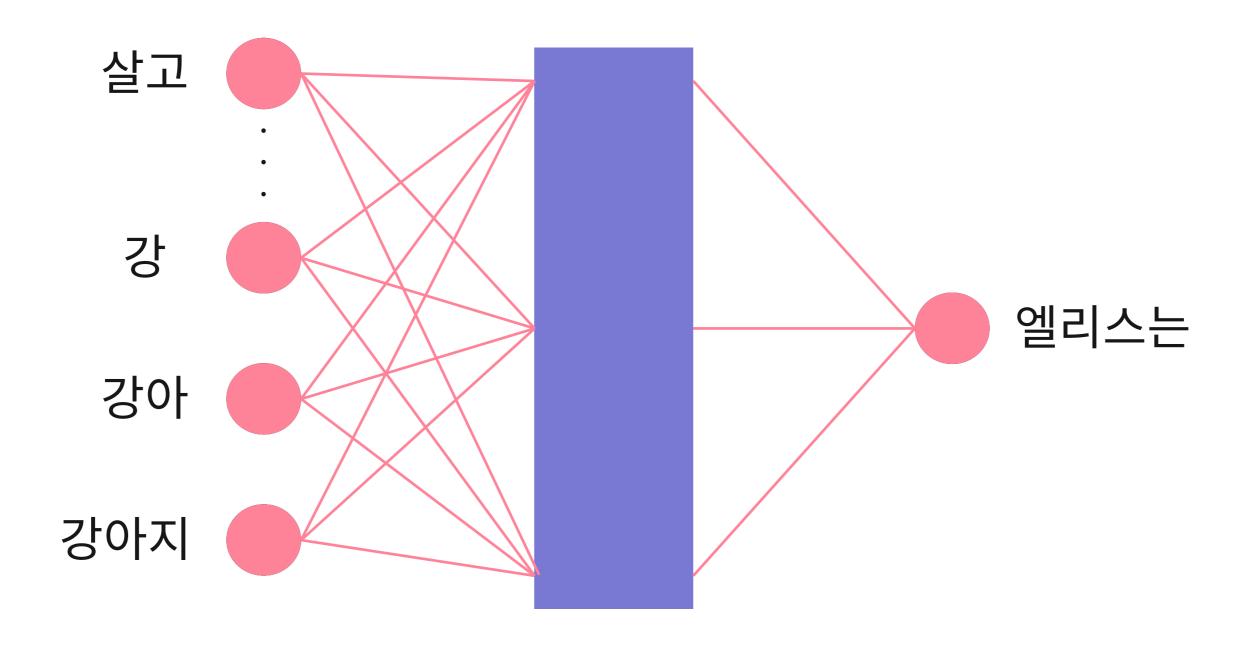
뉴욕에 살고 있는 찰리는 고양이를 좋아한다.



학습 데이터 내 존재하지 않았던 단어 벡터는 생성할 수 없음 (미등록 단어 문제, out-of-vocabulary)

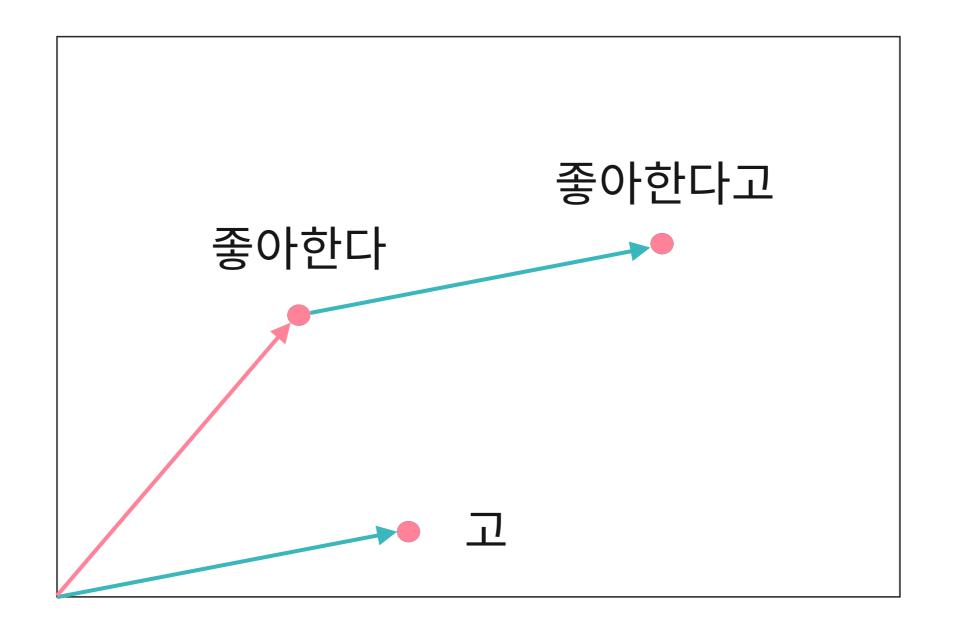
fastText

서울에 살고 있는 엘리스는 강아지를 좋아한다.



각 단어를 문자 단위로 나누어서 단어 임베딩 벡터를 학습

서울에 살고 있는 엘리스는 강아지를 좋아한다.



학습 데이터에 존재하지 않았던 단어의 임베딩 벡터 또한 생성 가능

gensim

Example

```
from gensim.models import FastText
doc = [["서울에", "살고", "있는", "엘리스는", "강아지를", "좋아한다"]]
ft_model = FastText(min_count=1, window=2, vector_size=300)
ft_model.build_vocab(doc)
ft_model.train(doc, total_examples=ft_model.corpus_count, epochs=20)
```

gensim

Example

```
similar_word = ft_model.wv.most_similar("엘리스는")
print(similar_word)
# [('좋아한다', 0.03110547922551632), ('살고', 0.015657681971788406),
 ('강아지를', -0.09297232329845428), ('서울에', -0.10255782306194305),
  ('있는', -0.10588616132736206)]
new_vector = ft_model.wv["좋아한다고"]
print(new_vector)
# array([-5.8544584e-04, -1.5485507e-03, -1.3994898e-03, -9.1309723e-04, ...
```

크레딧

/* elice */

코스 매니저 이재성

콘텐츠 제작자 김한결

강사 김한결

감수자 김한결

디자이너 강혜정

연락처

TEL

070-4633-2015

WEB

https://elice.io

E-MAIL

contact@elice.io

