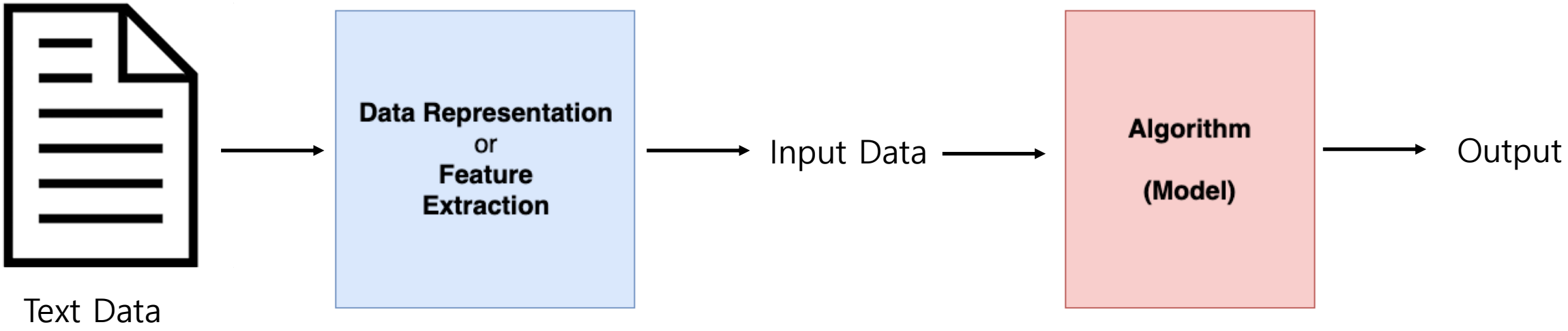
텐서플로와 머신러닝으로 시작하는 자연어처리

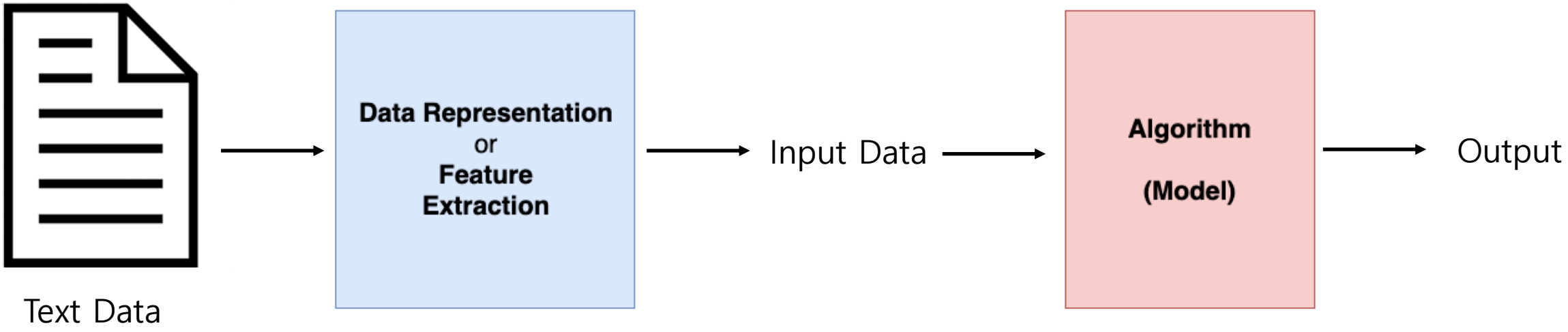
Lecture 4
2019. 06. 22

Lecture3 Review

Lecture3 Review

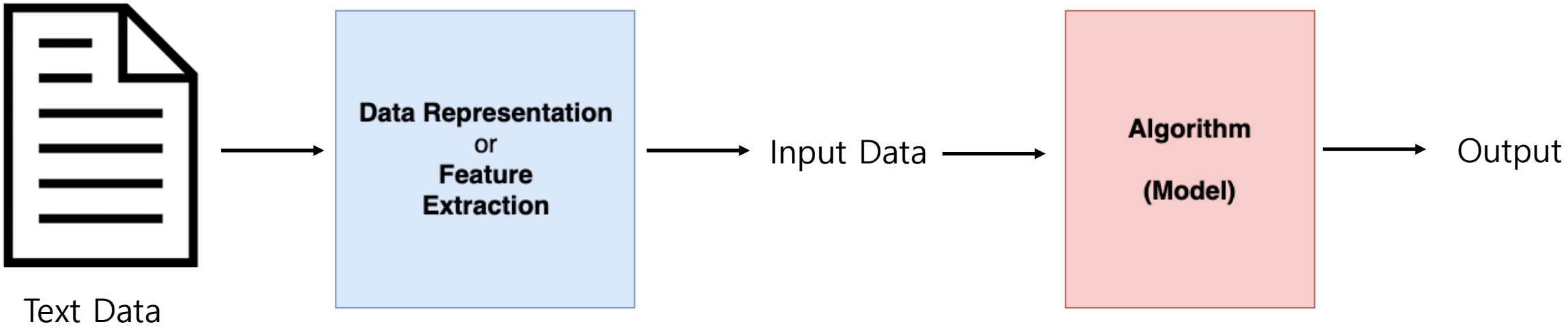


Lecture3 Review



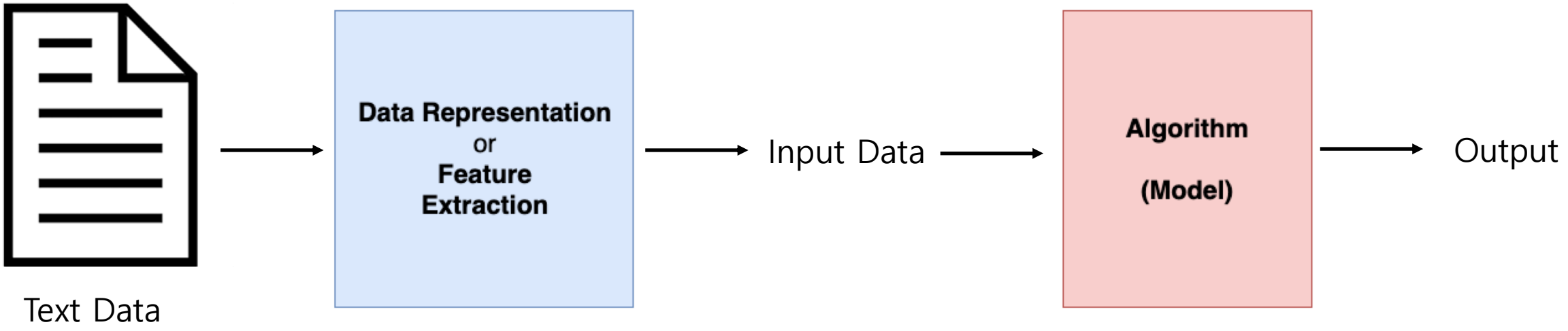
- Embedding vector

Lecture3 Review



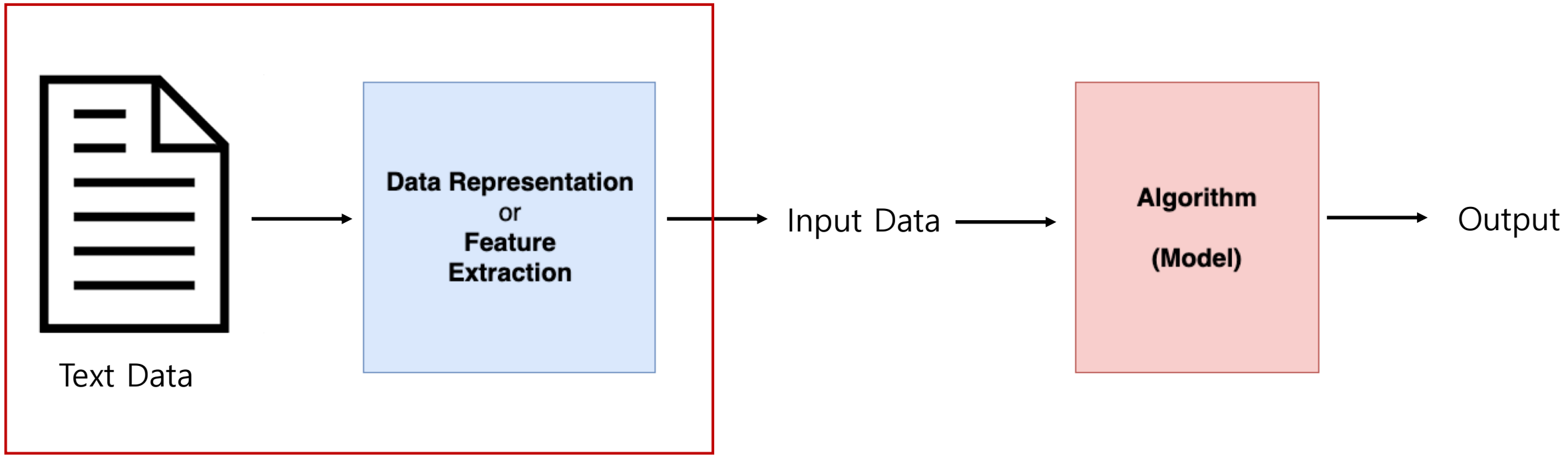
- Embedding vector
  - Pre-trained word2vec
  - Train while model training

Lecture3 Review

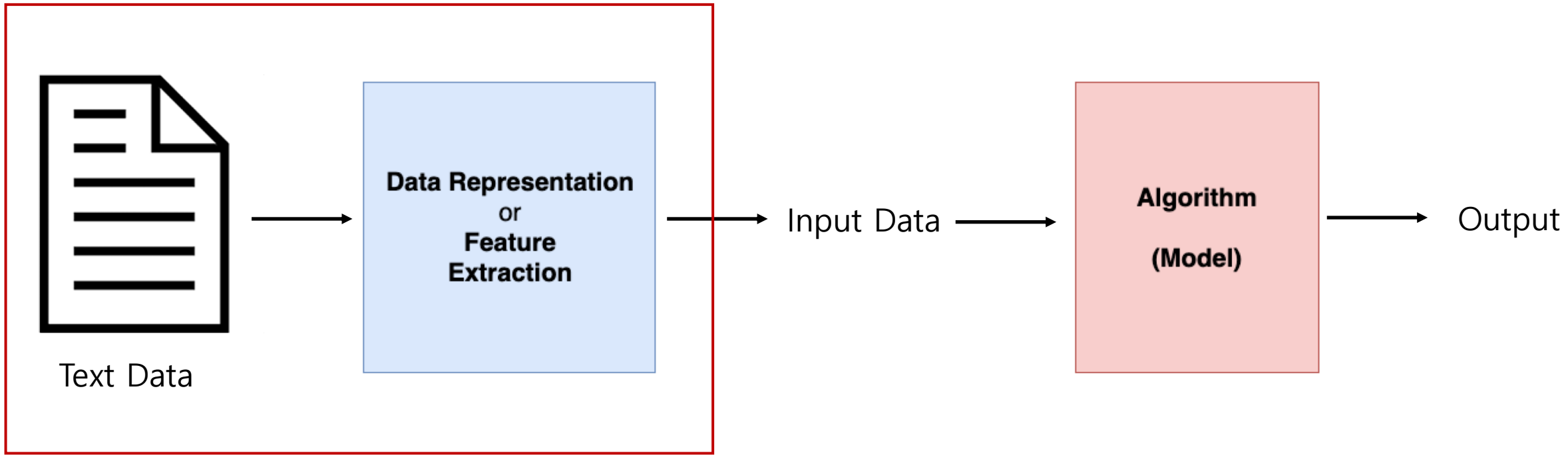


- Convolutional Neural Netwroks
- etc...

Lecture3 Review

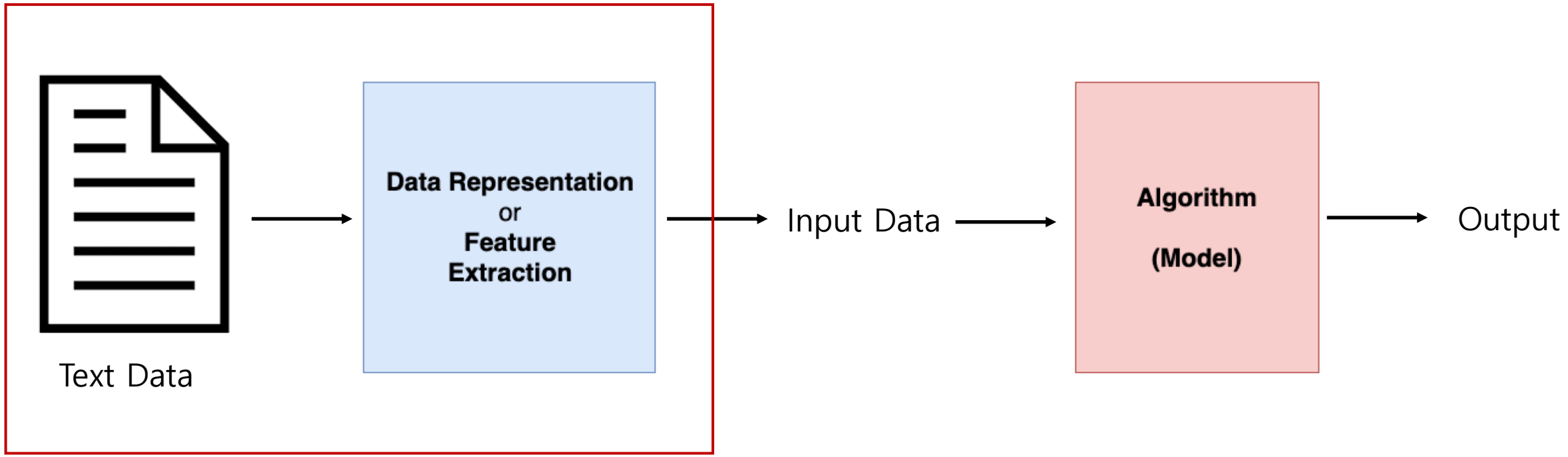


Lecture3 Review



- index vector
- embedding vector
- one hot / BoW

Lecture3 Review



- index vector ?
- embedding vector ?
- one hot / BoW ?

Alpha

TF-IDF ?

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = f_{t,d}$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = \begin{cases} 1, & \text{if t occur in d} \\ 0, & \text{otherwise} \end{cases}$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = log(1 + f_{t,d})$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Alpha

TF-IDF ?

Term-Frequency & Inverse Document Frequency

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$df(t, D) = \frac{|\{d \in D : t \in d\}|}{|D|}$$

Text Similarity

Text Similarity

- What is Text Similarity?
- Why do we need Text Similarity?
- Similarity 측정
- Measuring Similarity by Classification
- Natural Language Inference
- Natural Language Inference Dataset
- Similarity Algorithm
    - Ma-LSTM(Paper)
    - XG-Boost

What is Text Similarity

What is Text Similarity

- 주어진 한 쌍의 Text가 얼마나 유사한지 측정하는 Task
- 유사한지 측정하는 기준은 의미적(Semantic) or 문법적(Syntactic) or 어휘적(Lexical) 기준으로 분류
- 의미적으로 유사한지 측정하는 Task를 Semantic Textual Similarity Task라 부름

What is Text Similarity

- 주어진 한 쌍의 Text가 얼마나 유사한지 측정하는 Task
- 유사한지 측정하는 기준은 의미적(Semantic) or 문법적(Syntactic) or 어휘적(Lexical) 기준으로 분류
- 의미적으로 유사한지 측정하는 Task를 Semantic Textual Similarity Task라 부름

Text Similarity Demo

Why do we need Text Similarity

Why do we need Text Similarity

Similarity Application

Why do we need Text Similarity

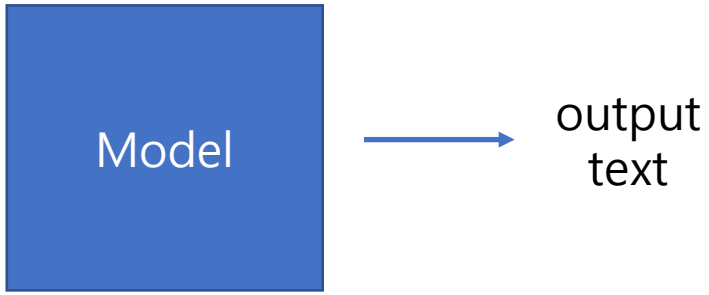
Similarity Application

- **Automatic Evaluation**

Why do we need Text Similarity

Similarity Application

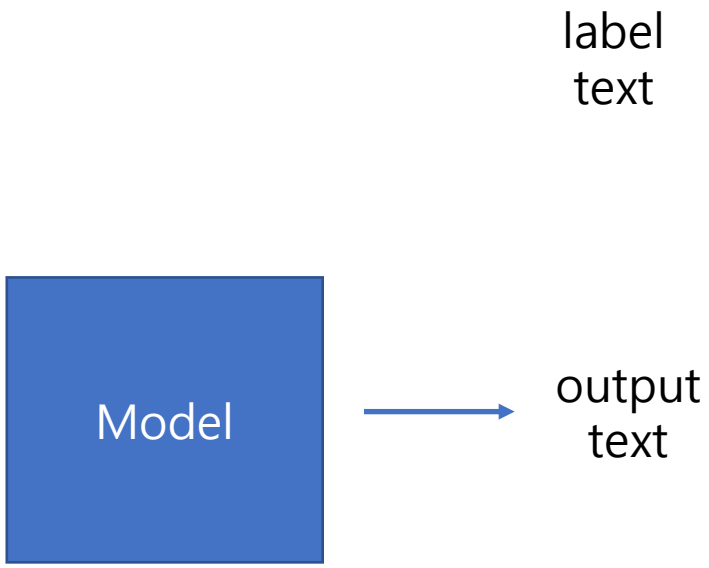
- **Automatic Evaluation**



Why do we need Text Similarity

Similarity Application

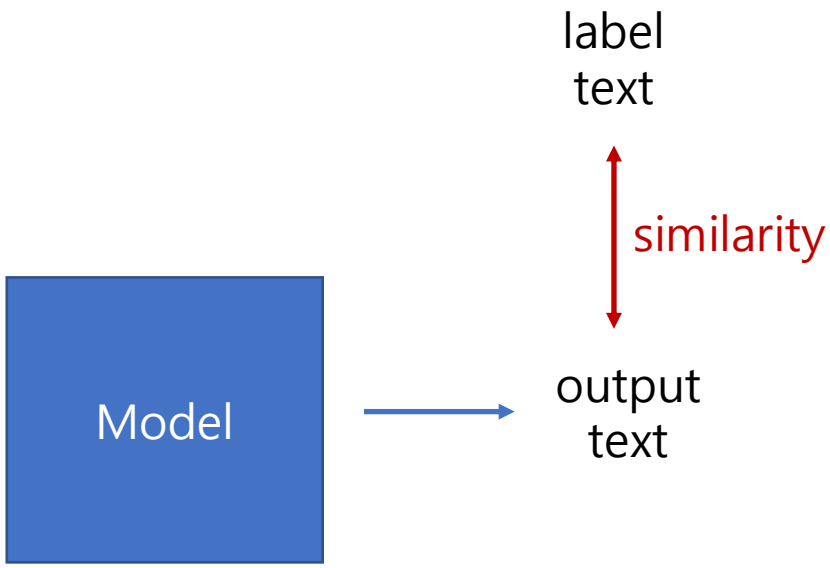
- **Automatic Evaluation**



Why do we need Text Similarity

Similarity Application

- **Automatic Evaluation**



Why do we need Text Similarity

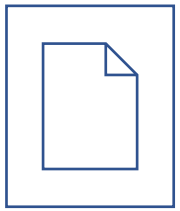
Similarity Application

- Automatic Evaluation
- **Information Retrieval (Search Engine)**
- Text Classification & Clusturing

Why do we need Text Similarity

Similarity Application

- Automatic Evaluation
- **Information Retrieval (Search Engine)**
- Text Classification & Clusturing

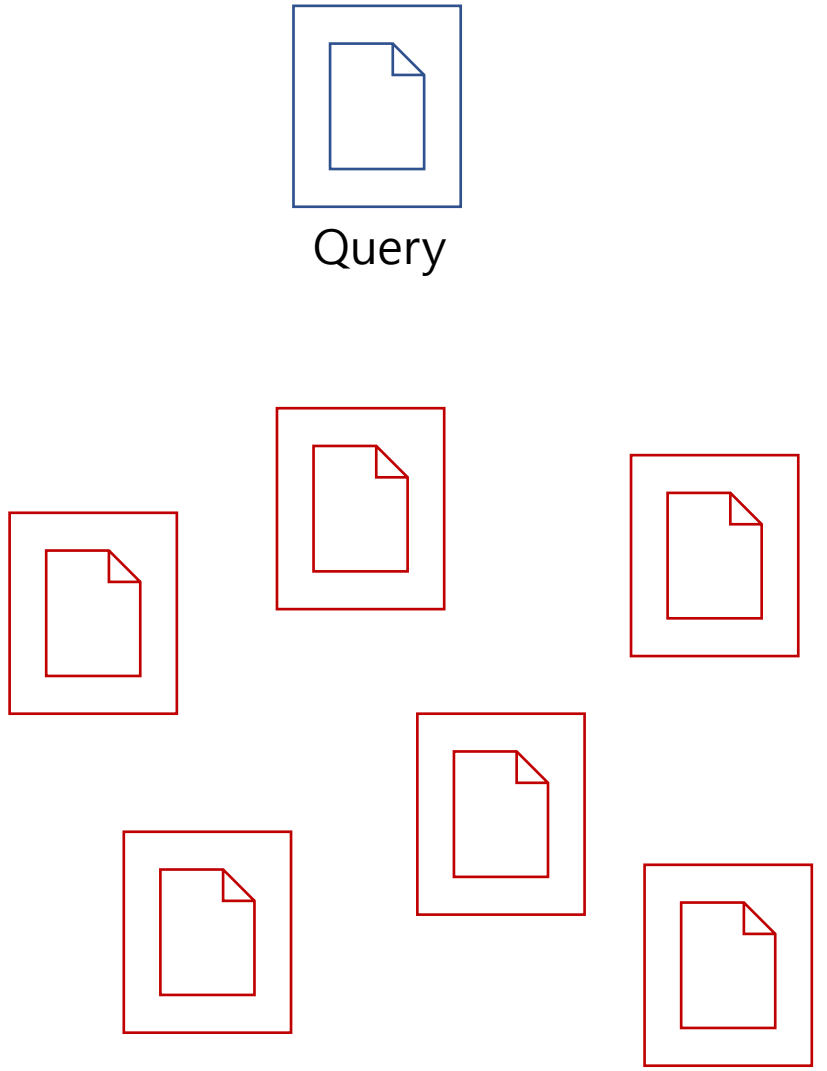


Query

Why do we need Text Similarity

Similarity Application

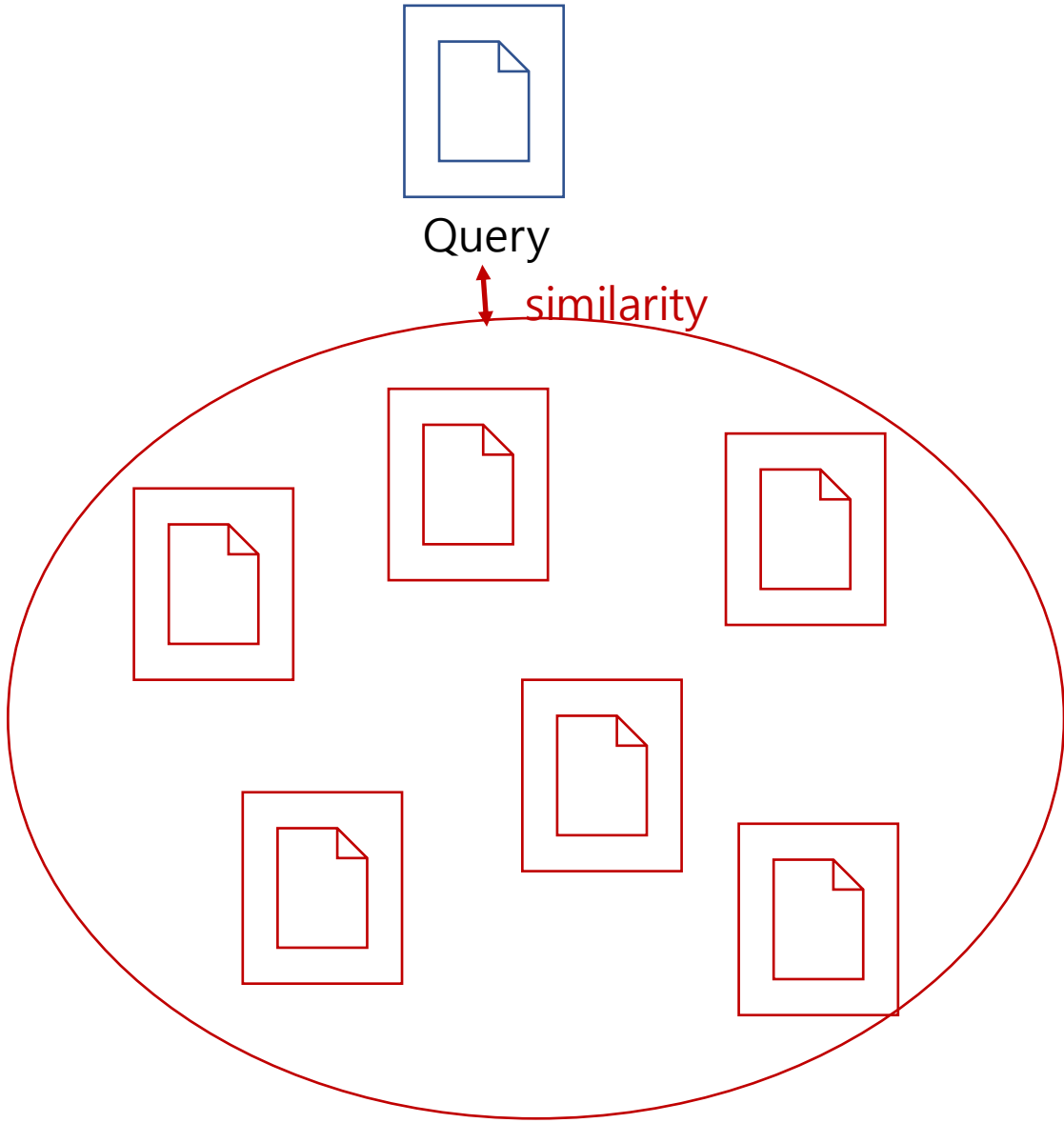
- Automatic Evaluation
- **Information Retrieval (Search Engine)**
- Text Classification & Clusturing



Why do we need Text Similarity

Similarity Application

- Automatic Evaluation
- **Information Retrieval (Search Engine)**
- Text Classification & Clusturing



Why do we need Text Similarity

Similarity Application

- Automatic Evaluation
- Information Retrieval (Search Engine)
- **Text Classification & Clusturing**

Text Similarity Dataset

Text Similarity Dataset

Semantic Textual Similarity

- SentEval
- Quora Question Pairs

Text Similarity Dataset

Semantic Textual Similarity

- SentEval
- Quora Question Pairs

SentEval

- SentEval is a library for evaluating the quality of sentence embeddings.
- It includes 17 downstream tasks, including common semantic textual similarity tasks.
- STS12 ~ STS16, STSB 데이터셋 존재

Text Similarity Dataset

Semantic Textual Similarity

- SentEval
- Quora Question Pairs

Quora Question Pairs

- 질문 사이트인 Quora의 질문 Title로 구성
- 한 쌍의 질문이 중복이 판단하기 위한 Data
- 전체 400,000 쌍의 질문

id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Similarity 측정

Similarity 측정

1. Jaccard Similarity
2. Measuring Similarity by Distance
    - Cosine Distance
    - Euclidean Distance
    - Manhattan Distance

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$A, B : the\ set\ of\ Sentence(or\ Word)$

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

A, B : $the\ set\ of\ Sentence(or\ Word)$

Example)

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$A, B : the\ set\ of\ Sentence(or\ Word)$

Example)

A: 텐서플로우, 케라스, 파이토치의 차이점을 알려주세요!

B: 딥러닝 프레임 워크(텐서플로우, 케라스, 파이토치)들은 어떤 차이점이 있나요?

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$A, B : the\ set\ of\ Sentence(or\ Word)$

Example)

A: 텐서플로우, 케라스, 파이토치의 차이점을 알려주세요!

B: 딥러닝 프레임 워크(텐서플로우, 케라스, 파이토치)들은 어떤 차이점이 있나요?

Jaccard Similarity

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$$A, B : the\ set\ of\ Sentence(or\ Word)$$

Example)

A: 텐서플로우, 케라스, 파이토치의 차이점을 알려주세요!

B: 딥러닝 프레임 워크(텐서플로우, 케라스, 파이토치)들은 어떤 차이점이 있나요?

Jaccard Similarity

$A \cap B$ = {텐서플로우, 케라스, 파이토치, 차이점}

$A \cup B$ = {딥러닝, 프레임, 워크, 알려주세요, 어떤, 들, 은, 을, 있나요}

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$A, B : the\ set\ of\ Sentence(or\ Word)$

Example)

A: 텐서플로우, 케라스, 파이토치의 차이점을 알려주세요!

B: 딥러닝 프레임 워크(텐서플로우, 케라스, 파이토치)들은 어떤 차이점이 있나요?

Jaccard Similarity

$A \cap B$ = {텐서플로우, 케라스, 파이토치, 차이점}

$A \cup B$ = {딥러닝, 프레임, 워크, 알려주세요, 어떤, 들, 은, 을, 있나요}

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{1}{2} = 0.5$$

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

A, B : $the\ set\ of\ Sentence(or\ Word)$

Example)

A: 딥러닝

B: 머신러닝

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$A, B : the\ set\ of\ Sentence(or\ Word)$

Example)

A: 딥러닝

B: 머신러닝

Jaccard Similarity

$A \cap B$ = {러, 닝}

$A \cup B$ = {딥, 러, 닝, 머, 신}

Similarity 측정

Jaccard Similarity

두 문장(or 단어)의 집합에서의 교집합의 크기를 측정함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}$$

$A, B : the\ set\ of\ Sentence(or\ Word)$

Example)

A: 딥러닝

B: 머신러닝

Jaccard Similarity

$A \cap B = \{$러, 닝$\}$

$A \cup B = \{$딥, 러, 닝, 머, 신$\}$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{2}{5} = 0.4$$

Similarity 측정

Measuring Similarity by Distance

주어진 문장(or 단어)를 고정된 길이의 벡터로 만든 후

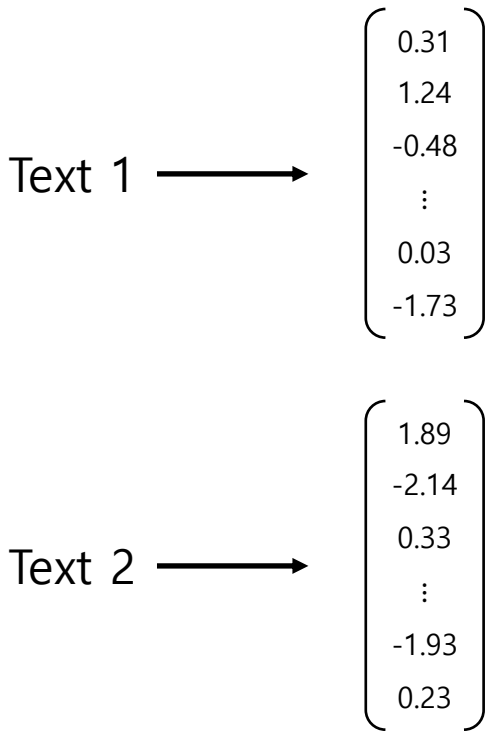
두 벡터 사이의 거리(Distance)를 측정

Similarity 측정

Measuring Similarity by Distance

주어진 문장(or 단어)를 고정된 길이의 벡터로 만든 후

두 벡터 사이의 거리(Distance)를 측정

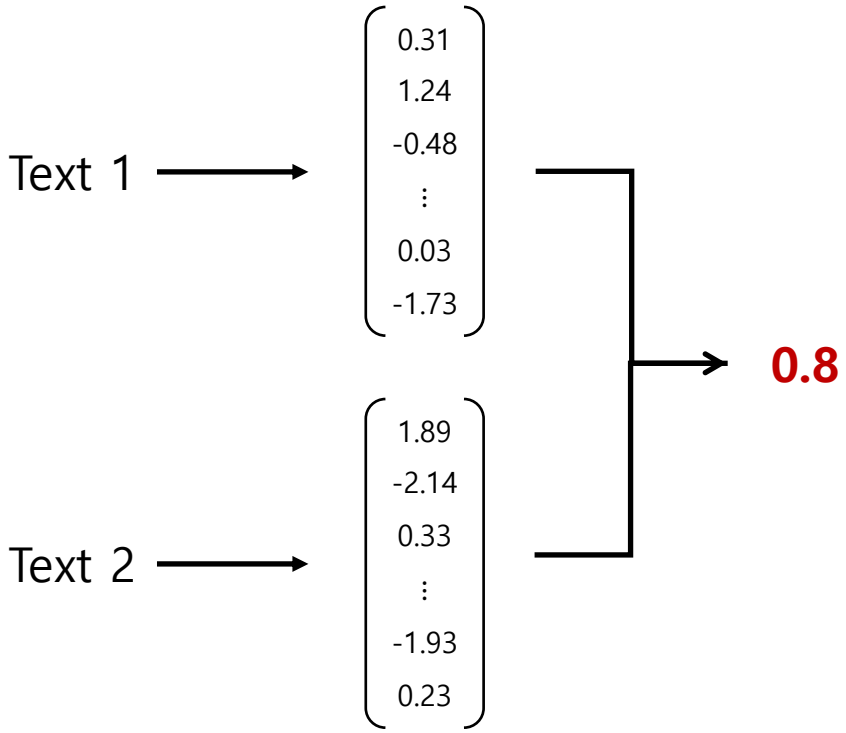


Similarity 측정

Measuring Similarity by Distance

주어진 문장(or 단어)를 고정된 길이의 벡터로 만든 후

두 벡터 사이의 거리(Distance)를 측정

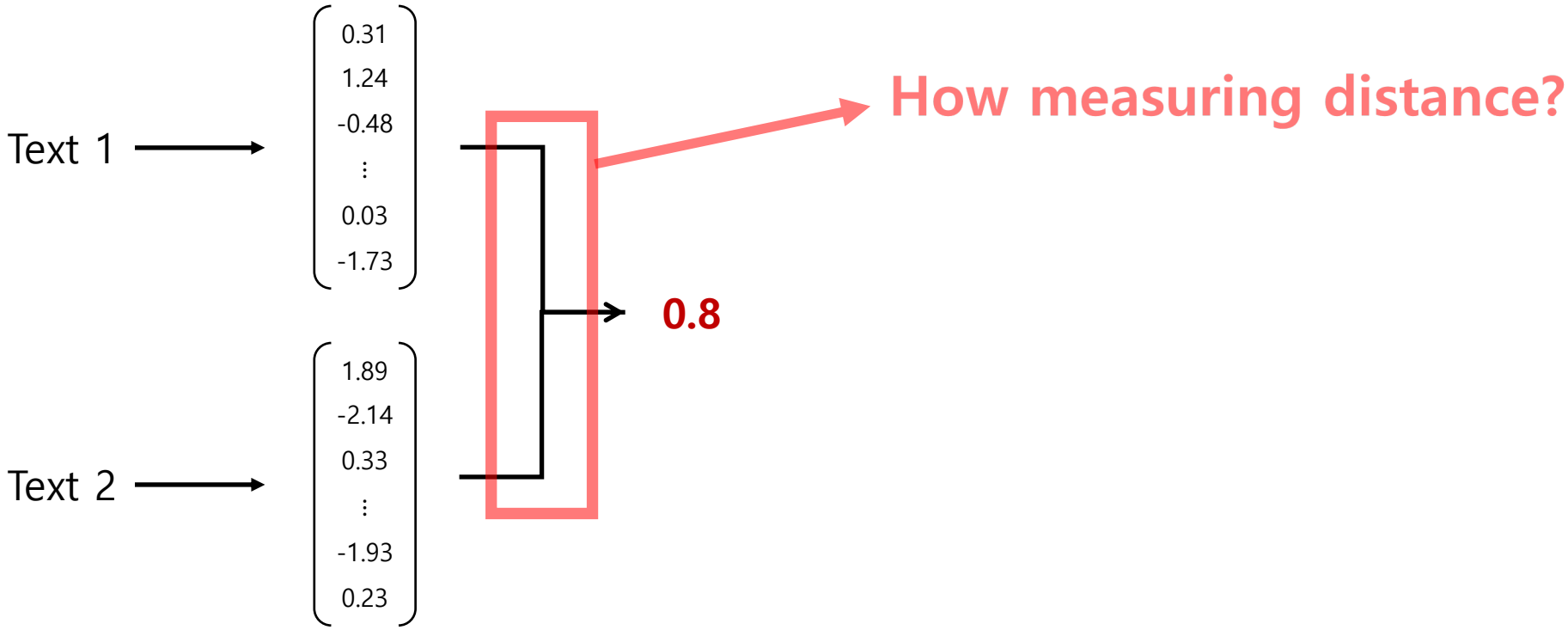


Similarity 측정

Measuring Similarity by Distance

주어진 문장(or 단어)를 고정된 길이의 벡터로 만든 후

두 벡터 사이의 거리(Distance)를 측정

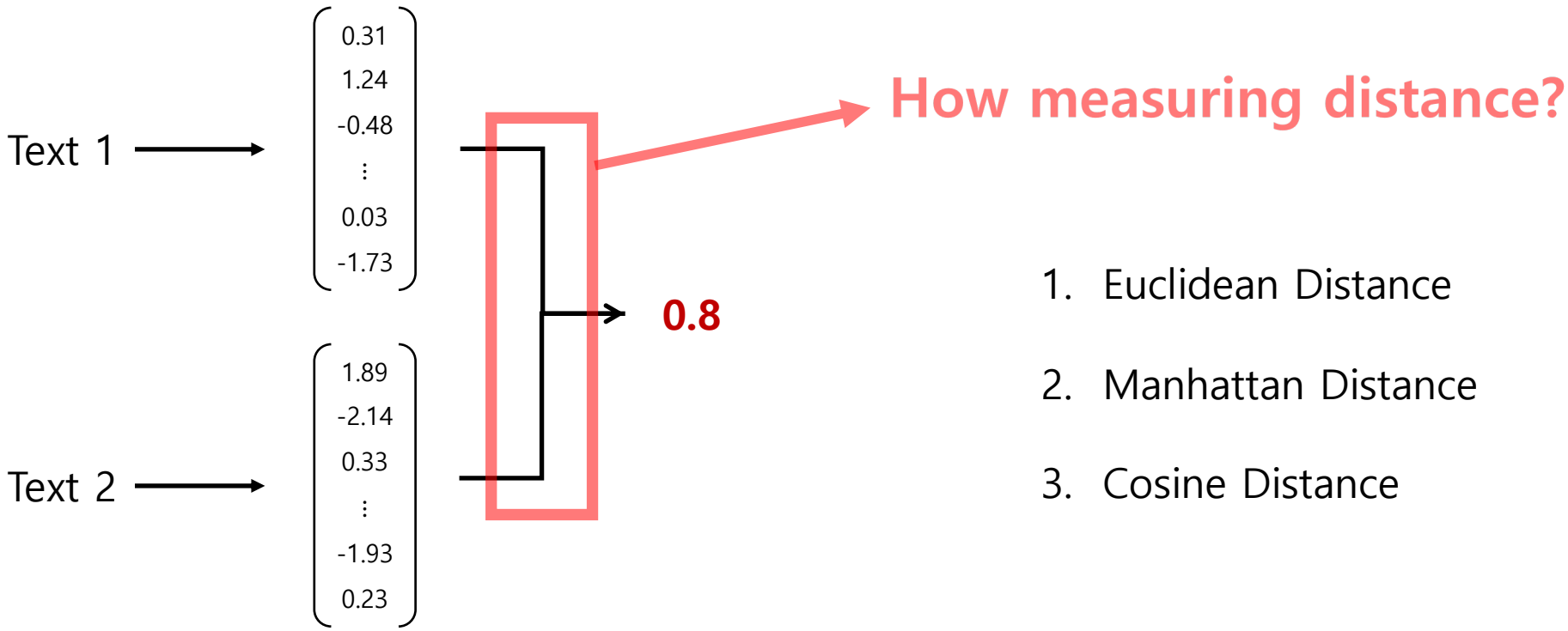


Similarity 측정

Measuring Similarity by Distance

주어진 문장(or 단어)를 고정된 길이의 벡터로 만든 후

두 벡터 사이의 거리(Distance)를 측정



1. Euclidean Distance
2. Manhattan Distance
3. Cosine Distance

Similarity 측정

Distance란? (In Mathematics)

Distance란 어떤 한 쌍의 값에 대해서 특정 조건을 만족시키면서
하나의 실수 값을 만드는 함수이다.

Similarity 측정

Distance란? (In Mathematics)

Distance란 어떤 한 쌍의 값에 대해서 특정 조건을 만족시키면서
하나의 실수 값을 만드는 함수이다.

Distance between x and y = $d(x, y)$

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \Leftrightarrow x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \geq d(x, y) + d(y, z)$

Similarity 측정

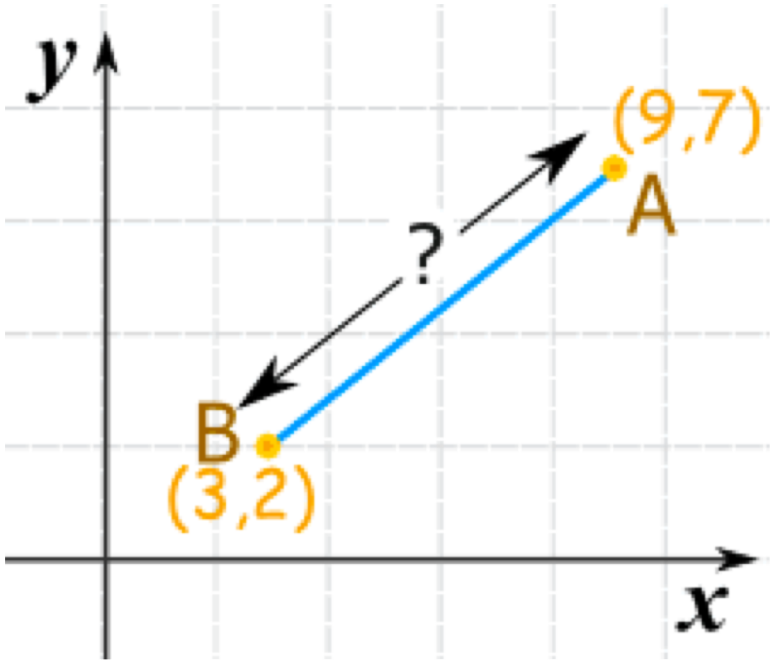
Euclidean Distance (L2 Distance)

흔히 거리라 부르는 개념, 일반적으로 두 점 사이를 자로 잴을 때 구해지는 값.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

for $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$



Similarity 측정

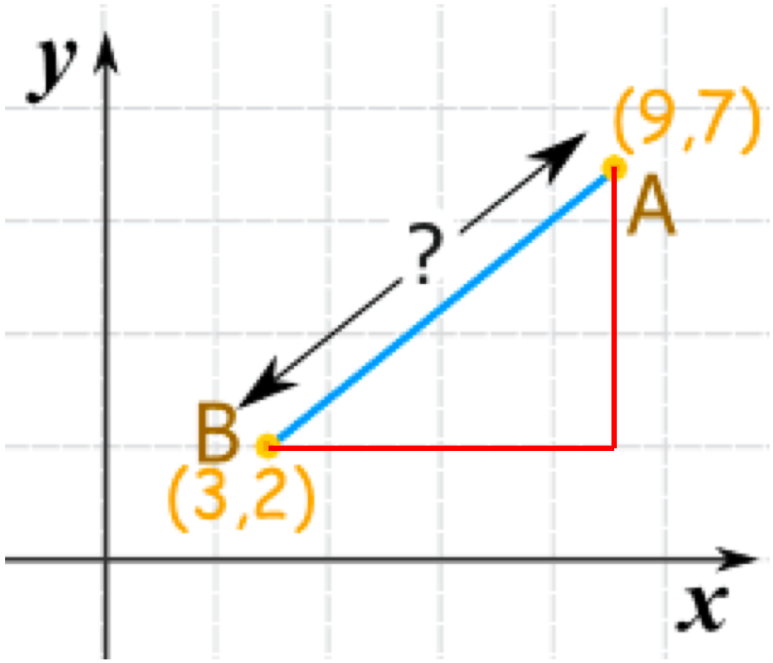
Euclidean Distance (L2 Distance)

흔히 거리라 부르는 개념, 일반적으로 두 점 사이를 자로 잴을 때 구해지는 값.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



Similarity 측정

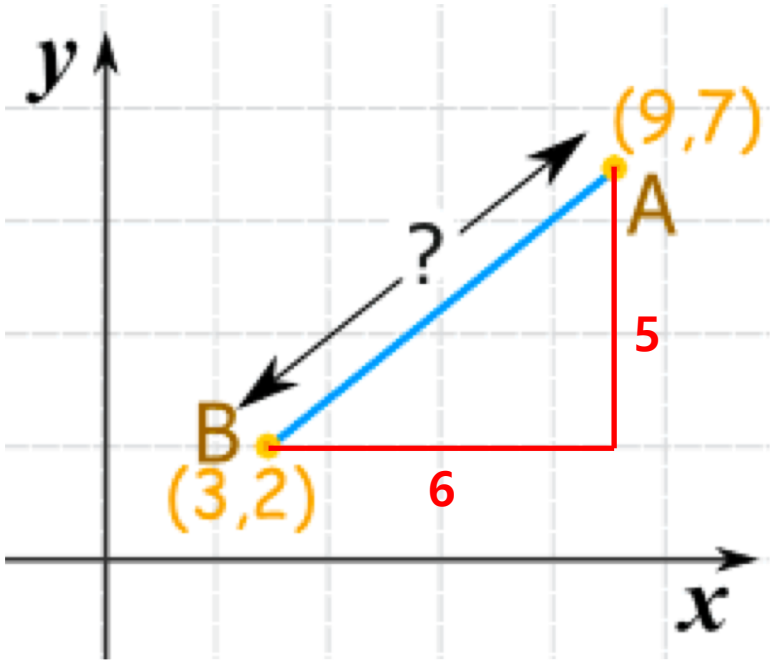
Euclidean Distance (L2 Distance)

흔히 거리라 부르는 개념, 일반적으로 두 점 사이를 자로 잴을 때 구해지는 값.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

for $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$



Similarity 측정

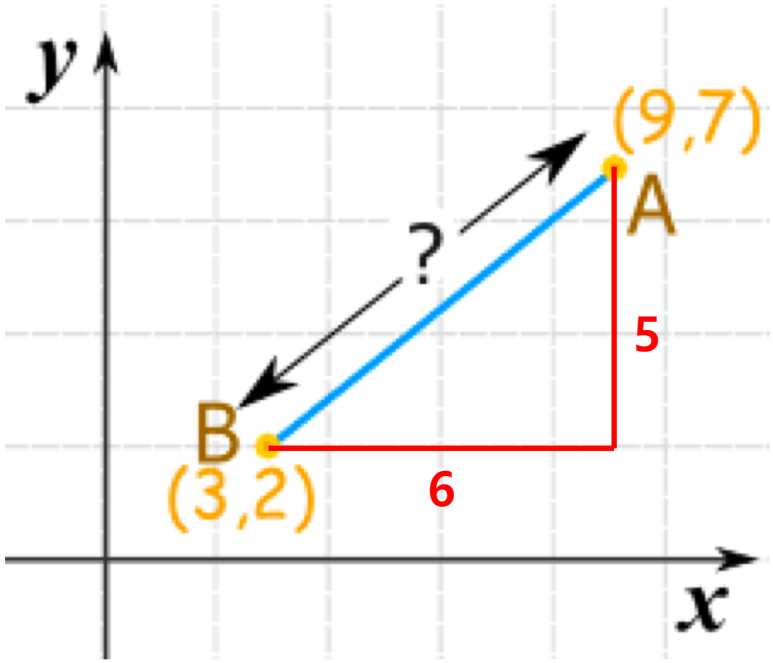
Euclidean Distance (L2 Distance)

흔히 거리라 부르는 개념, 일반적으로 두 점 사이를 자로 잴을 때 구해지는 값.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

for $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$



$$\sqrt{6^2 + 5^2} = \sqrt{61}$$

Similarity 측정

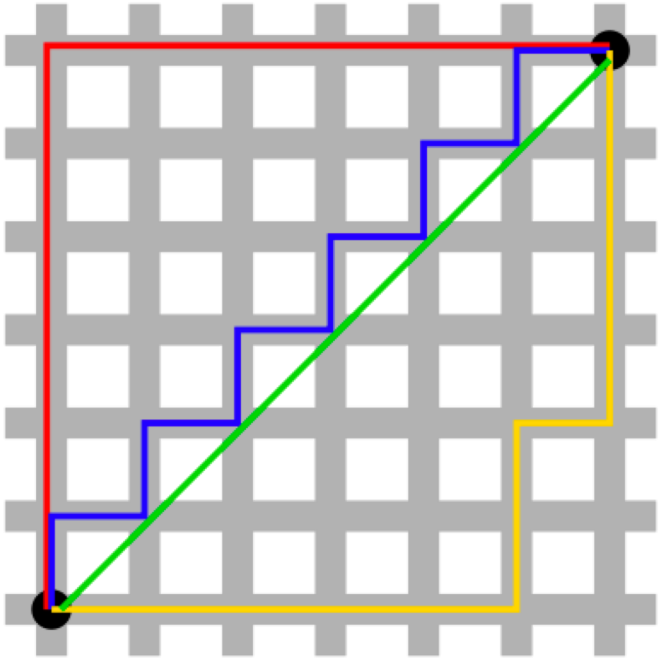
Manhattan Distance (L1 Distance)

비유클리드 기하학에서 자주 사용되는 거리.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_n - y_n)}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



Similarity 측정

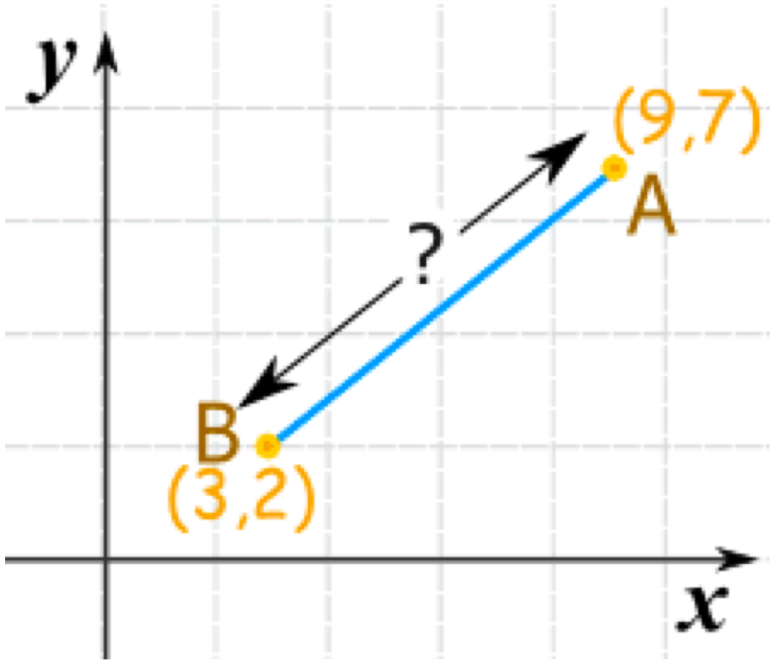
Manhattan Distance (L1 Distance)

비유클리드 기하학에서 자주 사용되는 거리.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_n - y_n)}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



Similarity 측정

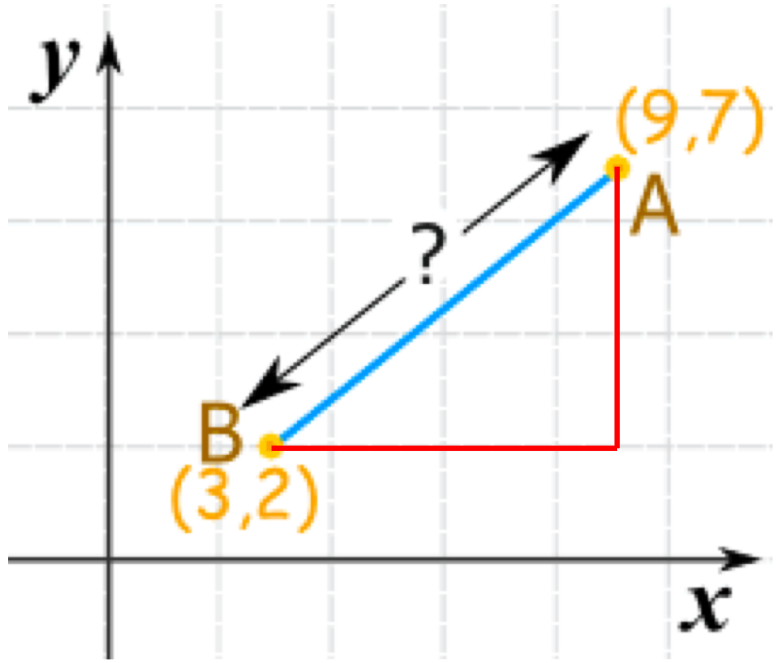
Manhattan Distance (L1 Distance)

비유클리드 기하학에서 자주 사용되는 거리.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_n - y_n)}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



Similarity 측정

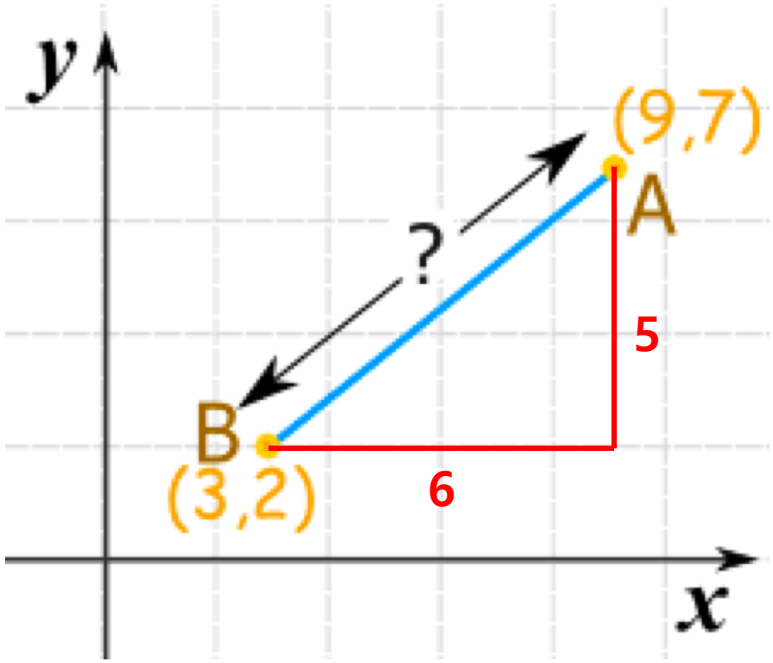
Manhattan Distance (L1 Distance)

비유클리드 기하학에서 자주 사용되는 거리.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_n - y_n)}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



Similarity 측정

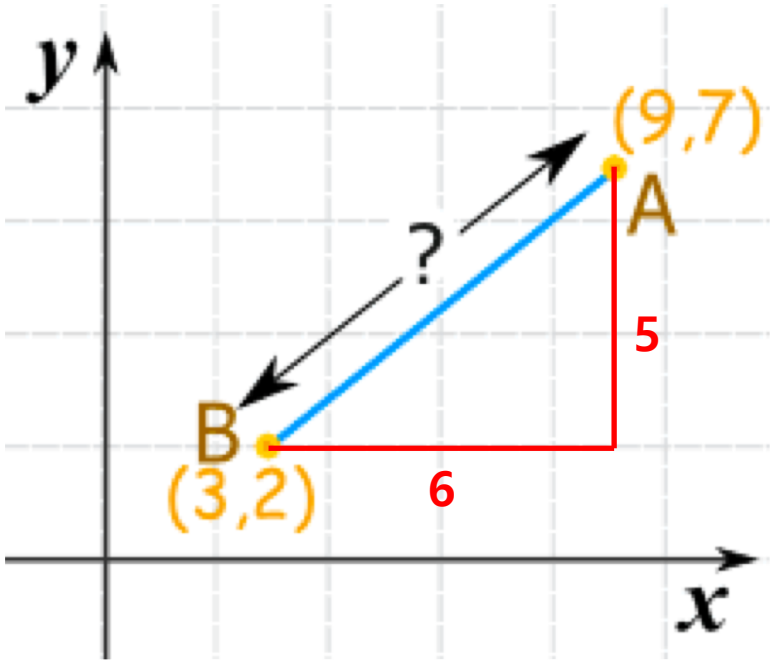
Manhattan Distance (L1 Distance)

비유클리드 기하학에서 자주 사용되는 거리.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_n - y_n)}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



6 + 5 = 11

Similarity 측정

Cosine Distance

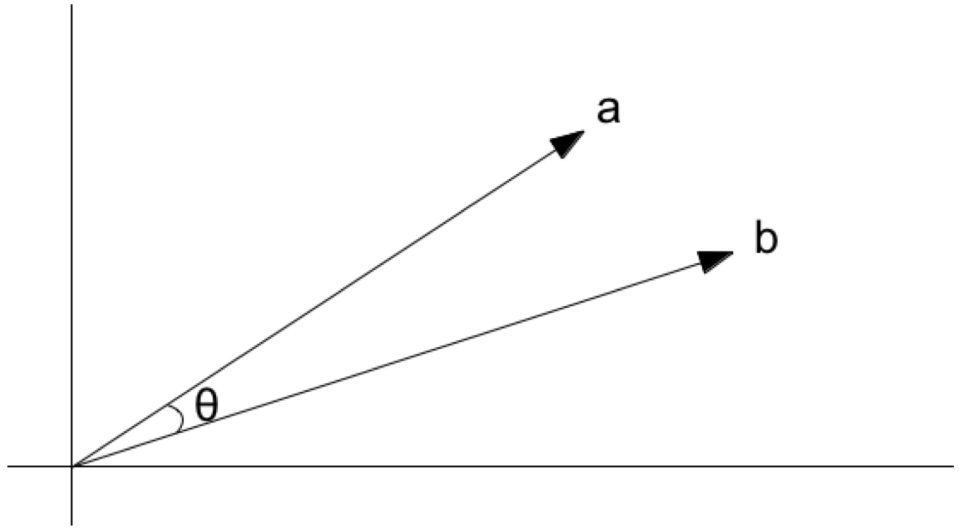
두 벡터 사이의 각도를 이용해 Distance를 구함.

$$d(\mathbf{x}, \mathbf{y}) = \cos(\theta)$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x} \cdot \mathbf{y}|}$$

$$= \frac{\sum_{i=1}^{n} \boldsymbol{x_i y_i}}{\sqrt{\sum_{i}^{n} \boldsymbol{x_i^2}} \sqrt{\sum_{i}^{n} \boldsymbol{y_i^2}}}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n)$$



Similarity 측정

Cosine Distance

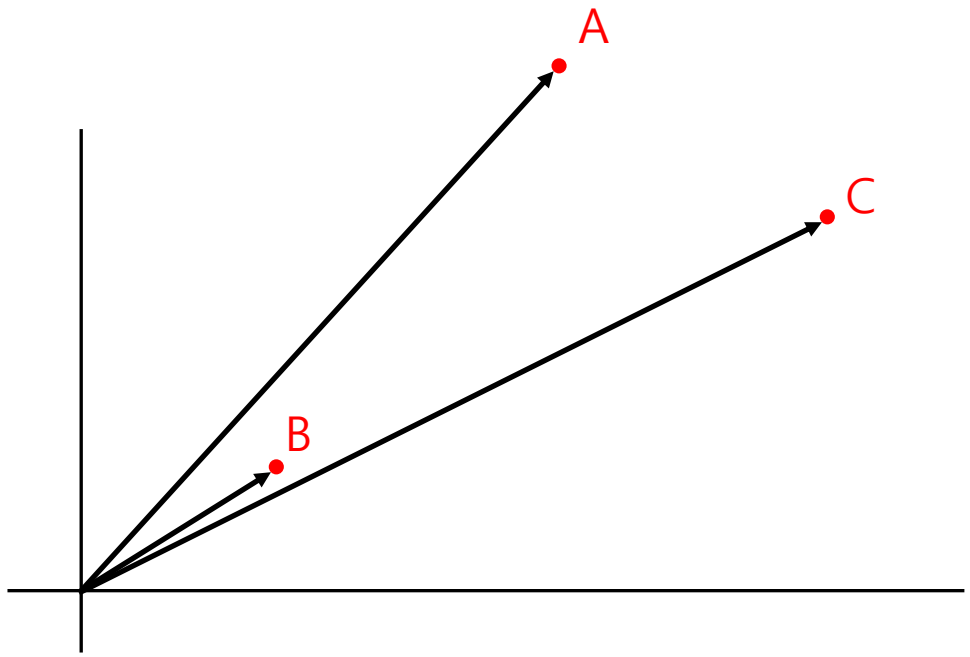
두 벡터 사이의 각도를 이용해 Distance를 구함.

$$d(\mathbf{x}, \mathbf{y}) = \cos(\theta)$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x} \cdot \mathbf{y}|}$$

$$= \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$



Similarity 측정

Cosine Distance

두 벡터 사이의 각도를 이용해 Distance를 구함.

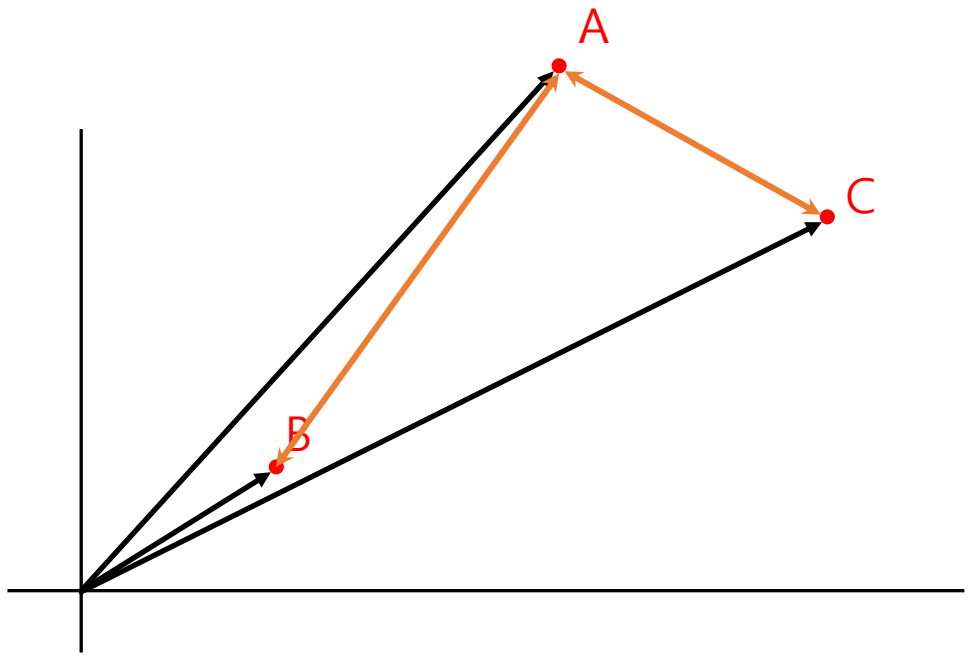
$$d(\mathbf{x}, \mathbf{y}) = \cos(\theta)$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x} \cdot \mathbf{y}|}$$

$$= \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$

Euclidean Distance
d(A,B) < d(A,C)



Similarity 측정

Cosine Distance

두 벡터 사이의 각도를 이용해 Distance를 구함.

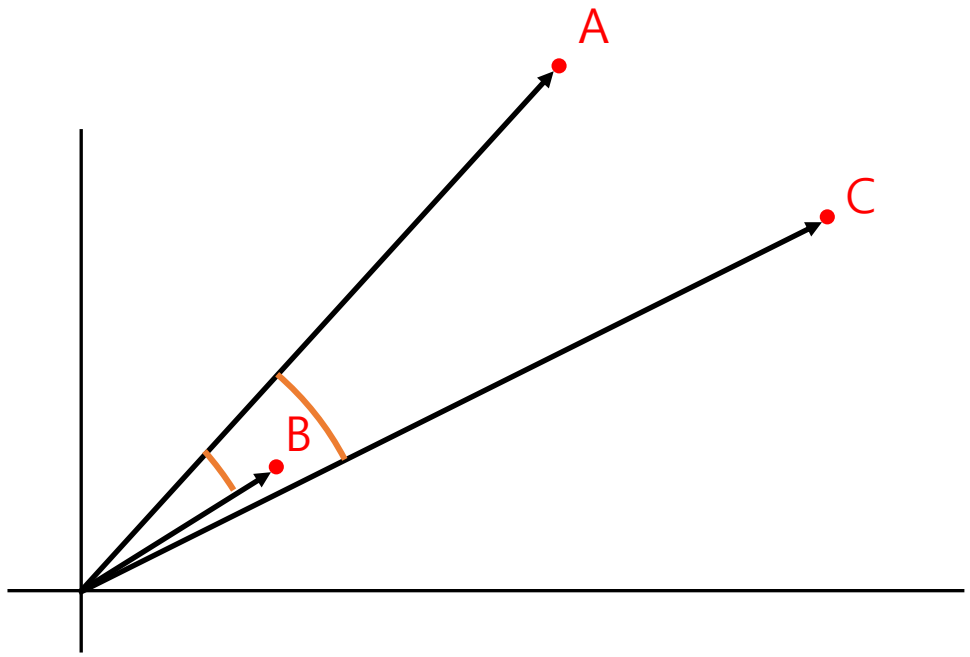
$$d(\mathbf{x}, \mathbf{y}) = \cos(\theta)$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x} \cdot \mathbf{y}|}$$

$$= \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

$$\text{for } \mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$$

Cosine Distance
d(A,B) > d(A,C)



Similarity 측정

Cosine Distance

두 벡터 사이의 각도를 이용해 Distance를 구함.

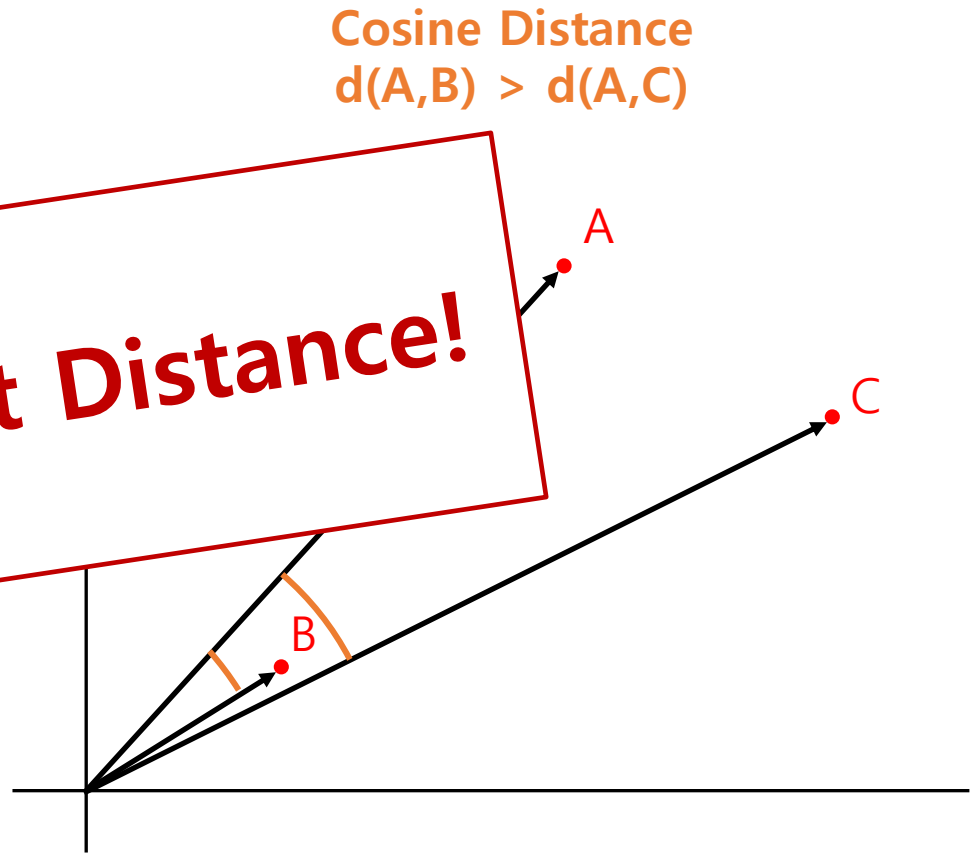
$$d(\mathbf{x}, \mathbf{y}) = \cos(\theta)$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x} \cdot \mathbf{y}|}$$

$$= \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

for $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$

But Cosine is not Distance!



Similarity 측정

Cosine Distance

두 벡터 사이의 각도를 이용해 Distance를 구함.

~~1. $d(x,y) \geq 0$~~ => Cosine 값 음수 될 수 있음

~~2. $d(x,y) = 0 \Leftrightarrow x = y$~~ => $x = y$ 일 때 cosine 값은 1

3. $d(x,y) = d(y,x)$

~~4. $d(x,z) \geq d(x,y) + d(y,z)$~~ =>성립하지 않음

Similarity 측정

Cosine Similarity

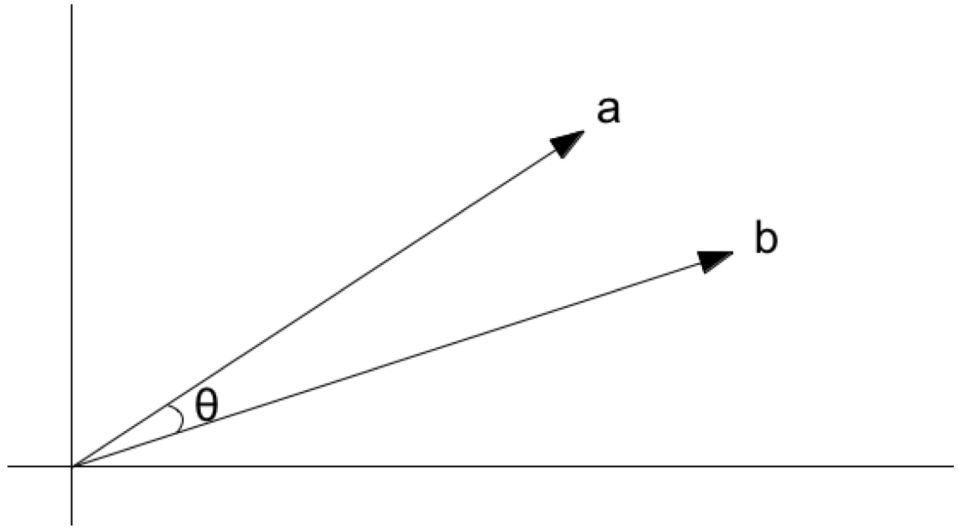
두 벡터 사이의 각도를 이용해 Similarity를 구함.

$$similarity(\mathbf{x}, \mathbf{y}) = \cos(\theta)$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x} \cdot \mathbf{y}|}$$

$$= \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

for $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$



Natural Language Inference

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에 의미적으로 수반(entail)하는지 결정하는 문제.

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에 의미적으로 수반(entail)하는지 결정하는 문제.

Entailment	h is definitely true given p
Neutral	h might be true given p
Contradiction	h is definitely **not** true given p

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에
의미적으로 수반(entail)하는지 결정하는 문제.

Promise: A man inspects the uniform of a figure in some East Asian country.

Hypothesis: The man is sleeping

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에
의미적으로 수반(entail)하는지 결정하는 문제.

Promise: A man inspects the uniform of a figure in some East Asian country.

Hypothesis: The man is sleeping

Contradiction

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에 의미적으로 수반(entail)하는지 결정하는 문제.

Promise: A soccer game with multiple males playing.

Hypothesis: Some men are playing a sport.

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에 의미적으로 수반(entail)하는지 결정하는 문제.

Promise: A soccer game with multiple males playing.

Hypothesis: Some men are playing a sport.

Entailment

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에
의미적으로 수반(entail)하는지 결정하는 문제.

Promise: An older and younger man smiling.

Hypothesis: Two men are smiling and laughing at the cats playing on the floor.

What is Natural Language Inference

하나의 문장(premise)이 주어졌을 때 해당 문장이 다른 문장(hypothesis)에
의미적으로 수반(entail)하는지 결정하는 문제.

Promise: An older and younger man smiling.

Hypothesis: Two men are smiling and laughing at the cats playing on the floor.

Neutral

Natural Language Inference Dataset

Natural Language Inference Dataset

Semantic Textual Similarity

- SNLI
- MultiNLI

Natural Language Inference Dataset

Semantic Textual Similarity

- SNLI
- MultiNLI

SNLI

- Stanford Natural Language Inference Corpus
- 57만개의 Human-written 영어 문장 쌍
- 3 Classes
  - Entailment
  - Contradiction
  - Neutral

Natural Language Inference Dataset

Semantic Textual Similarity

- SNLI
- MultiNLI

MultiNLI

- Multi-Genre Natural Language Inference Corpus
- 43만개의 Crowd-Sourced collection 문장 쌍
- SNLI를 기반으로 만들어짐
  - SNLI와 다른 점은 다양한 장르의 Text로 구성
- 3 Classes
  - Entailment
  - Contradiction
  - Neutral

Similarity Algorithm

MaLSTM

Siamese Recurrent Architectures for Learning Sentence Similarity

Jonas Mueller
Computer Science
& Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Aditya Thyagarajan
Department of Computer Science
and Engineering
M. S. Ramaiah Institute of Technology

Abstract

We present a siamese adaptation of the Long Short-Term Memory (LSTM) network for labeled data comprised of pairs of variable-length sequences. Our model is applied to assess semantic similarity between sentences, where we exceed state of the art, outperforming carefully handcrafted features and recently proposed neural network systems of greater complexity. For these applications, we provide word-embedding vectors supplemented with synonymic information to the LSTMs, which use a fixed size vector to encode the underlying meaning expressed in a sentence (irrespective of the particular wording/syntax). By restricting subsequent operations to rely on a simple Manhattan metric, we compel the sentence representations learned by our model to form a highly structured space whose geometry reflects complex semantic relationships. Our results are the latest in a line of findings that showcase LSTMs as powerful language models capable of tasks requiring intricate understanding.

Introduction

Text understanding and information retrieval are important tasks which may be greatly enhanced by modeling the underlying semantic similarity between sentences/phrases. In particular, a good model should not be susceptible to variations of wording/syntax used to express the same idea. Learning such a semantic textual similarity metric has thus generated a great deal of research interest (Marelli et al. 2014). However, this remains a hard problem, because labeled data is scarce, sentences have both variable length and complex structure, and bag-of-words/tf-IDF models, while dominant in natural language processing (NLP), are limited in this context by their inherent term-specificity (c.f. Mihalcea, Corley, and Strapparava 2006).

As an alternative to these ideas, Mikolov et al. (2013) and others have demonstrated the effectiveness of neural word representations for analogies and other NLP tasks. Recently, interests have shifted toward extensions of these ideas beyond the individual word-level to larger bodies of text such as sentences, where a mapping is learned to represent each sentence as a fixed-length vector (Kiros et al. 2015; Tai, Socher, and Manning 2015; Le and Mikolov 2014).

Naturally suited for variable-length inputs like sentences, recurrent neural networks (RNN), especially the Long Short-Term Memory model of Hochreiter and Schmidhuber (1997), have been particularly successful in this setting for tasks such as text classification (Graves 2012) and language translation (Sutskever, Vinyals, and Le 2014). RNNs adapt standard feedforward neural networks for sequence data $(x_1, \ldots, x_T)$, where at each $t \in \{1, \ldots, T\}$, updates to a hidden-state vector h_t are performed via

$$h_t = \text{sigmoid}\left(Wx_t + Uh_{t-1}\right) \tag{1}$$

While Siegelmann and Sontag (1995) have shown that the basic RNN is Turing-complete, optimization of the weight-matrices is difficult because its backpropagated gradients become vanishingly small over long sequences. Practically, the LSTM is superior to basic RNNs for learning long range dependencies through its use of memory cell units that can store/access information across lengthy input sequences. Like RNNs, the LSTM sequentially updates a hidden-state representation, but these steps also rely on a memory cell containing four components (which are real-valued vectors): a memory state c_t, an output gate o_t that determines how the memory state affects other units, as well as an input (and forget) gate i_t (and f_t) that controls what gets stored in (and omitted from) memory based on each new input and the current state. Below are the updates performed at each $t \in \{1, \ldots, T\}$ in an LSTM parameterized by weight matrices $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and bias-vectors b_i, b_f, b_c, b_o:

$$i_t = \text{sigmoid}\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{2}$$

$$f_t = \text{sigmoid}\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3}$$

$$\widetilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{4}$$

$$c_t = i_t \odot \widetilde{c}_t + f_t \odot c_{t-1} \tag{5}$$

$$o_t = \text{sigmoid}\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

A more thorough exposition of the LSTM model and its variants is provided by Graves (2012) and Greff et al. (2015).

Although the success of LSTM language models eludes current theoretical understanding, Sutskever, Vinyals, and Le (2014) empirically validate the intuition that an effectively trained network maps each sentence onto a fixed-length vector which encodes the underlying meaning expressed in the text. Recent works have proposed many

MaLSTM

- Jonas et al. (2016, AAAI)
- Vector 간의 Similarity 측정을 위해 Manhattan Distance 사용
- Kaggle의 Competition에서 높은 Score로 유명해짐

Siamese Recurrent Architectures for Learning Sentence Similarity

Jonas Mueller
Computer Science
& Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Aditya Thyagarajan
Department of Computer Science
and Engineering
M. S. Ramaiah Institute of Technology

Abstract

We present a siamese adaptation of the Long Short-Term Memory (LSTM) network for labeled data comprised of pairs of variable-length sequences. Our model is applied to assess semantic similarity between sentences, where we exceed state of the art, outperforming carefully handcrafted features and recently proposed neural network systems of greater complexity. For these applications, we provide word-embedding vectors supplemented with synonymic information to the LSTMs, which use a fixed size vector to encode the underlying meaning expressed in a sentence (irrespective of the particular wording/syntax). By restricting subsequent operations to rely on a simple Manhattan metric, we compel the sentence representations learned by our model to form a highly structured space whose geometry reflects complex semantic relationships. Our results are the latest in a line of findings that showcase LSTMs as powerful language models capable of tasks requiring intricate understanding.

Introduction

Text understanding and information retrieval are important tasks which may be greatly enhanced by modeling the underlying semantic similarity between sentences/phrases. In particular, a good model should not be susceptible to variations of wording/syntax used to express the same idea. Learning such a semantic textual similarity metric has thus generated a great deal of research interest (Marelli et al. 2014). However, this remains a hard problem, because labeled data is scarce, sentences have both variable length and complex structure, and bag-of-words/tf-IDF models, while dominant in natural language processing (NLP), are limited in this context by their inherent term-specificity (c.f. Mihalcea, Corley, and Strapparava 2006).

As an alternative to these ideas, Mikolov et al. (2013) and others have demonstrated the effectiveness of neural word representations for analogies and other NLP tasks. Recently, interests have shifted toward extensions of these ideas beyond the individual word-level to larger bodies of text such as sentences, where a mapping is learned to represent each sentence as a fixed-length vector (Kiros et al. 2015; Tai, Socher, and Manning 2015; Le and Mikolov 2014).

Naturally suited for variable-length inputs like sentences, recurrent neural networks (RNN), especially the Long Short-Term Memory model of Hochreiter and Schmidhuber (1997), have been particularly successful in this setting for tasks such as text classification (Graves 2012) and language translation (Sutskever, Vinyals, and Le 2014). RNNs adapt standard feedforward neural networks for sequence data $(x_1, \ldots, x_T)$, where at each $t \in \{1, \ldots, T\}$, updates to a hidden-state vector h_t are performed via

$$h_t = \text{sigmoid}\left(Wx_t + Uh_{t-1}\right) \quad (1)$$

While Siegelmann and Sontag (1995) have shown that the basic RNN is Turing-complete, optimization of the weight-matrices is difficult because its backpropagated gradients become vanishingly small over long sequences. Practically, the LSTM is superior to basic RNNs for learning long range dependencies through its use of memory cell units that can store/access information across lengthy input sequences. Like RNNs, the LSTM sequentially updates a hidden-state representation, but these steps also rely on a memory cell containing four components (which are real-valued vectors): a memory state c_t, an output gate o_t that determines how the memory state affects other units, as well as an input (and forget) gate i_t (and f_t) that controls what gets stored in (and omitted from) memory based on each new input and the current state. Below are the updates performed at each $t \in \{1, \ldots, T\}$ in an LSTM parameterized by weight matrices $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and bias-vectors b_i, b_f, b_c, b_o:

$$i_t = \text{sigmoid}\left(W_i x_t + U_i h_{t-1} + b_i\right) \quad (2)$$

$$f_t = \text{sigmoid}\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (3)$$

$$\widetilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \quad (4)$$

$$c_t = i_t \odot \widetilde{c}_t + f_t \odot c_{t-1} \quad (5)$$

$$o_t = \text{sigmoid}\left(W_o x_t + U_o h_{t-1} + b_o\right) \quad (6)$$

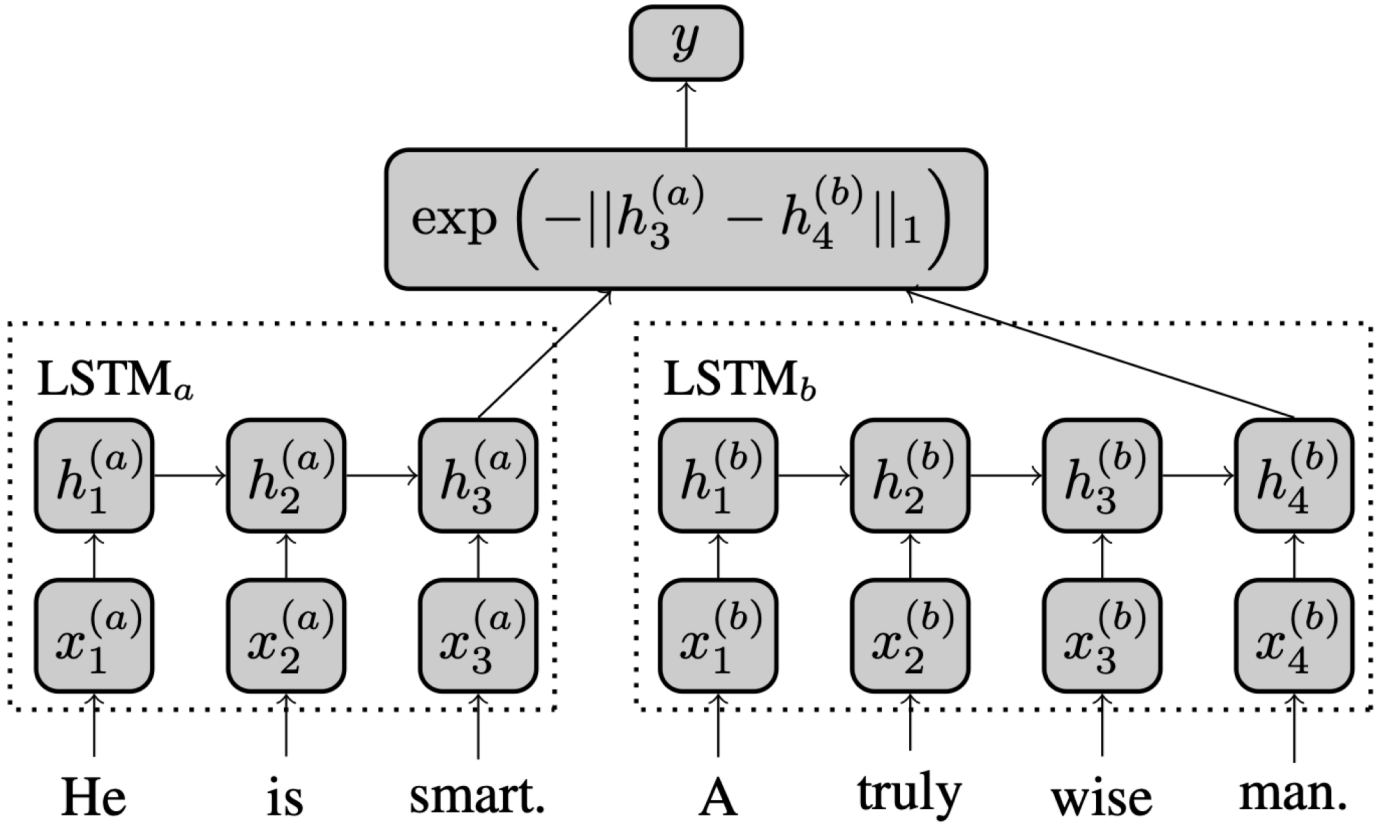
$$h_t = o_t \odot \tanh(c_t) \quad (7)$$

A more thorough exposition of the LSTM model and its variants is provided by Graves (2012) and Greff et al. (2015).

Although the success of LSTM language models eludes current theoretical understanding, Sutskever, Vinyals, and Le (2014) empirically validate the intuition that an effectively trained network maps each sentence onto a fixed-length vector which encodes the underlying meaning expressed in the text. Recent works have proposed many

MaLSTM

Model



Tree-Based Model

Tree-Based Model

- Decision Tree (CART)
- Bagging / Boosting
- Random Forest
- XG Boost