Words in Natural Language Processing

건국대학교 컴퓨터공학부 /
KAIST 전산학부 (겸직)

김학수

Word Frequency

- **Observation:** Some words are more common than others.
- **Statistics:** Most large collections of unstructured text documents have similar statistical characteristics. These statistics:
  - influence the effectiveness and efficiency of data structures used to index documents
  - many retrieval models rely on them

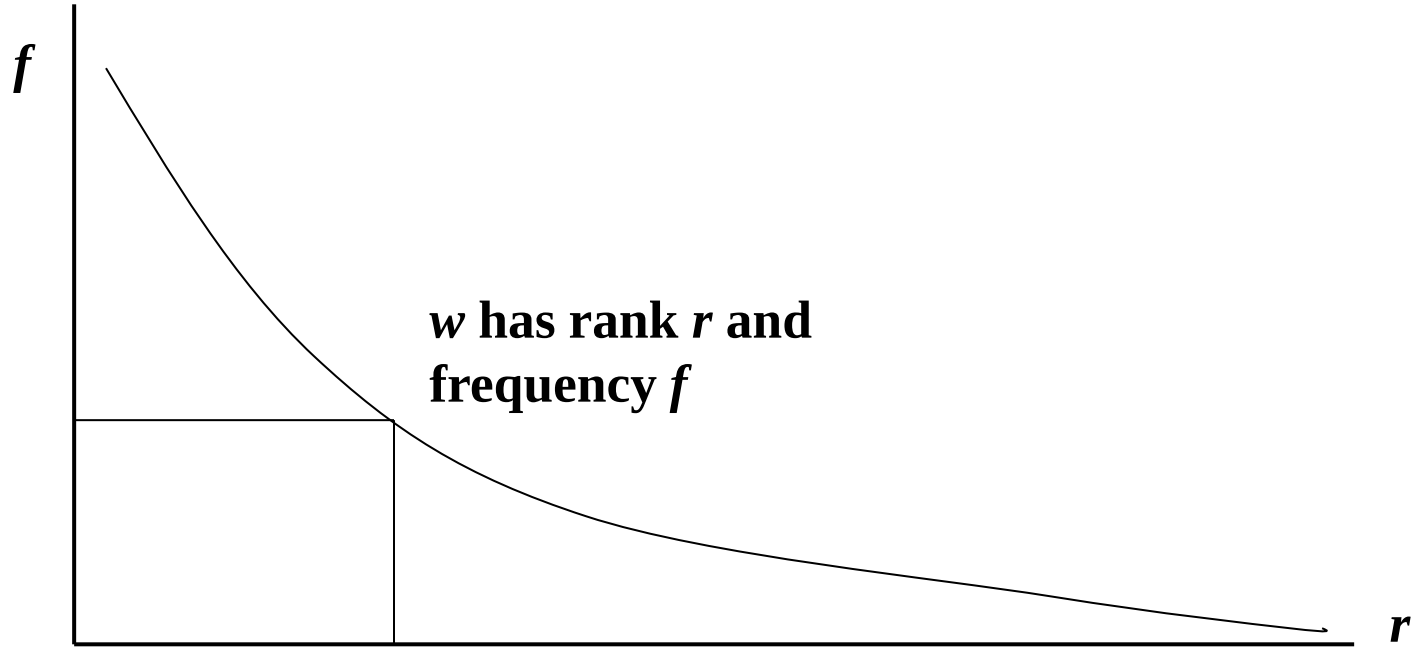
Word Frequency

- **Example**
  - The following example is taken from:
  - Jamie Callan, *Characteristics of Text*, 1997
  - Sample of 19 million words
  - The next slide shows the 50 commonest words in rank order (*r*), with their frequency (*f*).

	f		*f*		*f*
the	**1,130,021**	**from**	**96,900**	**or**	**54,958**
of	**547,311**	**he**	**94,585**	**about**	**53,713**
to	**516,635**	**million**	**93,515**	**market**	**52,110**
a	**464,736**	**year**	**90,104**	**they**	**51,359**
in	**390,819**	**its**	**86,774**	**this**	**50,933**
and	**387,703**	**be**	**85,588**	**would**	**50,828**
that	**204,351**	**was**	**83,398**	**you**	**49,281**
for	**199,340**	**company**	**83,070**	**which**	**48,273**
is	**152,483**	**an**	**76,974**	**bank**	**47,940**
said	**148,302**	**has**	**74,405**	**stock**	**47,401**
it	**134,323**	**are**	**74,097**	**trade**	**47,310**
on	**121,173**	**have**	**73,132**	**his**	**47,116**
by	**118,863**	**but**	**71,887**	**more**	**46,244**
as	**109,135**	**will**	**71,494**	**who**	**42,142**
at	**101,779**	**say**	**66,807**	**one**	**41,635**
mr	**101,679**	**new**	**64,456**	**their**	**40,910**
with	**101,210**	**share**	**63,925**		

Rank Frequency Distribution

- For all the words in a collection of documents, for each word *w*
  - *f* is the frequency that *w* appears
  - *r* is rank of *w* in order of frequency. (The most commonly occurring word has rank 1, etc.)



Rank Frequency Example

- The next slide shows the words in Callan's data normalized.
- In this example:
  - *r* is the rank of word *w* in the sample.
  - *f* is the frequency of word *w* in the sample.
  - *n* is the total number of word occurrences in the sample.

*rf**1000/*n*		*rf**1000/*n*		*rf**1000/*n*	
the	59	from	92	or	101
of	58	he	95	about	102
to	82	million	98	market	101
a	98	year	100	they	103
in	103	its	100	this	105
and	122	be	104	would	107
that	75	was	105	you	106
for	84	company	109	which	107
is	72	an	105	bank	109
said	78	has	106	stock	110
it	78	are	109	trade	112
on	77	have	112	his	114
by	81	but	114	more	114
as	80	will	117	who	106
at	80	say	113	one	107
mr	86	new	112	their	108
with	91	share	114		

Zipf's Law

- If the words, *w*, in a collection are ranked, *r*, by their frequency, *f*, they roughly fit the relation:
  - $r * f = c$
- Different collections have different constants *c*.
- In English text, *c* tends to be about *n* / 10, where *n* is the number of word occurrences in the collection, 19 million in the example.

Methods that Build on Zipf's Law

- **Stop lists:** Ignore the most frequent words (upper cut-off). *Used by almost all systems.*
- **Significant words:** Ignore the most frequent and least frequent words (upper and lower cut-off). *Rarely used.*
- **Term weighting:** Give differing weights to terms based on their frequency, with most frequent words weighed less. *Used by almost all ranking methods.*

KU KONKUK UNIVERSITY

KAIST

Text Representation

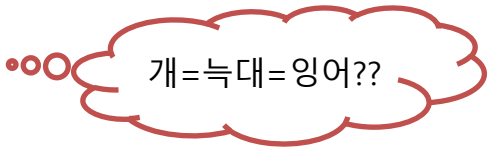
건국대학교 컴퓨터공학부 /
KAIST 전산학부 (겸직)

김학수

자연어처리연구실

이산 표현 (Discrete Representation)

- 원-핫 인코딩 (one-hot encoding)
  - 단어를 벡터로 표현하는 가장 간단한 방법
  - 단어 사전(dictionary)을 구성하고 해당 단어를 1로, 그 밖의 단어는 0으로 표현
    - 사전: 개, 고양이, 늑대, 사자, 송어, 잉어
    - 단어: 개 [1, 0, 0, 0, 0, 0], 고양이 [0, 1, 0, 0, 0, 0], 늑대 [0, 0, 1, 0, 0, 0]
- 원-핫 인코딩의 한계
  - 대용량 메모리 필요
    - 사전의 크기: 음성 인식 20K, 구문 분석: 50K, 대용량 사전: 500K, 구글 1T 말뭉치: 13M
  - 유사성 비교 불가능
    - Sim(개, 늑대) = AND([1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0]) = 0
    - Sim(개, 잉어) = AND([1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1]) = 0



분산 표현 (Distributed Representation)

- 분산 표현
  - 단어를 문맥에 기반하여 표현하는 방법
  - 비슷한 문맥에서 등장하는 단어는 비슷한 의미를 가질 것이라는 가정에서 출발

신종 **코로나** 바이러스 집단 감염증이 일상생활을 통해 지속 확산되고 있다.

부산 수영구 댄스 동호회발 신종 **코로나** 바이러스 집단 감염이 목욕탕으로 퍼지는 등 확산되고 있다.

코로나? 신종, 바이러스, 집단, 감염, 확산, 일상, 생활, ..., 목욕탕

프랑스는 지금까지 북부도시 릴에서 2명의 **메르스** 바이러스 감염환자가 발생했다.

메르스? 바이러스, 감염, 프랑스, 지금, ..., 환자, 발생

Sim(코로나, 메르스) → AND(코로나, 메르스) ≠0

공기 행렬 (Co-Occurrence Matrix)

[예문] I like deep learning. I like NLP. I enjoy flying.

Window size: 1

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Sim(NLP, learning) = ?

그림 출처: Kira Radinsky 교수 강의자료

코사인 유사도 (Cosine Similarity)

- 코사인 유사도
  - 길이로 정규화된 내적을 바탕으로 두 벡터 사이의 유사도를 측정하는 척도

$$\vec{X} \cdot \vec{Y} = |\vec{X}||\vec{Y}| \cos(\theta)$$

두 벡터 요소의 구성이 비슷하면 큰 값을 가짐

$$\cos(\vec{X}, \vec{Y}) = \frac{\vec{X} \cdot \vec{Y}}{|\vec{X}||\vec{Y}|}$$

두 벡터가 직각: 0
두 벡터가 동일: 1

$$\vec{X} \cdot \vec{Y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

$$|\vec{X}| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

$$|\vec{Y}| = \sqrt{y_1^2 + y_2^2 + \cdots + y_n^2}$$

cos(NLP, learning) = 1/2 = 0.5

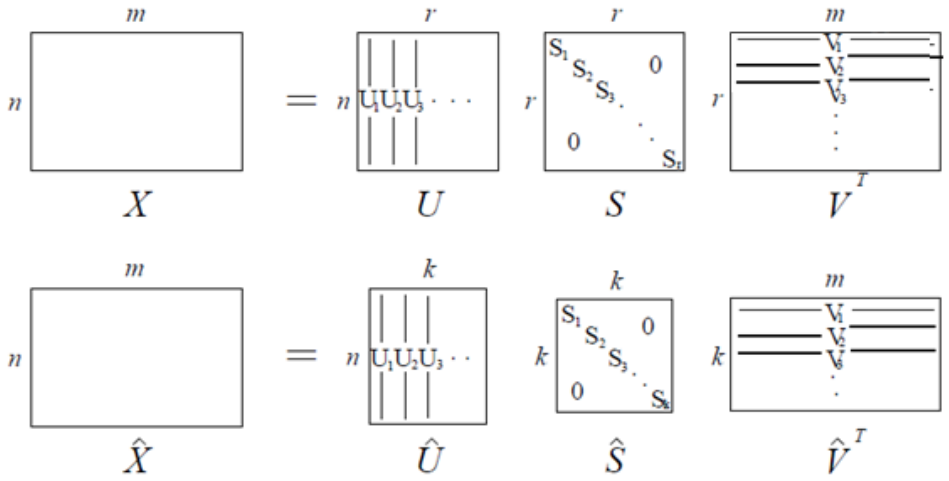
```
import torch

NLP = torch.FloatTensor([0,1,0,0,0,0,0,1])
learning = torch.FloatTensor([0,0,0,1,0,0,0,1])

print(torch.cosine_similarity(NLP, learning, dim=0))
```

```
tensor(0.5000)
```

Problems with Co-Occurrence Vectors

- 차원의 저주 (Curse of dimensionality)
  - 차원이 증가하면서 학습데이터의 수가 차원의 수보다 적어서 성능이 저하되는 현상 → 희소 데이터 문제 (sparse data problem)
- 특이값 분해 (SVD; Singular Value Decomposition)
  - 행렬을 특정한 구조로 분해하는 방식

SVD
(Singular Value Decomposition)



$\hat{X}$ is the best rank k approximation to X, in terms of least squares

그림 출처: Kira Radinsky 교수 강의자료

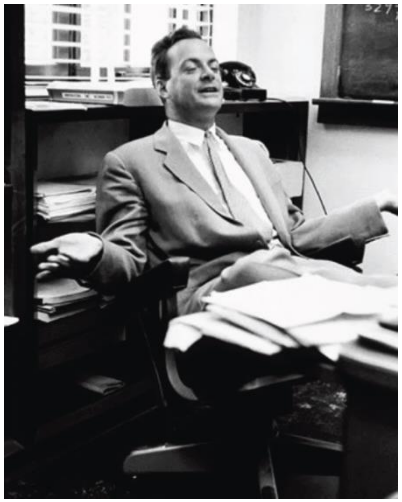
Effect of SVD

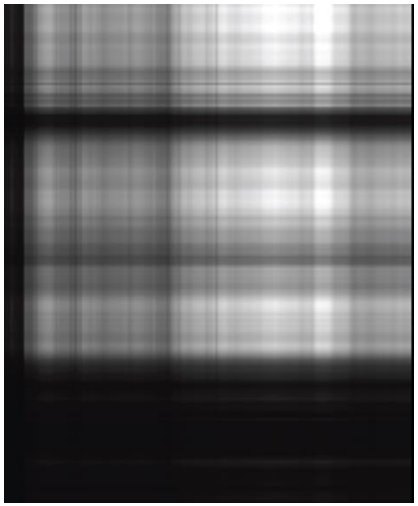
Original

1st Singular

1st & 2nd Singular

.....





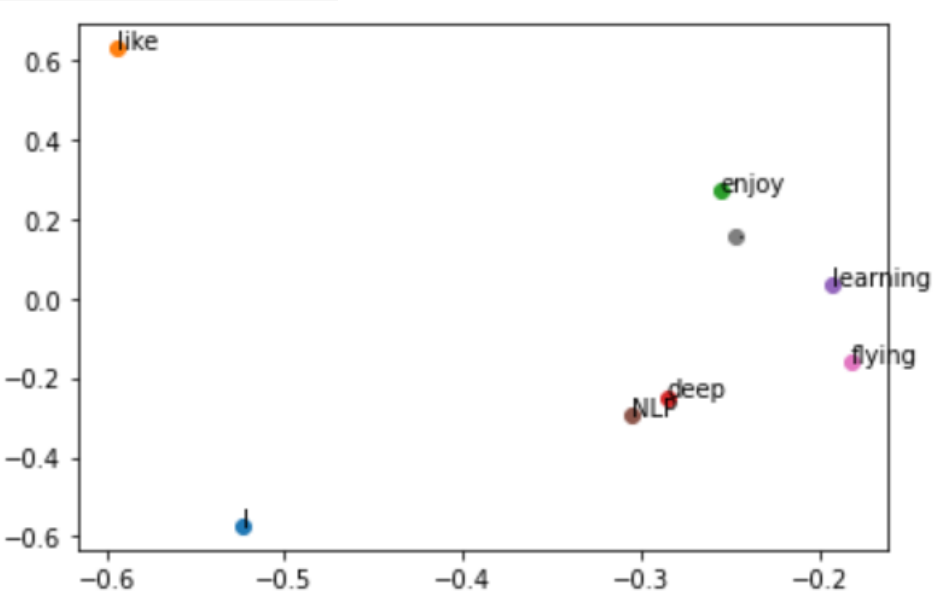
$$R_i = c_i \cdot SV_1 \qquad R_i = c_i \cdot SV_1 + b_i \cdot SV_2$$

그림 출처: Kira Radinsky 교수 강의자료

Simple SVD Word Vectors in Python

[예문] I like deep learning. I like NLP. I enjoy flying.

```
import numpy as np
import matplotlib.pyplot as plt

dic= ["I", "like", "enjoy", "deep", "learning", "NLP", "flying", "." ]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])
U, S, Vt =np.linalg.svd(X, full_matrices=False)

for i in range(len(dic)):
  plt.scatter(U[i,0], U[i,1])
  plt.text(U[i,0], U[i,1], dic[i])

plt.show()
```



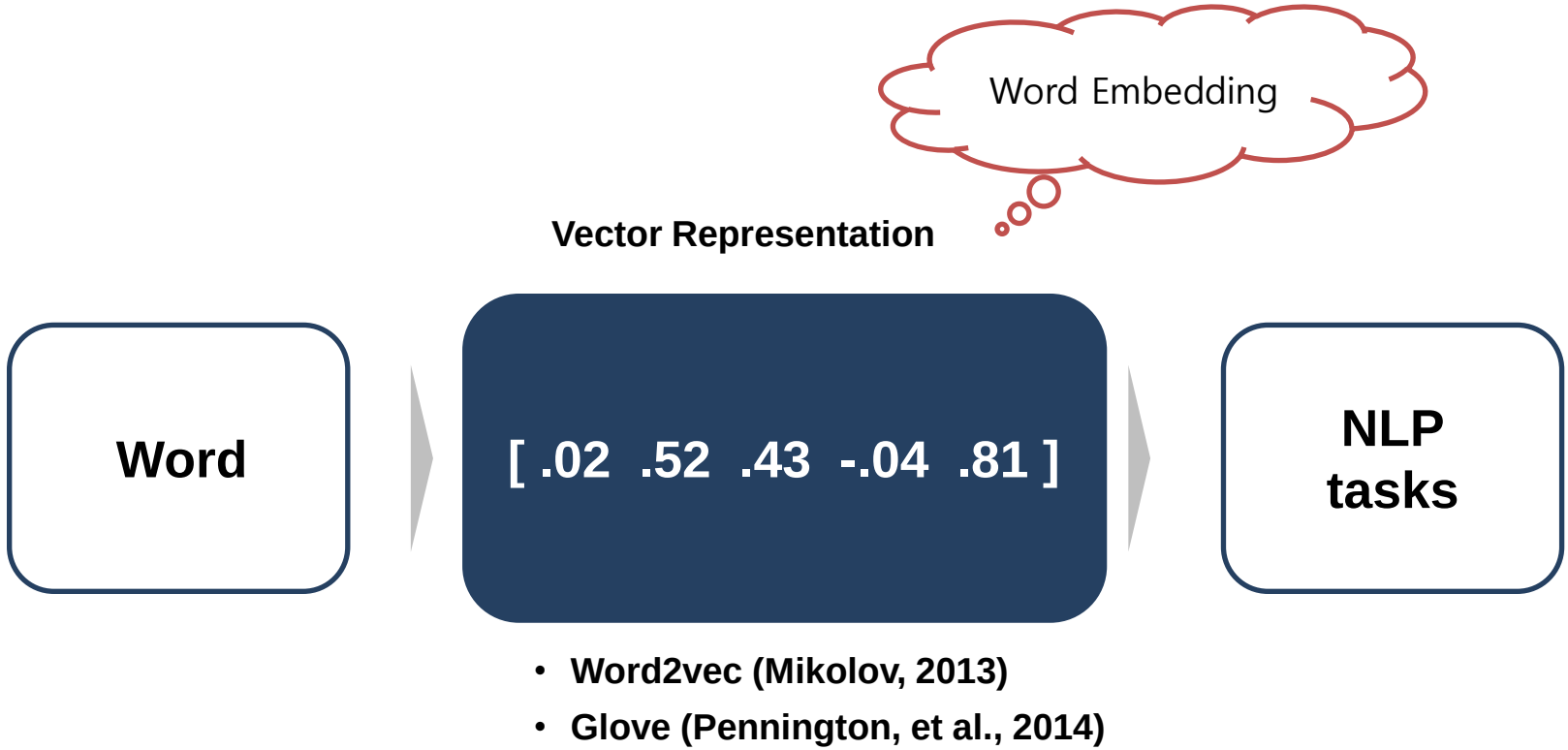
참고: Kira Radinsky 교수 강의자료

From SVD To Word2Vec

- SVD의 문제점
  - 계산에 너무 오랜 시간이 소요됨
    - n*m 행렬 계산 → $O(mn^2)$
  - 유연성이 떨어짐
    - 새로운 단어나 문서가 추가될 경우에 SVD를 처음부터 다시 수행
- 해결 방안
  - Learning representations by back-propagation errors (Rumelhart et al., 1986)
  - Neural probabilistic language model (Bengio et al., 2003)
  - NLP from Scratch (Collobert & Weston,2008)
  - Word2Vec (Mikolov et al., 2013)
    - Instead of capturing co-occurrence counts directly: Predict surrounding words of every word

Distributed Representation (Again)

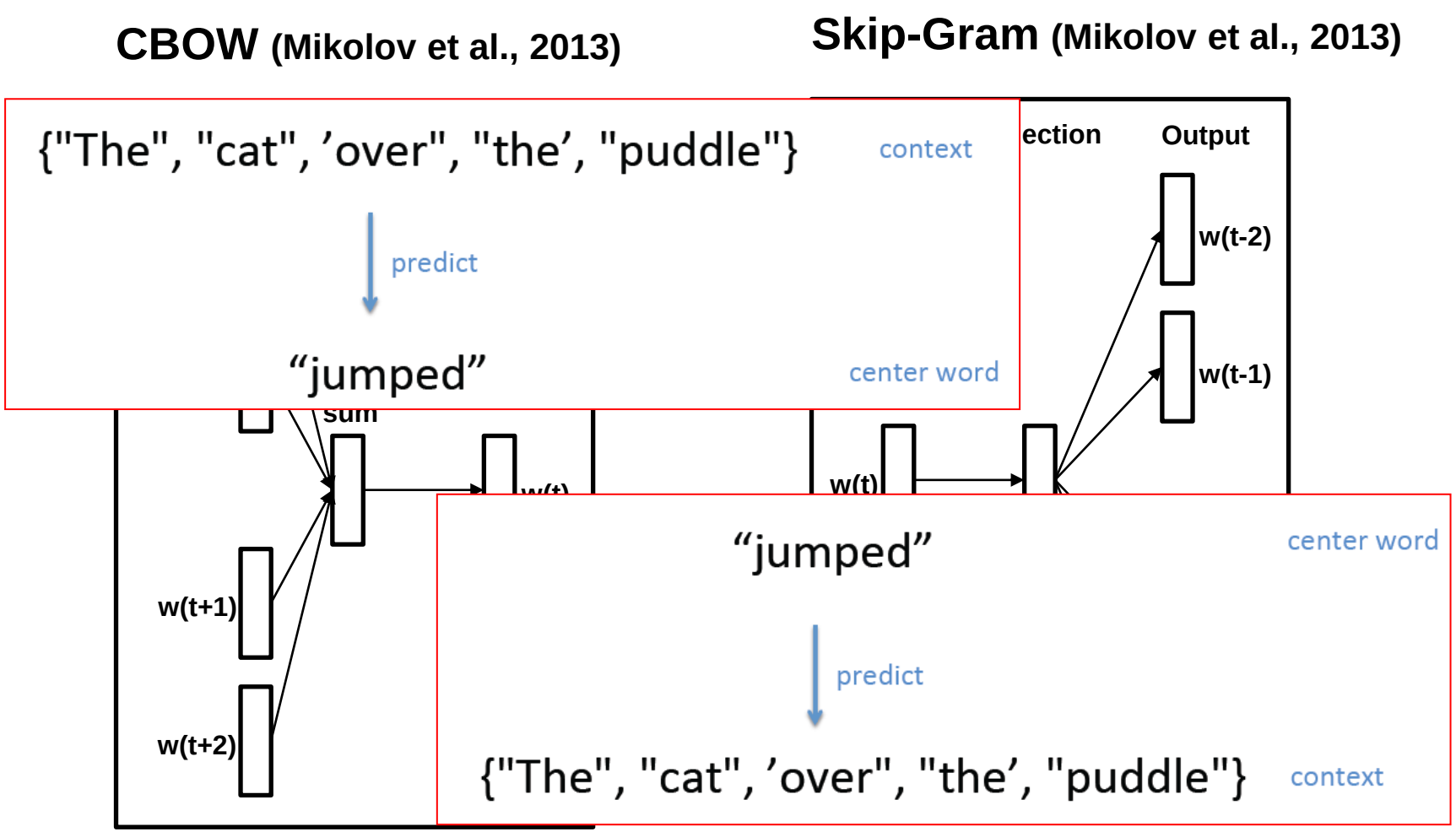
고차원 one-hot 벡터 → 저차원 실수 벡터
기본 아이디어: 비슷한 문맥에서 나온 단어는 비슷한 뜻을 가짐!



- **Word2vec (Mikolov, 2013)**
- **Glove (Pennington, et al., 2014)**

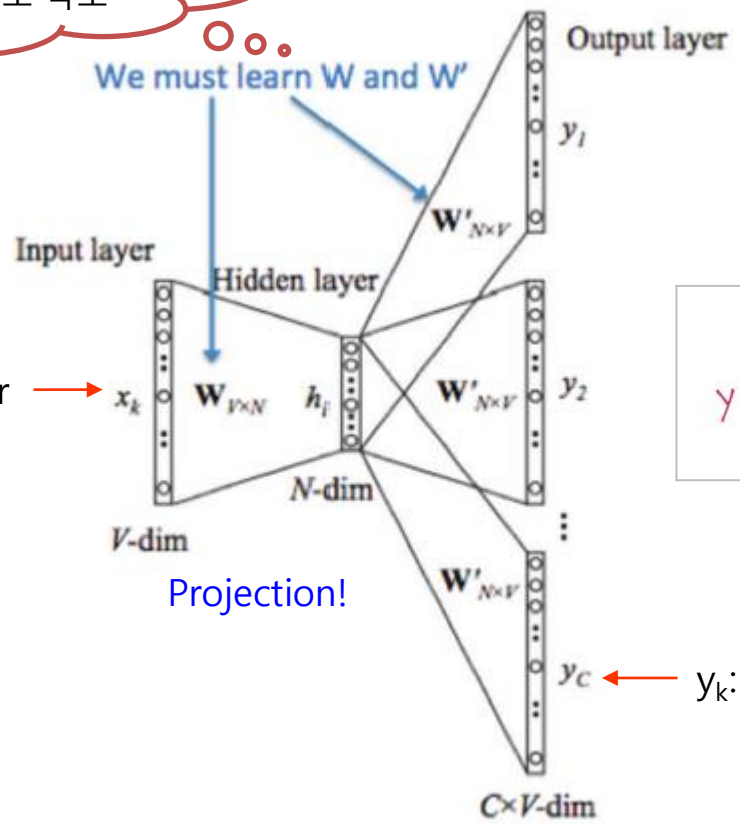
그림 출처: NLP 겨울학교(오혜연 교수)

Word2Vec


CBOW (Mikolov et al., 2013)
Skip-Gram (Mikolov et al., 2013)
{"The", "cat", 'over", "the', "puddle"}
context
predict
"jumped"
center word
ection
Output
w(t-2)
w(t-1)
sum
w(t)
w(t+1)
w(t+2)
"jumped"
center word
predict
{"The", "cat", 'over", "the', "puddle"}
context

How will it work?

N차원 word embedding:
V차원→ N차원으로 축소



x_k: V차원 one-hot vector

y_k: V차원 one-hot vector

Cost function? Cross entropy

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

WHY?

$$H(\hat{y}, y) = -y_i \log(\hat{y}_i)$$

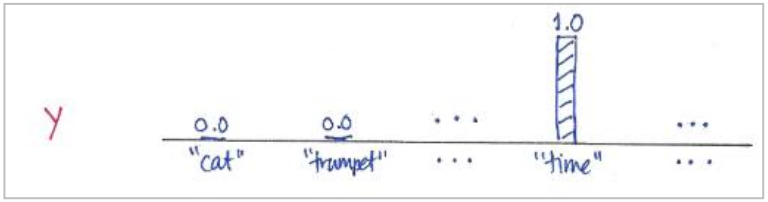


그림 출처: Kira Radinsky 교수 강의자료

SoftMax

Cost function? Cross entropy

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

WHY?

$$H(\hat{y}, y) = -y_i \log(\hat{y}_i)$$

Softmax: Linear regression generalization to multi-class

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\mathrm{T}} x)} \begin{bmatrix} \exp(\theta^{(1)\mathrm{T}} x) \\ \exp(\theta^{(2)\mathrm{T}} x) \\ \vdots \\ \exp(\theta^{(K)\mathrm{T}} x) \end{bmatrix}$$

Output: a K-dimensional vector (whose elements sum to 1)

확률로 만들려면?
V만큼의 계산 필요!

- Hierarchical SoftMax
- Negative Sampling
- Sub-Sampling Frequent Words

GloVe (Global Vectors)

Prediction → Probability

Main Insight: Ratio of co-occurrence probabilities can encode meaning

→ **Use global information (co-occurrence over corpus) while learning word vectors**

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

Solid가 문맥으로 주어졌을 때, ice와 steam의 비율이 크도록 학습!

Main Insight of GloVe

- Ratio of co-occurrence probabilities can encode meaning!

How can we capture this behavior in the word vector space?

Log-bilinear model: $w_i \cdot w_j = \log P(i|j)$

Vector differences: $w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$

x=solid, a=ice, b=steam

Think: a = “ice”, b = “steam”

The training objective of GloVe is to **learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence**.

Object Function of GloVe

그림 출처: Kira Radinsky 교수 강의자료

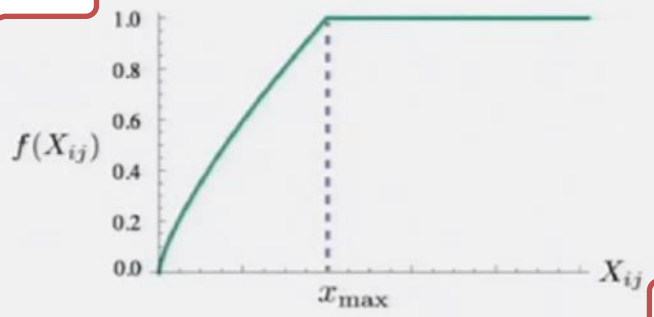
The log-bilinear model

$$J = \frac{1}{2}\sum_{ij} f(P_{ij})\left(w_i \cdot \tilde{w}_j - \log P_{ij}\right)^2$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

No Prob.

- X - cooccurrence matrix
- w - word vectors
- $\tilde{w}$ - context word vectors
- b - bias
- $\tilde{b}$ - bias



3/4

100

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Fast training, scalable to huge corpora,
Good performance even with small corpus, and small vectors

From One-hot Rep. to Distributed Rep.

- fastText (by facebook ← word2vec by google)
  - 부분 단어(subword)로 학습하여 노이즈에 강함

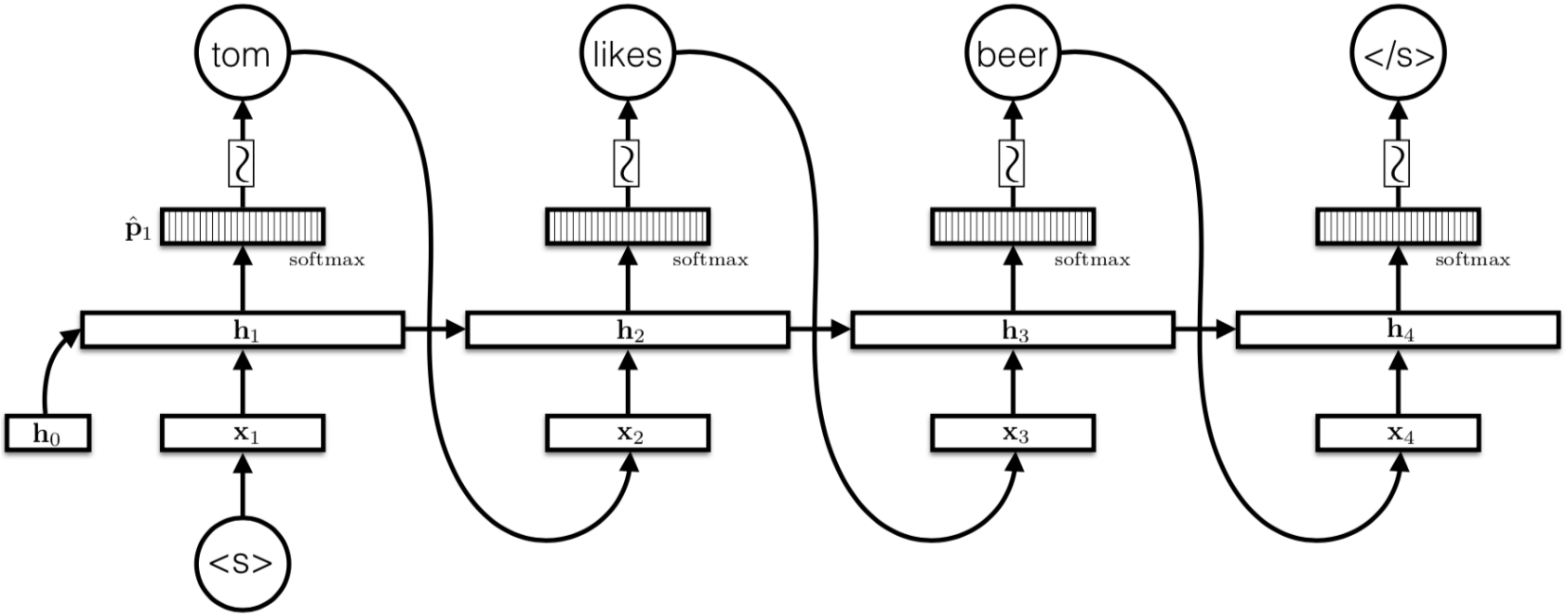
- GloVe (by stanford)
  - 동시 등장 확률을 함께 학습

Add Contexts

- CoVe (McCann et al., 2017)
  - MT-LSTM을 통한 사전학습; h=MT-LSTM(GloVe(w))
- ELMo (Peters et al., 2018)
  - Language Model을 통한 사전학습

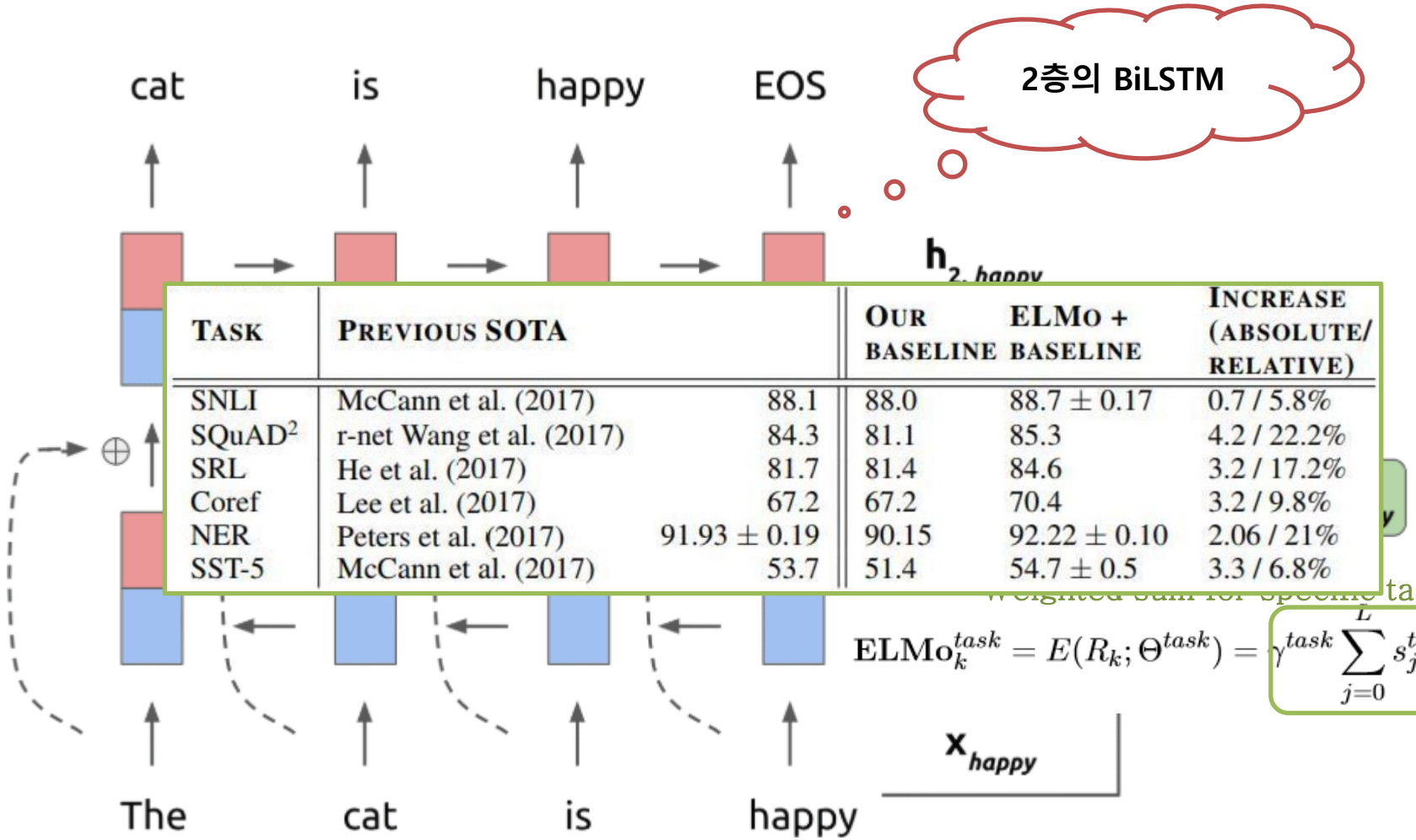
Language Model Using RNN

$$p(tom \mid \langle \mathtt{s} \rangle) \times p(likes \mid \langle \mathtt{s} \rangle, tom) \times p(beer \mid \langle \mathtt{s} \rangle, tom, likes) \times p(\langle /\mathtt{s} \rangle \mid \langle \mathtt{s} \rangle, tom, likes, beer)$$



* Figures by Chris Dyer

ELMo



Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SNLI	McCann et al. (2017)	88.1	88.0	88.7 ± 0.17	0.7 / 5.8%
SQuAD[2]	r-net Wang et al. (2017)	84.3	81.1	85.3	4.2 / 22.2%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

weighted sum for specific tasks

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

Pretrained Language Model: PART I

건국대학교 컴퓨터공학부 /
KAIST 전산학부 (겸직)

김학수

자연어처리연구실

10 Trends in Deep Learning NLP

1. Previous word embedding approaches are still important
2. Recurrent Neural Networks (RNNs) are no longer an NLP standard architecture
3. The Transformer will become the dominant NLP deep learning architecture
4. Pre-trained models will develop more general linguistic skills
5. Transfer learning will play more of a role
6. Fine-tuning models will get easier
7. BERT will transform the NLP application landscape
8. Chatbots will benefit most from this phase on NLP innovation
9. Zero shot learning will become more effective
10. Discussion about the dangers of AI could start to impact NLP research and applications

출처: https://blog.floydhub.com/ten-trends-in-deep-learning-nlp/

전이 학습 (Transfer Learning)

- 전이 학습
  - 사전 작업(source task)에 대하여 학습된 정보를 목표 작업(target task)에 활용하는 방법
  - 학습 데이터가 부족해도 목표 작업에 대한 수렴 속도 및 성능 향상을 꾀할 수 있음

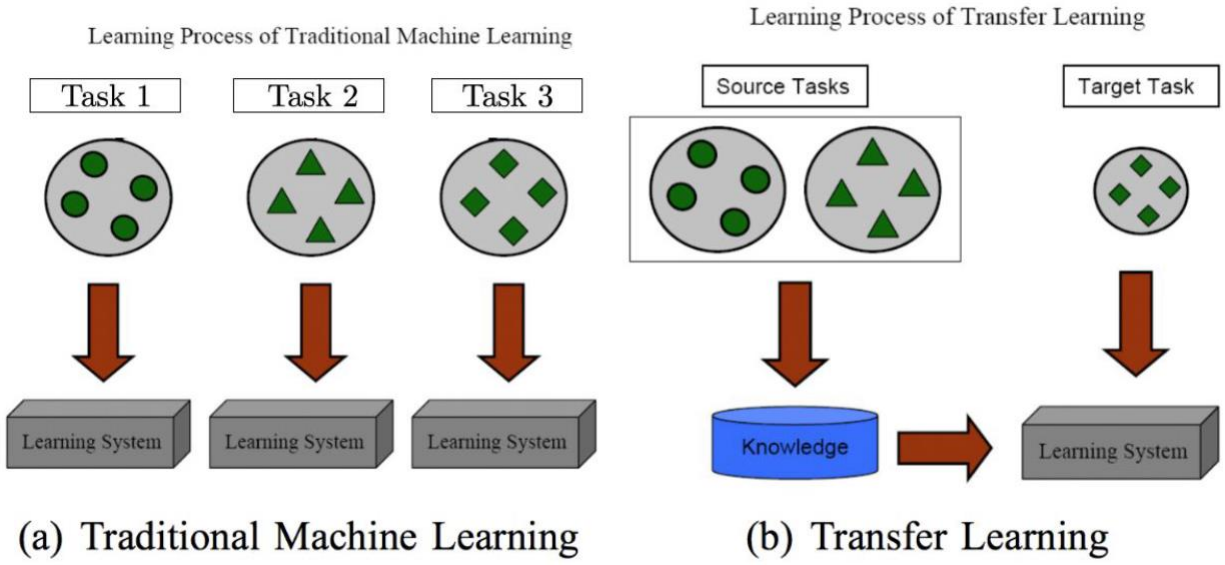
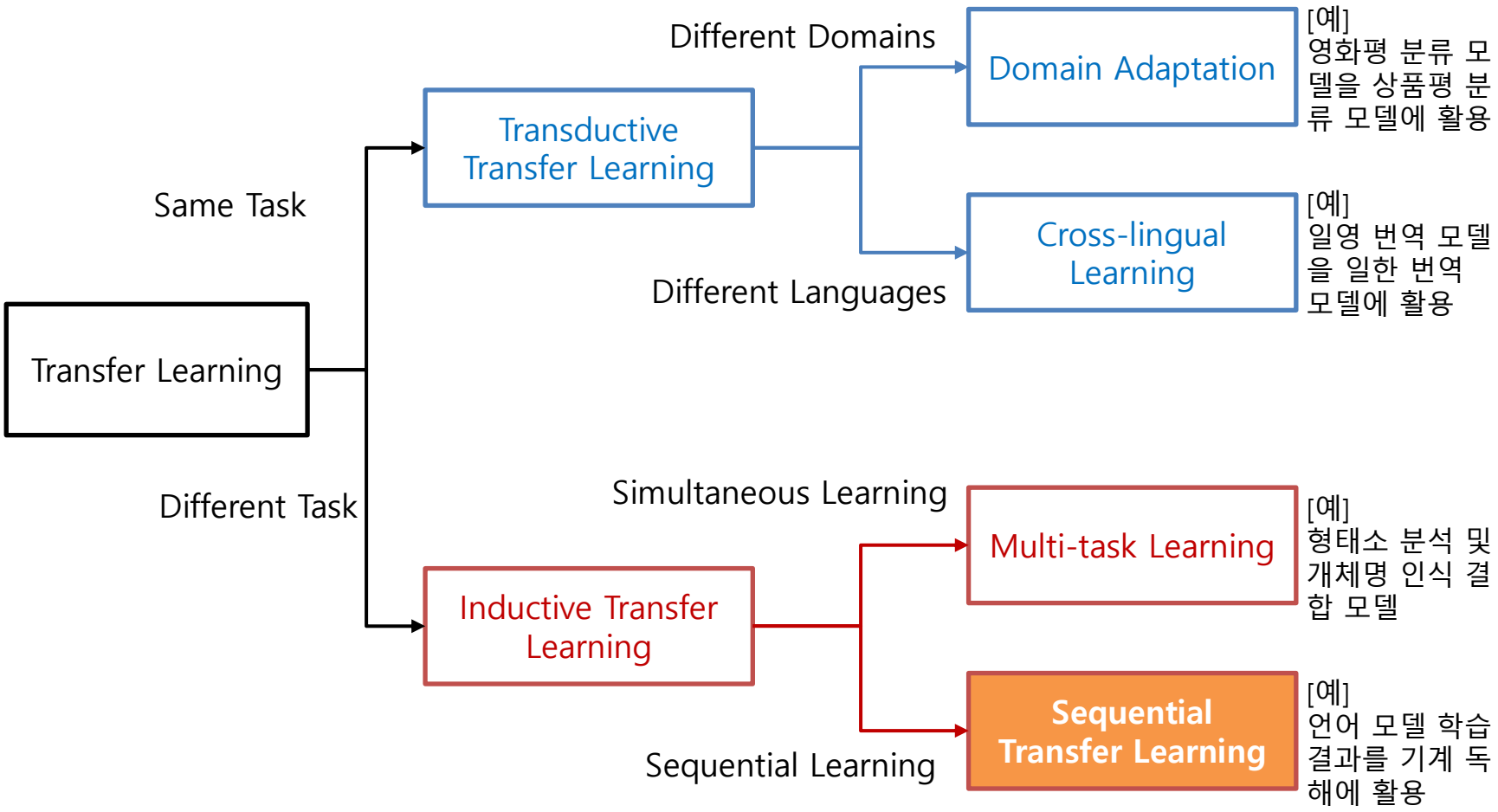


그림 출처: Pan and Yang 발표자료

전이 학습 유형

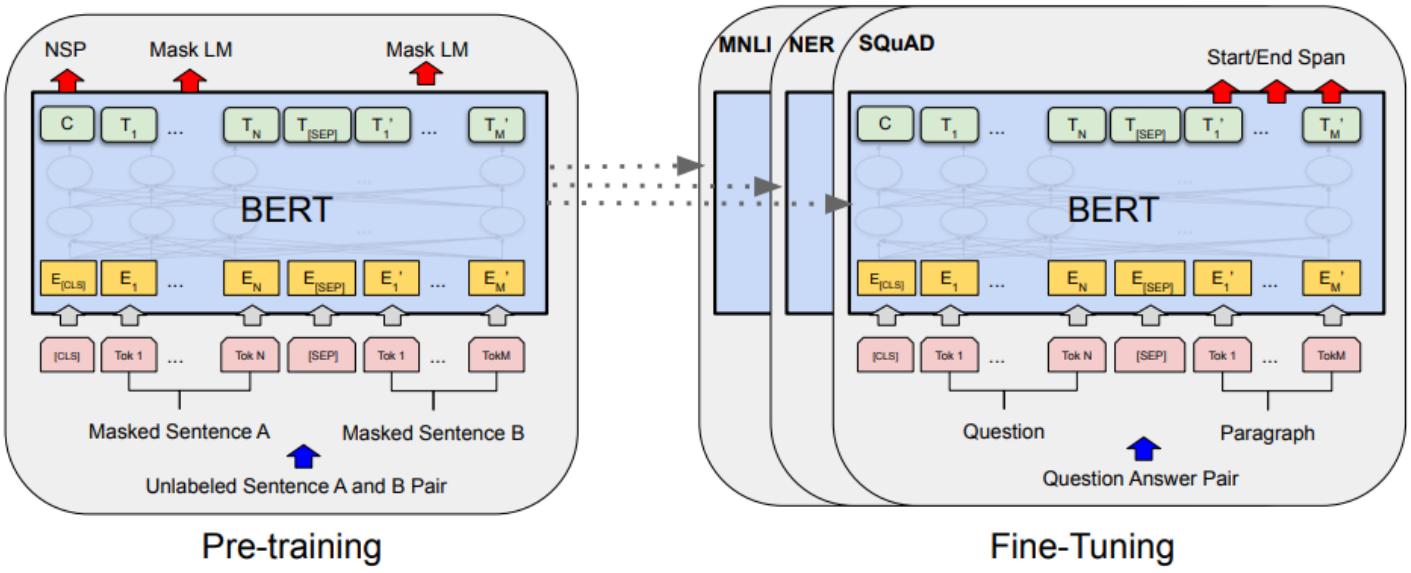

Transfer Learning
Same Task
Different Task
Transductive Transfer Learning
Inductive Transfer Learning
Different Domains
Different Languages
Simultaneous Learning
Sequential Learning
Domain Adaptation
Cross-lingual Learning
Multi-task Learning
Sequential Transfer Learning
[예] 영화평 분류 모델을 상품평 분류 모델에 활용
[예] 일영 번역 모델을 일한 번역 모델에 활용
[예] 형태소 분석 및 개체명 인식 결합 모델
[예] 언어 모델 학습 결과를 기계 독해에 활용

Inductive TL: Sequential Transfer Learning

- 순차 전이 학습
  - 사전 작업과 목표 작업이 다르고 각 작업에 대하여 순차적으로 학습을 수행하는 전이 학습 방법
    - 사전 작업과 목표 작업에 대한 데이터를 동시에 사용할 수 없는 경우
    - 사전 작업 학습 데이터가 목표 작업 학습 데이터보다 많은 경우
    - 여러 목표 작업에 대한 적용이 필요한 경우
- 대용량 사전 학습 언어 모델
  - Large-scaled pre-trained language model
    - BERT, GPT, ALBERT, RoBERTa, ELECTRA, XLNet 등
  - 다양한 NLP 문제에서 SOTA (State-Of-The-Art) 성능 획득

Self-Supervised Pre-training

- 사전 작업으로부터 저수준의 자질을 학습하고 이를 활용할 수 있는 여러 목표 작업에 학습과 성능 향상에 영향을 줄 수 있음
- 목표 작업에 대하여 적은 시간과 자원을 사용하여 비교적 높은 성능을 보일 수 있음



Pre-trained LM: BERT → ALBERT, RoBERTa → XLNet, ELECTRA, …

History of Pretrained LMs

Seq2Seq

Neural Machine Translation

Encoder - Decoder 구조의 모델

Attention + Seq2Seq

Neural Machine Translation

Decoder가 Source sentence의
중요한 정보에 집중하게 하자 !

Transformer

Neural Machine Translation

Self attention ,
Multi-head attention

GPT - 1
Transformer Decoder block

Task Agnostic
: Pretraining + Finetuning

언어 자체를 이해 할 수 있는
좋은 representation을 학습해보자 !

BERT
Transformer Encoder block

Task Agnostic
: Pretraining + Finetuning

언어를 더 잘 이해하기위해
Bidirectional하게 맥락을 파악해보자 !
모든 NLP Task에서 SOTA

GPT - 2
Transformer Decoder block

Task Agnostic
: Zeroshot task transfer

언어를 정확히 학습했다면
Finetuning없이 Zeroshot 만으로
좋은 성능을 낼 수 있다 !
7가지 NLP Task에서 SOTA

출처: 고려대 DSBA 연구실 이유경님의 발표자료

History of Pretrained LMs

XLNet
BERT + GPT ≅ AE + AR

Task Agnostic

BERT 이후 큰 성능향상을 보인 첫 모델
1) Factorization order를 고려하여 양방향 학습
2) AR formula를 통해 BERT한계 극복

RoBERTa
Optimize BERT

Task Agnostic

가장 최적화된 BERT를 만들어보자!
(학습시간, batch, train data 증가)

MASS
BERT + GPT ≅ AE + AR

Task Agnostic

Encoder와 Decoder에 상반된 Masking
1) Decoder : Encoder에서 masking된 단어 예측
2) Encoder : Masking되지 않은 단어 깊은 이해
3) Encoder, Decoder의 joint training 장려

BART
BERT + GPT ≅ AE + AR

Task Agnostic

Encoder에 다양한 noising 추가한
Text generation task에서 SOTA 달성

MT-DNN
Based on BERT

Task Agnostic

Multitask learning을 통해
universal representation을 생성해보자!
Pretrain 단계에서 multitask learning 진행

T5
Encoder-Decoder Transformer

Task Agnostic

모든 NLP task를 통합할 수 있도록
Text-to-text 프레임워크를 사용하자 !
현재 SOTA 모델

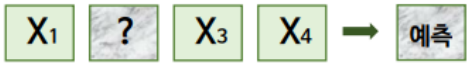
출처: 고려대 DSBA 연구실 이유경님의 발표자료

Auto Encoding vs. Auto Regressive

Pre-training의 대표적인 Objective

Auto Encoding

BERT는 Denoising AE라 볼 수 있음



Word sequence

$$\bar{\mathrm{x}} = [x_1, x_2, \cdots, x_T]$$

corrupted sequence

$$\hat{\mathrm{x}} = [x_1, [MASK], \cdots, x_T]$$

likelihood

$$p(\bar{\mathrm{x}}|\hat{\mathrm{x}}) \approx \prod_{t=1}^{T} p(x_t|\hat{\mathrm{x}})$$

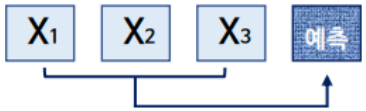
Objective function

$$Max_\theta \; logp_\theta(\bar{\mathrm{x}}|\hat{\mathrm{x}})$$

$$\approx \sum_{t=1}^{T} m_t logp_\theta(x_t|\hat{\mathrm{x}})$$

$$= \sum_{t=1}^{T} m_t log \frac{exp(H_\theta(\hat{\mathrm{x}})_t^T e(x_t))}{exp(H_\theta(\sum_{x'} exp(H_\theta(\hat{\mathrm{x}})_t^T e(x'))}$$

Auto Regressive



Word sequence

$$\mathrm{x} = [x_1, x_2, \cdots, x_T]$$

likelihood

$$p(\mathrm{x}) = \prod_{t=1}^{T} p(x_t|\mathrm{x}_{<t})$$

Objective function

$$Max_\theta \; logp_\theta(\mathrm{x})$$

$$= \sum_{t=1}^{T} logp_\theta(x_t|\mathrm{x}_{<t})$$

$$= \sum_{t=1}^{T} log \frac{exp(h_\theta(\mathrm{x}_{1:t-1})^T e(x_t))}{exp(h_\theta(\sum_{x'} exp(h_\theta(\mathrm{x}_{1:t-1})^T e(x'))}$$

출처: 고려대 DSBA 연구실 이유경님의 발표자료

OpenAI GPT vs. Google BERT

Pre-training a Transformer
Decoder for Language Modeling

Pre-training a Transformer
Encoder for Language Modeling

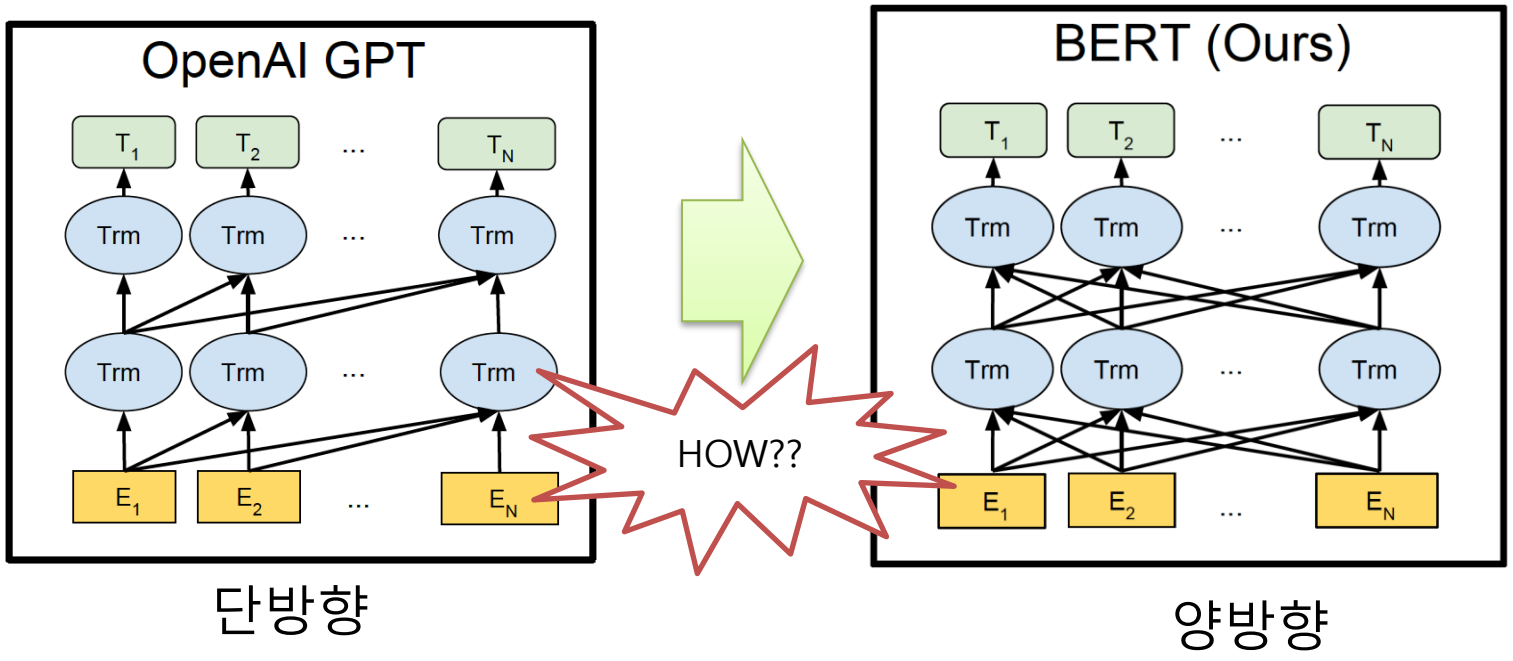


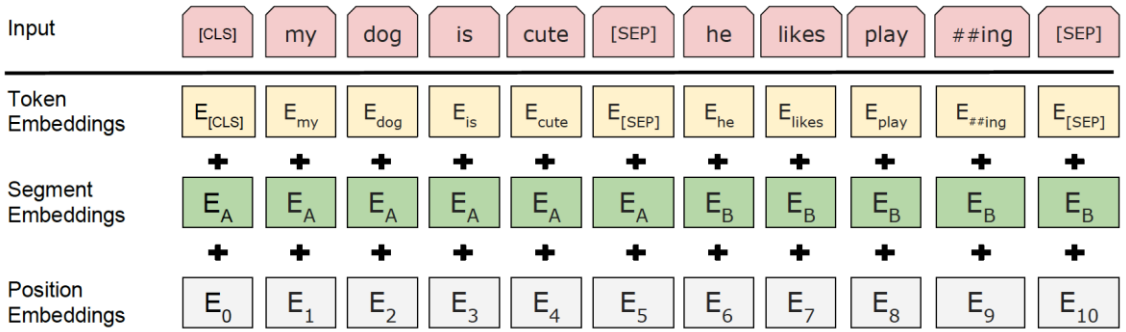
그림 출처: Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

BERT

- BERT Models
  - BERT-Base
    - L=12, A=12, H=768
    - Total Parameters=110M
  - BERT-Large
    - L=24, A=16, H=1024
    - Total Parameters=340M

L : # of layers
A : # of heads
H : Hidden size of FFN

- Input Representation



Pre-Training

- 학습 방법의 차별화

How to learn LM?
→ Word2Vec의 CBOW처럼
(Auto Encoding Model)

- Masked LM
  - 문장에 존재하는 단어를 마스킹 후 예측하게 함

- Next Sentence Prediction
  - 다음 문장 여부를 학습

BERT 목적 → Transfer Learning
(QA, NLI 등)

Masked LM

- 무작위 토큰 중 15%만 아래의 내용 적용
  - 80% of the time: Replace the word with the [MASK] token
    - my dog is hairy -> my dog is [MASK]
  - 10% of the time: Replace the word with a random word
    - my dog is hairy -> My dog is apple
  - 10% of the time: Keep the word unchanged,
    - my dog is hairy -> my dog is hairy

Next Sentence Prediction

- 주어진 문장이 다음 문장 관계인지 예측
  - 50%는 연속된 문장 학습
  - 50%는 랜덤으로 다른 문장 가져와서 학습

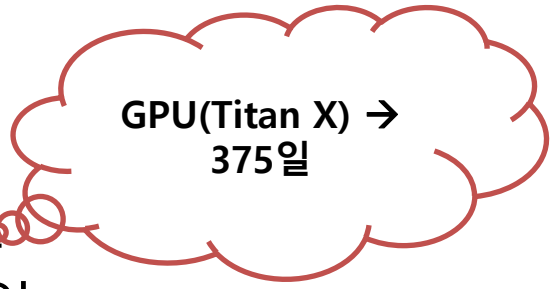
$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$
$$\text{Label} = \texttt{IsNext}$$

$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$
$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$
$$\text{Label} = \texttt{NotNext}$$

Datasets and Experimental Settings

- Datasets
  - Book Corpus (800M words)
  - English Wikipedia (2,500M words)

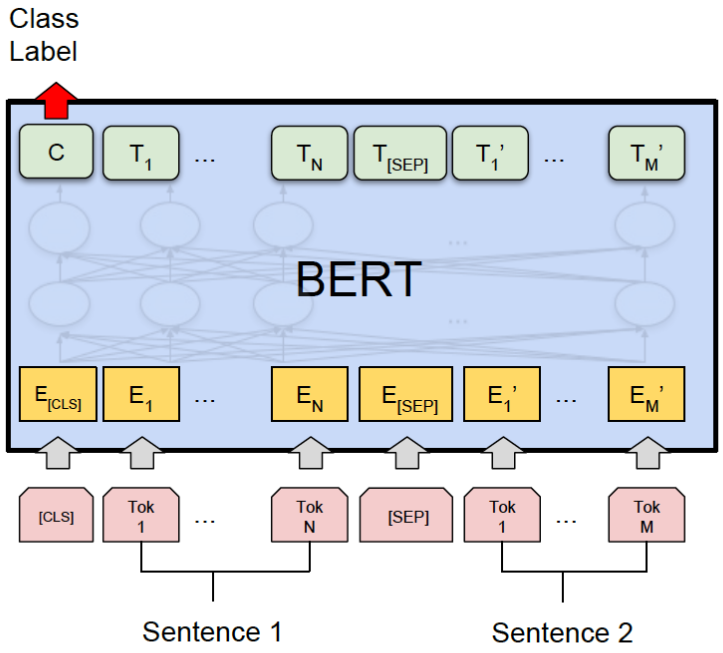
- Training
  - BERT-Base: Cloud TPU 16개로 4일
  - BERT-Large: Cloud TPU 64개로 4일



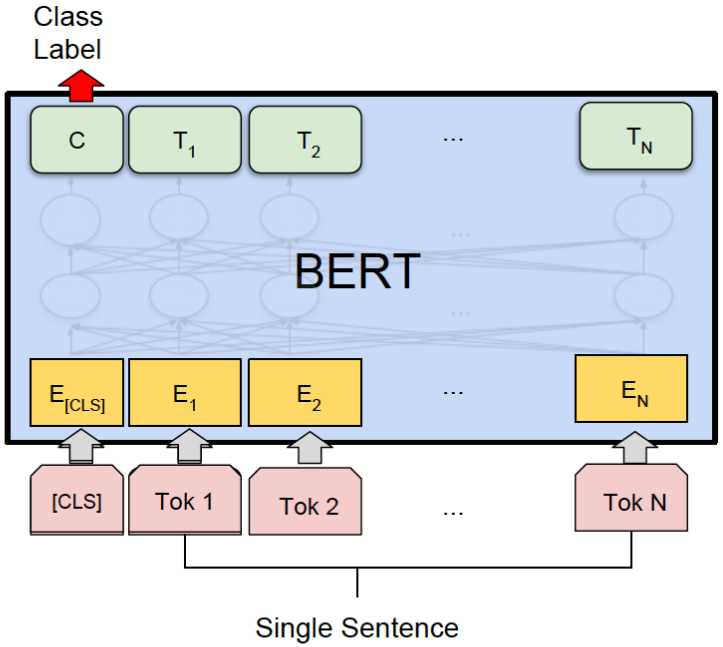
Experiments on Down-Stream Tasks

- Fine-tuning 방법으로 학습
- Classification task
- Sequence labeling task
- Span prediction task

Classification Task



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



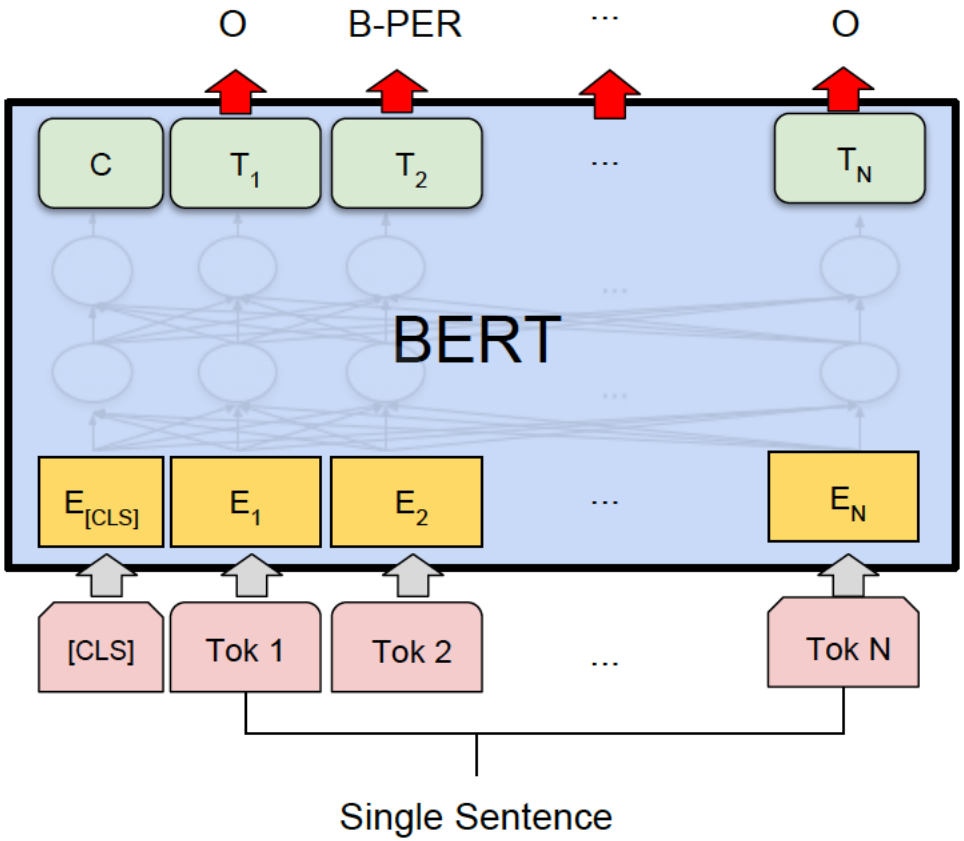
(b) Single Sentence Classification Tasks:
SST-2, CoLA

Classification Performance

- 문장/문서 분류

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Sequence Labeling Task

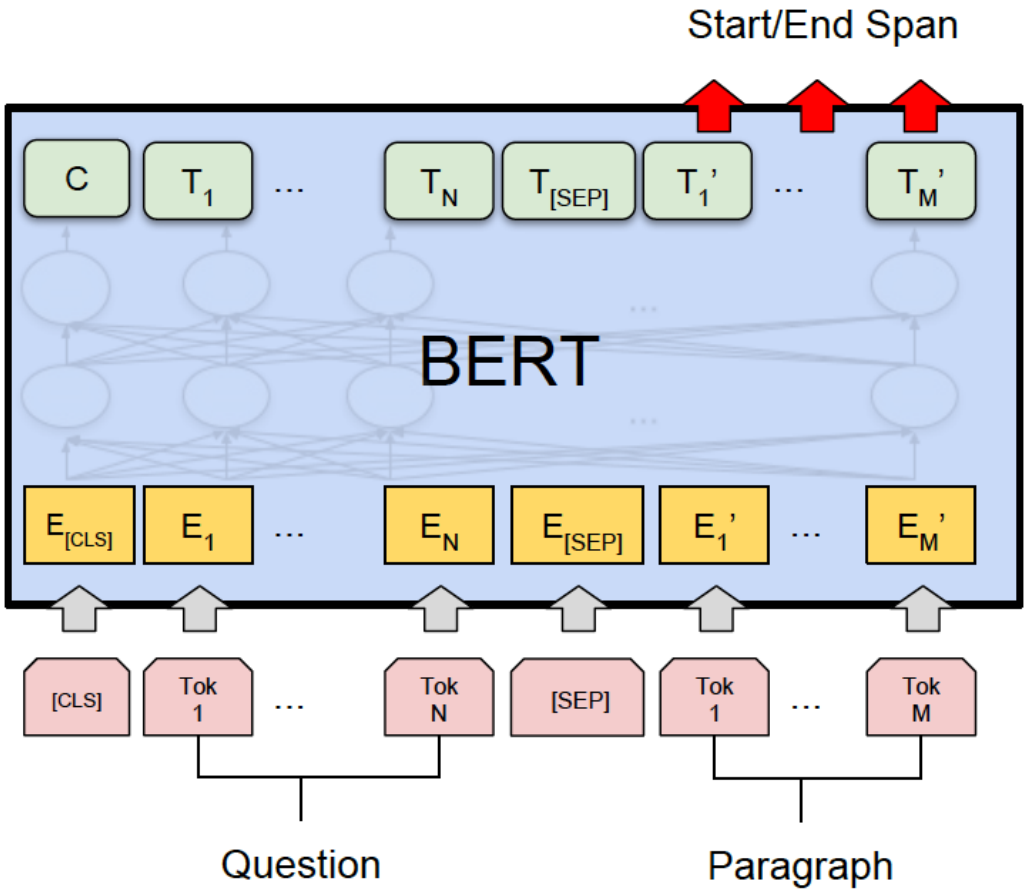


NER Performance

- 개체명 인식

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Span Prediction Task



SQuAD v1.1 Performance

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

RoBERTa

: BERT는 아직 Underfit된 모델, 가장 최적화된 BERT모델을 만들어보자 !

1. **Model 학습시간 증가, Batch 사이즈를 늘리고 train data 증가**
   - ➡ Pretrain에서 데이터 양을 늘릴수록 Downstream task의 성능이 증가함
2. **Next sentence prediction 제거함**
3. **Longer sequence를 추가함**
4. **Masking pattern을 dynamic하게 해줌**
   - ➡ BERT는 pretrain 전에 미리 masking 진행하는데, 학습이 진행될 때 똑같은 token이 masking된다는 문제가 있음 (Bias)
     1) 똑같은 데이터에 대해 masking을 10번 다르게 적용하여 학습
     2) Input이 들어갈 때 마다 masking진행

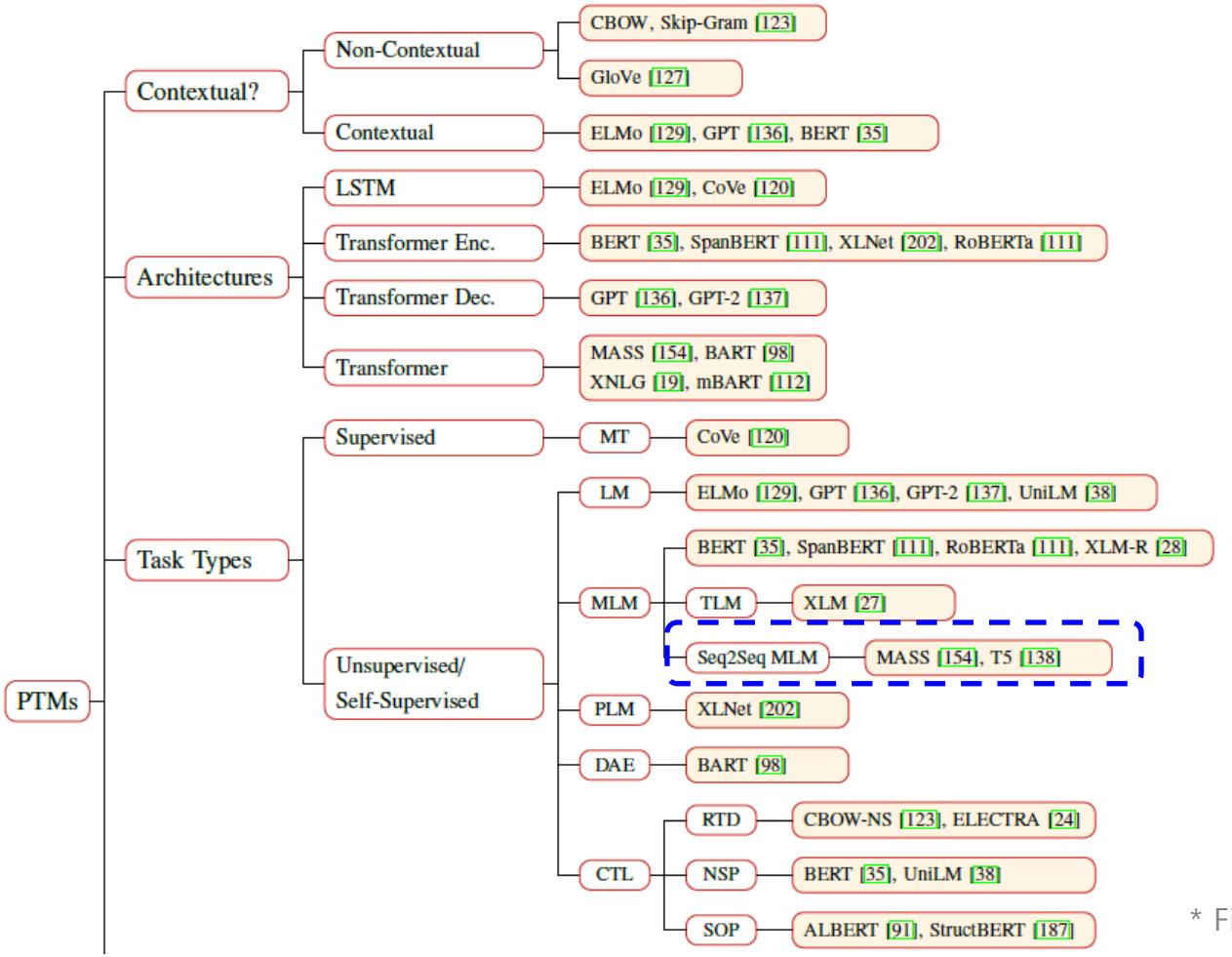
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

: model은 BERT와 동일하기 때문에 구조적 변화가 전혀 없으나 타 모델 대비 성능이 높음

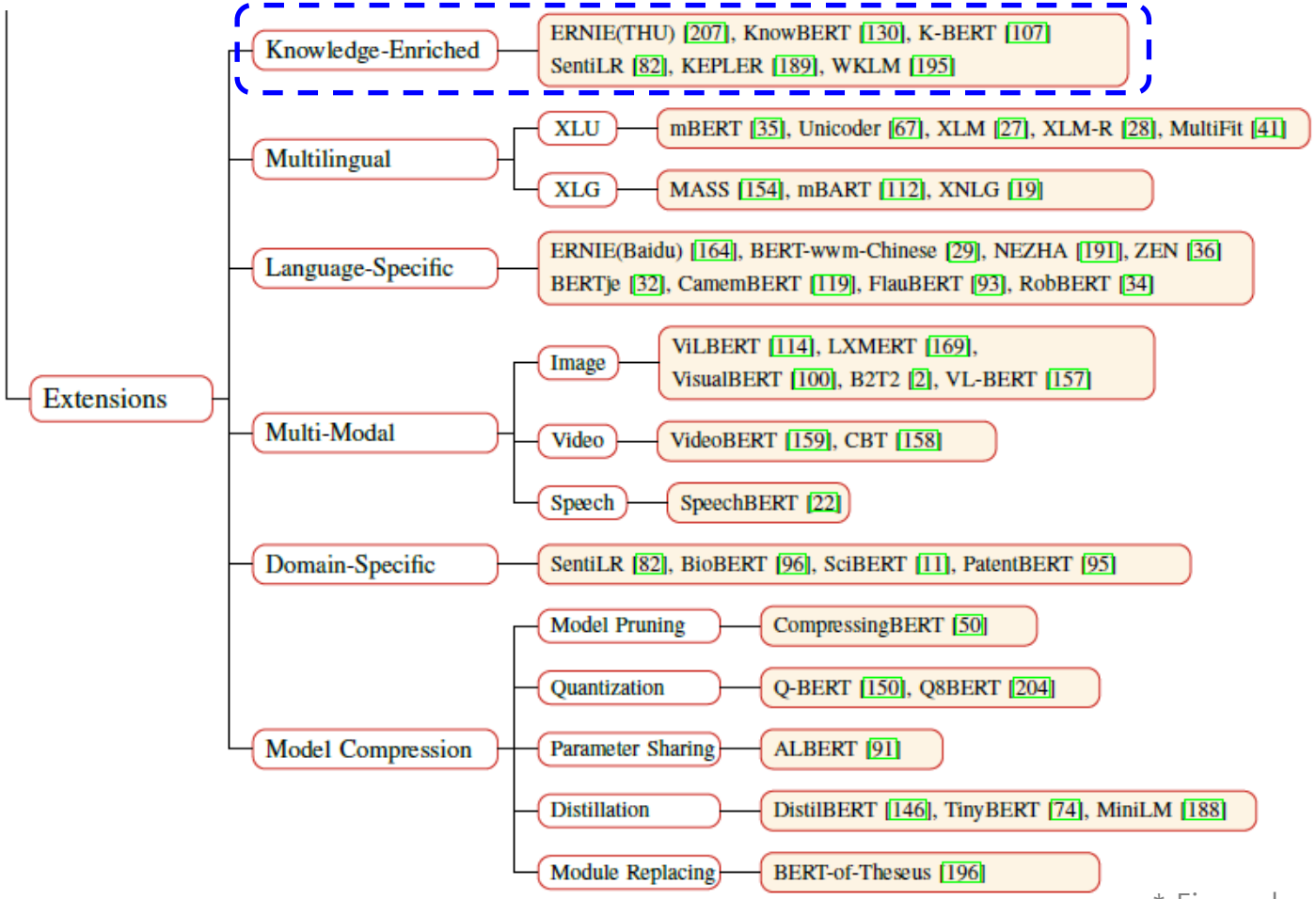
출처: 고려대 DSBA 연구실 이유경 발표자료

PTM Taxonomy



* Figure by Qiu et al.

PTM Taxonomy



* Figure by Qiu et al.

Edited by Harksoo Kim

KU KONKUK UNIVERSITY

KAIST

Pretrained Language Model: PART II

건국대학교 컴퓨터공학부 /
KAIST 전산학부 (겸직)

김학수

자연어처리연구실

KU KONKUK UNIVERSITY

KAIST

ALBERT: A Lite BERT

자연어처리연구실

Introduction

- Pre-traini
  - 다양한 M
  - 거대한 네 인 구글로 수행

- Pre-traini
  - 메모리의 서)
  - Fine-tun 없음
  - 학습 속

System	Seq Length	Max Batch Size
BERT-Base	64	64
...	128	32
...	256	16
...	320	14
...	384	12
...	512	6
BERT-Large	64	12
...	128	6
...	256	2
...	320	1
...	384	0
...	512	0

12GB Titan X GPU

파라미터 수를 획기적으로 줄인 BERT 필요!

Factorized Embedding Parameterization

- 기존 모델들은 embedding size와 hidden size를 비례시키는 경우가 많음
  - WordPiece Embedding은 문맥에 의존적이지 않은 표현 학습
  - Hidden-layer는 문맥에 의존적인 표현 학습
  - H>>E가 되면 모델을 효율적으로 학습할 수 있음

Factorized Embedding Parameterization

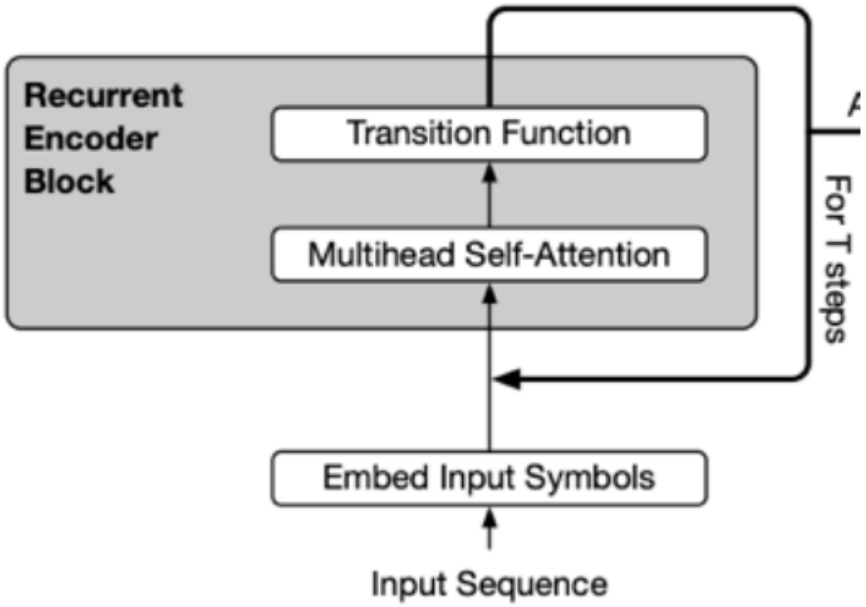
- 대부분의 NLP task에선 대용량 Vocab V를 사용
  - E≡H인 상황에서, V X E는 파라미터 개수가 쉽게 폭발적으로 상승
  - 이를 해결하기 위해 Embedding layer를 분해
    - H >> E일때 더욱 중요!

O(V x H) → O(V x E + E x H)

50,000 x 100 = 5,000,000 → 50,000 x 50 + 50 x 100 = 2,505,000

Cross-layer Parameter Sharing

- 파라미터의 효율성을 향상시키기 위한 방법
  - 다양한 공유 방법 존재: FFN? Attention layer?
  - 네트워크를 안정시키는 효과



Inter-Sentence Coherence Loss

- BERT : Next Sentence Prediction (NSP)
  - Masked language modeling에 비해 task의 난이도가 부족하기 때문에 비효율적
  - 주제 예측과 차이가 없음
  - Inter-sentence modeling은 언어 모델링에서 중요하기 때문에, loss를 Sentence Order Prediction(SOP)로 계산

Discourse-level의 일관성을 얻을 수 있음

- Positive sample : 같은 문서내의 두 연속된 문장
- Negative sample : Positive sample의 순서를 바꾼 문장

Experimental Setups

- BERT와 동일한 실험 환경 유지
  - BookCorpus, English Wikipedia
  - Maximum input length to 512
    - 10% 확률로 512보다 더 짧은 입력 생성
  - SentencePiece 사용
  - Batch size 4,096
  - LAMB optimizer 사용

Experiments: BERT vs. ALBERT

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

Models	Steps	Time	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
BERT-large	400k	34h	93.5/87.4	86.9/84.3	87.8	94.6	77.3	87.2
ALBERT-xxlarge	125k	32h	**94.0/88.1**	**88.3/85.3**	87.8	**95.4**	**82.5**	**88.7**

Experiments

- Factorized Embedding에 따른 성능 변화

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

- Parameter Sharing에 따른 성능 변화

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

Experiments

- Coherence Loss 사용에 따른 성능 변화

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Experiments: ALBERT vs. Others

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task si										
BERT-large		.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large		.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

# of training steps

Experiments: ALBERT vs. Others

Models	SQuAD1.1 dev	SQuAD2.0 dev	SQuAD2.0 test	RACE test (Middle/High)
Single model (from leaderboard as of Sept. 23, 2019)				
BERT-large	90.9/84.1	81.8/79.0	89.1/86.3	72.0 (76.6/70.1)
XLNet	94.5/89.0	88.8/86.1	89.1/86.3	81.8 (85.5/80.2)
RoBERTa	94.6/88.9	89.4/86.5	89.8/86.8	83.2 (86.5/81.3)
UPM	-	-	89.9/87.2	-
XLNet + SG-Net Verifier++	-	-	90.1/87.2	-
ALBERT (1M)	94.8/89.2	89.9/87.2	-	86.0 (88.2/85.1)
ALBERT (1.5M)	**94.8/89.3**	**90.2/87.4**	**90.9/88.1**	**86.5 (89.0/85.5)**
Ensembles (from leaderboard as of Sept. 23, 2019)				
BERT-large	92.2/86.2	-	-	-
XLNet + SG-Net Verifier	-	-	90.7/88.2	-
UPM	-	-	90.7/88.2	
XLNet + DAAF + Verifier	-	-	90.9/88.6	-
DCMN+	-	-	-	84.1 (88.5/82.3)
ALBERT	**95.5/90.1**	**91.4/88.9**	**92.2/89.7**	**89.4 (91.2/88.6)**

Conclusion

- ALBERT는 BERT에서 세가지가 개선됨
  - Embedding decomposition
  - Parameter sharing
  - SOP loss

- 현재까지 다른 pre-training model에 비해 빠른 학습속도와 좋은 성능을 보임

KU KONKUK UNIVERSITY

KAIST

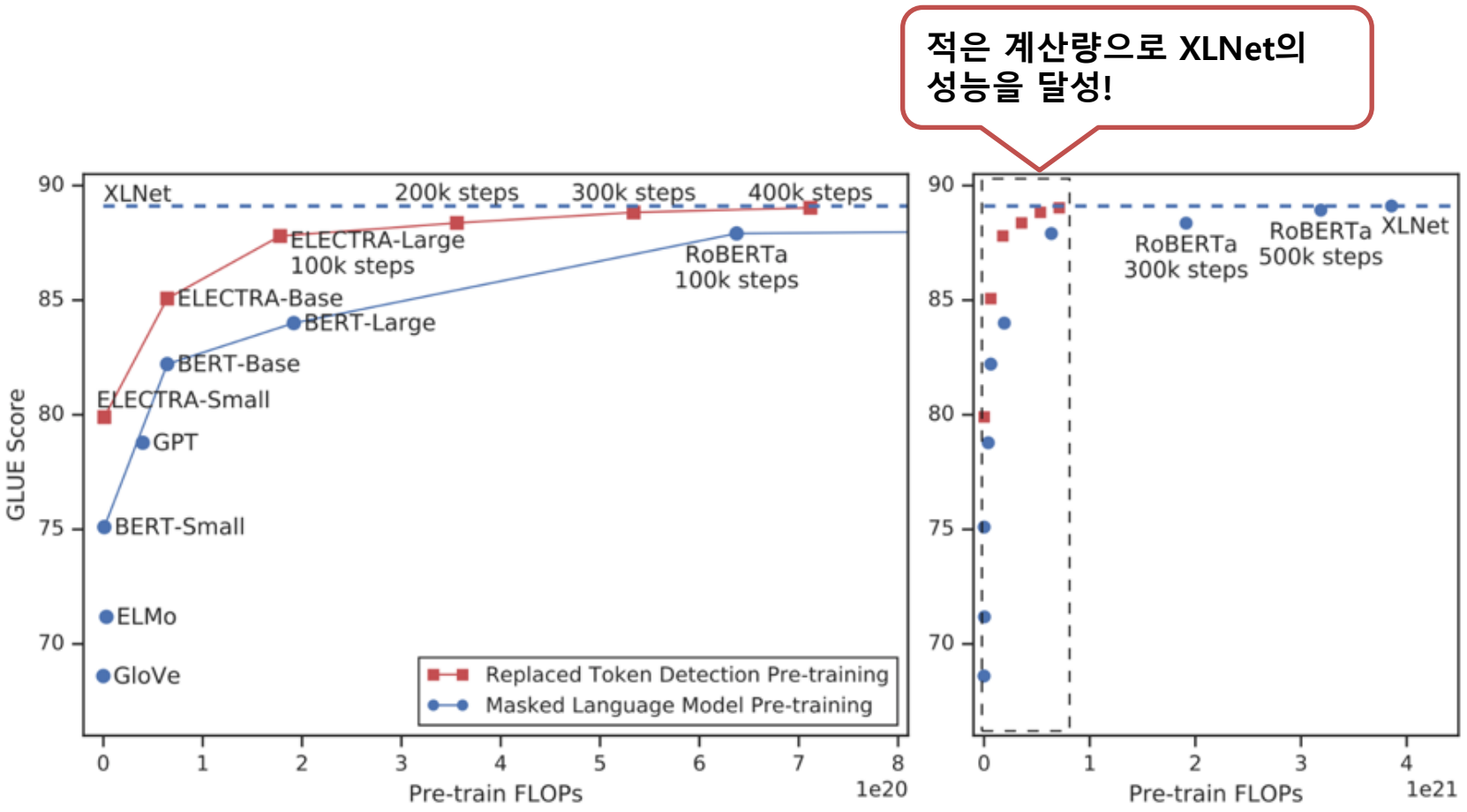
ELECTRA: Discriminator-based PTM

자연어처리연구실

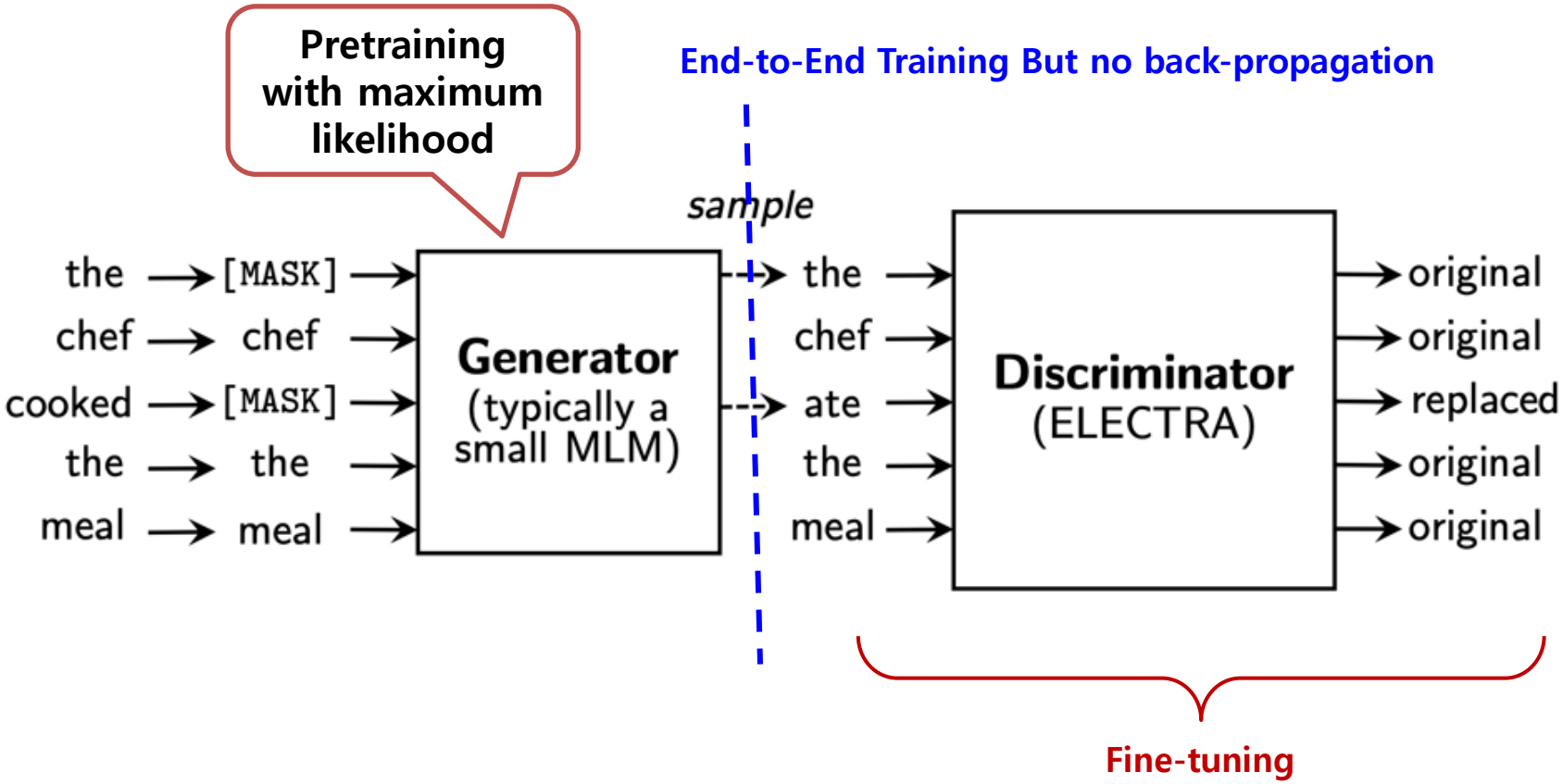
Introduction

- ELECTRA
  - **E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
  - 기존 Masked Language Modeling의 문제점
    - 전체 토큰 중 15%에 대해서만 loss가 발생한다. (= 하나의 example에 대해서 고작 15%만 학습함)
    - 학습 때는 [MASK] 토큰을 모델이 참고하여 예측하지만 실제(inference)로는 [MASK] 토큰이 존재하지 않는다.
    - Replaced Token Detection (RTD)이라는 새로운 pretraining 태스크를 제안

Token Detection vs. Masked LM



Model Architecture



Difference to GAN

- Generator가 동일 토큰을 생성한 경우
  - GAN: Negative Sample로 간주
  - ELECTRA: Positive Sample로 간주
- Adversarial 학습이 아니고 maximum likelihood를 학습
  - Generator의 샘플링 과정 때문에 역전파 불가능
  - 강화 학습을 시도했지만 maximum likelihood 보다 성능이 떨어짐
- Discriminator의 입력으로 노이즈를 넣지 않음

Experimental Setups

- 실험 모델 구성
  - Generator
    - BERT의 MLM과 동일
  - Discriminator
    - 입력 토큰 시퀀스에 대해서 각 토큰이 original인지 replaced 인지 이진 분류
  - Generator loss와 Discriminator loss의 합을 최소화
  - Pre-training을 마친 뒤에 Generator는 버리고 Discriminator만 취해서 downstream task로 fine-tuning

Experiments: Size and Speed

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

Experiments: Performances

- Dev Set

# of training steps

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

- Test Set

Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	**97.1**	**91.2**	92.0	90.5	**91.3**	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	**97.1**	90.5	**92.6**	90.4	90.9	–	88.5	**92.5**	89.1	–
ELECTRA	3.1e21 (1x)	**71.7**	**97.1**	90.7	92.5	**90.8**	**91.3**	**95.8**	**89.8**	**92.5**	**89.5**	**89.4**

KU KONKUK UNIVERSITY

KAIST

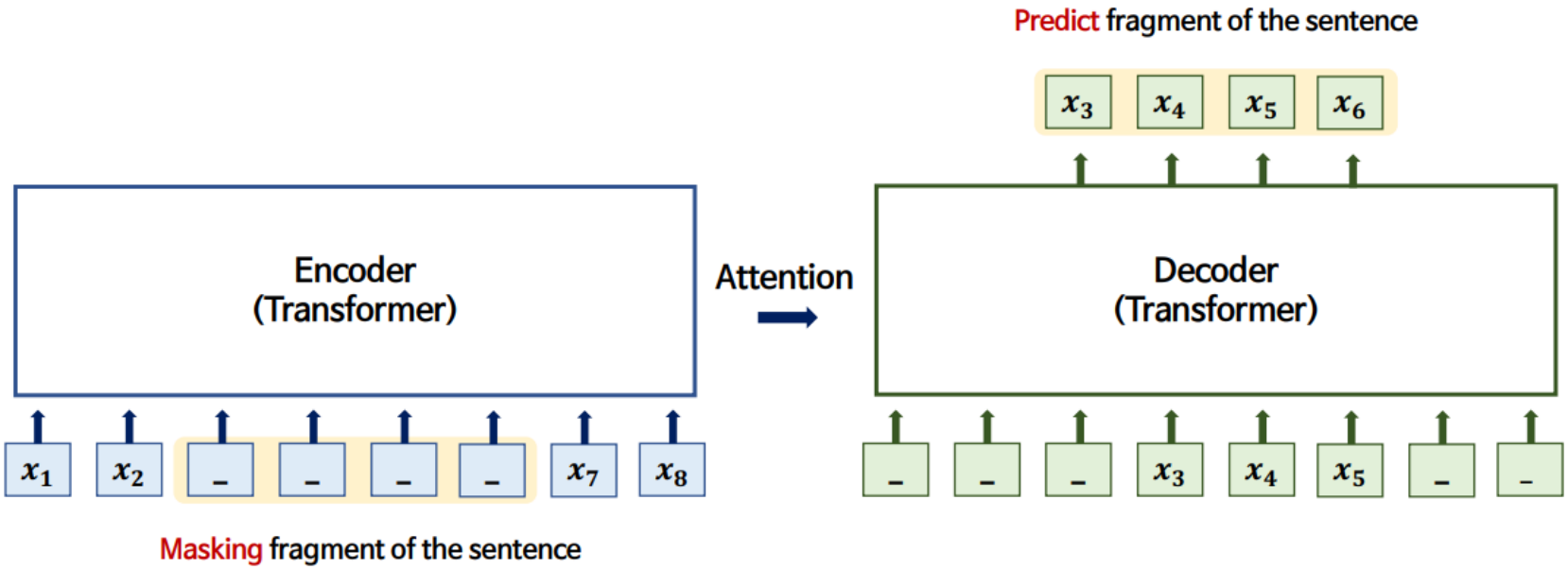
BART: Bidirectional and Auto-Regressive Transformer

자연어처리연구실

MASS

- **MASS: Masked Sequence to Sequence Pre-training for Language Generation**

Mass model structure



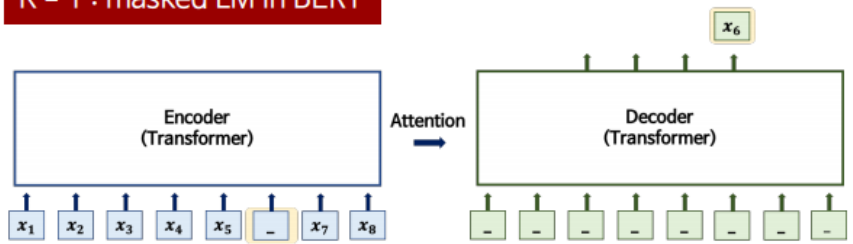
: Encoder와 Decoder의 input을 다르게 함으로서 Decoder가 Encoder에 더 의존할 수 있도록 함

출처: 고려대 DSBA 연구실 이유경님 발표자료

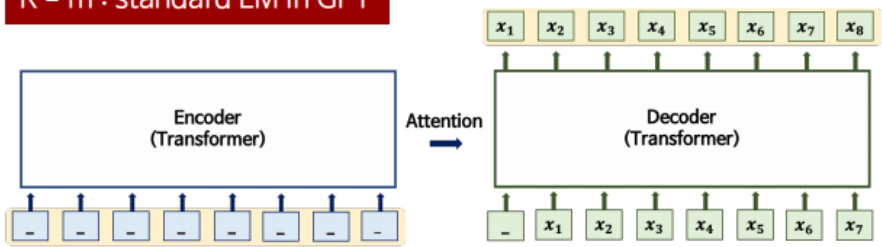
MASS

K : length of the masked fragment of the sentence

K = 1 : masked LM in BERT



K = m : standard LM in GPT



NMT

Newstest 2014, 2016

Method	Setting	en - fr	fr - en	en - de	de - en	en - ro	ro - en
Artetxe et al. (2017)	2-layer RNN	15.13	15.56	6.89	10.16	-	-
Lample et al. (2017)	3-layer RNN	15.05	14.31	9.75	13.33	-	-
Yang et al. (2018)	4-layer Transformer	16.97	15.58	10.86	14.62	-	-
Lample et al. (2018)	4-layer Transformer	25.14	24.18	17.16	21.00	21.18	19.44
XLM (Lample & Conneau, 2019)	6-layer Transformer	33.40	33.30	27.00	34.30	33.30	31.80
MASS	6-layer Transformer	**37.50**	**34.90**	**28.30**	**35.20**	**35.20**	**33.10**

Text summarization

Method	RG-1 (F)	RG-2 (F)	RG-L (F)
BERT+LM	37.75	18.45	34.85
DAE	35.97	17.17	33.14
MASS	**38.73**	**19.71**	**35.96**

Gigaword corpus 3.8M

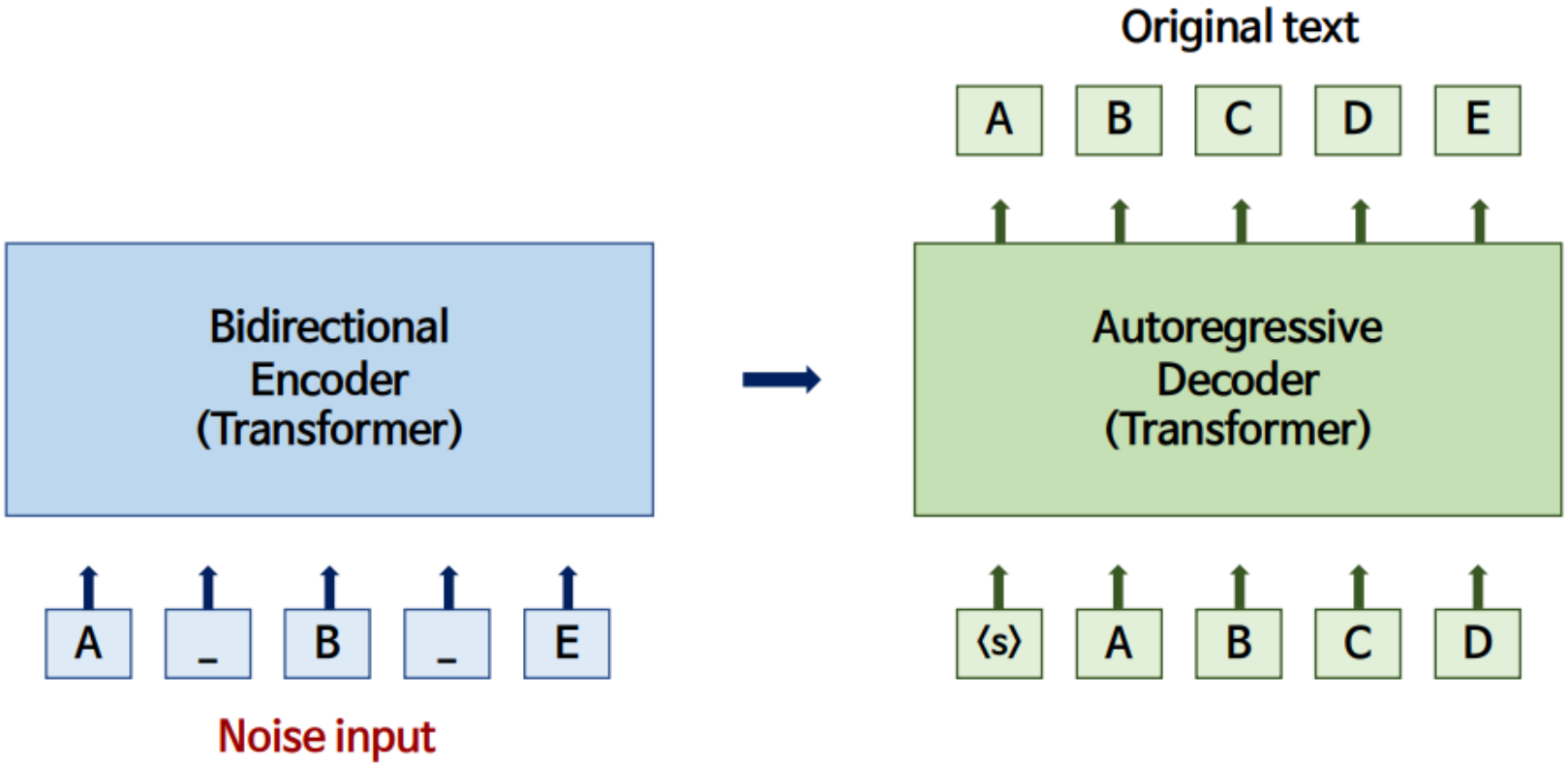
Conversational response generation

Method	Data = 10K	Data = 110K
Baseline	82.39	26.38
BERT+LM	80.11	24.84
MASS	**74.32**	**23.52**

Cornell movie dialog corpus

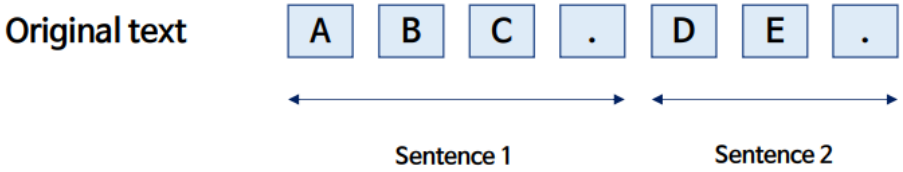
출처: 고려대 DSBA 연구실 이유경님 발표자료

Denoising Autoencoder



출처: 고려대 DSBA 연구실 이유경님 발표자료

Noises



1. Token Masking

A [M] C . [M] E .

- ✓ 임의의 Token을 [MASK]로 교체함.
- ✓ BERT의 Masking을 따르기 때문에 input sequence는 유지해야함
- ✓ [MASK] token이 무엇인지 예측해야 함

2. Token Deletion

A C E

- ✓ 임의의 Token을 삭제함
- ✓ 삭제한 token의 위치를 찾아야함

3. Text infilling

A [M] . D [M] E .

- ✓ Poisson dist. (Lambda =3)에서 span length를 뽑아 하나의 [MASK] token으로 대체함
- ✓ Span length가 0인 경우, 해당 위치에 [MASK] token을 추가함
- ✓ [MASK]로 대체된 token에 몇개의 token이 존재할지 예측해야 함

4. Sentence Permutation

D E . A B C .

- ✓ 문장의 순서를 랜덤으로 섞음

5. Document Rotation

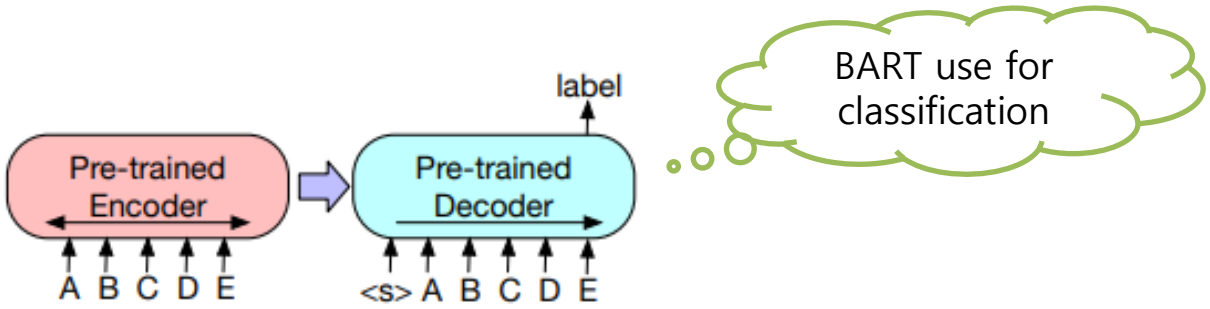
C . D E . A B

- ✓ 하나의 token을 uniformly하게 뽑은 후, 그 토큰을 시작점으로 회전함
- ✓ 모델이 문서의 start point를 찾도록 학습시킴

출처: 고려대 DSBA 연구실 이유경님 발표자료

Experiments

	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	MNLI m/mm	SST Acc	QQP Acc	QNLI Acc	STS-B Acc	RTE Acc	MRPC Acc	CoLA Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	**89.0**/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/**94.6**	**86.5/89.4**	**90.2/90.2**	96.4	92.2	94.7	**92.4**	86.6	**90.9**	**68.0**
BART	88.8/**94.6**	86.1/89.2	89.9/90.1	**96.6**	**92.5**	**94.9**	91.2	**87.0**	90.4	62.8



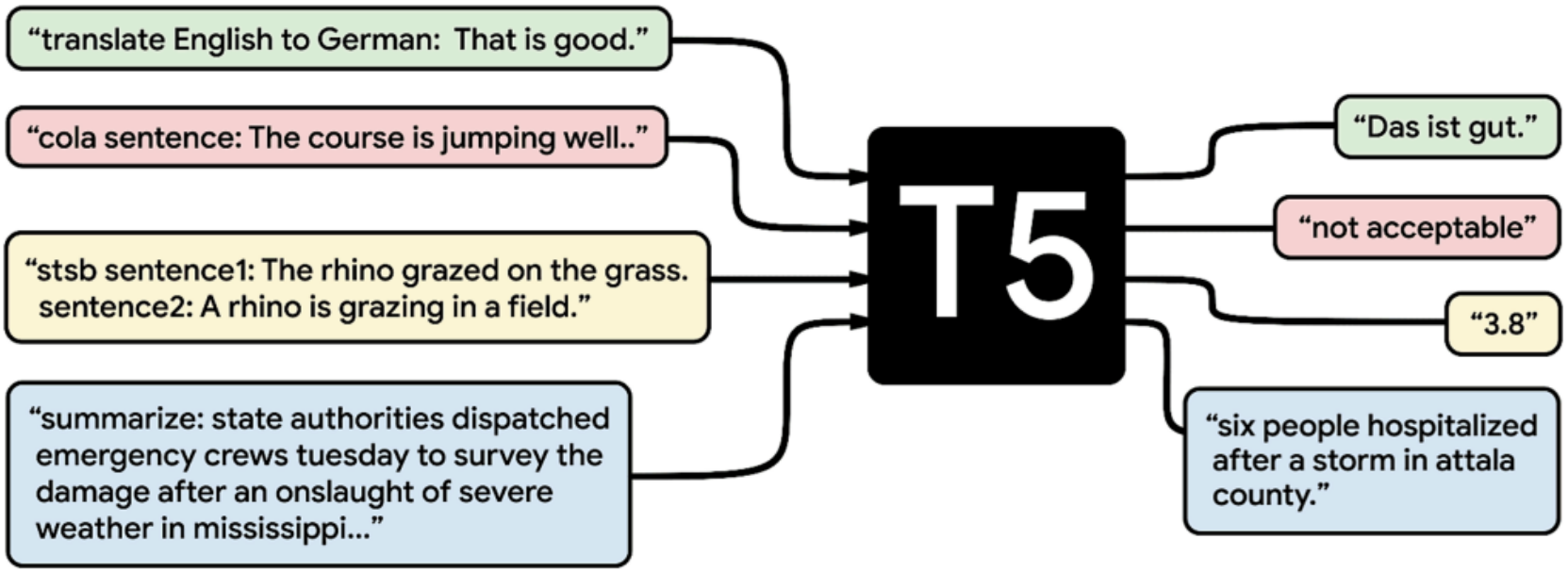
출처: 고려대 DSBA 연구실 이유경님 발표자료

KU KONKUK UNIVERSITY KAIST

T5: Transfer Learning with a Unified Text-to-Text Transformer

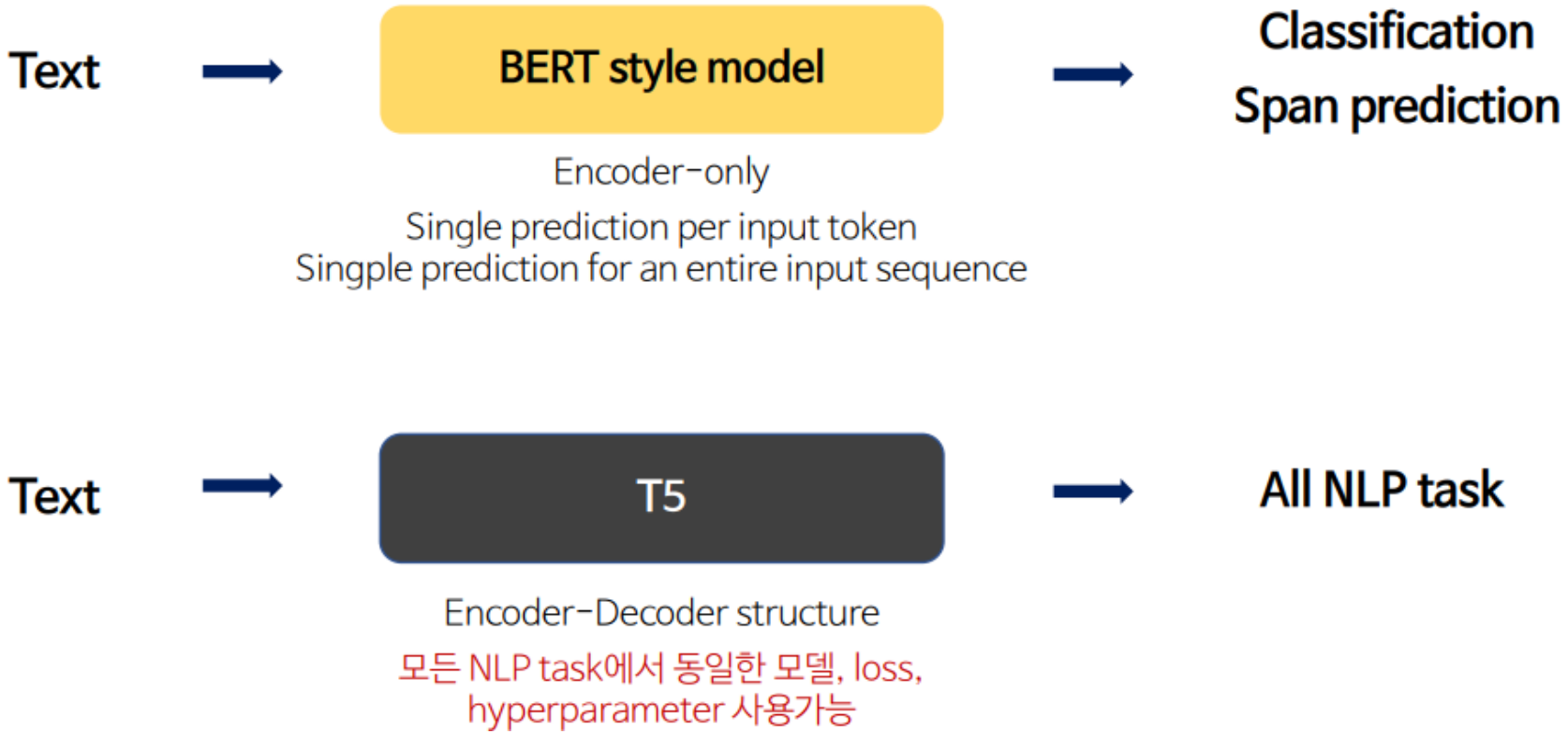
자연어처리연구실

Unified Framework



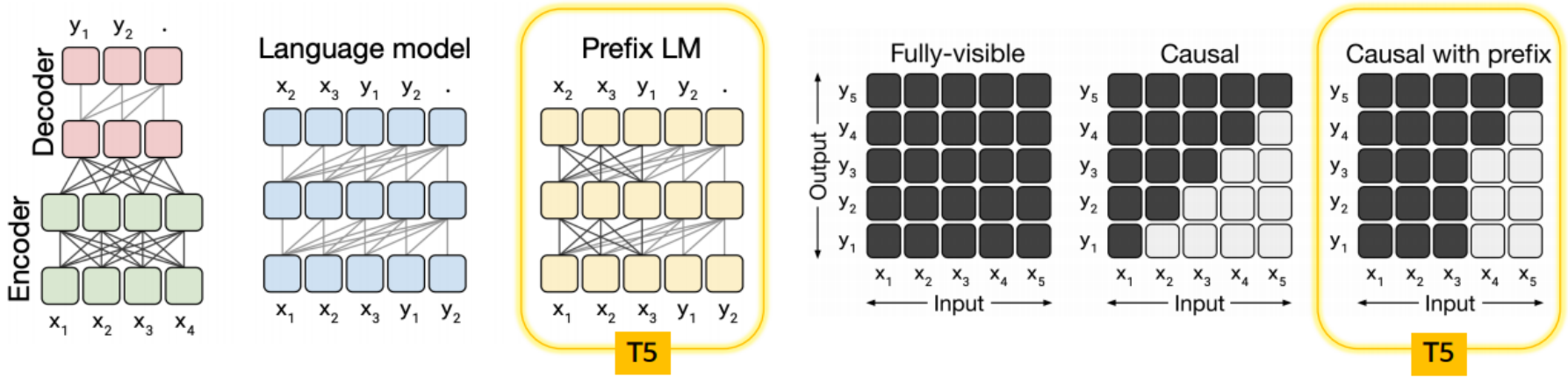
"Unified framework that converts every language problem into a text-to-text format"

Why Text-to-Text?



출처: 고려대 DSBA 연구실 이유경님 발표자료

Model Architecture



Model structure

1) Encoder - Decoder

Encoder : fed an input sequence
Decoder : produces a new output sequence

2) Language model

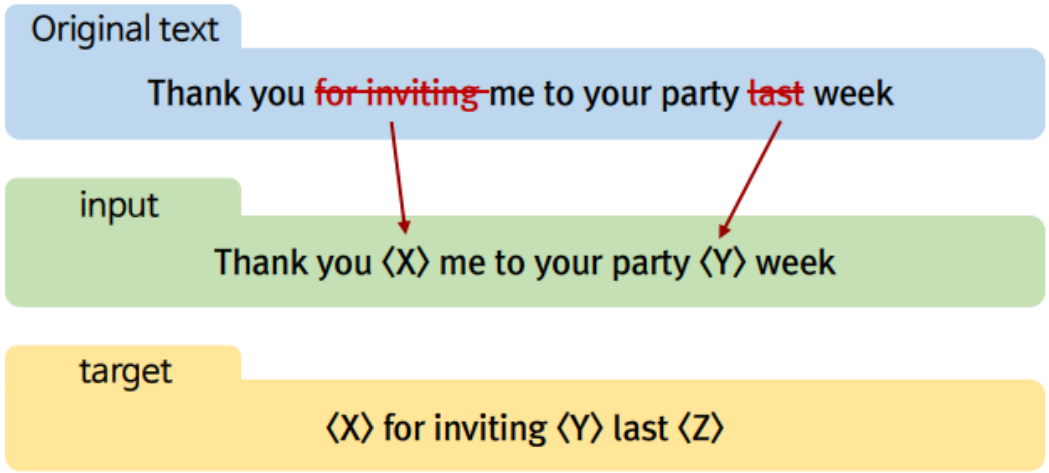
autoregressively produce an output sequence

3) Prefix Language model

Source data : bidirectional attention
Generation part : unidirectional attention

출처: 고려대 DSBA 연구실 이유경님 발표자료

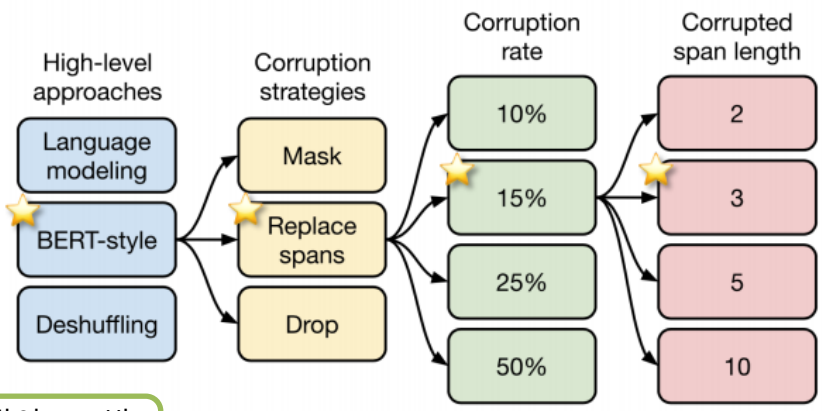
Training Objective



- MLM은 bidirectional model구조를 가짐
- BERT는 하나의 token에 masking을 하지만 T5 연속된 token을 하나의 mask로 바꿈 BART와 비슷
- Encoder-Decoder 구조로 input과 Target을 가지고 있음
- Input에서 mask 되지 않은 부분을 target에서 맞춰야 함 MASS와 비슷
- Output level에서 FFNN + Softmax을 통해 시퀀스 생성

출처: 고려대 DSBA 연구실 이유경님 발표자료

Corruption



High-level approach

Denoising 방법이 LM 방법 보다 좋음!

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style [Devlin et al., 2018]	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Corruption rate

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Corruption strategies

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style [Devlin et al., 2018]	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style [Song et al., 2019]	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Mask 대신에 random token으로 대체!

Corruption span length

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

출처: 고려대 DSBA 연구실 이유경님 발표자료

Experiments

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	**97.1**[a]	**93.6**[b]	**91.5**[b]	**92.7**[b]	**92.3**[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**89.7**	**70.8**	**97.1**	91.9	89.2	92.5	92.1

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	**74.8**[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	74.6	90.4	**92.0**	**91.7**	96.7	**92.5**	**93.2**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	88.95[d]	94.52[d]	84.6[e]	87.1[e]	90.5[e]	95.2[e]	90.6[e]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**90.06**	**95.64**	**88.9**	**91.0**	**93.0**	**96.4**	**94.8**

Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[e]	52.5[e]	90.6[e]	90.0[e]	88.2[e]	69.9[e]	89.0[e]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.2**	**62.3**	**93.3**	**92.5**	**92.5**	**76.1**	**93.8**

Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[f]	**43.8**[f]	**38.5**[g]	43.47[h]	20.30[h]	40.63[h]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

Previous best Model index

A : ALBERT (2)
B : StructBERT (5)
C : Freelb (1)
D : XLNet
E : RoBERTa
F : Understanding back-translation at scale (2)
G: Cross-lingual language model pretraining (1)
H: Unified language model pre-training for natural language understanding and generation.

Generation task

출처: 고려대 DSBA 연구실 이유경님 발표자료

Sketch of T5

1) Model Architecture

Encoder , Decoder only 모델 보다
Basic transformer 구조가 높은 성능을 보임

2) Pretraining Objectives

Pretraining에서 Noising 된 input을
Denoising하며 단어를 예측하는 방식이
가장 효율적인 방법임

3) Unlabeled datasets

Domain specific data는 task에 도움이 되지만
데이터의 크기가 작은경우 overfitting을 야기함

4) Training strategies

multitask learning이
unsupervised pre-training과 비슷한 성능 보임
학습시 task별 적절한 proportion이 필요함

5) Scaling

모델 크기를 늘리거나 ,앙상블을 시도하며 실험 진행.
작은모델을 큰 데이터로 학습하는게 효과적이라는것 발견함

6) Pushing the limits

110억개 파라미터를 가지는 모델을 훈련하여 SOTA 달성함
1 trillon 개가 넘는 token에 대해 훈련 진행함

출처: 고려대 DSBA 연구실 이유경님 발표자료

질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr