

Natural Language Processing with DeepLearning

week 11

The course

- A brief note on subword modeling
- Motivating model pretraining from word embeddings
- Model pretraining three ways
 - 1) Decoders
 - 2) Encoders
 - 3) Encoder-Decoders
- Interlude: what do we think pretraining is teaching?
- Very large models and in-context learning

1. A brief note on subword modeling

<Word structure and subword models>

단어 임베딩 단계에서 수만개의 단어로 이루어진 vocabulary를 이용해 훈련셋 만들면 모든 테스트 때 새롭게 마주한 단어는 UNK로 매핑됨

But 유한한 vocabulary 가정은 많은 언어들에서 효과적 X,
why? 대부분의 많은 언어 : 복잡한 형태/ 단어 구조, 길고 복잡한 복합어 일수록 단어는 적게 등장할 것

	word	vocab mapping	embedding
Common words	hat	→ pizza (index)	
	learn	→ tasty (index)	
Variations	taaaaasty	→ UNK (index)	
	laern	→ UNK (index)	
misspellings	Transformerify	→ UNK (index)	
novel items			

<The byte-pair encoding algorithm>

NLP의 subword 모델링: 단어 수준 기저에 깔린 구조에 대한 추론을 위한 광범위한 방법을 포함(단어 일부, 문자, 바이트)

- 현대에는 단어의 일부(하위 단어 토큰)의 vocabulary를 배움
- 훈련 및 테스트 시 각 단어는 알려진 subword 시퀀스로 분할

Byte-pair 인코딩: subword vocabulary를 정의하기 위한 간단하고 효과적인 전략

1. Character와 "end-of-word" 기호만 있는 vocabulary 시작
2. 텍스트 모음을 사용하여 가장 일반적인 인접 문자 "a,b"를 찾고 하위 단어로 "ab"를 추가
3. Character pair의 인스턴스를 새 하위 단어로 교체하고 원하는 어휘 크기까지 반복

ex) 시작 vocab: {a,b,...,z}, 완료 vocab: {a,...,z,apple,app#,#ly,...}

최근에는 사전 학습된 모델에서 유사한 방법인 WordPiece 등을 사용

=> 결과적으로 자주 등장하는 단어는 그 자체로 subword vocabulary가 되지만 자주 등장하지 않는 단어는 세부 컴포넌트로 나뉨, 최악의 경우, 단어가 가지고 있는 글자들로

전부 나눔

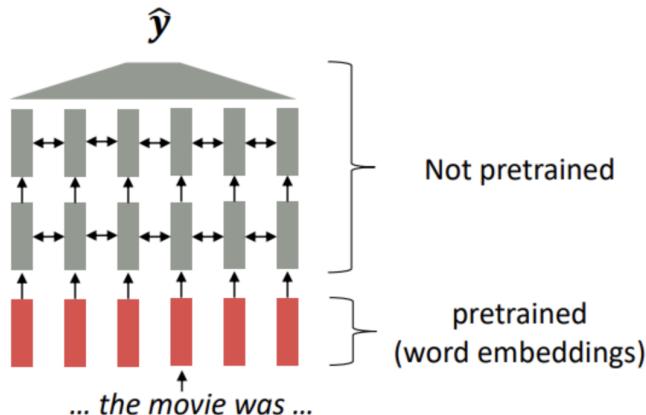
	word	vocab mapping	embedding
Common words	hat	→ hat	
	learn	→ learn	
Variations	taaaaasty	→ taaa## aaa## sty	
	laern	→ la## ern##	
misspellings			
novel items	Transformerify	→ Transformer## ify	

2. Motivating word meaning context

<pretrained word embeddings>

2017)

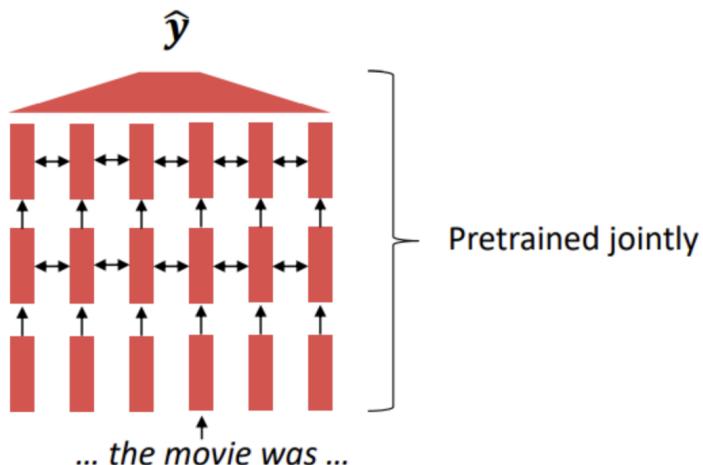
- 사전 훈련된 단어 임베딩으로 시작(문맥 고려하지 않음).
- Task에 대해 훈련하는 동안 LSTM 또는 Transformer에 문맥적 의미를 같이 학습.
- 주의) 학습데이터는 언어의 모든 맥락적 측면을 가르치기에 충분할 정도로 많아야 함



현대 NLP)

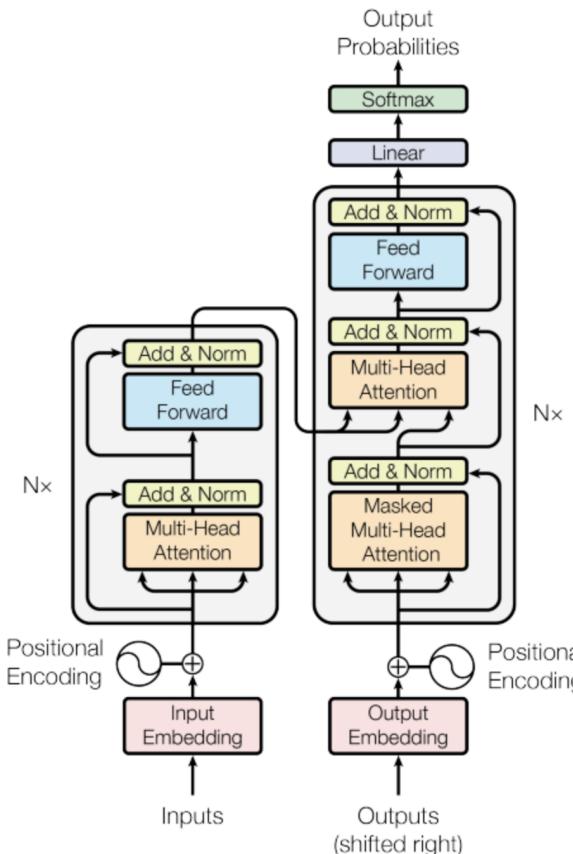
- NLP 네트워크의 모든(또는 거의 모든) 매개변수는 사전 훈련을 통해 초기화된다.
- 사전 훈련 방법은 모델에서 입력의 일부를 숨기고 해당 부분을 재구성하도록 모델을 훈련한다.

예를 들어 skip-gram처럼 주변을 숨기고 단어를 통해 예측하게 한다던가



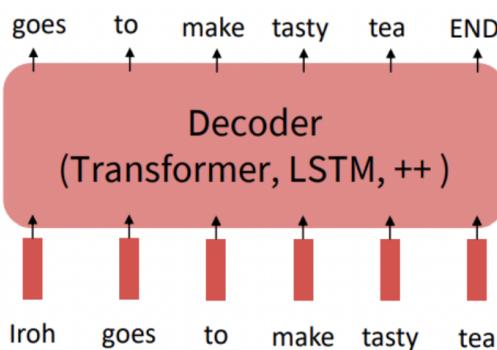
사전 학습된 모델은 강력한 효과를 보여주었음=> 전체 시퀀스에 대해 학습하기 때문에 더욱 강력한 언어 표현을 가짐, 출력층을 제외한 모델 내의 모든 매개변수들이 유기적으로 얹혀있는 상태로 잘 초기화 됨, 다음 단어를 예측하는 언어 모델과 같은 작업에 효과적임 => 모델은 사전학습에서 문장들의 빈칸을 학습하게 됨

<The Transformer Encoder-Decoder>



<Pretraining through language modeling>

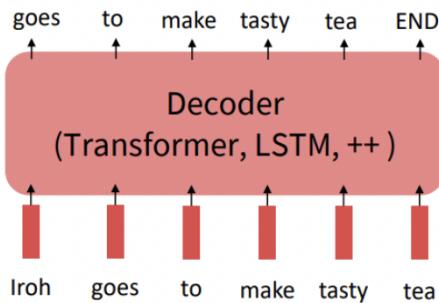
: 언어모델 task는 과거 문맥을 통해 다음 단어의 확률을 계산



=> 사전 학습: 이전에 언어 모델을 학습하던 것과 차이 X, 단순히 decoder가 transformer가 되고 많은 데이터에 대해 학습 후 매개변수들을 저장하는 것

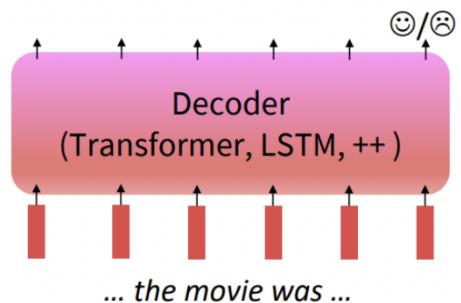
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



감성분류 모델 학습 예시

<Stochastic gradient descent and pretrain/finetune>

- 왜 사전학습 후 finetuning하는 것이 도움될까?

사전학습이 주는 효과는 1. local minima가 세타 햄 근처에 있어 일반화가 잘 됐다거나,
2. 그리고/또는 finetuning 손실의 기울기가 잘 전파된 것

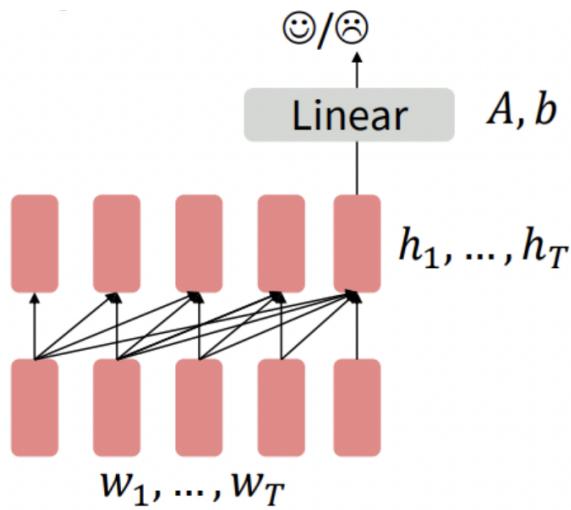
3. Pretraining for three types of architectures

Neural architecture는 downstream task와 사전 학습 종류에 영향을 받으며 세 가지 종류로 구분

- 디코더 (Decoders)
 - 지금까지 본 것과 같은 언어모델(LM)
 - 생성하기 좋으며, 미래를 볼 수 없음
- 인코더 (Encoders)
 - bidirectional LSTM이 one-directional 보다 성능이 좋았던 것과 같이 양방향 문맥을 학습
- 인코더-디코더 (Encoder-Decoder)
 - 위 두 방식에서 어떤 장점을 가져올 수 있을까?

<Pretraining decoders>

언어 모델로 사전학습된 디코더를 사용할 때, 사전 모델 $P_\theta(w_t | w_{1:t-1})$ 이 예측하도록 학습됐던 것을 무시한다. 이후 감성분류 작업을 한다면 분류기(Linear)를 마지막 단어의 은닉 상태(hidden state) 위에 올려서 finetuning을 학습한다. 당연히 분류기는 학습되지 않았기 때문에 해당 층의 매개 변수(A,b)는 임의로 초기화되며, 분류 결과의 손실을 통해서 전체 네트워크가 학습됨

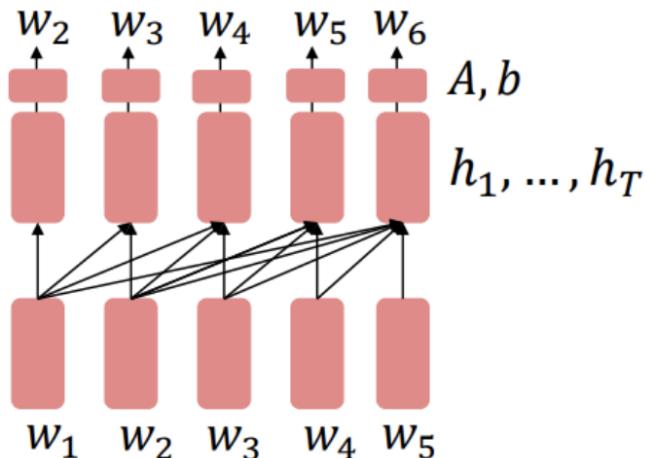


사전학습 된 디코더를 이용한 감성분류

디코더로부터 hidden state 연산: $h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$

분류기에서 연산: $y \sim Ah_t + b$

디코더를 언어 모델로 사전학습하는 것이 자연스럽기에 $P_\theta(w_t | w_{1:t-1})$ 로 finetuning하여 generator로 사용해보자. 출력은 사전학습한 vocabulary의 시퀀스이기 때문에 대화(context=dialogue history)나 요약(context=document) 같은 작업에 도움됨. 감성 분류와의 차이점은 사전학습 단계에서 Linear가 학습되는 것



디코더로부터 hidden state 연산: $h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$

분류기에서 연산: $w_t \sim Ah_{t-1} + b$

<Generative Pretrained Transformer (GPT)>

2018년 GPT는 사전학습된 디코더로 좋은 성능을 보였다

- 12개 층의 Transformer
- 768차원 hidden states, 3072차원 feed-forward hidden layers
- 40000까지 Byte-pair 인코딩 (굉장히 작은 크기)

- BookCorpus로 사전 학습: 7000개의 책
 - Long-distance dependencies를 학습하기 위해 긴 span의 연속적인 텍스트가 포함되어 있습니다.
- "GPT"라는 단어는 실제 논문에서 사용된 단어가 아닙니다. "Generative PreTraining" 또는 "Generative Pretrained Transformer"로 설명됨
=> 어떻게 해당 모델을 finetuning 작업에 맞게 입력을 변경했을까?
Natural Language Inference: 문장들을 entailing/contradictory/neutral로 라벨링

Premise: *The man is in the doorway* }
 Hypothesis: *The person is near the door* } entailment

위의 예시 문장을 모델에 입력으로 넣어주기 위해 특별한 토큰 사용

[START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

[DELIM]으로 두 문장을 구분하였으며, Linear 분류기는 [EXTRACT] 토큰의 representation에 적용됨

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	<u>82.1</u>	61.7
Finetuned Transformer LM (ours)	82.1	81.4	89.9	88.3	88.1	56.0

Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

<GPT2>

우리는 사전 훈련된 디코더가 언어 모델로서의 능력에서 어떻게 사용될 수 있는지 언급, 많은 데이터로 훈련된 GPT의 더 큰 버전인 GPT-2는 상대적으로 설득력 있는 자연어 샘플을 생성하는 것으로 나타남

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

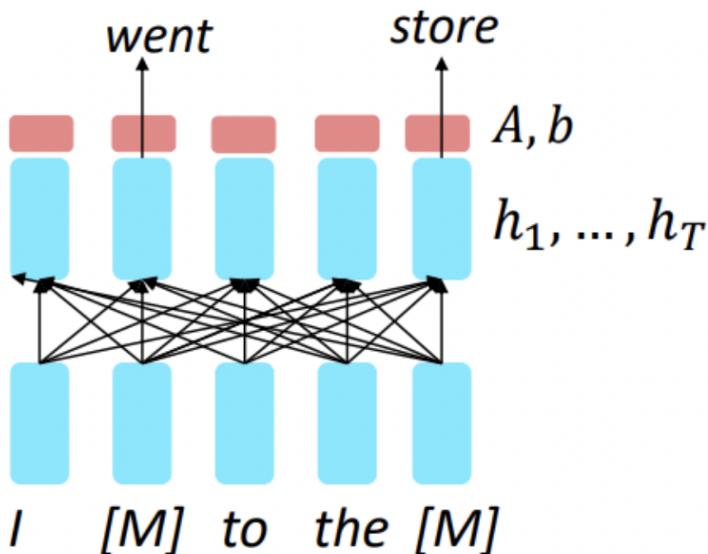
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

<Pretraining encoders>

: 인코더는 bidirectional context를 고려하기 때문에 언어 모델 방식을 적용할 수 없음=>
따라서 입력의 일부를 [Mask] 토큰으로 변경하여 이를 예측



인코더로부터 hidden state 연산: $h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$

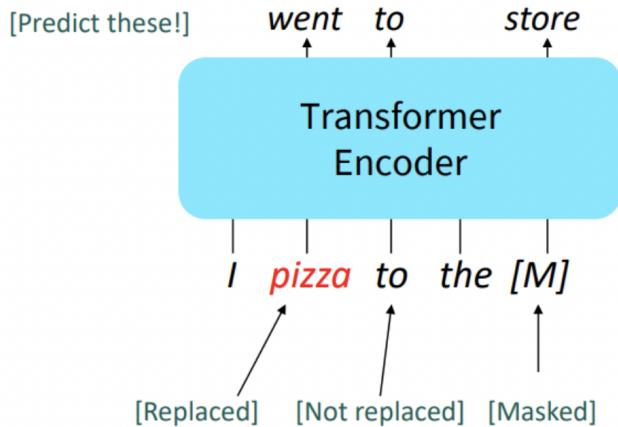
분류기에서 연산: $y_i \sim Aw_i + b$

손실은 "masked out"된 단어에 대해서만 수행하며, x hat이 x 가 mask된 것이라면 모델은

$p_\theta(x|\tilde{x})$ 를 학습하고 이를 Masked LM이라고 함.

2018은 "Masked LM" 목표를 제안하고 BERT라는 이름을 붙인 모델인 사전 훈련된 Transformer의 가중치를 발표

다만 위와 같이 학습을 진행할 경우 [Mask]만 잘 예측하려고하고 나머지 단어에 대해서는 representation을 잘 구축하려고하지 않기 때문에 다음과 같은 세부적인 기법들이 추가됨



- 임의의 (sub)word 토큰들의 15%를 예측
: 80:10:10의 비율로 [Mask], 임의의 토큰, 변화없음(그러나 여전히 예측은 수행)
변화를 줌

finetuning 시에는 [Mask] 토큰이 보이지 않을 것이므로, 모델이 mask되지 않은 단어에 대해서도 강력한 표현을 만드는 것이 목적

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	# # ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[\text{SEP}]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#\text{ing}}$	$E_{[\text{SEP}]}$
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

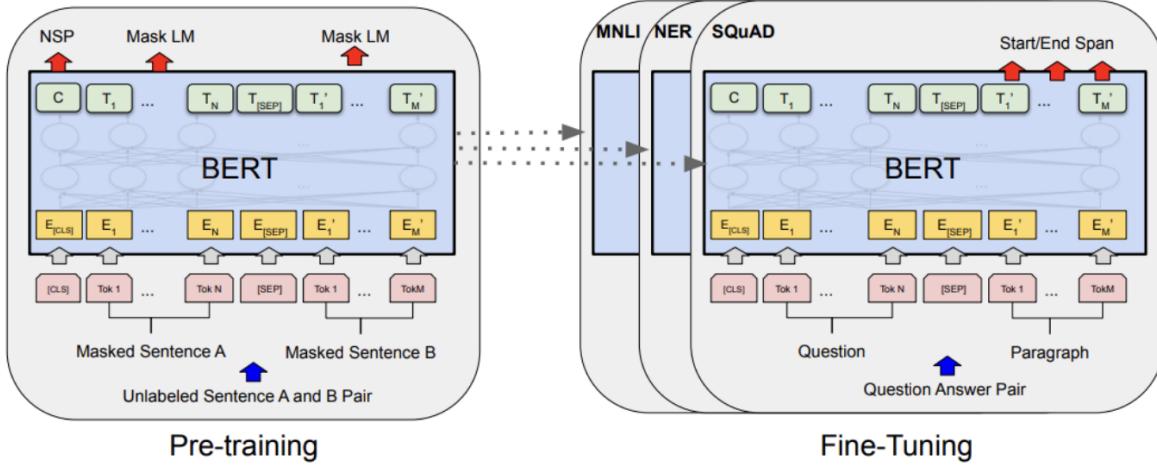
BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

추가적으로 BERT의 사전 학습에 사용된 기법은 입력이 두 분리된 텍스트의 둉어리로 들어감.

Downstream 작업에서 QA와 같이 문장의 관계를 유추하는 것이 진행될 경우를 문장 간의 관계를 잘 파악하기 위함이다.

두 문장은 보통은 텍스트에서 연속적인 것을 가져오지만 때때로는 임의 추출하여 상관없는 문장들을 넣어줌, 모델은 두 문장의 관계를 예측해야 하며, 이를 "next sentence prediction"이라고 함

이 후에 "next sentence prediction"은 필수적이지 않다고 했으며, 차라리 두 배 긴 문장을 넣어주는 것이 긴 문장의 문맥을 파악하는 것에 도움된다는 연구 결과가 있음



Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different downstream tasks. During fine-tuning, all parameters are fine-tuned. **[CLS]** is a special symbol added in front of every input example, and **[SEP]** is a special separator token (e.g. separating questions/answers).

BERT의 개요

- BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params
- BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params
- 학습
 - BooksCorpus (800 million words)
 - English Wikipedia (2,500 million words)
- 사전 학습은 단일 GPU로 수행하기에 무리
: BERT는 64TPU로 총 4일간 사전 학습되었음. (TPU는 tensor 연산에 최적화 된 하드웨어이다)
- Finetuning은 단일 GPU에서 수행이 가능하다 (hugging face에서 편리하게 사용할 수 있도록 배포한다)
: pretraining보다 적은 수의 배치로 학습을 진행하기 때문

BERT는 엄청난 인기를 얻었고 매우 다재다능했다. BERT를 finetuning하여 광범위한 작업에서 새로운 SOTA를 얻었다.

- QQP: Quora Question Pairs (detect paraphrase questions)
- QNLI: natural language inference over question answering data
- SST-2: sentiment analysis
- CoLA: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- STS-B: semantic textual similarity
- MRPC: microsoft paraphrase corpus
- RTE: a small natural language inference corpus

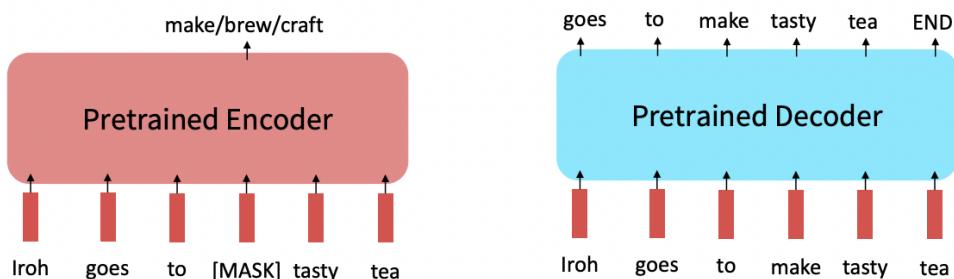
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

GLUE Test results, scored by the evaluation server (<https://gluebenchmark.com/leaderboard>). The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.8 BERT and OpenAI GPT are single model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Limitations of pretrained encoders

Those results looked great! Why not used pretrained encoders for everything?

If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice autoregressive (1-word-at-a-time) generation methods.

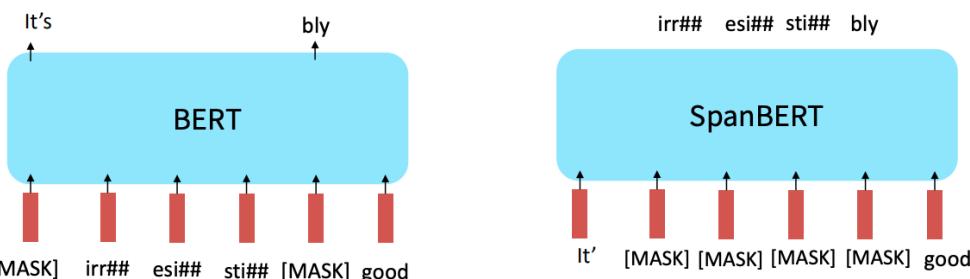


Extensions of BERT

You'll see a lot of BERT variants like RoBERTa, SpanBERT, +++

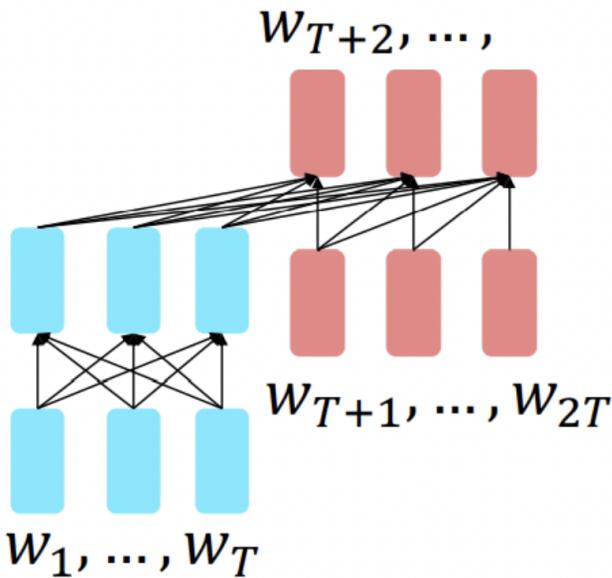
Some generally accepted improvements to the BERT pretraining formula:

- RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
- SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task



<Pretraining encoders-decoders>

인코더-디코더의 경우 언어 모델과 유사한 작업을 하지만 모든 입력의 prefix가 인코더에 제공되고 예측에 사용되지 않음



인코더로부터 hidden state 연산: $h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$

디코더로부터 hidden state 연산: $h_{T+1}, \dots, h_{2T} = \text{Decoder}(w_1, \dots, w_T, h_1, \dots, h_T)$

분류기에서 연산: $y_i \sim Aw_i + b, i > T$

=> 결과적으로 인코더 부분은 bidirectional context의 이점을 얻음. 디코더 부분은 언어 모델링을 통해 전체 모델을 훈련하는 데 사용

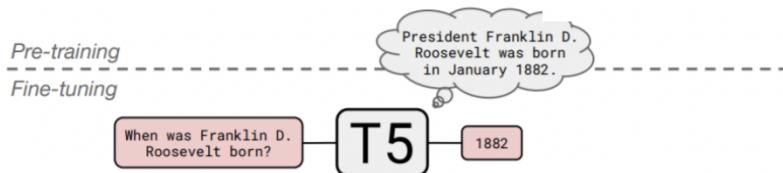
Pretraining encoder-decoders: what pretraining objective to use?

[Raffel et al., 2018](#) found encoder-decoders to work better than decoders for their tasks, and span corruption (denoising) to work better than language modeling.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	2P	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	Denoising	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	Denoising	P	M/2	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	2P	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	M/2	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Pretraining encoder-decoders: what pretraining objective to use?

A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.



NQ: Natural Questions

WQ: WebQuestions

TQA: Trivia QA

All “open-domain” versions

	NQ	WQ	TQA	
			dev	test
Karpukhin et al. (2020)	41.5	42.4	57.9	—
T5.1.1-Base	25.7	28.2	24.2	30.6
T5.1.1-Large	27.3	29.5	28.5	37.2
T5.1.1-XL	29.5	32.4	36.0	45.1
T5.1.1-XXL	32.8	35.6	42.9	52.5
T5.1.1-XXL + SSM	35.2	42.8	51.9	61.6

Doffel et al. 20101

4. Interlude : what do we think pretraining is teaching?

What kinds of things does pretraining learn?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language. Taking our examples from the start of class:

- Stanford University is located in _____, California. [Trivia]
- I put _____ fork down on the table. [syntax]
- The woman walked across the street, checking for traffic over _____ shoulder. [coreference]
- I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was _____. [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____. [some basic arithmetic; they don't learn the Fibonacci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.
- More on all this in the interpretability lecture!

5. Very large models in-context learning

GPT-3, In-context learning, and very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

GPT-3 has 175 billion parameters.

GPT-3, In-context learning, and very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

" thanks -> merci
hello -> bonjour
mint -> menthe
otter -> "

Output (conditional generations):

loutre..."

GPT-3, In-context learning, and very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

