Natural Language Processing: Pretraining

2024/10/29

Department of Computer Science
Hanyang University

Taeuk Kim

Motivating Model Pretraining From Word Embeddings

Motivating Word Meaning and Context

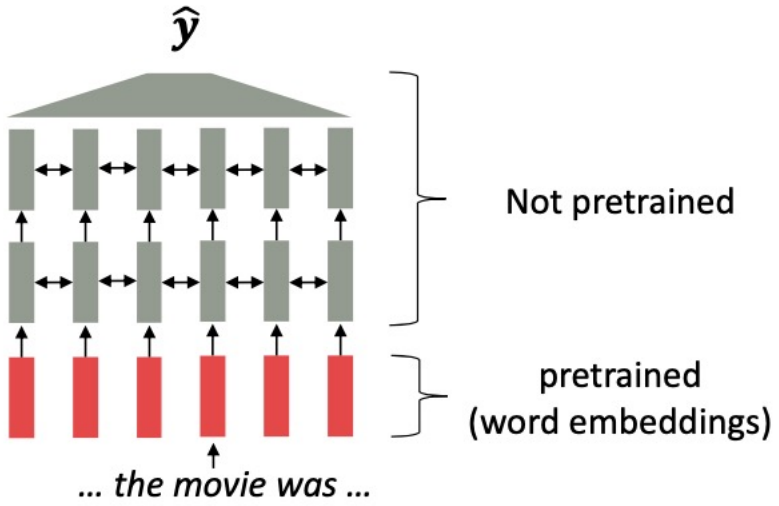
- **Recall the adage we mentioned at the beginning of the course:**
  - "You shall know a word by the company it keeps" (J. R. Firth 1957: 11)

- **This quote is a summary of distributional semantics, and motivated word2vec. But:**
  - "... the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously." (J. R. Firth 1935)
  - Consider I record the record: the two instances of record mean different things.

Where We Were: Pretrained Word Embeddings

- **Circa 2015:**
  - Start with pretrained word embeddings (no context!)
  - Learn how to incorporate context in an LSTM or Transformer while training on the task.

- **Some issues to think about:**
  - The training data we have for our **downstream task** (like question answering) must be sufficient to teach all contextual aspects of language.
  - Most of the parameters in our network are randomly initialized!

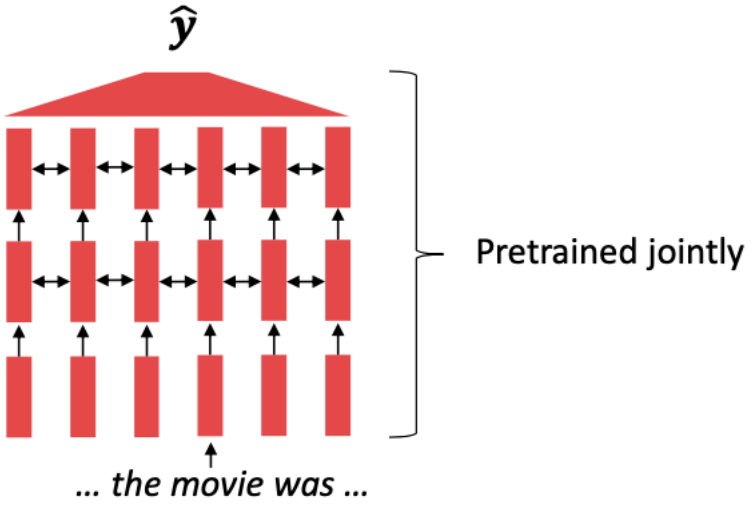


[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

Where We Are Going: Pretraining Whole Models

- **In modern NLP:**
  - All (or almost all) parameters in NLP networks are initialized via **pretraining**.
  - Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.

- **This has been exceptionally effective at building strong:**
  - **Representations of language.**
  - **Parameter initializations** for strong NLP models.
  - **Probability distributions** over language that we can sample from.



[This model has learned how to represent entire sentences through pretraining]

What Can We Learn from Reconstructing the Input?

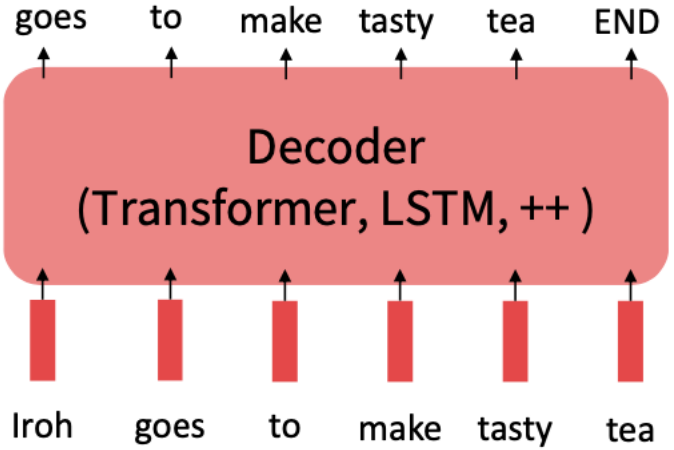
- **What kinds of things does pretraining teach?**
  - There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language.

  - Hanyang University is located in _____, Seoul, South Korea. **[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 may or may not learn the Fibonnaci sequence]**

Pretraining through Language Modeling

- **Recall the language modeling task:**
  - Model $p_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
  - There's lots of data for this (in English.)

- **Pretraining through language modeling:**
  - Train a neural network to perform language modeling on a large amount of text.
  - Save the network parameters.

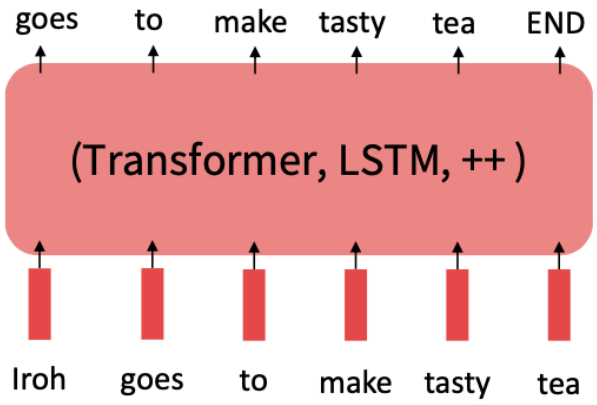


The Pretraining / Finetuning Paradigm

- **Pretraining can improve NLP applications by serving as parameter initialization.**

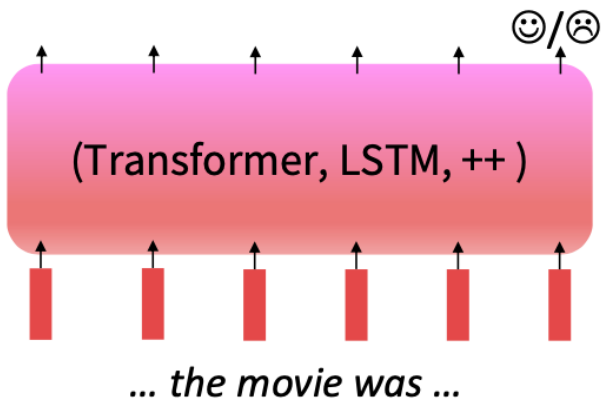
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!

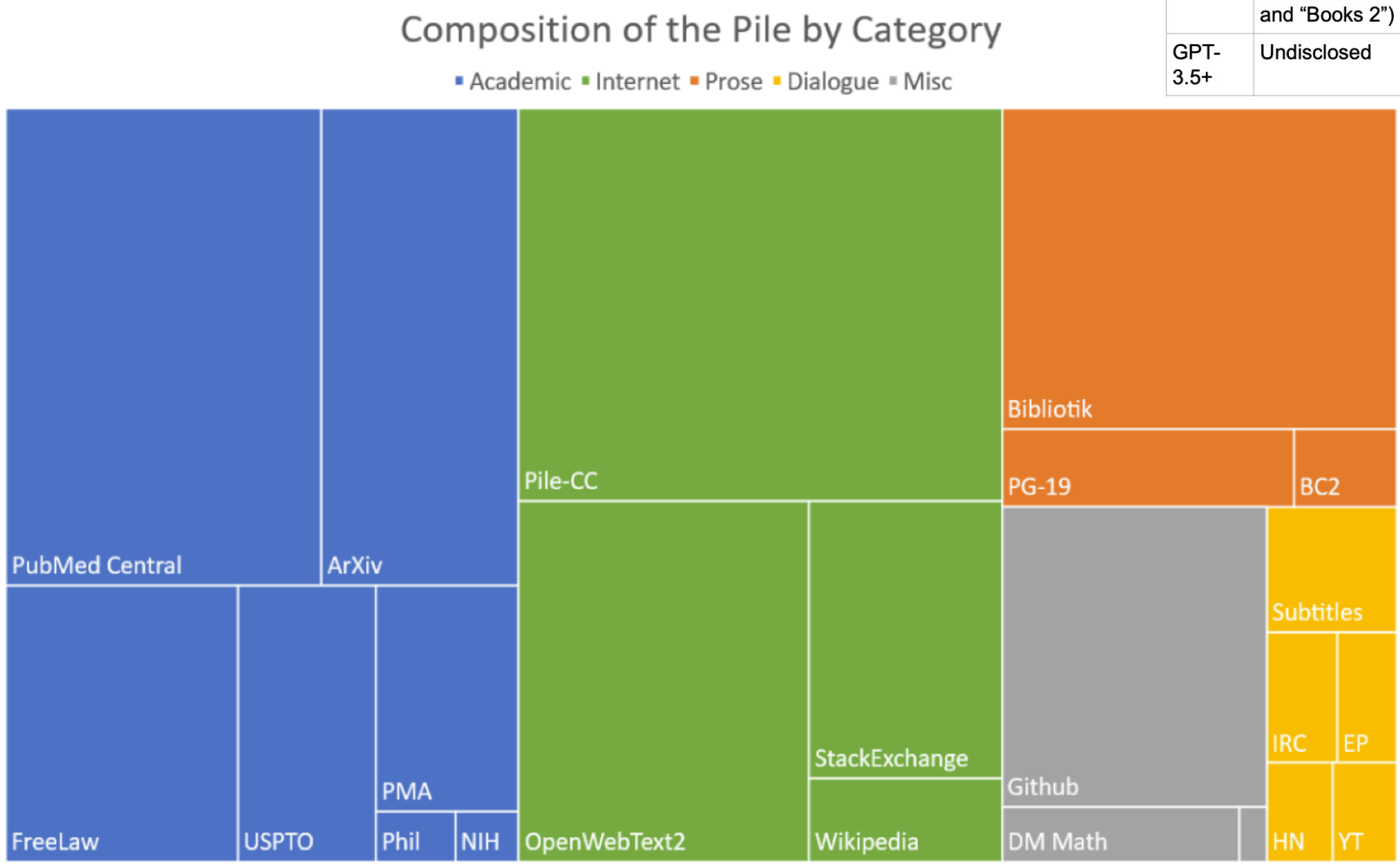


Stochastic Gradient Descent and Pretrain/Finetune

- **Why should pretraining and finetuning help, from a "training neural nets" perspective?**
  - Consider, provides parameters $\hat{\theta}$ by approximating $\min_{\theta} \mathcal{L}_{\text{pretrain}}(\theta)$ (the pretraining loss).
  - Finetuning approximates $\min_{\theta} \mathcal{L}_{\text{finetune}}(\theta)$ (the finetuning loss), starting at $\hat{\theta}$.
  - The pretraining may matter because stochastic gradient descent sticks (relatively) close to $\hat{\theta}$ during finetuning.
    - So, maybe the finetuning local minima near $\hat{\theta}$ tend to generalize well!
    - And/or, maybe the gradients of finetuning loss near $\hat{\theta}$ propagate nicely!

Where Does This Data Come From?

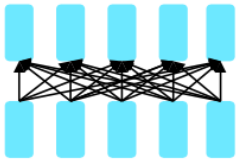
Model	Training Data
BERT	BookCorpus, English Wikipedia
GPT-1	BookCorpus
GPT-3	CommonCrawl, WebText, English Wikipedia, and 2 book databases (“Books 1” and “Books 2”)
GPT-3.5+	Undisclosed



Three Ways of Model Pretraining

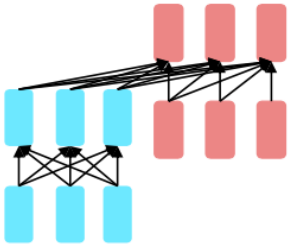
Pretraining for Three Types of Architectures

- **The neural architecture influences the type of pretraining, and natural use cases.**



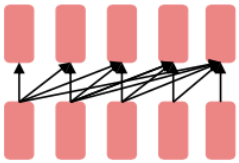
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

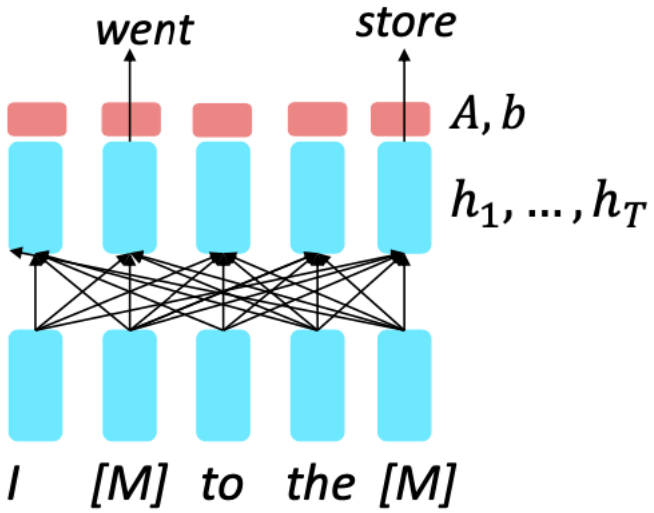
Pretraining Encoders: What Pretraining Objective to Use?

- **So far, we've looked at language model pretraining. But encoders get bidirectional context, so we can't do language modeling!**

- **Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.**

$$h_1, \dots, h_T = Encoder(w_1, \cdots, w_T)$$
$$y_i \sim Ah_i + b$$

- Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$. Called **Masked LM**.

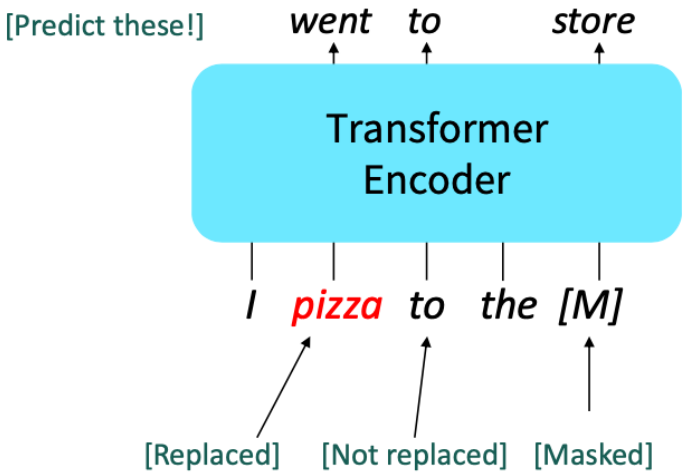


[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

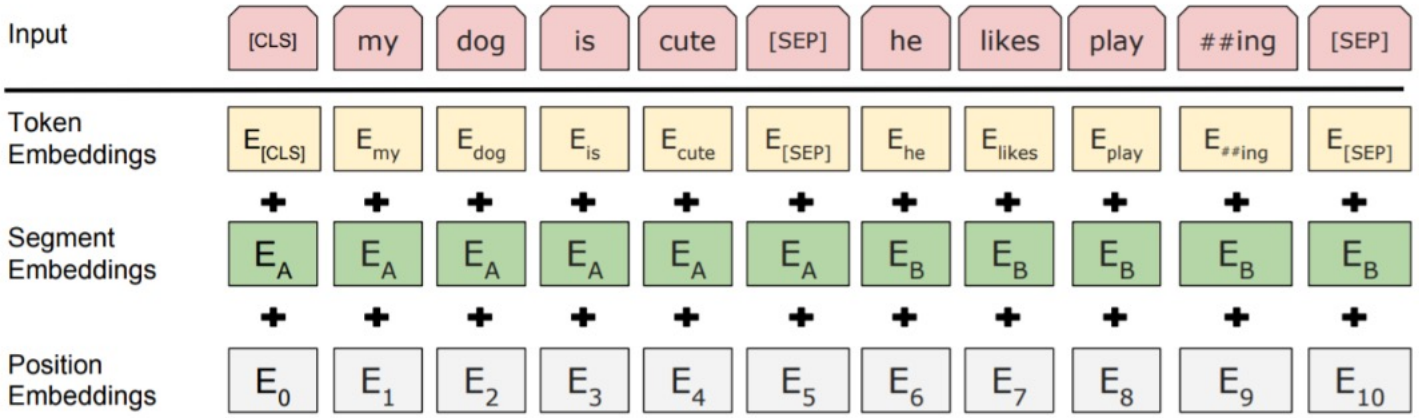
- **Devlin et al., 2018 proposed the "Masked LM" objective and released the weights of a pretrained Transformer, a model they labeled BERT.**

- **Some more details about Masked LM for BERT:**
  - Predict a random 15% of (sub)word tokens.
    - Replace input word with [MASK] 80% of the time.
    - Replace input word with a random token 10% of the time.
    - Leave input word unchanged 10% of the time (but still predict it!)
  - Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



BERT: Bidirectional Encoder Representations from Transformers

- **The pretraining input to BERT was two separate contiguous chunks of text:**



- **BERT was trained to predict whether one chunk follows the other or is randomly sampled.**
  - Later work has argued this “next sentence prediction” is not necessary.

BERT: Bidirectional Encoder Representations from Transformers

- **Two models were released:**
  - **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.

- **Trained on:**
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)

- **Pretraining is expensive and impractical on a single GPU.**
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)

- **Finetuning is practical and common on a single GPU.**
  - “Pretrain once, finetune many times.”

BERT: Bidirectional Encoder Representations from Transformers

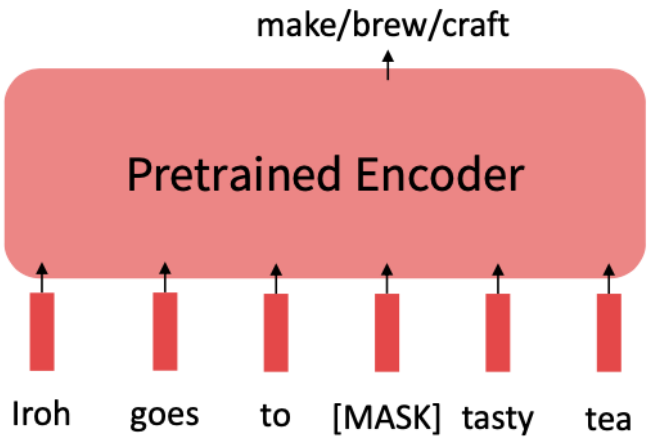
- **BERT was massively popular and hugely versatile; finetuning BERT led to new state-of- the-art results on a broad range of tasks.**
  - The GLUE benchmark (https://gluebenchmark.com)
  - General Language Understanding Evaluation (GLUE)

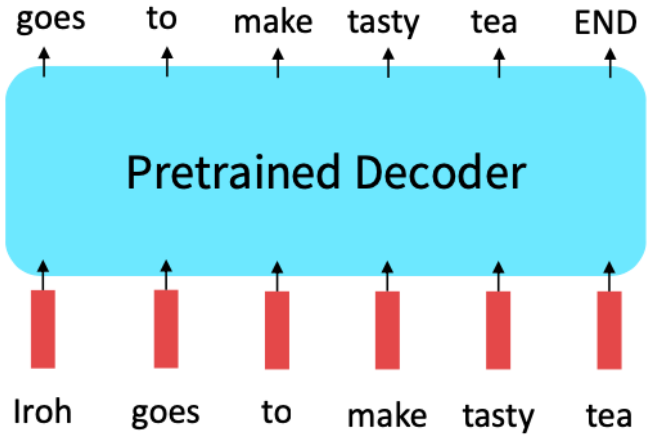
- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: 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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

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.

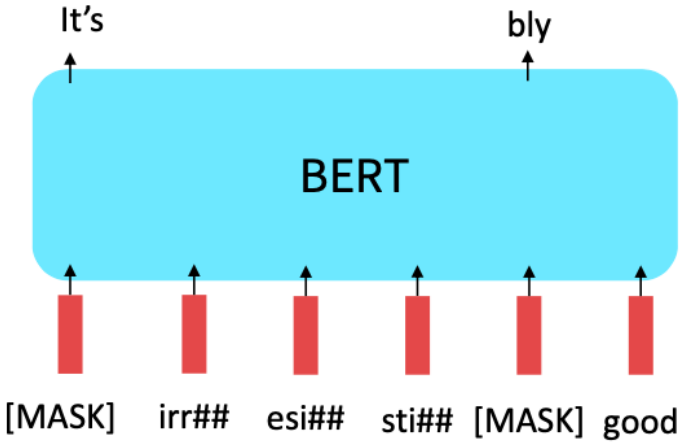


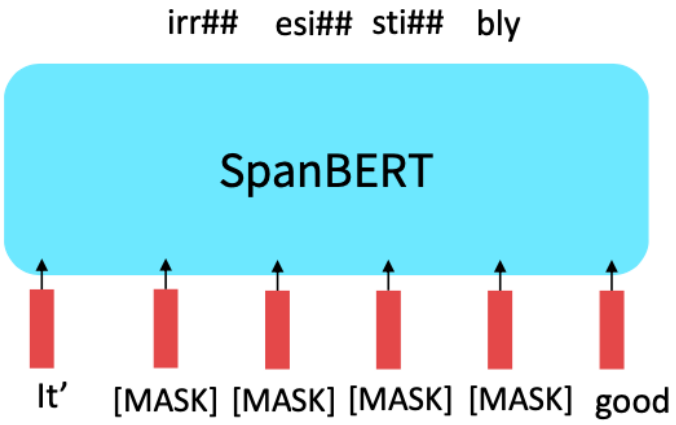


Extensions of BERT

- **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.





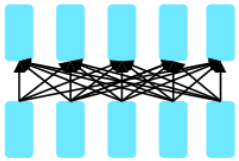
Extensions of BERT

- **A takeaway from the RoBERTa paper:**
  - More compute, more data can improve pretraining even when not changing the underlying Transformer encoder.

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

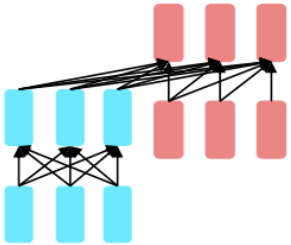
Pretraining for Three Types of Architectures

- **The neural architecture influences the type of pretraining, and natural use cases.**



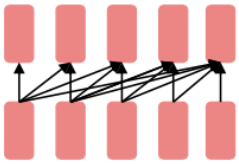
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



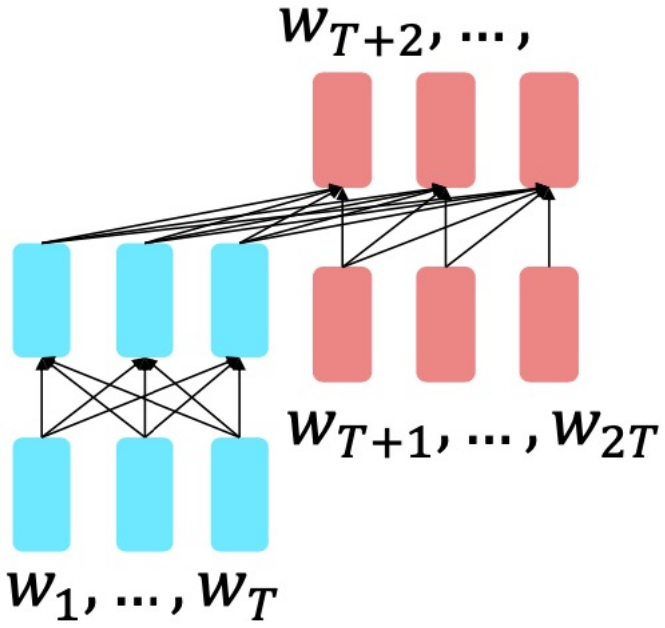
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pretraining Encoder-Decoders: What Pretraining Objective to Use?

- **For encoder-decoders, we could do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted.**
  - The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = Decoder(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$



Pretraining Encoder-Decoders: What Pretraining Objective to Use?

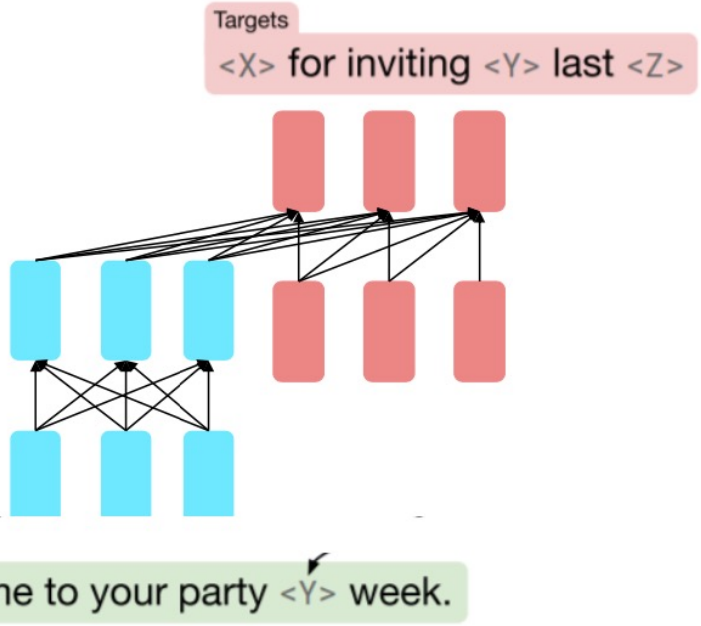
- **What Raffel et al., 2018 found to work best was span corruption.**
  - Their model: **T5**.

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

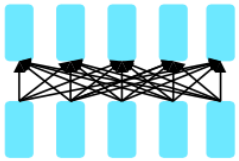
Thank you for inviting me to your party last week.

This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.



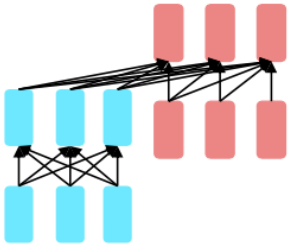
Pretraining for Three Types of Architectures

- **The neural architecture influences the type of pretraining, and natural use cases.**



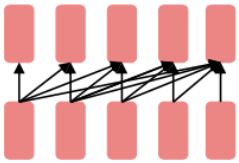
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

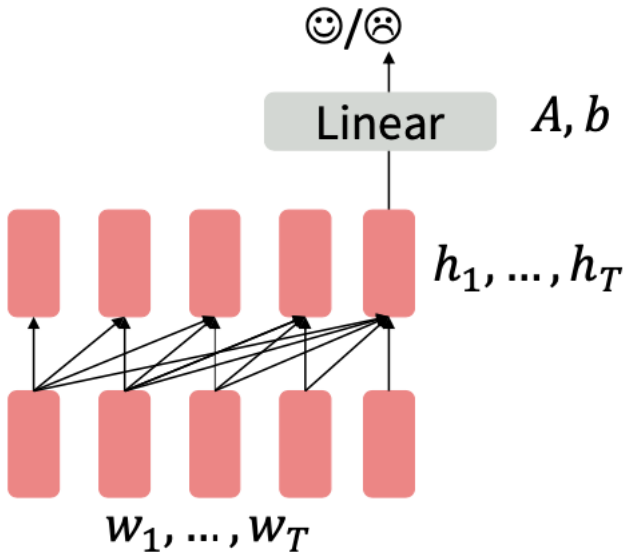


Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pretraining Decoders

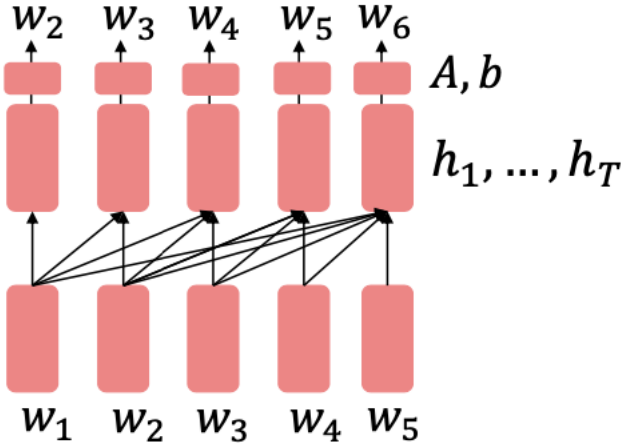
- **When using language model pretrained decoders, we can ignore that they were trained to model $p(w_t|w_{1:t-1})$.**
  - We can finetune them by training a softmax clasifier on the last word's hidden state.
  - $h_1, \ldots, h_T = Decoder(w_1, \ldots, w_T)$
    $y \sim Ah_T + b$
  - Where A and b are randomly initialized and specified by the downstream task.
  - Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

Pretraining Decoders

- **It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!**
  - This is helpful in tasks **where the output is a sqeunce** with a vocabulary like that at pretraining time!
    - Dialogue (context=dialogue history)
    - Summarization (context=document)
  - $h_1, \ldots, h_T = Decoder(w_1, \ldots, w_T)$
    $w_t \sim A h_{t-1} + b$
  - Where A, b were pretrained in the language model.



[Note how the linear layer has been pretrained.]

Generative Pretrained Transformer (GPT)

- **2018's GPT was a big success in pretraining a decoder!**
  - Transformer decoder with 12 layers, 117M parameters.
  - 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
  - Byte-pair encoding with 40,000 merges.
  - Trained on BooksCorpus: over 7,000 unique books.
    - Contains long spans of contiguous text, for learning long-distance dependencies.
  - The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer".

Increasingly Convincing Generations (GPT2)

- **We mentioned how pretrained decoders can be used in their capacities as language models.**
  - **GPT-2**, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

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.

GPT-2 Language Model Output (2019)

PROMPT (HUMAN-WRITTEN)	*A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.*
MODEL COMPLETION	The incident occurred on the downtown train line, which runs from Covington and Ashland stations. In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief. "The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation," said Tom Hicks, the U.S. Energy Secretary, in a statement. "Our top priority is to secure the theft and ensure it doesn't happen again." The stolen material was taken from the University of Cincinnati's Research Triangle Park nuclear research site, according to a news release from Department officials.

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.

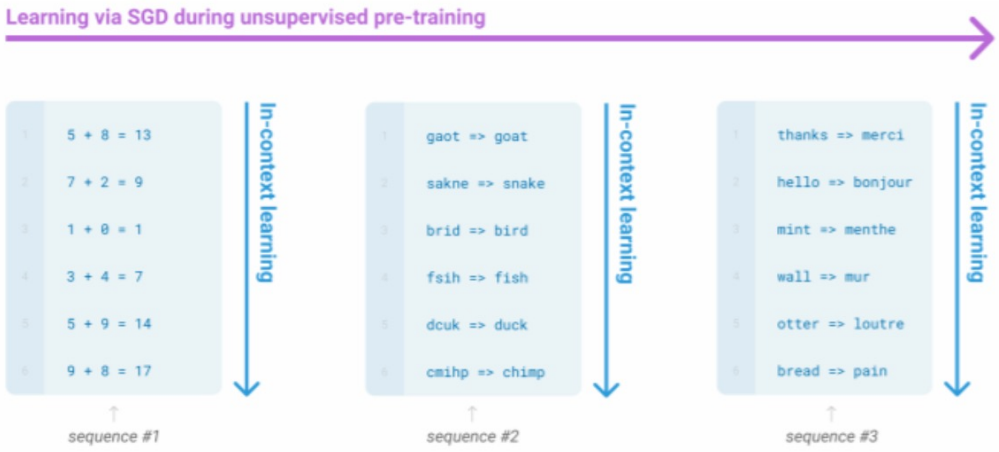
- **Very large language models seem to perform some kind of learning without gradient steps simply from examples you provide within their contexts.**
  - **GPT-3** is the canonical example of this. The largest T5 model had 11 billion parameters.
  - **GPT-3 has 175 billion parameters.**

- **ChatGPT/GPT-4/GPT-3.5 Turbo introduced a further instruction-tuning idea that we cover next lecture.**

GPT-3, In-Context Learning, and Very Large Models

- **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..."



Q & A