Pretrained Transformers

Pawan Goyal

CSE, IIT Kharagpur

CS60010

Initial days: pretrained word embeddings

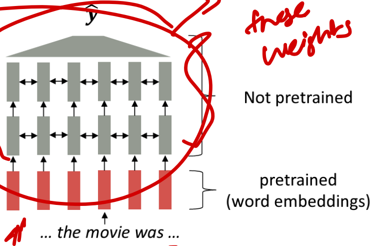
Circa 2017:

- 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!

learn all these weights



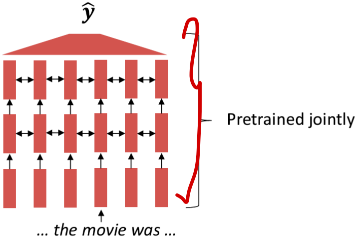
[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

ELMo

Now: 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]

self-supervised

What can we learn from reconstructing the input?

- *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 Fibonnaci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

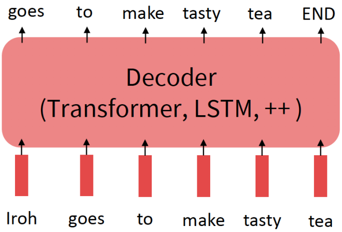
Pretraining through Language Modeling - General Paradigm

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

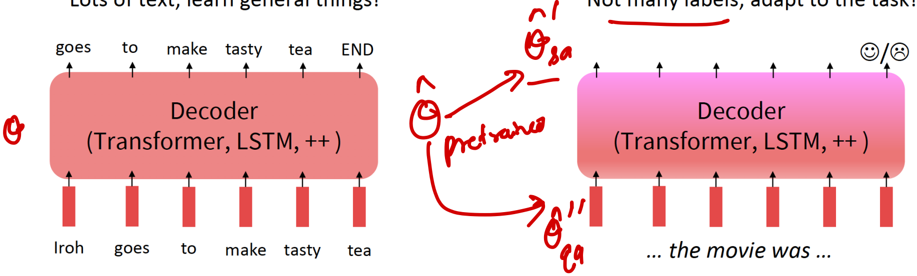
Pretraining can improve NLP applications by serving as parameter initialization.

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

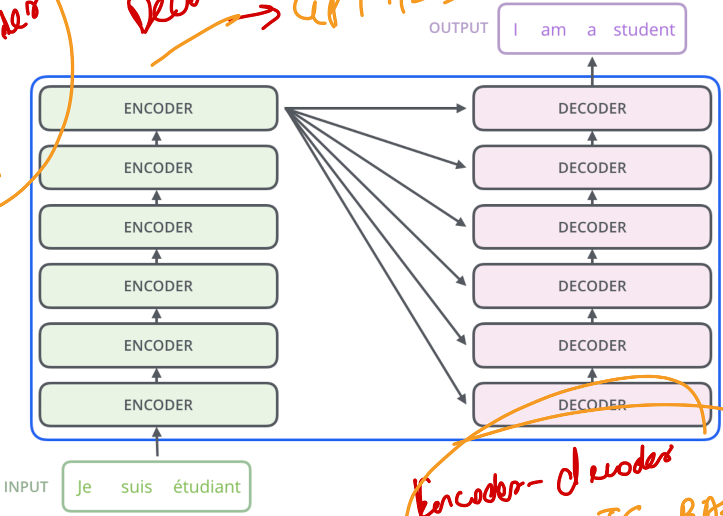
Why should pretraining and finetuning help, from a "training neural nets" perspective?

- Consider, provides parameters $\hat{\theta}$ by approximating $\min_{\theta} \mathcal{L}_{\text{pretrain}}(\theta)$.
  - (The pretraining loss.)
- Then, finetuning approximates $\min_{\theta} \mathcal{L}_{\text{finetune}}(\theta)$, starting at $\hat{\theta}$.
  - (The finetuning loss)
- 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!

Using Transformers for Pretraining

Encoder ↓ BERT

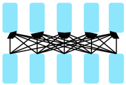
Decoder → GPT 1/2/3



Encoder-decoder → T5, BART

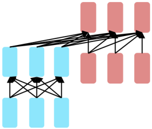
Pretraining for three types of architectures

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



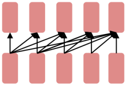
Encoders

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



Encoder-Decoders

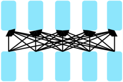
- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

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

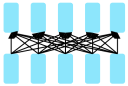
Pretraining Encoders



Encoders

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

Pretraining Encoders



Encoders

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

What would be the objective function?

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

Solution: Use Masks

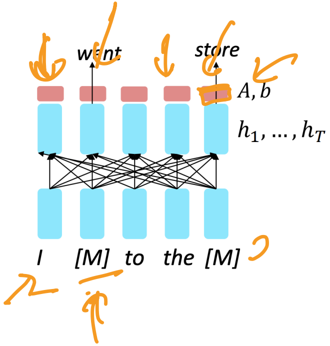
$-\log p(\text{word})$

Masked LM

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

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$

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**.

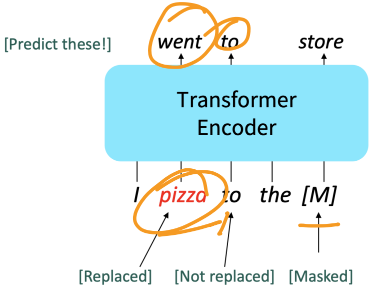


BERT: Bidirectional Encoder Representations from Transformers

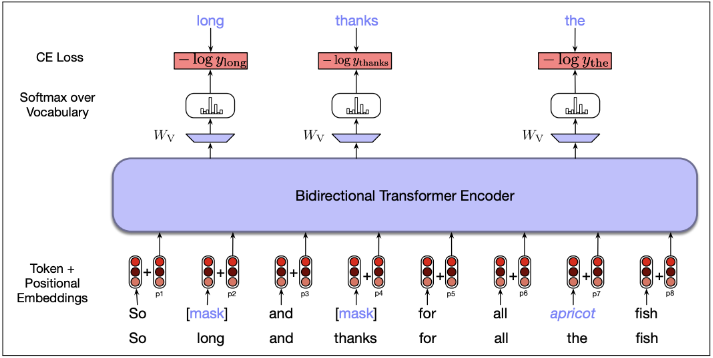
Devlin et al., 2018 proposed the “Masked LM” objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

Some more details about Masked LM for BERT:

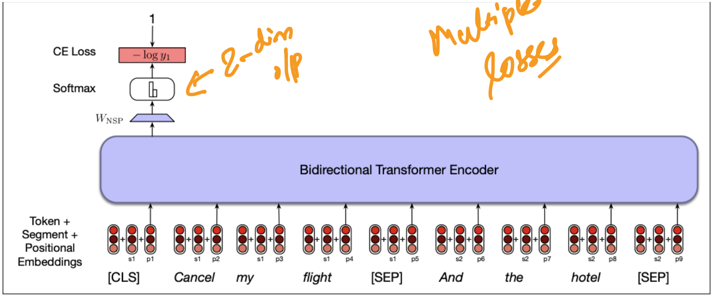
- Predict a random 15% of (sub)word tokens.
    - Replace input word with [MASK] 80% of the time
    - Replace input word with a random token 10% of the time
    - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn’t let the model get complacent and not build strong representations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



BERT: another view



$$y_i = softmax(W_V h_i), W_V \in R^{|V| \times d_h}, h_i \in R^{d_h}$$



2-dim o/p

Multiple losses

T S1 S2

$$y = softmax(W_{NSP}C), W_{NSP} \in R^{2\times d_h}, C \in R^{d_h}$$

BERT: Next Sentence Prediction

Why NSP?

- Masking focuses on predicting words from surrounding contexts so as to produce effective word-level representations.
- Many applications require relationship between two sentences, e.g.,
  - paraphrase detection (detecting if two sentences have similar meanings),
  - entailment (detecting if the meanings of two sentences entail or contradict each other)
  - discourse coherence (deciding if two neighboring sentences form a coherent discourse)

BERT: Bidirectional Encoder Representations from Transformers

dff 4x dh

Hrs

Details about BERT

- 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.”