Contextual word representations

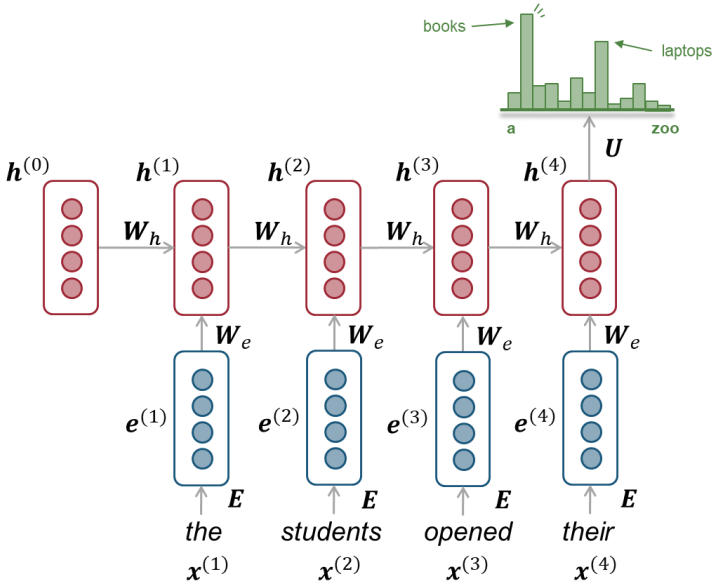
SLP 3 CH 11

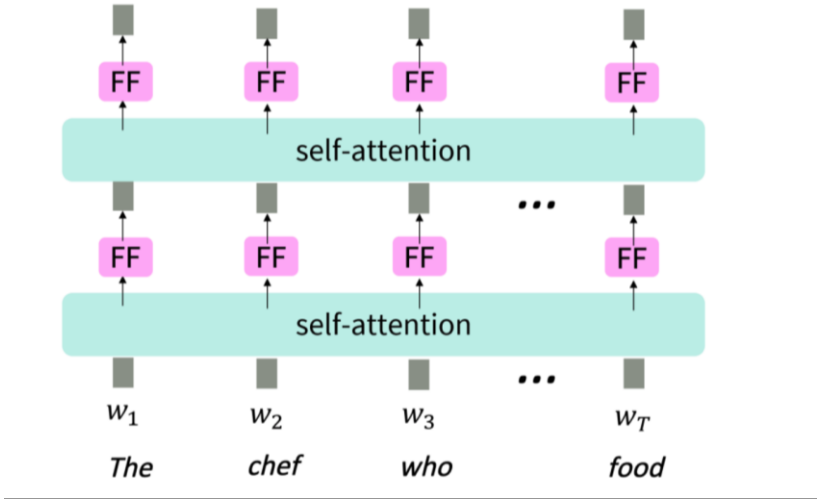
Word representation

- We've learned static word representation
  - LSA, Word2vec
  - Each word has exactly one vector representation
- But a word may have meanings and syntactic behaviors in different contexts!
  - "book a flight" vs. "read a book"
  - 我骑车差点摔倒，好在我一把把把把住了
- Wouldn't it be nice to have representations of words that change over contexts?
  - This is called Contextualized Word Embeddings

Did we already have a solution to this problem?

- In a neural LM (RNN, Transformer), the hidden vector at each position summarizes the context
- They are actually producing context-specific word representations at each position!



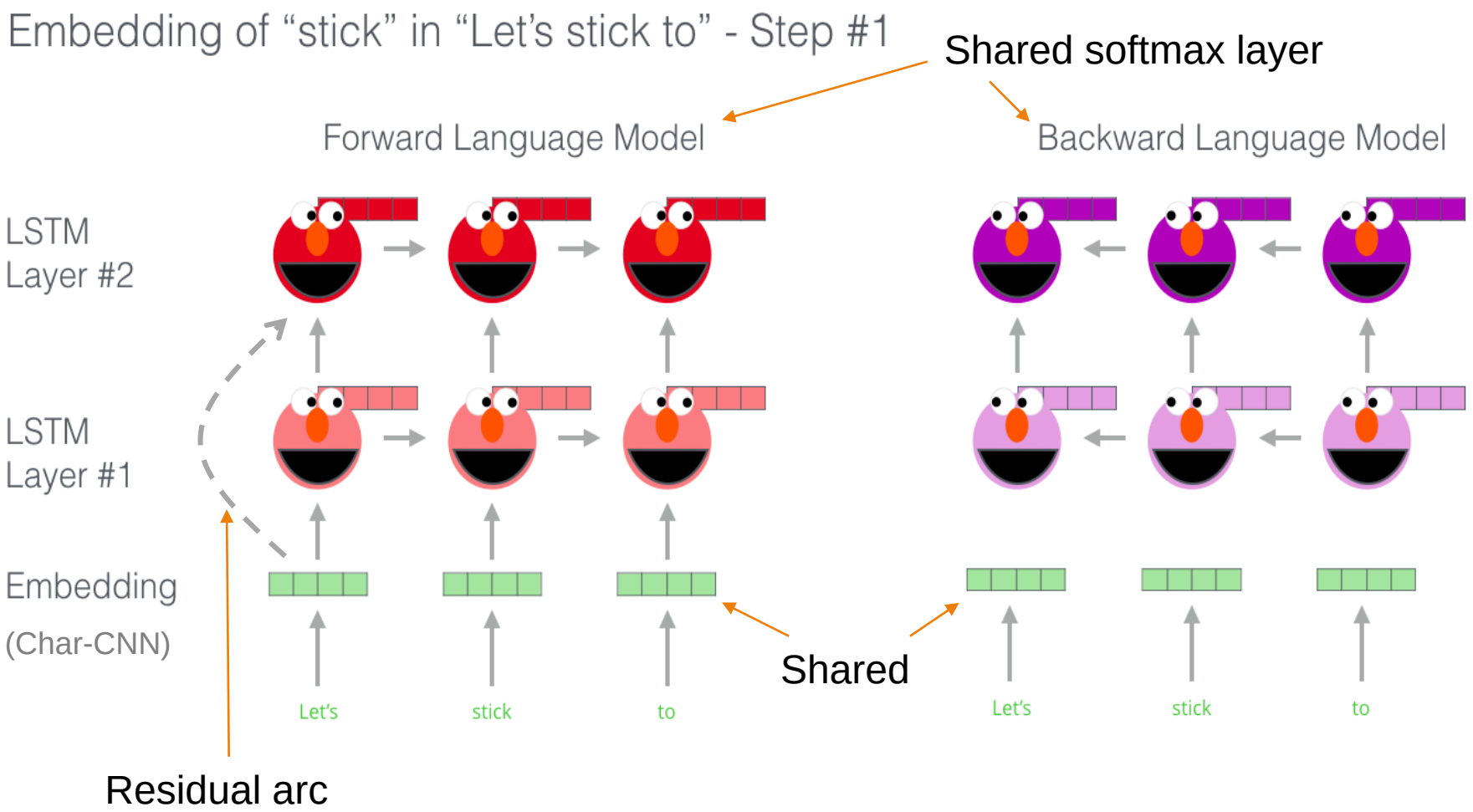


ELMo: Embeddings from Language Models

ELMo: Embeddings from Language Models

- Train a forward LSTM LM and a backward LSTM LM (bi-LSTM)
  - Use 2-layer LSTMs
  - Use character CNN to build initial word representation
  - Use a residual connection
  - Share the parameters for token representation and softmax word prediction layer in the two LSTMs

ELMo: Embeddings from Language Models



ELMo: Embeddings from Language Models

- ELMo learns task-specific combination of BiLSTM representations in different layers.

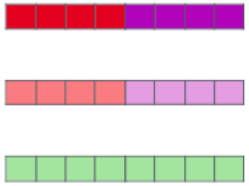
$$ELMo_k = \sum_{j=0}^{L} s_j \cdot h_{k,j}^{LM}$$

  - s_j is a learnable weight for layer j
- The two BiLSTM layers have differentiated uses
  - Lower layer is better for lower-level syntax, etc.
    - Part-of-speech tagging, syntactic dependencies, NER
  - Higher layer is better for higher-level semantics
    - Sentiment, Semantic role labeling, question answering, SNLI

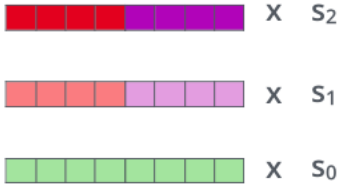
ELMo: Embeddings from Language Models

Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers

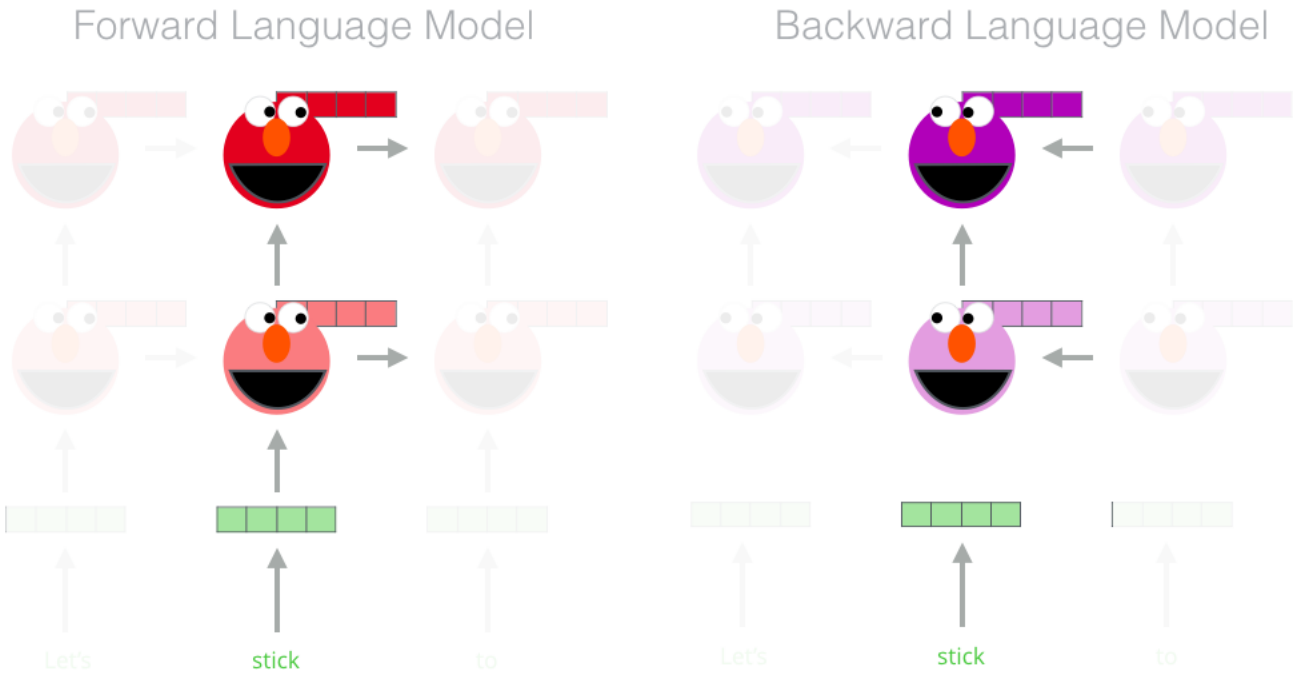


2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context



https://jalammar.github.io/illustrated-bert/

Use ELMo with a task

- Run ELMo on input sentences to get word representations
- Then let (whatever) end-task model use them
- When training the end-task model:
  - Freeze weights of ELMo
  - Or: also train weights of ELMo together with the end-task model, which typically leads to better performance
- This is the pretrain+finetune paradigm!
  - Pretrain: train the word embedding model on a general task with lots of data (e.g., LM)
  - Finetune: for a specific task, connect the word embedding model to the task model and continue its training together with the task model

BERT: Bidirectional Encoder Representations from Transformers

From ELMo to BERT

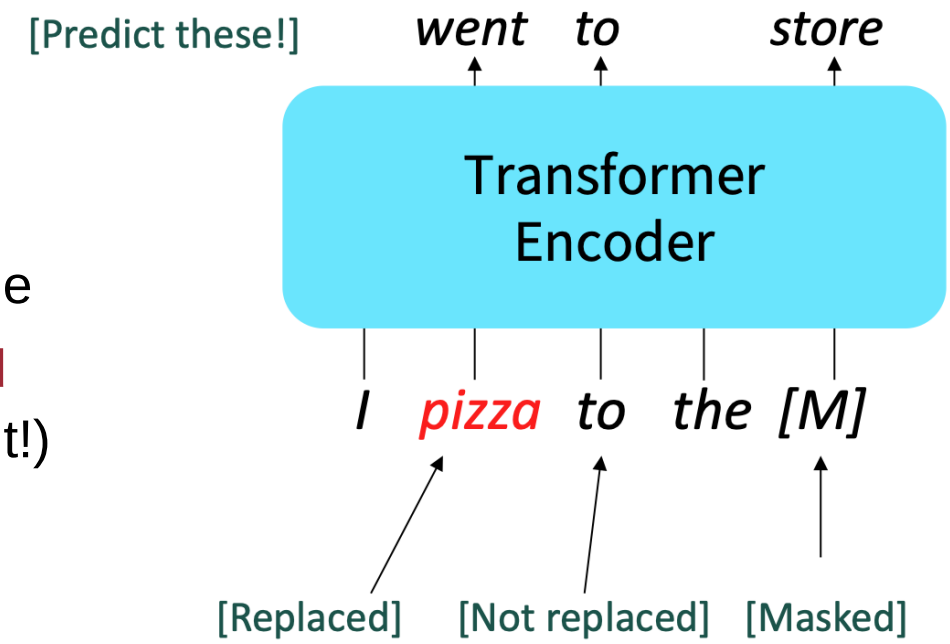
- ELMo: separate representations for the left context and right context
- Why not a unified representation for the whole context?
  - Because LM is unidirectional
- Solution: masked language model (MLM)
  - Mask out *k*% of the input words, and then predict the masked words

store ↑ gallon ↑

I went to the [MASK] to buy a [MASK] of milk

Masked Language Models in BERT

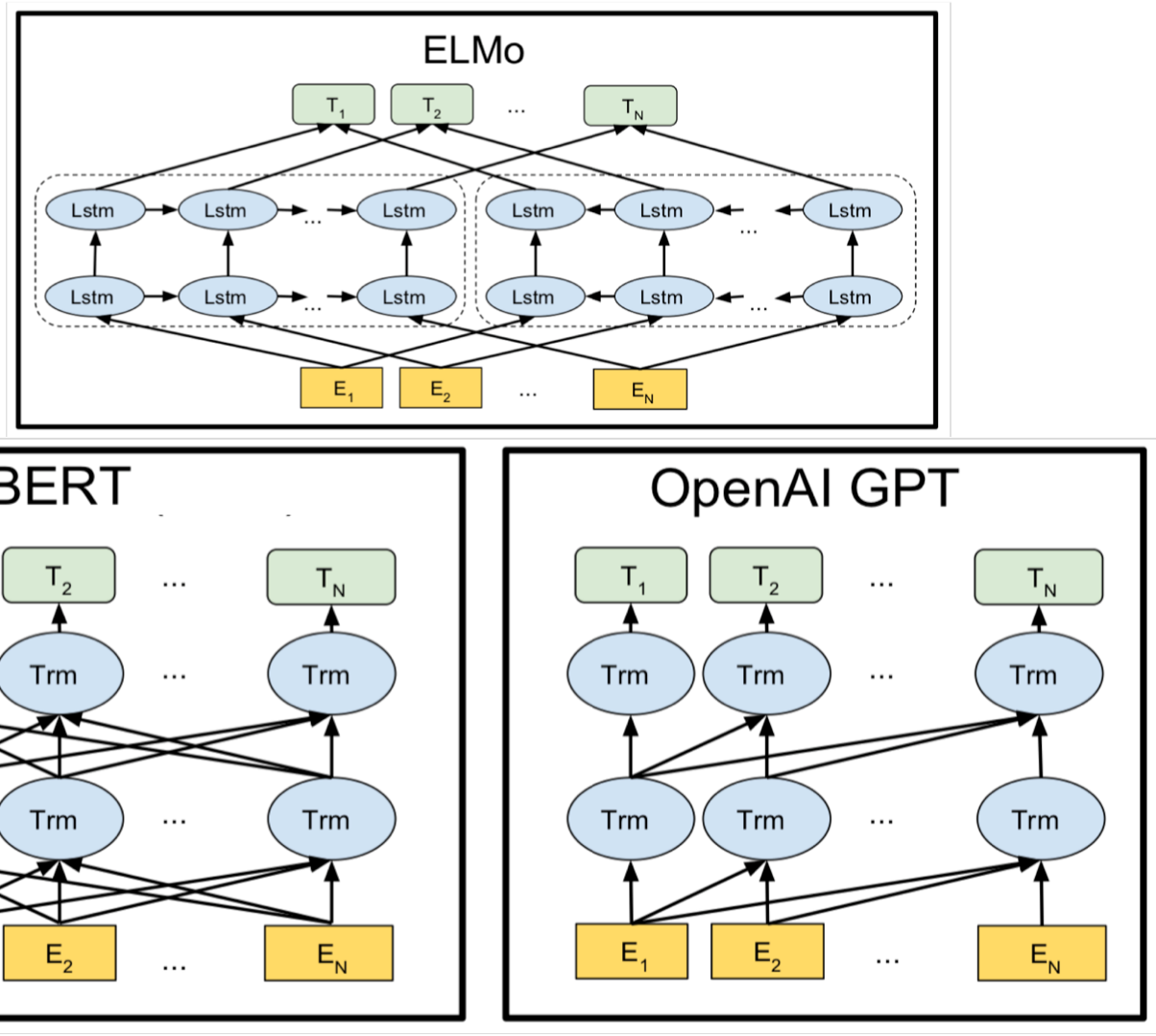
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (still predict it!)
- Why the last part?
  - No masking during fine-tuning and prediction
  - The model must learn to build strong representations of non-masked words



From ELMo to BERT

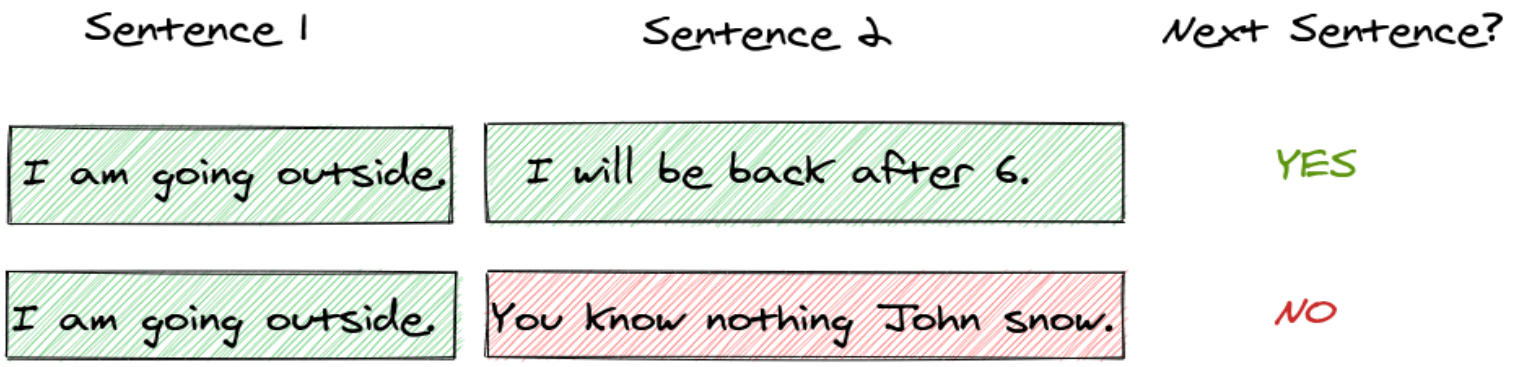
- Replace LSTM with Transformer
  - Long-range dependency & parallelizability
- Use subword tokenization (WordPiece)
  - Similar to BPE
- Use the top-layer as contextual representations

BERT, ELMo and GPT



BERT complication: next sentence prediction (NSP)

- To learn relationships between sentences, predict whether Sentence B is a sentence that proceeds Sentence A



[CLS] I am going outside [SEP] I will be back after 6 [SEP]

↓

FFN+Softmax → IsNext: 99%
NotNext: 1%

Later work argued NSP is not necessary.

Details about BERT

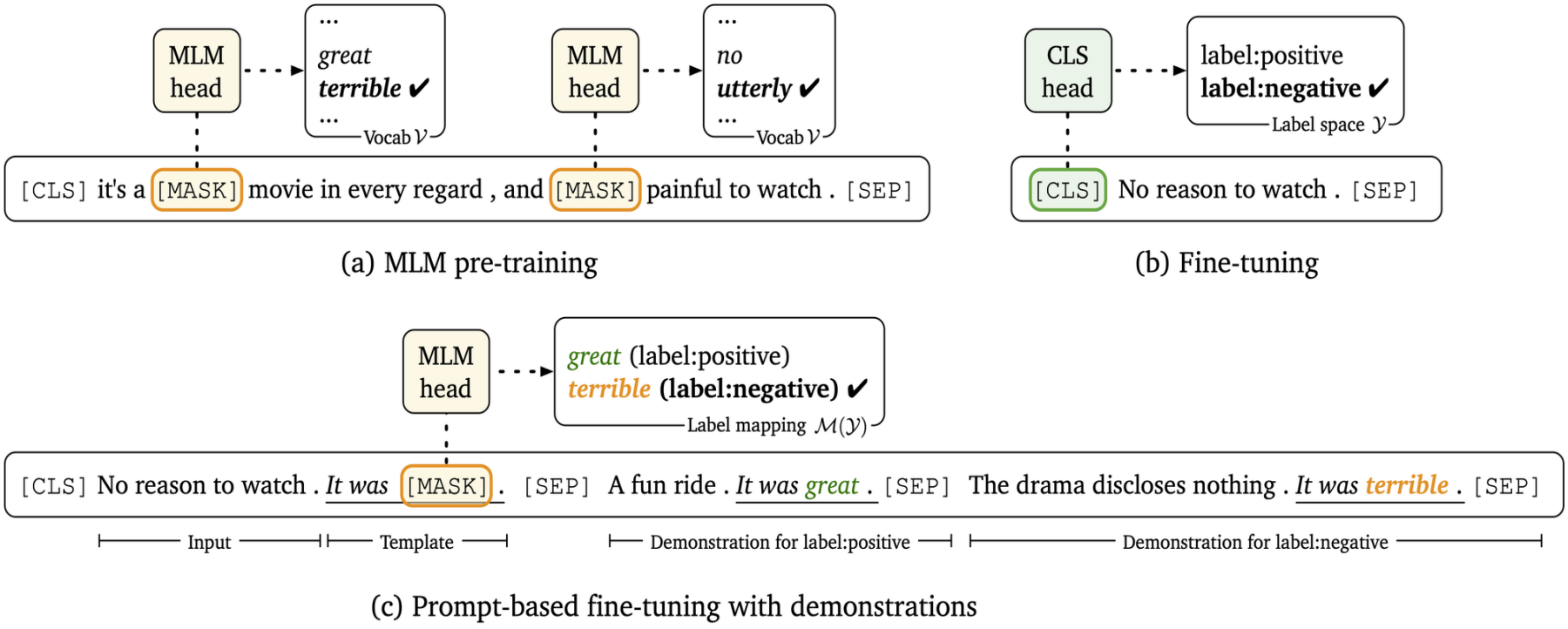
- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- 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.”

Extensions of BERT

- A lot of variants
  - 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
  - ERNIE: introduce entity/relation knowledge into pretraining
  - Many more…

Prompting: a new way to use pretrained LM

- Insert a piece of text (prompt) in the input to formulate a task as MLM.



https://arxiv.org/pdf/2012.15723.pdf

Summary

Contextual word representations

- ELMo
  - BiLSTM + LM
- BERT
  - Transformer + MLM (+NSP)
- Pretrain+finetune paradigm