CS 4774 Machine Learning

Lecture 11: Large Language Models

Yangfeng Ji

Information and Language Processing Lab

Department of Computer Science

University of Virginia

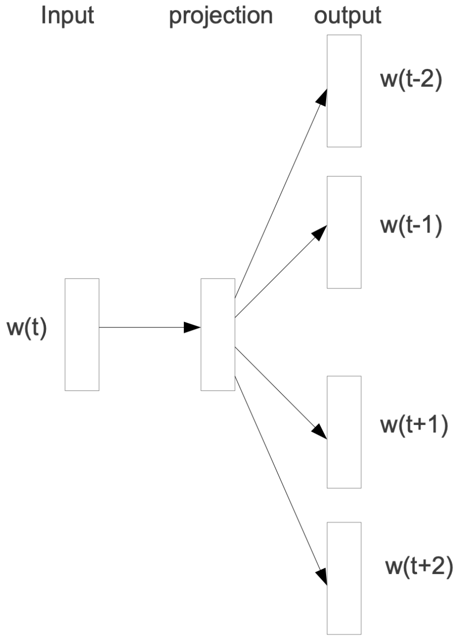
Word Embeddings

Word Embeddings

- In NLP, word embeddings are numeric representations/vectors of words $\{w_i\}$
- Simple algebraic operations can be used to measure word similarity
- For example
  - $w_i^\top w_j$: the semantic similarity between word i and j

Skip-gram Models

- The skip-gram model provides a way of learning word embeddings
- For a given sentence with words $\cdots, w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}, \cdots$, the skip-gram model builds word embeddings by using every word in the sentence to predict its surrounding words:



Examples

After learning, we can use word embeddings to identify some similar words, or calculate their semantic relations

- Nearest words in the embedding space
- Similarity between two words
- Word analogy

Link

Pre-training

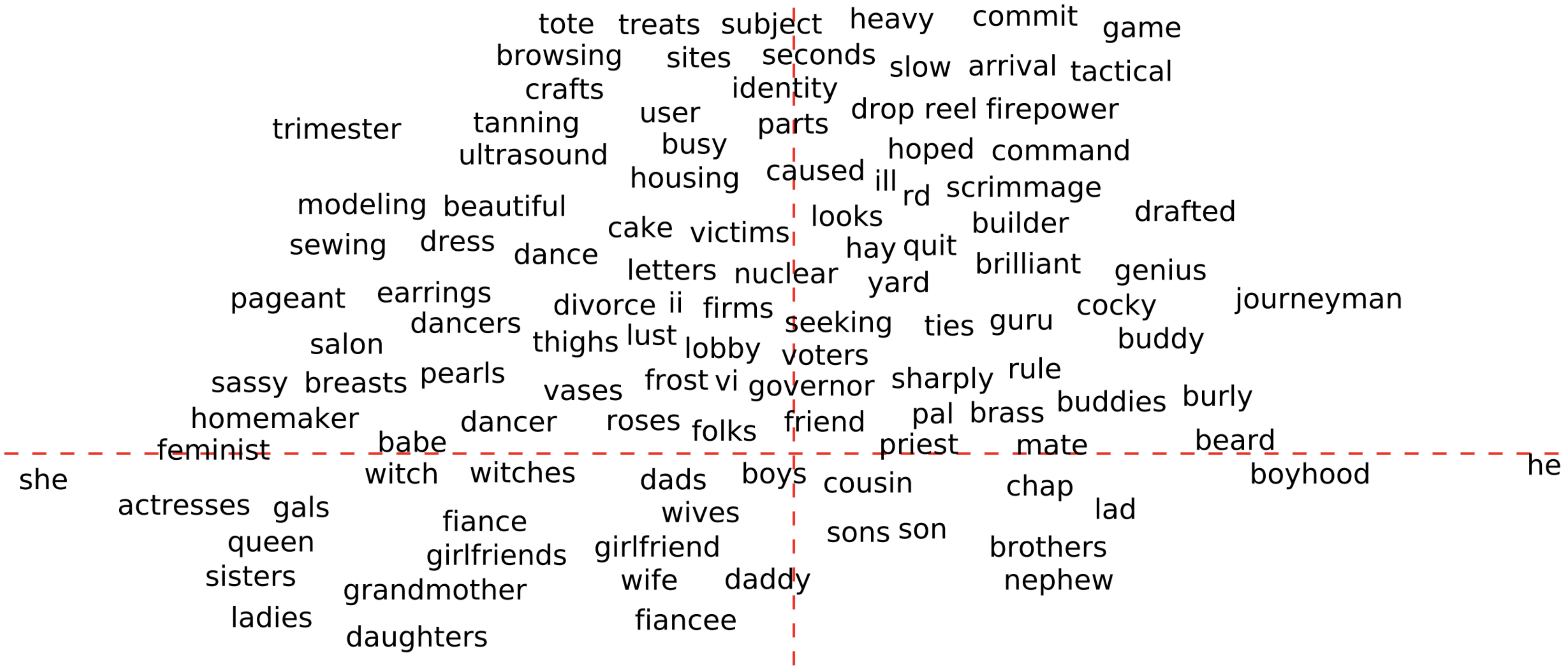
- The idea of pre-training *in this context* is to build word embeddings without having a pre-defined NLP application (e.g., text classification, text generation)
- These pre-trained embeddings can be used generically in many scenarios

Some example pre-trained word embeddings

- Google Word2vec
- Stanford GloVe

Bias in Word Embeddings

Pre-trained word embeddings may contain unexpected bias



Bolukbasi et al., 2016

From Word Embeddings to Sentence Representations

Simple Methods

For a given sentence with N words, some simple methods of calculating sentence representations with pre-trained word embeddings

- Average of word embeddings

$$s = \sum_{i=1}^{N} w_i$$

- Convolutional neural network

$$s = \mathrm{CNN}(w_1, \ldots, w_N)$$

Using RNNs

Using a bi-directional RNN

- Building a RNN from left to right, we can use h_i to replace w_i as contextualized word embeddings

$$\overrightarrow{h}_i = \overrightarrow{\mathrm{RNN}}(w_1, \ldots, w_i)$$

- We can also build another RNN from right to left, as

$$\overleftarrow{h}_i = \overrightarrow{\mathrm{RNN}}(w_i, \ldots, w_N)$$

- The final word embedding of word i is the concatenation of these two vectors

$$h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$$

ELMo

Embeddings from language models (ELMo)

With L-layer LSTM language model, each word w_i will have a list of representations

$$\{h_{i,l}^{LM}; 0 = 1, \ldots, L\}$$

where

- $h_{i,0}^{LM} = w_i$: the word embedding
- $h_{i,l}^{LM}$: the hidden state from the l-th layer of LSTM

ELMo (II)

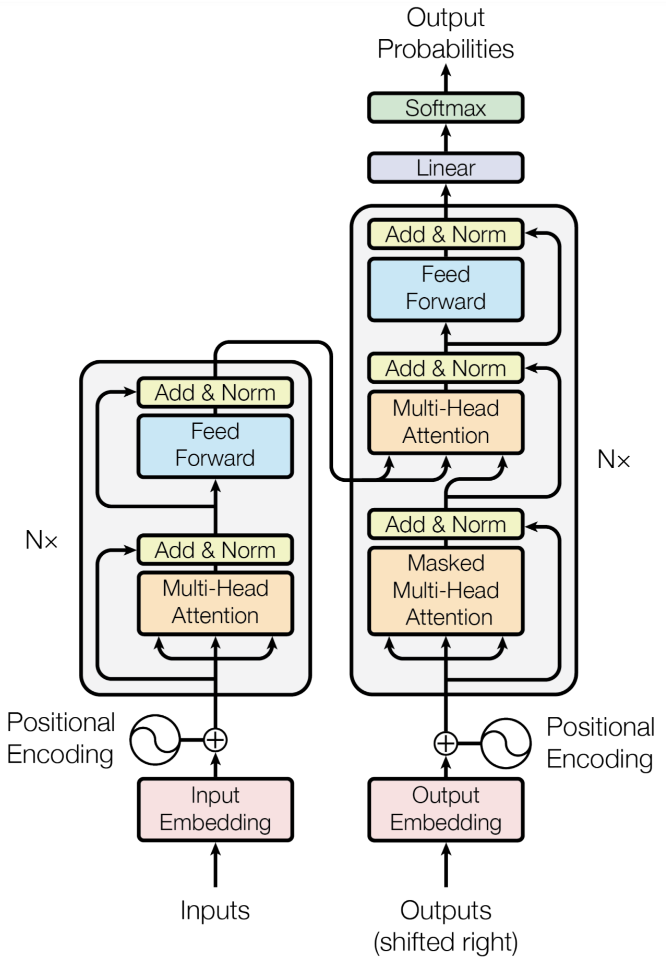
A task-specific word representation from ELMo embeddings is

$$e_i(\gamma, s_l) = \gamma \sum_{l=0}^{L} s_l \cdot h_{i,l}^{LM}$$

where $\gamma \in \mathbb{R}$ and $\{s_l \in \mathbb{R}\}_{l=0}^{L}$ are task-specific parameters

Pre-trained Language Models

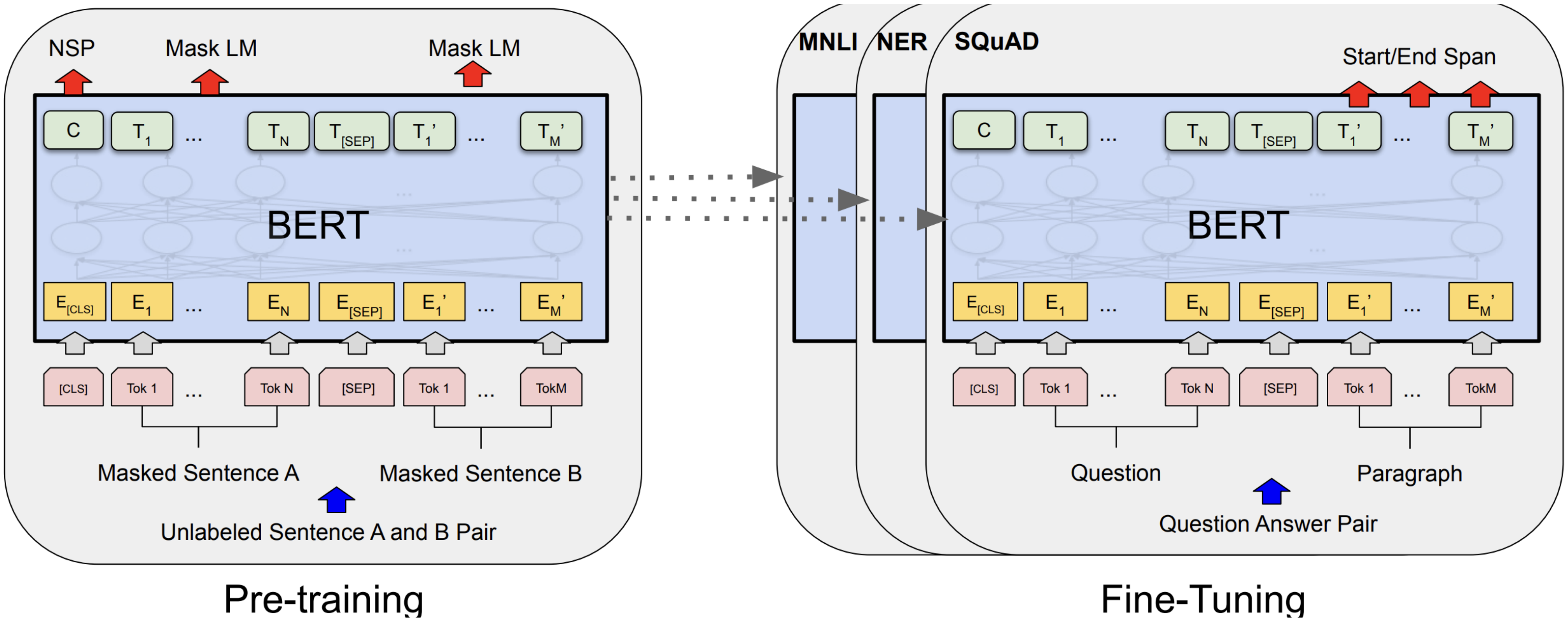
Transformer



Vaswani et al., 2017

BERT

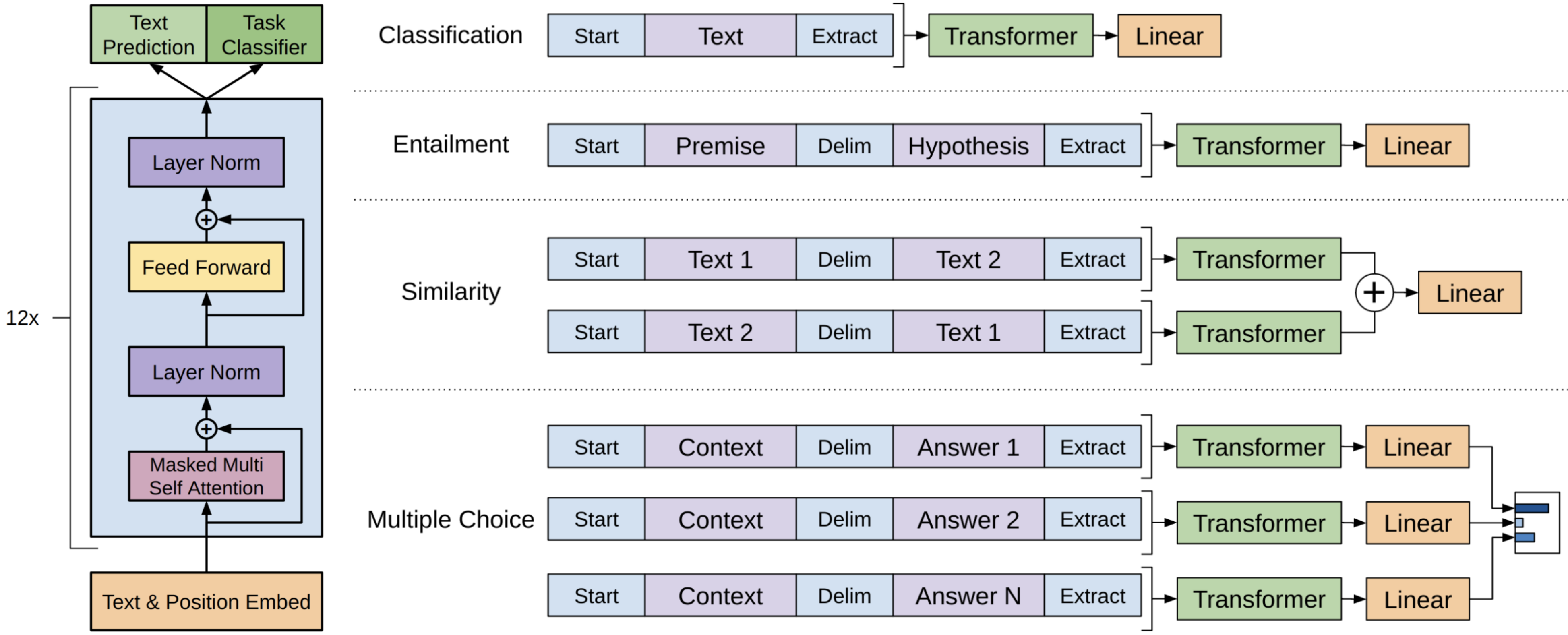
Bidirectional Encoder Representations from Transfromers



Devlin et al., 2019

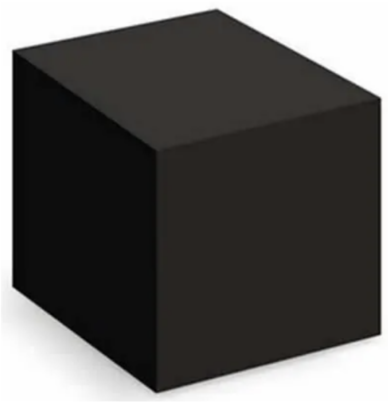
GPT

Generative Pre-trained Transformers



Radford et al., 2018

GPT 4



In the OpenAI's report on GPT-4:

> This report focuses on the capabilities, limitations, and safety properties of GPT-4. GPT-4 is a Transformer-style model [39] pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF) [40]. Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar.

Reinforcement Learning with Human Feedback

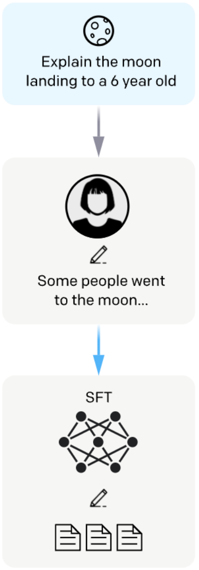
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



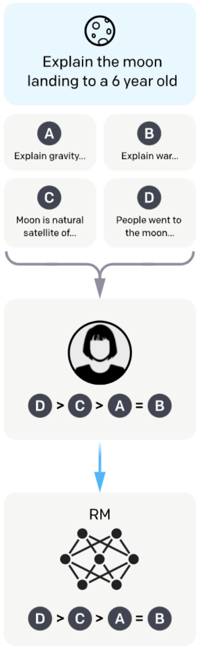
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

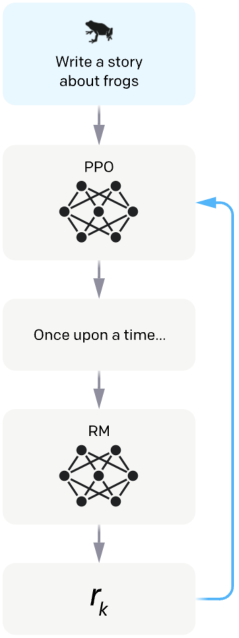
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Data

We don't know what data was used to training the later GPT models, but we can get some idea from the data used to further improve these models (e.g., GPT-3)

Thank You!