Representation Learning

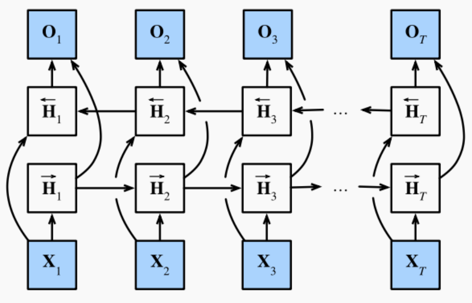
He He

New York University

November 17, 2021

Table of Contents

Recap: BiLSTM



- Classification: $p(y \mid x) = \text{softmax}(\text{linear}(\text{pooling}(o_1, \ldots, o_T)))$
- Sequence labeling: $p(y_t \mid x) = \text{softmax}(\text{linear}(o_t))$
- Sequence generation: decoder + attention

Tasks with two inputs

Natural language inference

Premise: 8 million in relief in the form of emergency housing.

Hypothesis: The 8 million dollars for emergency housing was still not enough to solve the problem.

Label: neutral

Reading comprehension

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion *Denver Broncos* defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

Question: Which team won Super Bowl 50?

Answer: Denver Broncos

Encode two inputs

Goal: $\mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y}$

Simple combination:

- Encode x_1 and x_2 in $\mathbb{R}^d$ separately
- Aggregate the two embeddings, e.g. $\mathrm{MLP}(\mathrm{pooling}(\mathrm{enc}(x_1), \mathrm{enc}(x_2)))$
- Pooling: concatenation, elementwise max, elementwise product etc.
- Modular, but less expressive

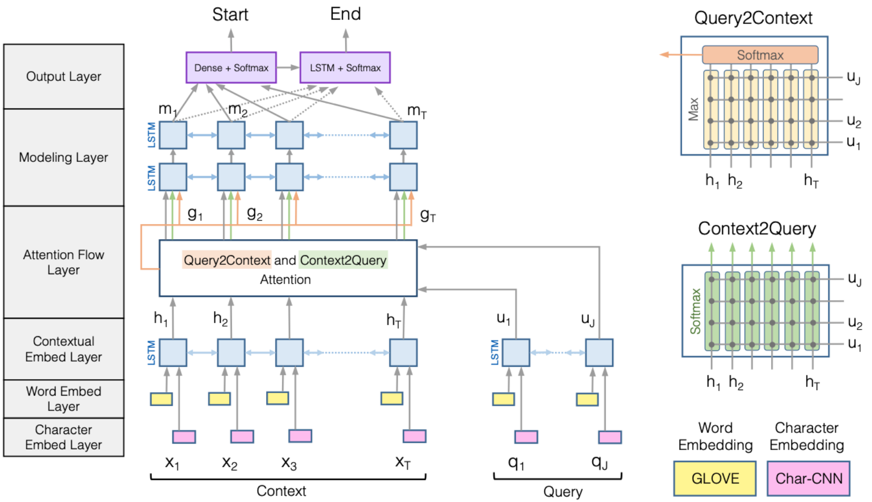
Finer-grained interaction between the two inputs:

- Can we use something similar to the attention mechanism in seq2seq?

BiDAF

Bi-Directional Attention Flow for Machine Comprehension [Seo+ 2017]

Key idea: representation of x_1 depends on x_2 and vice versa



Improve the efficiency of RNNs

Word embedding: represents the meaning of a word

Recurrent neural networks: captures dependence among words in a sentence

Attention mechanism: better modeling of long-range dependence

Multi-layer biLSTM with various attentions was the go-to architecture for most NLP tasks.

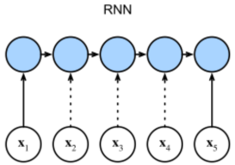
But, RNNs are sequential and difficult to scale up

- We want deeper models trained with larger data.

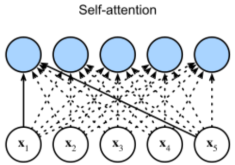
Can we handle dependencies in a more efficient way?

Attention is all you need?

Key idea: get rid of recurrence and only rely on attention



- Sequential O(n)
- Uni-directional and may forget past context
- Handle long sequence trivially



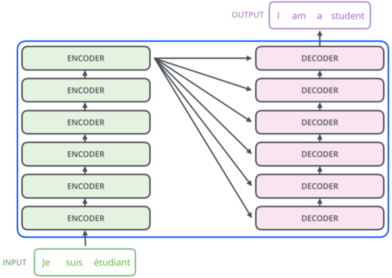
- Parallelizable $O(n^2)$
- Direct interaction between any word pair
- Maximum sequence length is fixed

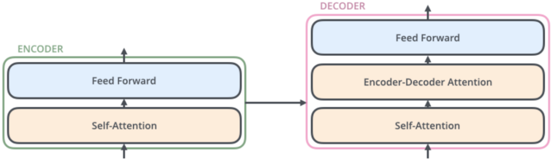
Transformer overview

Attention is all you need. [Vaswani+ 2017]

Replaces recurrence with self-attention:

- Multi-layer sequence-to-sequence model
- Self-attention based sequence representation

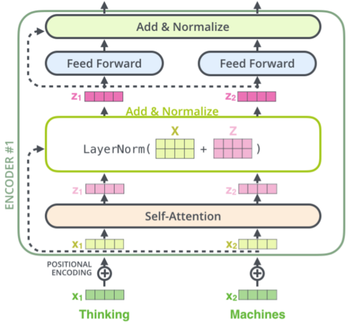




[https://jalammar.github.io/illustrated-transformer/]

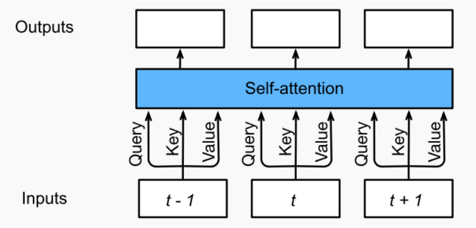
Transformer block

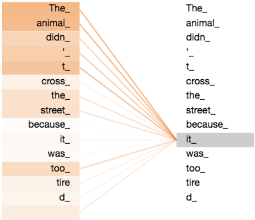
- Multi-head self-attention
  - Capture dependence among inputs
- Positional encoding
  - Capture order information
- Residual connection and layer normalization
  - Efficient optimization



[https://jalammar.github.io/illustrated-transformer/]

Self-attention





- Seq2seq attention: keys and values are the input words, and queries are the output (prefix).
- Self-attention: keys, values, and queries are all from the input words.
  - Input: a sequence of words
  - Output: (contextualized) embeddings for each word
- Each word (as a query) interacts with all words (keys/values) in the input
- Computation of the attention output for each word can be parallelized

Matrix representation

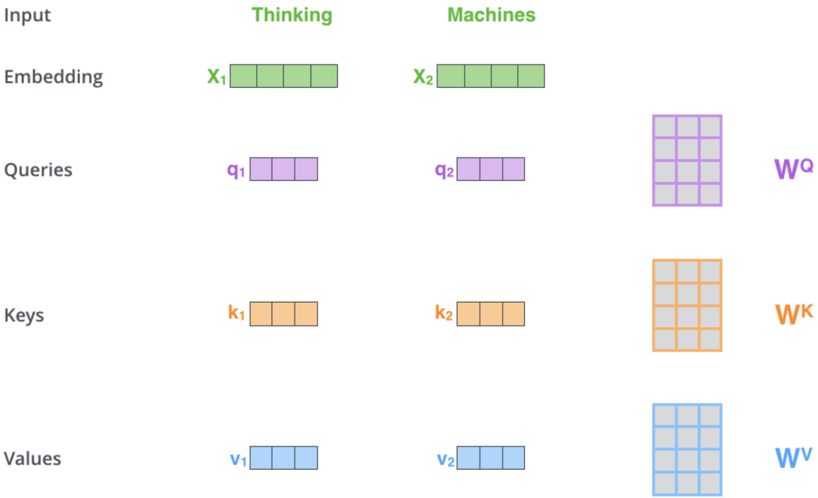


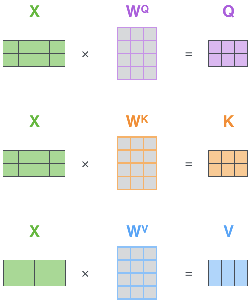
Figure: From “The Illustrated Transformer”

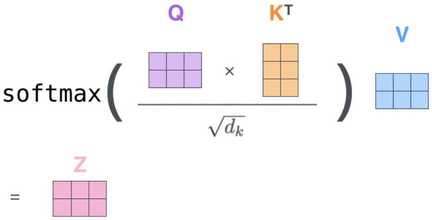
Scaled dot-product attention

Scaled dot-product attention

$$\alpha(q, k) = q \cdot k / \sqrt{d}$$

- $\sqrt{d}$: dimension of the key vector
- Avoids large attention weights that push the softmax function into regions of small gradients



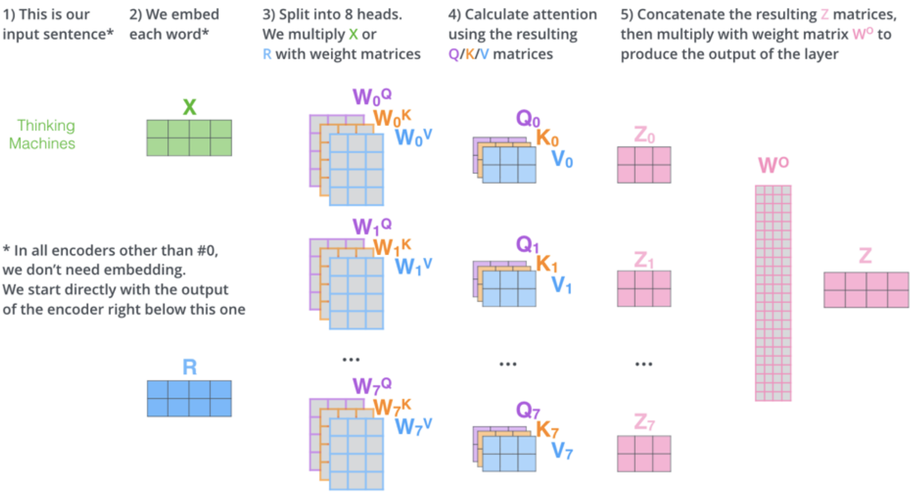


Multi-head attention: motivation

Time flies like an arrow

- Each word attends to all other words in the sentence
- Which words should "like" attend to?
  - Syntax: "flies", "arrow" (a preposition)
  - Semantics: "time", "arrow" (a metaphor)
- We want to represent different roles of a word in the sentence: need more than a single embedding
- Instantiation: multiple self-attention modules

Multi-head attention



Time complexity

Concatenate and project:

$$Z = W[Z_0, Z_1, \ldots, Z_m]$$

$$Z_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d}})V_i$$

- Problem size
  - Sequence length: n
  - Number of heads: m
  - Embedding size: d

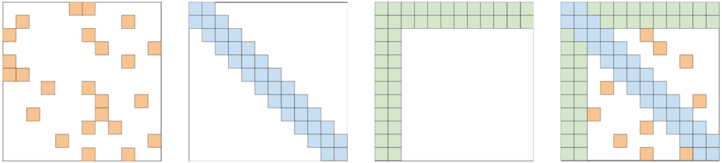
Expensive for long sequences!

- Time complexity
  - Attention score for a pair of word (dot product): O(d)
  - Self-attention (pairwise interaction): $O(n^2)$
  - Multi-head attention: O(m)
  - Overall: $O(mdn^2)$

Efficient self-attention

Goal: reduce the $O(n^2)$ time/space complexity for long sequence problems

- Sparse attention



[Zaheer et al., 2021]

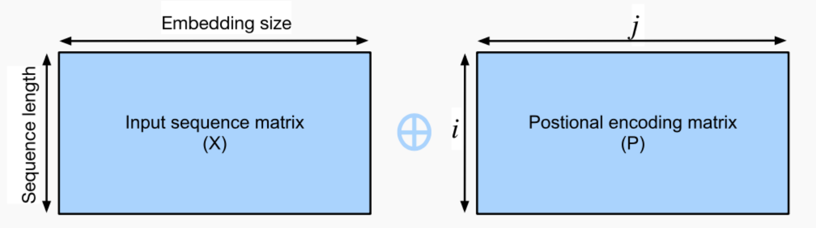
- Locality sensitive hashing
- Low-rank decomposition

$O(n^2) \rightarrow O(nk)$ where k is small

Position embedding

Motivation: model word order in the input sequence

Solution: add a position embedding to each word



Position embedding:

- Encode absolute and relative positions of a word
- (Same dimension as word embeddings)
- Learned or deterministic

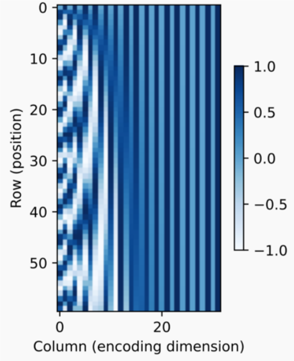
Sinusoidal position embedding

Intuition: binary encoding

- The frequency of bit flips increases from left to right

0 → 000
1 → 001
2 → 010
3 → 011
4 → 100
5 → 101
6 → 110
7 → 111

Continuous version →



[https://www.d2l.ai]

Col 1: $\sin(w_1 t)$
Col 2: $\cos(w_1 t)$
Col 3: $\sin(w_2 t)$
Col 4: $\cos(w_2 t)$
...

w_i: frequency
t: position

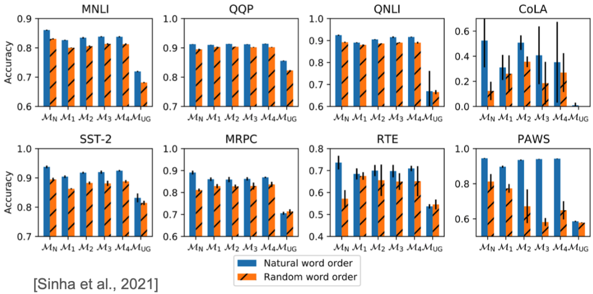
How important is word ordering?

Reasonable performance when trained on permuted n-grams!

Are word ordering unimportant?

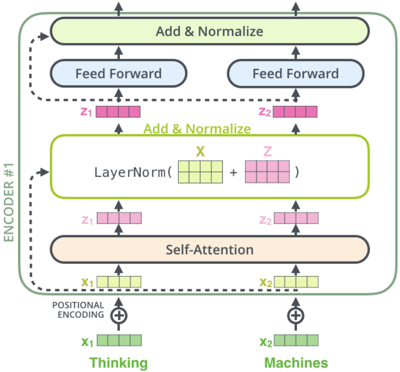
- May need better evaluation of “understanding”

- Results are only on English



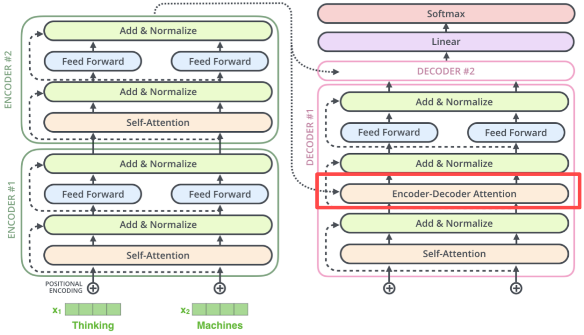
[Sinha et al., 2021]

Residual connection and layer normalization



- Residual connection: add input to the output of each layer
- Layer normalization: normalize (zero mean, unit variance) over all features for each sample in the batch
- Position-wise feed-forward networks: same mapping for all positions

Connect the decoder



- Same as the encoder with an additional attention module
- **Encoder-decoder attention**
  - Query: decoder state
  - Value/key: encoder embeddings

- Autoregressive generation
- Self-attention over prefix, encoder-decoder attention over inputs
- Output at each position:

$$p(y_t \mid x, y_{1:t-1})$$

- MLE training

Impact on NLP

- Initially designed for sequential data and obtained SOTA results on MT
- Replaced recurrent models (e.g. LSTM) on many tasks
- Enabled large-scale training which led to pre-trained models such as BERT and GPT-2

Limitation: fixed length input (see Longformer, Performer etc.)

Table of Contents

Representation learning

What are good representations?

Contains good features for downstream tasks

Example:

negative   the food is good but doesn't worth an hour wait

Simple features (e.g. BoW) require complex models.
Good features only need simple (e.g. linear) models.

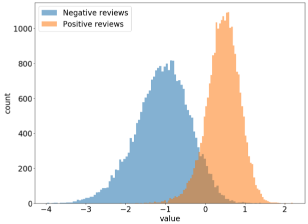


Figure: Sentiment neuron [Radford+ 2017]

Representation learning

Applications of good representations:

- Learning with small data: fine-tuning on learned representations
- Multi-task and transfer learning: one representation used for many tasks
- Metric learning: get a similarity metric for free

How do we learn the representations?

- Self-supervised learning: obtain representations through generative modeling

Self-supervised learning

Key idea: predict parts of the input from the other parts

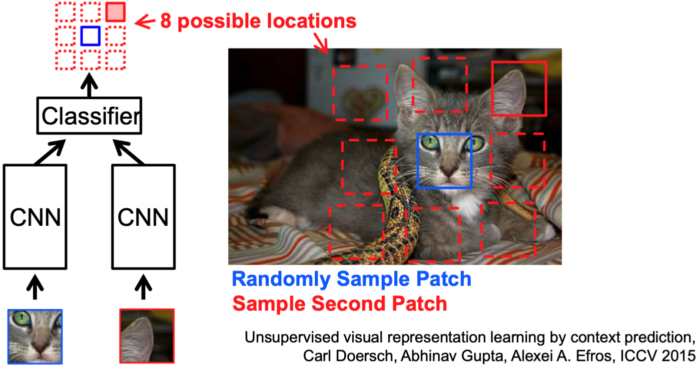


Figure: Slide from Andrew Zisserman

- Other supervision signals: color, rotation etc.
- Video: predict future frames from past frames

Representation learning in NLP

Word embeddings

- CBOW, Skip-gram, GloVe, fastText etc.
- Used as the input layer and aggregated to form sequence representations

Sentence embeddings

- Skip-thought, InferSent, universal sentence encoder etc.
- Challenge: sentence-level supervision

Can we learn something in between?

Word embedding with contextual information

Transfering knowledge from neural LM

Key idea: use representation from a generative model (i.e. an LM)

- Representation (e.g. hidden state at each word) is context-sensitive
- Contains relevant contextual information for predicting the next word

Early work:

- Fine-tune a recurrent LM for downstream tasks [Dai+ 2015, Howard+ 2018]
- Use word embedding from a pre-trained LM in addition to standard word embedding [Peters+ 2017]
- Promising results on a smaller scale

Embeddings from language models (ELMo) [Peters+ 2018]

- Use word embeddings from a bi-directional LM
- Success on multiple NLP tasks

ELMo pretraining

Forward/backward language models:

- $p_{\text{fwd}}(x) = \prod_{t=1}^{T} p(x_t \mid \underbrace{x_{1:t-1}}_{\text{past}}; \theta_{\text{fwd}})$
- $p_{\text{bwd}}(x) = \prod_{t=T}^{1} p(x_t \mid \underbrace{x_{t+1:T}}_{\text{future}}; \theta_{\text{bwd}})$
- Each LM is a two layer LSTM, with shared input embedding layer and softmax layer

Subword representation:

- First layer word embedding is from character convolutions

Data: one-billion word benchmark (monolingual data from WMT)

ELMo embeddings

Contextual embeddings capture word senses.

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Figure: From [Peters+ 2018].

ELMo Fine-tuning

Obtain contextual word embeddings from each layer $j \in 0, \ldots, L$ of biLM:

$$\text{Embed}(x_t, j) = \begin{cases} [\overrightarrow{h}_{t,j}; \overleftarrow{h}_{t,j}] & \text{for } j > 0 \\ \text{CharEmbed}(x_t) & \text{for } j = 0 \end{cases}$$

Task-specific combination of embeddings:

$$\text{Embed}(x_t) = \gamma \sum_{j=0}^{L} w_j \text{Embed}(x_t, j)$$

Fix biLM and use the contextual word embeddings as input to task-specific models. (Can also add to the output layer.)

Regularization is important: L_2 or dropout.

ELMo results

Improvement on a wide range on NLP tasks:

- *reading comprehension* (SQuAD)
- entailment/natural language inference (SNLI)
- semantic role labeling (SRL)
- coreference resolution (Coref)
- *named entity recognition* (NER)
- sentiment analysis (SST-5)

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Takeaways

- Main idea: use biLM for representaiton learning
- Outputs from all layers are useful
  - Lower layer is better for syntactic tasks, e.g. POS tagging, parsing
  - Hight layer is better for semantic tasks, e.g. question answering, NLI
  - Some fine-tuning of the pre-trained model is needed.
- *Large-scale training* is important

Next, pre-trained transformer models.

Transformer models

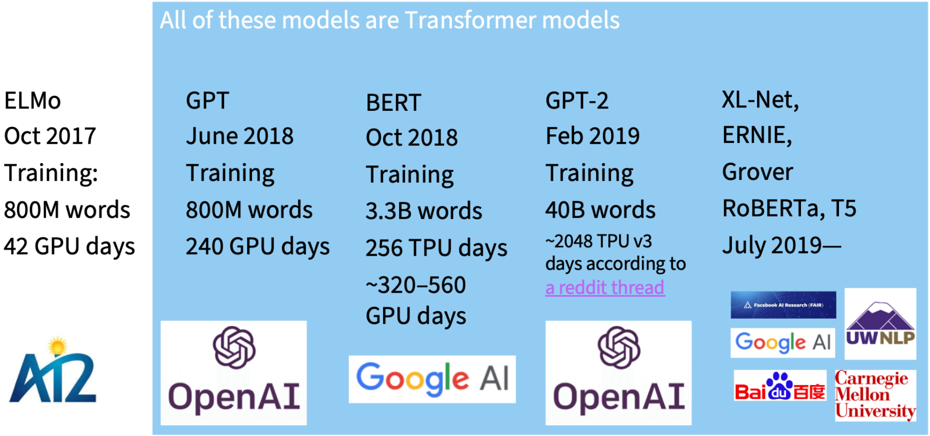


Figure: Slide from Chris Manning

Bidirectional Encoder Representations from Transformers (BERT)

Pre-training:

1. Masked LM:

$$\mathbb{E}_{x\sim\mathcal{D},i\sim p_{\text{mask}}} \log p(x_i \mid x_{-i};\theta)$$

   (not a LM)

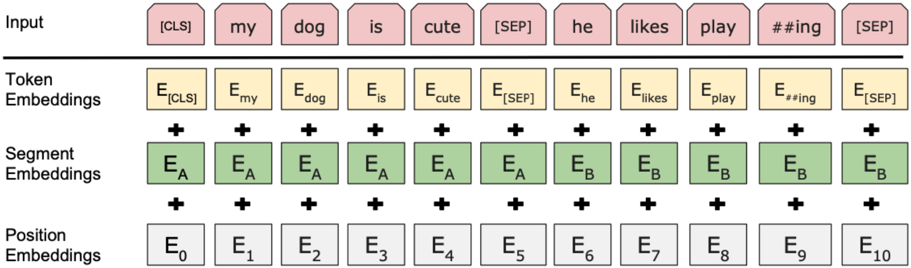
   - x_{-i}: noised version of x where x_i is replaced by `[MASK]`, a random token, or the original token
   - $p(x_i \mid x_{-i};\theta) = \text{Transformer}(x_{-i}, i)$

2. Next sentence prediction:

$$\mathbb{E}_{x^1\sim\mathcal{D},x^2\sim p_{\text{next}}} \log p(y \mid x^1, x^2)$$

   - y: whether x^2 follows x^1
   - Not as useful as masked LM

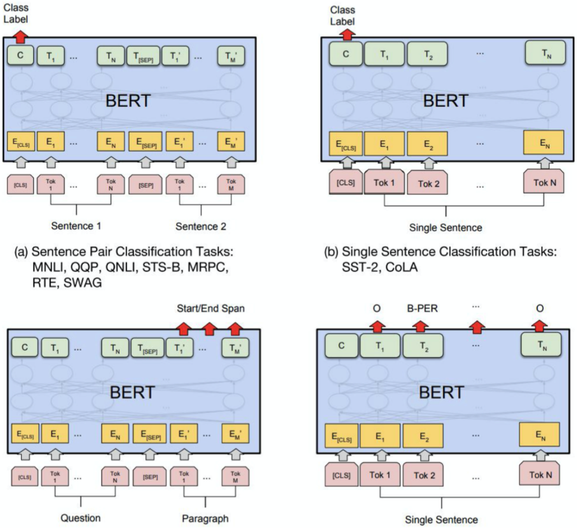
BERT sentence pair encoding



- [CLS]: first token of all sequences; used for next sentence prediction
- Distinguish two sentences in a pair: [SEP] and segment embedding
- Learned position embedding
- Subword unit: wordpiece (basically byte pair encoding)

BERT fine-tuning

All weights are fine-tuned (with a small learning rate)



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Recent progress

GLUE: benchmark of natural language understanding tasks

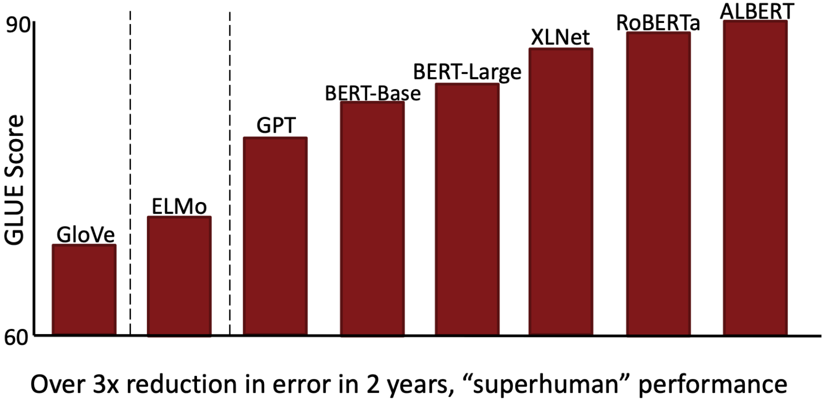


Figure: Slide from Chris Manning

The new pre-train then fine-tune paradigm

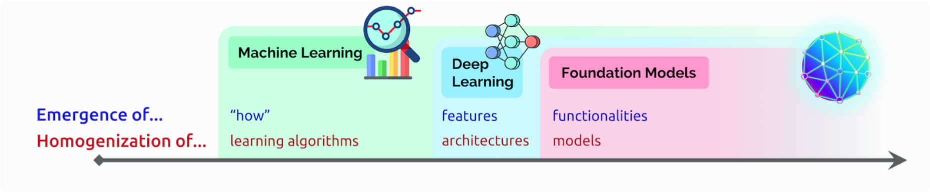


Figure: [Bommasani et al., 2021]

- *One model* to absorb large amounts of raw data from various domains and modalities
- Then *adapted* to different downstream tasks

Summary

Off-the-shelf solution for NLP tasks: fine-tune BERT (and friends)

What's next?

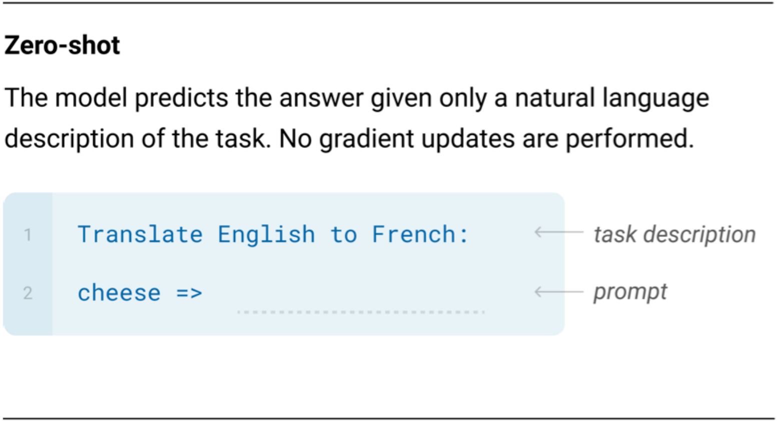
- Processing long text
- Efficient training/inference
- Learning with a small amount of data
- Generalize to new test distributions (solve tasks, not datasets)

Table of Contents

In-context learning

GPT-3 by OpenAI: Transformer-based LM with 175B parameters

Zero-shot learning given task instruction / prompt:



In-context learning

Few-shot learning with in-context examples (with no gradient update):

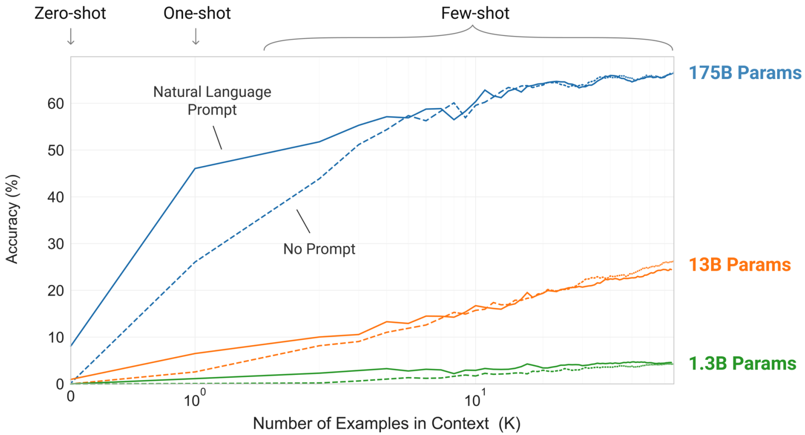
Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

1 `Translate English to French:`	*task description*
2 `sea otter => loutre de mer`	*examples*
3 `peppermint => menthe poivrée`	
4 `plush girafe => girafe peluche`	
5 `cheese =>`	*prompt*

In-context learning

Larger models make increasingly efficient use of in-context information



In-context learning

Pre-trained LMs can be adapted for multimodal learning too:

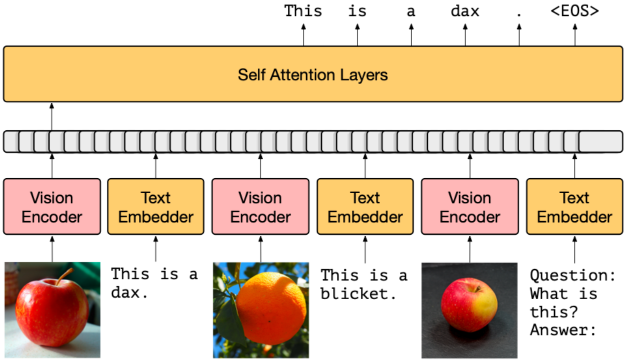


Figure: Multimodal Few-Shot Learning with Frozen Language Models

Text embedder and self-attention use frozen weights from pre-trained LM.