Fundamentals of Machine Learning:

Self-attention Networks and Course Wrapup

Prof. Andrew D. Bagdanov (`andrew.bagdanov AT unifi.it`)

UNIVERSITÀ DEGLI STUDI FIRENZE
DINFO
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Outline

Introduction

Lecture Objectives

- This lecture aims only to give a broad overview of self-attention layers and the Transformer network architecture.
- None of this will be on the exam.
- Relax and enjoy this look at the current state-of-the-art.

Self-attention Networks

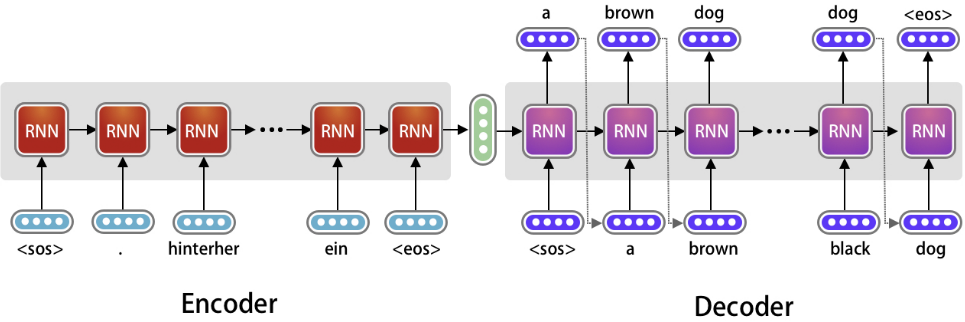
Variable length inputs and outputs

- Many types of input cannot be easily modeled as vectors of fixed dimensionality (e.g. $\mathbb{R}^d$).
- Similarly, some outputs might not be easily modeled as vectors of fixed dimensionality.
- A classic example is machine translation:

```
The black cat is on the wooden table.
--> Il gatto nero è sul tavolo di legno.
```

```
I wonder what Santa Claus will bring me for Christmas?
--> Mi chiedo cosa mi porterà Babbo Natale per Natale?
```

The classical approach: Recurrent Neural Networks (RNNs)



- This was the state-of-the-art, but has many problems.

The New School: Input embedding (Tokenization)

- First, we need to encode the input sequence into tokens.
- This involves learning a mapping from a sequence of one-hot vectors into vectors in a continuous vector space.
- Let $S = [\mathbf{w}_1^T; \mathbf{w}_2^T; \mathbf{w}_3^T; \ldots; \mathbf{w}_n^T]$ be a matrix whose rows are one-hot vectors in R^D (D is the size of the vocabulary and can be very large).
- We can then embed these words into a new space like by multiplying it by an embedding matrix:

$$T_0 = SW_e$$

- If $W_e \in \mathbb{R}^{Dxd}$ the embedding space has dimensionality d (typically $d \ll D$).
- W_e can be learned or we can use a standard tokenizer (e.g. from HuggingFace).

The New School: Queries, Keys, and Values

- Let's think for a minute about how old-school image search engines worked.
- A self-attention layer starts by mapping input tokens into three independent representations.
- This is done using our old friend the linear layer (without bias):

$$Q = T_0 W_q \quad K = T_0 W_k \quad V = T_0 W_v$$

- Our queries Q will be compared to keys K and the resulting similarities used to combine values V.
- This will be done for all pairs of input tokens.

The New School: Self-attention

- The purpose of attention is: for each output in the sequence, predict which input tokens to focus on and how much.
- We compare queries and keys using inner products (i.e. cosine similarity).
- But, we want our combination of values to be an affine combination (coefficients sum to 1).
- So, our attention weights are computed as:

$$A = \text{softmax}(QK^T) \text{ (softmax works along rows).}$$

- And the values are combined to form each output token:

$$T_1 = AV.$$

- And we have transformed our input into new tokens.

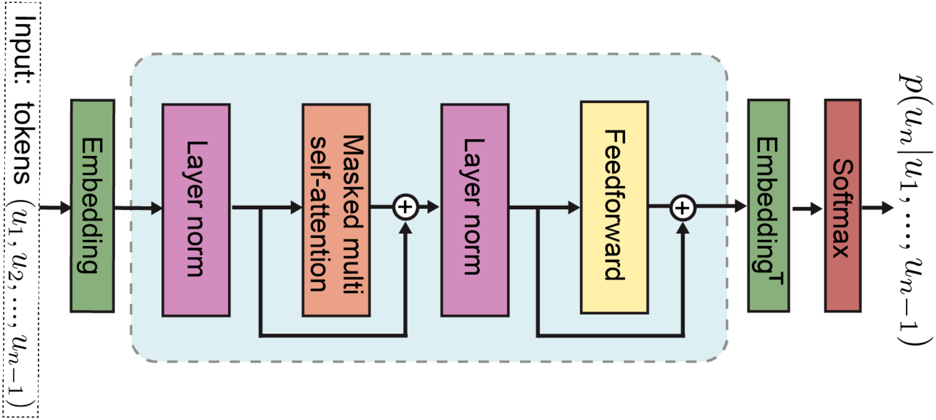
The New School: A simple example

- First, a deep breath.

From Self-attention to the Transformer

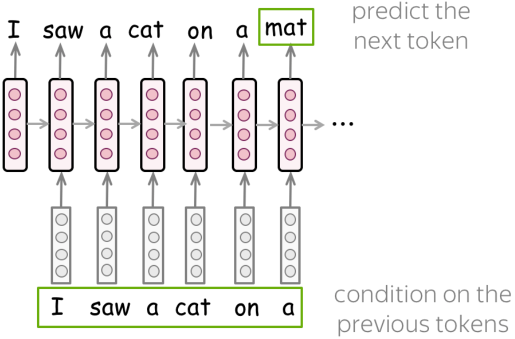
Autoregressive training

- To build a Transformer we stack multiple transformer layers in sequence.
- As usual, there are a *lot* of extra details.



Autoregressive training

- But that is basically *all there is*.
- GPT-2/3 were trained exclusively to *predict the next token*.
- Trained on a massive amount of text data.



Types of Transformers

- Encoder-Only Transformers:
  - Architecture: Consist only of the encoder stack.
  - Use Cases: Used for tasks where a fixed-length representation of the input is needed (e.g. text classification).
  - Example: BERT (Bidirectional Encoder Representations from Transformers).
- Decoder-Only Transformers:
  - Architecture: Consist only of the decoder stack.
  - Use Cases: Used for autoregressive tasks where the model generates output one token at a time.
  - Example: GPT (Generative Pre-trained Transformer).
- Encoder-Decoder Transformers:
  - Architecture: Both an encoder and a decoder.
  - Use Cases: Used for tasks that require sequence-to-sequence processing (e.g. machine translation).
  - Example: T5 (Text-to-Text Transfer Transformer).

Zero- and Few-shot Contextual Learning

The Fine-tuning Paradigm

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

`Translate English to French:`	← *task description*
`cheese =>`	← *prompt*

One-shot

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

```
Translate English to French:      ← task description
sea otter => loutre de mer        ← example
cheese =>                         ← prompt
```

Few-shot Contextual Learning

Few-shot

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

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

task description

examples

prompt

The GPT-3 Family

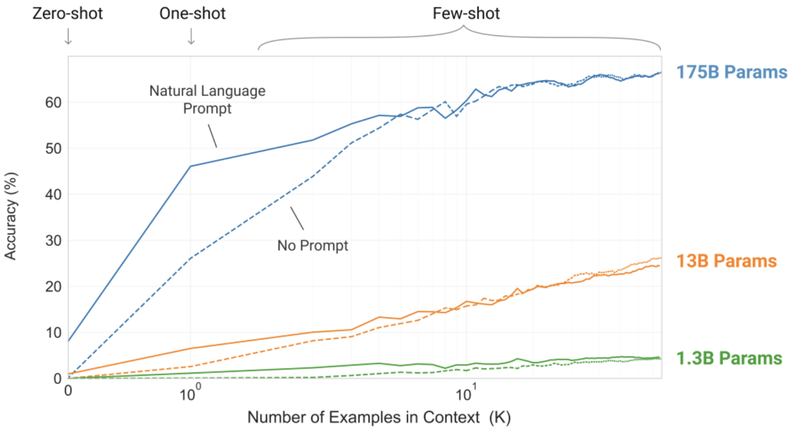
- How is this possible? Scale (in *parameters* and *training data*).

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

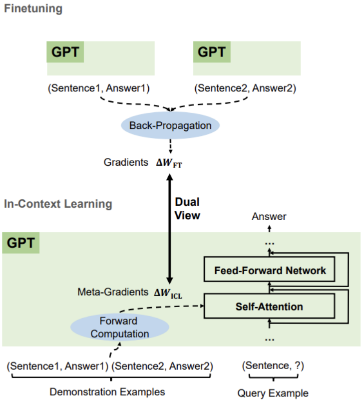
GPT-3 and the Miracle of Scale

- Something very special happens at over a hundred billion parameters.



WTF is going on here?

- How does GPT-3 (175B) do this without ever performing a gradient update?



Dai et al, "Why Can GPT Learn In-Context? Language Models Secretly Perform Gradient Descent as Meta-Optimizers."

GPT-X is Not ChatGPT

ChatGPT and Conversational Agents

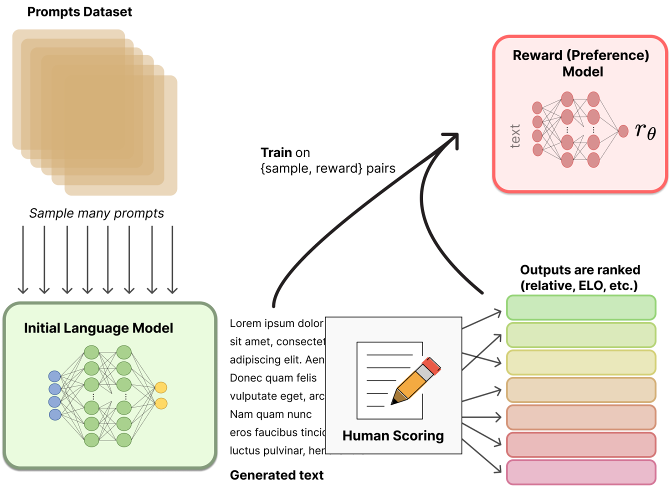
- GPT-3 is, in the end, a very large but still autoregressive model.
- It can generate diverse and compelling text from human prompts
- However, what makes a good text is hard to define and is subjective and context-dependent
- As the examples we have seen show, ChatGPT can:
  - Write stories that emulate styles and cite references (that may not exist!).
  - Produce executable code snippets that (probably?) do what the prompt asks.
  - (Attempt to) verify the truthfulness of what it produces.
- Writing a loss function to capture these characteristics is intractable, and our best language models are only trained for next token prediction.
- How does ChatGPT manage to do this?

Reinforcement Learning to the Rescue

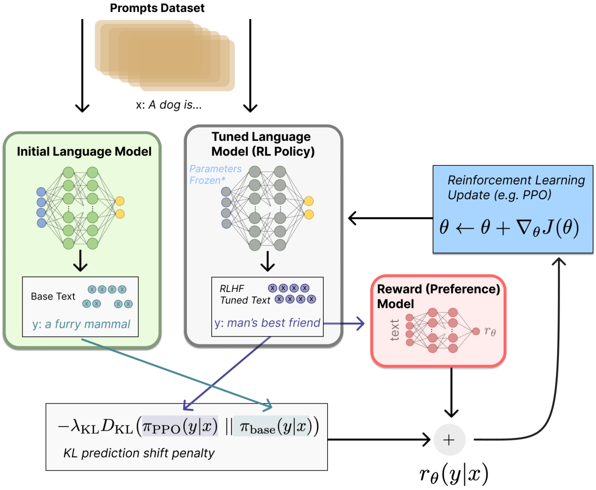
- OpenAI uses reinforcement learning to fine-tune their model to support high-quality interactions.
- They treat the LLM as an agent that:
  1. Observes the state – a context and possibly a prompt.
  2. Produces an action – the answer to the current contextualized prompt.
- It can then be trained with any reinforcement learning algorithm – they use Proximal Policy Optimization (PPO).
- This neatly sidesteps the need to define a supervised loss function.
- But... Don't we still need some sort of reward?

Schulman et al, "Proximal Policy Optimization Algorithms."

Inverse Reinforcement Learning to the Rescue!



Reinforcement Learning from Human Feedback



Discussion

Leftovers

- This course does not intend to be a comprehensive introduction to all of Machine Learning.
- There are many topics that could have been included, given more time.
- My hope is that the fundamentals you have acquired here are enough to allow you to acquire new skills and knowledge on your own.

Leftover: Unsupervised Learning

- Some techniques from unsupervised learning are great to have in your toolbox:
    - KMeans Clustering: Allows you to discover structure in unlabeled data.
    - Principal Component Analysis (PCA): Allows you to map high-dimensional data onto the principal axes of variation. Useful for dimensionality reduction and noise removal.
    - Gaussian Mixture Models (GMMs): Allows you to discover structure in unlabeled data and explain it with a statistical model.

Leftover: Recurrent Neural Networks (RNNs)

- What if our data has a temporal dimension of variable size?
- We already saw (very briefly) how self-attention networks (Transformers) can handle inputs of this type.
- Can we not apply traditional neural networks?
- The answer is yes (sort of):
    - A recurrent network is one that has loops (i.e. it is not a feed-forward network).
    - But wait, can we still use backpropagation is a graph that is not a DAG?
    - Yes: Back Propagation through Time (BPTT)

Leftover: Generative Models, Few-shot Learning, Meta-learning, Continual Learning, ...

- Well, you get the idea...
- When all else fails:
    - What is the input space $\mathcal{X}$?
    - What is the output space $\mathcal{Y}$?
    - What is the family of functions $\mathcal{H}$ that *makes sense* for my problem?
    - What is the loss function $\mathcal{L}$ that also *makes sense*?
    - What data $\mathcal{D}$ do I have to learn from?