Deep Learning

16 Transformer Applications

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2024

Transformer Layer - Powerful Building Block

1. Highly flexible building block $\rightarrow$ powerful models

Transformer Layer - Powerful Building Block

1. Highly flexible building block $\rightarrow$ powerful models
2. E.g., Large Language Models (LLMs)

Transformer Applications - NLP

Transformer Applications - NLP

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Three broad configurations - based on the form of i/p and o/p

Transformer Applications - NLP

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Sequential input to a single variable output (Transformer acts as an 'Encoder')

Transformer Applications - NLP

1. Sequential input to a single variable output (Transformer acts as an 'Encoder')
    - E.g., Sentiment classification

Transformer Applications - NLP

1. A single vector as input and a sequence as output (Transformer acts as a ‘Decoder’)

Transformer Applications - NLP

1. A single vector as input and a sequence as output (Transformer acts as a 'Decoder')
   - E.g., Caption generation from an image

Transformer Applications - NLP

1. Sequence-to-Sequence processing tasks

Transformer Applications - NLP

1. Sequence-to-Sequence processing tasks
   - E.g., Machine Translation

Decoder Transformers

1. Can be used as ‘Generative Models’

Decoder Transformers

1. Can be used as ‘Generative Models’
2. E.g., GPT (Generative Pre-trained Transformer)

Decoder Transformers

1. Can be used as 'Generative Models'
2. E.g., GPT (Generative Pre-trained Transformer)
3. Goal: use the transformer architecture to construct an 'Autoregressive' model

Decoder Transformers

1. Can be used as 'Generative Models'
2. E.g., GPT (Generative Pre-trained Transformer)
3. Goal: use the transformer architecture to construct an 'Autoregressive' model
4. $p(x_n/x_1, x_2, \ldots, x_{n-1})$

Decoder Transformers - GPT

1. Stack of transformer layers

Decoder Transformers - GPT

1. Stack of transformer layers
2. i/p - $x_1, x_2, \ldots, x_N$ each of D dimensions

Decoder Transformers - GPT

1. Stack of transformer layers
2. i/p - $x_1, x_2, \ldots, x_N$ each of D dimensions
3. o/p - $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N$

Decoder Transformers - GPT

1. Stack of transformer layers
2. i/p - $x_1, x_2, \ldots, x_N$ each of D dimensions
3. o/p - $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N$
4. Each o/p token needs to represent a probability distribution over the dictionary (say, K words)

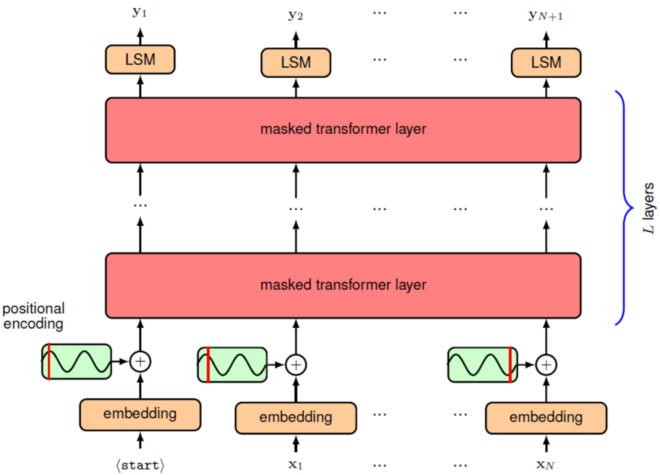
Decoder Transformers - GPT

1. Linear transformation of o/p tokens with $\mathbf{W^{(p)}}$ (dimensions - $K \times D$)

Decoder Transformers - GPT

1. Linear transformation of o/p tokens with $\mathbf{W^{(p)}}$ (dimensions - $K \times D$)
2. $\mathbf{Y} = \text{Softmax}(\tilde{\mathbf{X}}\mathbf{W^{(p)}})$

Decoder Transformers



Bishop's Book

Decoder Transformers

1. Can be trained over a large corpus of unlabelled text

Decoder Transformers

1. Can be trained over a large corpus of unlabelled text
2. Self-supervised approach

Decoder Transformers

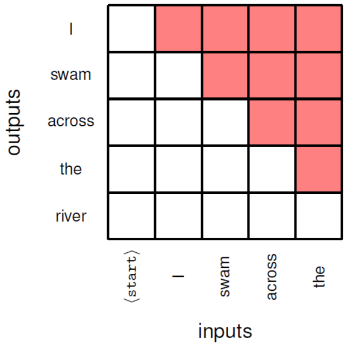
1. Can be trained over a large corpus of unlabelled text
2. Self-supervised approach
3. Predicting x_{n+1} from an input of $x_1, x_2, \ldots, x_{n-1}$

Decoder Transformers

1. Employs 'Masked' or 'Causal' attention

Decoder Transformers

1. Employs 'Masked' or 'Causal' attention
2. Sets the attention weights of all the 'later' tokens



Bishop's Book

Encoder Transformers

1. Take sequences as input and produce fixed-length vectors

Encoder Transformers

1. Take sequences as input and produce fixed-length vectors
   - E.g., class label (sentiment) as output

Encoder Transformers

1. Take sequences as input and produce fixed-length vectors
   - E.g., class label (sentiment) as output
2. E.g., BERT (Bidirectional Encoder Representations from Transformers)

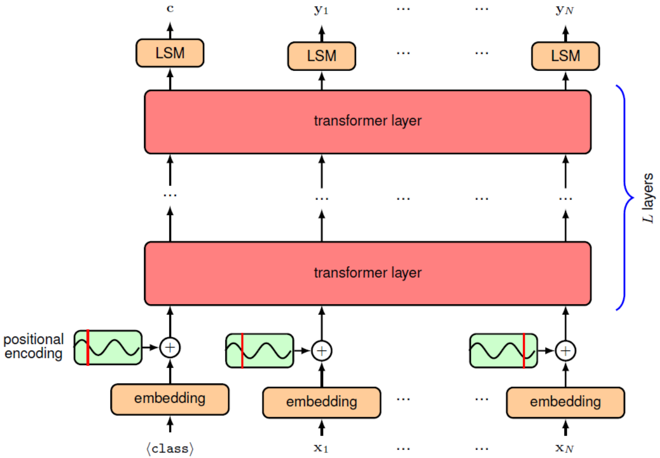
Encoder Transformers

1. Take sequences as input and produce fixed-length vectors
   - E.g., class label (sentiment) as output
2. E.g., BERT (Bidirectional Encoder Representations from Transformers)
3. Goal is to pre-train a language model using a large corpus of text

Encoder Transformers

1. Take sequences as input and produce fixed-length vectors
   - E.g., class label (sentiment) as output
2. E.g., BERT (Bidirectional Encoder Representations from Transformers)
3. Goal is to pre-train a language model using a large corpus of text
   - Then, to fine-tune it for a broad range of downstream tasks

Encoder Transformers



Bishop's Book

Decoder Transformers

1. First token of every input is a special token $< class >$

Decoder Transformers

1. First token of every input is a special token $< class >$
2. O/p of this is ignored during pre-training

Decoder Transformers

1. First token of every input is a special token $< class >$
2. O/p of this is ignored during pre-training
3. Pre-training goal is to predict the missing tokens

Decoder Transformers

1. A random 15% of the tokens are replaced with $< mask >$ and the training predicts them

Decoder Transformers

1. A random 15% of the tokens are replaced with $< mask >$ and the training predicts them
2. The cat <mask> sleeping on the <mask> next to the sofa.

Decoder Transformers

1. A random 15% of the tokens are replaced with $< mask >$ and the training predicts them
2. The cat <mask> sleeping on the <mask> next to the sofa.
3. Model should predict is and floor at 3 and 7 nodes respectively

Decoder Transformers

1. 'Bidirectional' $\leftarrow$ model can access words both before and after the masked word

Decoder Transformers

1. 'Bidirectional' $\leftarrow$ model can access words both before and after the masked word
2. Only a fraction of tokens act as labels

Decoder Transformers

1. 'Bidirectional' $\leftarrow$ model can access words both before and after the masked word
2. Only a fraction of tokens act as labels
3. Doesn't generate sequences

Decoder Transformers

1. After the pre-training, the Encoder model can be finetuned

Decoder Transformers

1. After the pre-training, the Encoder model can be finetuned
2. E.g., Tex classification: $< class >$ token is used for prediction

Decoder Transformers

1. After the pre-training, the Encoder model can be finetuned
2. E.g., Tex classification: $< class >$ token is used for prediction
3. A new layer (LSM in the figure) predicts the probability distribution over the dictionary

Sequnce-to-Sequence Transformers

1. Combines an encoder with a decoder

Sequnce-to-Sequence Transformers

1. Combines an encoder with a decoder
2. E.g., machine translation from English to French

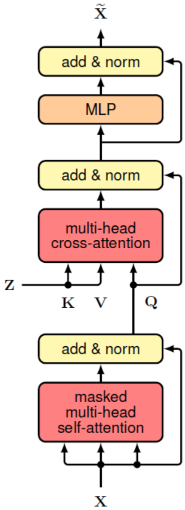
Sequnce-to-Sequence Transformers

1. Combines an encoder with a decoder
2. E.g., machine translation from English to French
3. Decoder model generates the token sequence corresponding to the French o/p

Sequnce-to-Sequence Transformers

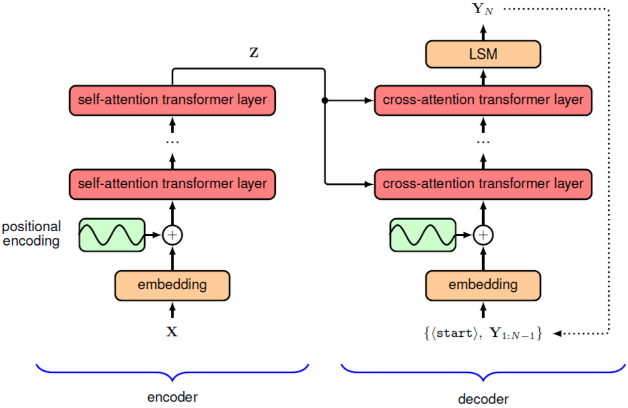
1. Combines an encoder with a decoder
2. E.g., machine translation from English to French
3. Decoder model generates the token sequence corresponding to the French o/p
4. Conditioned on the entire input sequence corresponding to the English sentence $\rightarrow$ 'cross attention'

Sequnce-to-Sequence Transformers



Bishop's Book

Sequnce-to-Sequence Transformers



Bishop's Book

LLM - Large Language Models

LLM

1. Recent development in ML and NLP

LLM

1. Recent development in ML and NLP
2. ‘Large’ $\rightarrow$ Billions of parameters

LLM

1. Recent development in ML and NLP
2. 'Large' $\rightarrow$ Billions of parameters
3. Large datasets and Powerful GPUs

LLM

1. Recent development in ML and NLP
2. 'Large' $\rightarrow$ Billions of parameters
3. Large datasets and Powerful GPUs
4. Unlike earlier language models, these are self-supervised first on large corpuses then finetuned with (small) labeled data

LLM

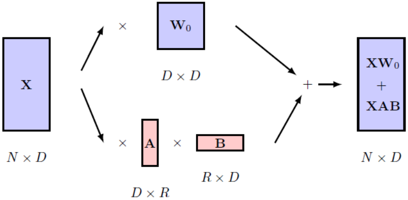
1. 'Foundation Model' $\leftarrow$ A model with broad capabilities that can be subsequently fine-tuned for specific tasks

LLM- Finetuning

1. An Efficient approach to fine-tuning is called low-rank adaptation (LoRA)

LLM- Finetuning

1. An Efficient approach to fine-tuning is called low-rank adaptation (LoRA)
2. A trained overparameterized model has a low intrinsic dimensionality with respect to fine-tuning



Bishop's Book

LLM - Finetuning

1. With their growing size, the need for fine-tuning is reducing

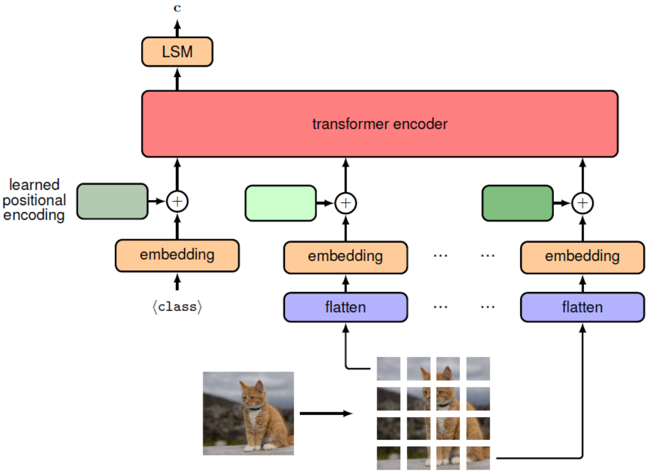
LLM - Finetuning

1. With their growing size, the need for fine-tuning is reducing
2. Generative language models are now able to solve a broad range of tasks through text-based interaction (prompt)

LLM - Finetuning

1. With their growing size, the need for fine-tuning is reducing
2. Generative language models are now able to solve a broad range of tasks through text-based interaction (prompt)
3. Fine-tuning large language models through human evaluation of generated output (e.g., reinforcement learning through human feedback or RLHF)

Transformers - Computer Vision



Bishop's Book