

Machine Learning for Economists

Class 16: Transformers - Encoder Only or Decoder Only?

葛雷

中国人民大学 - 数量经济

2025 年 5 月 28 日



中國人民大學
RENMIN UNIVERSITY OF CHINA

Previous class review

Self-Attention to LLM

Encoder-Only Transformer

Decoder-Only Transformer

RAG

Fine Tuning

Appendix

Previous class review

Self-Attention to LLM

Encoder-Only Transformer

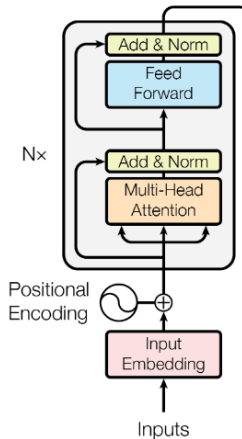
Decoder-Only Transformer

RAG

Fine Tuning

Appendix

Previously: Transformer



Previously: Self-Attention

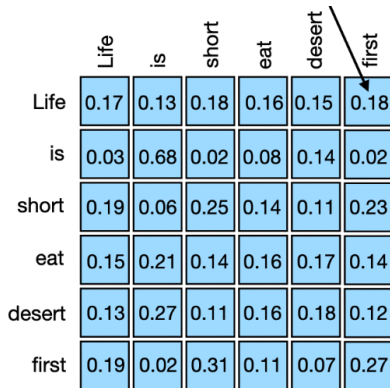
- Q K V matrices:

$$\text{Self-Attention Output} = \text{Attention Weights} \cdot V = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) \cdot V$$

- word vector X to contextualized word vector $X_{\text{contextualized}}$:
 $X_{\text{contextualized}} = \text{Self-Attention}(X)$

Previously: Attention Weights Matrix

Attention Weights Matrix for sequence "Life is short eat dessert first":



	Life	is	short	eat	dessert	first
Life	0.17	0.13	0.18	0.16	0.15	0.18
is	0.03	0.68	0.02	0.08	0.14	0.02
short	0.19	0.06	0.25	0.14	0.11	0.23
eat	0.15	0.21	0.14	0.16	0.17	0.14
dessert	0.13	0.27	0.11	0.16	0.18	0.12
first	0.19	0.02	0.31	0.11	0.07	0.27

Previously: Correlation Matrix vs Attention Matrix

- They are paralleling concepts
- Correlation Matrix: $S = \frac{1}{n} X_{\text{standardized}} X_{\text{standardized}}^T$
 - correlation between each variable
- Attention Matrix : $QK^T = (XW_Q)(XW_K)^T$
 - attention between each token in paragraph

Previously: word vector to contextualized word vector

- However about we give a percentage attention weights represent **霸**'s dependence on other words
- Assume: 你 (10%) 是 (5%) 个 (5%) 学 (50%) 霸 (30%)
- **霸 with contextual meaning** = original 霸 + 你 (10%) + 是 (5%) + 个 (5%) + 学 (50%) + 霸 (30%)

Bingo, this is self-attention

Huggingface: connected with genius around world

📝 **Trending** last 7 days

All **Models** Datasets Spaces

∞ meta-llama/Llama-3.3-70B-In...
📄 Text Generation • • ⬇ 182k • ⚡ • ❤ 1

👤 Datou1111/shou_xin
📄 Text-to-Image • • ⬇ 20.7k • ⚡ • ❤ 49

🇨🇳 tencent/HunyuanVideo
📄 Text-to-Video • U • • ⬇ 4.92k • ❤ 1.09k

✈️ CohereForAI/c4ai-command-r7...
📄 Text Generation • U • • ⬇ 2.84k • ❤ 228

🌲 black-forest-labs/FLUX.1-dev
📄 Text-to-Image • • ⬇ 1.38M • ⚡ • ❤ 7.

🦋 deepseek-ai/DeepSeek-V2.5-1...



Hugging Face

🔍 Search models, dataset

Tasks

Libraries Datasets Languages Licenses

Other

🔍 Filter Tasks by name

Multimodal

🗣️ Audio-Text-to-Text 📄 Image-Text-to-Text

📺 Visual Question Answering

📄 Document Question Answering

📄 Video-Text-to-Text 🔄 Any-to-Any

Computer Vision

📷 Depth Estimation 📄 Image Classification

📷 Object Detection 📄 Image Segmentation

📄 Text-to-Image 📄 Image-to-Text

📄 Image-to-Image 📄 Image-to-Video

Previous class review

Self-Attention to LLM

Encoder-Only Transformer

Decoder-Only Transformer

RAG

Fine Tuning

Appendix

How Self-Attention to LLM?

- ChatGPT, Bert, Llama, Gemini are all self-attention transformer model
- Self-Attention is core element of the Large Language Models (LLM)
- $\text{LLM} = \text{Self-Attention} + \text{Self-Attention} + \dots$
- But, how LLM do these amazing works? (generating articles, sentiment analysis, housing pricing, stock pricing and etc.)

How We Train Parameters in LLMs

- LLMs has different task targets Y (for example, chatgpt generating answers)
- Train the weights in matrices W_Q, W_K and W_V in the self-attention to achieve better Y (also the W_{emb} in the input embedding)
- Loss function: softmax and log loss

Recall: OLS also have target and loss, the target is Y and Loss is MSE, everything same

Encoder-Only Transformer vs Decoder-Only Transfer

- Difference: different task targets (Y)
- Encoder-Only transformer target: General Language Understanding (GLU task aka 文本理解)
- Decoder-Only transformer target: Natural Language Generation (NLG task aka 文本生成)

Special Tokens

LLM has special tokens for the training and inference propose:

MASK for masking

CLS for classification

PAD padding token

BOS Beginning of Sequence Token

EOS End of Sequence Token

Special Tokens: Example

- Before: "The quick brown fox jumps over the lazy dog while the energetic squirrel hops around the tall oak tree under the bright blue sky."
- After: "[CLS] The quick [MASK] fox jumps over the lazy dog while the energetic squirrel [MASK] around the tall [SEP] oak tree under the bright blue sky. [PAD] [PAD] [PAD]"

Previous class review

Self-Attention to LLM

Encoder-Only Transformer

Decoder-Only Transformer

RAG

Fine Tuning

Appendix

Encoder-Only Transformer

- BERT is a popular encoder-only transformer for financial and economic contextual analysis
<https://arxiv.org/abs/1810.04805> Read it
- Encoder-only models are designed to understand or "encode" input sequence (context)
- They excel at tasks that require deep comprehension of context (GLU)

What Bert can do for you?

- Bert, RoBERTa, FinBert (fine-tuned already) (Click Me)
- Extract contextual features for housing pricing, stock pricing
- Sentiment Analysis
- Document Classification
- NER: extract useful information from tons of documentations (let me give you a example)

How Bert works during prediction?

1. Input Sequence: "[CLS] The movie was great [SEP]"
2. Transformer Output: $[CLS]_{contextualized}$, it is contextualized token vector of [CLS] after passing our transformer
3. **Pooler Output:** pass the $[CLS]_{contextualized}$ representation through a dense layer with Tanh activation and get Pooler Output
4. Softmax output: apply dense layer and softmax to get the probability of each class (e.g., positive or negative sentiment)

How Bert works during prediction?

- $[CLS]_{contextualized}$ represents the contextual meaning of whole sequence (语义向量)
- Why $[CLS]_{contextualized}$ can represent meaning of whole article?
- Please follow me to Bert Training for the reason

Training: How did BERT Trained?

- Training target I: Masked Language Modeling (MLM)
- Training target II: Next Sentence Prediction (NSP)
- Target I + Target II \rightarrow deep understanding of context

Training: Masked Language Modeling (MLM)

1. Masking Tokens: During training, some input tokens are randomly masked. For example, "I have watched this movie and it was awesome" becomes "I have watched this [MASK] and it was awesome."
2. Predicting the Masked Tokens: The model predicts the masked tokens using the context from surrounding words, helping it learn bidirectional text representations.

Training: MLM example

Example Let's take a simple sentence to illustrate:

1. Input (X): "I have watched this [MASK] and it was awesome."
2. Output (Y): The model predicts the masked word "movie" based on the context provided by the other words in the sentence.
3. Math: $\text{Transformer}(X) + \text{Multiple Classification} = Y$

Training: Next Sentence Prediction (NSP)

- Example: [CLS] Sentence A [SEP] Sentence B [SEP]
- Predict whether or not, **Sentence B** is the Next Sentence of **Sentence A**.

Training: Next Sentence Prediction (NSP)

- True case: [CLS] The quick brown fox jumps over the lazy dog. [SEP] The dog is still sleeping. [SEP]
- False case: [CLS] The quick brown fox jumps over the lazy dog. [SEP] Python is fantastic. [SEP]

Training Results of MLM and NSP

Whole Article $\rightarrow [CLS]_{contextualized}$

整篇文章 \rightarrow 一个语义向量

Decoder-Only Transformer

- ChatGPT, Llama (套壳之王), Qwen, GLM, Gemini, Kimi, all Decoder-Only Transformer
- Good at generating answers, articles, financial report, reports
- Good tool to learn math and programming
- Please follow me to Copilot

ChatGPT and Kimi: Tool to learn new

- Open ChatGPT or Kimi or Copilot, 让它帮助你完成期末考试
- Learn **Optuna** codes: "Give me python template of using optuna to tune hyperparameters in Xgboost"
- Learn **Optuna** math: "Give me math about optuna"
- generative-LLM is awesome! But (how does it works?)

Prediction: autogressive-text-prediction

- Model generates one token (word) at a time
- Each token being predicted based on the previously generated tokens
- So, this is a sequential autogressive process

Prediction: autogressive-text-prediction

- $Input_0 = \text{"What is Python?"}$;
 $Output_0 = \text{Transformer}(Input_0) = \text{"a"}$
- $Input_1 = \text{"What is Python? a"}$;
 $Output_1 = \text{Transformer}(Input_1) = \text{"programming"}$
- $Input_2 = \text{"What is Python? a programming"}$;
 $Output_2 = \text{Transformer}(Input_2) = \text{"language"}$
- $Final\ Output = \text{"What is Python? a programming language"}$

note: the question or input is also called **prompt**

Training: sequence should follow causality

- So, how to train a transformer to predict future tokens in sequence?
- Model should not "see" future tokens during training
- Each token should only give attention to previous tokens
- In this way, the trained model can predict the future tokens in sequence

Masked Self-Attention

- The ordinary self-attention can not help here. Because, each token will attention all future tokens (Data Leakage)
- So, we turn to **Masked Self-Attention**
- Masked Self-Attention: each token only attention the tokens before it

Masked Self-Attention

Attention weight between the tokens corresponding to “Life” and “short”

	Life	is	short	eat	desert	first
Life	0.17	0.13	0.18	0.16	0.15	0.18
is	0.03	0.68	0.02	0.08	0.14	0.02
short	0.19	0.06	0.25	0.14	0.11	0.23
eat	0.15	0.21	0.14	0.16	0.17	0.14
desert	0.13	0.27	0.11	0.16	0.18	0.12
first	0.19	0.02	0.31	0.11	0.07	0.27

In the row for corresponding to “Life”, mask out all words that come after “Life”

	Life	is	short	eat	desert	first
Life	0.17	0.13	0.18	0.16	0.15	0.18
is	0.03	0.68	0.02	0.08	0.14	0.02
short	0.19	0.06	0.25	0.14	0.11	0.23
eat	0.15	0.21	0.14	0.16	0.17	0.14
desert	0.13	0.27	0.11	0.16	0.18	0.12
first	0.19	0.02	0.31	0.11	0.07	0.27

Previous class review

Self-Attention to LLM

Encoder-Only Transformer

Decoder-Only Transformer

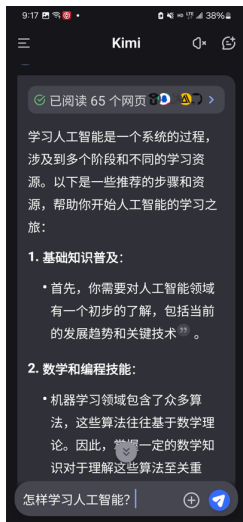
RAG

Fine Tuning

Appendix

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a technique that enhances the model's output by incorporating information from external knowledge bases



Advantages of RAG

- New Knowledge: RAG retrieves information from authoritative sources, ensuring accurate and up-to-date responses (ChatGPT use it)
- Accuracy: By referencing external databases, RAG enhances the credibility and relevance of content
- Cost-Effective: Allows continuous updates and domain-specific integration without retraining the model

How RAG works in math

- Input Question (prompt) \rightarrow input vector $X_{contextual}$
- Many External Articles \rightarrow vectors database
- Use input contextual vector $X_{contextual}$ to match similar vectors from database to find the relevant articles
- New Input Question (prompt) = Input Question + Relevant articles
- Put the New Input Question to the LLM for answer generating

What RAG can do for you?

- Credit Risk: Analyzes borrower profiles using diverse data sources to refine credit scoring and reduce default risks
- Report Writing: Synthesizes documents and reports from the specialized set of documents
- Market Analysis: Aggregates data from news, social media, and reports to assess market sentiment, aiding trading strategies

What RAG can do for you?

- Decision-Making: Integrates real-time data with generative models to provide accurate, context-aware insights
- Personalized Service: RAG retrieves customer-specific data to provide tailored responses, enhancing interaction accuracy and relevance

RAG coding

- RAG is easy to use
- Please turn to the codes

Fine Tuning (Transfer Learning)

- Fine tuning pretrained models such as Llama, Bert or ChatGPT to do specific tasks
- Using smaller specific dataset to retrain the **weights** in transformer model (Yes, still gradient method)
- better performance than RAG, but more costly

Popular python Dev: transformer Trainer, Unsloth Trainer, FlashAttention

Fine Tuning (PEFT)

Costly? How to tune efficiently with small labeled data?

- Freeze Layers: only train layers near the output and freeze other layers
- LORA (Low rank adaptation) (线性代数有用啊)
- P-tuning: only the input layer tuning (也许是你的首选)

Previous class review

Self-Attention to LLM

Encoder-Only Transformer

Decoder-Only Transformer

RAG

Fine Tuning

Appendix

Appendix: LLM techs

- Reinforcement Learning from Human Feedback (RLHF)
- Pre-training (majority data without label)
- Model Distill

Fine-Tuning Coding

Please turn to our llm codes

Appendix: Special Tokens

CLS (Classification Token): Often used at the beginning of the input sequence to aggregate information for classification tasks. This token is particularly important in models like BERT.

MASK (Mask Token): Used in tasks like Masked Language Modeling (MLM) where certain tokens in the input are masked and the model is trained to predict them based on the context provided by the surrounding tokens.

SEP (Separator Token): Used to separate different segments of text, such as different sentences or parts of a conversation. This token helps the model distinguish between different parts of the input...

Appendix: Special Tokens

PAD (Padding Token): Used to pad sequences to a uniform length, ensuring that all inputs in a batch have the same length. This is crucial for efficient batch processing.

BOS (Beginning of Sequence Token): Indicates the start of a sequence. This is often used in text generation tasks to signal the beginning of the generated text.

EOS (End of Sequence Token): Indicates the end of a sequence. This token helps the model understand where the input or generated text ends.

Appendix: Advanced Method (plz use this way)

plz use this way

- Huggingface Transformers
- Pytorch

Appendix: Easy Method

- Ollama (use model)
- Llama Factory (fine tuning)
- Anything LLM (RAG)

Reference

1. Huggingface
2. Wikipedia
3. w3schools
4. geeksforgeeks
5. Kaggle
6. Wikipedia
7. ChatGPT
8. DeepSeek
9. <https://github.com/HandsOnLLM/Hands-On-Large-Language-Models>