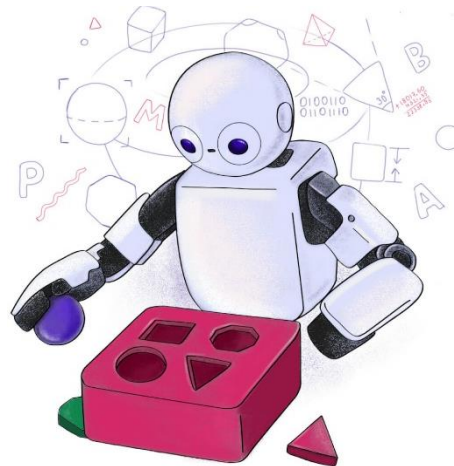


TP558 - Tópicos avançados em Machine Learning: *LoRA*



Inatel

Pedro Guerrato - 967
pedro.guerrato@Inatel.br,
pguerrato@gmail.com

Introdução

Neural Processing Language (NLP) sempre foi um desafio para a computação e AI.

Mesmo assim, no decorrer dos anos muitos avanços foram conquistados, por isso antes de entender o LoRA, vamos falar revisitar alguns pontos da historia do NLP

Introdução

Symbolic

- Sistema baseado em regras
- Exemplo: Modelo ELIZA

Exemplo:

- Lista de Regras de pronomes:
 - Eu → Você
- Lista de Sentimentos:
 - Feliz
 - Triste
 - Alegre
 - Entediado
 - ...



1950

Introdução

Eu estou feliz hoje.

Por que você está feliz hoje?



1950

Introdução

Mas ainda havia limitações:

Não funcionam se não houver as regras explícitas

Usuário: Nota baixa em TP558 me deixa triste.

ELIZA: Por que você está triste?



1950

Introdução

Statistical

- Baseado em distribuições de probabilidade
- Exemplo: Modelo N-Grams



1990

Introdução

Exemplo:

Eu gosto de aprender _____

Exemplo:

- a cozinhar (0.8);
- programação (0.7);
- novas línguas (0.68);
- ...

1990



Introdução

Mas ainda havia limitações:

- Não entendem contexto longo nem significado, somente estatística;
- O modelo desconhece o **sujeito**;
- Se algo não está no treino, a sua probabilidade é **zero**;
- **Semântica ausente**;



1990

Introdução

Exemplo:

Esperado: Encerramos o ciclo 2 de TP558.

Saídas:

- Encerramos o ciclo 2 de ciclo 2;
- Encerramos a matéria de TP558;
- Encerramos o ciclo 2 de bicicletas.



1990

Introdução

Neural

- Baseadas em similaridade semântica
- Redes neurais introduziram representações de palavras distribuídas
- Introduziram as ***embeddings***
- Exemplo: Word2Vec, Seq2Seq, LSTMs

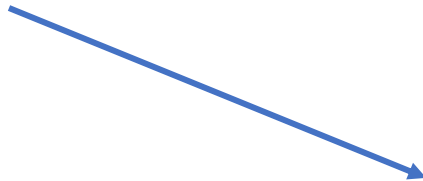


2013

Introdução

Exemplo:

Rei - Homem + Mulher \approx Rainha



[0.123, 0.233, 0.452, 0.986]



2013

Introdução

Mas ainda havia limitações:



2013

Introdução

Termo: Barracuda



=



2013

Introdução

Attention

- Modelos que “focam” em partes relevantes das sequências de entrada
- Exemplo: Bahdanau, Luong e **Transformers**



2014

Introdução

Exemplo:

Tiramos nota máxima em TP558. Isso é incrível!

Tirar nota máxima

2014



Introdução

Mas ainda havia limitações:

- Mesmo resolvendo os problemas clássicos, introduziu problemas de performance
- Poucos dados para treinamento;
- Modelos em geral unidirecional



2014

Introdução

Transfer Learning

- Uso de pré-treinamento + *fine-tuning* context bidirecional
- Exemplo: BERT



2018

Introdução

Exemplo:

Roma é a [MASK] da Itália.

Capital

2018



Introdução

Mas ainda havia limitações:

- Não é generativo;
- *Fine-tuning* em datasets pequenos pode ser catastrófico;
- Original BERT suporta **512 tokens max**;
- Não se adapta facilmente a vocabulários específicos (ex.: medicina, engenharia...)



2018

Introdução

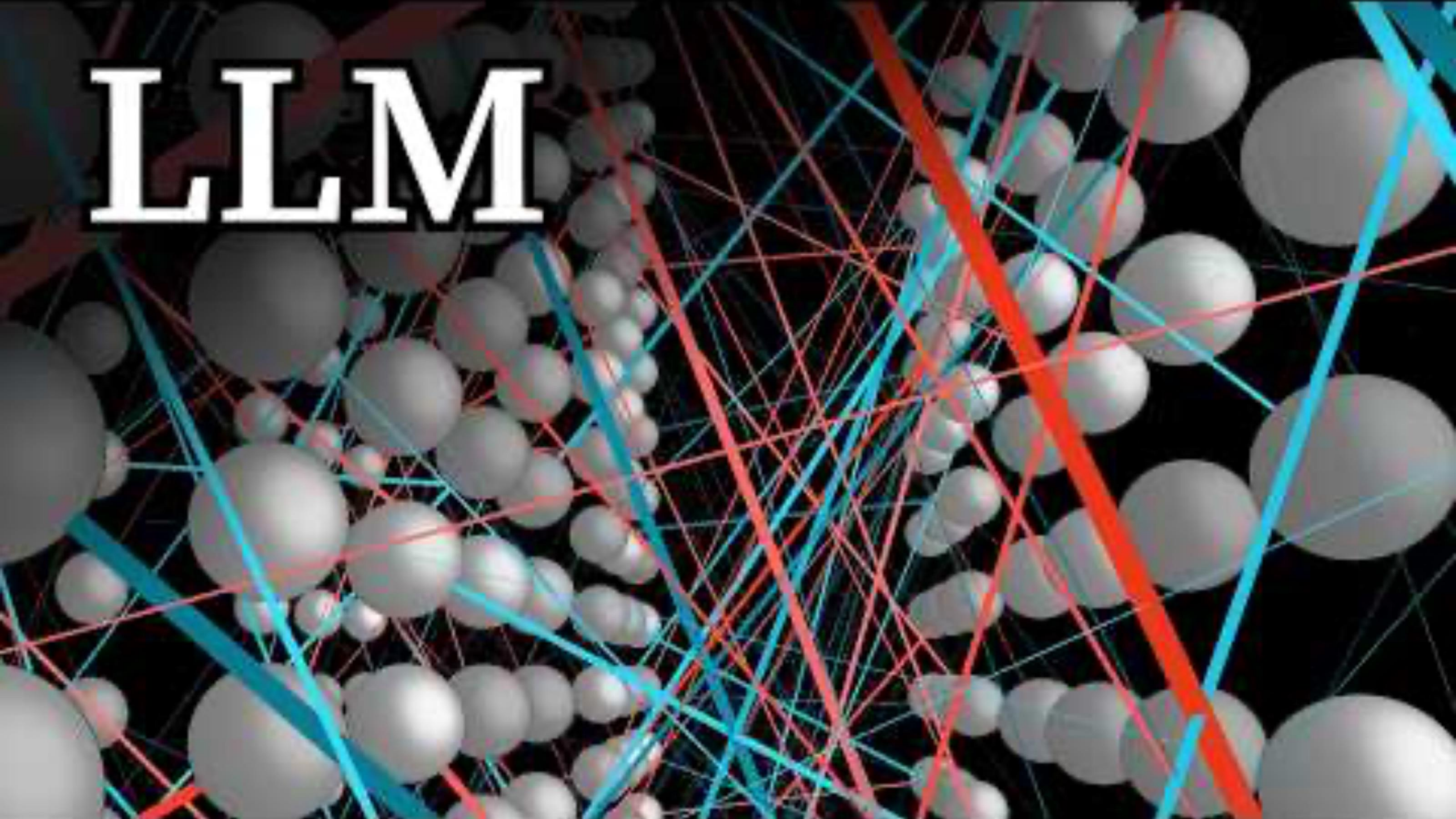
LLMs

- Uso do massive autoregressive Transformers
- Exemplo: GPT-3



Hoje

LLM



Introdução

Mas ainda há limitações:

- Proibitivamente pesado em vários ambientes;
- Alto custo computacional e treinamento lento;
- **Quando em domínios (vocabulários) específicos, ainda há muitos erros (alucinações);**



Hoje

Introdução

Usando *fine-tuning* aplicados ao LLM, é possível:

- Ajustar tons de resposta;
- **Especializar o modelo para aplicações específicas (jurídico, médico, financeiro, técnico, etc.);**
- **Melhorar em tarefas específicas (refinado para classificação de texto, sumarização, geração de código, extração de entidades, tradução técnica, etc.);**
- **Redução de Alucinação.**

Introdução

Usando *fine-tuning* tradicional aplicados ao LLM, é possível:

- Ajustar tons de resposta;
- **Especializar o modelo para aplicações específicas (jurídico, médico, financeiro, técnico, etc.);**
- **Melhorar em tarefas específicas (refinado para classificação de texto, sumarização, geração de código, extração de entidades, tradução técnica, etc.);**
- **Redução de Alucinação.**

Introdução

Só que...

Ainda há limitações!

Introdução

No *fine-tuning* tradicional:

- É preciso atualizar quase todos esses parâmetros
Requer **re-treinar todos os parâmetros** do modelo (milhões ou bilhões).
- Risco de *Overfitting*
Como todos os pesos mudam, o modelo pode perder parte do conhecimento geral aprendido no pré-treinamento (catastrophic forgetting).

Introdução

No *fine-tuning* tradicional:

- Baixa Reusabilidade

Se você fizer *fine-tuning* para jurídico e depois quiser um para saúde, precisa guardar dois modelos inteiros (vários GB cada) e **não dá para compartilhar só o “ajuste”, você precisa compartilhar tudo.**

- Escalabilidade Ruim

Domínio jurídico → 1 modelo completo.

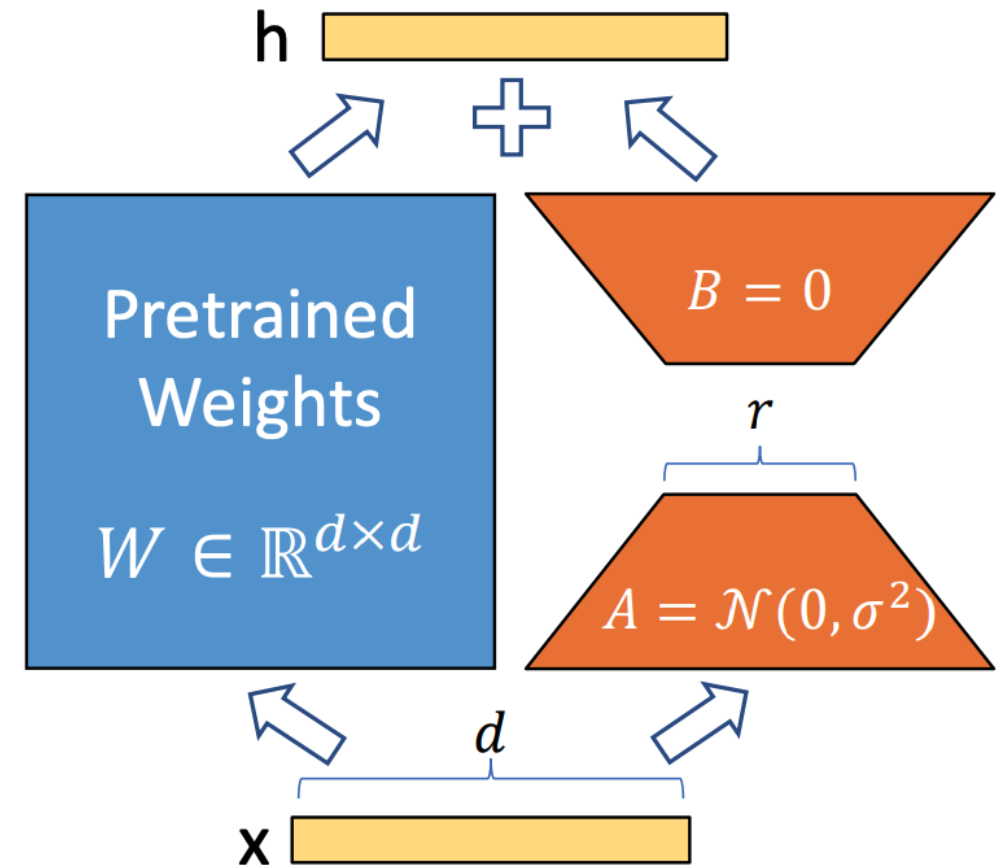
Domínio médico → outro modelo completo.

Domínio financeiro → outro modelo completo.

Introdução

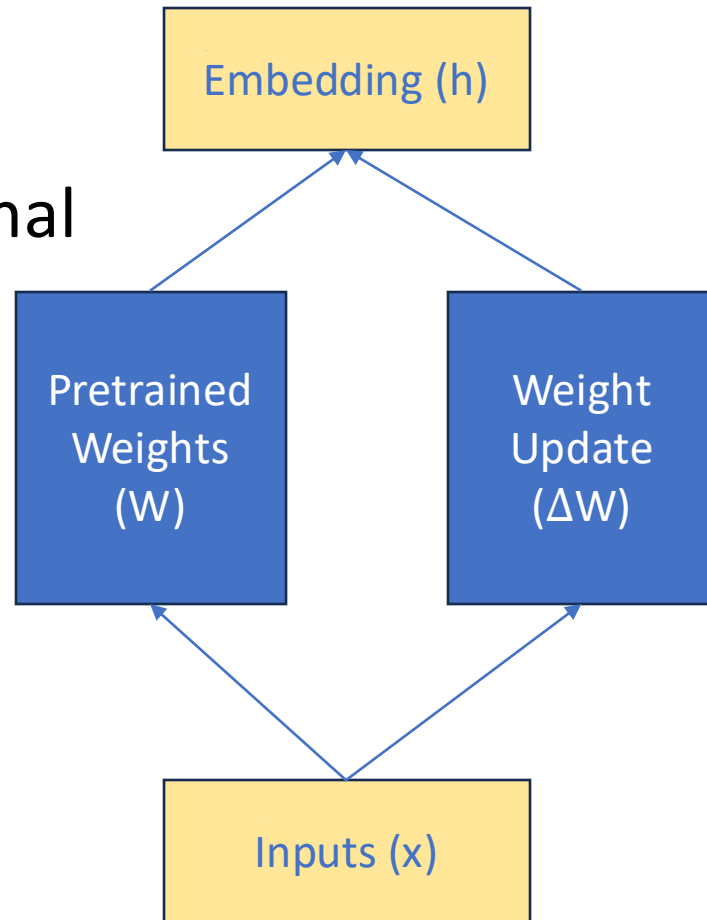
Então Hu et al. propuseram uma nova abordagem de *fine-tuning* chamado Low-Rank (LoRA).

A partir do congelamento de modelos pré-treinados e injeção de *ranks* em cada camada da arquitetura *Transformer*.



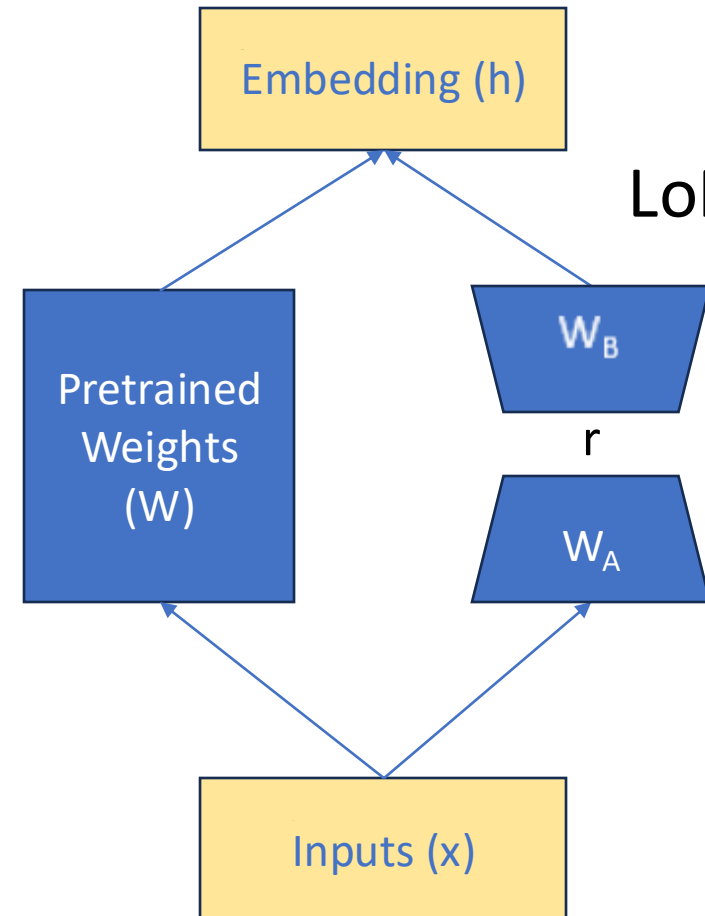
Fundamentação teórica

Tradicional



X

LoRA



Fundamentação teórica

Então Hu et al. propuseram uma nova abordagem de *fine-tuning* chamado Low-Rank (LoRA).

A partir do congelamento de modelos pré-treinados e injeção de *ranks* em cada camada da arquitetura *Transformer*.

Arquitetura e funcionamento

$$\max_{\Theta} \sum_{(x,y) \in \mathcal{Z}} \sum_{t=1}^{|y|} \log \left(p_{\Phi_0 + \Delta\Phi(\Theta)}(y_t \mid x, y_{<t}) \right)$$

Arquitetura e funcionamento

Onde $\Delta\Phi$ é explicado por:

$$h = W_0x + \Delta Wx = W_0x + BAx$$

Onde:

$$B \in \mathbb{R}^{d \times r} \quad \mathbb{R}, A \in \mathbb{R}^{r \times k}$$

Treinamento e otimização

Toda matriz tem um “Rank” (r), que é a quantidade de colunas linearmente independentes que a matriz tem.

Treinamento e otimização

$$\Delta W = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \end{bmatrix}, \in \mathbb{R}^{d \times k}$$

d = 3 e k=3

Treinamento e otimização

$$\Delta W = \begin{bmatrix} 0_{x2} = & 0 & 0 \\ 1_{x2} = & 2 & 1 \\ 1_{x2} = & 2 & 2 \end{bmatrix}$$

Linearmente Dependente

Treinamento e otimização

$$\Delta W = \begin{bmatrix} 0 & x_2 = 0 & 0 \\ 1 & x_2 = 2 & 1 \\ 1 & x_2 = 2 & 2 \end{bmatrix}$$

Linearmente Dependente

Treinamento e otimização

Linearmente Independente

$$\Delta W = \begin{bmatrix} 0_{x2} = 0 & 0 \\ 1_{x2} = 2 & 1 \\ 1_{x2} = 2 & 2 \end{bmatrix}$$

Linearmente Dependente

The diagram illustrates the linear independence of the columns of the matrix ΔW . A blue arrow points from the text 'Linearmente Independente' to the second column of the matrix, which contains the values 0, 1, and 2. A red bracket is placed under the first column, which contains the values 0, 2, and 2, with the text 'Linearmente Dependente' written below it. The matrix is defined as $\Delta W = \begin{bmatrix} 0_{x2} = 0 & 0 \\ 1_{x2} = 2 & 1 \\ 1_{x2} = 2 & 2 \end{bmatrix}$, where the red text $x2$ and $=$ are part of the matrix's internal structure.

Treinamento e otimização

Portanto $r = 1$

Treinamento e otimização

Dessa forma é possível reduzir a matriz usando **decomposição**



Treinamento e otimização

$$\Delta W_x = BA \mid B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$$

$$\Delta W_x = BA = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \times [a_1 \quad a_2 \quad a_3]$$

$$(d \times r)(r \times k) = (3 \times 1)(1 \times 3)$$

Treinamento e otimização

Então se escolhido um rank muito baixo, reduz-se muito a dimensionalidade e se perde informação, pois implicitamente excluiu-se colunas linearmente independente;

Treinamento e otimização

E, ao contrário, escolhendo muito alto, mantem-se muitos parâmetros que são linearmente dependentes, o que gasta-se recursos computacionais.

Treinamento e otimização

Por isso, inicializa-se A a partir de distribuição Gaussiana e $B = 0$. (p.4)

Vantagens e desvantagens

Vantagens

- Eficiência de Treinamento
- Economia de Armazenamento
- Melhor Generalização
- Reutilização do Modelo Base

Desvantagens

- Limitação de Ajuste Completo
- Em alguns casos pode ser mais complexo na integração;
- Se treinado em uma língua, pode não performer tão bem se usado em outra

Exemplo(s) de aplicação

- [LoRA tuning – PEFT](#)
- [SDD-LawLLM](#)
- [Medical QA T5 LoRA Model](#)
- [MedAlpaca](#)
- [Image classification using LoRA](#)
- <https://flux-ai.io/flux-dev-lora/>

Comparação com outros algoritmos

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB _{base} (FT)*	125.0M	87.6	94.8	90.2	63.6	92.8	91.9	78.7	91.2	86.4
RoB _{base} (BitFit)*	0.1M	84.7	93.7	92.7	62.0	91.8	84.0	81.5	90.8	85.2
RoB _{base} (Adpt ^D)*	0.3M	87.1 \pm .0	94.2 \pm .1	88.5 \pm 1.1	60.8 \pm .4	93.1 \pm .1	90.2 \pm .0	71.5 \pm 2.7	89.7 \pm .3	84.4
RoB _{base} (Adpt ^D)*	0.9M	87.3 \pm .1	94.7 \pm .3	88.4 \pm .1	62.6 \pm .9	93.0 \pm .2	90.6 \pm .0	75.9 \pm 2.2	90.3 \pm .1	85.4
RoB _{base} (LoRA)	0.3M	87.5 \pm .3	95.1\pm.2	89.7 \pm .7	63.4 \pm 1.2	93.3\pm.3	90.8 \pm .1	86.6\pm.7	91.5\pm.2	87.2
RoB _{large} (FT)*	355.0M	90.2	96.4	90.9	68.0	94.7	92.2	86.6	92.4	88.9
RoB _{large} (LoRA)	0.8M	90.6\pm.2	96.2 \pm .5	90.9\pm1.2	68.2\pm1.9	94.9\pm.3	91.6 \pm .1	87.4\pm2.5	92.6\pm.2	89.0
RoB _{large} (Adpt ^P)†	3.0M	90.2 \pm .3	96.1 \pm .3	90.2 \pm .7	68.3\pm1.0	94.8\pm.2	91.9\pm.1	83.8 \pm 2.9	92.1 \pm .7	88.4
RoB _{large} (Adpt ^P)†	0.8M	90.5\pm.3	96.6\pm.2	89.7 \pm 1.2	67.8 \pm 2.5	94.8\pm.3	91.7 \pm .2	80.1 \pm 2.9	91.9 \pm .4	87.9
RoB _{large} (Adpt ^H)†	6.0M	89.9 \pm .5	96.2 \pm .3	88.7 \pm 2.9	66.5 \pm 4.4	94.7 \pm .2	92.1 \pm .1	83.4 \pm 1.1	91.0 \pm 1.7	87.8
RoB _{large} (Adpt ^H)†	0.8M	90.3 \pm .3	96.3 \pm .5	87.7 \pm 1.7	66.3 \pm 2.0	94.7 \pm .2	91.5 \pm .1	72.9 \pm 2.9	91.5 \pm .5	86.4
RoB _{large} (LoRA)†	0.8M	90.6\pm.2	96.2 \pm .5	90.2\pm1.0	68.2 \pm 1.9	94.8\pm.3	91.6 \pm .2	85.2\pm1.1	92.3\pm.5	88.6
DeB _{XXL} (FT)*	1500.0M	91.8	97.2	92.0	72.0	96.0	92.7	93.9	92.9	91.1
DeB _{XXL} (LoRA)	4.7M	91.9\pm.2	96.9 \pm .2	92.6\pm.6	72.4\pm1.1	96.0\pm.1	92.9\pm.1	94.9\pm.4	93.0\pm.2	91.3

Comparação com outros algoritmos

Model & Method	# Trainable Parameters	E2E NLG Challenge				
		BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter ^L)*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter ^L)*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter ^H)	11.09M	67.3 \pm .6	8.50 \pm .07	46.0 \pm .2	70.7 \pm .2	2.44 \pm .01
GPT-2 M (FT ^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	70.4 \pm .1	8.85 \pm .02	46.8 \pm .2	71.8 \pm .1	2.53 \pm .02
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter ^L)	0.88M	69.1 \pm .1	8.68 \pm .03	46.3 \pm .0	71.4 \pm .2	2.49 \pm .0
GPT-2 L (Adapter ^L)	23.00M	68.9 \pm .3	8.70 \pm .04	46.1 \pm .1	71.3 \pm .2	2.45 \pm .02
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	70.4 \pm .1	8.89 \pm .02	46.8 \pm .2	72.0 \pm .2	2.47 \pm .02

Referências

- <https://arxiv.org/pdf/2106.09685>
- <https://www.fishi-pedia.com/wp-content/uploads/2019/06/baracuda.jpg>
- <https://di-uploads-pod15.dealerinspire.com/capecoralchryslerdodgejeepram/uploads/2024/02/Dodge-Barracuda.jpg>
- <https://www.youtube.com/watch?v=LPZh9BOjkQs>
- <https://www.youtube.com/watch?v=KEv-F5UkhxU>
- <https://huggingface.co/docs/peft/en/index>

Obrigado!

Perguntas?

