Incorporação de passagens de texto a partir da edição de associações factuais em LLMs

segunda entrega

GitHub - eduseiti/rome at add_new_models

Eduardo Seiti de Oliveira, RA 940011

Atividades executadas

- Estudo da implementação original
  - Detalhes implementação; identificação de pontos de alteração; compreensão dos parâmetros descritivos dos modelos.
- Cálculo do *causal trace* para modelo Microsoft Phi 1.5
  - Determinação da camada de edição.
- Execução de testes de edição

Técnica: Identificação dos *Causal Traces*

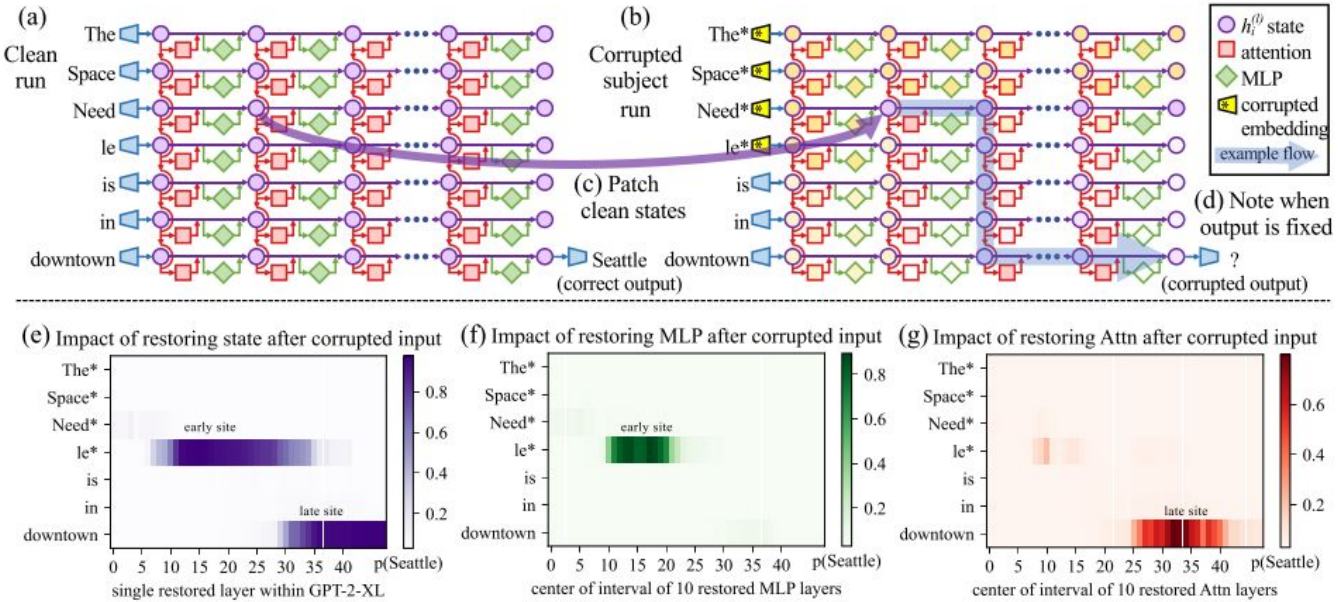


Figure 1: **Causal Traces** compute the causal effect of neuron activations by running the network twice: (a) once normally, and (b) once where we corrupt the subject token and then (c) restore selected internal activations to their clean value. (d) Some sets of activations cause the output to return to the original prediction; the light blue path shows an example of information flow. The causal impact on output probability is mapped for the effect of (e) each hidden state on the prediction, (f) only MLP activations, and (g) only attention activations.

Reproduzido de [1]

Detalhes causal trace

- Execução demorada (~16h): cálculo do efeito médio de 1208 relações * 24 camadas * 490 tokens (média) > 14M combinações.
- Lógica em `experiments/causal_trace.py/calculate_hidden_flow()`:
  - Cada relação é analisada num batch de 11 cópias — 1 referência (sem corrupção) + 10 réplicas para cálculo do efeito médio.
  - Uso de register_forward_hook() para acessar resultados do forward pass.
  - Dependência dos nomes das camadas — busca *FeedForward* e *Attention*.
- Cálculo dos efeitos e determinação da camada de edição não estão modularizados — `notebooks/average_causal_effects.ipynb`

Técnica: Edição das associações factuais

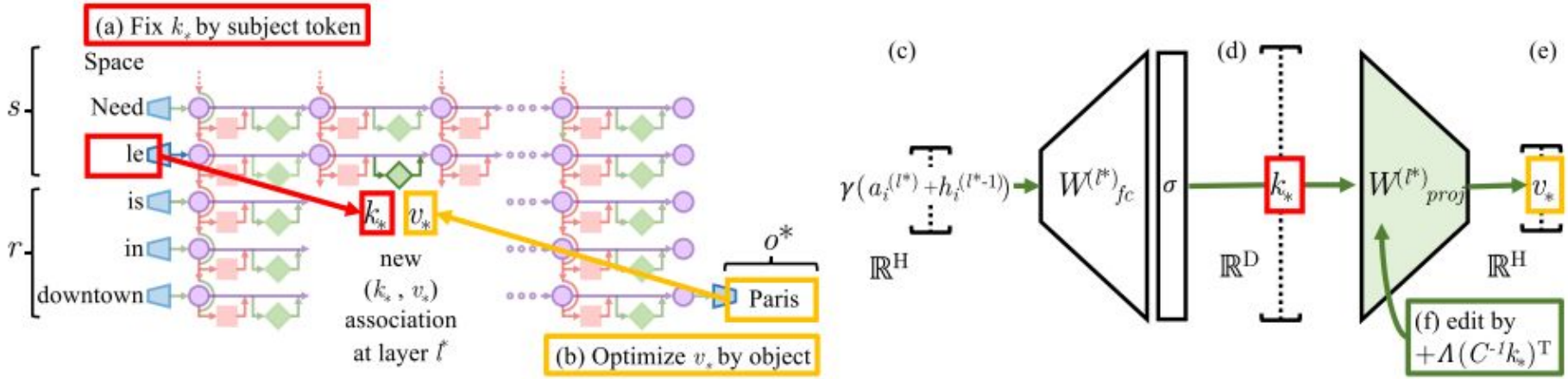


Figure 4: **Editing one MLP layer with ROME**. To associate *Space Needle* with *Paris*, the ROME method inserts a new (k_*, v_*) association into layer l^*, where (a) key k_* is determined by the subject and (b) value v_* is optimized to select the object. (c) Hidden state at layer l^* and token i is expanded to produce (d) the key vector k_* for the subject. (e) To write new value vector v_* into the layer, (f) we calculate a rank-one update $\Lambda(C^{-1}k_*)^T$ to cause $\hat{W}^{(l)}_{proj}k_* = v_*$ while minimizing interference with other memories stored in the layer.

Reproduzido de [1]

Detalhes edição das associações causais (1/3)

- Calcula o parâmetro k* como ativação média (20 amostras precedidas de saídas aleatórias) depois da não-linearidade da MLP:

$$k_* = \frac{1}{N}\sum_{j=1}^{N} k(x_j + s), \text{ where } k(x) = \sigma\left(W_{fc}^{(l^*)}\ \gamma(a_{[x],i}^{(l^*)} + h_{[x],i}^{(l^*-1)})\right)$$

- Calcula o parâmetro v* otimizando **delta = rank-1 update** a partir da loss:

$$\frac{1}{N}\sum_{j=1}^{N} \underbrace{-\log \mathbb{P}_{G(m_i^{(l^*)}:=z)}\left[o^* \mid x_j + p\right]}_{\text{(a) Maximizing } o^* \text{ probability}} + \underbrace{D_{\mathrm{KL}}\left(\mathbb{P}_{G(m_{i'}^{(l^*)}:=z)}\left[x \mid p'\right] \middle\| \mathbb{P}_G\left[x \mid p'\right]\right)}_{\text{(b) Controlling essence drift}}$$

Detalhes edição das associações causais (2/3)

- **delta** é o mesmo aplicado a todos os tokens do OBJECT.
- Loop de otimização similar ao de treinamento, mas atuando apenas sobre o parâmetro **delta**:

```
# Optimizer
opt = torch.optim.Adam([delta], lr=hparams.v_lr)
nethook.set_requires_grad(False, model)

# Execute optimization
for it in range(hparams.v_num_grad_steps):
    opt.zero_grad()

    # Forward propagation
    with nethook.TraceDict(
        module=model,
        layers=[
            hparams.layer_module_tmp.format(loss_layer),
            hparams.mlp_module_tmp.format(layer),
        ],
        retain_input=False,
        retain_output=True,
        edit_output=edit_output_fn,
    ) as tr:
        logits = model(**input_tok).logits
```

```
# Aggregate total losses
nll_loss_each = -(loss * mask).sum(1) / target_ids.size(0)
nll_loss = nll_loss_each.mean()
kl_loss = hparams.kl_factor * torch.nn.functional.kl_div(
    kl_distr_init, kl_log_probs, log_target=True, reduction="batchmean"
)
weight_decay = hparams.v_weight_decay * (
    torch.norm(delta) / torch.norm(target_init) ** 2
)
# weight_decay = hparams.v_weight_decay * torch.norm(delta) ** 2
loss = nll_loss + kl_loss + weight_decay
```

```
# Backpropagate
loss.backward()
opt.step()
```

Detalhes edição das associações causais (3/3)

- Edição final dos pesos depende do cálculo da covariância de k para conjunto representativo de textos:

$$\hat{W} = W + \Lambda(C^{-1}k_*)^T \qquad \Lambda = \frac{v_* - Wk_*}{(C^{-1}k_*)^T k_*}$$

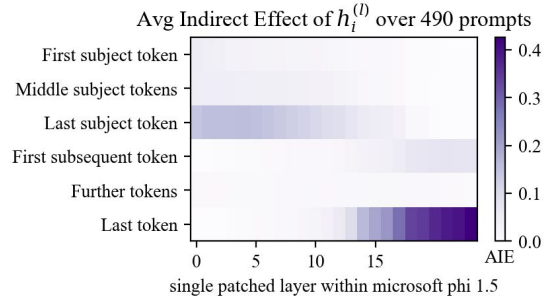
Adaptado de [1]

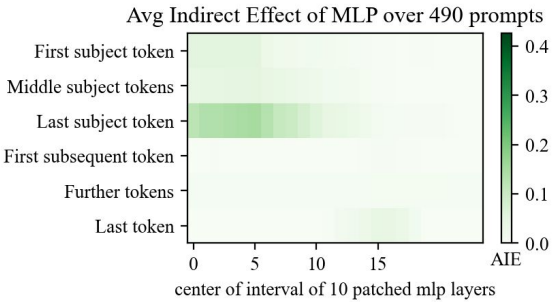
- Cálculo da matriz de covariância também é demorado (~9h).

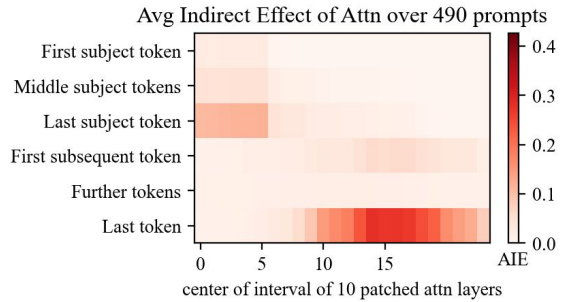
Parâmetros Microsoft Phi 1.5

hparams > ROME > {} microsoft_phi-1_5.json > ...

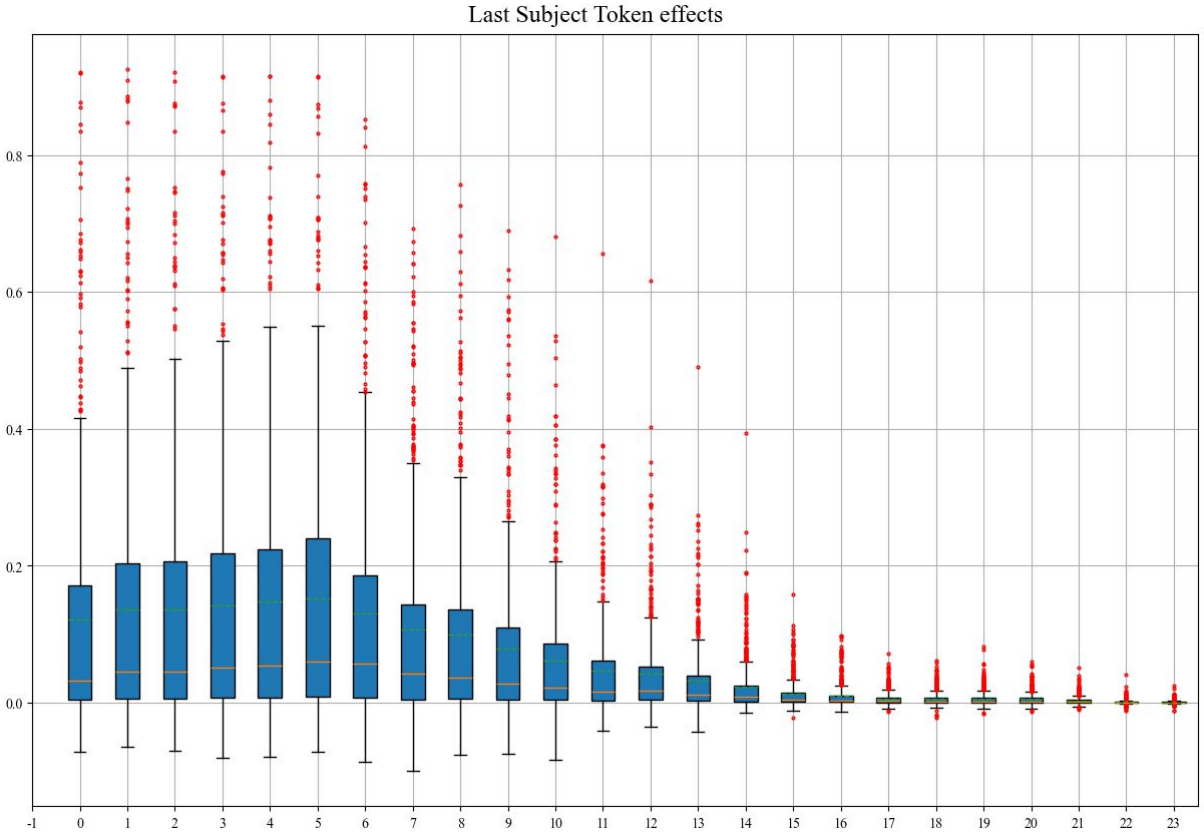
```json
{
    "layers": [
        5
    ],
    "fact_token": "subject_last",
    "v_num_grad_steps": 20,
    "v_lr": 5e-1,
    "v_loss_layer": 24,
    "v_weight_decay": 0.5,
    "clamp_norm_factor": 4,
    "kl_factor": 0.0625,
    "mom2_adjustment": true,
    "context_template_length_params": [[5, 10], [10, 10]],
    "rewrite_module_tmp": "model.layers.{}.mlp.fc2",
    "layer_module_tmp": "model.layers.{}",
    "mlp_module_tmp": "model.layers.{}.mlp",
    "attn_module_tmp": "model.layers.{}.attn",
    "ln_f_module": "model.final_layernorm",
    "lm_head_module": "lm_head",
    "mom2_dataset": "wikipedia",
    "mom2_n_samples": 100000,
    "mom2_dtype": "float32",
    "config_n_positions": "max_position_embeddings"
}
```







Efeito do último token do SUJEITO por camada



Referências

1. Meng, Kevin, David Bau, Alex Andonian, and Yonatan Belinkov. "Locating and editing factual associations in GPT." Advances in Neural Information Processing Systems 35 (2022): 17359-17372.
2. Meng, Kevin, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. "Mass-editing memory in a transformer." arXiv preprint arXiv:2210.07229 (2022).
3. Du, Yilun, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. "Improving factuality and reasoning in language models through multiagent debate." arXiv preprint arXiv:2305.14325 (2023).

Cronograma

Lista de atividades a serem feitas antes de cada entrega:

- 06 de junho - entrega I — Plano de Trabalho;
- **13 de junho - entrega II — Técnica aplicada a novo LLM;**
- 20 de junho - entrega III — Avaliações fatos compostos e múltiplos fatos;
- 27 de junho - entrega final — Avaliação da incorporação de passagem.