



Código para Aula 3

IA368DD\_2023S1: Deep Learning aplicado a Sistemas de Buscas

Marcus Vinícius Borela de Castro

Classificação de Texto e Rerankeadores

Criei rastro no neptune.ai dos treinamentos.

Base de validação fixa, leiaute qp, para comparação e mais próxima do teste no reranking)  
As acurácias foram respectivamente: 93,9%, 93,3% e 94,2%

Parece interessante dobrar os dados de treinamento  
(pendente de ser confirmado no reranking)

Experimentei 3 input\_layout: /pq/qppq  
qp: query+passage;  
pq: passage+query  
qppq: qp union pq

Obs.: qppq com o dobro de registros no treinamento

Resultados interessantes

Usei asserts no meio do código

Validei os códigos das funções com o chatgpt

Criei dados fictícios para testar a classe do dataset

Técnicas usadas para buscar a correção

No dataset: Precisamos gerar ou o modelo gera para nós quando não passamos?


Dúvida: Não salvei o tokenizador pois ele não foi alterado. Precisa?

Dúvidas

## Observações

- Criei a minha classe de Dataset. Nela, gerei os token\_type\_ids (0 e 1 para as 2 sequências)
- Ajustado erro no apóstrofe no texto de entrada (como em don't)
- Usei chat\_gpt em todas as fases. Inclusive se mostrou muito útil pra documentar funções. (Veja no início: inicializa\_seed, config\_debug e config\_display)
- Retomei (e aproveitei) conteúdo de cadernos de cursos anteriores na Unicamp

## Ressalvas



Na fase do reranking, o `ndcg_cut_10` deu baixo 0.0441.

Suspeita de causa: não ter usado `torch.save(model)` ao invés de `model.save_pretrained()`.

Segundo ChatGPT, há uma diferença: `model.save_pretrained()` salva os pesos do modelo, bem como todos os outros artefatos necessários para carregar e inicializar o modelo em uma instância específica de um modelo do Transformers, enquanto `torch.save(model)` salva apenas os pesos do modelo.

Obs.: Fiz o ajuste no caderno e deixei treinando para ver a diferença.