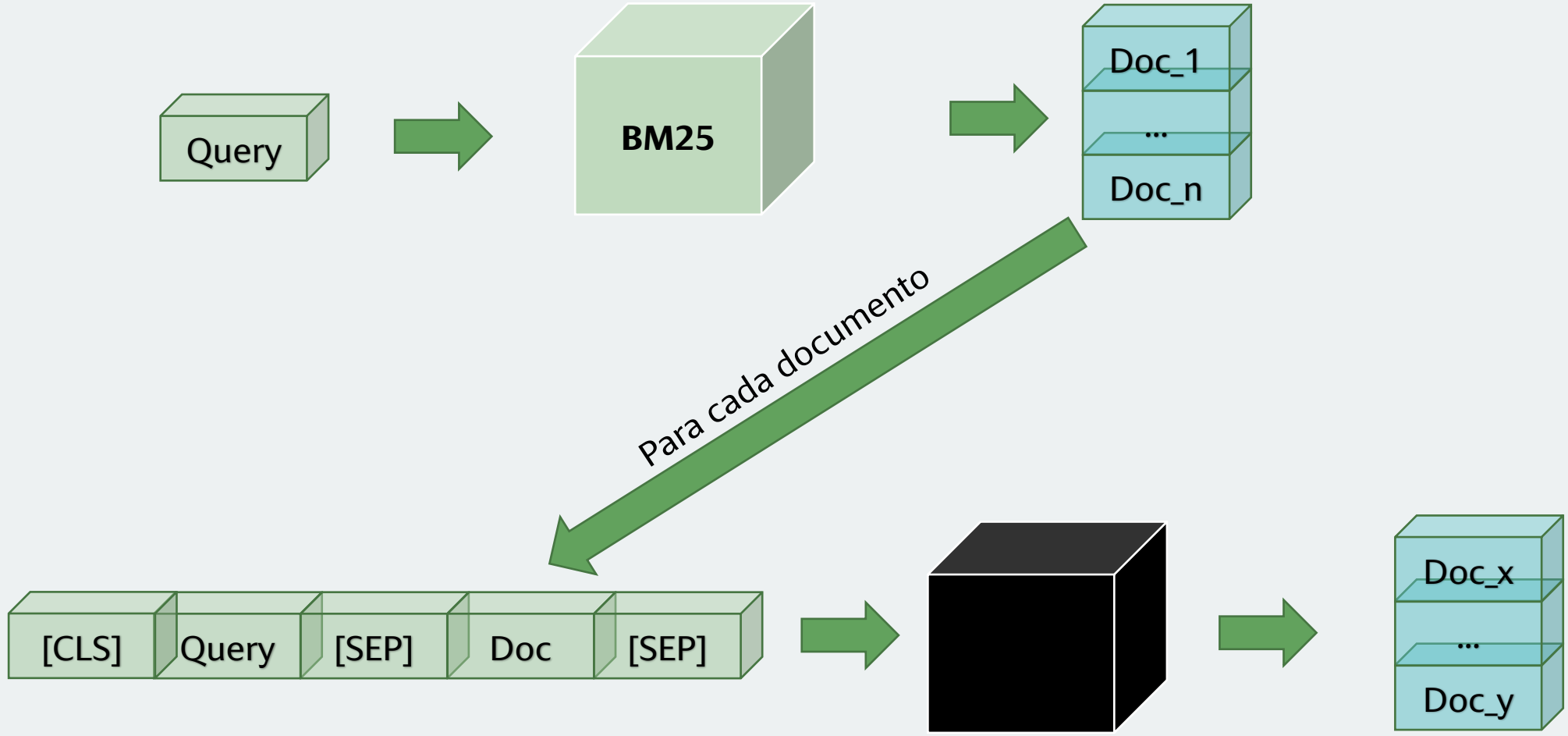
Notebook – Inpars

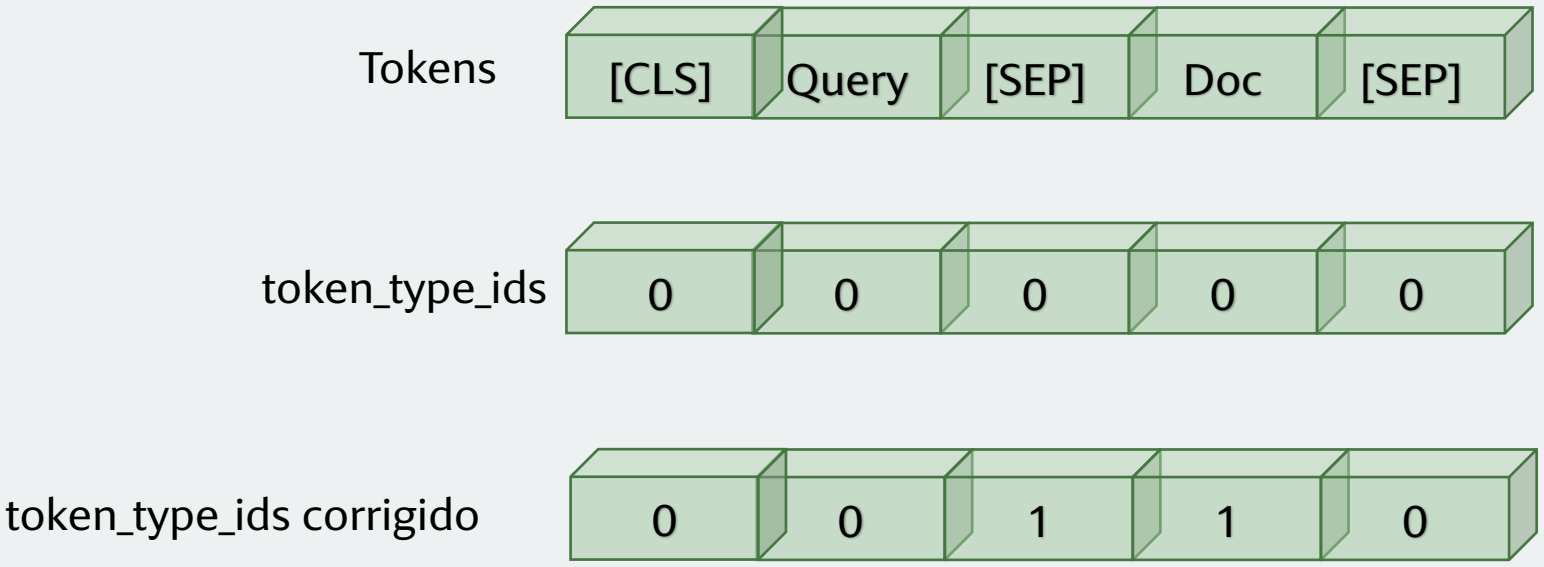
Leandro Carísio

Conceitos do exercício



Problemas e soluções no desenvolvimento

1. Primeira implementação sem separar a query do documento via token_type_ids:

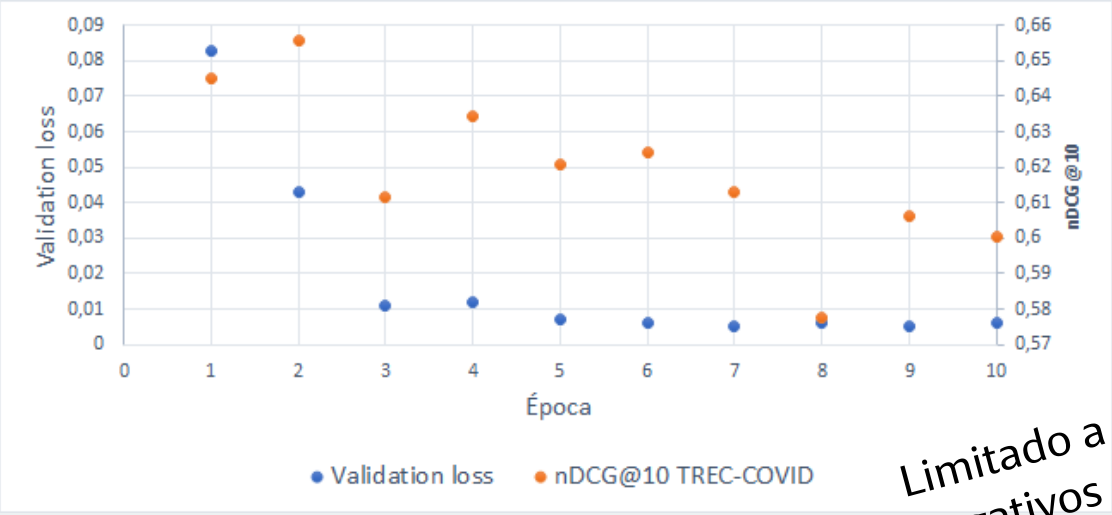


	nDCG@10	nDCG@10 após correção
cross-encoder/ms-marco-MiniLM-L-6-v2 (sem fine-tuning)	0,2949	0,7131

Problemas e soluções no desenvolvimento

2. Fine-tuning com AutoModelForSequenceClassification não funcionou e eu não faço ideia do motivo

	Antes do fine-tuning	Depois do fine-tuning
Sem corrigir token_type_ids	0,2949	0,6236
Corrigindo token_type_ids	0,7131	0,6557



Limitado a 5 exemplos negativos por query

Problemas e soluções no desenvolvimento

2. Fine-tuning com AutoModelForSequenceClassification não funcionou e eu não faço ideia do motivo

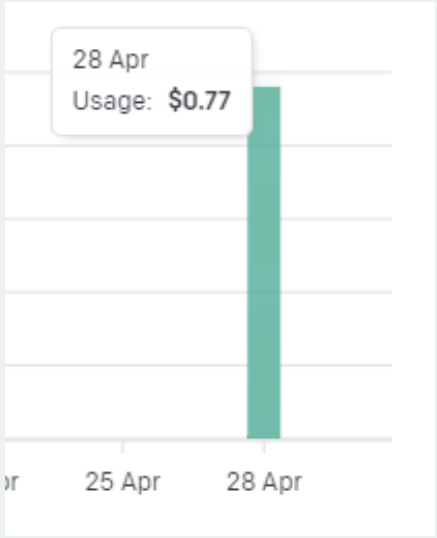
Tentativas pra resolver a questão:

- Limitar a no máximo 5 exemplos negativos por query
- Balancear o dataset (limitar a 1 exemplo negativo por query)
- Testar só com o meu dataset
- Reranking com 100 hits em vez de 1000 hits

A maior diferença foi em fazer o reranking só em 100 hits (chegou a 5 pontos no nDCG ***dependendo*** do modelo)

Resultados

Custo de US$ 0.77 para gerar 1.000 queries usando gpt-3.5-turbo



nDCG@10

0,7131 sem fine-tuning

0,6798 com fine-tuning e reranking em 1000 documentos

0,6827 com fine-tuning e reranking em 100 documentos

Dúvida básica

No cálculo da loss do modelo ele usa o mesmo cálculo de similaridade que encontramos na busca densa.

O Inpars é uma busca densa com o fine-tuning feito a partir de dados gerados por um LLM?

Training code: train_bi-encoder_margin-mse.py

MarginMSELoss is based on the paper of Hofstätter et al. As for MultipleNegativesRankingLoss, we have triplets: `(query, passage1, passage2)`. In contrast to MultipleNegativesRankingLoss, `passage1` and `passage2` do not have to be strictly positive/negative, both can be relevant or not relevant for a given query.

We then compute the Cross-Encoder score for `(query, passage1)` and `(query, passage2)`. We provide scores for 160 million such pairs in our msmarco-hard-negatives dataset. We then compute the distance: `CE_distance = CEScore(query, passage1) - CEScore(query, passage2)`

```
train_loss = losses.MarginMSELoss(model=model)
```

```
def __init__(self, model, similarity_fct = util.pairwise_dot_score):
```

```
    scores_pos = self.similarity_fct(embeddings_query, embeddings_pos)
    scores_neg = self.similarity_fct(embeddings_query, embeddings_neg)
    margin_pred = scores_pos - scores_neg
```

Dúvida básica (?)

Qual o truque pra fazer o fine-tuning funcionar usando

AutoModelForSequenceClassification?

Obrigado

Leandro Carísio

carisio@gmail.com