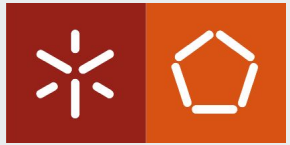
Information Retrieval

RepositoriUM

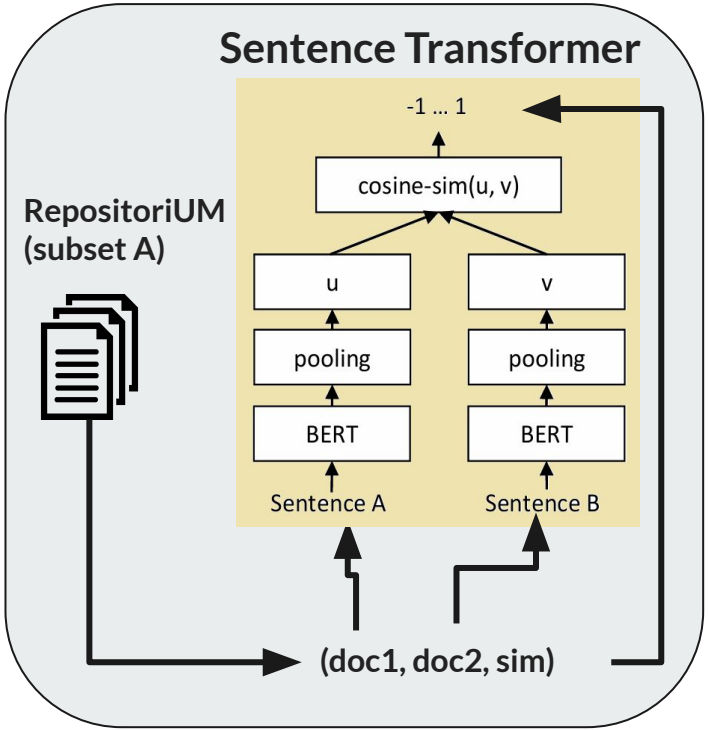
Luís Filipe Cunha
lfc@di.uminho.pt

José João Almeida
jj@di.uminho.pt

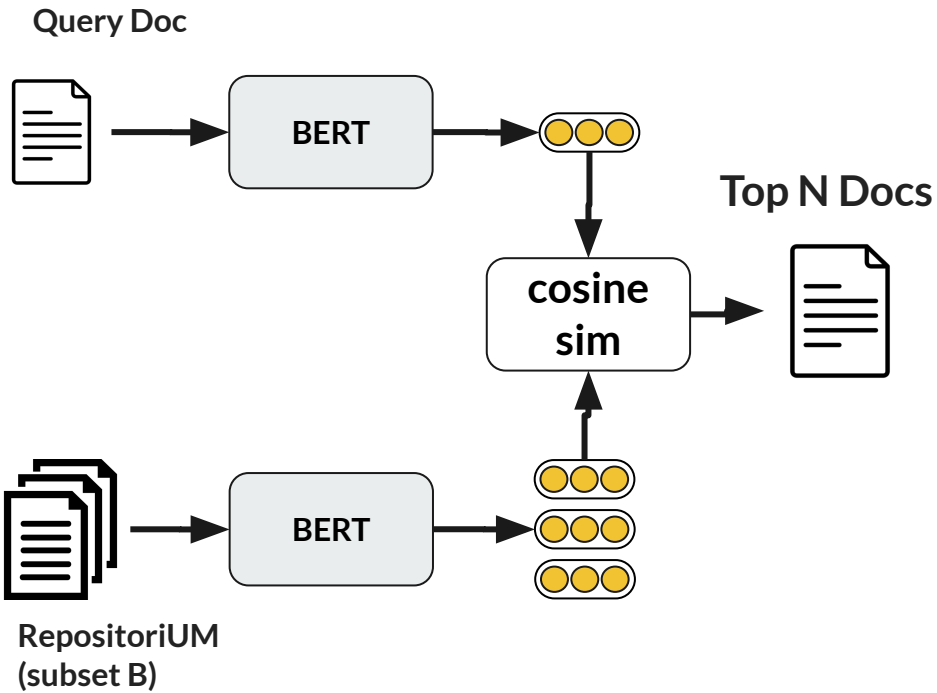


RoadMap

Model Training



Model Inference



Toolkits

```
#Calculo de combinações
from itertools import combinations
combinations(lista, 2) # pares


#Balancear dados
undersampled_majority_class = resample(majority_class,
                                       replace=False,     # Don't duplicate samples
                                       n_samples= len(minority_class),  # Match minority
                                       random_state=42)


#Criar train, test split estratificados
scores = [score for _, _, score, *_ in balanced_abstract_pairs]
train_data, test_data = train_test_split(
    balanced_abstract_pairs,
    test_size=0.2,
    random_state=42,
    stratify=scores
)
```

Model training

```
from sentence_transformers import SentenceTransformerTrainer, SentenceTransformerTrainingArguments
from sentence_transformers.similarity_functions import SimilarityFunction
from sentence_transformers.evaluation import EmbeddingSimilarityEvaluator
```

```
args = SentenceTransformerTrainingArguments(
    # Required parameter:
    output_dir="models/sentence_transformers/bert-base-portuguese",
    report_to="none",
    # Optional training parameters:
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    fp16=True,  # Set to False if you get an error that your GPU can't run on FP16
    bf16=False,  # Set to True if you have a GPU that supports BF16
    # Optional tracking/debugging parameters:
    eval_strategy="epoch",
    save_strategy="epoch",
    save_total_limit=2,
    load_best_model_at_end=True,
)
```

Model training

```
# Create the evaluator
dev_evaluator = EmbeddingSimilarityEvaluator(
    test_dataset['abstract1'],  # Assuming these are the sentence pairs for evaluation
    test_dataset['abstract2'],
    test_dataset['score'],  # Assuming this contains the similarity scores
    main_similarity=SimilarityFunction.COSINE,
)

# 6. Create the trainer & start training
trainer = SentenceTransformerTrainer(
    model=model,
    args=args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    loss=loss,
    evaluator=dev_evaluator,
)
```

Inference

```
from sentence_transformers import util
import torch

embeddings = model.encode(abstracts, convert_to_tensor=True)
query_embedding = model.encode(query_text, convert_to_tensor=True)

# Calculate the similarity between the query and the abstracts
cosine_scores = util.pytorch_cos_sim(query_embedding, embeddings)
retrieval_results = torch.topk(cosine_scores, k=15)

retrieval_results.values  -> lista de pares (score, index)
```

Information Retrieval

RepositoriUM

Luís Filipe Cunha
lfc@di.uminho.pt

José João Almeida
jj@di.uminho.pt