Trabalho Prático II

Filipe Cunha José João

Maio 2025

Contents

1 Enunciado geral

Fazer um módulo de Information Retrival

- com base no RepsitoriUM
  - numa subcoleção a extrair / usar
- Calculador de similaridades de texto

1.1 Extrair uma coleção a partir do RepositoriUM

- script python com `request.get(urlbase, params)` ... → XML

1.1.1 repositorium → XML

```
url = "https://repositorium.sdum.uminho.pt/oai/oai"
col = "col_1822_21316"   #(msctesis do DI; 1822_2 msc; 1822_3 phd
n = #( 0, 100, 200, ...)

params = {"verb": "ListRecords",
          "resumptionToken": f"dim///{col}/{n}" }
r = requests.get(url, params=params).text
if "noRecordsMatch" not in r:  #( XML += r    Juntar r ao XML )
```

1.1.2

- onde "dim" é o "metadataPrefix" ("dim" é o mais completo)
- "col" é a coleção (correspondente ao parametro "Set" do OAI-PMH) ver:

`https://repositorium.sdum.uminho.pt/oai/oai?verb=ListSets`

- n é o offset do próximo grupo de docs (0, 100, 200, ...)

1.2 Calcular Coleção documental (XML → Json)

ColDoc.json = XML→Json(OAI.xml)

Arrumando a informação: Limpando, filtrando, normalizando

1.3 Calcular Coleção-treino-similaridades

1.3.1 ColTrain

ColTrain :: (txt,txt,sim)*

```
for d1,d2 in ColDoc.json²:
   ColTrain.append( (d1.abst, d2.abst, guess_sim(d1,d1)
filtrar os pares relevantes
```

1.3.2 Guess_sim

Estudar heurísticas para guess_sim(d1,d2) usando:

- keywords em comum; mas ter em conta:
  - nº de key de cada doc
  - raridade/trivialidades das keys
- subject UDC
- subject fos
- Coleções (diferentes das usadas na query)

1.4 Treinar sentence-transformer

- com base numa coleção (doc1, doc2, similarity)*

```
model = train(BERT, ColTrain)
```

1.5 Usar modelo

```
retrive( quest, Col ) =
  mostrelevant (
    [ (doc, model.similarity(quest,Col) ) for doc in Col ]
  )
```