2. Indexing & Document Retrieval

Task

- Download an existing IR dataset - Cranfield collection [http://ir.dcs.gla.ac.uk/resources/test_collections/cran/]
  - Or you can use already preprocessed data cranfield.zip (d-documents, q-queries, r-relevances (a set of relevant document ids for each query id))
- Represent each document and query using the Vector Space Model with all following weightings:
  - Use Binary representation
  - Use pure Term Frequency
  - Use TF-IDF
- Compute relevance scores for each combination of query, document
  - Use Euclidean distance
  - Use Cosine similarity measure
- Evaluate quality and difference of both scores and different weighting schemas
  - Compute Precision, Recall, F-measure (you can limit to top N relevant documents for each query)

Instructions for submitting

In your private namespace on EDUX provide the following information:

- Describe results of different weighting schemas and different relevance scores
- Describe/summarise/compare evaluation metrics
- Provide the link to your implementation
  - You can use: https://gitlab.fit.cvut.cz [https://gitlab.fit.cvut.cz] or https://github.com/ [https://github.com/]
- Provide the link to computed relevance scores for each combination (e.g. csv file)
- Comment on
  - issues during the design/implementation
  - ideas for extensions/improvements/future work

Example

```
# import
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

# prepare corpus
corpus = []
for d in range(1400):
    f = open("./d/"+str(d+1)+".txt")
    corpus.append(f.read())
# add query to corpus
for q in [1]:
    f = open("./q/"+str(q)+".txt")
    corpus.append(f.read())

# init vectorizer
tfidf_vectorizer = TfidfVectorizer()
```

```
# prepare matrix
tfidf_matrix = tfidf_vectorizer.fit_transform(corpus)

# compute similarity between query and all docs (tf-idf) and get top 10 relevant
sim = np.array(cosine_similarity(tfidf_matrix[len(corpus)-1], tfidf_matrix[0:(len(corpus)-1)])[0])
topRelevant = sim.argsort()[-10:][::-1]+1
print(topRelevant)
```

/mnt/www/courses/MI-DDW.16/data/pages/hw/02/start.txt · Poslední úprava: 2017/03/12 21:37 autor: kuchajar