222112058_Feza Raffa Arnanda_Penugasan Praktikum 8

October 24, 2023

Penugasan Praktikum Pertemuan 8 Information Retrieval

Nama : Feza Raffa Arnanda

NIM : 222112058

Kelas : 3SD2

0.1 Soal

Buat fungsi main untuk menampilkan 3 list dokumen yang terurut berdasarkan BM25 pada folder "berita" dengan query "vaksin corona jakarta". Bandingkan dengan hasil perankingan cosine similarity pada modul 5.

0.2 Import package BM25

[ ]:
```
from rank_bm25 import BM25Okapi
```

0.3 Preprocessing

Tokenisasi

[ ]:
```
import nltk
from nltk.tokenize import word_tokenize
def tokenisasi(text):
    tokens = word_tokenize(text)
    return tokens
```

Stemming

[ ]:
```
def stemming(text):
    from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
    # create stemmer
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    # stemming process
    output = stemmer.stem(text)
    return output
```

[ ]:
```
def stemming_sentence(text):
    stemmed_tokens = [stemming(token) for token in tokenisasi(text)]
    output = ' '.join(stemmed_tokens)
    return output
```

[ ]:
```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
def stemming_sastrawi(tokens):
    # Membuat stemmer
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    return [stemmer.stem(token) for token in tokens]
```

Case folding

[ ]:
```
def case_folding(text):
    text = text.lower()
    return text
```

Stopwords Elimination

[ ]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('Indonesian'))

def eliminasi_stopword(token):
    return [kata for kata in token if kata not in stop_words]
```

Path folder berita

[ ]:
```
path = "C:/Users/FEZA/My Drive/00. Drive PC/1.STIS/5. Semester 5/Information␣
↪Retrieval [IR] P/berita"
```

Open file pada folder berita

[ ]:
```
import re
import os
inverted_index = {}
doc_dict = {}
i = 1
for filename in os.listdir(path):
    if (filename.endswith('.txt')):
        file_path = os.path.join(path, filename)
        # Ekstrak angka dari nama file menggunakan regular expressions
        match = re.search(r'\d+', filename)
        if match:
            doc_id = match.group() # Mengambil angka dari nama file sebagai␣
↪dokumen ID
            with open (file_path, mode='r', encoding='utf-8') as file:
                text = file.read()
                hasil_case_folding = case_folding(text)
```

```
                token = tokenisasi(hasil_case_folding)
                token_bersih = eliminasi_stopword(token)
                stemm_token = stemming_sastrawi(token_bersih)
                stemm_token_final = [item for item in stemm_token if item !=␣
↪''] # membersihkan term kosong pada hasil stemming sebelumnya
                # Menggabungkan hasil stemming menjadi sebuah teks/paragraf
                doc_dict[doc_id] = ' '.join(stemm_token_final)
                for term in set(stemm_token_final): # penggunaan set untuk␣
↪mengantisipasi duplikasi term pada sebuah dokumen
                    if term in inverted_index:
                        inverted_index[term].append(doc_id)
                    else:
                        inverted_index[term] = [doc_id]
```

Membuat tokenized corpus, dimana akan dibuat list hasil preprocessing teks pada setiap dokumen

[ ]:
```
tokenized_corpus = [tokenisasi(doc_dict[doc_id]) for doc_id in doc_dict]
```

0.4 BM25

Inisiaslisasi query dan hasil skor menggunakan BM25

[ ]:
```
query = "vaksin corona jakarta"

tokenized_query = tokenisasi(query)

bm25 = BM25Okapi(tokenized_corpus)

doc_scores = bm25.get_scores(tokenized_query)

print(doc_scores)
```

```
[0.17910081 0.80460786 1.10319823 0.35216875 0.53675825]
```

Function untuk me-retrieve top K dokumen dari hasil skore BM25

[ ]:
```
from collections import OrderedDict

def exact_top_k(doc_dict, rank_score, k):
    relevance_scores = {}
    i = 0
    for doc_id in doc_dict.keys():
        relevance_scores[doc_id] = rank_score[i]
        i = i + 1
    sorted_value = OrderedDict(sorted(relevance_scores.items(), key=lambda x:␣
↪x[1], reverse = True))
    top_k = {j: sorted_value[j] for j in list(sorted_value)[:k]}
    return top_k
```

```
[ ]: top_k = exact_top_k(doc_dict,doc_scores,3)
```

Print hasil ranking document

```
[ ]: for i, (doc_id, score) in enumerate(top_k.items()):
         print(f"Rank {i + 1}: berita{doc_id}.txt, Skor: {score}")
```

```
Rank 1: berita3.txt, Skor: 1.1031982334570316
Rank 2: berita2.txt, Skor: 0.8046078586167857
Rank 3: berita5.txt, Skor: 0.5367582471433582
```

Hasil perankingan top 3 dokumen: 1. berita2.txt dengan nilai cosine similarity = [0.19658299] 2. berita3.txt dengan nilai cosine similarity = [0.13375627] 3. berita5.txt dengan nilai cosine similarity = [0.0569143]

0.5 Kesimpulan

Berdasarkan hasil antara SVM menggunakan BM25 dan Cosine Similarity, top 3 dokumen yang dihasilkan berbeda pada ranking 1 dan 2 yang terbalik. Pada hasil menggunakan BM25, urutan ranked documentnya adalah berita3, berita2, dan berita5. Sedangkan jika menggunakan cosine similarity, urutan ranked documentnya adalah berita3, berita4, dan berita2.

Pengunaan BM25 berdasarkan teori pada materi perkuliahan lebih baik daripada menggunakan bobot TF.IDf, karena pada BM25 memperhitungkan panjang dokumen dari Term Frequency, sedangkan TF.IDf tidak memperhitungkan itu. BM25 lebih baik dalam penanganan term yang jarang muncul pada dokumen sehingga dapat menghasilkan top K document yang lebih relevan dan baik.