

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI
PROGRAM STUDI ILMU KOMPUTER DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA

TEMU KEMBALI INFORMASI

Tugas 8 Metode Perhitungan Skor Pencarian



DISUSUN OLEH:

ADAM YOGISYAH PUTRA	20/455439/PA/19654
MUHAMMAD ARSYA PUTRA	20/462186/PA/20158
HIZKYA FIRSTADIPA HARTOKO	20/455447/PA/19662

DOSEN:

Dr. Lukman Heryawan, S.T., M.T.

Metode Perhitungan Skor Pencarian

I. Metode

Metode yang digunakan pada saat melakukan perhitungan skor sistem temu kembali informasi adalah dengan menggunakan TF-IDF. Metode ini mempertimbangkan kemunculan suatu term dalam dokumen, yaitu sebagai Term Frequency, dan jumlah dokumen yang di dalamnya terdapat term tersebut yaitu Document Frequency yang di Invert menjadi Inverted Document Frequency.

II. Proses Scoring

A. Import Dataset

Melakukan import dataset yang akan digunakan untuk melakukan scoring. Dataset merupakan Document Inverted Index yang berisi term-term beserta frekuensi kemunculan term dalam dokumen dan posting list (daftar index dokumen yang memuat term).

```
Import Dataset
```

```
import pandas as pd
import numpy as np

url = 'https://raw.githubusercontent.com/HizkiFirst/tki-scraping-obat/main/dataset/scoring%20dataset.csv'
df = pd.read_csv(url)
df.head(5)
```

	term	docFrequency	postingList
0	a	8	[273, 344, 344, 273, 45, 18, 276, 29]
1	aa	2	[421, 421]
2	abdomen	1	[381]
3	abnormal	6	[252, 250, 254, 250, 252, 254]
4	aborsi	4	[210, 210, 210, 210]

```
[ ] #jumlah dokumen
n = 434
```

B. Proses Definisi Query Term

Mendefinisikan query yang digunakan untuk uji skoring dan melakukan tokenisasi terhadap query tersebut agar memperoleh query terms.



```
query = "demam tinggi"
query = query.split(' ')
query

['demam', 'tinggi']
```

C. Perhitungan TF-IDF

Menghitung skor dengan metode TF-IDF

Untuk tiap query term dilakukan perhitungan IDF dengan rumus:

$$IDF = \log_{10} \frac{n}{DF}$$

Lalu dilakukan perhitungan TF dari query term tersebut pada tiap document .

Skor diperoleh dengan mengalikan hasil TF dengan IDF.

▼ Perhitungan TF-IDF

```
import math
import operator as op

skor = [0]*n
docindex = []

df_skor = pd.DataFrame()

for qterm in query:
    i= df.index[df['term']==qterm]
    docindex.append(i[0])

    doclist = df.iloc[i[0]]["postingList"]
    doclist = doclist.strip('['').split(", ")
    docFreq = len(set(doclist))

    docUnique = set(doclist)
    docUnique = [eval(i) for i in docUnique]
    docUniqueList = sorted(list(docUnique))

    # Perhitungan IDF
    idf = math.log(n/docFreq,10)

    # Perhitungan TF
    for doc in docUniqueList:
        tf = op.countOf(doclist, str(doc))
        skor[doc] = tf*idf
```

D. Pengurutan Hasil Skor

Hasil skor diurutkan dari terbesar ke terkecil

Ranked Score

```
df_skor['skor'] = skor
df_ranked = df_skor.sort_values("skor",ascending=False)
df_ranked.head(10)
```

	skor
373	5.029114
89	4.384195
68	4.384195
150	4.384195
88	4.384195
24	4.384195
152	2.922797
25	2.922797
72	2.922797
123	2.922797

III. Hasil

Hasil yang diperoleh adalah Skor dan document ID yang sudah terurut berdasarkan perhitungan TF-IDF

Link dataset

<https://github.com/HizkiFirst/tki-scraping-obat/blob/main/dataset/scoring%20dataset.csv>

IV. Lampiran

Repository: <https://github.com/HizkiFirst/tki-scraping-obat>

Source code:

https://colab.research.google.com/drive/1-hC3LDkF1oVQd_QCgd7LveIvqB7TeRtE?usp=sharing