

Penerapan OCR dalam Mesin Pencari File dengan Konten Tertentu

Disusun untuk Memenuhi Tugas Mata Kuliah

Temu Kembali Informasi

Dosen Pengampu: **Arif Nurwidyanto, M.Cs**



Disusun oleh Kelompok 4:

Muhammad Faqih Husaen (19/442478/PA/19227)

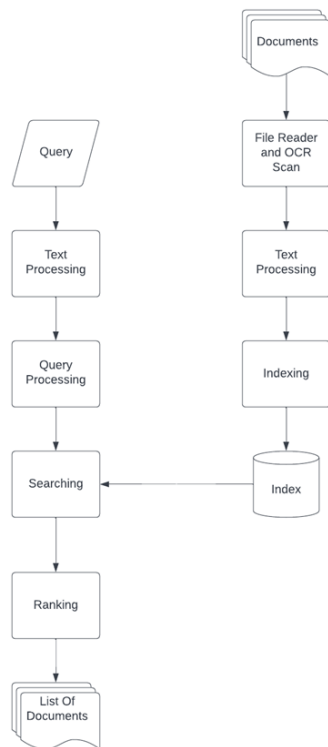
Fajar Merah Diwangkara (20/459264/PA/19925)

Ryan Novianno (20/462191/PA/20163)

**PROGRAM STUDI S1 ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA FAKULTAS
MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
2022**

1. Arsitektur dan Komponen

1.1 Arsitektur

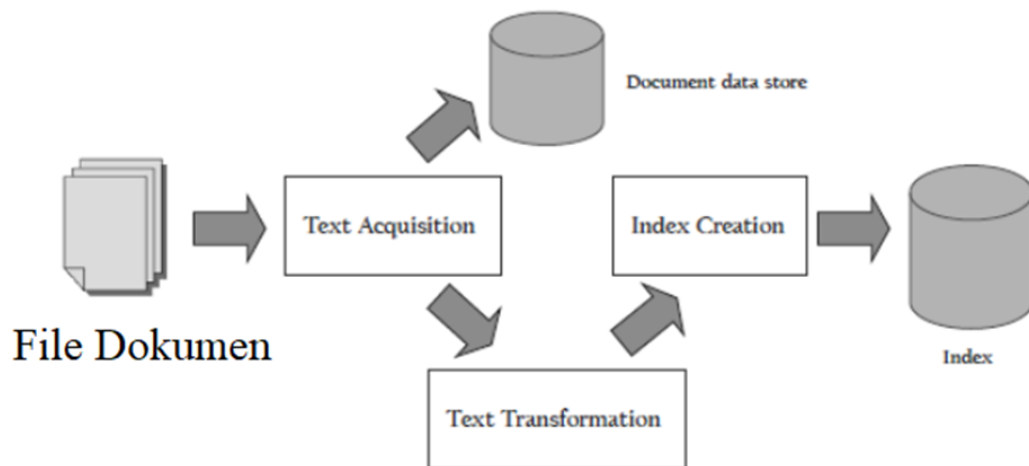
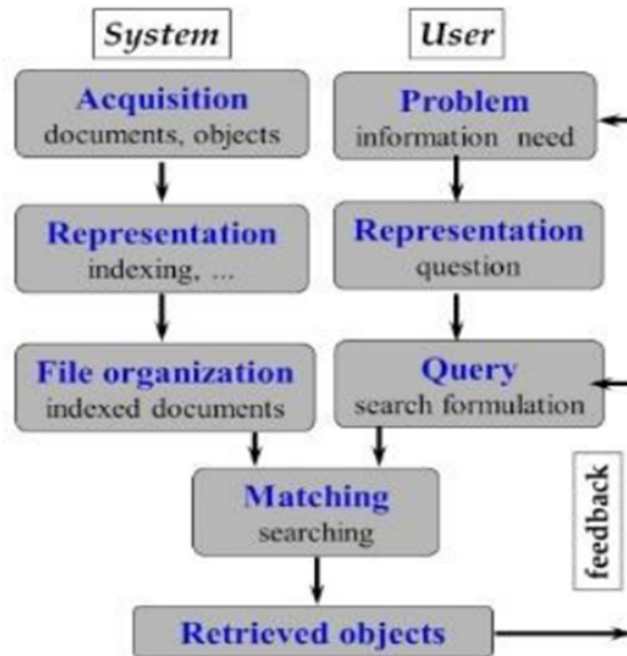


Sistem temu kembali informasi yang kami buat membaca dokumen-dokumen yang dipilih oleh user, jika terdapat gambar pada dokumen maka dilakukan scan ocr untuk mengekstrak informasi text dari gambar. Pemrosesan teks kemudian diterapkan pada informasi text dan dilakukan pembuatan indeks berdasarkan hasil pemrosesan tersebut. Hasil pembuatan index akan disimpan ke dalam database untuk selanjutnya dapat dilakukan pencarian.

Untuk menggunakan sistem pencarian dokumen, user memasukkan informasi yang akan dicari/query ke pencarian, lalu sistem akan melakukan teks processing dan query processing pada query. Hasil pemrosesan query lalu dimasukkan ke proses searching untuk menemukan dokumen yang paling relevan pada database indeks. Hasil searching kemudian di ranking berdasarkan relevansi dan ditampilkan ke users.

1.2 Komponen

Komponen sistem temu kembali informasi yang kami buat terdiri dari beberapa fungsi utama yaitu akuisisi teks pada dokumen, indexing dokumen, pemrosesan permintaan pengguna/user's query, pencocokan query pengguna dengan database indexing, dan pengambilan item yang cocok dengan query.



Komponen sistem temu kembali bagian indexing yang kami buat terdiri dari Text Acquisition, Text Transformation, dan Index Creation:

- Sistem dapat menerima file dengan format yang berbeda-beda, contohnya .txt, .docx, .pdf, dan lain sebagainya. Fungsi Text Acquisition adalah proses pengambilan teks pada dokumen berdasarkan format file yang diberikan. Jika terdapat gambar dalam dokumen, maka fungsi Text Acquisition pada sistem kami yaitu mengambil teks yang ada dalam gambar menggunakan OCR.
- Teks hasil akuisisi masih mengandung banyak informasi yang tidak penting sehingga dapat dihilangkan atau disederhanakan bentuknya. Text transformation biasanya terdiri dari 3 proses, yaitu tokenization, stopword removal atau filtering, dan stemming. Tokenisasi merupakan proses untuk memisahkan kalimat menjadi kata penyusunnya berdasarkan spasi. Filtering merupakan proses mengurangi kata pada hasil tokenisasi dengan menghilangkan kata yang termasuk ke dalam stopwords. Kata stopwords merupakan kata umum yang tidak mengandung informasi penting sehingga bisa dihilangkan. Stemming merupakan proses mengubah kata hasil filtering ke bentuk dasarnya.
- Index Creation berfungsi untuk membuat index untuk dokumen input dan menyimpannya ke index database.

2. Data dan Metode Akuisisi

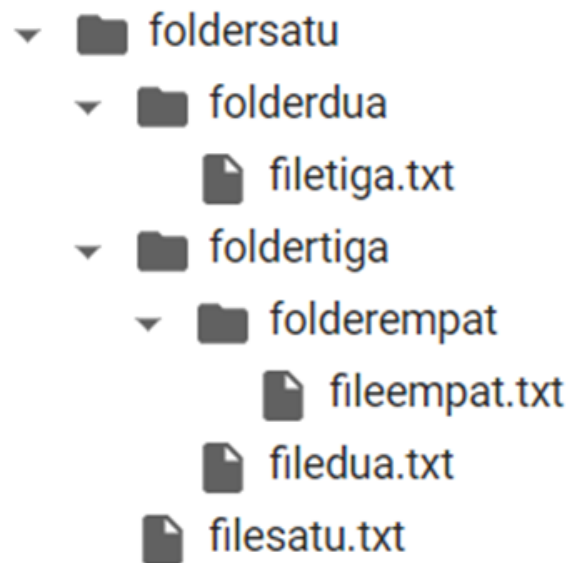
2.1 Metode Crawling

Sistem Temu Kembali Informasi yang kami buat dapat melakukan pencarian file pada komputer dengan konten tertentu. User akan memasukkan query lalu STKI kami mencari file yang mempunyai konten yang sesuai dengan query tersebut. STKI yang kami buat tidak melakukan crawling di internet karena STKI kami hanya bekerja untuk file pada sistem lokal saja. Sebagai gantinya, kami melakukan hal serupa dengan crawling yang bekerja dengan membaca direktori setiap file yang ada pada folder.

Sistem Temu Kembali Informasi yang kami buat mendapatkan data dari file pada sistem lokal. User memasukkan daftar file dan folder yang ingin diproses oleh STKI kami. Jika user memasukkan path ke folder, maka STKI kami akan membaca semua file dan sub folder yang ada pada folder tersebut. Jika yang terbaca adalah sebuah sub folder, sistem

Temu Kembali Informasi yang kami kembangkan juga akan memproses folder tersebut, dan seterusnya hingga semua file terbaca.

Sebagai contoh, jika STKI kami berawal dari folder satu, dan susunan folder dan file seperti pada gambar di bawah, maka STKI kami akan membaca filesatu.txt. Lalu STKI kami akan masuk ke folderdua dan foldertiga lalu membaca filetiga.txt dan filedua.txt. Terakhir, STKI kami akan masuk ke folderempat dan membaca fileempat.txt. Kemudian didapatkan hasil akhir yang berisi direktori dari keempat file tersebut.



3. Pembuatan Indeks

3.1 Metode Pembuatan Indeks

Untuk metode pembuatan index, kami memilih untuk menggunakan metode Single Pass In Memory Indexing. SPIMI adalah salah satu metode indexing pada Information Retrieval yang mana proses pembangunan inverted indexnya dapat terjadi di memori dan di disk.

3.2 Metode SPIMI

Metode SPIMI memiliki cara kerja membentuk dictionary yang memuat kata atau term beserta posting list. Posting list berisi dokumen id dari term yang bersangkutan.

Dictionary tersebut kemudian akan dimasukkan ke dalam file-file block. File-file block kemudian akan digabungkan untuk membentuk inverted index secara keseluruhan.

```

SPIMI-INVERT(token_stream)
1  output_file = NEWFILE()
2  dictionary = NEWHASH()
3  while (free memory available)
4  do token  $\leftarrow$  next(token_stream)
5      if term(token)  $\notin$  dictionary
6          then postings_list = ADDTODICTIONARY(dictionary, term(token))
7          else postings_list = GETPOSTINGSLIST(dictionary, term(token))
8          if full(postings_list)
9              then postings_list = DOUBLEPOSTINGSLIST(dictionary, term(token))
10         ADDTOPOSTINGSLIST(postings_list, docID(token))
11  sorted_terms  $\leftarrow$  SORTTERMS(dictionary)
12  WRITEBLOCKTODISK(sorted_terms, dictionary, output_file)
13  return output_file

```

4. Perhitungan Skor Pencarian

4.1 Metode Perhitungan Skor

Metode yang digunakan untuk menghitung skor pencarian adalah metode cosine. Metode cosine similarity merupakan metode untuk menghitung kesamaan antara dua buah objek yang dinyatakan dalam dua buah vector dengan menggunakan keywords (kata kunci) dari sebuah dokumen sebagai ukuran. Cosine similarity memiliki cara kerja sebagai berikut :

```

COSINESCORE(q)
1  float Scores[N] = 0
2  float Length[N]
3  for each query term t
4  do calculate  $w_{t,q}$  and fetch postings list for t
5      for each pair(d,  $tf_{t,d}$ ) in postings list
6          do  $Scores[d] + = w_{t,d} \times w_{t,q}$ 
7  Read the array Length
8  for each d
9      do  $Scores[d] = Scores[d] / Length[d]$ 
10 return Top K components of Scores[]

```

5. Evaluasi

5.1 Pengukuran Evaluasi

Untuk pengukuran evaluasi kami menggunakan Mean average precision (MAP).





$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$

5.2 Dataset

Kami menggunakan dataset Cranfield dalam melakukan proses evaluasi sistem temu kembali yang kami kembangkan. Dataset cranfield terdiri dari 365 query dan 1400 dokumen yang merupakan bagian abstrak dari paper aerodinamik yang dikumpulkan di UK pada tahun 1950 an. Kami berencana menggunakan 50 dokumen dari 1400 dokumen.

5.3 Penggunaan Dataset

Pada dataset Cranfield, terdapat 3 file, yaitu **cran.all.1400**, **cran.qry**, dan **cranqrel**.

 cran.all.1400	10/17/1997 6:09 PM	1400 File	1,607 KB
 cran.qry	11/18/1996 6:36 PM	QRY File	29 KB
 cranqrel	2/14/1997 7:13 AM	File	20 KB
 cranqrel.readme	2/14/1997 7:26 AM	README File	2 KB

cran.all.1400 adalah file yang berisi semua dokumen pada dataset.

cran.qry adalah file yang berisi semua query pada dataset.

cranqrel adalah file yang berisi pasangan query dan dokumen yang relevan beserta skor relevansi.

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Jika untuk query qj search engine mengeluarkan hasil dokumen dk, dan pasangan query dan dokumen tersebut ada di **cranqrel**, maka dokumen tersebut termasuk relevant dan retrieved atau true positive.

- Jika untuk query q_j search engine mengeluarkan hasil dokumen dk , dan pasangan query dan dokumen tersebut tidak ada di **cranqrel**, maka dokumen tersebut termasuk nonrelevant dan retrieved atau false positive.
- Jika untuk query q_j search engine tidak mengeluarkan hasil dokumen dk , dan pasangan query dan dokumen tersebut ada di **cranqrel**, maka dokumen tersebut termasuk relevant dan not retrieved atau false negative.
- Jika untuk query q_j search engine tidak mengeluarkan hasil dokumen dk , dan pasangan query dan dokumen tersebut tidak ada di **cranqrel**, maka dokumen tersebut termasuk nonrelevant dan not retrieved atau true negative.

5.4 Hasil evaluasi

Kami telah melakukan evaluasi STKI yang dibuat untuk semua query dan dokumen yang ada di dataset Cranfield. mean average precision yaitu 0.31541034204554125, rata-rata recall yaitu 0.9988778918974998.

Source code: <https://github.com/faqihhusaen/STKI/tree/main/Tugas%208>

Metode Penghitungan Skor

Metode yang digunakan untuk menghitung skor pencarian adalah metode cosine. Metode cosine similarity merupakan metode untuk menghitung kesamaan antara dua buah objek yang dinyatakan dalam dua buah vector dengan menggunakan keywords (kata kunci) dari sebuah dokumen sebagai ukuran. Cosine similarity memiliki cara kerja sebagai berikut :

COSINESCORE(q)

```
1  float Scores[ $N$ ] = 0
2  float Length[ $N$ ]
3  for each query term  $t$ 
4  do calculate  $w_{t,q}$  and fetch postings list for  $t$ 
5      for each pair( $d, tf_{t,d}$ ) in postings list
6      do Scores[ $d$ ] +=  $w_{t,d} \times w_{t,q}$ 
7  Read the array Length
8  for each  $d$ 
9  do Scores[ $d$ ] = Scores[ $d$ ]/Length[ $d$ ]
10 return Top  $K$  components of Scores[]
```

Implementasi Penghitungan Skor

Fungsi unique() digunakan untuk mendapatkan daftar term tanpa duplikat dari sebuah daftar term.

```
def unique(x):
    list_set = set(x)
    unique_list = (list(list_set))
    return unique_list
```

Fungsi get_idf() berfungsi untuk mendapatkan nilai idf dari masing-masing term. Proses perhitungan dilakukan dengan pertama kali menghitung kemunculan terms pada keseluruhan dokumen terlebih dahulu dan kemudian dilakukan perhitungan idf dengan rumus $\log((1+N)/(df+1))+1$.

```
def get_idf(corpus):
    idf_dict=defaultdict(lambda: 0)
    terms_only = []
    for n in corpus:
```

```

temp = corpus[n]
for j in range(0, len(temp)):
    terms_only.append(temp[j])
unique_word = unique(terms_only)
for word in unique_word :
    count = 0
    for n in terms_only:
        if word == n:
            count += 1
    idf_dict[word] = (math.log((1+len(corpus))/(count+1)))+1
return idf_dict

```

Fungsi Term_frequency() berfungsi untuk menghitung kemunculan term pada suatu dokumen.

```

def Term_frequency(term, unique_word):
    term_frequency = defaultdict(lambda: 0)
    for word in unique_word :
        for i in range(len(term)):
            if word == term[i]:
                term_frequency[word] += 1
    return term_frequency

```

Fungsi tf_idf berfungsi untuk menghitung nilai weight tf-idf dengan menggunakan nilai term frequency (tf) dan inverse document frequency (idf).

```

def tf_idf(tf, idf):
    tf_idf = (math.log(1+tf))*idf
    return tf_idf

```

Fungsi cosine berfungsi untuk mendapatkan skor cosine untuk masing-masing dokumen. Proses perhitungan dimulai dengan mengambil setiap kata unik dari query dan dilanjutkan menghitung Term frequency (tf) dari query. Kemudian untuk setiap dokumen dilakukan perhitungan Term frequency (tf) dan tf idf. Langkah terakhir adalah menghitung score menggunakan nilai tf idf yang sudah didapatkan.

```

def cosine(document_corpus, query_corpus, N, idf):
    tf_idf_document = defaultdict(lambda: 0)

```

```

tf_idf_query = defaultdict(lambda: 0)
score = {}
query_unique_terms = unique(query_corpus)
query_tf = Term_frequency(query_corpus, query_unique_terms)
for n in document_corpus:
    product = []
    document_terms = document_corpus[n]

    document_tf = Term_frequency(document_terms, query_unique_terms)

    for w in query_unique_terms:
        weight_document = tf_idf(document_tf[w], idf[w])
        weight_query = tf_idf(query_tf[w], idf[w])
        product.append(weight_query*weight_document)
    score[n] = sum(product)/len(document_terms)
return score

```

Contoh penggunaan kode penghitungan skor

1. Pertama definisikan query

```

query = ["A", "man", "is", "so", "small", "in", "the", "wilderness",
"believe", "me", "The", "way", "how",
"people", "is", "now", "we", "are", "not",
"tailored", "to", "live", "there", "So", "when", "Stew",
"say", "he", "stumble", "across", "a",
"house", "in", "the", "middle", "of", "the", "mountain",
"my", "ears", "prick", "up"]

```

2. Definisikan term dan dokumen. Untuk di contoh yang kami pakai, kami menggunakan term dan dokumen yang berasal dari inverted index hasil indexing. Fungsi crawl_and_scrape berfungsi untuk melakukan indexing lalu menghasilkan inverted index, lalu menghasilkan data yang dibutuhkan untuk pengukuran skor dokumen

```

document_terms, index = crawl_and_scrape("foldersatu", "hasil_scraping")

```

3. Hitung nilai idf dari masing-masing term

```

idf = get_idf(document_terms)
idf

```

Contoh hasil penghitungan nilai idf masing-masing term

```
'dianigeria': 2.7047480922384253,  
'grenadinespopulation115437': 2.7047480922384253,  
'prevail': 2.01160091167848,  
'02043058date': 2.7047480922384253,  
'1327016': 2.7047480922384253,  
'oldid5662': 2.7047480922384253,  
'bcrobesson': 2.7047480922384253,  
'newid197': 2.7047480922384253,  
'newid233': 2.7047480922384253,  
'wagon': 2.7047480922384253,  
'3994347': 2.7047480922384253,  
'fnn': 2.7047480922384253,  
'cash25237': 2.7047480922384253,  
'elimini': 2.7047480922384253,  
'anybodi': 2.7047480922384253,  
'republicexportcommoditiesmanubgfactur': 2.7047480922384253,  
'oldid12663': 2.7047480922384253,  
'beveragetextilelight': 2.7047480922384253,  
'g19g77gcciaeaibrdicaoidbifadifc': 2.7047480922384253,  
'bgfactnanamalandbouderiescolombiacosta': 2.7047480922384253
```

4. Hitung skor cosine masing-masing dokumen

```
rangking = cosine(satu, query, len(satu), idf)  
rangking
```

Contoh hasil penghitungan skor cosine masing-masing dokumen

```
{1: 0.0,  
 2: 0.07504981891323427,  
 3: 0.013592243991978442,  
 4: 0.0,  
 5: 0.0,  
 6: 0.002835317589479854,  
 7: 0.0018672764076150758,  
 8: 0.0,  
 9: 0.0,  
10: 0.0}
```

