Nama & NRP : 1672063 - Yoel Oscar Werinussa

1672068 - Rifaldi

1772014 - Kelvin

1772010 - Silvia Tiffani

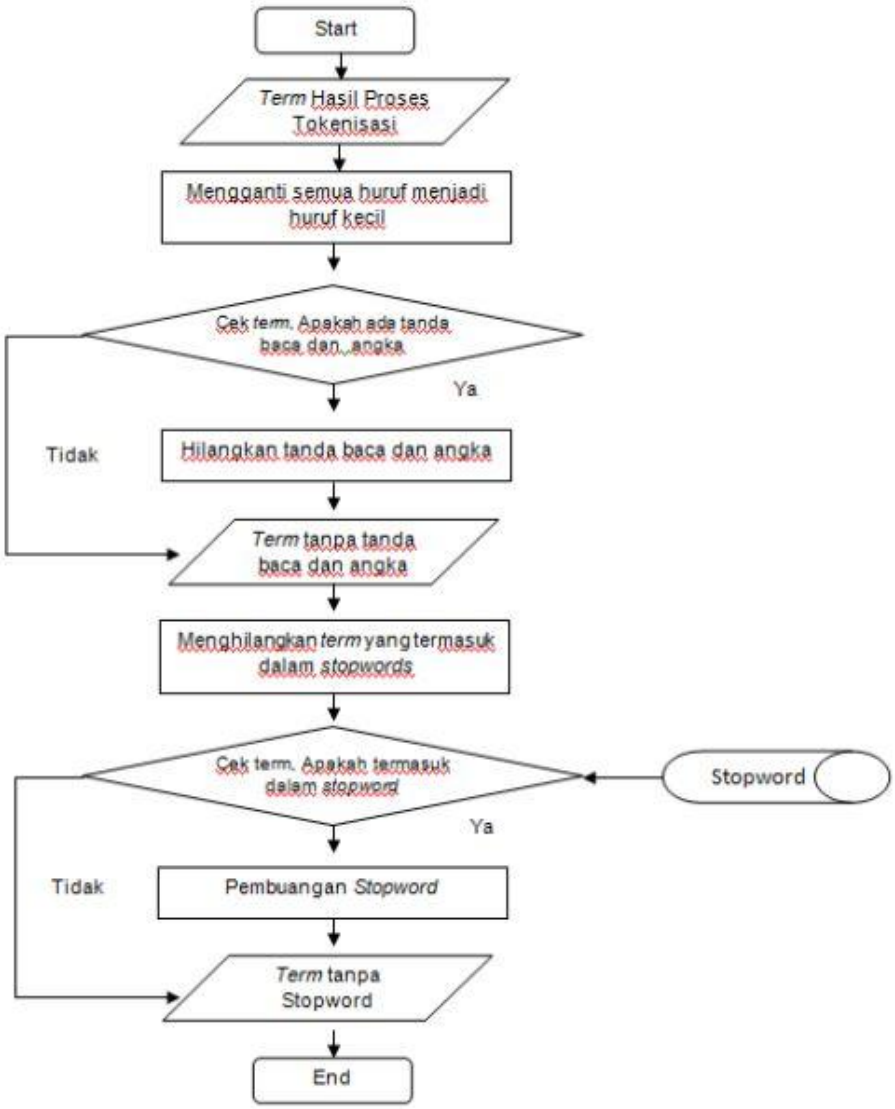
Tugas : Pertemuan 13 (Analisis Sistem Temu Balik)

1. Jelaskan alur kerja sistem. Bisa digambarkan misalnya dengan activity diagram atau flowchart.
2. Jelaskan struktur data yang digunakan untuk representasi dokumen dan kueri, misalnya: matrix, dictionary atau lainnya.
3. Bagaimanakah evaluasi performa temu balik dilakukan? Jelaskan perhitungan precision dan recall yang terjadi.
4. Gambarkan grafik P/R hasil interpolasi untuk top-10 hasil temu balik untuk kueri yang dicontohkan.
5. Berikan hasil eksekusi sistem temu balik sesuai dengan koleksi dokumen yang diberikan sebagai contoh. Bisa diberikan misalnya screen shot sesuai alur kerja sistem.

1. Jelaskan alur kerja sistem. Bisa digambarkan misalnya dengan activity diagram atau flowchart.

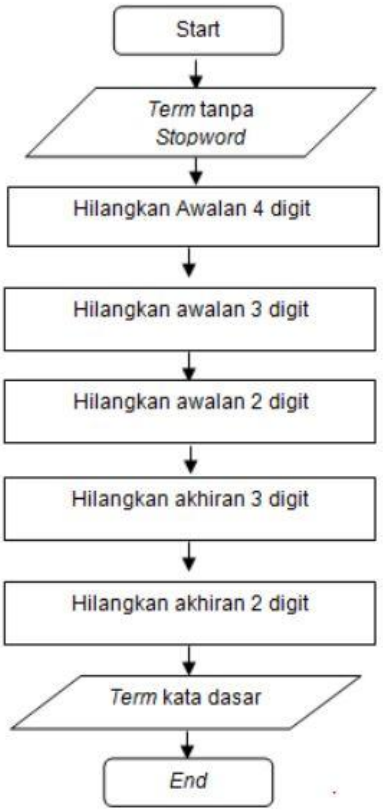
A. Flowchart Filtering

Proses Filtering dirancang untuk menghasilkan term tanpa stopwords. Flowchart filtering dimulai dengan mengganti huruf kapital menjadi huruf kecil, menghilangkan tanda baca dan angka, dan menghilangkan term yang termasuk dalam stopwords. Flowchart dapat dilihat pada gambar berikut:



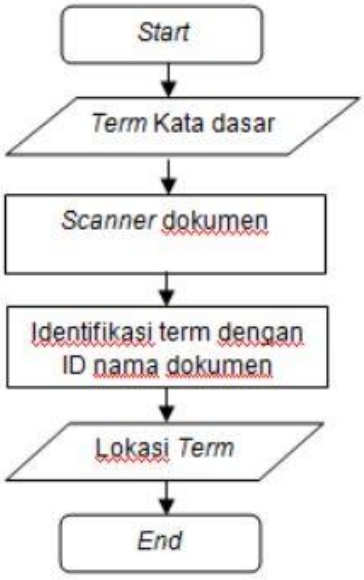

B. Flowchart Stemming

Proses stemming dirancang agar term hasil filtering diubah menjadi term kata dasar. Proses stemming dimulai dengan menghilangkan awalan dan akhiran. Proses ini juga dirancang dapat melakukan replace ketika awalan dihilangkan dan menggantinya dengan huruf yang sesuai. Proses menghilangkan awalan, akhiran, dan replace sisipan dilakukan dalam satu tahap proses. Berikut ini menunjukkan flowchart stemming:




C. Flowchar Indexing

Term kata dasar hasil proses stemming selanjutnya dimasukkan dalam tabel untuk diproses pada perhitungan Vector Space Model. Proses indexing menggunakan metode inverted indexing, yaitu dengan membedakan letak tiap term dalam dokumen. Berikut ini menunjukkan flowchart indexing:




D. Flowchart Hitung VSM

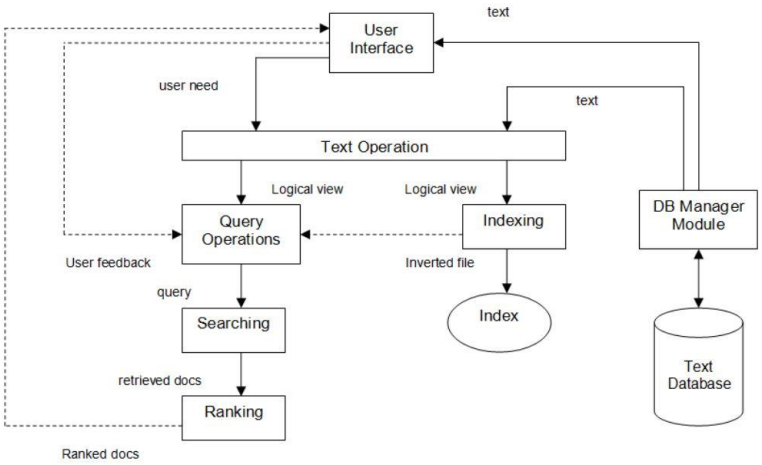
Proses selanjutnya adalah proses perhitungan pembobotan menggunakan metode VSM. Proses ini dimulai dengan perhitungan tf, idf, tfidf, jarak dokumen dan query, similaritas dan Cosine Similarity. Proses hitung VSM dirancang menghasilkan dokumen hasil pencarian disertai dengan letak dokumen dan bobot dokumen. Berikut ini menunjukkan flowchart hitung VSM:


2. Jelaskan Struktur data yang digunakan untuk representasi dokumen dan kueri,
    A. Metode
        - Information Retrieval System (IRS)

            menemukan informasi yang biasanya dalam bentuk dokumen dari sebuah data yang tidak terstruktur dalam bentuk teks untuk memenuhi kebutuhan informasi dari koleksi data yang sangat besar umumnya tersimpan dalam database computer.

            information retrieval (IRS) merupakan suatu sistem yang menemukan informasi yang sesuai dengan kebutuhan user dari kumpulan informasi secara otomatis. Aplikasi Information Retrieval System sudah digunakan dalam banyak bidang seperti dikedokteran, perusahaan dan lain sebagainya.
        - Arsitektur Information Retrieval System

            Proses Information Retrieval System seperti pada gambar yang dapat dilihat dibawah, menggunakan arsitektur yang sederhana. Sebelum dilakukannya proses temu kembali diperlukan pendefinisian database. Selanjutnya mengikuti tahapan proses; Dokumen-dokumen yang akan digunakan, Operasi yang akan digunakan dalam pencarian, dan model pengolahan teks.




- Korpus

  Penelitian dengan menggunakan database pada aplikasinya biasanya memakai korpus untuk proses pembuatan tabel pendukungnya. Penelitian empiris dapat dilakukan dengan menggunakan teks tertulis atau lisan, seperti teks-teks dasar dari berbagai jenis sastra dan analisis linguistik. Tapi gagasan tentang korpus sebagai dasar untuk sebuah bentuk linguistic empiris berbeda dalam beberapa cara mendasar dari teks-teks tertentu.

- Proses Tokenisasi

  Proses Tokenisasi merupakan proses pemisahan suatu rangkaian karakter berdasarkan karakter spasi, dan mungkin pada waktu yang bersamaan dilakukan juga proses penghapusan karakter tertentu, seperti tanda baca.

- Proses Filtering Proses

  Selanjutnya setelah dilakukan pemisahan kata pada dokumen adalah proses filtering. Filtering akan memproses kata hasil tokenisasi menjadi lebih sedikit dengan cara mengurangi kata tersebut dengan kata yang termasuk dalam stopwords. Eliminasi stopwords memiliki banyak keuntungan, yaitu akan mengurangi space pada tabel term index hingga 40% atau lebih.

- Proses Stemming

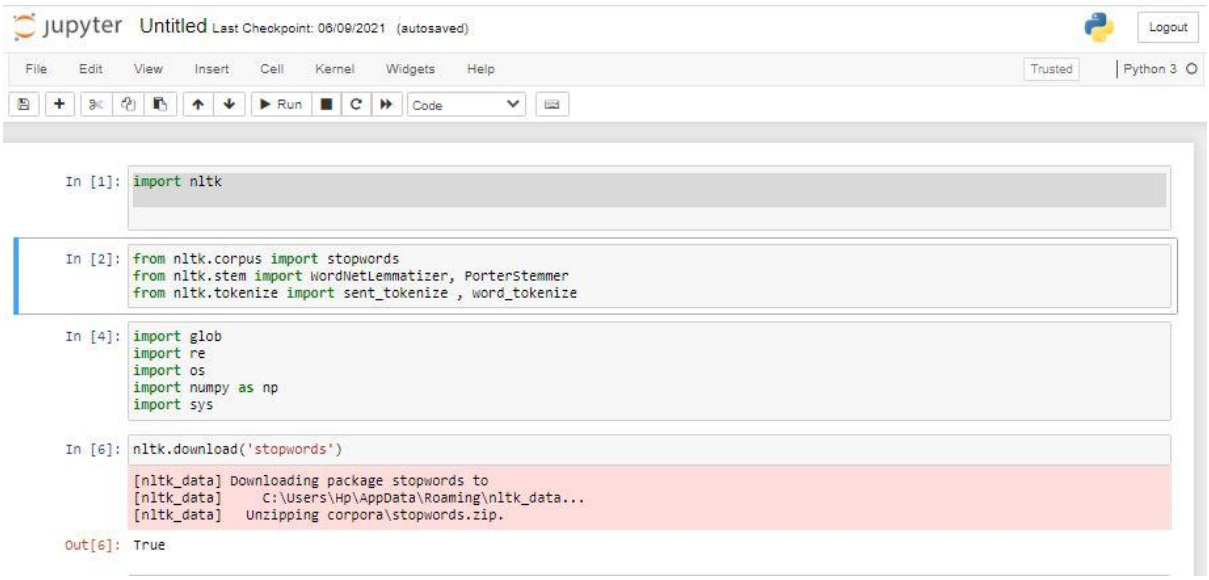
  Proses Stemming digunakan untuk mengubah term yang masih melekat dalam term tersebut awalan, sisipan, dan akhiran. Selanjutnya term tersebut diproses untuk dihilangkan awalan, sisipan dan akhiran sehingga menjadi term kata dasar.

3. Hasil tes penelusuran melalui pendekatan subjek dari 10 kueri yang berbeda, menyimpulkan bahwa rata-rata precission 0.84 dan rata-rata recall 0.04 yang berarti tingkat relevan search engine tersebut dalam menampilkan hasil yang diminta pada saat diinput menunjukan hasil yang kurang relevan dan dari rentang 1 sampai 7 saja yang relevan.

| | Recall | Precision |
|---|---|---|
| **Query 1** | 0.05 | 1.00 |
| **Query 2** | 0.02 | 1.00 |
| **Query 3** | 0.04 | 0.82 |
| **Query 4** | 0.03 | 0.80 |
| **Query 5** | 0.07 | 0.77 |
| **Query 6** | 0.06 | 0.89 |
| **Query 7** | 0.06 | 0.78 |
| **Query 8** | 0.04 | 0.77 |
| **Query 9** | 0.04 | 0.77 |
| **Query 10** | 0.04 | 0.67 |
| **Average** | 0.04 | 0.84 |

4. Pengujian *recall (P)* dan *precision (R)* dilakukan dengan cara *input query* ke dalam Information Retrieval System *input* 1 *term*, 2 *term* dan 3 *term, 4 term, 5 term, 6 term dan 7 term*. Perhitungan *recall* dan *precision* menggunakan persamaan  dan persamaan . Hasil pengujian *recall* dan *precision* dengan menguji 1 *term*, 2 *term* dan 3 *term* sampai dengan 7 term menunjukkan bahwa jika *recall* rendah maka *precision* akan tinggi,

5. Disini kami menggunakan Jupyter notebook untuk mengeksekusi program yang kami buat




Pada gambar di atas, kami mencoba mendownload package package yang dibutuhkan.

```
In [7]: StopWords = set(stopwords.words('english'))

In [8]: all_words = []
        dict_global = {}
        file_folder = 'data/*'

In [11]: idx = 1
         file_with_index = {}
         for file in glob.glob(file_folder):
             print(file)
             fname = file
             file = open(file, "r")
             text = file.read()
             text = remove_special_characters(text)
             text = re.sub(re.compile('/d'),'',text )
             sentences = sent_tokenize(text)
             words = word_tokenize(text)
             words = [word for word in words if len(words)>1]
             words = [words.lower() for word in words]
             words = [word for word in words if word not in Stopwords]
             dict_global.update(finding_all_unique_words_and_freq(words))
             files_with_index[idx] = os.path.basename(fname)
             idx = idx +1

         unique_words_all = set(dict_global.keys())
```

dan Pada gambar di atas, kami mencoba untuk melakukan penghapusan terhadap karakter karakter special dan melakukan tokenisasi dari setiap word dengan stopwords

```
In [13]: def finding_all_unique_words_and_freq(words):
             words_unique = []
             word_freq = {}
             for word in words:
                 if word not in words_unique:
                     words_unique.append(word)
             for word in words_unique:
                 word_freq[word] = words.count(word)
             return word_freq

In [14]: def finding_freq_of_word_in_doc(word,words):
             freq = words.count(word)

In [15]: def remove_special_characters(text):
             regex = re.compile('[^a-zA-Z0-9\s]')
             text_returned = re.sub(regex, '', text)
             return text_returned

In [16]: class Node:
             def __init__(self, docId, freq = None):
                 self.freq = freq
                 self.doc = docId
                 self.nextval = None
```

Diisini kita membuat fungsi untuk mencari karakter unik sampai penghapusan karakter special didalam kueri

In [16]:
```
class Node:
    def __init__(self, docId, freq = None):
        self.freq = freq
        self.doc = docId
        self.nextval = None
```

In [18]:
```
class SlinkedList:
    def __init__(self, head = None):
        self.head = head
```

In [19]:
```
linked_list_data = {}
for word in unique_words_all:
    linked_list_data[word] = SlinkedList()
    linked_list_data[word].head = Nde(1,node)
```

In [20]:
```
word_freq_in_doc = {}
idx = 1
for file in glob.glob(file_folder):
    file = open(file, "r")
    text = file.read()
    text = remove_special_characters(text)
    text = re.sub(re.compile('\d'),'', text)
    sentences = sent_tokenize(text)
    words = word_tokenize(text)
    words = [word for word in words if len(words)>1]
    words = [word.lower() for word in words]
    words = [word for word in words if word not in Stopwords]
    word_freq_in_doc = finding_all_unique_words_and_freq(words)
    for word in word_freq_in_doc.keys():
        linked_list = linked_list_data[word].head
        while linked_list.nextval is not None:
            linked_list = linked_list.nextval
        linked_list.nextval = Node(idx, word_freq_in_doc[word])
    idx = idx =1
```



dan disini kita melakukan tokenisasi tehadap kata kata yang saling berkaitan dan melakukan penghapusan jika ada karakter special didalam nya

In [ ]:
```
query = input("Enter Your Query :")
query = word_tokenize(query)
```

In [ ]:
```
connecting_words = []
cnt = 1
```

In [ ]:
```
different_word = []
```

In [ ]:
```
for word in query:
    if word.lower() != "and" and word.lower() != "or" and word.lower() != "not":
        different_words.append(word.lower())
    else:
        connecting_words.append(word.lower())
```

pada tahap ini, kami mencoba membuat kueri untuk memasukan data yang ingin kami input

In [ ]:
```
print(connecting_words)
total_files = len(files_with_index)
```

In [ ]:
```
zeroes_and_ones = []
zeroes_and_ones_of_all_words = []
for word in (different_words):
    if word.lower() in unique_words_all:
        zeroes_and_ones = [0] * total_files
        linkedlist = linked_list_data[word].head
        print(word)
        while linkedlist.nextval is not None:
            zeroes_and_ones[linkedlist.nextval.doc - 1] = 1
            linkedlist = linkedlist.nextval
        zeroes_and_ones_of_all_words.append(zeroes_and_ones)
    else:
        print(word," not found")
        sys.exit()
print(zeroes_and_ones_of_all_words)
```

In [ ]:
```
for word in connecting_words:
    word_list1 = zeroes_and_ones_of_all_words[0]
    word_list2 = zeroes_and_ones_of_all_words[1]
    if word == "and":
        bitwise_op = [w1 & w2 for (w1,w2) in
```

Dan disini kita mencoba untuk melakukan print terhadap apa yang kita kueri kan

In [ ]:
```
zip(word_list1,word_list2)]
        zeroes_and_ones_of_all_words.remove(word_list1)
        zeroes_and_ones_of_all_words.remove(word_list2)
        zeroes_and_ones_of_all_words.insert(0, bitwise_op);
    elif word == "or":
        bitwise_op = [w1 | w2 for (w1,w2) in
```

In [ ]:
```
zip(word_list1,word_list2)]
        zeroes_and_ones_of_all_words.remove(word_list1)
        zeroes_and_ones_of_all_words.remove(word_list2)
        zeroes_and_ones_of_all_words.insert(0, bitwise_op);
    elif word == "not":
        bitwise_op = [not w1 for w1 in word_list2]
        bitwise_op = [int(b == True) for b in bitwise_op]
        zeroes_and_ones_of_all_words.remove(word_list2)
        zeroes_and_ones_of_all_words.remove(word_list1)
        bitwise_op = [w1 & w2 for (w1,w2) in
```

In [ ]:
```
zeroes_and_ones_of_all_words.insert(0, bitwise_op);
```

In [ ]:
```
files = []
print(zeroes_and_ones_of_all_words)
lis = zeroes_and_ones_of_all_words[0]
cnt = 1
for index in lis:
    if index == 1:
        files.append(files_with_index[cnt])
    cnt = cnt+1

print(files)
```

dan tahap ini kami mencoba melakukan compressi terhadap kueri yang kita masukan dan melakukan print terhadap kueri tersebut.