Project Pemrosesan Text

Pemrosesan Data

Wahyu Ikbal Maulana

3323600056

D3 SDT B

Politeknik Elektronika Negeri Surabaya

In [ ]:
```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemover
from math import log10, sqrt

factory = StemmerFactory()
stemmer = factory.create_stemmer()

stopwords_factory = StopWordRemoverFactory()
stopwords = stopwords_factory.get_stop_words()

# Kata kunci
query = 'Kuning Hijau merah biru ada empat'

# Dokumen
D1 = 'Balonku ada dua rupa-rupa warnanya'
D2 = 'Hijau kuning kelabu merah muda dan biru'
D3 = 'Meletus balon hijau duar hati wahyu sangat kacau'
D4 = 'Balonku tinggal satu Kupegang erat-erat'

list_of_dokumen = [D1,D2,D3,D4]
len_of_dokumen = len(list_of_dokumen)
len_of_dokumen_with_query = len([query, D1, D2, D3, D4])
```

In [ ]:
```
# Fungsi untuk mendapatkan list dari seluruh kata yang ada pada list dokume
def get_list_of_word(list_of_dokumen, stopwords):
    list_of_word = []
    for sentence in list_of_dokumen:
        for word in stemmer.stem(sentence).split():
            stemmed_word = stemmer.stem(word)
            if word not in stopwords and stemmed_word not in list_of_word:
                list_of_word.append(stemmed_word)
    return list_of_word

list_of_word = get_list_of_word(list_of_dokumen, stopwords)
print('List of word:', list_of_word)
```

```
List of word: ['balon', 'rupa', 'warna', 'hijau', 'kuning', 'kelabu', 'mera
h', 'muda', 'biru', 'letus', 'duar', 'hati', 'wahyu', 'sangat', 'kacau', 'ti
nggal', 'satu', 'pegang', 'erat']
```

In [ ]:
```
def create_term_frequency(list_of_word, len_dokumen_with_query):
    term_frequency = []

    for i in range(len_dokumen_with_query):
        term_frequency.append(
            dict(zip(list_of_word,[0 for _ in range(len(list_of_word))])))
    return term_frequency
```

In [ ]:
```
term_frequency = create_term_frequency(list_of_word, len_of_dokumen_with_qu

for index, sentence in enumerate([query, D1, D2, D3, D4]):
```

```
    for word in stemmer.stem(sentence).split(' '):
        if word in term_frequency[index]:
            term_frequency[index][word] += 1

print(term_frequency)
```

```
[{'balon': 0, 'rupa': 0, 'warna': 0, 'hijau': 1, 'kuning': 1, 'kelabu': 0,
'merah': 1, 'muda': 0, 'biru': 1, 'letus': 0, 'duar': 0, 'hati': 0, 'wahyu':
0, 'sangat': 0, 'kacau': 0, 'tinggal': 0, 'satu': 0, 'pegang': 0, 'erat':
0}, {'balon': 1, 'rupa': 1, 'warna': 1, 'hijau': 0, 'kuning': 0, 'kelabu':
0, 'merah': 0, 'muda': 0, 'biru': 0, 'letus': 0, 'duar': 0, 'hati': 0, 'wahy
u': 0, 'sangat': 0, 'kacau': 0, 'tinggal': 0, 'satu': 0, 'pegang': 0, 'era
t': 0}, {'balon': 0, 'rupa': 0, 'warna': 0, 'hijau': 1, 'kuning': 1, 'kelab
u': 1, 'merah': 1, 'muda': 1, 'biru': 1, 'letus': 0, 'duar': 0, 'hati': 0,
'wahyu': 0, 'sangat': 0, 'kacau': 0, 'tinggal': 0, 'satu': 0, 'pegang': 0,
'erat': 0}, {'balon': 1, 'rupa': 0, 'warna': 0, 'hijau': 1, 'kuning': 0, 'ke
labu': 0, 'merah': 0, 'muda': 0, 'biru': 0, 'letus': 1, 'duar': 1, 'hati':
1, 'wahyu': 1, 'sangat': 1, 'kacau': 1, 'tinggal': 0, 'satu': 0, 'pegang':
0, 'erat': 0}, {'balon': 1, 'rupa': 0, 'warna': 0, 'hijau': 0, 'kuning': 0,
'kelabu': 0, 'merah': 0, 'muda': 0, 'biru': 0, 'letus': 0, 'duar': 0, 'hat
i': 0, 'wahyu': 0, 'sangat': 0, 'kacau': 0, 'tinggal': 1, 'satu': 1, 'pegan
g': 1, 'erat': 1}]
```

In [ ]:
```
def create_document_frequency(list_of_word):
    return dict(zip(list_of_word, [0 for _ in range(len(list_of_word))]))

dokumen_frequency = create_document_frequency(list_of_word)

for index, sentence in enumerate(term_frequency):
    if index > 0:
        for key, value in sentence.items():
            if value:
                dokumen_frequency[key] += 1

print(dokumen_frequency)
```

```
{'balon': 3, 'rupa': 1, 'warna': 1, 'hijau': 2, 'kuning': 1, 'kelabu': 1, 'm
erah': 1, 'muda': 1, 'biru': 1, 'letus': 1, 'duar': 1, 'hati': 1, 'wahyu':
1, 'sangat': 1, 'kacau': 1, 'tinggal': 1, 'satu': 1, 'pegang': 1, 'erat': 1}
```

In [ ]:
```
# Mendapatkan d_df. Di mana itu adalah pembagian antara jumlah dokumen dan
def get_d_df(length_of_document, document_frequency):
  d_df = {}

  for key, value in document_frequency.items():
    d_df[key] = length_of_document / value

  return d_df

# Mendapatkan niali idf dari d_df
def get_idf(d_df):
  idf = {}

  for key, value in d_df.items():
    idf[key] = round(log10(value), 3)
```

```
  return idf

d_df = get_d_df(len_of_dokumen, dokumen_frequency)
print(d_df)

idf = get_idf(d_df)
print(idf)
```

```
{'balon': 1.3333333333333333, 'rupa': 4.0, 'warna': 4.0, 'hijau': 2.0, 'kuni
ng': 4.0, 'kelabu': 4.0, 'merah': 4.0, 'muda': 4.0, 'biru': 4.0, 'letus': 4.
0, 'duar': 4.0, 'hati': 4.0, 'wahyu': 4.0, 'sangat': 4.0, 'kacau': 4.0, 'tin
ggal': 4.0, 'satu': 4.0, 'pegang': 4.0, 'erat': 4.0}
{'balon': 0.125, 'rupa': 0.602, 'warna': 0.602, 'hijau': 0.301, 'kuning': 0.
602, 'kelabu': 0.602, 'merah': 0.602, 'muda': 0.602, 'biru': 0.602, 'letus':
0.602, 'duar': 0.602, 'hati': 0.602, 'wahyu': 0.602, 'sangat': 0.602, 'kaca
u': 0.602, 'tinggal': 0.602, 'satu': 0.602, 'pegang': 0.602, 'erat': 0.602}
```

In [ ]:
```
def get_w_q_t(term_frequency, idf):
  w_q_t = []

  for index, document in enumerate(term_frequency):
    w_q_t.append({})

    for key, value in document.items():
      w_q_t[index][key] = value * idf[key]

  return w_q_t

w_q_t = get_w_q_t(term_frequency, idf)
print(w_q_t)
```

```
[{'balon': 0.0, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.301, 'kuning': 0.602,
'kelabu': 0.0, 'merah': 0.602, 'muda': 0.0, 'biru': 0.602, 'letus': 0.0, 'du
ar': 0.0, 'hati': 0.0, 'wahyu': 0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal':
0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0}, {'balon': 0.125, 'rupa': 0.60
2, 'warna': 0.602, 'hijau': 0.0, 'kuning': 0.0, 'kelabu': 0.0, 'merah': 0.0,
'muda': 0.0, 'biru': 0.0, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu':
0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal': 0.0, 'satu': 0.0, 'pegang': 0.
0, 'erat': 0.0}, {'balon': 0.0, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.301,
'kuning': 0.602, 'kelabu': 0.602, 'merah': 0.602, 'muda': 0.602, 'biru': 0.6
02, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu': 0.0, 'sangat': 0.0, 'ka
cau': 0.0, 'tinggal': 0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0}, {'balo
n': 0.125, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.301, 'kuning': 0.0, 'kelab
u': 0.0, 'merah': 0.0, 'muda': 0.0, 'biru': 0.0, 'letus': 0.602, 'duar': 0.6
02, 'hati': 0.602, 'wahyu': 0.602, 'sangat': 0.602, 'kacau': 0.602, 'tingga
l': 0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0}, {'balon': 0.125, 'rupa':
0.0, 'warna': 0.0, 'hijau': 0.0, 'kuning': 0.0, 'kelabu': 0.0, 'merah': 0.0,
'muda': 0.0, 'biru': 0.0, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu':
0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal': 0.602, 'satu': 0.602, 'pegang':
0.602, 'erat': 0.602}]
```

In [ ]:
```
def get_bobot_query(w_q_t, query):
    bobot_query = []
```

```
    # Iterate over each document in w_q_t, starting from the second documen
    for index, dokumen in enumerate(w_q_t):
        if index > 0:
            bobot_query.append({})
            # Stem the query and split into words
            stemmed_query_words = stemmer.stem(query).split(" ")
            for word in stemmed_query_words:
                if word in dokumen:  # Check if the word exists in the docu
                    bobot_query[index-1][word] = dokumen[word]
                else:
                    bobot_query[index-1][word] = 0  # If word not found, se

    return bobot_query


# Mendapatkan q_d di mana q_d adalah hasil pangkat dari nilai-nilai w_q_t
def get_q_d(w_q_t):
  q_d = []

  for index, document in enumerate(w_q_t):
    q_d.append({})
    total = 0

    for key, value in document.items():
      q_d[index][key] = round(value ** 2, 3)
      total += q_d[index][key]
    q_d[index]["total"] = round(sqrt(total), 3)

  # q_d.pop()
  return q_d

def get_sum_of_tf_q_d(term_frequency, bobot_kata_kunci_q_d):
  sum_of_tf_q_d = []

  for index, document in enumerate(term_frequency):
    if index > 0:
      sum_of_tf_q_d.append({})

      for key, value in document.items():
        if key in bobot_kata_kunci_q_d:
          sum_of_tf_q_d[index-1][key] = value * bobot_kata_kunci_q_d[key]

  return sum_of_tf_q_d

def get_bobot_kata_kunci_q_d(q_d, kata_kunci):
  bobot_kata_kunci_q_d = {}

  for word in stemmer.stem(kata_kunci).split(" "):
    bobot_kata_kunci_q_d[word] = q_d[0][word]

  return bobot_kata_kunci_q_d
```

In [ ]:

```
bobot_query = get_bobot_query(w_q_t, query)
print(bobot_query)
```

```
[{'kuning': 0.0, 'hijau': 0.0, 'merah': 0.0, 'biru': 0.0, 'ada': 0, 'empat': 0}, {'kuning': 0.602, 'hijau': 0.301, 'merah': 0.602, 'biru': 0.602, 'ada': 0, 'empat': 0}, {'kuning': 0.0, 'hijau': 0.301, 'merah': 0.0, 'biru': 0.0, 'ada': 0, 'empat': 0}, {'kuning': 0.0, 'hijau': 0.0, 'merah': 0.0, 'biru': 0.0, 'ada': 0, 'empat': 0}]
```

In [ ]:
```
q_d = get_q_d(w_q_t)
print(q_d)
```

```
[{'balon': 0.0, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.091, 'kuning': 0.362, 'kelabu': 0.0, 'merah': 0.362, 'muda': 0.0, 'biru': 0.362, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu': 0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal': 0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0, 'total': 1.085}, {'balon': 0.016, 'rupa': 0.362, 'warna': 0.362, 'hijau': 0.0, 'kuning': 0.0, 'kelabu': 0.0, 'merah': 0.0, 'muda': 0.0, 'biru': 0.0, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu': 0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal': 0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0, 'total': 0.86}, {'balon': 0.0, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.091, 'kuning': 0.362, 'kelabu': 0.362, 'merah': 0.362, 'muda': 0.362, 'biru': 0.362, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu': 0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal': 0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0, 'total': 1.379}, {'balon': 0.016, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.091, 'kuning': 0.0, 'kelabu': 0.0, 'merah': 0.0, 'muda': 0.0, 'biru': 0.0, 'letus': 0.362, 'duar': 0.362, 'hati': 0.362, 'wahyu': 0.362, 'sangat': 0.362, 'kacau': 0.362, 'tinggal': 0.0, 'satu': 0.0, 'pegang': 0.0, 'erat': 0.0, 'total': 1.51}, {'balon': 0.016, 'rupa': 0.0, 'warna': 0.0, 'hijau': 0.0, 'kuning': 0.0, 'kelabu': 0.0, 'merah': 0.0, 'muda': 0.0, 'biru': 0.0, 'letus': 0.0, 'duar': 0.0, 'hati': 0.0, 'wahyu': 0.0, 'sangat': 0.0, 'kacau': 0.0, 'tinggal': 0.362, 'satu': 0.362, 'pegang': 0.362, 'erat': 0.362, 'total': 1.21}]
```

In [ ]:
```
def get_bobot_kk_and_document(q_d):
  bobot_kk_and_document = {}

  for index, dokumen in enumerate(q_d):
    for key, value in dokumen.items():
      if key == "total":
        if index == 0:
          bobot_kk_and_document["bobot_kata_kunci"] = value
        else:
          bobot_kk_and_document[f"bobot_dokumen_{index}"] = value

  return bobot_kk_and_document

def get_bobot_sum_of_tf_q_d(sum_of_tf_q_d):
  bobot_sum_of_tf_q_d = {}

  for index, document in enumerate(sum_of_tf_q_d):
    total = 0
    for _, value in document.items():
      total += value
    bobot_sum_of_tf_q_d[f"bobot_sum_of_tf_q_d_{index+1}"] = total

  return bobot_sum_of_tf_q_d
```

In [ ]:

```
# Mendapatkan list dari seluruh kata-kata yang ada pada list dokumen
# di mana kata-kata tersebut bukan stopwords dan sudah di-stem
def get_list_of_word(list_of_document, stopwords):
  list_of_word = []

  for sentence in list_of_document:
    for word in stemmer.stem(sentence).split(" "):
      stemmed_word = stemmer.stem(word)
      if word not in stopwords and stemmed_word not in list_of_word:
        list_of_word.append(stemmed_word)

  return list_of_word


# membuat sebuah list yang berisi kumpulan
# word yang nilai awalnya adalah 0
def create_term_frequency(list_of_word, length_of_document_with_kk):
  term_frequency = []

  for _ in range(length_of_document_with_kk):
    term_frequency.append(
      dict(zip(list_of_word, [0 for _ in range(len(list_of_word))]))
    )

  return term_frequency

# Membuat dokumen frequency yaitu sebuah dictionary yang berisi kata-kata u
def create_document_frequency(list_of_word):
  return dict(zip(list_of_word, [0 for _ in range(len(list_of_word))]))

# Mendapatkan d_df. Di mana itu adalah pembagian antara jumlah dokumen dan
def get_d_df(length_of_document, document_frequency):
  d_df = {}

  for key, value in document_frequency.items():
    d_df[key] = length_of_document / value

  return d_df

# Mendapatkan niali idf dari d_df
def get_idf(d_df):
  idf = {}

  for key, value in d_df.items():
    idf[key] = round(log10(value), 3)

  return idf

#  Mendapatkan W_q_t. Di mana itu merupakann perkalian antara value dengan
def get_w_q_t(term_frequency, idf):
  w_q_t = []

  for index, document in enumerate(term_frequency):
    w_q_t.append({})
```

```
    for key, value in document.items():
      w_q_t[index][key] = value * idf[key]

  return w_q_t


# mendapatkan bobot kata kunci
def get_bobot_kata_kunci(w_q_t, kata_kunci):
  bobot_kata_kunci = []

  for index, document in enumerate(w_q_t):
    if index > 0:
      bobot_kata_kunci.append({})
      for word in stemmer.stem(kata_kunci).split(" "):
        bobot_kata_kunci[index-1][word] = document[word]

  return bobot_kata_kunci

# Mendapatkan q_d di mana q_d adalah hasil pangkat dari nilai-nilai w_q_t
def get_q_d(w_q_t):
  q_d = []

  for index, document in enumerate(w_q_t):
    q_d.append({})
    total = 0

    for key, value in document.items():
      q_d[index][key] = round(value ** 2, 3)
      total += q_d[index][key]
    q_d[index]["total"] = round(sqrt(total), 3)

  # q_d.pop()
  return q_d


def get_bobot_kata_kunci_q_d(q_d, kata_kunci):
  bobot_kata_kunci_q_d = {}

  for word in stemmer.stem(kata_kunci).split(" "):
    bobot_kata_kunci_q_d[word] = q_d[0][word]

  return bobot_kata_kunci_q_d


def get_bobot_kk_and_document(q_d):
  bobot_kk_and_document = {}

  for index, dokumen in enumerate(q_d):
    for key, value in dokumen.items():
      if key == "total":
        if index == 0:
          bobot_kk_and_document["bobot_kata_kunci"] = value
        else:
          bobot_kk_and_document[f"bobot_dokumen_{index}"] = value
```

```
  return bobot_kk_and_document

def get_sum_of_tf_q_d(term_frequency, bobot_kata_kunci_q_d):
  sum_of_tf_q_d = []

  for index, document in enumerate(term_frequency):
    if index > 0:
      sum_of_tf_q_d.append({})

      for key, value in document.items():
        if key in bobot_kata_kunci_q_d:
          sum_of_tf_q_d[index-1][key] = value * bobot_kata_kunci_q_d[key]

  return sum_of_tf_q_d

def get_bobot_sum_of_tf_q_d(sum_of_tf_q_d):
  bobot_sum_of_tf_q_d = {}

  for index, document in enumerate(sum_of_tf_q_d):
    total = 0
    for _, value in document.items():
      total += value
    bobot_sum_of_tf_q_d[f"bobot_sum_of_tf_q_d_{index+1}"] = total

  return bobot_sum_of_tf_q_d


def get_bobot_document_result(bobot_sum_of_tf_q_d, bobot_kata_kunci, bobot_
  return round(sqrt(bobot_sum_of_tf_q_d) / (bobot_kata_kunci / bobot_docume

# Kata Kunci
kata_kunci = "pengetahuan logistik"

# Stopwords
stopwords = stopwords_factory.get_stop_words()

# Document
document_1 = "manajemen transaksi logistik"
document_2 = "pengetahuan antara individu"
document_3 = "dalam manajemen pengetahuan terdapat transfer pengetahuan log

# List yang berisi kumpulan document dan panjang dari document
list_of_document = [document_1, document_2, document_3]
length_of_document = len(list_of_document)
length_of_document_with_kk = len(
  [kata_kunci, document_1, document_2, document_3]
)

# Kata Kunci
kata_kunci = "pengetahuan logistik"

# Stopwords
stopwords = stopwords_factory.get_stop_words()
```

```python
# Document
document_1 = "manajemen transaksi logistik"
document_2 = "pengetahuan antara individu"
document_3 = "dalam manajemen pengetahuan terdapat transfer pengetahuan log

# List yang berisi kumpulan document dan panjang dari document
list_of_document = [document_1, document_2, document_3]
length_of_document = len(list_of_document)
length_of_document_with_kk = len(
  [kata_kunci, document_1, document_2, document_3]
)

# Berisi kata-kata yang berasal dari list document
list_of_word = get_list_of_word(list_of_document, stopwords)

term_frequency = create_term_frequency(list_of_word, length_of_document_wit

# Menambahkan nilai dari term frequency sesuai dengan kemunculan di tiap do
for index, sentence in enumerate([kata_kunci, document_1, document_2, docum
  for word in stemmer.stem(sentence).split(" "):
    if word in term_frequency[index]:
      term_frequency[index][word] += 1

document_frequency = create_document_frequency(list_of_word)

# Menambahkan nilai untuk document frequency dengan cara menambahkan sesuai
for index, sentence in enumerate(term_frequency):
  if index > 0:
    for key, value in sentence.items():
      if value:
        document_frequency[key] += 1

d_df = get_d_df(length_of_document, document_frequency)

idf = get_idf(d_df)

w_q_t = get_w_q_t(term_frequency, idf)

bobot_kata_kunci = get_bobot_kata_kunci(w_q_t, kata_kunci)

q_d = get_q_d(w_q_t)

bobot_kata_kunci_q_d = get_bobot_kata_kunci_q_d(q_d, kata_kunci)

bobot_kata_kunci_and_document = get_bobot_kk_and_document(q_d)

sum_of_tf_q_d = get_sum_of_tf_q_d(term_frequency, bobot_kata_kunci_q_d)

bobot_sum_of_tf_q_d = get_bobot_sum_of_tf_q_d(sum_of_tf_q_d)


# Total bobot dari kk dan dokumen berdasarkan q/d.
total_bobot_kk = bobot_kata_kunci_and_document["bobot_kata_kunci"]
total_bobot_document_1 = bobot_kata_kunci_and_document["bobot_dokumen_1"]
total_bobot_document_2 = bobot_kata_kunci_and_document["bobot_dokumen_2"]
```

```
total_bobot_document_3 = bobot_kata_kunci_and_document["bobot_dokumen_3"]

# Total bobot dokumen dari Tf * (Wq ^ 2)
bobot_sum_of_tf_q_d_1 = bobot_sum_of_tf_q_d["bobot_sum_of_tf_q_d_1"]
bobot_sum_of_tf_q_d_2 = bobot_sum_of_tf_q_d["bobot_sum_of_tf_q_d_2"]
bobot_sum_of_tf_q_d_3 = bobot_sum_of_tf_q_d["bobot_sum_of_tf_q_d_3"]

print("Total bobot dari kk dan dokumen berdasarkan q/d: \n")

print(total_bobot_kk)
print(total_bobot_document_1)
print(total_bobot_document_2)
print(total_bobot_document_3)

print("\nTotal bobot dokumen dari Tf * (Wq ^ 2): \n")

print(bobot_sum_of_tf_q_d_1)
print(bobot_sum_of_tf_q_d_2)
print(bobot_sum_of_tf_q_d_3)

document_1_result = get_bobot_document_result(bobot_sum_of_tf_q_d_1, total_
document_2_result = get_bobot_document_result(bobot_sum_of_tf_q_d_2, total_
document_3_result = get_bobot_document_result(bobot_sum_of_tf_q_d_3, total_

print("\nHasil akhir dokumen: \n")

print(document_1_result)
print(document_2_result)
print(document_3_result)

hasil_akhir = [
  {"nama": "Dokumen 1", "nilai": document_1_result},
  {"nama": "Dokumen 2", "nilai": document_2_result},
  {"nama": "Dokumen 3", "nilai": document_3_result}
]

hasil_akhir.sort(key=lambda item: item.get("nilai"), reverse=True)

print("\nUrutan Dokumen: \n")

for item in hasil_akhir:
  print(item)
```

```
Total bobot dari kk dan dokumen berdasarkan q/d:

0.249
0.539
0.509
0.643

Total bobot dokumen dari Tf * (Wq ^ 2):

0.031
0.031
0.093

Hasil akhir dokumen:

0.381
0.36
0.788

Urutan Dokumen:

{'nama': 'Dokumen 3', 'nilai': 0.788}
{'nama': 'Dokumen 1', 'nilai': 0.381}
{'nama': 'Dokumen 2', 'nilai': 0.36}
```

In [ ]:
```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemover
stop_factory = StopWordRemoverFactory()
# more_stopword = ['dengan', 'ia','bahwa','oleh']
data = stop_factory.get_stop_words()
# stopword = stop_factory.create_stop_word_remover()
if "hanya" in data:
  print("ben")
```

```
ben
```

In [ ]:
```
print(bobot_sum_of_tf_q_d)
```

```
{'bobot_sum_of_tf_q_d_1': 0.031, 'bobot_sum_of_tf_q_d_2': 0.031, 'bobot_sum_
of_tf_q_d_3': 0.093}
```

Project Akhir Python OOP

Pemrograman 2

Dosen Pengampu Tri Hadiah Muliawati