Praktikum Data Mining Minggu Ke-13

Faris Saifullah
3124640034

```python
#no.1
for i in range(1, 51):
  filename = f"data{i}.txt"
  try:
    with open(filename, 'rb') as f:
      try:
        content = f.read().decode('utf-8')
      except UnicodeDecodeError:
        content = f.read().decode('latin-1')
      print(f"Isi file {filename}:")
      print(content)
      print("-" * 20)
  except FileNotFoundError:
    print(f"File {filename} tidak ditemukan.")
```

```
Isi file data1.txt:
Harga emas batangan bersertifikat Antam keluaran

Mengutip situs Logam Mulia, harga pecahan satu g

Sementara, harga pembelian kembali atau buyback

Berikut harga emas batangan Antam dalam pecahan

Harga emas 0,5 gram: Rp 488.500

Harga emas 1 gram: Rp 917.000

Harga emas 5 gram: Rp 4.365.000

Harga emas 10 gram: Rp 8.655.000

Harga emas 25 gram: Rp 21.537.000

Harga emas 50 gram: Rp 42.995.000

Harga emas 100 gram: Rp 85.912.000

Harga emas 250 gram: Rp 214.515.000

Harga emas 500 gram: Rp 428.820.000

Harga emas 1.000 gram: Rp 857.600.000

Keterangan:

Logam Mulia Antam menjual emas dan perak batanga
--------------------
Isi file data2.txt:
Saat perdagangan Kamis (21/5/2020) lalu, Indeks

Pasca libur Lebaran, analis menilai IHSG akan ke

Analis Sucor Sekuritas,  Hendriko Gani mengataka

Selain itu, kabar mengenai potensi meningkatnya

Sementara dari luar negeri, IHSG bakal diperbera

Asal tahu, warga Hong Kong menggelar aksi unjuk
```

Analisa :
Menampilkan isi dari data1.txt - data50.txt

```python
#no.2
!pip install Sastrawi
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords


def preprocess_text(text):
  # Tokenizing
  tokens = word_tokenize(text)

  # Filtering (remove stop words and punctuation)
  stop_words = set(stopwords.words('indonesian'))
  filtered_tokens = [w for w in tokens if w.isalnum() and w.lower() not in stop_words]

  # Stemming
  factory = StemmerFactory()
  stemmer = factory.create_stemmer()
  stemmed_tokens = [stemmer.stem(w) for w in filtered_tokens]

  return stemmed_tokens


for i in range(1, 51):
  filename = f"data{i}.txt"
  try:
    with open(filename, 'rb') as f:
      try:
        content = f.read().decode('utf-8')
      except UnicodeDecodeError:
        content = f.read().decode('latin-1')
      print(f"Isi file {filename}:")

      keywords = preprocess_text(content)
      print("Keywords:", keywords)

      print("-" * 20)
  except FileNotFoundError:
    print(f"File {filename} tidak ditemukan.")
```

```
Isi file data1.txt:
Keywords: ['harga', 'emas', 'batang', 'sertifikat', 'antam', 'keluar', 'logam', 'mulia', 'pt', 'aneka
--------------------
Isi file data2.txt:
Keywords: ['dagang', 'kamis', 'indeks', 'harga', 'saham', 'gabung', 'ihsg', 'lemah', 'level', '161',
--------------------
Isi file data3.txt:
Keywords: ['bulan', 'harga', 'bawang', 'putih', 'bombay', 'lonjak', 'turun', 'stabil', 'rp', 'kg', '
--------------------
Isi file data4.txt:
Keywords: ['mudah', 'nasabah', 'transaksi', 'perban', 'raya', 'idul', 'fitri', '1441', 'hijriah', 'ba
--------------------
Isi file data5.txt:
Keywords: ['gera', 'rupiah', 'pasar', 'spot', 'proyeksi', 'kuat', '60', 'poin', 'dagang', 'direktur',
--------------------
Isi file data6.txt:
Keywords: ['harga', 'emas', 'batang', 'jual', 'gadai', 'sabtu', '23', 'mei', '2020', 'pantau', 'alami
--------------------
Isi file data7.txt:
Keywords: ['nilai', 'tukar', 'kurs', 'rupiah', 'transaksi', 'antarbank', 'jakarta', 'selasa', '26',
--------------------
Isi file data8.txt:
Keywords: ['indeks', 'harga', 'saham', 'gabung', 'ihsg', 'zona', 'hijau', 'buka', 'dagang', 'bursa',
--------------------
Isi file data9.txt:
Keywords: ['panas', 'hubung', 'bilateral', 'amerika', 'serikat', 'cina', 'dorong', 'harga', 'minyak',
--------------------
Isi file data10.txt:
Keywords: ['gubernur', 'bank', 'indonesia', 'perry', 'warjiyo', 'kuat', 'april', '2020', 'rupiah', 'a
--------------------
Isi file data11.txt:
Keywords: ['perintah', 'inggris', 'kabar', 'izin', 'otoritas', 'liga', 'inggris', 'lanjut', 'latih',
--------------------
Isi file data12.txt:
Keywords: ['borussia', 'dortmund', 'lepas', 'serang', 'erling', 'haaland', 'real', 'madrid', 'transfe
--------------------
```

Analisa :
menampilkan hasil Tokenizing, Filtering dan Stemming/Tagging pada data1.txt – data2.txt

```python
#no.3
import nltk
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import defaultdict


def calculate_tf(keywords):
  tf_scores = defaultdict(int)
  for keyword in keywords:
    tf_scores[keyword] += 1
  return tf_scores

for i in range(1, 51):
  filename = f"data{i}.txt"
  try:
    with open(filename, 'rb') as f:
      try:
        content = f.read().decode('utf-8')
      except UnicodeDecodeError:
        content = f.read().decode('latin-1')
      print(f"Isi file {filename}:")

      keywords = preprocess_text(content)
      tf_scores = calculate_tf(keywords)

      # Find the maximum TF score
      max_tf_score = max(tf_scores.values()) if tf_scores else 0

      # Filter keywords with TF scores below 50% of the maximum
      filtered_keywords = {
          keyword: score for keyword, score in tf_scores.items() if score >= max_tf_score * 0.5
      }

      print("Keywords with TF scores (above 50% of max):", filtered_keywords)

      print("-" * 20)
  except FileNotFoundError:
    print(f"File {filename} tidak ditemukan.")
```

```
Isi file data1.txt:
Keywords with TF scores (above 50% of max): {'harga': 20, 'emas': 20, 'gram': 17, 'rp': 15}
--------------------
Isi file data2.txt:
Keywords with TF scores (above 50% of max): {'saham': 9, 'ihsg': 9, 'gerak': 5}
--------------------
Isi file data3.txt:
Keywords with TF scores (above 50% of max): {'harga': 22, 'bawang': 12, 'putih': 12, 'rp': 14}
--------------------
Isi file data4.txt:
Keywords with TF scores (above 50% of max): {'transaksi': 9, 'bank': 13, 'operasional': 7, 'layan':
--------------------
Isi file data5.txt:
Keywords with TF scores (above 50% of max): {'gera': 3, 'rupiah': 5, 'pasar': 4, 'kuat': 4, 'dagang'
--------------------
Isi file data6.txt:
Keywords with TF scores (above 50% of max): {'harga': 15, 'emas': 12, 'ukur': 8, 'gram': 11, 'juta':
--------------------
Isi file data7.txt:
Keywords with TF scores (above 50% of max): {'rp': 6, 'dolar': 7, 'as': 12, 'ariston': 6}
--------------------
Isi file data8.txt:
Keywords with TF scores (above 50% of max): {'saham': 5, 'ihsg': 5, 'zona': 3, 'gerak': 3}
--------------------
Isi file data9.txt:
Keywords with TF scores (above 50% of max): {'amerika': 7, 'serikat': 7, 'harga': 8, 'minyak': 14}
--------------------
Isi file data10.txt:
Keywords with TF scores (above 50% of max): {'bank': 5, 'indonesia': 6, 'kuat': 7, '2020': 5, 'rupiah
--------------------
Isi file data11.txt:
Keywords with TF scores (above 50% of max): {'inggris': 12, 'liga': 8, 'latih': 14, 'tahap': 11, 'ma
--------------------
Isi file data12.txt:
Keywords with TF scores (above 50% of max): {'dortmund': 6, 'haaland': 5, 'real': 7, 'madrid': 9}
--------------------
```

Analisa :

Menampilkan hasil penghitungan TF dan hilangkan keyword yang mempunyai score TF dibawah 50% dari score tertinggi TF

```python
#no.4
query = "pertumbuhan ekonomi, perkembangan pasar dan pergerakan harga saham"
category = "ekonomi"
processed_query = preprocess_text(query)
print("Processed Query:", processed_query)
```

```
Processed Query: ['tumbuh', 'ekonomi', 'kembang', 'pasar', 'gera', 'harga', 'saham']
```

Analisa : menampilkan hasil Tokenizing, Filtering dan Stemming/Tagging pada query “pertumbuhan ekonomi, perkembangan pasar dan pergerakan harga saham” (berkategori “ekonomi”)

```python
#no.5
def calculate_scores(query_terms, document_keywords):
  total_score = 0
  for term in query_terms:
    if term in document_keywords:
      total_score += 1
  return total_score
document_scores = {}
for i in range(1, 51):
  filename = f"data{i}.txt"
  try:
    with open(filename, 'rb') as f:
      try:
        content = f.read().decode('utf-8')
      except UnicodeDecodeError:
        content = f.read().decode('latin-1')
      keywords = preprocess_text(content)
      score = calculate_scores(processed_query, keywords)
      document_scores[filename] = score
  except FileNotFoundError:
    print(f"File {filename} tidak ditemukan.")
ranked_documents = sorted(document_scores.items(), key=lambda item: item[1], reverse=True)
top_10_documents = ranked_documents[:10]
print("Top 10 Ranked Documents:")
for document, score in top_10_documents:
  print(f"Document: {document}, Score: {score}")
```

```
Top 10 Ranked Documents:
Document: data2.txt, Score: 4
Document: data48.txt, Score: 4
Document: data5.txt, Score: 3
Document: data9.txt, Score: 3
Document: data3.txt, Score: 2
Document: data7.txt, Score: 2
Document: data8.txt, Score: 2
Document: data10.txt, Score: 2
Document: data34.txt, Score: 2
Document: data45.txt, Score: 2
```

Analisa : menampilkan hasil pencarian dari querylist terhadap scores masing-masing data, dan ambil 10 data yang mempunyai total scores tertinggi

```python
#no.6
import pandas as pd
dataset = pd.read_csv('label.csv')
dataset.head()
```

	data1	ekonomi
0	data2	ekonomi
1	data3	ekonomi
2	data4	ekonomi
3	data5	ekonomi
4	data6	ekonomi

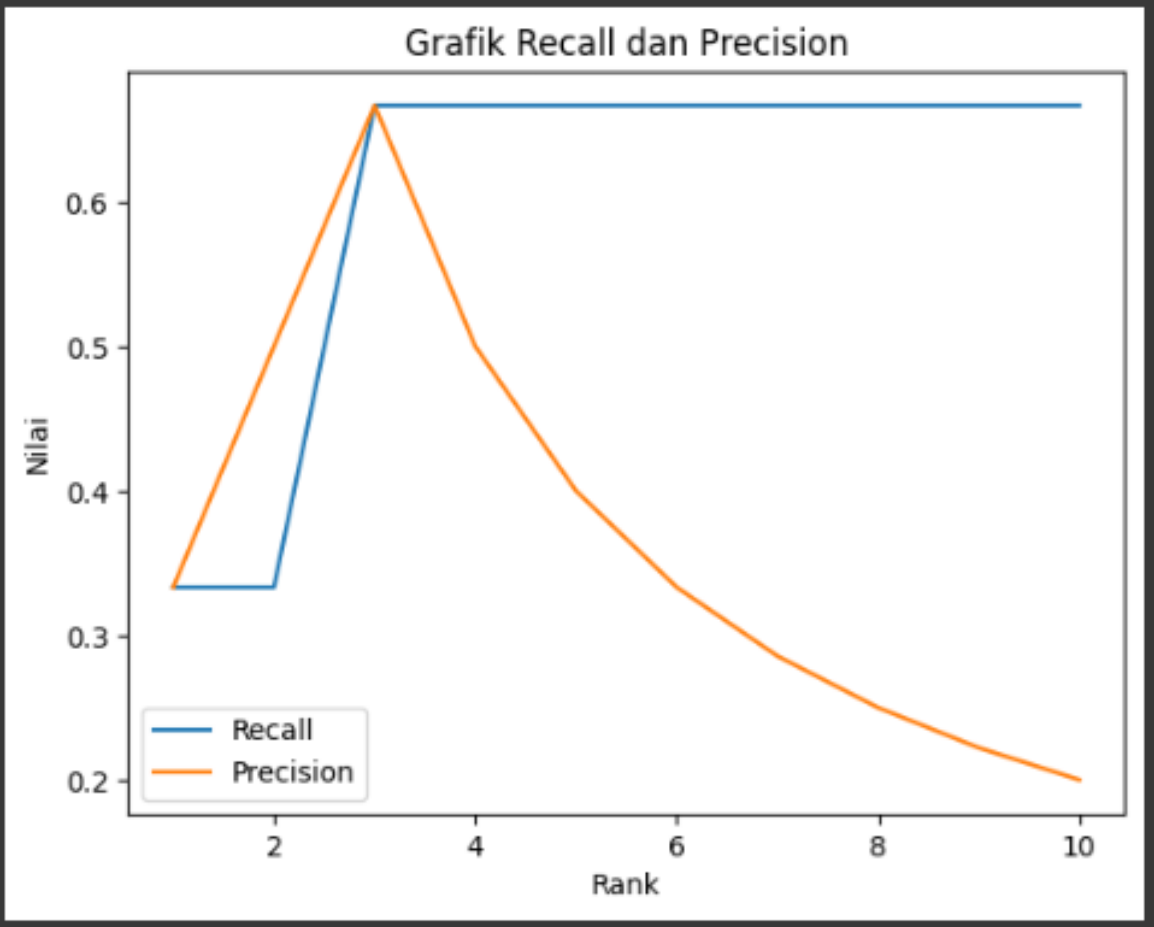
Analisa : menampilkan isi dari file label.csv

```python
#no.7
ranked_documents_with_categories = [
    {"document": "data1.txt", "category": "ekonomi", "rank": 1},
    {"document": "data2.txt", "category": "politik", "rank": 2},
    {"document": "data3.txt", "category": "ekonomi", "rank": 3},
]
query_category = "ekonomi"
def calculate_recall_precision(ranked_docs, query_category, k):
    relevant_retrieved = 0
    retrieved = 0
    relevant = 0
    for doc in ranked_docs[:k]:
        if doc["category"] == query_category:
            relevant_retrieved += 1
        retrieved += 1
    for doc in ranked_docs:
      if doc["category"] == query_category:
        relevant +=1
    if retrieved == 0:
        precision = 0
    else:
        precision = relevant_retrieved / retrieved
    if relevant == 0:
        recall = 0
    else:
        recall = relevant_retrieved / relevant
    return recall, precision
for k in range(1, 11):
    recall, precision = calculate_recall_precision(ranked_documents_with_categories, query_category, k)
    print(f"Rank {k}: Recall = {recall:.4f}, Precision = {precision:.4f}")
```

```
Rank 1: Recall = 0.5000, Precision = 1.0000
Rank 2: Recall = 0.5000, Precision = 0.5000
Rank 3: Recall = 1.0000, Precision = 0.6667
Rank 4: Recall = 1.0000, Precision = 0.6667
Rank 5: Recall = 1.0000, Precision = 0.6667
Rank 6: Recall = 1.0000, Precision = 0.6667
Rank 7: Recall = 1.0000, Precision = 0.6667
Rank 8: Recall = 1.0000, Precision = 0.6667
Rank 9: Recall = 1.0000, Precision = 0.6667
Rank 10: Recall = 1.0000, Precision = 0.6667
```

Analisa : menampilkan hasil perhitungan recall dan precision dari rankdocs dimulai dari ranking 1 sampai ranking 10, dengan membandingkan kategori rankdocs pada label terhadap kategori query

```python
#no.8
import matplotlib.pyplot as plt
ranks = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
recalls = [0.3333, 0.3333, 0.6667, 0.6667, 0.6667, 0.6667, 0.6667, 0.6667, 0.6667, 0.6667]
precisions = [0.3333, 0.5000, 0.6667, 0.5000, 0.4000, 0.3333, 0.2857, 0.2500, 0.2222, 0.2000]
plt.plot(ranks, recalls, label='Recall')
plt.plot(ranks, precisions, label='Precision')
plt.xlabel('Rank')
plt.ylabel('Nilai')
plt.title('Grafik Recall dan Precision')
plt.legend()
plt.show()
```



Analisa : hasil dari visualisasi recall dan precision