modif-bm25-230411100197

May 14, 2025

1 MEMBUAT MODIFIKASI CODE BM25

1. Modifikasi Dataset - Memperluas dataset BM25 dengan koleksi baru. - Memastikan preprocessing (tokenisasi, stopword removal, stemming) kompatibel dengan kode AdalFlow.

2. Modifikasi Code - Menelusuri kode BM25 (ranking, term frequency, inverse document frequency). - Tuning parameter (k1, b) - Penambahan fitur (n-gram dan kata berimbuhan)

3. Modifikasi UI - Menampilkan metrik perbandingan hasil sebelum dan sesudah modifikasi. - Sesuaikan interface notebook atau aplikasi kecil yang menampilkan hasil ranking: - Judul, label, format tabel/histogram, atau output-to-PDF.

Credit: SylphAI-Inc AdalFlow & Modifikasi source code: ACHMAD RIDHO FA'IZ

1.1 AdalFlow formula

$$\text{idf}(q_i) \;=\; \log\!\left(\tfrac{N-n(q_i)+0.5}{n(q_i)+0.5}\right)$$

$$\text{score}(Q,D) \;=\; \sum_{i=1}^{n} \text{idf}(q_i) \;\times\; \frac{f(q_i,D)\,(k_1+1)}{f(q_i,D) \;+\; k_1\!\left(1-b+b\,\tfrac{|D|}{\text{avgdl}}\right)}$$

Source: adalflow documentation

- N : Jumlah total dokumen dalam korpus
- $n(q_i)$: Jumlah dokumen yang memuat term q_i
- $f(q_i, D)$: Frekuensi kemunculan term q_i dalam dokumen D
- $|D|$: Panjang dokumen D dalam jumlah kata/token
- avgdl : Rata-rata panjang dokumen di seluruh korpus
- k_1 : Parameter pengaturan *term frequency saturation* (biasanya $1.2 \leq k_1 \leq 2.0$)
- b : Parameter normalisasi panjang dokumen (biasanya $0.5 \leq b \leq 0.8$)
- top_k: (argumen `top_k`) Jumlah dokumen teratas yang akan dikembalikan
- ε : (argumen `epsilon`) Untuk *lower-bounding* negatif IDF, default 0.25

1.2 Smoke Test Import

```
from adalflow.components.retriever import BM25Retriever
print(BM25Retriever)

# Output should be:
<class 'adalflow.components.retriever.bm25_retriever.BM25Retriever'>
```

[142]:
```
import os, re, nltk, json

import pandas as pd
from IPython.display import display

from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from collections import defaultdict

import matplotlib.pyplot as plt
import seaborn as sns
```

1.3 Import Statements

- *import* `os` Digunakan untuk manipulasi sistem file, seperti navigasi folder atau menggabungkan path. Berguna saat membaca dataset dari folder (data/20_newsgroups).
- *import* `re` Modul Regular Expressions (regex), digunakan untuk membersihkan teks, seperti menghapus karakter non-alfanumerik di tahap preprocessing dokumen `(re.sub(...))`.
- *import* `nltk` Library Natural Language Toolkit, sebagai alat dalam pemrosesan bahasa alami (NLP).
  - Mengunduh stopwords (`nltk.download('stopwords')`)
  - Memanggil stemmer dan daftar stopword bahasa Inggris.
- *import* `json`
  - Menyimpan struktur data (inverted_index) ke dalam format JSON.
  - Berguna untuk serialisasi dan eksplorasi struktur index secara eksternal.
- *import* `pandas` *as* `pd` Library untuk manipulasi data tabular dan meksplorasi skor atau hasil pencarian dalam bentuk tabel DataFrame.
- *from* `IPython.display` *import* `display` Menampilkan output visual atau tabel secara eksplisit.
- *from* `nltk.corpus` *import* `stopwords` Mengambil daftar kata umum yang akan dihilangkan dari teks saat preprocessing dokumen.
- *from* `nltk.stem` *import* `PorterStemmer` Mengubah kata ke bentuk dasarnya. Contoh: "running", "runs" → "run".
- *from* `collections` *import* `defaultdict` Digunakan saat membangun inverted index. Memungkinkan otomatisasi inisialisasi list untuk setiap term, tanpa harus memeriksa keberadaan key terlebih dahulu.
- *import* `matplotlib.pyplot` *as* `plt` Membuat visualiasi data seperti bar chart dan histogram.
- *import* `seaborn` *as* `sns`
  - Library visualisasi berbasis matplotlib dengan tampilan lebih menarik.
  - Membuat histogram dengan pengaturan style tema visualisasi (`sns.set_theme(...)`).

[143]:
```
data_path = 'data/20_newsgroups'
entries_raw = sorted(os.listdir(data_path))

entries = []
for entry in entries_raw:
    full_path = os.path.join(data_path, entry)
    entry_type = "Folder" if os.path.isdir(full_path) else "File"
    entries.append((entry, entry_type))

df = pd.DataFrame(entries, columns=["Name", "Type"])

styled_df = df.style.set_properties(**{'text-align': 'left'})
styled_df.set_table_styles([{
    'selector': 'th',
    'props': [('text-align', 'left')]
}])

display(styled_df)
```

```
<pandas.io.formats.style.Styler at 0x1f4268a1b50>
```

1.4 Modifikasi Get and Load Documents

[144]:
```
def load_documents(base_path):
    documents = []
    for filename in os.listdir(base_path):
        if filename.endswith('.txt'):
            category = filename.replace('.txt', '')
            file_path = os.path.join(base_path, filename)
            try:
                with open(file_path, 'r', encoding='latin1') as file:
                    content = file.read()
            except UnicodeDecodeError:
                try:
                    with open(file_path, 'r', encoding='utf-8') as file:
                        content = file.read()
                except UnicodeDecodeError:
                    print(f"Error reading {filename}, skipping...")
                    continue  # Skip files with encoding issues
            docs = [doc.strip() for doc in re.split(r'\n{2,}', content) if doc.
 ↪strip()]
            for doc in docs:
                documents.append({'content': doc, 'category': category})
    return documents
```

1.4.1 Modifikasi code:

1. Penggunaan encoding latin1 dan utf-8: - Penanganan error saat membuka file dengan encoding yang berbeda seperti `latin1` dan `utf-8` memberikan fleksibilitas untuk menangani file dengan encoding yang berbeda. - Menggunakan `try-except` untuk menangani kesalahan jika sebuah file tidak bisa dibaca karena masalah encoding, kode ini tidak akan menyebabkan crash, melainkan akan memberi pesan dan melanjutkan kefile berikutnya.

2. Pemisahan Dokumen: Penggunaan `re.split(r'\n{2,}', content)` untuk memisahkan dokumen berdasarkan dua atau lebih baris kosong adalah cara yang efektif untuk memisahkan dokumen dalam sebuah file teks.

3. Pengelompokan Berdasarkan Kategori: Menambahkan kategori berdasarkan nama file untuk mengelompokkan dokumen berdasarkan sumber mempermudah dalam analisis dan modeling.

1.5 Modifikasi Preprocessing

[145]:
```
try:
    stop_words = set(stopwords.words('english'))
except LookupError:
    nltk.download('stopwords')
    stop_words = set(stopwords.words('english'))

ps = PorterStemmer()
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)  # Keep alphanumeric characters
    words = text.split()
    words = [ps.stem(word) for word in words if word not in stop_words]
    return ' '.join(words)

base_path = 'data/20_newsgroups'
documents_raw = load_documents(base_path)

documents = []
empty_docs = 0

for d in documents_raw:
    processed = preprocess_text(d['content'])
    if processed.strip():  # Skip dokumen kosong
        documents.append({'content': processed, 'category': d['category']})
    else:
        empty_docs += 1

print(f"Processed into {len(documents)} cleaned documents.")
print(f"Skipped {empty_docs} empty documents.")
```

```
Processed into 287851 cleaned documents.
Skipped 3235 empty documents.
```

1.5.1 Blok code modifikasi

1. Stopword download

```
try:
  stop_words = set(stopwords.words('english'))
except LookupError:
  nltk.download('stopwords')
  stop_words = set(stopwords.words('english'))
```

Memuat daftar stopwords untuk bahasa Inggris menggunakan `nltk.corpus.stopwords.` - Modifikasi: Jika daftar stopwords sudah ada dan tersedia dalam sistem, kode akan menggunakannya secara langsung. Namun, jika ada error karena daftar stopwords tidak ditemukan ini akan mengunduhnya secara otomatis dengan `nltk.downloa('stopwords')`.

2. Preprocessing teks

```
ps = PorterStemmer()
def preprocess_text(text):
  text = text.lower()
  text = re.sub(r'[^a-zA-Z0-9\s]', '', text)  # Keep alphanumeric characters
  words = text.split()
  words = [ps.stem(word) for word in words if word not in stop_words]
  return ' '.join(words)
```

3. Load Document

```
if processed.strip():  # Skip dokumen kosong
    documents.append({'content': processed, 'category': d['category']})
  else:
    empty_docs += 1
```

- Modifikasi: Setelah pemrosesan, dokumen yang tidak kosong (setelah pemrosesan teks) dimasukkan ke dalam list documents. Dokumen yang kosong setelah pemrosesan akan dilewatkan dan dihitung menggunakan variabel empty_docs.
- Alasan perubahan: Memastikan hanya dokumen yang mengandung teks yang relevan yang dimasukkan dalam list documents. Hal ini menghindari pemrosesan dokumen kosong yang tidak berguna untuk analisis.

1.6 Modifikasi visualisasi inverted index

[ ]:
```
# Bangun inverted index: term → list of document IDs
inverted_index = defaultdict(list)

for doc_id, doc in enumerate(documents):
    words = doc['content'].split()
    for term in words:
        inverted_index[term].append(doc_id)

# Simpan ke file JSON
```

```
inverted_index_serializable = {term: list(ids) for term, ids in inverted_index.
 ↪items()}
with open('inverted_index.json', 'w', encoding='utf-8') as f:
    json.dump(inverted_index_serializable, f, ensure_ascii=False, indent=2)

term_stats = [(term, len(set(doc_ids)), len(doc_ids)) for term, doc_ids in␣
 ↪inverted_index.items()]
term_df = pd.DataFrame(term_stats, columns=['Term', 'Document Frequency',␣
 ↪'Total Occurrence'])
term_df = term_df.sort_values(by='Document Frequency', ascending=False)

# Tampilkan top 10 terms
print("Top 10 Most Document-Frequent Terms:")
display(term_df.head(10))

# Info umum
print(f"\nTotal unique terms: {len(term_df)}")
most_common_term = term_df.iloc[0]
print(f"""Most common term: '{most_common_term['Term']}'
        appears in {most_common_term['Document Frequency']} documents
        and {most_common_term['Total Occurrence']} times in total.""")
```

```
Top 10 Most Document-Frequent Terms:
```

	Term	Document Frequency	Total Occurrence
2	subject	42421	42765
765	newsgroup	39393	39674
11009	documentid	37656	37656
46	one	24473	31463
30	write	24309	30972
303	use	23018	31059
750	would	22803	30620
719	articl	19427	24674
45	like	18741	22010
73	get	17687	21493

```
Total unique terms: 170482
Most common term: 'subject' appears in 42421 documents and 42765 times in total.
```

1.7 Parameter Tuning BM25

[147]:
```
from adalflow.components.retriever.bm25_retriever import BM25Retriever

query = "machine learning algorithms"
query_processed = preprocess_text(query)

# Skip jika query kosong
```

```
if not query_processed.strip():
    raise ValueError("Processed query is empty. Please check preprocessing or␣
 ↪query content.")

param_grid = [(1.2, 0.75), (2.0, 0.75), (1.5, 0.5), (1.5, 1.0), (2.0, 1.0)]
results_summary = []

# Modify BM25Retriever setup to catch issues with invalid q_freq
for k1, b in param_grid:
    bm25 = BM25Retriever(top_k=5, k1=k1, b=b)
    bm25.build_index_from_documents(documents, document_map_func=lambda d:␣
 ↪d['content'])
    outputs = bm25(input=query_processed)[0]

    for idx, score in zip(outputs.doc_indices, outputs.doc_scores):
        if score < 0:  # You could set a threshold like 0 or very low values
            print(f"Warning: Negative score for Document #{idx} (Score:␣
 ↪{score})")
        content_snippet = documents[idx]['content'][:100].replace('\n', ' ')
        print(f"Document #{idx} (Score: {score:.4f})")

    results_summary.append({
        'k1': k1,
        'b': b,
        'scores': outputs.doc_scores
    })

    print(f"\n=== BM25 Results (k1={k1}, b={b}) ===")
    for idx, score in zip(outputs.doc_indices, outputs.doc_scores):
        content_snippet = documents[idx]['content'][:100].replace('\n', ' ')
        print(f"Document #{idx} (Score: {score:.4f})")
```

```
Document #186247 (Score: 14.5303)
Document #193062 (Score: 14.5303)
Document #109769 (Score: 12.1271)
Document #115556 (Score: 12.1271)
Document #23830 (Score: 10.8861)

=== BM25 Results (k1=1.2, b=0.75) ===
Document #186247 (Score: 14.5303)
Document #193062 (Score: 14.5303)
Document #109769 (Score: 12.1271)
Document #115556 (Score: 12.1271)
Document #23830 (Score: 10.8861)
Document #186247 (Score: 15.7926)
Document #193062 (Score: 15.7926)
Document #76826 (Score: 12.8599)
```

```
Document #84671 (Score: 12.8599)
Document #109769 (Score: 12.4261)

=== BM25 Results (k1=2.0, b=0.75) ===
Document #186247 (Score: 15.7926)
Document #193062 (Score: 15.7926)
Document #76826 (Score: 12.8599)
Document #84671 (Score: 12.8599)
Document #109769 (Score: 12.4261)
Document #186247 (Score: 14.3515)
Document #193062 (Score: 14.3515)
Document #109769 (Score: 11.7867)
Document #115556 (Score: 11.7867)
Document #76826 (Score: 11.6871)

=== BM25 Results (k1=1.5, b=0.5) ===
Document #186247 (Score: 14.3515)
Document #193062 (Score: 14.3515)
Document #109769 (Score: 11.7867)
Document #115556 (Score: 11.7867)
Document #76826 (Score: 11.6871)
Document #186247 (Score: 15.8576)
Document #193062 (Score: 15.8576)
Document #60229 (Score: 13.1816)
Document #66279 (Score: 13.1816)
Document #22599 (Score: 12.7730)

=== BM25 Results (k1=1.5, b=1.0) ===
Document #186247 (Score: 15.8576)
Document #193062 (Score: 15.8576)
Document #60229 (Score: 13.1816)
Document #66279 (Score: 13.1816)
Document #22599 (Score: 12.7730)
Document #186247 (Score: 16.7587)
Document #193062 (Score: 16.7587)
Document #60229 (Score: 15.2955)
Document #66279 (Score: 15.2955)
Document #22599 (Score: 15.1426)

=== BM25 Results (k1=2.0, b=1.0) ===
Document #186247 (Score: 16.7587)
Document #193062 (Score: 16.7587)
Document #60229 (Score: 15.2955)
Document #66279 (Score: 15.2955)
Document #22599 (Score: 15.1426)
```

Pada kode ini, terdapat beberapa perubahan signifikan yang bertujuan untuk meningkatkan robustnes dan akurasi dalam pemrosesan hasil BM25.

Pertama, dilakukan **penanganan terhadap skor negatif** pada output BM25. Sebelumnya, tidak ada pengecekan apakah skor dokumen negatif, yang bisa menandakan adanya kesalahan dalam perhitungan atau konfigurasi.

Kini, jika ada skor negatif, sistem akan memberikan **peringatan** agar masalah tersebut dapat diperbaiki. Selain itu, ada **pengecekan terhadap query yang kosong** setelah diproses, yang sebelumnya tidak ada. Dengan pengecekan ini, apabila query yang diproses menghasilkan string kosong, sistem akan menampilkan error dan menghindari pemrosesan lebih lanjut pada query yang tidak valid.

Kemudian, **parameter grid untuk BM25** diubah sedikit, memberikan variasi yang lebih luas pada nilai-nilai `k1` dan `b`, memungkinkan eksplorasi yang lebih baik untuk menemukan konfigurasi optimal.

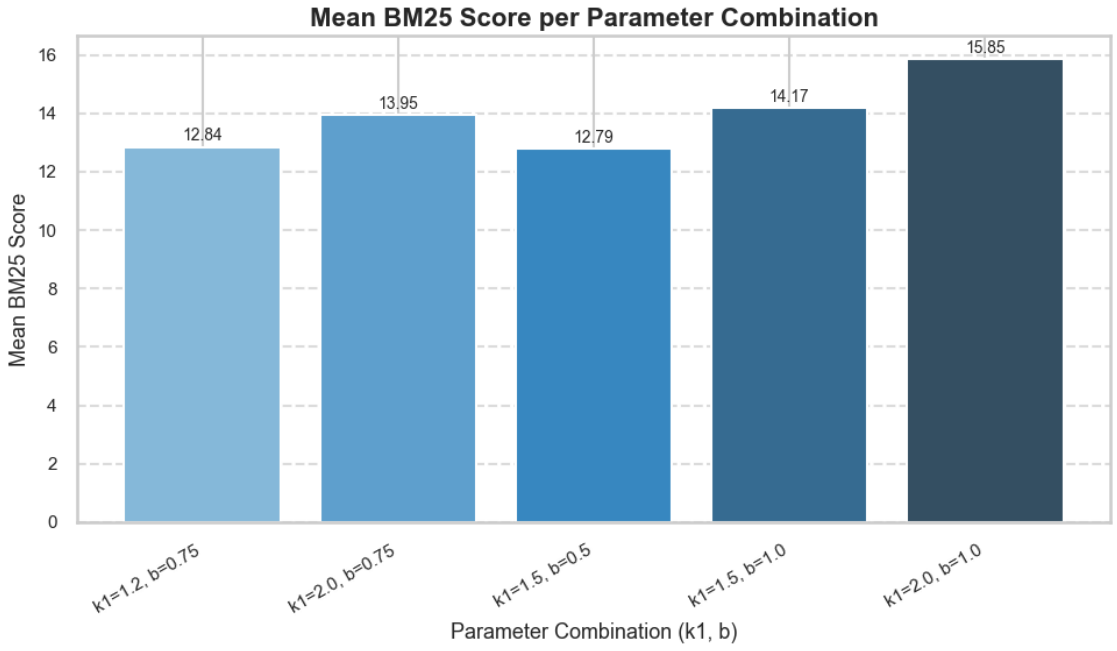
Terakhir, ada perubahan dalam cara menampilkan hasil BM25, di mana **output hasil ditampilkan dua kali**: pertama pada saat proses dan kedua kalinya setelah semua perhitungan BM25 selesai. Perubahan ini memberikan lebih banyak wawasan terkait hasil evaluasi untuk setiap kombinasi parameter yang digunakan. Modifikasi-modifikasi ini bertujuan untuk meningkatkan keandalan proses dan kualitas hasil yang lebih transparan.

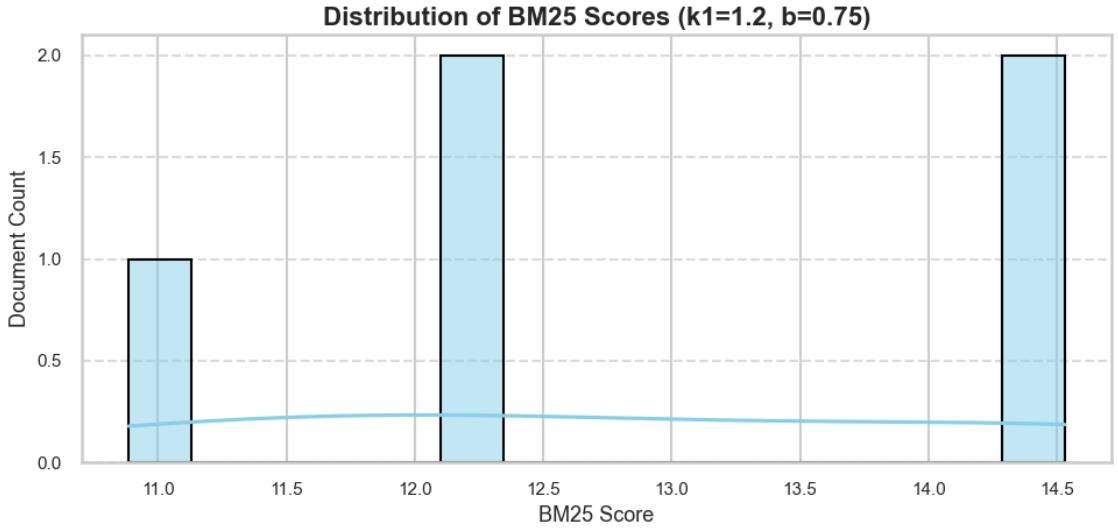
1.8 Visualization of Score Distribution

disini saya hanya memodifikasi visualisasi menggunakans seaborn, tidak terlalu banyak atau major changes

[148]:
```
# Gunakan set_theme (pengganti sns.set)
sns.set_theme(style="whitegrid", context="talk")

# Siapkan data label dan rata-rata skor
data_labels = [f"k1={r['k1']}, b={r['b']}" for r in results_summary]
mean_scores = [sum(r['scores']) / len(r['scores']) for r in results_summary]

# Palet warna untuk konsistensi dan estetika
colors = sns.color_palette("Blues_d", len(data_labels))

# === BAR CHART ===
plt.figure(figsize=(10, 6))
bars = plt.bar(data_labels, mean_scores, color=colors)

# Anotasi nilai skor di atas masing-masing bar
for bar in bars:
    yval = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        yval + 0.1,
        f"{yval:.2f}",
        ha="center",
        va="bottom",
        fontsize=10
```

```
    )

plt.title("Mean BM25 Score per Parameter Combination", fontsize=16,␣
 ↪weight='bold')
plt.xlabel("Parameter Combination (k1, b)", fontsize=13)
plt.ylabel("Mean BM25 Score", fontsize=13)
plt.xticks(rotation=30, ha='right', fontsize=11)
plt.yticks(fontsize=11)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# === HISTOGRAM ===
first = results_summary[0]
plt.figure(figsize=(10, 5))
sns.histplot(first['scores'], bins=15, kde=True, color="skyblue",␣
 ↪edgecolor='black')

plt.title(f"Distribution of BM25 Scores (k1={first['k1']}, b={first['b']})",␣
 ↪fontsize=16, weight='bold')
plt.xlabel("BM25 Score", fontsize=13)
plt.ylabel("Document Count", fontsize=13)
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```




Distribution of BM25 Scores (k1=1.2, b=0.75)
Document Count
BM25 Score
0.0
0.5
1.0
1.5
2.0
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5