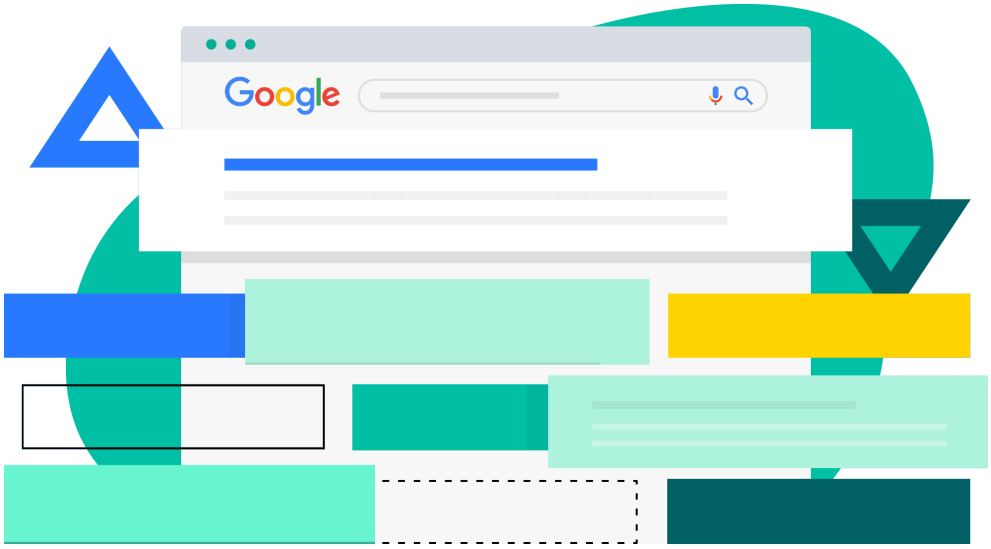
BM25 Ranking



سیدمحمدحسین هاشمی ۴۰۲۲۳۶۳۱۴۳

فروردین ۱۴۰۳

فهرست مطالب

۱ کتابخانه‌های مورد نیاز ۲

۲ کلاس Inverted Index ۲

۳ متد –init– ۲

۴ متد build_index ۳

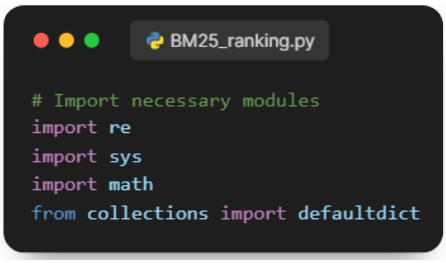
۵ متد tokenize ۳

۶ متد bm25_score ۴

۷ متد search ۴

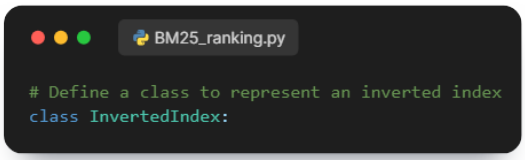
۸ استفاده ۵

۱ کتابخانه‌های مورد نیاز



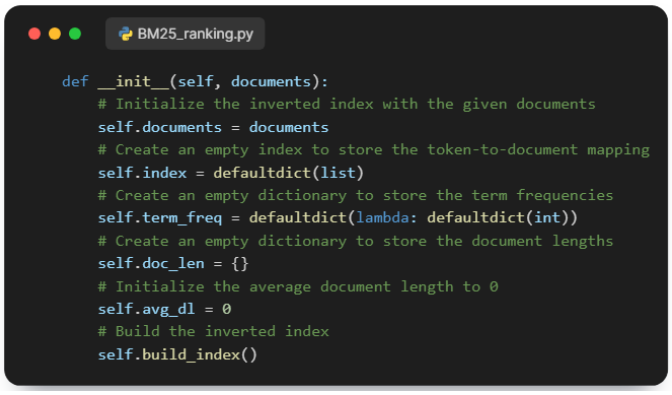
کتابخانه‌های مورد نیاز که در پروژه لود شده‌اند.

۲ کلاس Inverted Index



تمام کدهای مربوطه در این کلاس نوشته می‌شود

۳ متد –init–



بدر اینجا داکیومنت‌ها به عنوان ورودی دریافت شده و پس از تعریف متغیرهای مورد نیاز با فراخوانی build_index، Inverted Index ایجاد می‌شود.

۴ متد build_index

BM25_ranking.py

```python
def build_index(self):
    # Iterate over the documents and tokenize them
    for doc_id, document in enumerate(self.documents):
        # Tokenize the document
        tokens = self.tokenize(document)
        # Store the length of the document
        self.doc_len[doc_id] = len(tokens)
        # Add to the total document length
        self.avg_dl += len(tokens)
        # Iterate over the tokens and update the index and term frequencies
        for token in tokens:
            # Add the document ID to the token's posting list
            self.index[token].append(doc_id)
            # Increment the term frequency for the token in the document
            self.term_freq[token][doc_id] += 1
    # Calculate the average document length
    self.avg_dl /= len(self.documents)
```

در این تابع Inverted Index ساخته‌می‌شود و علاوه بر آن طول هر داکیومنت به همراه میانگین طول داکیومنت‌ها محاسبه می‌شود.

۵ متد tokenize

BM25_ranking

```python
def tokenize(self, text):
    # Tokenize the text by splitting on whitespace and converting to lowercase
    return re.split('[^a-zA-z]', text.lower())
```

در این متد از کلمات از متن استخراج می‌شوند.

۶ متد bm25_score

BM25_ranking.py

```python
def bm25_score(self, query, doc_id):
    # Calculate the BM25 score for the query and document
    score = 0
    for token in self.tokenize(query):
        # Check if the token is in the index
        if token not in self.index:
            continue
        # Calculate the document frequency (df) of the token
        df = len(self.index[token])
        # Calculate the inverse document frequency (idf) of the token
        idf = math.log((len(self.documents) - df + 0.5) / (df + 0.5))
        # Calculate the term frequency (tf) of the token in the document
        tf = self.term_freq[token][doc_id]
        # Calculate the document length (dl) of the document
        dl = self.doc_len[doc_id]
        # Calculate the BM25 score for the token
        k1 = 1.2
        b = 0.75
        numerator = tf * (k1 + 1)
        denominator = tf + k1 * (1 - b + b * dl / self.avg_dl)
        score += idf * (numerator / denominator)
    return score
```

در اینجا داکیومنت‌ها بر اساس فرمول BR25 امتیازدهی می‌شوند که فرمول آن به‌شرح زیر است:

$$score(d) = \sum_{t \in q} \left(\frac{f_{t,d} \cdot (k_۱ + ۱)}{k_۱ \cdot \left((۱ - b) + b \cdot \frac{|d|}{avgdl} \right) + f_{t,d}} \right) \cdot \log \left(\frac{N - n_t + ۰{,}۵}{n_t + ۰{,}۵} \right) \cdot \frac{(k_۲ + ۱) \cdot f_{t,q}}{k_۲ + f_{t,q}}$$

۷ متد search

BM25_ranking.py

```python
def search(self, query):
    # Calculate the BM25 score for the query and each document
    scores = {}
    for doc_id in range(len(self.documents)):
        scores[doc_id] = self.bm25_score(query, doc_id)
    # Return the sorted scores in descending order
    return sorted(scores.items(), key=lambda x: x[1], reverse=True)
```

در این متد کلمات از رشته ورودی استخراج شده و پس از آن با استفاده از متد bm25_score دسته بندی می‌شوند و براساس امتیاز مرتب شده و برگشت‌داده می‌شود.

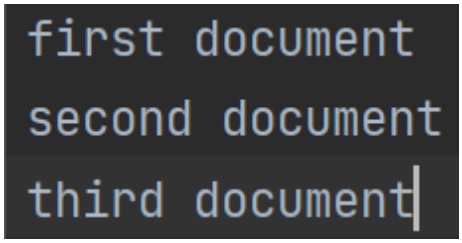
۸ استفاده

BM25_ranking.py

```python
# Usage
if __name__ == '__main__':
    # Check if the script is being run from the command line
    if len(sys.argv) != 2:
        print("Usage: python BM25_ranking.py [file_name]")
        sys.exit(1)

    # Get the filename from the command-line argument
    filename = sys.argv[1]

    # Read the documents from the file
    documents = []

    with open(filename, encoding="utf8") as file:
        for line in file:
            documents.append(line.strip())

    # Create an instance of the InvertedIndex class
    ii = InvertedIndex(documents)

    # Prompt the user to enter a query
    query = input('Search: ')

    # Search for the query and get the top 10 results
    results = ii.search(query)
    print("Results for query '%s':" % query)
    for doc_id, score in results[0: 11]:
        print("Doc %d: score: %.5f" % (doc_id, score))
```

برای استفاده از کلاس گفته شده این کد نوشته شده که بعد از فراخوانی کلاس ایجاد شده و پس از خواندن داکیومنت‌ها و ایجاد Inverted Index در آن با استفاده از رنکینگ BM25 جست‌وجو انجام می‌شود و ۱۰ نتیجه اول به همراه امتیاز خروجی داده می‌شود.

برای مثال برای ورودی مانند تصویر زیر:



خروجی مانند تصویر زیر تولید می‌شود.