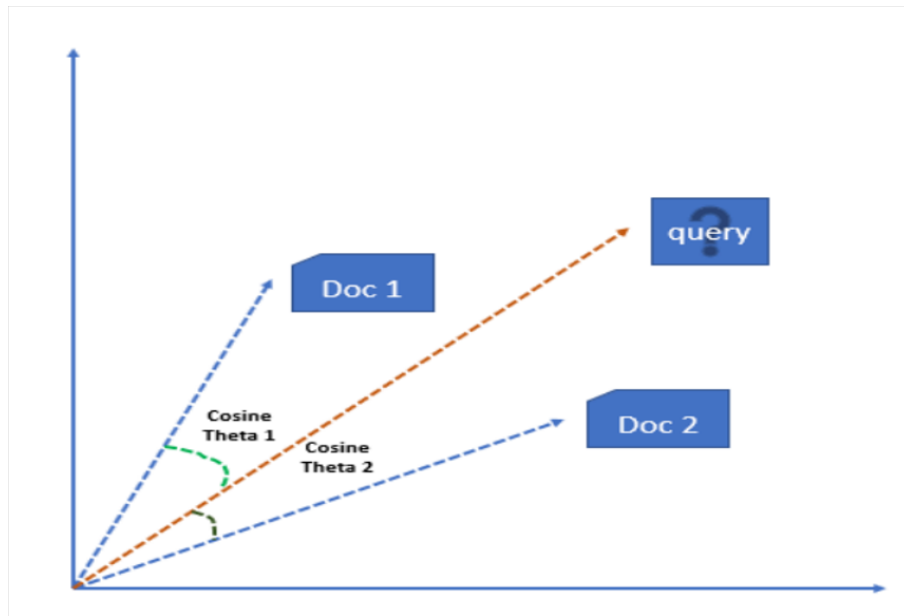


# TF-IDF cosine Ranging



سید محمد حسین هاشمی ۴۰۲۲۳۶۳۱۴۳

فروردین ۱۴۰۳

## فهرست مطالب

۲	۱ کتابخانه‌های مورد نیاز
۲	۲ کلاس Inverted Index
۲	۳ متد <code>-init-</code>
۳	۴ متد <code>__preprocess</code>
۳	۵ متد <code>__build_index</code>
۳	۶ متد <code>__tf_df</code>
۴	۷ متد <code>__compute_tfidf</code>
۴	۸ متد <code>__magnitude</code>
۴	۹ متد <code>__magnitude</code>
۵	۱۰ متد <code>__cosine_similarity</code>
۵	۱۱ متد <code>search</code>
۶	۱۲ استفاده

## ۱ کتابخانه‌های مورد نیاز

```
tf-idf-cosine_ranking.py

"""
Importing necessary libraries:
- re for regular expressions
- sys for system-specific parameters and functions
- math for mathematical functions
- collections for data structures like defaultdict
"""

import re
import sys
import math
from collections import defaultdict
```

کتابخانه‌های مورد نیاز که در پروژه لود شده‌اند.

## ۲ کلاس Inverted Index

```
tf-idf-cosine_ranking.py

"""
Class InvertedIndex: represents an inverted index data structure
- documents: list of documents to build the index from
"""

class InvertedIndex:
```

تمام کدهای مربوطه در این کلاس نوشته می‌شود

## ۳ متد \_\_init\_\_

```
tf-idf-cosine_ranking.py

def __init__(self, documents):
    # Initialize the InvertedIndex object
    self.num_documents = len(documents) # number of documents
    self.tokenized_documents = [self._preprocess(doc) for doc in documents] # tokenize each document
    self.inverted_index = defaultdict(list) # inverted index
    self.tf = defaultdict(lambda: defaultdict(int)) # term frequency (document -> term -> count)
    self.df = defaultdict(int) # document frequency (term -> count)
    self.tfidf = defaultdict(lambda: defaultdict(float)) # TF-IDF (document -> term -> score)
    self._build_index() # build the index
    self._tf_df() # compute term frequencies and document frequencies
    self._compute_tfidf() # compute TF-IDF scores
```

در اینجا متغیرهای مورد نیاز ایجاد و متدهای `build_index`، `tf_df` و `compute_tfidf` به ترتیب برای ساخت ایندکس معکوس، محاسبه `tf` و `idf` استفاده می‌شود.

## ۴ متد \_\_preprocess

```
tf-idf-cosine_ranking.py

"""
Preprocess a document by splitting it into individual tokens
"""
def __preprocess(self, text):
    return re.split('[^a-zA-z]', text.lower())
```

در این متد جداسازی کلمات از متن انجام می‌شود.

## ۵ متد \_\_build\_index

```
tf-idf-cosine_ranking.py

"""
Build the inverted index
"""
def __build_index(self):
    for doc_id, tokens in enumerate(self.tokenized_documents):
        for token in set(tokens):
            self.inverted_index[token].append(doc_id)
```

در این متد ایندکس معکوس ساخته‌شده و ذخیره می‌شود.

## ۶ متد \_\_tf\_df

```
tf-idf-cosine_ranking.py

"""
Compute term frequencies and document frequencies
"""
def __tf_df(self):
    for doc_id, tokens in enumerate(self.tokenized_documents):
        token_counts = defaultdict(int)
        for token in tokens:
            token_counts[token] += 1
        for token, count in token_counts.items():
            self.tf[doc_id][token] = count
            self.df[token] += 1
```

در این متد tf و df محاسبه و ذخیره می‌شوند.

## ۷ متد \_\_compute\_tfidf

```
tf-idf-cosine_ranking.py

"""
Compute TF-IDF scores
"""
def _compute_tfidf(self):
    for doc_id, terms in self.tf.items():
        for term, count in terms.items():
            self.tfidf[doc_id][term] = (count / len(self.tokenized_documents[doc_id])) * math.log(
                self.num_documents / (self.df[term]))
```

در این متد امتیاز tf idf برای هر داکيومنت محاسبه و ذخیره می‌شود.

## ۸ متد \_\_magnitude

```
tf-idf-cosine_ranking.py

"""
Compute the dot product of two vectors
"""
def _dot_product(self, vec1, vec2):
    return sum(vec1[term] * vec2.get(term, 0.0) for term in vec1)
```

در این متد ضرب داخلی دو بردار محاسبه می‌شود.

## ۹ متد \_\_magnitude

```
tf-idf-cosine_ranking.py

"""
Compute the magnitude of a vector
"""
def _magnitude(self, vec):
    return math.sqrt(sum(val ** 2 for val in vec.values()))
```

در این متد اندازه بردار محاسبه می‌شود.

## ۱۰ متد \_\_cosine\_\_similarity

```
tf-idf-cosine_ranking.py

"""
Compute the cosine similarity between two vectors
"""

def __cosine_similarity(self, vec1, vec2):
    return self._dot_product(vec1, vec2) / (self._magnitude(vec1) * self._magnitude(vec2))
```

در اینجا امتیاز شباهت دو بردار محاسبه می‌شود و به طور کلی فرمول آن به شرح زیر است:

$$\text{cosine\_similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

## ۱۱ متد search

```
tf-idf-cosine_ranking.py

"""
Search for documents matching a query
"""

def search(self, query):
    # Preprocess the query
    query_tokens = self._preprocess(query)
    query_tf = defaultdict(int)
    for token in query_tokens:
        query_tf[token] += 1
    query_tfidf = {
        term: (count / len(query_tokens)) * math.log(self.num_documents / (self.df.get(term, self.num_documents)))
        for term, count in query_tf.items()
    }

    # Compute similarities between the query and each document
    similarities = {}
    for doc_id in range(self.num_documents):
        similarities[doc_id] = self.__cosine_similarity(query_tfidf, self.tfidf[doc_id])

    # Return the top 10 documents with highest similarity scores
    return sorted(similarities.items(), key=lambda x: x[1], reverse=True)
```

در این متد جست‌وجو براساس امتیازاتی که در متد \_\_cosine\_\_similarity محاسبه شد انجام و ۱۰ نتیجه اول به ترتیب تشابه برگشت داده می‌شود.

```
tf-idf-cosine_ranking.py

if __name__ == '__main__':
    # Check command-line arguments
    if len(sys.argv) != 2:
        print("Usage: python BM25_ranking.py [file_name]")
        sys.exit(1)

    # Read documents from file
    filename = sys.argv[1]
    documents = []
    with open(filename, encoding="utf8") as file:
        for line in file:
            documents.append(line.strip())

    # Create an InvertedIndex object
    ii = InvertedIndex(documents)

    # Get query from user
    query = input('Search: ')

    # Search for documents matching the query
    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 داکيومنت‌ها و ایجاد Inverted Index، بردار tf-idf ایجاد و در نهایت با دریافت ورودی با استفاده از فرمول تشابه cosine نتایج دریافت و چاپ می‌شود. برای مثال برای ورودی مانند تصویر زیر:

```
first document
second document
third document|
```

خروجی مانند تصویر زیر تولید می‌شود.

```
Search: third
Results for query 'third':
Doc 2: score: 1.00000
Doc 0: score: 0.00000
Doc 1: score: 0.00000
```