Type '/' to search projects

rank-bm25 0.2.2

✓ Versión más reciente

```
pip install rank-bm25
```

Publicación: 16 feb 2022

Various BM25 algorithms for document ranking

Navegación

Descripción de proyecto

Histórico de versiones

Archivos de descarga

Verified details

These details have been verified by PyPI

Responsables

dorianbrown

Unverified details

*These details have **not** been verified by PyPI*

Enlaces del proyecto

Homepage

Metainformación

- **Licencia:** Apache2.0
- **Autor:** D. Brown

Descripción de proyecto

Rank-BM25: A two line search engine

pytest no status | pypi package 0.2.2 | DOI

A collection of algorithms for querying a set of documents and returning the ones most relevant to the query. The most common use case for these algorithms is, as you might have guessed, to create search engines.

So far the algorithms that have been implemented are:

- [x] Okapi BM25
- [x] BM25L
- [x] BM25+
- [ ] BM25-Adpt
- [ ] BM25T

These algorithms were taken from this paper, which gives a nice overview of each method, and also benchmarks them against each other. A nice inclusion is that they compare different kinds of preprocessing like stemming vs no-stemming, stopword removal or not, etc. Great read if you're new to the topic.

Installation

The easiest way to install this package is through `pip`, using

```
pip install rank_bm25
```

If you want to be sure you're getting the newest version, you can install it directly from github with

```
pip install git+ssh://git@github.com/dorianbrown/rank
```

Usage

For this example we'll be using the `BM25Okapi` algorithm, but the others are used in pretty much the same way.

Initalizing

First thing to do is create an instance of the BM25 class, which reads in a corpus of text and does some indexing on it:

```
from rank_bm25 import BM25Okapi

corpus = [
    "Hello there good man!",
    "It is quite windy in London",
    "How is the weather today?"
]

tokenized_corpus = [doc.split(" ") for doc in corpus]

bm25 = BM25Okapi(tokenized_corpus)
# <rank_bm25.BM25Okapi at 0x1047881d0>
```

Note that this package doesn't do any text preprocessing. If you want to do things like lowercasing, stopword removal, stemming, etc, you need to do it yourself.

The only requirements is that the class receives a list of lists of strings, which are the document tokens.

Ranking of documents

Now that we've created our document indexes, we can give it queries and see which documents are the most relevant:

```
query = "windy London"
tokenized_query = query.split(" ")

doc_scores = bm25.get_scores(tokenized_query)
# array([0.        , 0.93729472, 0.        ])
```

Good to note that we also need to tokenize our query, and apply the same preprocessing steps we did to the documents in order to have an apples-to-apples comparison

```
..      ..      .
```

Instead of getting the document scores, you can also just retrieve the best documents with

```
bm25.get_top_n(tokenized_query, corpus, n=1)
# ['It is quite windy in London']
```

And that's pretty much it!

Desarrollado y mantenido por la comunidad de Python para la comunidad de Python.
¡Done hoy mismo!

Mapa del sitio

Cambiar a la versión de escritorio

English › español français 日本語 português (Brasil) українська Ελληνικά Deutsch 中文 (简体) 中文 (繁體) русский עברית Esperanto 한국어