Search projects

Help Sponsor Log in Register

sqlite-fts4 0.5.2

Latest version

```
pip install sqlite-fts4
```

Released: Jan 9, 2019

Python functions for working with SQLite FTS4 search

Navigation

Project description

Release history

Download files

Project links

Homepage

Statistics

GitHub statistics:

Stars: 5

Project description

sqlite-fts4

pypi v0.5.2 | build passing | license Apache 2.0

Custom SQLite functions written in Python for ranking documents indexed using the FTS4 extension.

Read Exploring search relevance algorithms with SQLite for further details on this project.

Demo

You can try out these SQL functions using this interactive demo.

Usage

Meta

License: Apache License, Version 2.0

Author: Simon Willison

Maintainers

simonw

This module implements several custom SQLite3 functions. You can register them against an existing SQLite connection like so:

```
import sqlite3
from sqlite_fts4 import register_functions

conn = sqlite3.connect(":memory:")
register_functions(conn)
```

If you only want a subset of the functions registered you can do so like this:

```
from sqlite_fts4 import rank_score

conn = sqlite3.connect(":memory:")
conn.create_function("rank_score", 1, rank_score)
```

if you want to use these functions with Datasette you can enable them by installing the datasette-sqlite-fts4 plugin:

```
pip install datasette-sqlite-fts4
```

rank_score()

This is an extremely simple ranking function, based on an example in the SQLite documentation. It generates a score for each document using the sum of the score for each column. The score for each column is calculated as the number of search matches in that column divided by the number of search matches for every column in the index - a classic TF-IDF calculation.

You can use it in a query like this:

```
select *, rank_score(matchinfo(docs, "pcx")) as score
from docs where docs match "dog"
order by score desc
```

You *must* use the `"pcx"` matchinfo format string here, or you will get incorrect results.

rank_bm25()

An implementation of the Okapi BM25 scoring algorithm. Use it in a query like this:

```
select *, rank_bm25(matchinfo(docs, "pcnalx")) as score
from docs where docs match "dog"
order by score desc
```

You *must* use the `"pcnalx"` matchinfo format string here, or you will get incorrect results. If you see any `math domain` errors in your logs it may be because you did not use exactly the right format string here.

decode_matchinfo()

SQLite's built-in matchinfo() function returns results as a binary string. This binary represents a list of 32 bit unsigned integers, but reading the binary results is not particularly human-friendly.

The `decode_matchinfo()` function decodes the binary string and converts it into a JSON list of integers.

Usage:

```
select *, decode_matchinfo(matchinfo(docs, "pcx"))
from docs where docs match "dog"
```

Example output:

```
hello dog, [1, 1, 1, 1, 1]
```

annotate_matchinfo()

This function decodes the matchinfo document into a verbose JSON structure that describes exactly what each of the returned integers actually means.

Full documentation for the different format string options can be found here: https://www.sqlite.org/fts3.html#matchinfo

You need to call this function with the same format string as was passed to `matchinfo()` - for example:

```
select annotate_matchinfo(matchinfo(docs, "pcxnal"), "pc
from docs where docs match "dog"
```

The returned JSON will include a key for each letter in the format string. For example:

```
{
    "p": {
        "value": 1,
        "title": "Number of matchable phrases in the que
    },
    "c": {
        "value": 1,
        "title": "Number of user defined columns in the
    },
    "x": {
        "value": [
            {
                "column_index": 0,
                "phrase_index": 0,
                "hits_this_column_this_row": 1,
                "hits_this_column_all_rows": 2,
                "docs_with_hits": 2
            }
        ],
        "title": "Details for each phrase/column combina
    },
    "n": {
        "value": 3,
        "title": "Number of rows in the FTS4 table"
    },
    "a": {
        "title":"Average number of tokens in the text va
```

```
        "value": [
            {
                "column_index": 0,
                "average_num_tokens": 2
            }
        ]
    },
    "l": {
        "title": "Length of value stored in current row (
        "value": [
            {
                "column_index": 0,
                "length_of_value": 2
            }
        ]
    }
}
```


Help

Installing packages
Uploading packages
User guide
FAQs

About PyPI

PyPI on Twitter
Infrastructure dashboard
Package index name retention
Our sponsors

Contributing to PyPI

Bugs and feedback
Contribute on GitHub
Translate PyPI
Development credits

Using PyPI

Code of conduct
Report security issue
Privacy policy
Terms of use