Useful Full Text Search (FTS) queries with sqlite in Python 3.7 (https://saraswatmks.github.io/2020/04/sqlite-fts-search-queries.html)

8 minute read

Introduction

Sqlite offers a powerful way to build applications enabled with text similarity functionality. Being written in C, it's execution speed in unparallel. The ease of use and its flexibity is super nice. You could use it to build a powerful search engine in no time. In fact, I've also used for a near real time text search task in a microservice. Here are few quick pointer you should remember:

- sqlite creates an in-memory database i.e. everything gets saved into RAM. So, make sure your machine has sufficient ram, incase you are working on large datasets.
- The straight forward way of using sqlite is using an conda environment (shown below). Otherwise, it's a bit tricky to enable full text search (FTS) module with sqlite.
- Full text search (FTS) module is quite old in sqlite. We'll use the latest version, FTS5.

In this tutorial, I'll provide you some common & useful fts search queries which I've often used in my projects.

Table of Contents

1. FTS Basics

Sqlite's FTS module creates a virtual table. Think of it as a normal table in a database but on steroids i.e. powered with a blazing fast text search capabilities.

Remember the following points while using a virtual table:

- You can't specify the column types (like schema), every column would be mapped as a text (string) column.
- It supports standard table operations such as INSERT, DELETE, UPDATE.
- The table assigns an implicit `rowid` ID for each row which is easily accessible (shown below).
- The recommended way to do matching is using the MATCH operator.
- The default scoring algorithm used is BM25 (https://en.wikipedia.org/wiki/Okapi_BM25) (Best matching 25) algorithm. The best match gets highest score. The default sorting in sqlite, is sort by ascending. Hence, in order to keep the result consistent, the score is multiplied by -1. This way the best match can be ranked first. Don't get confused if you see negative scores.

2. Setup conda environment

Like mentioned above, the simplest way to use fts5 module is using sqlite3 installation in conda. Let's create a new conda environment. Go to your terminal and execute the following commands:

```
>>> conda create -n pyenv python=3.7
>>> conda activate pyenv
```

Now, simply launch ipython or jupyter notebook inside the environment to access full functionalities of sqlite3.

3. Load Dataset

For this tutorial, we'll use the classic imdb data set which contains ratings for each movie.

```
import sqlite3
import pandas as pd

df = pd.read_csv('imdb_1000.csv')
```

Let's quickly explore the data and understand what we've got.

```
r, c = df.shape
print(f"The data has {r} row and {c} columns")
```

```
The data has 979 row and 6 columns
```

```
df.sample(5)
```

	star_rating	title	content_rating	genre	duration	actors_list
88	8.4	The Kid	NOT RATED	Comedy	68	[u'Charles Chaplin', u'Edna Purviance', u'Jack...
759	7.6	Robin Hood	G	Animation	83	[u'Brian Bedford', u'Phil Harris', u'Roger Mil...
572	7.8	The Birds	PG-13	Horror	119	[u'Rod Taylor', u'Tippi Hedren', u'Suzanne Ple...
773	7.6	Disconnect	R	Drama	115	[u'Jason Bateman', u'Jonah Bobo', u'Haley Ramm']
84	8.4	Requiem for a Dream	R	Drama	102	[u'Ellen Burstyn', u'Jared Leto', u'Jennifer C...

Note: For now, we'll be using just `title` , `genre` and `rating` field for query matching with basic text cleaning.

4. Indexing pandas dataframe into sqlite

Before indexing the data, let's do basic text cleaning.

```
# clean text
df['title'] = ((df['title']
                .str
                 #replace everything which is not a digit or alphabet
                .replace(r'[^a-zA-Z0-9]',' ')
                .str
                .split()
                .apply(lambda x: ' '.join([i.strip() for i in x]))
                 #convert to lowercase
                .str.lower()))
df['genre'] = df['genre'].str.lower()


# create sqlite database into local memory (RAM)
db = sqlite3.connect(':memory:')
cur = db.cursor()


# create table
cur.execute('create virtual table imdb using fts5(title, genre, rating, tokenize="porter unicode61");')
```

```
<sqlite3.Cursor at 0x11e2cd5e0>
```

sqlite offers powerful inbuilt tokenizer options. Those are:

- **unicode61**: This is the default. It normalises all tokens into unicode characters.
- **ascii**: It converts all non-ascii characters like ã, Â and matches them with their ascii version. For example: `porteño` would be matched with `porteno`.
- **porter**: It implements porter stemming algorithm. It would match happening, happened, happens to happen.

For our case, we are using a combination of unicode61 and porter tokenizer.

```
# bulk index records
cur.executemany('insert into imdb (title, genre, rating) values (?,?,?);', df[['title',
'genre','star_rating']].to_records(index=False))
db.commit()
```

5. Useful text search queries

Query 1

* wildcard is quite powerful here. It can match sub queries (matches god -> godfather) as well.

```
# match any tokens which begins with this prefix
q = 'god*'

res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH "{q}" and rating > 6
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('the godfather', 'crime', 9.2, -5.50716086129246),
 ('city of god', 'crime', 8.7, -5.126347695889207),
 ('the godfather part ii', 'crime', 9.1, -4.794793805845165),
 ('the godfather part iii', 'crime', 7.6, -4.794793805845165),
 ('aguirre the wrath of god', 'adventure', 8.0, -4.503522057306445)]
```

Query 2

```
# everything starts with this word
q = '^ The'

res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('the godfather', 'crime', 9.2, -1.30318200237765),
 ('the matrix', 'action', 8.7, -1.30318200237765),
 ('the intouchables', 'biography', 8.6, -1.30318200237765),
 ('the pianist', 'biography', 8.5, -1.30318200237765),
 ('the departed', 'crime', 8.5, -1.30318200237765)]
```

Query 3

Here it matches the titles where exists maximum 3 tokens between `the` and `a` token.

```
# match all title where there are maximum 3 tokens between "the" and "a".
# good way to match phrases
res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH 'NEAR(the a, 3)'
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('the perks of being a wallflower', 'drama', 8.1, -3.0937328318311303),
 ('perfume the story of a murderer', 'crime', 7.5, -3.0937328318311303),
 ('once upon a time in the west', 'western', 8.6, -2.9261560330036995)]
```

Query 4

Easy to use boolean operators between tokens. `AND`, `OR` and `NOT` are reserved keywords in sqlite.

```
q = 'The OR GodFather'
res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('the godfather', 'crime', 9.2, -6.81247515893631),
 ('the godfather part ii', 'crime', 9.1, -5.931261954617385),
 ('the godfather part iii', 'crime', 7.6, -5.931261954617385),
 ('the lord of the rings the return of the king', 'adventure', 8.9, -1.0803071105367759),
 ('the lord of the rings the fellowship of the ring', 'adventure', 8.8, -1.0803071105367759)]
```

Query 5

```
# hybrid query
q = 'The AND GodFather AND P*'
res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('the godfather part ii', 'crime', 9.1, -7.960962698923183),
 ('the godfather part iii', 'crime', 7.6, -7.960962698923183)]
```

Query 6

```
# hybrid query
q = 'a AND the OR a NOT of*'
res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('once upon a time in the west', 'western', 8.6, -5.2526889873547455),
 ('a prophet', 'crime', 7.9, -3.1906709638269364),
 ('boy a', 'drama', 7.7, -3.1906709638269364),
 ('the perks of being a wallflower', 'drama', 8.1, -3.0937328318311303),
 ('perfume the story of a murderer', 'crime', 7.5, -3.0937328318311303)]
```

Query 7

```
# hybrid query
q = 'com*'
res = cur.execute(f"""select *, rank
                      from imdb
                      where title MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('the ten commandments', 'adventure', 7.9, -5.326063808508073),
 ('the king of comedy', 'comedy', 7.8, -4.981592992424006),
 ('guess who s coming to dinner', 'comedy', 7.8, -4.411015485745623),
 ('master and commander the far side of the world', 'action', 7.4, -3.764289031066617)]
```

Query 8

Instead of using the search field in the sql query, we can also mention it in the search query instead.

```
# hybrid query
q = 'title : of the'
res = cur.execute(f"""select *, rank
                      from imdb
                      where imdb MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('nausicaa of the valley of the wind', 'animation', 8.2, -3.276218025553669),
 ('dawn of the planet of the apes', 'action', 7.7, -3.276218025553669),
 ('rise of the planet of the apes', 'action', 7.6, -3.276218025553669),
 ('the lord of the rings the return of the king', 'adventure', 8.9, -3.213583634242071),
 ('the lord of the rings the fellowship of the ring', 'adventure', 8.8, -3.213583634242071)]
```

Following queries would demonstrate the use of available auxilary functions in sqlite.

Query 9

`highlight` as the name suggest is useful to highlight the selected text using given values.

```
# highlight text with brackets
q = 'title : of the'
res = cur.execute(f"""select highlight(imdb, 0, '[', ']')
                      from imdb
                      where imdb MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('nausicaa [of] [the] valley [of] [the] wind',),
 ('dawn [of] [the] planet [of] [the] apes',),
 ('rise [of] [the] planet [of] [the] apes',),
 ('[the] lord [of] [the] rings [the] return [of] [the] king',),
 ('[the] lord [of] [the] rings [the] fellowship [of] [the] ring',)]
```

Query 10

`snippet` is used to extract the given search query from the text. Below, we extract upto three words around the search query.

```
# hybrid query
q = 'title : the'
res = cur.execute(f"""select snippet(imdb, 0, '[', ']', '', 3)
                      from imdb
                      where imdb MATCH "{q}"
                      ORDER BY rank
                      limit 5""").fetchall()
res
```

```
[('[the] lord of',),
 ('[the] lord of',),
 ('[the] good [the]',),
 ('[the] lord of',),
 ('[the] hobbit [the]',)]
```

Query 11

`bm25` is also provided as a auxilary function. By default, `bm25` provides equal weights to all the fields, however here we have the option to provide custom weight. Here, we provide weight = 10 for title and weight = 5 for genre.

```
# hybrid query
q = '(title : the OR of) AND (genre: Action OR Comedy)'
res = cur.execute(f"""select rowid, *, bm25(imdb, 10.0, 5.0)
                      from imdb
                      where imdb MATCH "{q}"
                      ORDER BY bm25(imdb, 10.0, 5.0)
                      limit 5""").fetchall()
res
```

```
[(581, 'the king of comedy', 'comedy', 7.8, -8.870776402745351),
 (624, 'dawn of the planet of the apes', 'action', 7.7, -8.68632546114357),
 (788, 'rise of the planet of the apes', 'action', 7.6, -8.68632546114357),
 (197, 'guardians of the galaxy', 'action', 8.1, -8.666804134844527),
 (510, 'the last of the mohicans', 'action', 7.8, -8.624016931126334)]
```

Summary

In this tutorial, we learnt about the basics of sqlite and how to write powerful fts queries for matching text using python 3.7.

Tags: python sqlite text matching

Updated: April 30, 2020

COMMENTS

What do you think?

1 Response

V **Venkat Raghavan**

a year ago edited

Hi Manish. Your post is very useful. I tried to use sqlite3 where FTS5 is not enabled. getting error as no such module:fts5
Please suggest how to overcome this error. Thanks in advance

My mail id: venkat.raghavan2018@gmail.com