querydict

Release 0.0.1

Mar 07, 2020

Contents:

1 Installation **1**

2 Quickstart **3**
- 2.1 Queries . . . 3
- 2.2 Ambiguous queries . . . 4

3 API **5**
- 3.1 Parsing queries and matching dictionary data . . . 5

4 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER 1

Installation

At the command line:

```
$ pip install querydict
```

CHAPTER 2

Quickstart

A simple example:

```
>>> from querydict.parser import QueryEngine
>>> john_1 = { "name": "John", "eye_colour": "Blue" }
>>> john_2 = { "name": "John", "eye_colour": "Green" }
>>> q = QueryEngine("name:Bob AND eye_colour:Blue")
>>> q.match(john_1)
True
>>> q.match(john_2)
False
```

2.1 Queries

More complicated queries are possible, including nested dictionaries:

```
>>> data = { "foo": { "bar": { "baz": { "wibble": "wobble" }}}}
>>> q = QueryEngine("foo.bar.baz.wibble:wobble")
>>> q.match(data)     # => True
```

And grouping inside the query:

```
>>> england = { "country": "England", "continent": "Europe", "weather": "Rainy" }
>>> spain = { "country": "Spain", "continent": "Europe", "weather": "Sunny" }
>>> canada = { "country": "Canada", "continent": "North America", "weather": "Snowy"}
>>> q = QueryEngine("(continent:Europe AND weather:Sunny) OR country:England")
>>> q.match(england)     # => True
True
>>> q.match(spain)
True
>>> q.match(canada)
False
```

2.2 Ambiguous queries

A query is considered ambiguous if the parser cannot determine how to combine different search terms. This usually happens when multiple terms are included without specifying AND or OR, for example:

```
>>> q = QueryEngine("state:Arizona weather:Wet")     # An ambiguous query, neither AND/
↪OR is specified
>>> q = QueryEngine("state:Arizona OR weather:Wet") # Well formed query, OR is given
```

The default behaviour is that AND will be used, ensuring all query terms need to match. This can be changed to OR if required, or an exception can be forced (for example to validate queries and provide feedback to the user).

```
>>> q = QueryEngine("state:Arizona weather:Wet", ambiguous_action="Exception")
Traceback (most recent call last):
...
querydict.parser.QueryException: Query contains an ambiguous (unknown) operation, use␣
↪AND or OR
```

CHAPTER 3

API

3.1 Parsing queries and matching dictionary data

This is the core of the library, which can be used to match dictionaries against a query.

3.1.1 querydict.parser

This module implements the core of querydict, in the *QueryEngine* class. Sample usage:

```
>>> from querydict.parser import QueryEngine
>>> data = { "name", "Bob" }
>>> query = QueryEngine("name:Bob")
>>> query.match("data")  # True
```

exception querydict.parser.**MatchException**
Exception raised when matching fails, for example if input dictionary contains keys with the wrong data type for the specified query.

class querydict.parser.**QueryEngine**(*query: str*, *short_circuit: bool = True*, *ambiguous_action: str = 'AND'*, *allow_bare_field: bool = False*, *max_depth: int = 10*)
Match a Lucene style query against dict data, using an abstract tree parser.

Parameters

- **query** – A Lucene style query
- **short_circuit** – Whether to terminate matching early inside AND or OR conditions (default: True)
- **ambiguous_action** – The action to use for ambiguous queries, for example "field1:value1 field2:value2" (default: "AND")
- **allow_bare_field** – Whether to allow a search term without a specified field, for example "value1" (default: False).

- **max_depth** – The maximum recursion depth when parsing a query (default: 10).

Raises *QueryException* – If the input *query* is too complex, or uses unsupported features.

match(*data: dict*, *default_field: str = None*) → bool
Match a dictionary against the configured query.

Parameters

- **data** – A dictionary containing fields and values to match against.
- **default_field** – The name of a field to use for unqualified values.

Returns True if there is a match, False otherwise

Raises *MatchException* – If there is a problem with the input data dictionary.

exception querydict.parser.**QueryException**
Exception raised when parsing a query string fails, for example if the query is invalid or uses unsupported features.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

q

querydict.parser, 5

Index

M

match() (*querydict.parser.QueryEngine method*), 6
MatchException, 5

Q

querydict.parser (*module*), 5
QueryEngine (*class in querydict.parser*), 5
QueryException, 6