# mongo-rdkit

## Chemistry in MongoDB

Google Summer of Code 2020

Christopher Zou
cwzou@berkeley.edu

## Quick Introduction

**UC Berkeley: Computer Science and Biochemistry (2018 - 2022)**

- Teaching data structures
- Healthcare consulting
- Neurotech hacking

**Let’s get in touch!**

- https://github.com/chriswzou
- https://chriswzou.github.io
- https://linkedin.com/in/chriswzou

# Overview

1. Framework + Motivation
2. mongo-rdkit:
    a. Database
    b. Search
    c. User Story
3. Results + Next Steps

# mongoDB - a NoSQL Database System

A MongoDB document

```
{
"_id": "5cf0029caff505659",
"first": "Greg",
"last": "Landrum",
"title": "mentor",
}
```

A MongoDB query

```
database.collection.find({“first”:
{“$match”: “Greg”}, “last”:
{“$match”: “Landrum”}})
```

**Object-oriented, flexible schema JSON representation**

**JSON-format query language**

**Support for:**

- Multi-field indexing
- Load balancing via sharding
- Aggregation pipelines
- Drivers for Python, Java, Ruby, etc.
- Cloud via **MongoDB Atlas**

## Why chemistry in MongoDB?

1. We can store multiple versions of the same molecule, with only *relevant information* for each molecule.

→ **store textbook *and* biologically/physiologically relevant versions**

2. Highly flexible for non-relational data

→ **no need to specify lots of NaNs**

3. MongoDB supports high-performance queries across large, complex datasets.

→ **screen the Real Database for lead molecules with high speed and specificity**

# Mongo-rdkit Overview

https://github.com/rdkit/mongo-rdkit

**Features:**

- Database: rich storage of molecules, fingerprint generation, CRUD operations with the database, scheme generation
- Search: MongoDB-optimized similarity and substructure search

**Supporting:**

- Azure pipeline
- Configurable pytest framework
- Documentation

**Package Contents:**

```
mongordkit |
    Database |
        registration.py
        write.py
        tests
    Search   |
        similarity.py
        substructure.py
        tests
```

## Database

`registration.py`

```
# Form of a molecule document, as determined by a
MolDocScheme object
molDoc = {
'rdmol': Binary(rdmol.ToBinary()),
'index': self.get_index_value(rdmol),
'smiles': Chem.MolToSmiles(rdmol),
'scheme': self.scheme_name,
'hashes': {hash_name: HASH_FUNCTIONS[hash_name](rdmol)
for hash_name in self.hashes},
'fingerprints': {fp: fp_method(rdmol) for fp, fp_method
in self.fingerprints.items()},
'value_data': {field_name: value for field_name, value in
self.value_fields.items()}
}
```

`write.py`

- Write molecules in from SDF and lists of rdmol objects
- Specify chunk size, number to write in, molecular schema

## Search

`similarity.py`

- Screen a database with a Tanimoto similarity threshold based on Morgan fingerprints.

**Optimizations:**

- Aggregation pipeline (Swain)
- Locality sensitive hashing (ChEMBL)

`substructure.py`

- Screen a database with rdkit's pattern fingerprints and HasSubstructMatch method.

**Optimizations:**

- Pattern fingerprint screening

```python
import pymongo
from rdkit import Chem
from mongordkit import Search
from mongordkit.Database import registration, write
from mongordkit.Search import similarity, substructure

# Obtain an SDF file with relevant molecules.
ChEMBL = 'chembl_27.sdf'

# Define a molecule scheme object and modify it
# so that molecules written in include all available hashes.
mol_scheme = registration.MolDocScheme()
mol_scheme.add_all_hashes()

# Write molecules into a Mongo collection.
db = pymongo.MongoClient()
write.WriteFromSDF(db.molecules, ChEMBL, mol_scheme)

# Prepare the Mongo collection for search.
Search.PrepareForSearch(db.molecules, db.fp_counts)

# Query for molecules with a Tanimoto similarity of 0.8 or above.
query_mol = Chem.MolFromSmiles("ccO")
similarity.SimSearchAggregate(query_mol, db.molecules, db.fp_counts,
0.8)

# Query for molecules that contain "ccO" as a substructure.
substructure.SubSearch(query_mol, db.molecules)
```

## Performance Results

**Write performance:**
Performance with small set: 0.03s/molecule, negligible hash calculation time
*TODO:* performance issues with datasets upward of 1M molecules.

**Similarity performance:**
Performance on 100,000 molecules: ~0.08s/molecule

**Substructure performance:**
Performance on 100,000 molecules: 0.06s/molecule

## Next Steps

PRs are very welcome! Currently proof-of-concept; lots of potential.

Extra features:

1. Multi-molecule similarity or substructure search.
2. Building out identity search with tautomer insensitive options.
3. Adding extra hashes and fingerprints, eventually allowing substructure and similarity search to have multiple fingerprint options.
4. Interface — working with something like MongoDB Compass

Optimization:

1. Multiprocessing for similarity and substructure queries.
2. Investigating write times, which are a large barrier to usage

## Acknowledgements

**Google Summer of Code**

**Mentors:**

Greg Landrum, Marco Stenta, and Peter Gedeck

**Methods:**

Matt Swain, kott of ChEMBL

Full list of references in project proposal linked here

Thanks! Any questions?