Review for: "Open Chemistry, JupyterLab, REST, and Quantum Chemistry"

Anonymous IJQC Reviewer[1]

[1]Affiliation not available

September 2, 2020

Abstract

The authors provide an overview of a client/server/compute model of computational chemistry and the associated data, visualization, and reproducibility benefits this brings. The paper offers a discussion of all layers involved from computing and web servers to a demonstration of the diversity in front-ends such as a website and Jupyter notebook that the backend web server can provide.

The paper should be after revisions for additional citations and to make it more friendly to readers not familiar with web stacks.

Please anonymize my review.

1 Referee Report

2

The authors discuss a variety of serialization formats (JSON, msgpack, JSONb), but this section would be strengthened by several additional points:

- All of the formats above describe data at rest/transit (serialized) and not the description of the data in the various programs (dictionaries in Python, maps or ptrees in C++, etc). It is important to separate these ideas as JSON isn't a formal part of any language beyond JS and the ideas should describe when this data is in use as well.
- The positives and negatives of the serialization formats were not discussed, why was JSON chosen as the format of choice? For example, Arrays are common in quantum chemistry, but are known to be slow, lossy, and larger through JSON compared to binary formats.

General comments:

- It may be worth highlighting programs like CCLib, RDKit, ASE, and more when it comes to translating in addition to Open Babel.
- According to the QCSchema documentation (https://molssi-qc-schema.readthedocs.io/en/latest/auto_topology.html), there is a definition for connectivity available.
- QCSchema basis sets appear to have also been released which supports the new Basis Set Exchange format (https://github.com/MolSSI/QCSchema/pull/62).
- It would be good to cite Jupyter, Jupyter Lab, Pub Chem, and ChemSpider as they are used in the paper.
- Many of the audience are unlikely to be aware of REST interfaces or what they empower, it would be good to describe this in more detail.

- The reasons of selecting Girder could be indicative of common Python web frameworks such as Django, Flask, FastAPI, and more. Are there specific capabilities Girder supplies over these general frameworks?
- In the *Molecule(Resource)* demonstration, the *MoleculeModel* is never explained. It appears to be an ORM, but I believe is unclear to readers. The second example also has lines for pagination and index setting which are not explained.
- Containers are not typically available on the majority of supercomputing platforms so the exclusive choice of this seems limiting. Perhaps the authors could comment on this choice and availability of this platform without containers.
- Binder and QCArchive should be cited, TorchANI's GitHub should be cited in leui of a paper. (It appears ANI is in the references, but not in the paper)

2.0.1