Postera Coding Assessment Readme

Molecule Similarity Problem

Background

A common question in chemistry is: Given a molecule, what other similar molecules have been studied previously? The ability to find similar molecules helps drug hunters better prioritize molecules (perhaps they'll want to skip molecules that previously failed common toxicological assays).

While molecule similarity is an open problem, here we will focus on the commonly used Tanimoto similarity over a vectorized representation of molecules. Each molecule is represented as a fixed-length vector of 0's and 1's, and the Tanimoto similarity (aka Jaccard coefficient) of two bit vectors is defined as the size of the intersection of their bits, divided by size of the union of the bits.

For example, the Tanimoto similarity for two bit vectors `[0, 1, 1]` and `[1, 1, 0]` is 1/3: their intersection contains 1 bit (`[0, 1, 0]`) and their union contains 3 bits (`[1, 1, 1]`).

Prompt

Given a query molecule, a list of known molecules, your task is to return the index of the known molecules, sorted by descending Tanimoto similarity. Ties should be broken by the original ordering within the list of existing molecules.

Please complete the functions in the `molecular_similarity.py` script, such that all test cases pass when running: `python molecular_similarity.py` . You will also need to fill out the function to load in test cases from the `testcases/input*.txt` files.

Each test case file follows a similar pattern. Each file contains a single query molecule and multiple known molecules as well as metadata relating to both. The general format is as follows:

1. Bit vector length of the query molecule
2. Column representing query Molecule
3. Number of known molecules
4. Bit vector length of known molecules
5. Rows of known molecules

Followup

If you were not limited to running Python in an online IDE, what different decisions would you make to design a fast similarity search service over billions of molecules?

Response to Follow-up

Designing a molecular similarity search service capable of handling billions of molecules requires several key architectural and technological decisions. I have outlined a few potential strategies below:

NoSQL Databases for Molecular Data Storage

I would consider utilizing a NoSQL database, such as MongoDB or DynamoDB, for storing molecular data. These databases excel at handling unstructured or semi-structured data, which is common in molecular datasets.

Reasons for this approach:

- NoSQL databases scale well horizontally
- They can handle increased loads and spikes by distributing data across multiple nodes
- They maintain high performance even as the dataset grows (especially for this particular molecule comparison task, which is a read heavy operation)
- They are great at handling queries across large volumes of data where entity relationships are not overly complex

Caching Mechanism for Frequently Searched Molecules

To further improve speed and performance, I would also look into caching. Specifically, I would implement logic that caches the most commonly retrieved molecules to reduce the amount of read operations conducted on the molecule database. Services such as Redis could come in handy here

Reasons for this approach:

- This approach can significantly reduce query times for these frequently accessed data, enhancing the efficiency of the system

Parallel Computing for Similarity Calculations

The similarity score calculation between a query molecule and known molecules will be parallelized to improve computational efficiency. Frameworks such as Apache Spark could be utilized for a distributed computing approach

Reasons for this approach:

- Frameworks such as Apaches spark utilize MapReduce to distribute the calculation workload across multiple nodes. When dealing with potentially millions of comparisons per query molecule, this could present noticeable performance gains