**Project Report for MSA Viewer**

Project Background

Multiple sequence alignment (MSA) is a process of aligning three or more biological sequences (DNA, RNA, or protein) in order to infer sequence homology and conduct phylogenetic analysis. Results of these analysis can be used to predict structures, functions, and evolutionary relationships between biological sequences. There are multiple tools which perform MSA, such as Clustal Omega, MAFFT, MUSCLE, and T-Coffee. These and others are available as web interfaces from the European Molecular Biology Laboratory European Bioinformatics Institute (EMBL-EBI) at <https://www.ebi.ac.uk/Tools/msa/>. They are also available as web services for programmatic access using representation state transfer (REST) and simple object access protocol (SOAP) sample clients.

All of the tools available from EMBL-EBI are related to the process of MSA, but they are unique in their function and requirements. For example, EMBOSS Cons creates a consensus sequence from an input multiple alignment, and MView creates a new web view using HTML and CSS from an input multiple alignment. Additionally, of the tools performing an MSA, only Clustal Omega, MAFFT, MUSCLE, and T-Coffee can output alignments in ClustalW format which is simple but informative output (detailed at <https://mccb.umassmed.edu/meme/doc/clustalw-format.html>).

Project Proposal (Original)

MUSCLE (Multiple Sequence Comparison by Log-Expectation) is one of the MSA tools available from EMBL-EBI. It is mostly used for protein alignments (though it can be used for nucleotide sequences) due to its high accuracy and high speed for medium-sized protein sequence alignments. The web interface tool is available at <https://www.ebi.ac.uk/Tools/msa/muscle/>, and documentation on its REST and SOAP web service clients and open API interface is available at <https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT/MUSCLE+Help+and+Documentation>.

Project Proposal (Revised)

Project Technologies and Methods

The following technologies and methods covered in the class which this project will use are the UNIX OS and filesystem, the placement and organization of files within a web application, python CGI programming, CGI and HTML templating, HTML5 document markup, page styling with CSS, and JavaScript and jQuery client-side interaction. The technologies which the project will not use are the relational database schemas and design, MySQL, and the python module to connect to MySQL.

However, the project will use the python requests module (as an “alternative” to the MySQL connector module) in order to perform HTTP requests (as an “alternative” to retrieving data from a relational database), and data from these requests is saved.

Project Discussion

Technical Difficulties

Project File Structure

**/var/www/html/kgees1/apcc-bfx**

**/css/** contains the css used for the entire project

**/files/**

**list.txt** a file containing the job IDs from running the program

**\*** Multiple files containing the Clustal results from running the program

**/img/** contains images used by the project

**/proposal\_and\_report/** contains final and draft versions of the proposal and report

**/scripts/** contains scripts which were used within the cgi

**/templates/** contains template files (used for MSA views)

**demo.cgi** creates an MSA view without input (has hardcoded job ID)

**input.html** template used for job ID input

**main.cgi** creates and MSA view with input

**README.md** provides documentation on the project

Project Links

**demo\_input.html** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/demo_input.html>

(**demo\_input.html** is a fake version of **input.html**: the form input is fake and the submit button is a button link to **demo\_view.html**)

**demo\_output.html** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/templates/demo_output.html>

(**demo\_output.html** is identical to **output.html** besides the heading (“MSA Viewer – Demo”))

**demo.cgi** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/demo.cgi>

(cgi file for **demo\_output.html**, this is a blank page (it should be showing **demo\_view.html**))

**demo\_view.html** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/templates/demo_view.html>

(**demo\_view.html** was generated by running **demo.cgi** using the **demo\_output.html** template)

**input.html** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/input.html>

(Real input form html file)

**output.html** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/templates/output.html>

(Real output html file)

**main.cgi** <http://bfx3.aap.jhu.edu/kgees1/apcc-bfx/main.cgi>

(cgi file for **input.html**, this page is a 500 internal server error page (it should be showing results using the **output.html**)

References

To Do List:

* Readme file
* Project report (1-2 pages, background, significance to field, my implementation, and discussion)
* Deploy project on server (place all files and test)
* Project tarball (tar.gz)
* Project GitHub (keep committing)
* Project links
* Project background
* Project proposal (original): describe initial plan and problems
* Project proposal (revised): describe changes and outcomes
* Project technologies and methods
* Project discussion

Notes:  
  
- install requests package  
  
- issues with running job:  
 - translating to x-www-form-urlencoded  
 - receiving a job ID to receive results  
  
- used https://www.ebi.ac.uk/Tools/services/rest/muscle/resulttypes/<jobid> to get result types  
- result types list https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT/MUSCLE+Help+and+Documentation was incomplete  
  
- considered adding a checkboxradio button to select other MSA but I can get MSA type from job ID  
- installed jinja2 package  
  
- tried javascript for 7 hours, it didn't work and I ended up losing it all

MUSCLE is available from EMBL-EBI (European Molecular Biology Laboratory European Bioinformatics Institute) as a web-based tool at <https://www.ebi.ac.uk/Tools/msa/muscle/> as part of their suite of eight web-based MSA tools. It is also available in Perl and Python as a representational state transfer (REST) sample client.

\*For this project I am going to use the Python REST MUSCLE tool.

Running the MUSCLE tool has two parts: (1) input sequences, and (2) selecting the output format. For input the tool can use either sequence input window or a sequence file upload. The acceptable formats are GCG, FATA, EMBL, GenBank, PIR, NBRF, PHYLIP, and UniProtKB/Swiss-Prot. The output formats are Pearson/FASTA, ClustalW, ClustalW (strict), HTML, GCG MSL, Phylip interleaved, and Phylip sequential. Examples of the output formats can be viewed at <https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT/Multiple+Sequence+Alignment+Tool+Output+Examples>. None of the output formats represent the data in way that is both clear and summarized, and therefore one of the main “problems” which this project will address is output redesign.

Some “redesign” ideas include summary information about conserved residues and properties of proteins. This information is shown in some formats using symbols and font color (respectively), but there is never a summary, nor can this information be viewed in any of the “result summary” files.

Project Description

The project I am proposing for Advanced Practical Computer Concepts for Bioinformatics is creating a web-based bioinformatics analysis application that uses the Python REST MUSCLE tool to perform protein MSA. The analysis output will be an improvement upon the outputs currently available at EMBL-EBI and will be stored locally\*.

\*As the tool does not use the MySQL/Chado/etc. databases, I will store the results of the submitted analyses.

How the project will run:

1. A user inputs 3 or more protein sequences into a text input field on the web-based user interface. (CSS, HTML, and JavaScript will be involved.)
2. On the server side the MUSCLE program will be run. The results will be saved and then parsed. (CGI and the server/file system will be involved.)
3. The MUSCLE results will be presented on the web-based user interface. (CSS, HTML, and JavaScript will be involved.)

Project Technologies

In general, the files will be located on the bioinformatics server in the same organization as previous projects (i.e., JavaScript files in the JavaScript folder and CSS files in the CSS folder) and the CGI will be run from the command line to “generate” the initial web-page. An additional folder will be created to store the results of submitted analysis.

The project will also be located on GitHub in my “apcc-bfx” repository.

Project Challenge

The major challenge of this project will be learning to use a REST API in the CGI. I have found some tutorials and references on how to do this, but it will be difficult because I have never done anything like it before and the majority of the documentation is on command line usage for RESTs (that I have found).