June 9, 2017
CSE: 182

Final Project 1 Report

Team Name: W/e
Team Members: Mauro Chavez, Rose Mutong Niu
Sequences: 700-800
Extra Credit: GO term collection, Indexed file creation

1) Workflow:

1a) BLASTp on UniProt(Reviewed Swiss-Prot) Data Base:

We downloaded BLASTp so that it could be run locally. The database we provided for BLASTp was downloaded from uniprot.org. We chose to use the SwissProt reviewed database downloaded as a .fasta file. BLASTp was run like it is from the command line but it was called from our python script using os.system(). This provided an output file with a list of hits to the database ordered in decreasing quality. The output file only contained the match ID and data associated with the quality of the match so we used the UniProt website to pull the webpage associated with that match ID in the form of a .txt file. These two files were saved locally and the UniProt webpage was parsed for useful information. Each line began with two characters that described the type of information contained on that line so we first checked these. We pulled information from DE(description), OC(organism classification), and DR(contained IDs associated with other databases) lines. We were able to retrieve RecNames and AltNames for the matched proteins, covering not only the name of the protein but also older names the protein was given in past publications. Additionally, we pulled the organism classification hierarchy and provided this data to Partially Hydrolyzed People. They used this data to create a graph of the most common phrases in the organism classification hierarchy. It wasn't too interesting because the broadest terms for bacterial classification dominated all the more specific family tree key words. However, if we adjusted to only graphing the most specific organism classification keywords we might have been able to get a more interesting result. Some groups wanted functional information instead of matched protein names so we also pulled the functional description for the protein from the UniProt webpage, marked with a "-!- FUNCTION" tag. Finally, we pulled the KEGG IDs and GO terms associated with this matched protein that were stored on DR lines.

1b) HMMScan on Pfam:

We used cURL to search our query sequence against the Pfam database. This was called like it is from the command line but was called from our python script using the sub process module. We used the sub process approach because it made retrieving the data returned from the cURL command easier. The hmmscan output was saved, written to a file, and was parsed for Pfam match ID's and match descriptions, marked with "acc=" and "desc=" tags. A few other groups discussed dealing with common matches returned by Pfam that complicated their data but our approach didn't seem to return these frequently matched hits. From looking at our

data, there aren't frequently returned matches that seem nonspecific to the sequence. In fact, most of the time, the top hit returned from Pfam complimented the data being returned by BLASTp. With the Pfam match ID, we pulled the Pfam website page associated with this ID as an HTML because Pfam doesn't have the functionality to display webpages as .txt files. We read line by line and found where the Pfam functional description was stored, marked with a "pfamData" flag followed by a "</h1>" flag marking the end of the header and a "<p>" flag to mark where the description was stored. We located this spot in the HTML file and saved it. There were cases where the functional description was hidden behind other tabs on the webpage and they weren't so easily located in the HTML file. However, for most of our sequences we were able to get some sort of functional description from the Pfam website.

1c) ExPASy ScanProsite package:

Prosite was very easily searched using the ScanProsite package in biopython. Very similar to our approach with BLASTp on UniProt, we pulled the ID of the best-matched hit and saved the Prosite webpage associated with that match ID in its .txt format. The Prosite match was nicely structured like the UniProt one where the .txt could be nicely parsed for keywords by looking at the first two characters of a line, finding the "DE" tag for descriptors, and saving that line.

2) Output processing:

2a) Data we collected:

In total, we saved the raw output from our searches against UniProt, Pfam, and Prosite. For each of these searches against a database, we saved the webpage of the best hit from each database's website. From the Pfam and Prosite web pages we pulled description information and protein name. From the UniProt web page we were able to pull protein name, functional description, organism classification, KEGG IDs, and GO terms. All of this data was stored in strings so that they could easily be written in any format to an output file. We collaborated with three project teams (RIP, HaramBaes, and Partially Hydrolyzed People), and each one wanted a slightly different output file. Saving all the collected data in string variables allowed us to easily accommodate the different output file structures for each group.

2b) Raw file data collection:

We stored all the output from our searches against the database, this included raw output and webpages associated with best hit, in the form of .txt files. We created a directory for each database and in these directories stored the .txt files. These files were provided to project teams HaramBaes and RIP so that they could implement hyperlink functionality to the raw data for each sequence.

2c) Indexed file (.json) creation:

At one point, all the output from the searches against each database and the webpages associated with the best hit were stored in strings. These strings were used to create a dictionary for each sequence. These dictionaries would take the database name and "databaseMatch" as keys when this data was returned from our

searches, and would store the raw output. Additionally, we used "query" as a key to store the raw sequence from the provided sequences.fasta file associated with each IDsu. Each sequence had a dictionary of this structure built for it. Every sequence's dictionary was then stored in a new dictionary where the sequence ID served as a key that was mapped to the sequences raw data dictionary. At the end of our script, we used the json package to turn this two level dictionary into an indexed file.

2d) GO-Slim terms:
We didn't add any functionality specifically for finding GO-Slim terms because they were already found from the data returned by BLASTp. We figured that the most detailed data we were getting was from BLASTp on UniProt and that they had already used this data to assign GO terms so us taking the same data to compute Go terms would be redundant. We took the best BLASTp hit against the SwissProt database and searched its webpage for GO terms. These GO terms were provided to the project 2 teams and stored when the .txt version of the webpage was saved as a local file and as a text string in the indexed file.

3) Commenting procedure:
When commenting on our sequences, we mostly relied on the functional descriptions returned from the UniProt and Pfam web pages. We created our own .tsv that contained UniProt protein name, UniProt protein function, Pfam domain name, Pfam domain function, and Prosite domain. For an initial assessment of what the query sequence might do, we looked at the UniProt match function. Most of the time, this information was supported by the Pfam data. When this was the case, we used the Pfam functional descriptions to add to the narrative of query sequence function described by the UniProt data. Very rarely did the Pfam data contradict the UniProt data, when this was the case we wrote that the query sequence could potentially do x or y. We could have implemented a scoring function to take the quality of the hit to each database and combine it with the quality of annotation to determine which database is providing the better information. Sometimes the Pfam data provided information on one functional component of the protein, for example Helix-Turn-Helix domain. When this was the case, we explained that the function described by UniProt was potentially made possible by the characteristics of the functional domain described by Pfam. In the cases where no Pfam hit was found, we used the UniProt functional description and commented on weather or not the Prosite data confirmed or complicated things. All of our sequences provided at least 1 hit from BLASTp. There were a few cases where two queries provided the same data from BLASTp , when this was a case we made a note in our comments as to which one had the larger BitScore and lower E-value. We also made a note when we had a result with a E-value of zero. There were also some cases where UniProt provided less information than Pfam. In this scenario we looked for common terms across the data returned for the given sequence. If there was an agreement of keywords, we relied more heavily on the Pfam description and noted that the query sequence could potentially take on one or more of the characteristics of the protein domain described by Pfam.

4) Proteins of interest

ABO12227:Multidrug export protein EmrA

We found this protein interesting because it confers resistance to antibiotics potentially through exporting of poisonous ions. The proteome we were given was from Acinetobacter baumanni, a bacteria with a high rate of antibiotic resistance, so we paid close attention to any results discussing antibiotics. If this protein is not commonly found in other bacteria cells, it could be one of the features contributing to the problems associated with Acinetobacter baumannii. We also found it interesting that another sequence with different ID mapped to the same UniProt protein but with different function. In this case, there were two separate annotations for the same protein. This got us thinking that it would be interesting to provide information on the quality of annotation to help users make sense of these types of results.

ABO12585:Luciferase-like monooxygenase

We found this protein interesting because it is probably an oxidative enzyme that produces bioluminescence. This could be useful when designing experimental approaches to identifying and studying Acinetobacter baumannii.

ABO13250:Multidrug transporter PA4990

Similar to previous proteins, we found this protein valuable because it confers resistance to ethidium bromide, acriflavine and methyl viologen, possibly by the exporting of toxins. Therefore this protein might play a role in Acinetobacter baumannii's increased antibiotic resistance and its targeting might be clinically valuable.

ABO11887:Alkyl hydroperoxide reductase subunit F;EC=1.8.1.-

This protein uses either NADH or NADPH as reducing agent of alkyl hydroperoxides to protect the cell against DNA damage. If this protein is specific to Acinetobacter baumannii, it could be a valuable target when trying to kill this bacterium in cases where antibiotics are no longer effective by increasing the degradation of its DNA.

ABO1260:Replicative DNA helicase;EC=3.6.4.12

This protein unwinds the DNA duplex at the Escherichia coli chromosome replication fork and participates initiation and elongation during chromosome replication. Targeting this protein could potentially prevent Acinetobacter baumannii from replicating aiding in the prevention of its growth.

ABO12622:Chaperone protein ClpB

This protein is involved in the recovery and repair of the cell when it suffers from heat-induced damage. This could potentially prevent Acinetobacter baumannii from being killed off in the sterilization of medical instruments. If this is the case, it should be targeted to prevent the bacteria from spreading in medical environments.

5) Next Steps:

5a) Improve ability to pull information from Pfam website:

After hearing other project 1 groups discuss their projects, we realized that our issues with the Pfam webpages could be solved using a python package for HTML parsing. If we were to continue working on this project we would include one of these packages so that we could more reliably pull meaningful functional data from the Pfam website regardless of where it appeared on the webpage itself.

5b) Boil down functional descriptions into concise keywords to allow for easier collection of metadata:

An interesting thing we experienced in this project was a discrepancy between what kinds of data project 2 teams wanted. Partially Hydrolyzed People were fine getting protein names (RecNames and AltNames) separated by semicolons. On the other hand, RIP and HaramBaes wanted functional descriptions of the proteins. We liked Partially Hydrolyzed People's approach because we could easily return concise protein names that provided information on what the query might do. These keywords separated by semicolons could be easily collected for meaningful metadata. However, RIP and HaramBae's data, taking the form of English descriptions of proteins, would give a better sense of what exactly the protein was doing. If we were to continue working on the project we would like to do some more advanced keyword processing on the functional descriptions of proteins so that they could be read quicker and more concisely paint a picture of the function of the query sequence. Processing functional descriptions into its most crucial keywords could provide information more helpful in the collection of metadata and make it easier for users to understand the function of their query at a quick glance that doesn't require reading full English sentences. We realize that GO terms are a way of simplifying a protein's functional description but it would still be interesting to try this ourselves as a programming exercise.

5c) Quality assessment:

In every case, we took the best hit from each database and provided that to the project 2 teams. However, this approach potentially lost valuable information when the top hits for a query sequences were very similar in BitScore and E-value. We would like to add functionality so that people reading our data also got a sense of how good the data is. This would also serve useful when looking at query sequences that returned the same data. Comparing E-values and BitScores of these sequences could help users determine which query sequence provided better results, and potentially the relationship between these sequences. On top of E-values and BitScores, we would have liked to provide information on the quality of the data provided by UniProt. When it comes to the annotation of a protein sequence, UniProt gives a score as to how good the data provided for the sequence is. We had a case where two sequences mapped to two different annotations of the same protein. Each annotation was given a different score by UniProt. Providing users the quality of annotation described by UniProt would give them extra information on how good the data is.