Sequence Cluster Merging using Normalized Phylogenetic Tree Distances *

Towards partial requirement of L529

Arvind Gopu
School of Informatics
Indiana University, Bloomington IN
agopu@cs.indiana.edu

April 22, 2004

1 Introduction

In this report, we explain and illustrate a novel and simple method to merge sequence cluster fragments. The method uses normalized phylogenetic tree distances as its base. We intend to have this method as one of the various methods that will be included in a cluster merging framework that is under development. The popular multiple alignment program *Clustalw* has been used to generate sequence profiles and subsequent phylogenetic trees.

The organization of this report is as follows: Section 2 explains the basic technique involved – how the tree distance is used and how it is normalized, etc. In section 3 we briefly discuss our implementation of the method – in perl as part of our merge framework. We explain briefly our experiments in section 4. Section 5 mentions a few ideas which are in the pipeline or are possibilities for future work. We then conclude with inferences from our experiments in section 6.

2 Description of Technique

In this section, we explain the algorithm involved to generate phylogenetic trees for combined fragment cluster profiles and then how distance data from those trees are used to decide whether or not to merge two fragments.

*This is part of the author's ongoing research with Dr. Sun Kim, Asst. Professor at School of Informatics, Indiana University

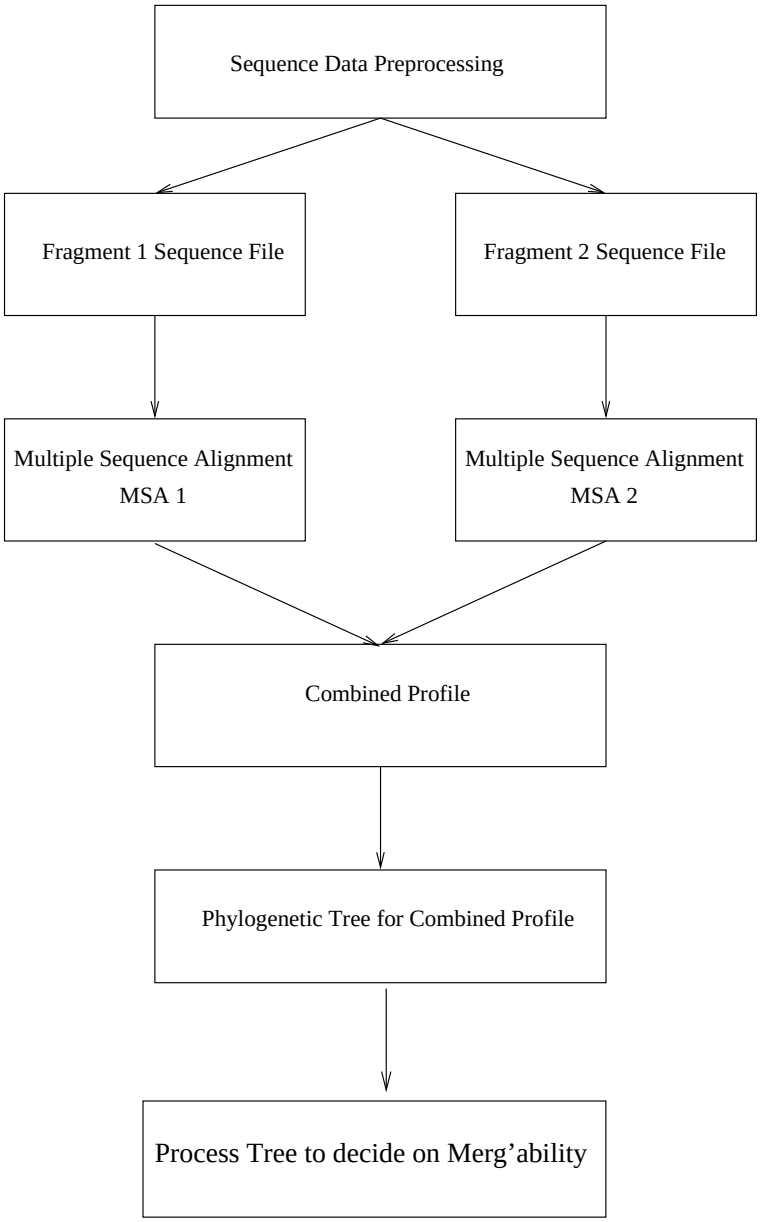


Figure 1: Simple Illustration of Merge Technique

2.1 Preparation of Data

Before any processing can be done, there are some format requirements for the data. The first step, as will be explained in the next subsection, is multiple

sequence alignment (MSA). Clustalw, which is what we have used, and most other MSA programs take in a set of sequences in fasta format to generate the alignments. So we have a script which generates sequence files for fragmented clusters generated by our clustering algorithm BAG [1].

This step might or might not be needed, depending on the type of output a clustering algorithm generates. Also it should be noted that apart from preliminary artifact experiments, we process only fragment clusters which are suggested as mergable candidates by the clustering algorithm itself. So the efficacy of our algorithm depends upto an extent on the meaningfulness of those suggestions i.e. most suggestions will be rejected if they are inherently bad ones.

2.2 Generation of Profile and Phylogenetic Tree

The generation of multiple alignment profiles leading to phylogenetic trees can be discussed with the help of figure 1. Some of the steps shown in the figure are actually applicable to other methods in our cluster merging framework; so re-use of processed data is possible. We will not discuss this any further in this report.

We start with two sequence files corresponding to two cluster fragments. Then the following steps are carried out:

1. Individual MSA profiles are generated for the two sequence files. We have used *clustalw* to do this.

2. The two individual profiles are input to a profile alignment program to produce a single combined profile. *Clustalw* actually does this type of profile alignment too and serves our purpose.

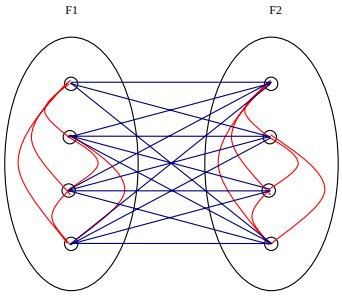
3. The combined profile generated in the previous step is then fed into a phylogenetic tree generation program. Again, we have used *clustalw* for the tree generation.
   It should be noted that our framework is flexible enough, so it should be straight forward to use other programs like the ones available in the Phylip suite to generate the trees.

2.3 Merge or Not? Decision

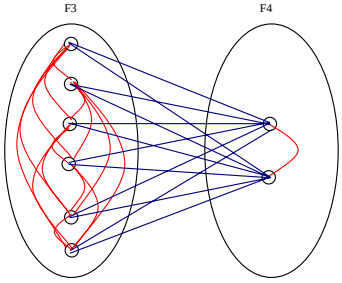
Once a phylogenetic tree has been constructed for a combined profile of two fragment clusters, we process the tree file to parse out a set of 'tree distance' values between sequences. This distance is nothing but the percentage divergence for two respective sequences as explained in [2]. Our decision on whether to merge or not is based on a very intuitive notion – and that is explained below.

If we consider two sequence clusters to be sets (in a venn diagram), then any score between sequences, like FASTA34 score or tree distance, corresponds to an edge between the members of the two sets. Edges between members of a single set are analogous to intra cluster-edges while ones between members of

two clusters are analogous to inter-cluster edges in graph theory terms. This is illustrated in figure 2.



Example of two fragment clusters with equal number of sequences.
Number of intra-cluster edges show in red = (6 + 6) = 12
Number of inter-cluster edges shown in blue = 16.



Example of two fragment clusters with unequal number of sequences.
Number of intra-cluster edges show in red = (6 + 6) = 12
Number of inter-cluster edges shown in blue = 16.

Figure 2: Fragment cluster cartoons to show need for normalization

NOTE (Important correction): In the above figure, the second sub-figure should have read as follows:
Example of two fragment clusters with unequal number of sequences.

Number of intra-cluster edges shown in red = (15 + 1) = 16
Number of inter-cluster edges shown in blue = 12

It is quite intuitive to expect intra-cluster edges to have shorter distances than inter-cluster distances. So in our case, to decide on whether two clusters need to be merged or not, intra-cluster edges do not have much influence. What we are interested is in the inter-cluster distances. The only hurdle is that though we know the fragment cluster a sequence belongs to, we do not know yet whether two fragments are sub-sets of a bigger cluster or if they are indeed two independent clusters. That is exactly what we are trying to figure out! So we have looked at the notion of inter-cluster distances in terms of "bad" distances and "good" distances. These two parameters could be made variables i.e. program parameters. Again the "good" distances between members of two clusters get abstracted out along with the intra-cluster distances. But the "bad" distances stand out and that is what we have targetted our attention on.

There is one more consideration: the number of inter-cluster edges is directly proportional to the product of number of sequences in the two individual fragment clusters. This gives raise to a need for normalization of our criterion i.e. "bad" distance. This is explained below.

Normalization:
As mentioned earlier, the number of edges across the two sets in the top fragment-set in figure 2 is directly proportional to the product of number of sequences in the two individual sets. If the difference in fragment cluster (set) size is huge, then the number of inter-edges is going to be skewed with respect to the net number of intra-cluster edges – which, as mentioned before, falls within the category of "good" distances.

To illustrate this, let us consider the following two cases shown in figure 2. We assume first up that the two fragment clusters are indeed independent of each other. If the top case is considered, then we would ideally expect $(4*4) = 16$ edges across the sets (or clusters) to be "bad" and the number of intra-cluster distance values to be $(6 + 6) = 12$. But in the lower case, though we have the same net number of sequence, the number of inter-cluster distance values (which are "bad") would be only $(6 * 2) = 12$ while the number of intra-cluster distance values would be $(15 + 1) = 16$! Obviously, we need to normalize the number of "bad" edges before making a decision on whether if it is bad enough i.e. reject merge OR if it is permissible i.e. allow merge.

3 Implementation

We have implemented the technique explained in section 2 using a combination of perl scripts and modules. Actually we have made it part of a bigger framework for cluster merging, which has been our primary area of research for the past 8

months or so. Eventually we visualize the framework to work as follows: Users who want to use BAG (or other clustering algorithms as long as they can bridge data format to our requirements) to cluster biological sequences, will be given an interactive framework in which they can choose a cluster merging technique. The technique we have worked on in the past uses statistical measures – relative entropy, etc. Then we have the tree distance method which is what this report is about. We also plan to work on another method using the concept of 'In-betweenness' in the near future.

Our framework, as it is at present, expects clustering results in a specific format and also a set of 'merge suggestions'. The format of these things is just a question of changing parse-regexs – quite straight forward. Once the data is supplied to the framework along with the choice of method for merging, it does the rest – producing MSA profiles, trees, etc as per the requirements of the respective method. Then finally it prints a file with the merge candidates. At this point another program takes over to do some further processing. Explaining the followup programs or their purpose is out of scope of this report.

4 Experiments

We have experimented our merge method with two types of data.

4.1 Data constructed from COG families

We constructed test cases based on the COG family classification. Essentially all we did was: select sets of two COG families and then construct 'good' & 'bad' test cases. The former contained sequences from just one COG family while the latter contained seqeunces from both the chosen COG families. We built a bunch of test cases like this choosing arbitrary COG families – with some considerations like size of the chosen COG families and how related they are to each other. The results of this experiment is tabulated below. The two COG families used and then various sizes of the data extracted out for each of those is listed. The expected and observed merge-decisions are listed for each row.

COG family (size)	$n(F1)$	$n(F2)$	Expected	Observed
COG0001 (35)	10	10	Good	Good
COG0005 (30)	10	10	Bad	Bad
	10	4	Good	Good
	10	4	Bad	Bad
	4	10	Good	Good
	4	10	Bad	Bad
	4	4	Good	Good
	4	4	Bad	Bad
	4	2	Good	Good
	4	2	Bad	Bad

COG family (size)	$n(F1)$	$n(F2)$	Expected	Observed
COG0160 (79)	10	10	Good	Good
COG0161 (49) 10	10		Bad	Good
	10	4	Good	Good
	10	4	Bad	*Good*
	4	2	Good	Good
	4	2	Bad	*Good*
COG0142 (74)	10	10	Good	Good
COG0183 (116)	10	10	Bad	Bad
	10	4	Good	*Bad*
	10	4	Bad	Bad
	3	3	Good	Good
	3	3	Bad	Bad
COG0380 (15)	8	7	Good	Good
COG0383 (13)	8	7	Bad	Bad
	8	2	Good	*Bad*
	8	2	Bad	Bad
	8	4	Good	Good
	8	4	Bad	Bad

In the result shown above, there are a few points to be noted.

- The method works really well with similarly sized fragment clusters.
- It works fairly well with differently sized fragments. Errors are on the safer side – false negatives.
- The method fails in the case of COG0160 and COG0161. This was almost expected since sequences from these two clusters contain the same domains as reported by a HmmPfam search [4]. We argue that the sequences from these two families have been separated into two families only because of manual efforts of the curators of the COG database.

4.2 Merge suggestions from BAG

Our tests on BAG-recommended merges has not shown any clear cut results as of yet. One main issue is that the suggestions themselves don't improve the clustering results too much – even a perfect merging scheme does not improve the Containment Index (CI) or the Fragmentation Index (FI) [3].

To summarize, we have not been able to find an ideal parameter set that can classify the method as a complete success for all kinds of data. But we have seen enough success in the limited experimentation that we have been able to do. The fact that we plan to use the method in an interactive setup makes it attractive, since the user has considerable influence on the choice of method, parameters and even possibly the merging decisions itself.

5 Misc Experiments & Future Work

There are two main points that we have been thinking about, related to our work so far in the tree distance based merge technique.

- The phylogenetic tree generated (as explained in section 2) can be visualized using readily available tools. We have already tried doing such experiments and came across some results which could be very useful if investigated more. For example, figures 3 and 4 show two trees which have been visualized using the *drawtree* program – this program is part of the *Phylip* package. The first one corresponds to a good merge and the second to a bad merge.

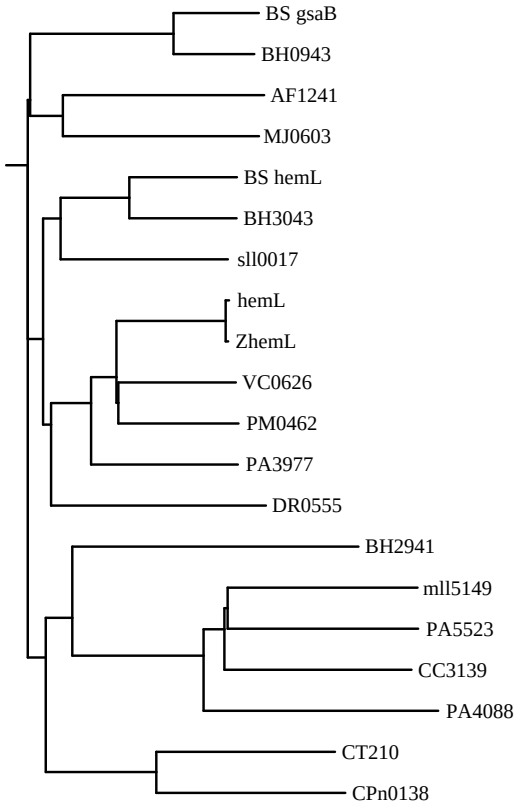


Figure 3: Example - Phylogenetic tree for a *good* merge case

It is evident that the divergence shown in the two cases is different. If a method can be devised to automate the visualization process, especially if it can fit into our cluster merging framework, then it could be very useful for the user to visualize certain doubtful merge suggestions before accepting any of the merge methods' decision.

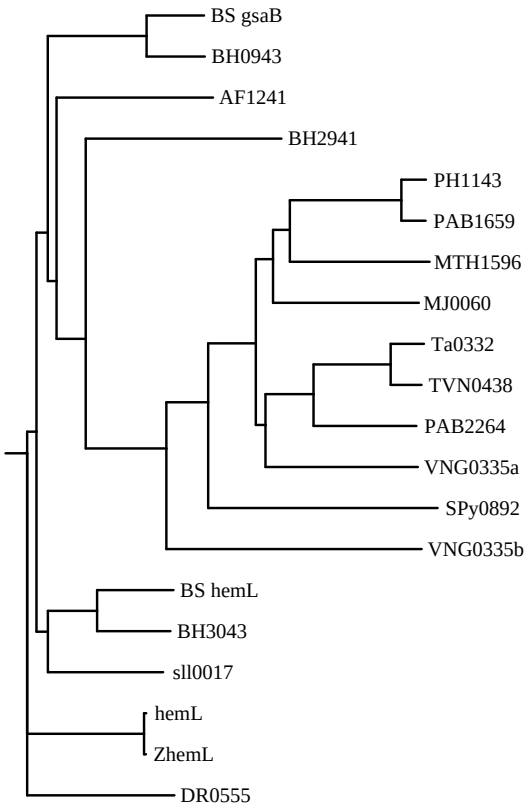


Figure 4: Example - Phylogenetic tree for a *bad* merge case

- There could be further experiments in using other tree generating tools (Example: *Phylip*) apart from *clustalw* to check if that has a big influence. For instance, the method used by *Phylip*) is very different from the one used by *clustalw* in generation of a phylogenetic tree.
- Another obvious issue is further experimentation with direct clustering

result data to check out how effective the method is.

6 Conclusion

We have described a novel and simple method to merge fragmented sequence clusters. The use of normalized phylogenetic tree distances to this effect is clearly explained. Experimental results corraborate our claims about its effectiveness. As mentioned earlier, we intend to have this as one of the multiple methods in a cluster merging framework (under development). We also plan to investigate further the use of this method as a supplement to the BAG clustering algorithm as well as some of the visualization possibilities discussed in the previous section.

References

[1] Kim S., *BAG: A Graph Theoretic Sequence Clustering Algorithm*

[2] Chenna R, Sugawara H, Koike T, Lopez R, Gibson TJ, Higgins DG, Thompson JD., *Multiple sequence alignment with the Clustal series of programs*, Nucleic Acids Res. (2003)

[3] Kim S., Gopu A., *Cluster Utility: A new metric to guide sequence clustering*, (IU Bioinformatics Research Lab - Internal Technical Report).

[4] HmmPfam Search URL *http://pfam.wustl.edu/hmmsearch.shtml*