Research Statement Draft

David H. Rich *Grant Applicant*

1 Introduction

In the realm of bioinformatics, the task of sequence alignment is a critical component of many research goals, such as gene annotation or homology search. HMMER and MMseqs are two of the most commonly used sequence alignment suites designed for such tasks. These tools both excel in different aspects of this: MMseqs generally performs the sequence alignment much faster, while HMMER is generally much more accurate.

Both of these tools function in much the same way. First, they both take in the entire query and target sequence/model. Then, they pass them through a series of increasingly complex and stringent scoring algorithms. They pass only those query/target sections (windows) which reach the scoring threshold to the next filter. Many of these filtering algorithms used by the two suites are very similar. HMMER uses a Multiple-Segment Viterbi pre-filter (MSV), which feeds into a gapped Viterbi filter, then finally a Forward-Backward filter. Meanwhile, MMseqs uses double k-mer matching pre-filter, which feeds into an ungapped Viterbi filter, then finally a gapped Viterbi filter. The primary differentiators between the two pipelines is the starting MMseqs' double k-mer matching pre-filter and HMMER's final Forward-Backward filter. Even when using MMseqs "sensitive" settings, which filters less aggressively in the pre-filter stage, HMMER still tends to out-perform MMseqs in trials. This suggests that Forward-Backward is contributing greatly to HMMER's accuracy. However, the downside of this is that Forward-Backward is also a substantial contributor to the runtime.

2 Methods

Because Forward-Backward is such a costly yet important element of the pipeline, we propose to design a heuristic, pruning version of the Forward-Backward algorithm, (currently being called Cloud Search). In the current implementation of HMMER's Forward-Backward, the previous Viterbi filter passes a window range of the query/target, with a query subsequence length Q and target subsequence length of T. This window can be represented by a dynamic programming matrix. Each cell in the window is computed by summing the probabilities of all possible paths which end in the given cell, for all $Q * T$ cells. However, many of these cells have extremely low probabilities, which have little to no chance of effecting the outcome of the final alignment.

Alternatively for Cloud Search, we start from a different position. First, we use the "backtrace" of Viterbi, a method by which we can recover the exact maximal alignment from the Viterbi filter. This means that rather than starting the Forward algorithm in the corner of an arbitrary window, we can start at the beginning of the maximal alignment.

From this point, we will compute the cells anti-diagonally from the starting node. At each step, we will compute the maximum current sum, M. For some tuning parameter k, $0 < k < 1$, we will check the scores along the anti-diagonal. Any cell which falls below the given score, kM, will be pruned and paths passing through this cell shall no longer be computed. When all pruning has been performed the forward direction, we will do the same in the backward direction. These pruning bounds will give us a Forward-Backward of the entire "high-probability cloud" around the maximal alignment. This means rather than computing the entire square window, with $O(N^2)$ time complexity, we can reduce our search space to a small, roughly diagonal subset of this, which should be $\approx O(N)$ in most cases. This should drastically reduce the number of cells to be computed and thus the runtime of the algorithm.

3 Assessment

We will benchmark the Cloud Search by replacing the current Forward-Backward implementation in the HMMER pipeline. Then the two differing pipelines will used to search against a subset database from Swiss-Prot, a collection of known annotated protein sequences. These will be used as the true targets. The sequences will then be shuffled and added to the target database as false targets (this will maintain the amino acid background frequency while destroying it underlying structure). Queries will be generated by taking proteins from the true targets and mutating them through a series of random insertions and deletions. This will be done until they have between 30 and 70 percent identity.

True-positives will be defined as when a protein query is matched with its true target, and false-positives will be when a protein query is matched with any false target. All other cases will be disregarded. We will then compare the results by calculating the AUC (area under the curve) up to the FFP (first false positive). This will count the percentage of true-positives found by the search before the first false-positive is found, sorted according to its p-score. In addition, each algorithm will be compared according to their wall clock time. We will repeat the Cloud Search under different choices of k in order to the compare and optimize parameter values.

4 Intellectual Merit

This Cloud Search could give a substantial speed boost to the current HMMER pipeline at minimal cost to accuracy. Alternatively, its addition to the end of MMseqs pipeline may give MMseqs an increase in accuracy while minimizing the cost.

If this heuristic algorithm proves successful, it will mean more accurate genomic searches can be implemented in less time. This will allow for greater throughput, which can speed up genome-related research.

5 Broader Impacts

This research could result in a number of potential benefits to society, through the acceleration of research in a diverse array of fields. From pathology, to drug and gene therapy, to evolutionary biology, many scientific fields potentially depend on homology search as part of their research. This work can in turn expedite improvements in public health and our understanding of the biological world at large.