Comparing Protein Sequences using String Kernels

By: Munir Nur & Zaid Al Rakabi

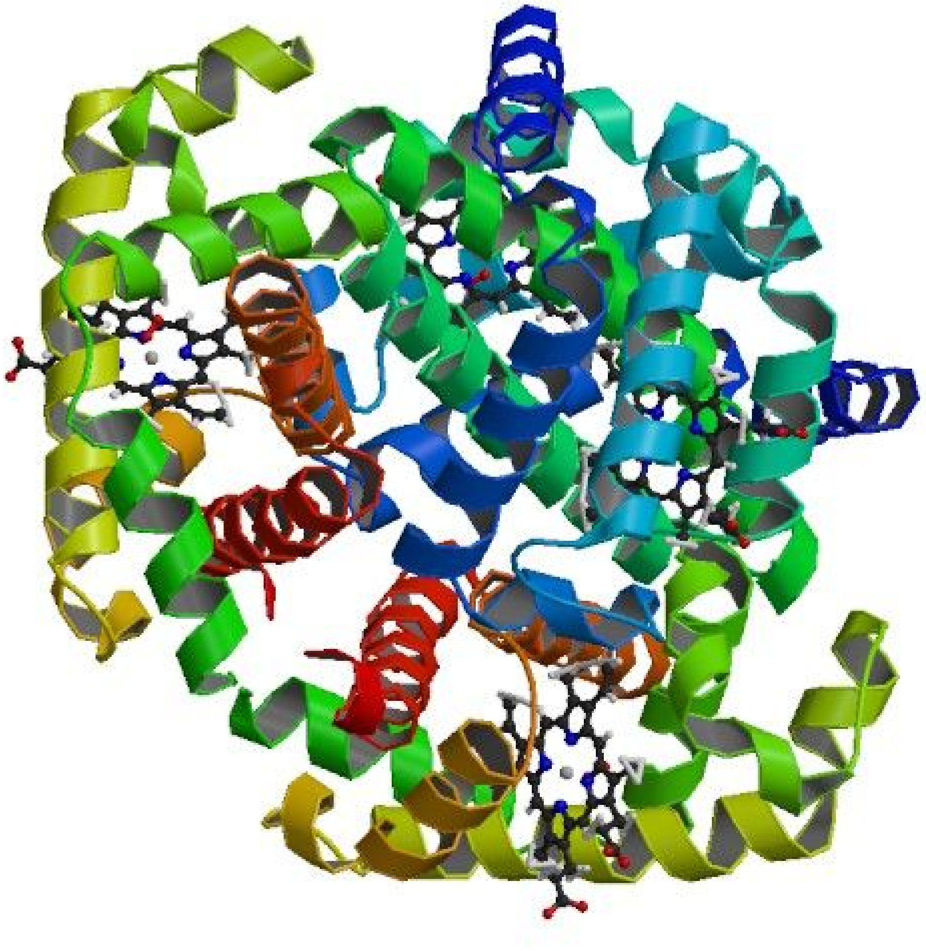


Image Source: https://www.rcsb.org/structure/6BB5

Introduction:

Proteins have remained mysterious for as long as they've been discovered. The intricate mannerisms in which they fold, both locally and globally, continue to baffle scientists at a molecular level. The unique folds that a protein creates are critical in analyzing the structural to functional relation within it. This relation allows us to predict applications of other, similarly structured proteins, synthesize replacement proteins, and so much more. Although these complex structures have seemingly indistinguishable folding patterns, amino acid sequences that make up proteins are key in allowing us to predict structure. Once information about protein structure is attained, predicting function becomes somewhat easier.

Analyzing a single protein's amino acid sequence alone isn't completely satisfactory if we want to determine the structure. We must compare this sequence to sequences of which we already know the structure and function. This comparison is vital to infer the original sequence structure, function, and possibly it's identity.

Although the need for accurate sequence comparison is necessary, it's increasingly difficult to create a solid consensus of the method between Computational biologists. The comparison step between strings of amino acid sequences is extremely subjective, which results in a natural plethora of methods that exist. In the broadest sense, these methods can be divided into two categories: alignment and alignment-free methods.

Alignment algorithms align two protein sequences and compare them by generating a score while taking into account penalties for differences or gaps. Global alignment of protein sequences aligns the proteins from end to end assuming the proteins are homologous. Then using a dynamic programming algorithm like the one implemented by Needleman and Wunsch determines the similarity of the two sequences by applying gap penalties and matching scores [4,5]. Gap penalties can be determined in various ways by setting a constant value or more effectively by using a scoring matrix like PAM that groups based on similar side chain groups of proteins. By applying gap penalties based on the group of proteins this allows for a more accurate score. For example if a hydrophobic amino acid in a sequence differed from another hydrophobic amino acid the value should be positive because that is more common than if there was a hydrophilic amino acid in place. Local alignment unlike global alignment does not try to align the entire sequence, rather it finds smaller regions of high similarity. Local alignment algorithms like that of Smith-Waterman utilize a scoring system based on matches and mismatches and also use a scoring matrix system to identify high scoring regions [3,5]. These high scoring regions are what determine the similarity between the two sequences. Both Local and global alignments have benefits when comparing protein sequences in that they guarantee optimal alignments of the sequences being tested. If the sequences that are being compared have very different amino acids and one wanted to identify regions that were similar in both sequences a more local approach would be better. However if one wanted to identify the similarity between two sequences that have a higher chance (>30%) of being similar a global approach would be better [9].

In addition to local and global alignment, heuristic alignment methods such as BLAST introduced by Altschul et al.[1] are some of the most widely used algorithms for sequence comparisons due to there time efficiency in providing approximate local alignments through rapid sequence comparisons [1]. While BLAST cannot guarantee optimal alignments of sequences it makes up for that by reducing the time of traditional alignment algorithms by an order of magnitude. BLAST also utilizes a scoring matrix to provide an accurate score of the sequence. BLAST generates this score after searching through the two sequences to find a local match and extends that match to generate an alignment with a high scoring pair above a statistical threshold E-value[5].

Computational biologists utilize alignment-free algorithms to bypass various pitfalls of alignment-based algorithms. Rather than looking for similarities between sequences, and adjusting (and applying some representative gap penalty) multiple times for a more leveled comparison, alignment-free algorithms rely on other methods. A popular alignment-free method focuses on word frequencies, inspecting many subsequences and utilizing vector spaces to interpret distances [8]. These methods focus on k-mers (subsequences, or words of length k), and compute statistical frequency distributions to determine the similarity between two sequences. Some of these algorithms focus on exact word matches, while others allow for some variation in less important amino acid positions (i.e. spaced seed methods) [8]. The similarity is termed the distance between the two sequences, which is computationally defined differently throughout different algorithms, based on frequency distributions of the k-mers [10]. The computational definitions vary, depending on if scoring is limited to exact k-mer matches, or if there are pre-introduced scoring schemes.

Another category of alignment-free comparison primarily focuses on the use of a string kernel, which is also based on k-mers. The main difference between string kernel methods and those described above is the mathematical function of a string kernel. A string kernel operates on two strings that often contain amino acid sequences. The kernel is most simplistically defined as the dot product of statistical similarity measurements between the sequences [10]. In protein sequence analysis, a string kernel is often paired with a substitution matrix, to create a weighted string kernel that's more efficient in determining similarity [10]. The reasoning behind this is the likelihood that certain amino acids favor being replaced by other amino acids with similar chemical or structural qualities.

In this paper, we will use an alignment-free method for string kernel comparison. This method differs from other alignment-free methods because it combines the local kernel alignment method approach that uses a scoring matrix to sum up all the contributions of the local string kernels [2,10,11] and the string kernel approach that takes the sum of all possible k-mers in the sequence [2,10,12]. Our method utilizes the BLOSUM62 scoring matrix for accurate comparisons and generates a standardized metric for similarity between two sequences to better determine the homology of the protein sequences.

Methods:

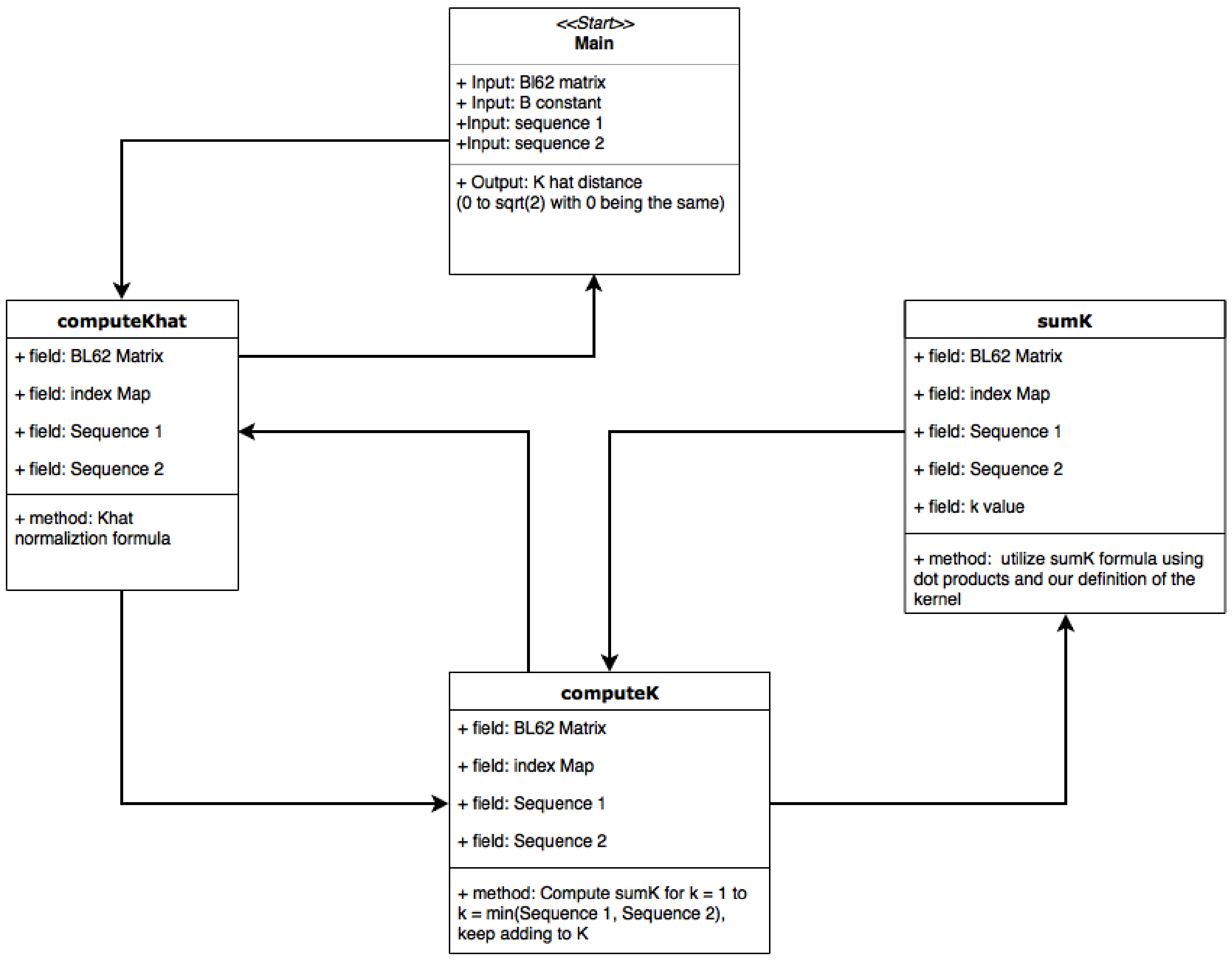


Figure 0: Sequence kernel program: kernel.cpp (Seq Kernel)

The sequence kernel program (Fig. 0) that we implemented is based off of an alignment free approach introduced by Stephen Smale et al. [13] and a program SeqKernel implemented by Nojoomi and Koehl [10, 2]. We aim to describe our method and how it satisfies the triangular inequality to define a metric for sequence similarity.

Fig. 0 depicts the outline for our sequence kernel program Seq Kernel. This program takes as input sequences $S = s1, s2, ..., sn$, and $T = t1, t2, ...tm$ and computes the score by comparing individual elements in the sequences using the formula in Fig. 1. The formula in Fig.

1 computes the score of similarity of the individual elements in the sequences using the BLOSUM62 matrix and the parameter $\beta = 0.01$. The BLOSUM62 matrix is used because the matrix is constructed by taking into account the chemical properties of the amino acids and the tolerance of an amino acid to accept another amino acid in its place [14].

$$K^1(s,t) = BL62(s,t)^\beta$$

Figure 1: K^1 formula for computing score of length 1 kmer

Knowing how to compute the value of an individual element we can expand our kernel length to two comparing pairs of letters with $K^2(s_1 s_2, t_1 t_2) = K^1(s_1, t_1) * K^1(s_2, t_2)$. This approach can be generalized for greater length k-mers with value k indicating the length (Fig. 2). The k-mers are a continuous length of the sequences (gaps are not being considered for the score) with a score of $SumK_k(s,t)$. $SumK_k(s,t)$ uses this property of determining the kernel score and can compute the total score for any length k.

$$SumK_k(s,t) = \prod_{i=0}^{k-1} k^1(S1i, S2i)$$

Figure 2: SumK formula for computing K1 values of the kth kernel

With a score found for a specific k-mer we then continue to calculate the total score for all possible k with $k \leq min\ length(S,T)$ (Fig. 3).

$$K(s,t) = \sum_{k=1}^{L} SumK_k(s,t)$$

Figure 3: K formula for computing total K value with L=min(s,t)

With the total score of S and T we then define a score $\hat{K}(s,t)$ which takes the value of the total score K(s,t) and divides by $\sqrt{K(s,s) * K(t,t)}$ (Fig. 4) this is done to normalize the value of the sum from the two sequences.

$$\hat{K} = \frac{K(s1,s2)}{\sqrt{K(s1,s1) * K(s2,s2)}}$$

Figure 4: K hat formula for normalized score of the K value

Having the normalized score is not enough, we still need a metric for similarity that satisfies the triangular inequality. To get a metric we use the distance formula defined in (Fig. 5) to get a value $dist(\hat{K})$ which describes the similarity of two sequences S, and T with the value zero meaning they are the same and $\sqrt{(2)}$ meaning they are different.

$$dist_k(s,t) = \sqrt{2 * (1 - \hat{K}(s,t))}$$

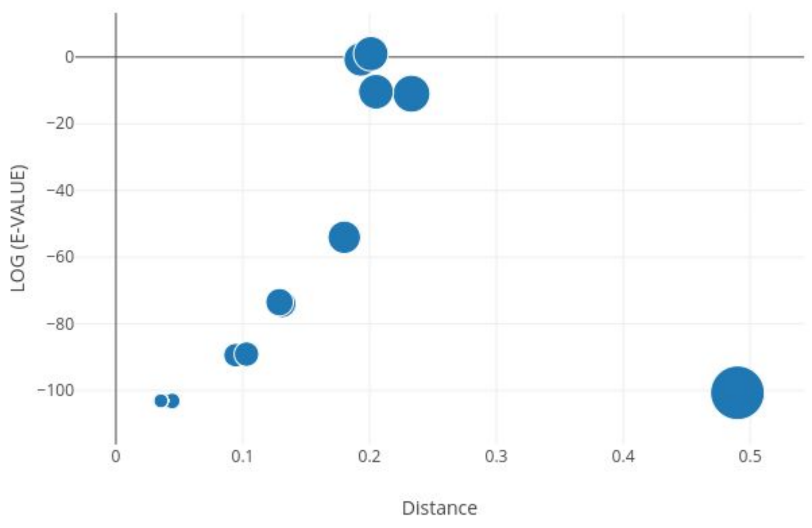
Figure 5: distance formula for making $\hat{K}$ into a metric

Data and Analysis:

Hemoglobin is present in almost all living organisms. It is a protein that functions as an oxygen transporter in the blood and is key to regulating blood pressure [15]. Since the sequence of the protein relates to the structure of the protein, we will attempt to show and analyze the relationship between human Oxyhemoglobin and other variations of the hemoglobin protein with known structure found both in humans and other organisms. By using our Sequence Kernel program and comparing the first chain of the hemoglobin protein sequences we will be able to detect if the other hemoglobin proteins are homologs of the human Oxyhemoglobin. We will also compare our findings to widely used alignment scoring methods both that utilize and don't utilize heuristics to determine our accuracy and efficiency.

Figure 6: LOG (BLAST E-VALUE) VS. Seq Kernel Distance

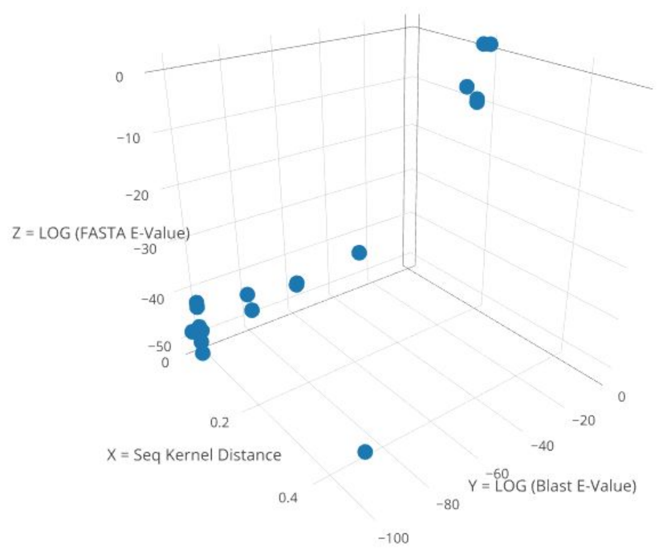
Circle Size: The circle size is based on the value of the distance.



The most prominent difference between Seq Kernel and the heuristic category of alignment comparison is that the latter returns an E-value, indicating the expected number of hits (i.e. sequences) that we can expect to see by random chance, given a database of fixed size. Although database sizes can vary, one can reasonably estimate statistical similarity with extremely low E-values the typical threshold for similarity being $E \leq 1 \times 10^{-5}$ [5]. We plotted our Seq Kernel distance results against the BLAST E-values and log (BLAST E-values) for varying Hemoglobin sequences among different species. This type of semi-log plot maintains the relative positions of E-values to each other but scales them in a way where extremities are much clearer to see, and the non-linear correlations are exemplified more clearly. In Figure 6, our findings suggest a strong parallel between the different algorithmic results, where low Seq Kernel distances almost always correspond to low E-values/log (E-values).

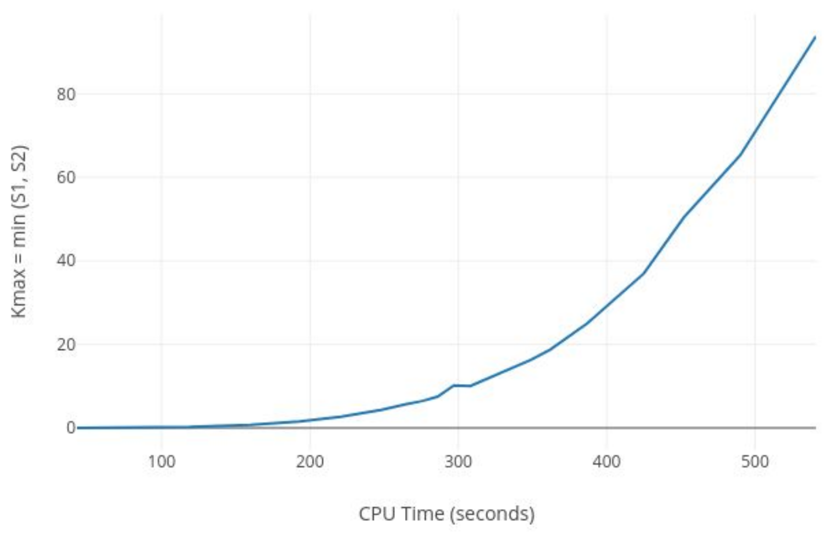
Although BLAST and Seq Kernel produced relatively similar results, numerous outliers were present in the data. Figure 6 revealed how much more sensitive Seq Kernel is to length differences and is seen in the case of our base Hemoglobin sequence (Human Oxyhemoglobin, 6BB5) versus Deoxy Recombinant Hemoglobin (1ABW). The latter was approximately twice the size of our base sequence, so the extra noise caused for inaccuracy of homology detection. Besides this inaccuracy, our results revealed a cluster of sequences that only Seq Kernel was able to link relation to our base. Among this cluster included worm, Hagfish, and Mycobacterium hemoglobin sequences. BLAST was essentially unable to find any relation, where Seq Kernel found a significant one for these particular sequences (Distance ~ 0.2). These are all varying forms of the hemoglobin protein among different organisms; so for the most part, their sequences should certainly not be random to each other.

Figure 7: LOG (Blast E-VALUE) vs. Seq Kernel Distance vs. LOG (FASTA E-Value)



After further comparing Seq Kernel with FASTA, a similar heuristic method (also based on local alignment strategies to generate an E-value), the FASTA data was essentially consistent with BLAST. Figure 7 displays our resulting E-values from both heuristic methods. The same cluster existed where the E-values signified little to no relation, whereas Seq Kernel produced an accurate distance [Fig. 7]. It's also important to note that both widely used heuristic methods don't perform an accurate local alignment, and essentially try to expand k-mers within sequences to find word matches. This results in a computationally fast, yet non-optimal comparison strategy. In addition to this analytical shortcoming, heuristic methods didn't give us consistent E-values with duplicate tests. They resulted in fluctuations of about 1×10^{-5} each rerun, while the deterministic Seq Kernel produced the same distance every rerun. We graphed the E-value of the first comparison trial but ensured the fluctuation wasn't largely different from other trials.

Figure 8: CPU Time with Kmax = min (S1, S2)



As Seq Kernel is an analytical comparison algorithm, rather than a heuristic database search, a brief note about the time complexity is appropriate. With n being the length of sequence 1, and m being the length of sequence 2, the time complexity of Seq Kernel is $O(nmK_{max})$, where K_{max} is by default limited to $min(n,\ m)$. This can be problematic, as sequences of thousands of amino acids can start taking minutes to receive a distance. With our dataset of hemoglobin sequences, lengths of any individual sequence is at most ~250, which would result in an instantaneous distance. It's worthy to note that at this level of sequence length, Seq Kernel proved to give instantaneous distances, and proved to be just as fast (if not faster) as the heuristic approaches when comparing 2 individual sequences.

Figure 9: SMITH WATERMAN SCORE VS. Seq Kernel Distance

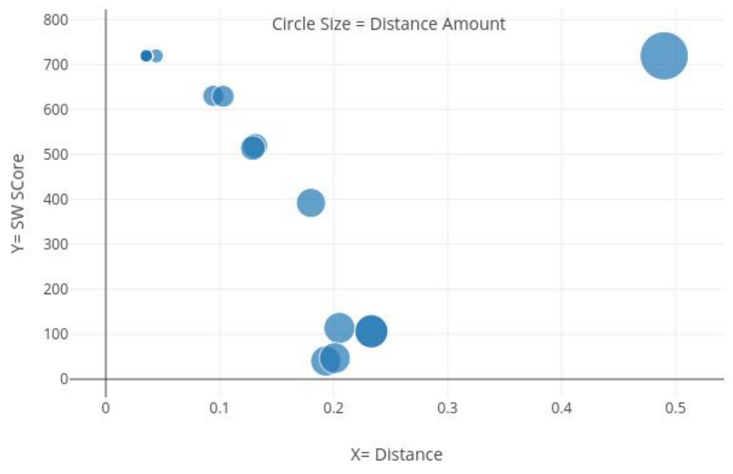
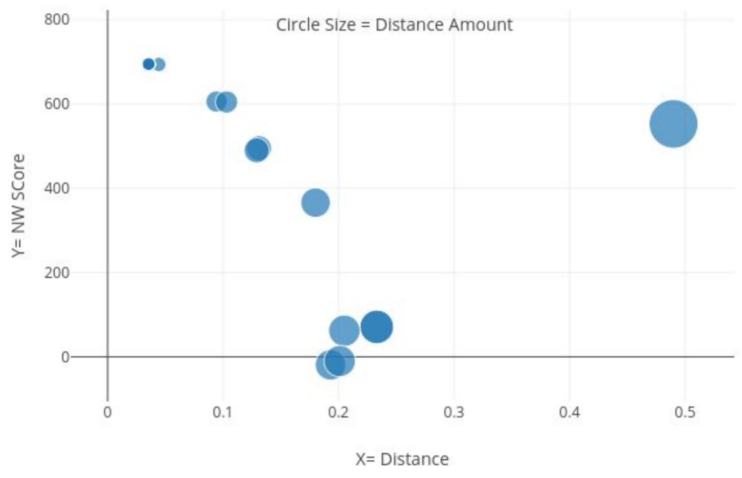


Figure 10: NEEDLEMAN WUNSCH SCORE VS. Seq Kernel Distance



When comparing the Seq Kernel distances to non heuristic algorithms both local alignment, Smith and Waterman [Fig. 9], and global alignment, Needleman and Wunsch [Fig. 10], we witnessed similar results. Both the local and global alignment methods showed a strong correlation between low distances and high max alignment scores despite Needleman and Wunsch applying negative scores to sequences. However, there remained outliers with high distance and high alignment scores because the alignment-free Seq Kernel relies on sequences with similar lengths. When the lengths of the sequences that are being compared varied greatly the distance score increases. The alignment methods don't experience this effect because they take gaps of the sequences into account to produce the max alignment score. However, Seq kernel was more accurate in detecting the sequence similarity between sequences that didn't align well but were similar because they are ortholog hemoglobin proteins from the human hemoglobin protein being compared to. Seq Kernel's cluster accuracy is still evident when compared to standard alignment algorithms that guarantee optimal alignments. As a result, this further emphasizes the improvements that Seq Kernel has in detecting homology when compared to more widely used alignment algorithms. Seq Kernel is also able to process sequences at a similar time as Smith-Waterman that has a time complexity of $O(m^2n)$ but required more time than Needleman-Wunsch that has a time complexity of $O(mn)$ [3,4].

Discussion:

When comparing similar sequences, significant E-values between protein sequence pairs can be hard to differentiate, due to their length-dependency nature and random fluctuation. This is where Seq Kernel may excel, building upon its definition as a deterministic metric. Specific distances between protein sequences allow us to quantitatively compare two sequences, which is extremely powerful when distances are very close. Although our Hemoglobin dataset contained a limited number of protein sequences and only compared single chains from each sequence, Seq Kernel can definitely be applied to much larger datasets to help identify specific clusters. Seq Kernel's distance metric can be largely beneficial to classify sequences into different groups within a huge "supertype" group of proteins [13]. This can be accomplished by using distance information for cluster tree construction to provide more thorough analysis [7]. This application of Seq Kernel is proven possible (and may be greatly effective), but is extremely naive and not fully understood.

Despite the improvements that Seq Kernel provides with more accurate clustering of protein sequences and a consistent metric that can accurately measure sequence similarity, there still remains more work that must be done before we can say that our algorithm successfully computes whether two sequences are homologous, both accurately and efficiently.

In order to conclude accurately if any two sequences are homologous, we have to limit the effect that sequence length has on the distance metric. As witnessed in our trials of hemoglobin sequences comparing heuristic methods in Fig. 6 and Fig. 7, and comparing conventional alignment methods in Fig. 9 and Fig. 10, the larger length difference between the human Oxyhemoglobin tested with the Deoxy Recombinant Hemoglobin led to a higher distance score. Other scoring algorithms were able to recognize that the two sequences were in fact similar. It may be the case that two sequences with varying lengths are different, however, that should not drastically affect the results of our metric – especially if the two proteins are homologous. A possible solution might be to limit the length of the longer sequence so that when SumK is calculated, only the part of the sequence that is similar in length would be computed.

We would need to make sure that we choose a partition of the length that accurately yields the best distance and ensure that the metric is maintained. Another solution would be to implement a proper weighting scheme like the Mean weighting scheme introduced by Najoomi and Koehl [2] that would multiply a varied coefficient $w(k)$ to the total value of each string kernel K , and takes into account the length of the sequence, to help fully eliminate the impact of the sequence length.

To efficiently conclude that two sequences are homologous, we have to implement a method that can minimize the total run time of Seq Kernel. An easy way to improve runtime would be to make K_{max} , the longest length k-mer, be significantly less than the current $K_{max} = min(s, t)$. Finding an appropriate cut off for the K_{max} value would require additional studies to determine how limiting K_{max} affects the total distance and the time while still being able to accurately determine similarity. Making our calculation methods recursive could also reduce our time significantly. A potential recursive algorithm to explore in reducing the total time would be Merge Sort because splitting the sequences in half and recursively calculating the total K value would require fewer checks over the entire sequence, and also have a similar effect that limiting the K_{max} value would have on time efficiency.

In addition to working to improve the optimality and accuracy of Seq Kernel we also want to make more progress on other applications to the sequencing algorithm. Theoretically, the method used in Seq Kernel to generate a metric of string similarity can be applicable to DNA and RNA sequences. A significant drawback is the rise of redundancy and increased margin for error, which can occur when we go backward in the central dogma of molecular biology (DNA $\rightarrow$ RNA $\rightarrow$ Protein). Besides this, sequence complexity is the biggest reason we chose to compare protein sequences rather than DNA or RNA sequences. 20 letters are much better than 4 in determining detailed substitution matrices, i.e. BLOSUM62. However, Seq Kernel can be applicable to DNA sequence comparison when protein comparison gives you too similar of a result. For example, comparison of two human protein sequences will likely be close to identical, while DNA sequence comparison may prove to show bigger distances, and be useful to indicate specific variations. In Seq Kernel's case, its sensitivity to length must be minimized in order to better compare DNA sequences in order to dodge negative effects from increased redundancy, interfering introns, and increased gaps.

Conclusion:

Overall we are far from done in solving the problem of understanding protein structure with the protein sequence. This is a step in the right direction. Seq Kernel aims to take away the dependence on aligning sequences to better analyze the relationship between the sequences with a clear metric that stems from more than just an exact alignment. Seq Kernel is just the beginning of how we look at and understand proteins. With more research and improvements to the algorithm, we hope to be able to accurately and efficiently detect homology and uncover relationships no only in proteins but also in sequences of DNA and RNA.

References:

1. Altschul, S. “Basic Local Alignment Search Tool.” *Journal of Molecular Biology*, vol. 215, no. 3, 1990, pp. 403–410., doi:10.1006/jmbi.1990.9999.
2. Nojoomi, Saghi, and Patrice Koehl. “A Weighted String Kernel for Protein Fold Recognition.” BMC Bioinformatics, vol. 18, no. 1, 2017, doi:10.1186/s12859-017-1795-5.
3. Smith, T.f., and M.s. Waterman. “Identification of Common Molecular Subsequences.” Journal of Molecular Biology, vol. 147, no. 1, 1981, pp. 195–197., doi:10.1016/0022-2836(81)90087-5.
4. Needleman, Saul B., and Christian D. Wunsch. “A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins.” Journal of Molecular Biology, vol. 48, no. 3, 1970, pp. 443–453., doi:10.1016/0022-2836(70)90057-4.
5. Choudhuri, Supratim. “Sequence Alignment and Similarity Searching in Genomic Databases.” Bioinformatics for Beginners, 2014, pp. 133–155., doi:10.1016/b978-0-12-410471-6.00006-2.
6. Pearson, W. R., & Sierk, M. L. (2005). The limits of protein sequence comparison? *Current Opinion in Structural Biology*, *15*(3), 254–260. http://doi.org/10.1016/j.sbi.2005.05.005
7. Zielezinski, Andrzej, et al. “Alignment-Free Sequence Comparison: Benefits, Applications, and Tools.” Genome Biology, vol. 18, no. 1, 2017, doi:10.1186/s13059-017-1319-7.
8. Susana Vinga, Jonas Almeida; Alignment-free sequence comparison—a review, Bioinformatics, Volume 19, Issue 4, 1 March 2003, Pages 513–523.
9. Rost B. Twilight zone of protein sequence alignments. Protein Eng. 1999;12:85–94.
10. Nojoomi, Saghi, and Patrice Koehl. “String Kernels for Protein Sequence Comparisons: Improved Fold Recognition.” BMC Bioinformatics, vol. 18, no. 1, 2017, doi:10.1186/s12859-017-1560-9.
11. Saigo H, Vert JP, Ueda N, Akutsu T. Protein homology detection using string alignment kernels. Bioinforma. 2004;20:1682–9.
12. G. Rätsch, S. Sonnenburg, B. Schölkopf; RASE: recognition of alternatively spliced exons in C.elegans, Bioinformatics, Volume 21, Issue suppl_1, 1 June 2005, Pages i369–i377.
13. W.-J. Shen, H.-S. Wong, Q.-W. Xiao, X. Guo, and S. Smale. Towards a mathematical foundation of immunology and amino acid chains. 2012.
14. Henikoff S, Henikoff J. Amino Acid Substitution Matrices from Protein Blocks. Proc Natl Acad Sci (USA). 1992;89:10915–9.
15. Goodsell David, and Shuchismita Dutta. “Hemoglobin.” *RCSB Protein Data Bank*, May 2003, doi:10.2210/rcsb_pdb/mom_2003_5.