Data-centric Approaches for better Multiple Sequence Alignment

by

Kuang Mengmeng 匡盟盟

Supervised by

Dr. Hing-fung Ting

June 2020

Abstract of thesis entitled

"Data-centric Approaches for better Multiple Sequence Alignment"

Submitted by

Kuang Mengmeng 匡盟盟

for the degree of Master of Philosophy in Computer Science
at The University of Hong Kong
in June 2020

In this thesis, we investigated the use of the data-centric approach to tackle the Multiple Sequence Alignment (MSA) construction problems. Unlike the algorithm-centric approach, which reduces the construction problem to a combinatorial optimization problem based on an abstract mathematical model, the data-centric approach uses models trained from existing data to guide the construction.

In our first study, we identified a simple classifier to help us choose the best alignment tool. Then to correct the original alignment error, we added a post-processing process, which is a region-centric realignment process. At the same time, we performed a classifier for different families to adopt the appropriate realignment strategy. In our second study, we delved deeper into how to add deep-learning methods to the underlying steps of the progressive alignment method. To improve the accuracy of the progressive alignment method, we first determined the best promotion part and then trained a decision-making model for that part to guide the MSA construction process.

Accordingly, we released two complete new MSA tools based on the two studies: MLProbs in the first study and DLPAlign in the second. We compared them with about 10 other popular MSA tools against several commonly used empirical benchmarks. The results showed that these two tools improved the accuracy of MSA to a certain extent on all tests. Furthermore, when we tested them on low-similarity protein families, our methods had unexpectedly good results. MLProbs resulted in a 2.9% TC-score improvement on families with PID $\leq$ 50%, while DLPAlign achieve 2.8% TC-score growth on families with PID $\leq$ 30%. Moreover, these two new MSA methods can obtain good results in real-life applications.

Data-centric Approaches for better Multiple Sequence Alignment

by

Kuang Mengmeng 匡盟盟

B.Eng. *Harbin Institute of Technology*

Supervised by

Dr. Hing-fung Ting

A thesis submitted in partial fulfillment of the requirements for the Degree of Master of Philosophy in Computer Science at The University of Hong Kong.

June 2020

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed .....

Kuang Mengmeng 匡盟盟

Acknowledgements

The two years of M.Phil. study experience at the University of Hong Kong is significant to me. These two years have enabled me, from an undergraduate student who only knows about studying from books, to step by step through independent learner, programmer, and researcher.

First of all, I would like to thank my supervisor, Dr. Hing-fung Ting, who has taught me tirelessly over the past two years, who step by step made me from knowing nothing about the field of Multiple Sequence Alignment to obtain today's research results. For the past two years, Dr. Ting has taken the time to guide my research every day. I have to say that he is an intelligent man, and I admire his attitude as a supervisor. He often taught me to do things rigorously, meticulously, and conscientiously, and not to let go of any details. In my research, he encouraged me to look at every piece of data carefully. When I wrote a paper, I was also guided by every sentence and every paragraph. He not only encouraged me to go deeper into our research fields step by step but also encouraged and urged me to read other papers and research projects related to our research field.

I would also like to thank the University of Hong Kong and Bioinformatics Algorithms Laboratory, which I worked at for providing the necessary support to my studies and research work in the past two years, which allowed me to learn and research in-depth.

Dr. Yong Zhang is an amiable and patient teacher and friend. I want to thank his discussion of the modification of our projects and the participation of my papers.

Han Zhaofeng who is a Ph.D. candidate in Civil at HKU and Gao Lufei from the Open University of Hong Kong have been my best friends in Hong Kong for two years. They have been taking me to integrate into Hong Kong and solving problems in life quickly.

Finally, I want to thank all my friends who have helped me mentally and academically in the past two years. Thank you for your care and support.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN Convolutional neural network, page 40

FFT fast Fourier Transform, page 4

GRU Gated Recurrent Unit networks, page 40

HMM hidden Markov model, page 4

LSTM Long Short Term Memory networks, page 40

MSA Multiple Sequence Alignment, page 1

PDB Protein Data Bank, page 7

PID percentage identity, page 5

RF Robinson-Foulds, page 30

RMS Root Mean Square, page 35

RNN recurrent neural network, page 40

RRR realign reliable regions, page 23

RUR realign unreliable regions, page 23

SCOP Structural Classification of Proteins, page 7

SP sum-of-pairs, page 7

TC total-column, page 7

UPGMA unweighted pair group method with arithmetic mean, page 35

WPGMA weighted pair group method with arithmetic mean, page 35

Chapter 1

Introduction

1.1 Background

A Multiple Sequence Alignment (MSA) of a family of protein sequences is a table, constructed by putting each sequence in a distinct row of the table, with spaces appropriately inserted. The MSA construction problem is to construct an MSA so that among the sequences in the table, homologous residues originating from a common ancestral residue are aligned in the same column of the table

MSA construction is common in many biological analyses and post-genomic research. Biologists sometimes need to construct MSAs for hundreds of sequences, each with hundreds or more residues. To help handle these daunting and tedious tasks, there has been considerable research into automating the construction process. Since the early 80s, the problem has been tackled using the algorithm-centric approach, in which algorithms are designed to solve the combinatorial optimization problem, in which every possible column of residues is associated with a column score and the objective is to find the alignment column with the largest total. Many interesting and sophisticated mathematical and algorithmic techniques have been applied to solve the MSA problem (e.g., combinatorics and graph theoretic techniques [1], genetic algorithms [2], simulated annealing [3], fast Fourier transform [4], the constraint based method [5], the divide-and-conquer method [6], iterative refinement [7], Gibbs sampling [8], consensus [9], homology extension [10] and the progressive method [11]).

We now have useful MSA software tools that can construct good alignments for protein families with high similarity. For those with low similarity, however, no existing tools can consistently construct satisfactory alignments, and for them, biologists usually need some external information, such as the 3D crystal structure of the sequences, to determine their correct alignment. It has long been a challenge for the research community to develop new tools that can construct better MSAs. Even small improvements can have a significant impact because positioning even a few more critical residues correctly in the

alignment can save biologists a lot of time and effort by allowing them to focus quickly on the correct regions for downstream analysis.

After decades of research, almost all the mathematical and algorithmic techniques that are applicable to the MSA problem have been exhausted, and in the past few years, no fundamentally new techniques have been proposed. To strive for a breakthrough, we noted that the algorithm-centric approach the MSA research community has been using so far is not suitable for tackling the MSA problem because the problem is data-centric in nature. An MSA is basically a statement of homology, identifying various sets of residues in a protein family so that all residues in a set are homologous and evolved from the same ancestor residue after a long sequence of mutation events spanning thousands or even millions of years. The algorithm-centric approach attempts to capture these events using abstract models, from which feasible computational problems are formulated. Therefore, the models have to be simple, but then they are not general or powerful enough. For example, one popular abstract model is the substitution matrix model, which gives a score to each of 190 possible pairs of the 20 amino acids, with the score given to a pair estimating the probability of the pair having the same ancestor. Obviously, these 190 scores are far from enough to capture the long evolutionary history of tens of thousands of protein families.

This research explores a different approach, namely the data-centric approach, to tackle the MSA construction problem. Instead of relying on abstract models, we studied how to apply machine-learning algorithms or deep-learning algorithms to develop models from protein family data and use them to construct better MSAs.

Using the data-centric method to tackle difficult problems in computer science is rapidly gaining popularity because of recent advances in machine learning and deep learning, which have also been applied successfully in Bioinformatics [12]–[16]. However, there is still no notable research on applying machine learning or deep learning to the MSA construction problem.

1.2 Problem definition

MSA construction problem can be defined as the following mathematical problem. Given n sequences $S_i, i = 1, 2, \cdots, n$

$$S := \begin{cases} S_1 = (S_{11}, S_{12}, \cdots, S_{1m_1}) \\ S_2 = (S_{21}, S_{22}, \cdots, S_{2m_2}) \\ \vdots \\ S_n = (S_{n1}, S_{n2}, \cdots, S_{nm_n}) \end{cases}$$

an MSA is constructed from this set of sequences by inserting an appropriate amount of gaps needed into each of the S_i sequences of S until the modified sequences, S_i', all conform to a same length l and no values in the sequences of S of the same column m, consists of only gaps. The mathematical form of an MSA of the above sequence set is shown below:

$$S' := \begin{cases} S_1' = (S_{11}', S_{12}', \cdots, S_{1l}') \\ S_2' = (S_{21}', S_{22}', \cdots, S_{2l}') \\ \vdots \\ S_n' = (S_{n1}', S_{n2}', \cdots, S_{nl}') \end{cases}$$

1.3 Previous research

1.3.1 Algorithm-centric approaches in MSA

The Needleman-Wunsch algorithm [17] and the Smith-Waterman algorithm [18] are two representative sequence alignment algorithms that resulted from dynamic programming thought in computer science. However, the time complexity of the dynamic programming algorithm is $O(LN)$, where L stands for the length of the longest sequence in a protein family and N is the size of that protein family, which means the two methods are very time consuming. Subsequently, many exciting and sophisticated mathematical and algorithmic techniques were applied to solve the MSA construction problem (e.g., fast Fourier transform, the constraint-based strategy, the divide-and-conquer strategy, iterative refinement, homology extension, and the progressive and non-progressive strategies).

The progressive alignment strategy is one of the most mature MSA strategies, with considerable research validation and the highest accuracy. Progressive methods usually contain five main steps: (1) posterior probability matrix calculation, (2) distance matrix calculation, (3) "Guide Tree" generation by clustering methods, (4) consistency transformation and (5) refinement. This

process has determined the direction of many studies.

CLUSTAL [19], the representative procedure for the second and third steps, calculated the distance matrix to construct a tree structure and then progressively aligned every two sequences to get the MSA. This method dramatically speeded up the MSA construction process, so that it was feasible to run an MSA program on a personal computer. But with the improvement in efficiency, accuracy decreased. Instead of calculating the distance matrix, T-Coffee [20] introduced the substitution matrix for tree construction, called "Guide Tree". However, it was very time consuming. Then, the fast Fourier transform (FFT) method was adopted to count the number of exact character matches between two sequences, which could be applied to build a data structure called an approximate steering tree. Unlike previous studies, MUSCLE [21] used k-mer counting for the distance computation between every two sequences and included a post-processing module to improve the quality of the MSA, providing a balance between efficiency and accuracy. MAFFT [4] applied the FFT method to recognize homologous regions in various sequences for special processing. At the same time, a simplified scoring scheme was introduced in MAFFT, which significantly reduced the running time of the program, so that even when a large amount of data was processed, it still had good results. PartTree [22] constructed the guide tree in $O(N \log N)$ time using an approximation algorithm, where N is the size of a single protein family.

While the hidden Markov model (HMM) proved to be beneficial in calculating posterior probabilities, ProbCons [23] adopted this in a posterior probability matrix calculation and introduced a term called "probabilistic consistency", which could be employed in an MSA construction and improved its accuracy. ProbAlign [24] introduced a method called a partition function to calculate the posterior probabilities faster and more accurately. MSAProbs [25] combines the two calculation methods on a posterior probability matrix, which results in a more accurate MSA construction. GLProbs [26] calculates the sequence similarity, adaptively deciding which posterior probability matrix calculation method to use according to the sequence similarity (also known as percentage identity (PID)) as the breakthrough points. These MSA methods greatly improved accuracy.

A more recent development, QuickProbs [27], performs MSAProbs with the OpenCL library, significantly reducing the running time for constructing an MSA. In QuickProbs 2 [28], accuracy was improved by changing the scoring matrix and column-based refinement.

Expresso [29] made an attempt to identify a structure for every sequence and subsequently applied a structural aligner onto the templates associated with every pair of sequences. PSI-Coffee [30] combined homology extension and consistency based progressive alignment. It was designed for aligning distantly related proteins for which no structural information was available. These two methods are the main examples of introducing external information into the traditional progressive alignment strategy to improve the accuracy.

PicXAA [31] greedily builds up the alignment from sequence regions with high local similarity, thereby yielding an accurate global alignment that effectively grasps local similarity among sequences. This is the only complete non-progressive alignment method mentioned in this thesis.

A similar non-progressive strategy was integrated into PnpProbs [32]. The non-progressive strategy has proved to be more accurate in constructing MSAs from divergent protein families.

With the emergence of large amounts of protein family data, the challenge is that MSA construction methods are very complicated when encountering extensive data. When facing the problem of super-large sequence alignment, many MSA tools have problems such as low accuracy and long running time. Therefore, the decomposition strategy was proposed and developed rapidly. Inspired by the divide-and-conquer algorithm, the fundamental idea of this method is to split large protein families into several small subsets, run MSA tools on different subsets to get the alignment results of the subgroups, and finally merge them to get the outcomes. SATé-I [33] uses the maximum likelihood estimation to resolve how to separate subsets and deals with a huge amount of data recursively. Researchers further improved the accuracy and time efficiency of the tool by applying different dividing strategies and different tree construction techniques, producing a new tool, named SATé-II [34]. PASTA [35] proposed a new design in the tree construction process, which further improved the time efficiency by binary merging. UPP [36] successfully applied machine learning methods to the MSA task, proving that machine learning can indeed be beneficial to MSA construction. But the accuracy of these MSA methods on large-scale datasets is relatively low.

1.3.2 Data-centric approaches in Bioinformatics

With the coming changes in the amount and diversity of datasets, data-centric approaches that compute on massive amounts of data (often called “Big Data” [37], [38]) to discover patterns and to make clinically relevant predictions would

be increasingly common in translational bioinformatics [39].

In 2005, [40] described a data-centric software architecture for bioinformatics workflows and a rule-based workflow enactment system that used declarative specifications of data dependencies between steps to automatically order the execution of those steps. WebLab [41] was a data-centric knowledge-sharing platform which was designed for biologists to fetch, analyze, manipulate and share data under an intuitive web interface. In 2010, [42] has shown that a proposed model for the preparation of data input can substantially improve the utility of motif finding software. At the same year, a data-centric system for integrating bioinformatics applications was built whose name was EvolvingSpace [43] which generalized data annotations in bioinformatics field to a set of data entities and provided a decentralized data management system for storing and retrieving these data entities. In 2011, [44] applied trend analysis to the EMR data from 98 patients to "learn" a data-driven guideline on how to provide care for a 13-year-old girl with systemic lupus erythematous which was particularly useful when derivation of a formal guideline was not feasible from a practical standpoint. DeepNovo [14], introduced deep learning to de novo peptide sequencing from tandem Mass Spectrometry (MS) data, the key technology for protein characterization in proteomics research, which achieved a major improvement of sequencing accuracy over the state-of-art methods and subsequently enabled the complete assembly of protein sequences without assisting databases. In 2019, [45] further extended DeepNovo on Data Independent Acquisition (DIA) MS data and proposed DeepNovo-DIA, the first de novo peptide sequencing algorithm for DIA MS/MS spectrums. In addition to these listed above, there are many applications of data-centric approaches represented by deep learning or machine learning in Bioinformatics [46], [47], [48], [49], [50], [51], [52], [53].

1.4 Benchmarks and measurements

After decades of development, many sequences manually labeled by biologists have been collected as different benchmarks. The effect of all MSA methods and programs can be determined against specific indicators in these benchmarks.

1) **BAliBASE** [54] is a large scale benchmark designed explicitly for multiple sequence alignment, providing high-quality reference alignments based on 3D structural superpositions. Alignment test cases are manually refined to ensure the correct alignment of conserved residues in this benchmark.

We obtained a part of the extension set of BAliBASE, which we named BAliBASE-X.

2) The **OXBench** [55] data set is made up of domain families obtained from the 3Dee database [56] of protein structural domains. After filtering these families using different criteria, we determined reference structural alignments with the STAMP algorithm [57]. The initial reference data set of domain family alignments was extended and subdivided in various ways to allow the study of different aspects of the protein sequence alignment problem. We also obtained an extension set of OXBench, named OXBench-X.

3) **SABmark** [58] provides sets of multiple alignment problems derived from the SCOP [59] classification. These sets, Twilight Zone and Superfamilies, cover the entire known fold space using sequences with very low to low, and low to intermediate similarity, respectively

4) **SISYPHUS** [60] contains a collection of manually curated structural alignments and their inter-relationships. The alignments are constructed for protein structural regions that range from oligomeric biological units or individual domains to fragments of different size.

5) **HOMSTRAD** [61] is a database that provides combined protein sequence and structure information extracted from the Protein Data Bank (PDB) [62], a primary protein structure repository. HOMSTRAD relies heavily on other databases, especially Pfam [63] and SCOP. It contains about 2,700 families, just under half of which are multi-sequence.

6) **Mattbench** [64] is a protein structural alignment benchmark used to test and refine protein sequence aligners, which relies on the Matt [65] protein structural aligner.

Table 1 summarizes for each benchmark dataset the number of families and the total number of sequences in it.

We chose BAliBASE v3, OXBench v1.3 and SABMark v1.65, which are commonly adopted by most MSA tools, as evaluation benchmarks in our research. The detailed information in these three benchmarks is shown in Table 2. The three real empirical benchmarks are obtained from "BENCH" [66], which includes many multiple sequence alignment benchmarks in a standard FASTA format.

	Num. of families	Total num. of sequences
BAliBASE	386	11082
OXBench	395	3292
SABmark	423	2418
SISYPHUS	126	1772
HOMSTRAD	1030	3454
Mattbench	259	1698
Total	2619	23716

Table 1 The number of families and the total number of sequences in each benchmark.

Information	BAliBASE	OXBench	SABMark
Number of Families	386	395	423
MinimumLength of Sequences	36	42	30
Maximum Length of Sequences	7923	544	796
Average Length of Sequences	**344.23**	**124.82**	**173.87**
Minimum Number of Sequences	4	3	3
Maximum Number of Sequences	142	122	25
Average Number of Sequences	**28.71**	**8.33**	**5.72**

Table 2 The Information of three empirical benchmarks

To measure MSA quality, the total-column score (TC-score), which was first introduced in BAliBASE [67], is the most popular measurement in many alignment benchmark tests. The TC-score represents the percentage of the correctly aligned columns compared with the references. Also, the sum-of-pairs score (SP-score), denoting the sum of all pairwise induced alignment scores, is widely used by other MSA tools to evaluate their accuracy. These are the main indicators compared in this thesis. Qscore [68], an MSA quality scoring program that can obtain important indicators like the SP-score and TC-score, was adopted in this thesis.

1.5 Main contributions

The research results listed in this thesis include two main levels:

(1) at the high level, we use classifiers to choose a better MSA tool to construct a basic MSA, and then, based on this temporary MSA, we use another MSA tool to realign the different regions, which is determined by another classifier; and

(2) at the low level, we first select the most prominent part from the standard progressive alignment method, and then use deep learning to train a decision-making model. Then, based on this decision model, we propose a new progressive alignment tool for multiple protein sequences.

For these two different research results, we published two data-centric MSA methods: MLProbs for the high level and DLPAlign for the low level. Next, we explain the separate contributions of these two methods.

1.5.1 MLProbs: A Data-centric Pipeline for better Multiple Sequence Alignment

A critical factor for the success of a machine-learning application is whether there is enough high-quality data to train an effective model. The training data used in this part are the 6000+ protein families obtained from the "BENCH" website. Since this number of families may not be enough to train deep-learning models such as CNN and LSTM, we applied shallow machine-learning algorithms like Random Forest for the training. We used the resulting models to construct MSAs. We evaluated the alignment accuracy of our methods using the latest versions of the golden benchmark databases SABmark, BAliBASE, and OXBench, along with its extension set OXBench-X.

Details of our methods and their implementation are shown in Chapter 2.

Using this method, we built a pipeline, called MLProbs, which applies our method to construct an MSA. We compared the alignment accuracy of MLProbs with 10 other popular MSA tools: PnpProbs, QuickProbs, GLProbs, PicXAA, ProbCons, MSAProbs, MAFFT, Muscle, ClustalΩ [19] and ProbAlign. Tables8, 9, 10 and 11 in Chapter 2 show the alignment accuracy of the MSAs constructed by the tools for families in SABMark, BAliBASE, OXBench and OXBench-X. For all four databases, MLProbs consistently achieved the highest TC-score of all the tools.

MLProbs performed particularly well for families with low similarity. To provide an overall picture, Figure 6 compares the TC scores of the tools of all families in SABMark, BAliBASE, OXBench and OXBench-X with PID $\leq 50\%$ (1,356 such families in total). MLProbs had the highest TC-score, and its improvement over the second best tool was more than triple the second-best tool's improvement over the third best tool. As we mentioned earlier, obtaining better alignments for these families with low similarity is always a challenge for research in MSA construction, so the MLProbs improvement will have a significant impact.

To aid verification of our results, we uploaded MLProbs, as well as copies of SABMark, BAliBASE, OXBench and OXBench-X, to GitHub (`https://github.com/kuangmeng/MLProbs`).

1.5.2 DLPAlign: A Deep Learning based Progressive Alignment for Multiple Protein Sequences

After determining which specific part in the progressive MSA method to improve, we transformed the classification of MSA families into the classification of sequence pairs, thus obtaining large-scale training data (954,854 sequence pairs). We took this protein-sequence data from the following datasets: SISYPHUS, SABmark, BAliBASE, OXBench, HOMSTRAD and Mattbench.

We used deep-learning methods to train decision models to help us choose the most appropriate calculation method for each specific part. We provide more details of the decision-making model and the implementation in Chapter 3.

Based on the most accurate decision-making model, we built a new progressive MSA tool, called DLPAlign. We compared DLPAlign with the 10 other MSA tools mentioned above on three empirical benchmarks: SABmark, BAliBASE and OXBench. Tables 18, 19 and 20 in Chapter 3 show the alignment accuracy

of the MSAs constructed using the tools for families in BAliBASE, OXBench and SABMark, respectively. DLPAlign achieved the highest TC-score of all the tools against all benchmarks.

Families with low similarity have always been the most challenging part of MSA. DLPAlign performed better on low and medium similarity protein families, which the other progressive methods were not good at. Figures 16 and 17 compare the average TC-scores for low similarity families (PID $\leq$ 30%; 711 families in all) and medium similarity (30% $<$ PID $\leq$ 60%; 352 families in all) in BAliBASE, OXBench and SABmark. As the figures show, the improvement in DLPAlign is pronounced, especially for low similarity families.

We think this tool can be used in actual MSA construction tasks, so we uploaded the source code, as well as the benchmarks for testing, to GitHub (`https://github.com/kuangmeng/DLPAlign`).

1.6 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 provides an overview of MLProbs, a high-level data-centric MSA method, followed by elaboration of its implementation and discussion of its experimental results and applications. Chapter 3 describes a novel low-level data-centric progressive alignment method for multiple protein sequences, namely DLPAlign, from the perspective of the best promotion part selection and the best decision-making model. Chapter 4 proposes discussions and future works related to this research.

Chapter 2

MLProbs: A Data-centric Pipeline for better Multiple Sequence Alignment

2.1 Overview

This part of our research explores the data-centric approach to tackling the MSA construction problem. Instead of relying on abstract models, we studied how to apply machine-learning algorithms to learn models from protein families data and use them to help construct better MSAs.

This section explains how we use the machine-learning methods to help make decisions based on existing MSA tools at a high level. There are three main aspects to our research:

(1) how to use machine learning to help us make decisions and choose the best decisions from existing MSA tools;

(2) how to get better column-based realignment for the MSA results we have already obtained and using machine learning to choose the best realignment strategy; and

(3) how to combine the top performers in parts 1 and 2 to obtain a new, highly accurate pipeline, which we called MLProbs.

In the following sections, we explore how machine learning can be used to answer the following questions:

(i) how to choose the best tools to align a family; and

(ii) whether and how much to carry out realignment of an MSA for an input family to improve its accuracy.

We describe our methods and provide details about their implementation in each section.

2.2 $\mathscr{P}^{\texttt{Aln}}_{\texttt{U,V}}$: A data-centric method for choosing better tools

Over the past few decades, many MSA construction tools have been designed and implemented, and they have their own strengths and weaknesses. For example, progressive alignment tools work better in general, while non-progressive alignment tools are more suitable for aligning divergent families. An obvious way to improve MSA construction is as follows:

> Given an input family, we first decide which MSA tool will give the best result, and then use that tool to construct the MSA.

We propose applying machine learning to help us make the right decision. Our study focuses on the case when there are only two tools to choose. Consider any two MSA tools `U` and `V`. We define the following binary classification $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$ on protein families:

> $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$ has two classes `0` an `1`. A protein family $\mathscr{F}$ is in class `1` if the MSA constructed by `V` is better than that constructed by `U`; otherwise, $\mathscr{F}$ is in `0`.

We used the popular *TC score* to measure the goodness of an alignment. The TC score of an alignment $\mathscr{M}$ is the percentage of columns in $\mathscr{M}$ that are identical to the corresponding columns in the reference alignment (which is given in the Section 1.4). We note that different implementations may have slightly different ways to compute the TC scores (e.g., some do not consider columns with gapped entries). In this research, we used qscore (`http://www.drive5.com/qscore`) to compute all the TC scores.

The key concern is how to build an accurate model for $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$, which can naturally lead to way to outperform `U` and `V`: We use machine learning algorithm to construct a model (or classifier) for $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$. Then, we construct the alignments using the pipeline $\mathscr{P}^{\texttt{Aln}}_{\texttt{U,V}}$, which, given an input family $\mathscr{F}$, it first uses the classifier to determine to which class in $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$ $\mathscr{F}$ belongs, and it uses `V` to construct an alignment for $\mathscr{F}$ if the family belongs to the class `1`; otherwise, it uses `U`.

We have implemented the pipelines $\mathscr{P}^{\texttt{Aln}}_{\texttt{U,V}}$ for various tools `U` and `V`. We have particular interest in the tool PnpProbs. To construct an MSA for input family $\mathscr{F}$, PnpProbs first computes the average PID of $\mathscr{F}$, and if it is no smaller than 18%, PnpProbs calls a progressive alignment procedure `P` to construct the MSA; otherwise it calls a non-progressive alignment procedure `NP`. We have

tried very hard to find other statistical conditions and algorithmic methods for helping us make better decision on the choice of P or NP, but all our efforts were in vain. It is interesting to find out if our data-centric method can help us make better decision, or more precisely, whether our trained classifier $\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ has better precision and recall. This is indeed the case; as can be seen from Table 3, the precision and recall of $\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ are significantly higher than those of the 18%-rule of PnpProbs.

	Precision	Recall
$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$	86.28	93.70
The 18%-rule	79.43	80.80

Table 3 Comparing $\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ and PnpProbs's 18%-rule.

Based on $\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$, we have implemented the pipeline $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$, and we have also implemented the pipelines $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{Pnp,Q}}$, $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{GL,MSA}}$, $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{GL,Pic}}$, and $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{MSA,MAF}}$ for PnpProbs(Pnp), QuickProbs (Q), GLProbs (GL), MSAProb (MSA), PicXAA (Pic), and MAFFT (MAF). Table 4 summarizes the accuracy of the classifiers, and Table 5 compares the TC scores of the tools and pipelines on the four benchmark databases.

	Precision	Recall	F_1-score
$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$	86.28	93.70	89.84
$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{Pnp,Q}}$	82.36	86.27	84.27
$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{GL,MSA}}$	84.66	87.97	86.28
$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{GL,Pic}}$	86.12	91.38	88.67
$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{MSA,MAF}}$	84.23	86.88	85.53

Table 4 Testing results for the classifiers $\mathscr{C}^{\mathtt{Aln}}_{\mathtt{U,V}}$.

We note that the TC scores obtained by $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ are consistently and significantly higher than those by PnpProbs. Moreover, among all the fifteen tools and pipelines in the table, $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ has the highest TC scores for OXBench-X, OXBench and SABMark. MLProbs is based on $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$, and with additional help from the realignment methods given in the next subsection, it achieves the best alignment accuracy for all four databases.

$\mathscr{C}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ are not exceptionally good, but Table 5 shows that $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ can still help improve the alignment accuracy of PnpProbs. In fact, in many cases, even

though the classifier $\mathscr{C}^{\texttt{Aln}}_{\texttt{P,NP}}$ advises a wrong tool for input family $\mathscr{F}$, the TC scores of the alignments constructed by `P` and `NP` for $\mathscr{F}$ are very close, and thus the mistake does not have much effect on the alignment accuracy. It is noteworthy that if we can choose `P` or `NP` correctly for every family, then the average TC scores are 63.86, 60.45, 82.95, 42.70 for BAliBASE, OXBench-X, OXBench and SABmark, respectively. Hence, there is still room for improvement if we can train some better model for $\mathscr{C}^{\texttt{Aln}}_{\texttt{P,NP}}$.

For the other four pipelines, they all obtain higher scores for the three benchmark databases OXBench-X, OXBench and SABMark. However, on BAliBASE our approach is not effective for the three pipelines $\mathscr{P}^{\texttt{Aln}}_{\texttt{Pnp,Q}}$, $\mathscr{P}^{\texttt{Aln}}_{\texttt{GL,MSA}}$ and $\mathscr{P}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$. We observe that for these three pipelines, the TC scores between the pair of tools on BALiBASE differ by at least 2.42%, for all the other cases (i.e., for the case when the pipelines have improvement) the differences are no more than 1.43%. In fact, it is quite clear why $\mathscr{P}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$ can make no improvement on BALiBASE. MSAProbs (with score **64.51**) is dominatingly better than MAFFT (with score **50.08**) and we should choose MSAProbs for most of the families in BAliBASE, and for the families that MAFFT is better, its TC scores are not much higher than those of MSAProbs. Thus, mistakes made by the classifier $\mathscr{C}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$ that chooses MAFFT instead of MSA are serious enough to reduce the average TC score of the pipeline $\mathscr{P}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$ to one smaller than that of MSAProbs.

2.3 $\mathscr{P}^{\texttt{Ral}}_{\texttt{U}}$: A data-centric method for better realignment

Realignment is often the last step of an MSA construction tool for improving alignment accuracy. This paper focuses on the following realignment approach proposed in [69], [70]: Given an alignment $\mathscr{M}$, we identify regions in $\mathscr{M}$ (i.e., blocks of consecutive columns of $\mathscr{M}$) that are unreliable, and then realign these regions to repair some of the misaligned parts.

Unlike sequence segmentation based refinement, such as iterative refinement or tree-based refinement, our column-based realignment is more able to find small local errors in the alignment process. Figure 1 shows an alignment example of before and after performing our realignment process in the OXBench-X database. From this figure, we could see that the realignment is very useful in some extents.

There are many algorithmic techniques proposed for determining unreliable regions [69]–[74], and ours is based on column scores. The *score of a column*

	BAliBASE	OXBench-X	OXBench	SABMark
P	62.17	59.53	82.18	41.39
NP	60.74	57.77	82.05	41.28
$\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$	63.30	60.10	82.43	42.13
PnpProbs	62.46	59.54	82.06	41.48
QuickProbs	65.41	59.44	81.77	40.65
$\mathscr{P}^{\mathtt{Aln}}_{\mathtt{Pnp,Q}}$	64.31	60.07	82.31	41.81
GLProbs	62.09	59.34	81.93	41.22
MSAProbs	64.51	59.37	81.50	40.04
$\mathscr{P}^{\mathtt{Aln}}_{\mathtt{GL,MSA}}$	63.83	59.89	82.19	41.56
GLProbs	62.09	59.34	81.93	41.22
PicXAA	60.97	58.85	81.14	38.44
$\mathscr{P}^{\mathtt{Aln}}_{\mathtt{GL,Pic}}$	62.81	59.66	82.34	41.67
MSAProbs	64.51	59.37	81.50	40.04
MAFFT	50.08	56.90	78.15	33.00
$\mathscr{P}^{\mathtt{Aln}}_{\mathtt{MSA,MAF}}$	64.38	60.03	81.91	40.19

Table 5 TC scores obtained by the pipeline $\mathscr{P}^{\mathtt{ALn}}_{\mathtt{U,V}}$.

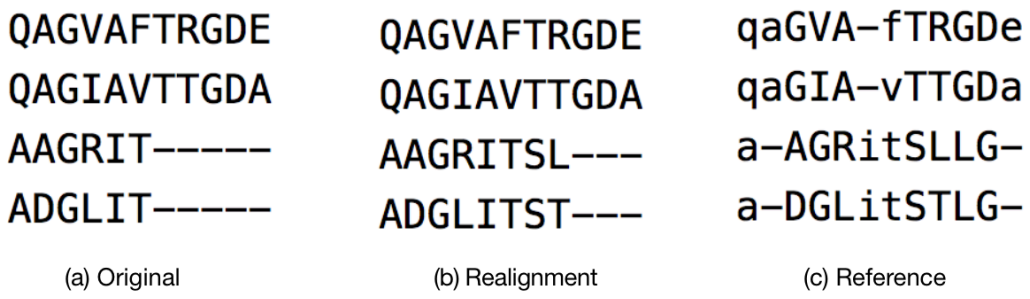

Figure 1 The original alignment, realignment and reference alignment of a real protein family in OXBench-X. *The 8th and 9th columns in "Original" have been corrected.*

is defined to be the average score of the amino acid pairs[1] in the column as illustrated at Figure 2, and a column is unreliable if its score is smaller than some predetermined threshold. Unreliable regions are simply blocks of maximal consecutive of unreliable columns. Intuitively we should realign them. However, we observe that there are two decisions needed to be made correctly in order to make our realignment procedure effective.

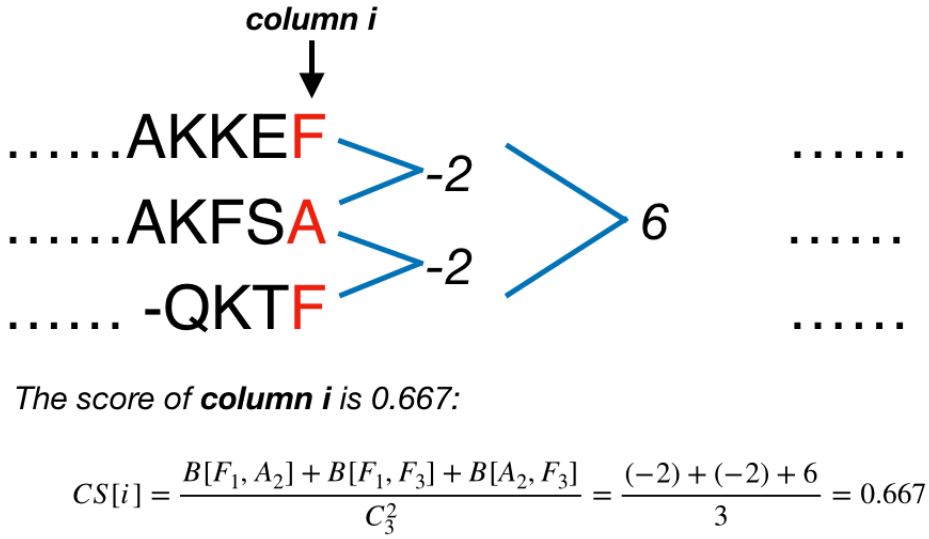

Figure 2 An example of the calculation of *score of a column*.

To realign or not to realign?

Our study shows that realignment is rather effective for conserved families, and our explanation is that for such families, the reliable regions (i.e., the regions between the unreliable ones) are often correctly aligned, and hence can correctly "isolate" the subsequences in the unreliable regions, and none of their residues would be aligned to any residue outside the regions. Therefore, it is safe to focus on realigning the subsequences in an unreliable region, and by ignoring the noises from outside, we have a better chance of getting a better alignment.

However, the situation is different for divergent families. For such families, the subsequences in an unreliable region are often highly dissimilar, and without extra information, even biologists may not be able to construct a good alignment. Thus, realigning these subsequences will not help, and it may even reduce the quality of the original alignment because when constructing the

[1] We use the BLOSUM62 substitution matrix to determine the score of any pair of amino acids.

original alignment, the tool has the advantage of using information from other parts of the family to help align this unreliable region (for example, for a progressive alignment tool, the hidden Markov model constructed based on the whole family is likely better than that constructed based on the subsequences in an unreliable region). Therefore, for divergent families, it may be better not to realign their unreliable regions. In fact, our experiments show that for these families, realigning their reliable regions occasionally improve the alignments.

Therefore, given an MSA, we need to decide whether we should realign its reliable regions or realign their unreliable regions. To help make the decision, we resort to machine learning again. We define the following classification $\mathscr{C}_{\mathtt{U}}^{\mathtt{Ral}}$ on all possible MSAs, where `U` is the alignment tool used to do the realignment:

> $\mathscr{C}_{\mathtt{U}}^{\mathtt{Ral}}$ has two classes `1` and `0`, and an MSA $\mathscr{M}$ is in class `1` if it is better to realign the reliable regions (i.e., if the MSA constructed by realigning $\mathscr{M}$'s reliable regions has a TC score higher than the one constructed by realigning $\mathscr{M}$'s unreliable regions); otherwise, $\mathscr{M}$ is in class `0`.

How wide should an unreliable region be?

Realigning an unreliable region with only one or two columns is unlikely to improve the whole alignment. Thus, we should skip unreliable regions that are too narrow, and only realign those with at least a minimum width (i.e., a minimum number of columns). Our study showed that different families might adopt different minimum widths in order to make the realignment step most effective. To help decide the right one, we define the following classification $\mathscr{C}_{\mathtt{U}}^{\mathtt{mw}}$, where `U` is the tool used to realign the unreliable regions.

> Given an MSA $\mathscr{M}$, let $\mathscr{M}_i$ denote the alignment obtained by using `U` to realign all the unreliable regions with width no smaller than i. The classification $\mathscr{C}_{\mathtt{U}}^{\mathtt{mw}}$ has four classes `2`, `10`, `20`, `30`, and $\mathscr{M}$ is in class i if $\mathscr{M}_i$ has the maximum TC score among those of $\mathscr{M}_2$, $\mathscr{M}_{10}$, $\mathscr{M}_{20}$ and $\mathscr{M}_{30}$.

The classifications $\mathscr{C}_{\mathtt{U}}^{\mathtt{mw}}$ and $\mathscr{C}_{\mathtt{U}}^{\mathtt{Ral}}$ suggest a natural pipeline $\mathscr{P}_{\mathtt{U}}^{\mathtt{Ral}}$ for the realignment step: Given any MSA $\mathscr{M}$, we first determine the class in $\mathscr{C}_{\mathtt{U}}^{\mathtt{Ral}}$ to which $\mathscr{M}$ belongs . If $\mathscr{M}$ is in class `1`, we return the new MSA obtained by using `U` to realign the reliable regions of $\boldsymbol{M}$. Otherwise, we determine the class i in $\mathscr{C}_{\mathtt{U}}^{\mathtt{mw}}$ that $\mathscr{M}$ belongs, and then return the alignment obtained by using `U` to realign all the unreliable regions of $\mathscr{M}$ with width no smaller than i.

To evaluate our realignment approach, we have implemented the pipelines $\mathscr{P}_{\mathtt{Q}}^{\mathtt{Ral}}$, $\mathscr{P}_{\mathtt{GL}}^{\mathtt{Ral}}$, $\mathscr{P}_{\mathtt{MSA}}^{\mathtt{Ral}}$ and $\mathscr{P}_{\mathtt{MAF}}^{\mathtt{Ral}}$. Our preliminary study shows that to get better results, we need to refine our notion of reliable and unreliable regions as follows. We say that a column of an MSA is

- reliable if its column score is greater than 2;
- it is *fuzzy* if its score is between 1.2 and 2;
- it is unreliable if its score is smaller than 1.2 and greater than zero, and
- it is *messy* if its score is negative.

We define a reliable region to be a maximal consecutive of reliable columns. We define fuzzy region, unreliable region and confusing region similarly. Our refinement identifies some of the regions that we do not have much confidence, namely the messy regions (because the subsequences in these regions are so different that we have little hope to make any improvement) and the fuzzy regions (because it is hard to decide whether they are reliable or unreliable), and our realignment procedure will ignore the fuzzy and the messy regions and focus on the reliable and unreliable regions.

Table 6 summarises the results of our testing of the classifiers for the four pipelines we have trained.

To measure the effectiveness of a realignment pipeline $\mathscr{P}_{\mathtt{U}}^{\mathtt{Ral}}$, we compare the alignment accuracy of $\mathtt{U}$ and that of the combination of $\mathscr{P}_{\mathtt{U}}^{\mathtt{Ral}}$ and $\mathtt{U}$, denoted by $\mathscr{P}_{\mathtt{U}}^{\mathtt{Ral}} \circ \mathtt{U}$, which works as follows:

> Given an input family $\mathscr{F}$, first uses $\mathtt{U}$ to construct an MSA $\mathscr{M}$ for $\mathscr{F}$, and then uses $\mathscr{P}_{\mathtt{U}}^{\mathtt{Ral}}$ to realign the reliable/unreliable regions of $\mathscr{M}$.

We have implemented the pipeline $\mathscr{P}_{\mathtt{U}}^{\mathtt{Ral}} \circ \mathtt{U}$ for the tools QuickProbs (`Q`), GLProbs (`GL`), MSAProbs (`MSA`) and MAFFT(`MAF`). Table 6 summarises the accuracy of the classifiers, and Table 7 summarizes the TC scores obtained by the pipelines for BAliBASE, OXBench-X, OXBench and SABMark. We note that the realignment step improves the scores in all cases. Furthermore, $\mathscr{C}_{\mathtt{Q}}^{\mathtt{Ral}} \circ \mathtt{Q}$ has the highest TC scores for all four databases.

	Precision	Recall	F_1-score
$\mathscr{C}^{\mathtt{Ral}}_{\mathtt{Q}}$	87.35	93.64	90.39
$\mathscr{C}^{\mathtt{Ral}}_{\mathtt{GL}}$	88.95	84.07	86.44
$\mathscr{C}^{\mathtt{Ral}}_{\mathtt{MSA}}$	81.27	84.65	82.93
$\mathscr{C}^{\mathtt{Ral}}_{\mathtt{MAF}}$	90.85	87.65	89.22

Table 6 Testing results for the classifiers $\mathscr{C}^{\mathtt{Ral}}_{\mathtt{U}}$

	BAliBASE	OXBench-X	OXBench	SABMark
QuickProbs	65.41	59.44	81.77	40.65
$\mathscr{P}^{\mathtt{Ral}}_{\mathtt{Q}} \circ \mathtt{Q}$	65.42	59.74	81.95	40.73
GLProbs	62.09	59.34	81.93	41.22
$\mathscr{P}^{\mathtt{Ral}}_{\mathtt{GL}} \circ \mathtt{GL}$	62.12	59.49	82.24	41.41
MSAProbs	64.51	59.37	81.50	40.04
$\mathscr{P}^{\mathtt{Ral}}_{\mathtt{MSA}} \circ \mathtt{MSA}$	64.52	59.42	81.74	40.08
MAFFT	50.08	56.90	78.15	33.00
$\mathscr{P}^{\mathtt{Ral}}_{\mathtt{MAF}} \circ \mathtt{MAF}$	50.42	56.99	78.30	33.07

Table 7 Average TC-scores of the pipelines $\mathscr{P}^{\mathtt{Ral}}_{\mathtt{U}} \circ \mathtt{U}$.

2.4 Trainings and evaluations of the classifiers

In the previous two sections, we mentioned three classifiers. In this section, we will integrate them to explain how to train and use them.

We use shallow machine learning algorithms to construct the classifiers, and the challenge is to determine appropriate features for representing the inputs so that based on them our learning algorithms can train good models without the need of excessively large amounts of training data. To describe the features we use, we need some definitions.

Consider any two protein sequences s_1 and s_2. We let $\texttt{ln}(s_1, s_2)$ denote the length of the optimal (pairwise) alignment $\mathscr{M}$ of s_1 and s_2, and define $\texttt{pid}(s_1, s_2)$, the percentage identity of s_1 and s_2, to be the percentage of columns of $\mathscr{M}$ with identical amino acids, and $\texttt{sc}(s_1, s_2)$, the sum of column scores of s_1 and s_2, to be the total sum of scores of amino acid pairs at the same columns of $\mathscr{M}$.

Consider any protein family $\mathscr{F}$. We let $\texttt{sz}(\mathscr{F})$ denote the total number of sequences in $\mathscr{F}$, and define $\texttt{av}(\texttt{pid}(\mathscr{F}))$ and $\texttt{sd}(\texttt{pid}(\mathscr{F}))$ to be the average and the standard deviation of the $\texttt{pid}(s_1, s_2)$'s over all pairs of sequences s_1 and s_2 in $\mathscr{F}$ (and we will simply write $\texttt{av}(\texttt{pid})$ and $\texttt{sd}(\texttt{pid})$ if there is no risk of confusion). Define $\texttt{av}(\texttt{sc})$, $\texttt{sd}(\texttt{sc})$, $\texttt{av}(\texttt{ln})$ and $\texttt{sd}(\texttt{ln})$ similarly.

Consider any multiple sequence alignment $\mathscr{M}$. Recall that the column score of any column C of $\mathscr{M}$ is defined to be the sum of the scores of all possible amino acids pairs at C. Fix any small constant $\delta > 0$ (in all of our experiments, we set $\delta = 1$). Define the *peak-length ratio* of $\mathscr{M}$, denoted as $\texttt{pl}_\delta(\mathscr{M})$ or simply $\texttt{pl}(\mathscr{M})$, to be the ratio between the total number of columns of $\mathscr{M}$ with column scores greater than δ and the total number of columns of $\mathscr{M}$.

We use the following features for trainings the classifications $\mathscr{C}^{\texttt{aln}}_{\texttt{U,V}}$, $\mathscr{C}^{\texttt{Ral}}_{\texttt{U}}$ and $\mathscr{C}^{\texttt{mw}}_{\texttt{U}}$:

- $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$: $\texttt{av}(\texttt{pid}), \texttt{av}(\texttt{sc}), \texttt{av}(\texttt{ln})$, $\texttt{pl}$ and $\texttt{sz}$.
- $\mathscr{C}^{\texttt{Ral}}_{\texttt{U}}$: $\texttt{av}(\texttt{pid}), \texttt{av}(\texttt{sc}), \texttt{sd}(\texttt{sc})$ and $\texttt{pl}$.
- $\mathscr{C}^{\texttt{mw}}_{\texttt{V}}$: $\texttt{av}(\texttt{pid}), \texttt{sd}(\texttt{pid})$, $\texttt{av}(\texttt{ln})$, and $\texttt{sz}$

We use the Random Forest machine learning algorithm to train all the classifiers. The data we use are obtained from the website BENCH, which has totally 6592 protein families. In each training, we randomly pick 70% of them

for training, and the remaining 30% for testing. Note that BENCH contains both DNA and protein families, and our experiments only use the protein families. Following is a summary of the data we use.

- The total number of protein families we used is 6592, and 4214 of them have more than two sequences.
- The total number of protein sequences is 151,340.
- The minimum, average and maximum length of the sequences are 24, 210.7 and 7923, respectively.

Some details about the Precision, Recall and F_1-score of $\mathscr{C}^{\texttt{Aln}}_{\texttt{U,V}}$ and $\mathscr{C}^{\texttt{Ral}}_{\texttt{U}}$ have been mentioned in the previous two sections. Here we only show the comprehensive performance of $\mathscr{C}^{\texttt{mw}}_{\texttt{V}}$ on four benchmarks in Figure 3.

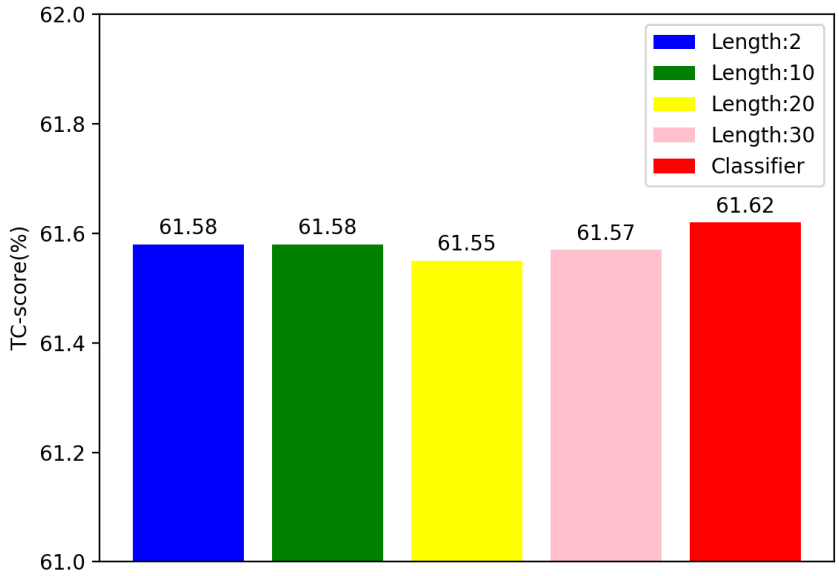

Figure 3 Average TC-scores on four empirical benchmarks (BAliBASE, OXBench, OXBench-X, SABMark) with different minimum realignment widths

2.5 An implementation of the pipeline $\mathscr{P}^{\texttt{Ral}}_{\texttt{Q}} \circ \mathscr{P}^{\texttt{Aln}}_{\texttt{P,NP}}$

Tables 5 and 7 show that $\mathscr{P}^{\texttt{Aln}}_{\texttt{P,NP}}$ and $\mathscr{P}^{\texttt{Ral}}_{\texttt{Q}}$ are the top performers. And the comprehensive performance of these two pipelines on the four benchmarks can be seen from the Figures 4 and 5, which are really good.

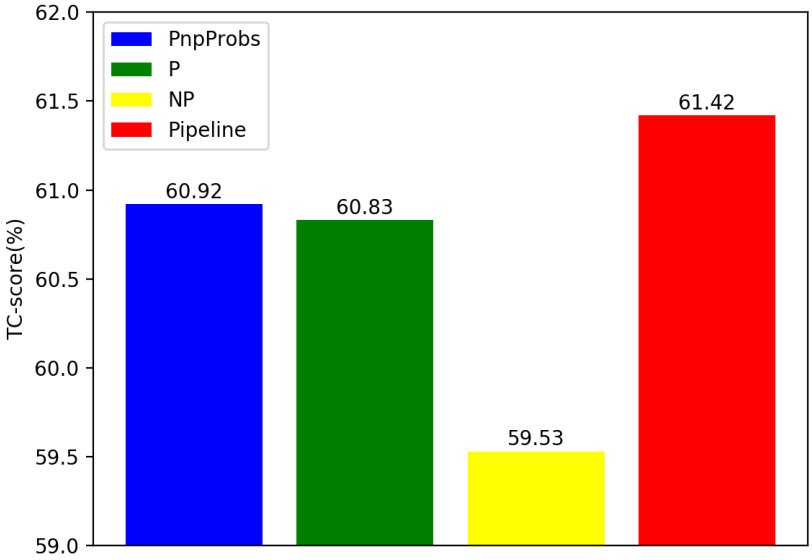

Figure 4 Average TC-scores on four empirical benchmarks (BAliBASE, OXBench, OXBench-X, SABMark) of PnpProbs, `P`, `NP` and the pipeline $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$

It is interesting to find out how well their combination performs. Thus, we implement and test the pipeline $\mathscr{P}^{\mathtt{Ral}}_{\mathtt{Q}} \circ \mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$, which, given any input family $\mathscr{F}$, first uses $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{P,NP}}$ to construct an alignment of $\mathscr{F}$, and then uses $\mathscr{P}^{\mathtt{Ral}}_{\mathtt{Q}}$ to improve this alignment. We call this pipeline MLProbs.

2.6 Comparing MLProbs with other MSA tools

This section compares the performance of MLProbs and ten other popular MSA tools, including PnpProbs, QuickProbs, GLProbs, PicXAA, ProbCons, MSAProbs, MAFFT, Muscle, ClustalΩ and ProbAlign.

2.6.1 Accuracy results

We compare the accuracy of their alignments for families in the four benchmark databases SABMark, OXBench, OXBench-X and BaliBASE.

Table 8 shows the accuracy of the alignments constructed by the tools for all families in SABMark, as well as those in its two subsets, the Superfamily and the Twilight Zone subsets. Superfamily contains different SCOP superfamilies with PID no more than 50%, and Twilight Zone contains different SCOP

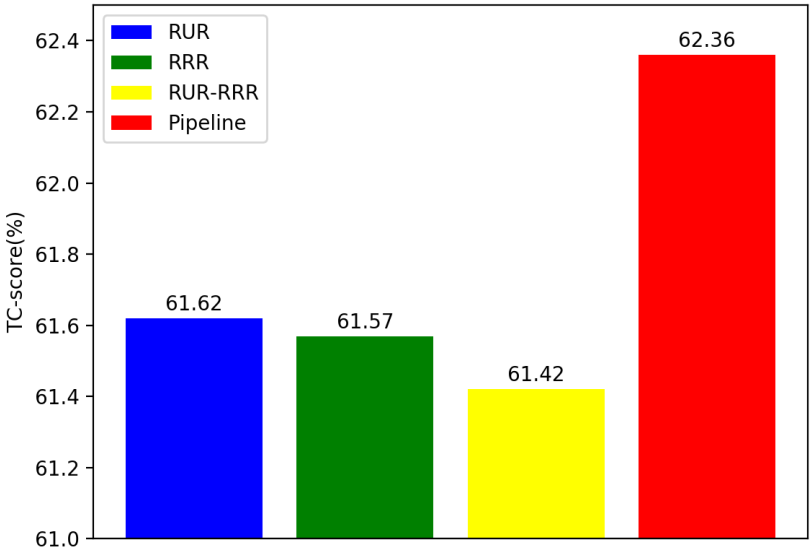

Figure 5 Average TC-scores on four empirical benchmarks (BAliBASE, OXBench, OXBench-X, SABMark) of using QuickProbs to realign unreliable regions (RUR), using QuickProbs to realign reliable regions (RRR) their simple combination and the pipeline $\mathscr{P}^{\texttt{Ral}}_{\texttt{Q}}$

subsets with PID no more than 25%. Besides TC scores, we have also compared the SP scores of the alignments, which also measure the goodness of alignments, though are less commonly used than TC scores. For all three sets of families no tools can obtained TC scores greater than 50; it is very difficult to construct good MSA for them. Note that MLProbs has the best TC and SP scores in all cases. The last row of Table 8 shows the improved percentage of MLProbs' scores over the second best scores. MLProbs' improved percentage of the TC score is 2.65% for all the families in SABMark, and is 3.40% for Superfamily. Even for Twilight Zone, whose families are notoriously very difficult to align, MLProbs has an improved percentage of 1.26%.

	All		Superfamily		Twilight Zone	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**42.58**	**62.10**	**48.63**	**67.92**	**24.94**	**44.78**
QuickProbs	40.65	61.05	46.56	66.86	23.40	**44.11**
PnpProbs	**41.48**	**61.25**	**47.03**	66.84	**24.63**	43.94
GLProbs	41.22	61.30	46.95	**66.97**	23.86	43.74
MSAProbs	40.04	60.24	45.81	66.00	22.60	42.46
ProbCons	39.17	59.69	44.64	65.27	22.57	42.42
PicXAA	38.44	59.06	44.45	65.18	20.33	40.23
MAFFT	33.00	53.15	39.10	60.05	14.72	32.14
Muscle	33.47	54.51	39.13	61.30	16.96	34.70
ClustalΩ	35.47	55.02	41.43	61.70	18.10	35.55
ProbAlign	38.63	59.53	44.11	65.40	22.64	42.43
Improved %	2.65%	1.38%	3.40%	1.42%	1.26%	1.52%

Table 8 Average TC-scores and SP-scores for SABMark (Chapter 2)

Table 9 shows the results for BAliBASE, as well as those for its two subsets RV11 and RV12. RV11 contains families with PID smaller than 20% and RV12 contains those greater than 20%. Note that for RV12, seven out of the eleven tools can already construct good alignments for its families; they all have TC scores greater than 85. However, for RV11, the TC scores of all tools are smaller than 50, and the 2.5% improvement of MLProbs is significant.

We now consider the benchmark database OXBench, and its extension OXBench-X. Both databases contain the same set of 395 families, but in OXBench-X, many new sequences have been added to the families and thus their sizes are much larger. In fact, the average size of the families in OXBench is 8.33, while

	All		RV11		RV12	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**65.80**	**89.50**	**48.14**	**70.33**	**87.64**	**94.79**
QuickProbs	**65.41**	**89.41**	**46.93**	**69.59**	87.03	94.52
PnpProbs	62.46	88.75	45.15	68.80	87.25	94.79
GLProbs	62.09	88.84	44.68	69.27	**87.38**	**94.83**
MSAProbs	64.51	89.09	44.40	68.18	87.03	94.63
ProbCons	61.89	88.31	40.89	65.26	84.14	92.03
PicXAA	59.97	87.84	46.64	68.98	86.60	94.61
MAFFT	50.08	82.24	28.23	52.54	75.57	88.17
Muscle	53.17	84.33	32.06	57.15	58.90	90.26
ClustalΩ	56.20	83.97	36.22	59.01	79.38	90.60
ProbAlign	60.68	87.78	45.69	69.50	86.69	94.64
Improved %	0.60%	0.1%	2.5%	1%	0.3%	-0.04%

Table 9 Average TC-scores and SP-scores for BAliBASE (Chapter 2)

it is 122.49 for OXBench-X.

Table 10 shows our results for OXBench. Besides the complete set of families, the table also shows the alignment accuracy for families with PID no less than 30%, and for those above 30%. Note from the table that all tools can construct very good alignments for families with high similarity, but for those with PID smaller than 30%, only MLProbs has a TC score greater than 45, and its improved percentage over the 2nd best tool is 2.7%.

The results for OXBench-X are quite different. From Table 11 we note that no tools can construct satisfactory alignments even for families with high similarity. It is not surprising because the families in OXBench-X is much larger. Even though MLProbs can still obtain the best TC scores, its improvement is not as significant as we have seen in the other benchmarks. A reason might be because BENCH, the dataset that we use to train MLProbs, has an average family size 34.79, which is much smaller than that of OXBench-X (whose average family size is 122.49).

For all four databases, MLProbs consistently achieves the highest TC scores among all tools. More importantly, MLProbs performs particularly well for families with low similarity. To give an overall picture, Figure 6 compares the TC scores of the tools over all families in SABMark, BAliBASE, OXBench

	All		0 - 30%		30% - 100%	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**82.72**	**90.57**	**45.13**	**66.56**	**89.86**	**95.12**
QuickProbs	81.77	90.17	41.50	64.50	88.35	94.46
PnpProbs	**82.06**	**90.19**	**43.96**	**66.33**	89.54	**95.01**
GLProbs	81.93	90.13	43.34	66.11	**89.55**	94.99
MSAProbs	81.50	89.83	42.81	65.22	89.08	94.78
ProbCons	80.68	89.45	41.50	64.50	88.35	94.46
PicXAA	81.14	89.61	39.78	63.16	89.32	94.93
MAFFT	78.15	88.07	35.93	60.68	86.40	93.53
Muscle	80.67	89.50	40.95	63.96	88.21	94.34
Clustal**Ω**	79.99	88.91	37.39	60.78	88.08	94.25
ProbAlign	81.68	89.97	41.06	63.80	89.39	94.93
Improved %	0.80%	0.42%	2.66%	0.35%	0.35%	0.12%

Table 10 Average TC-scores and SP-scores for OXBench (Chapter 2)

	All		0 - 30%		30% - 100%	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**59.80**	**66.14**	**41.68**	**51.11**	**68.49**	73.34
QuickProbs	59.44	65.86	40.72	**50.74**	68.41	73.11
PnpProbs	**59.54**	**65.95**	40.97	50.56	**68.44**	73.32
GLProbs	59.34	65.81	**41.00**	50.53	68.14	73.14
MSAProbs	59.37	65.82	40.60	50.39	68.37	73.22
ProbCons	58.93	65.62	39.41	49.43	68.29	**73.39**
PicXAA	58.85	65.39	39.00	49.19	68.37	73.15
MAFFT	56.90	64.20	37.22	47.25	66.33	72.32
Muscle	56.83	64.39	36.32	47.70	66.66	72.40
Clustal**Ω**	58.05	64.81	39.10	48.67	67.14	72.55
ProbAlign	59.27	65.71	39.80	49.65	68.60	**73.41**
Improved %	0.43%	0.29%	1.66%	0.73%	0.05%	-

Table 11 Average TC-scores and SP-scores for OXBench-X (Chapter 2)

and OXBench-X with PID $\leq 50\%$ (there are totally 1356 such families). We note that MLProbs has the highest TC score, and its improvement over the 2nd best tool is more than triple of the 2nd best tool's improvement over the 3rd best tool. As we have mentioned earlier, obtaining better alignments for these families with low similarity is always a challenge for the research in MSA construction, and MLProbs' improvement would have a significant impact. MLProbs also get some improvements on high similarity protein families in average TC-score from Figure 7.

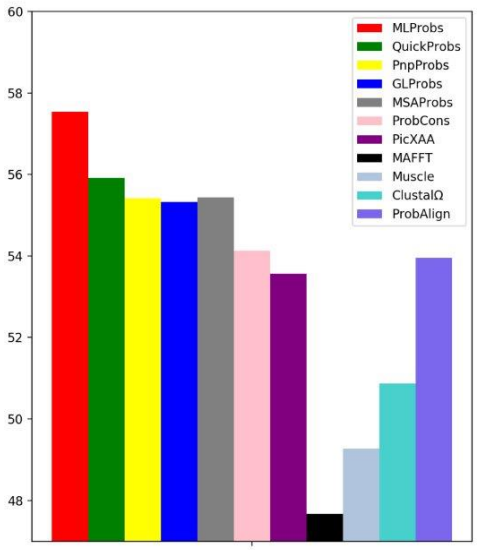

Figure 6 Over 1356 families in BAliBASE, OXBench, OXBench-X and SABmark with PID $\leq 50\%$. MLProbs gets the best average TC-score 57.54 and improves about 2.9% over the second best one which is 55.41.

2.6.2 Efficiency results

We compare, for each of the four benchmark databases, the average running time of MLProbs and other MSA tools for constructing an alignment. All the tools run on a Dell desktop computer with four Intel Cores i5-6500 (3.20GHz) and main memory of size 7.6GB. Table 12 shows the results.

Note that the running time of MLProbs is mainly composed of three parts: the time to get an alignment using PnpProbs, the time to realign some regions using QuickProbs, and the running times of the classifiers. The total running

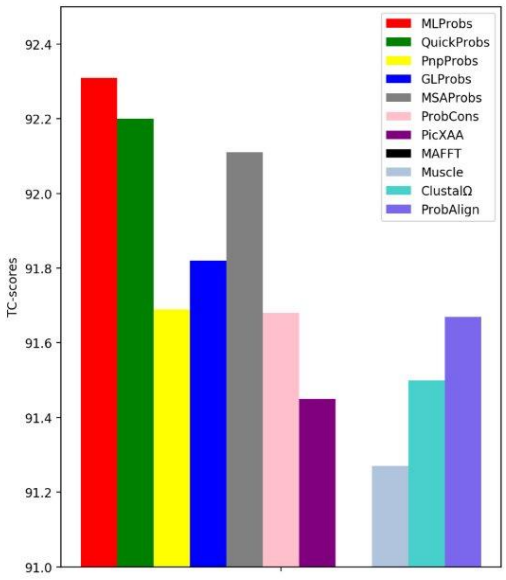

Figure 7 Over 243 families in BAliBASE, OXBench, OXBench-X and SABmark with PID $> 50\%$. All the MSA tools could achieve more than 91.00 on average TC-score.

time of the classifiers are negligible (as can be corroborated by the column for BAliBASE in Table 12, which shows the running time of MLProbs roughly equals the sum of those of PnpProbs and QuickProbs). We note that the differences between the running time of MLProbs and the sum of that of PnpProbs and QuickProbs are positive for OXBench and SABMark, but is negative for OXBench-X. This is because for families in OXBench and SABMark, we usually need to realign only a few small regions, while for those in OXBench-X, we need to realign many large regions. This is not surprising because the sizes of the families in OXBench-X are very large, and thus those constructed by PnpProbs are not very reliable.

2.7 Applications of MLProbs

2.7.1 Phylogenetic tree construction

A popular way to construct a phylogenetic tree for a protein family is first to construct an MSA for the family, and then convert it to a phylogenetic tree. As remark in [75], the quality of the constructed phylogenies depends much on the accuracy of the MSAs. Since our experiments show that MLProbs has the

	BAliBASE	OXBench	OXBench-X	SABMark
MLProbs	16.7325	0.8689	43.4129	0.8171
QuickProbs	5.8985	0.1053	15.9758	0.0604
PnpProbs	11.2323	0.3065	31.7131	0.2106
GLProbs	12.0890	0.2808	32.1543	0.1492
MSAProbs	8.6230	0.1404	30.7912	0.0738
ProbCons	27.2583	0.4576	96.3181	0.2263
PicXAA	23.5338	0.4050	106.6262	0.2973
MAFFT	0.3191	0.1226	0.2345	0.1056
Muscle	1.5939	0.0339	2.7322	0.0715
ClustalΩ	1.1442	0.0313	0.8776	0.0466
ProbAlign	16.1930	0.2618	73.7960	0.1112

Table 12 Average running time (in seconds) for constructing an MSA

best alignment accuracy among the tools, we expect that the phylogenetic trees constructed from its alignments should be good. To verify this, we compare the quality of the phylogenetic trees constructed from the MSAs obtained by MLProbs, QuickProbs, PnpProbs, GLProbs, MSAProbs, ProbCons and PicXAA for families in TreeFam [76]. The phylogenetic trees are constructed as follows.

> For each tool `U` and each family $\mathscr{F}$, we construct an MSA $\mathscr{M}$ for $\mathscr{F}$ using `U`. Then, we use the phylogenetic tree construction tool MEGA X [77] to construct a phylogenetic tree from $\mathscr{M}$.

We measure the quality of a phylogenetic tree by its unweighted Robinson-Foulds (RF) distance [78] between the tree and the reference tree given in TreeFam; the smaller the distance the better the tree. We use the package DendroPy [79] to compute the distances. Table 13 summaries our results. We note that for all but one family, namely TF105311, the phylogenetic trees constructed by MLProbs's alignments have the smallest unweighted RF distance, and for TF105311, MLProbs's tree has the second smallest distance, and it differs from the best one by only 2.

Figures 8 and 9 show the phylogenetic tree of TF105063 constructed by MLProbs and the reference.

TreeFam ID	MLProbs	QuickProbs	PnpProbs	GLProbs	MSAProbs	ProbCons	PicXAA
TF105063	**128**	130	132	132	134	140	130
TF105073	**112**	112	112	112	114	116	124
TF105311	94	**92**	94	94	94	92	102
TF105313	**18**	22	18	18	22	22	18
TF105629	**108**	112	110	110	108	114	120
TF105801	**140**	140	140	140	144	150	148
TF313227	**208**	212	228	228	230	224	220

Table 13 The unweighted RF-distances for the phylogenetic trees constructed

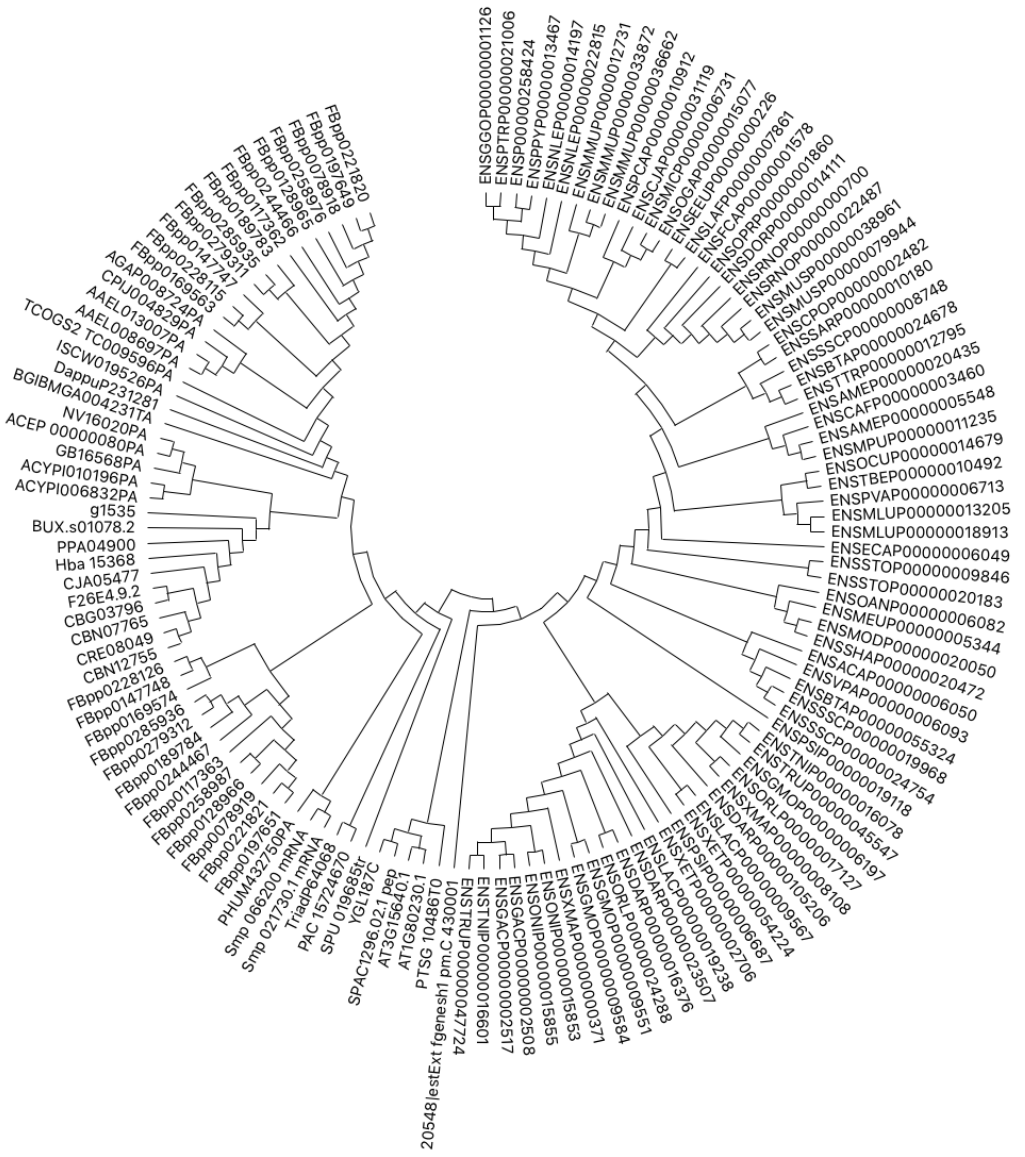

Figure 8 The phylogenetic tree of TF105063 constructed by MLProbs.

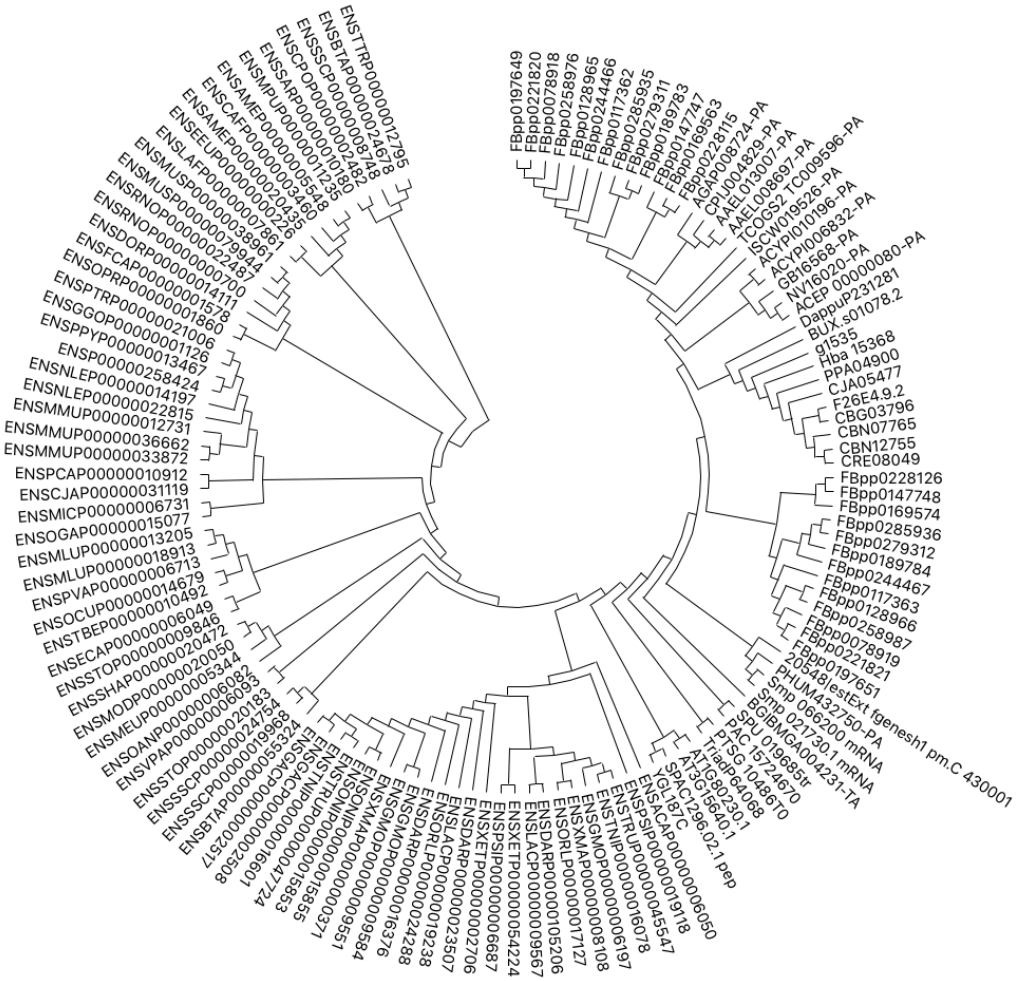

Figure 9 The reference phylogenetic tree of TF105063.

2.7.2 Protein secondary structure prediction

Another common application of MSA construction is to predict the secondary structure of proteins [80], and again the quality of the MSAs affects accuracy of the predictions. We use MLProbs and other MSA tools to predict the secondary structure of the following protein sequences, namely 1U24 [81], 6HN6 [82], 5TFD [83], 5DFD [84], 2GDF, 3GDF [85] and 9GAF [86] as follows.

> Given a protein sequence s, we use Jpred 4 [87] to search protein sequences similar to this sequence. Then, we construct an MSA $\mathscr{M}$ for these sequences and s, and then use the secondary structure prediction tool provided on Jpred 4 to predict the secondary structure of s.

Table 14 shows the number of wrongly aligned residues made by various tools. We note that in all cases, the numbers of MLProbs's wrongly aligned residues are the smallest.

PDB ID	MLProbs	QuickProbs	PnpProbs	GLProbs	MSAProbs	ProbCons	MAFFT
1U24	1	1	1	2	2	4	5
5TFD	6	6	6	6	7	6	7
6HN6	11	12	23	22	19	13	24
5DFD	2	2	2	6	7	4	2
2GDF	50	50	52	52	50	60	53
3GDF	4	5	6	6	5	7	8
9GAF	10	12	10	14	13	14	13

Table 14 Number of wrongly aligned residues in the predicted secondary structures (Chapter 2)

2.8 Conclusions

This research explores using the data-centric approach to improve the accuracy of MSA construction. We have identified three classification problems that may help improve alignment construction, and used the shadow machine learning algorithm Random Forest to train models for them. Then, we build a pipeline line MLProbs that make use of these models to help construct MSAs, and empirical evaluation shows that MLProbs' alignment accuracy is significantly better than many popular MSA tools.

The efficiency test shows that the running time of MLProbs is acceptable because the running time of classifiers is negligible.

MLProbs can achieve good results in two well-known high-level real-life applications.

An interesting question is whether we can make further improvement if we use deep learning algorithms for the trainings. We propose some research directions related to this in the Chapter 4.

Chapter 3

DLPAlign: A Deep Learning based Progressive Alignment for Multiple Protein Sequences

3.1 Overview

After using machine-learning methods to get good results in the selection of high-level MSA tools, our next step was to examine whether machine-learning or deep-learning algorithms could achieve better results in the specific progressive alignment strategy at a low level.

A general progressive alignment includes the steps shown in Figure 10.

The most popular MSA tools adopting a progressive strategy in the past 10 years are as follows:

(1) ProbCons uses a pair-hidden Markov model (HMM) to calculate the posterior probability matrix, an unweighted probabilistic consistency transformation, using an unweighted pair group method with the arithmetic mean (UPGMA) hierarchical clustering method to generate a guide tree and iterative refinement to construct an MSA.

(2) Probalign, another popular, highly accurate MSA tool, uses a partition function instead of the ProbCons pair HMM to calculate the posterior probability matrix.

(3) MSAProbs combines (1) and (2), using the Root Mean Square (RMS) of pair HMM and the partition function as the calculation method for the posterior probability matrix, and adopts a weighted consistency transformation.

(4) GLProbs introduced random HMM, and adaptively uses the partition function, global pair HMM, the RMS of global pair HMM, and random HMM to calculate the posterior probability matrix using the different average pairwise percentage identity (PID) of each protein family. PID stands for the percentage of the number of homologous positions in the pairwise alignment of two sequences.

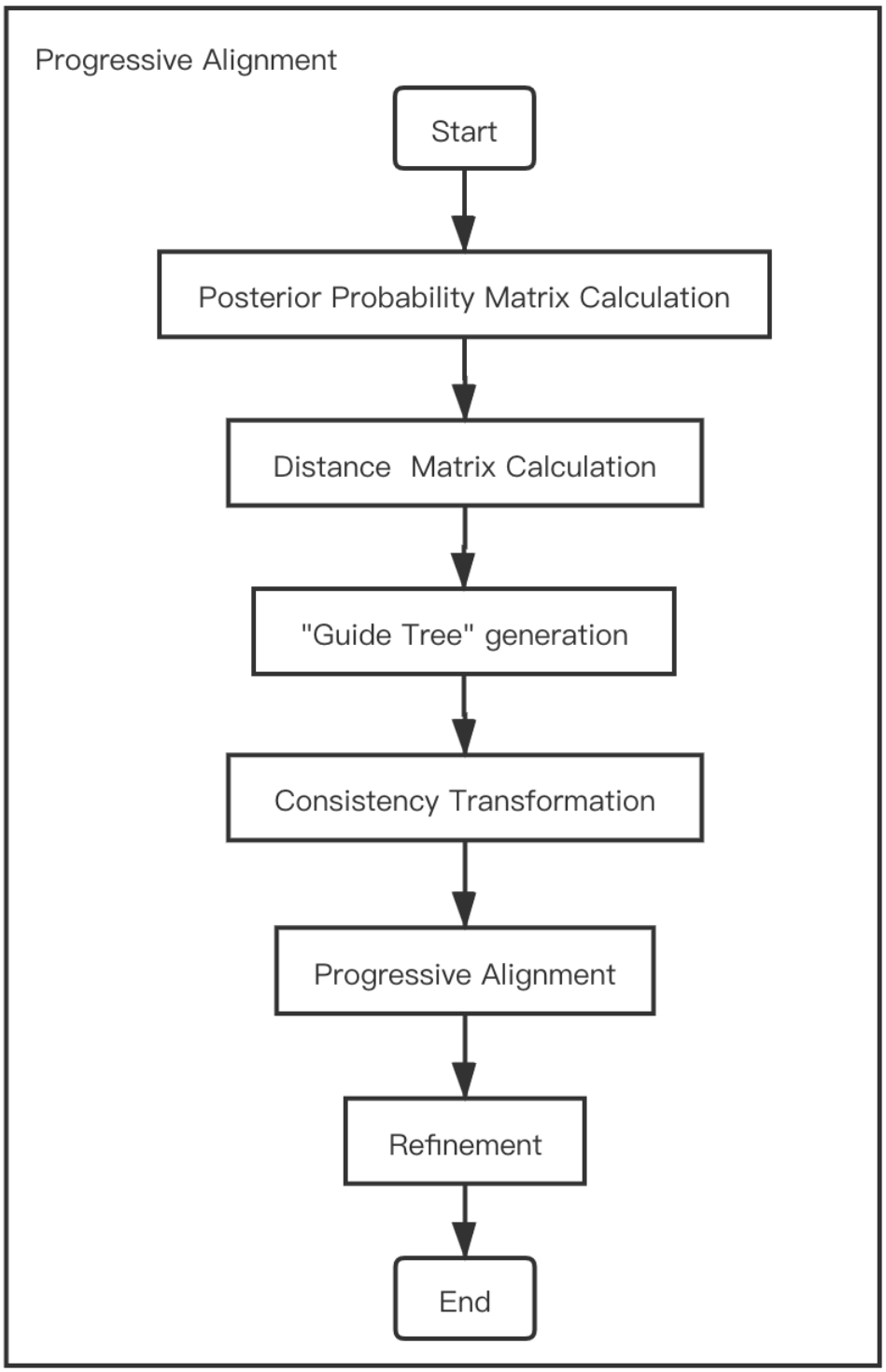

Figure 10 The processes of a general progressive alignment.

(5) PnpProbs applies UPGMA and the weighted pair group method with arithmetical mean (WPGMA) [88] adaptively to generate the guide tree in its progressive branch.

Although these MSA tools can achieve relatively high accuracy on the whole, when it comes to a specific protein family, the accuracy of the different tools is often different. Most importantly, after so many years of research, the algorithm-centric method has not significantly improved accuracy. As for the progressive alignment strategy, protein families with low similarity have always been the most challenging part.

In this part of the thesis, we first determined which specific part in the progressive MSA methods could provide the greatest improvement in accuracy. Then we used deep learning to train decision-making models to help us choose the most appropriate calculation method for each part.

Based on the most accurate decision-making model, we built a new progressive MSA tool, called DLPAlign. Then we compared the performance of DLPAlign with 10 other popular MSA tools on three empirical benchmarks and a real-life application. The findings showed that DLPAlign was the most accurate on all tests.

3.2 Selection of promotion parts

As we mentioned in Section 3.1, a typical progressive alignment method consists of the following steps: posterior probability matrix calculation, distance matrix calculation, "Guide Tree" generation by clustering methods, consistency transformation and refinement. The most studied of these steps are the posterior probability matrix calculation, clustering methods for generating a guide tree, and consistency transformation. We chose these three parts as our candidate promotion parts and refer to the posterior probability matrix calculation as **Part A**, guide tree generation as **Part B**, and consistency transformation as **Part C**. For each part, we extracted several candidate options from previous studies, as shown below:

Options for Part A:

1. Pair-HMM
2. Partition function
3. the RMS of pair-HMM and partition function

4. the RMS of pair-HMM, partition function and random HMM

Options for Part B:

1. UPGMA
2. WPGMA

Options for Part C:

1. Unweighted consistency transformation
2. Weighted consistency transformation

When we test a single part, we need to use the same methods for the other parts. We used the calculation method of each part numbered (1) as the default method for that part. For the other parts that we did not select, such as the distance matrix calculation and refinement, we adopted the same implementation method as in GLProbs.

To evaluate the advantages and disadvantages of several methods in Part A and to what extent they could be improved, we got four different pipelines by using different calculation methods of the posterior probability matrix in the GLProbs' code and implemented the calculation in Parts B and C by default, naming them $\mathscr{P}_A^i, i = 1,2,3,4$, which respectively represents the different options of Part A mentioned above. We implemented $\mathscr{P}_B^i, i = 1,2$, where i denoted the different clustering methods for guide tree generation in Part B, and $\mathscr{P}_C^i, i = 1,2$, where i denoted the different calculation of consistency transformation in Part C in the same way.

We used the TC-score as our judgment standard in the following comparison. In the evaluation part, we used only the famous BAliBASE, OXBench, and SABmark benchmarks as the evaluation materials in this section.

Table 15 summarizes for each MSA tool in $\mathscr{P}_A^i$, $\mathscr{P}_B^i$ and $\mathscr{P}_C^i$ and each benchmark database the average TC scores of the alignments constructed by the MSA tool for the families in the database.

The critical concern is the upper bounds of the various calculations in different parts of the progressive alignment strategy, and in which part the maximum

[1]Upper Bound: The highest TC-score could be obtained when each family chooses the method which could get the best TC-score in this part.

[2]Max. Improvement: The proportion that the upper bound can improve compared with the best-existed result of this part.

	BAliBASE	OXBench	SABMark
$\mathscr{P}_A^1$	62.17	80.95	39.22
$\mathscr{P}_A^2$	61.06	81.73	39.00
$\mathscr{P}_A^3$	64.42	81.57	40.11
$\mathscr{P}_A^4$	64.27	81.56	40.62
Upper Bound [1]	66.50	82.93	42.84
Max. Improvement [2]	**3.23%**	**1.47%**	**5.47%**
$\mathscr{P}_B^1$	62.17	80.95	39.22
$\mathscr{P}_B^2$	62.42	80.93	39.20
Upper Bound	62.74	80.96	39.28
Max. Improvement	0.51%	0.01%	0.15%
$\mathscr{P}_C^1$	62.17	80.95	39.22
$\mathscr{P}_C^2$	62.95	81.00	39.09
Upper Bound	63.65	81.21	39.69
Max. Improvement	1.11%	0.26%	1.20%

Table 15 Average TC-scores of each tool on the three empirical benchmark databases

improvement can be made. Table 15 shows that if a particular decision is used in Part A to assist in selecting different calculation methods, the theoretical maximum promotion proportion can be obtained. So next, we chose the right decision-making method for pipelines $\mathscr{P}_A$.

3.3 Deep-learning-based decision-making method

In $\mathscr{P}_A^i$ of the progressive alignment strategy, for a specific protein family, $\mathscr{F}$, choosing the method with the highest accuracy on which to build MSA $\mathscr{M}$ can be expressed as the classification problem $\mathscr{C}_{\mathscr{P}_A^i}^{Aln}$, where i denotes the different methods described in the above Section: 3.2.

Classification $\mathscr{C}_{\mathscr{P}_A^i}^{Aln}$ of a protein family is defined as follows:

> $\mathscr{C}_{\mathscr{P}_A^i}^{Aln}$ has four classes $\mathscr{P}_A^1$, $\mathscr{P}_A^2$, $\mathscr{P}_A^3$ and $\mathscr{P}_A^4$. A protein family $\mathscr{F}$ is in class $\mathscr{P}_A^i$ if the MSA constructed by $\mathscr{P}_A^i$ could get better TC-score than those constructed by others, where $i = 1, 2, 3$ or 4.

Data augmentation

In the past few years, there have been significant developments in deep learning, which have been applied in bioinformatics [46], mainly due to the continuous expansion of the data scale. It is not sufficient to use just 2,000 protein families or 20,000 sequences to train deep-learning models, so we considered coupling any two sequences in the same protein family as an independent piece of data. If a family has n sequences, we can get $C_n^2 = \frac{n \times (n-1)}{2}$ sequence pairs from it. In this way, our data was expanded to 954,854 pairs.

The structure of candidate deep-learning models

Because the length of the sequence pairs was not very consistent, we normalized the length before choosing the neural networks. We unified all pairs into a fixed length α ($\alpha = 512$ in our structure). When the length of a pair was insufficient, it was filled by gaps at the end to increase the length to α. When the length of the sequence exceeded α, only the leading fragments of length α were intercepted.

When we regard each character as a single word, if we convert it into a one-hot-word vector, the size of the vector is a little large, so we first used the word-embedding [89] technique to convert each word into a small size (eight-dimensional vector).

Even so, our input scale was still relatively large, so we applied convolutional neural networks (CNNs), which had made a significant breakthrough in computer vision to reduce the dimensionality of the data while retaining its characteristics, as the first two layers of our models.

There were order relationships between every character in a protein sequence pair, so we added a recurrent neural network (RNN) [90] layer after the CNNs. The improved versions of the recurrent neural network, long short-term memory network (LSTM) [91] and gated recurrent unit network (GRU) [92], and their bi-directional versions (BiLSTM, BiGRU) have many advantages, so they were alternatives.

Subsequently, two full connection layers were connected. To reduce overfitting, we added a specific dropout rate to the first full connection layer.

This kind of neural network structure is very suitable and widely used for classification tasks [93], [94], [95], [96].

Model accuracy measurement

We implemented the network structure mentioned in Section 3.3 and named it CNN, CNN + RNN, CNN + LSTM, CNN + BiLSTM, CNN + GRU and CNN + BiGRU, according to the different recurrent neural network layers used.

We divided the collected pair data into two parts: (1) 80% was randomly selected for model training, and (2) the remaining 20% was used for final testing. In the training process, a five-fold cross-validation was performed. This kind of cross-validation method proved to be the most efficient [97]. In the process of training, we also set early stopping to further reduce overfitting.

Table 16 shows the macro average (averaging the unweighted mean per label) [98] of the precision, recall and F_1-score of the multi-class labels in the 20% test data.

If $\mathscr{P}_i$ is the precision and $\mathscr{R}_i$ is the recall rate of class i, where $i = 1, 2, 3$ or 4 in this paper, the macro average precision ($\mathscr{P}_{macro}$), recall ($\mathscr{R}_{macro}$) can be calculated by Formula (1).

$$\begin{aligned} \mathscr{P}_{macro} &= \frac{1}{N}\sum_{i=1}^{N} P_i \\ \mathscr{R}_{macro} &= \frac{1}{N}\sum_{i=1}^{N} R_i \end{aligned} \tag{1}$$

where $N = 4$ in this paper, which means there are 4 categories.

The macro average F_1-score ($\mathscr{F}_{macro}$) is equal to the harmonic average of $\mathscr{P}_{macro}$ and $\mathscr{R}_{macro}$.

According to the F_1-score in Table 16, we decided to use CNN + BiLSTM as our decision-making model.

Table 17 shows in more detail the precision, recall, and F_1-score in the four different categories of the CNN + BiLSTM model we finally selected.

Models	$\mathscr{P}_{macro}$	$\mathscr{R}_{macro}$	$\mathscr{F}_{macro}$
CNN	87.13	86.69	86.91
CNN + RNN	86.90	88.13	87.51
CNN + LSTM	87.13	87.71	87.41
CNN + BiLSTM	87.70	88.68	**88.19**
CNN + GRU	87.93	86.71	87.32
CNN + BiGRU	88.23	87.89	88.06

Table 16 The macro average precision, recall and F_1-score on the test data.

Category	Precision	Recall	F_1-score
$\mathscr{P}_A^1$	82.97	96.90	89.39
$\mathscr{P}_A^2$	79.46	92.83	85.62
$\mathscr{P}_A^3$	95.44	77.81	85.73
$\mathscr{P}_A^4$	92.93	87.18	89.96
Macro Avg.	87.70	88.68	88.19

Table 17 The precision, recall and F_1-score on the test data in different categories by CNN + BiLSTM structure.

Although there is a difference in the precision or recall rate among the different categories, the F_1-score of each category is generally good; they were all over 85%, indicating that our model can handle all types well.

Figure 11 further shows the confusion matrix obtained using CNN + BiLSTM as the decision-making model to predict the four categories $\mathscr{P}_A^i$, where $i = 1$, 2, 3 or 4. The confusion matrix is a table that is often used to describe the performance of a classifier on a set of test data for which the true values are known.

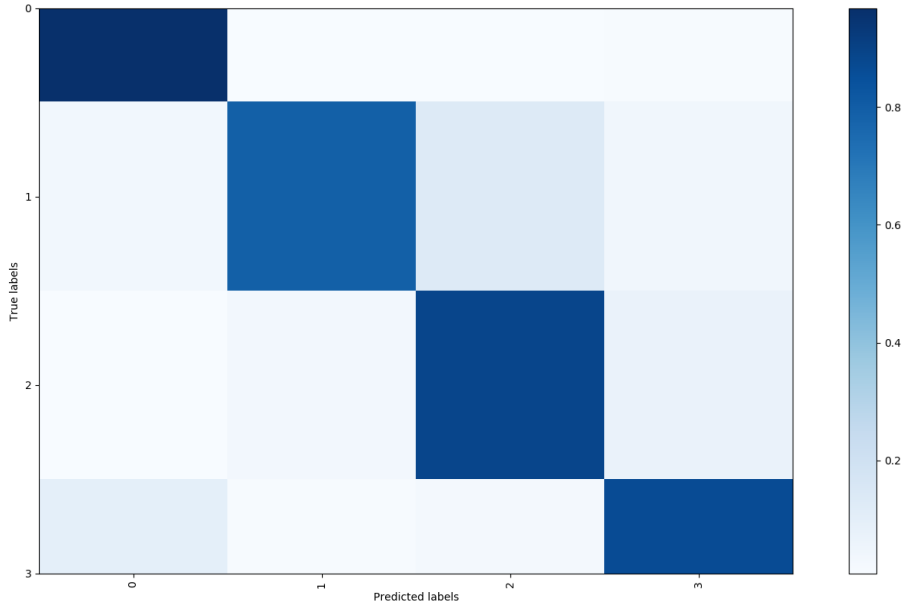

Figure 11 The confusion matrix of the decision-making model trained by CNN + BiLSTM. *The darker the color of the grid of the predicted label and the corresponding true label, the higher the accuracy of the prediction of this category (category $\mathscr{P}_A^i$ is represented by the label $i-1$). The color depth of all four categories exceeds 0.8, indicating that the classification accuracy of each category is higher than 80%. This result also corresponds to Table 17.*

The structure of the model we used is shown in Figure 12.

3.4 Decision-making model-based progressive alignment method

Given the high accuracy of our decision-making model (CNN + BiLSTM above), we integrated it into existing progressive alignment methods to con-

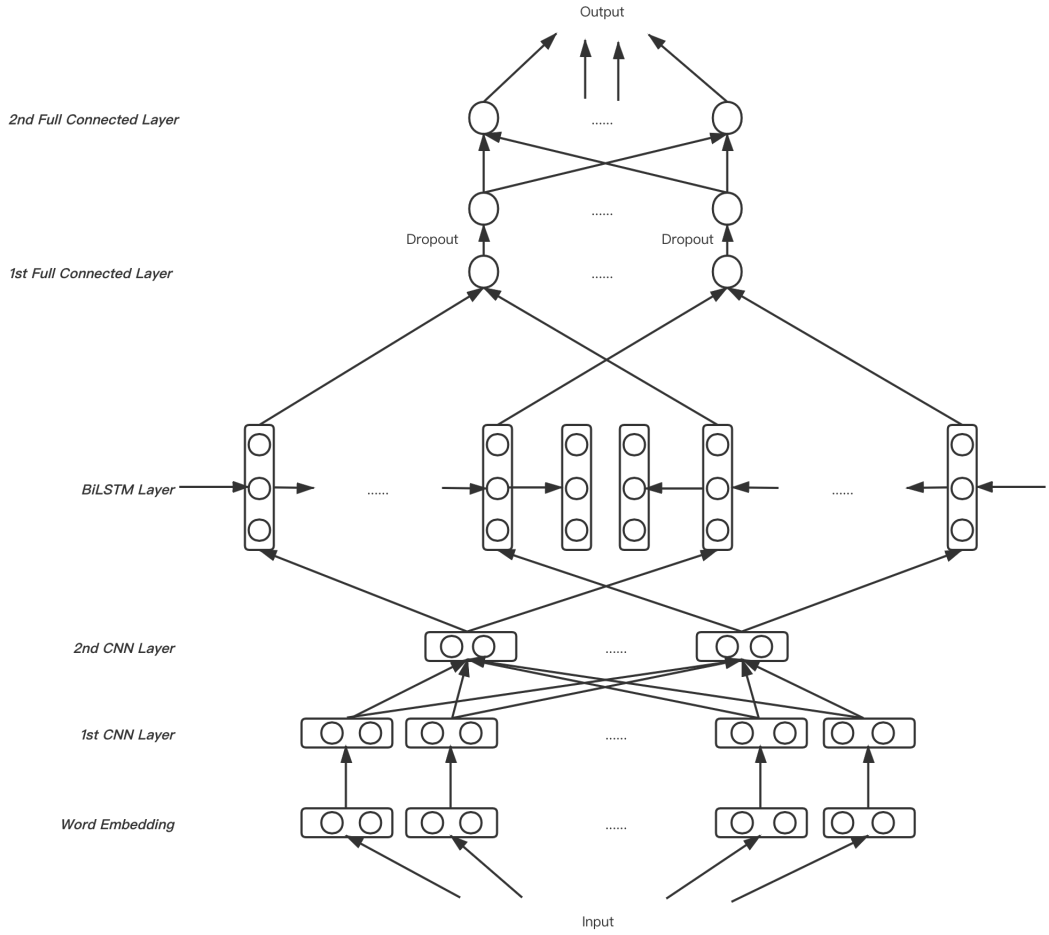

Figure 12 The neural network structure of our decision-making model. *The input can be a protein pair of any length. Firstly the input is transformed through the Word embedding layer into a 512 × 8 matrix. Then the matrix passes through two CNN layers with filter sizes of 6 and 3 respectively (each CNN layer is followed by a max-pooling layer of size 2). Next, the output of the previous layer goes through the Bi-directional LSTM layer with a hidden size of 64. Finally, two full connection layers are connected and a 0.5 dropout to the first full connection layer is set. The output is a* 4×1 *matrix that represents the final category.*

struct a new progressive alignment for multiple protein sequences, which we named DLPAlign. Given a protein family $\mathscr{F}$, DLPAlign produces a multiple sequence alignment using mainly the following steps..

Decision-making of the posterior probability matrix calculation

Each pair x, y in $\mathscr{F}$ is inseted into the model with the highest accuracy described in the previous section to get a label (say, $label_{x,y}$). These labels represent the specific calculation method of the posterior probability matrix that should be used. Which calculation method is chosen for the protein family $\mathscr{F}$ is determined by the dominant proportion of labels that all of its pairs get after passing through the decision-making model.

Because we already know the percentage of each correct label of $\mathscr{C}^{Aln}_{\mathscr{P}^i_A}$, where $i = 1$, 2, 3 or 4, the dominate proportion of predicted labels can be calculated using the following Formula (2).

$$\text{dominate\_proportion} = \text{argmax}_i(\frac{PLP_i}{TLP_i}) \tag{2}$$

where PLP_i means the percentage of the i^{th} predicted label, TLP_i means the proportion of the i^{th} true label and $i = 1$, 2, 3 or 4.

This process is shown in Figure 13.

Depending on the final family category, we use (1) pair HMM, (2) the partition function, (3) the RMS of pair HMM and the partition function, or (4) the RMS of pair HMM, the partition function, and random HMM to accomplish the calculation of the posterior probability matrix.

Distance matrix and guide tree generation

By finding the maximum summing path (or maximum weight trace) through the posterior probability matrix, a pairwise alignment is computed on every pair x, y in $\mathscr{F}$ and the maximum sum is saved as *Prob(x, y)*. The distance between sequences x and y can be measured by Formula (3).

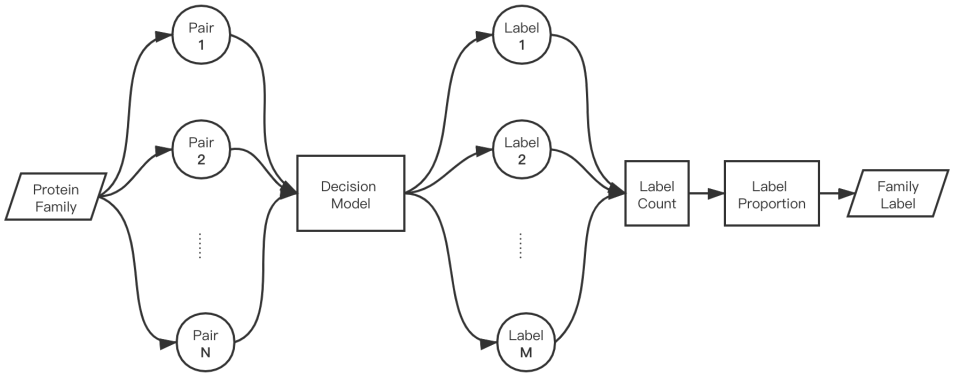

Figure 13 The process of splitting a protein family into pairs and using decision-making model to determine the label of each pair, and finally calculating the mode to get the family label

$$Distance[x][y] = 1 - \frac{Prob(x,y)}{min(x.Length,y.Length)} \tag{3}$$

where *x.Length*, *y.Length* represent the length of sequences x, y respectively.

A guide tree is a data structure used to determine the relationship between (1) sequence and sequence, (2) sequence and profile, and (3) profile and profile. We define two clusters, A and B. The distance between their union and another cluster, C, can be expressed as Formula (4), which is also the specific implementation of UPGMA.

$$Distance[\mathrm{A} \cup \mathrm{B}][\mathrm{C}] = \frac{|A| \times Distance[A][C] + |B| \times Distance[A][C]}{|A| + |B|} \tag{4}$$

where $|A|$, $|B|$ and $|C|$ represent the weight of clusters A, B and C.

According to this distance calculation formula and the distance matrix computed, we can start from the sequences of the minimum distance and gradually build a binary tree, named "Guide Tree".

Figure 14 shows an example of a guide tree calculated using DLPAlign.

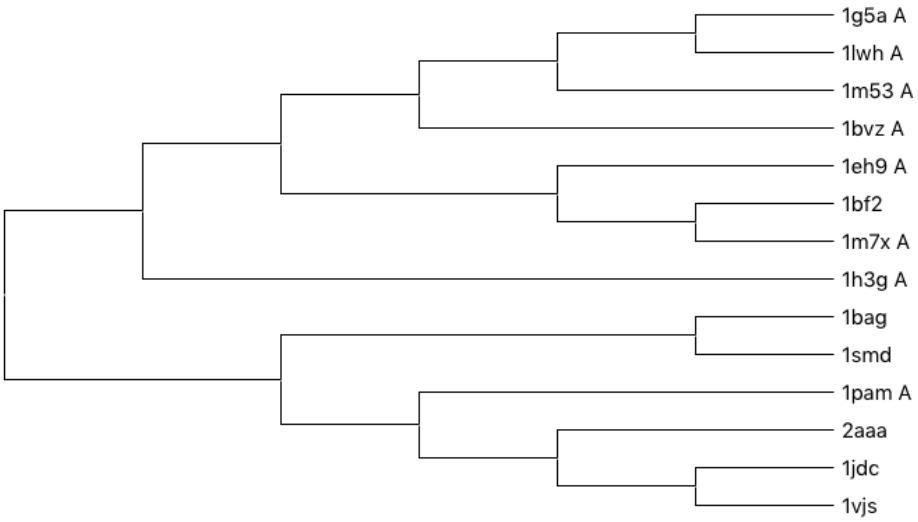

Figure 14 The guide tree of family "BB11018" in BAliBASE calculated by DLPAlign.

Consistency transformation

In this step, we use other sequences to relax the posterior probability matrix of every pair x and y (written as $P_{x,y}$), which we calculated in step 3.4 to determine the substitution scores for the following steps. The relaxation process can be expressed by Formula (5).

$$P'_{x,y} = \frac{1}{|S|} \left(2 \times P_{x,y} + \sum_{z \in S} P_{x,z} \times P_{z,y}\right) \tag{5}$$

where S stands for the sequences set in protein family $\mathscr{F}$, and $P'_{x,y}$ is the new transformed posterior matrix of pair $< x, y >$.

Progressive alignment

Based on the Guide Tree we determined in the Step 3.4 and the relaxed posterior probability matrix in Step 3.4, we can merge two child nodes (sequences) from the deepest node to get a profile, and then merge them to the root node of the respective tree to get a complete MSA containing all sequences in $\mathscr{F}$.

Figure 15 shows an example of an order of progressive alignment in DLPAlign.

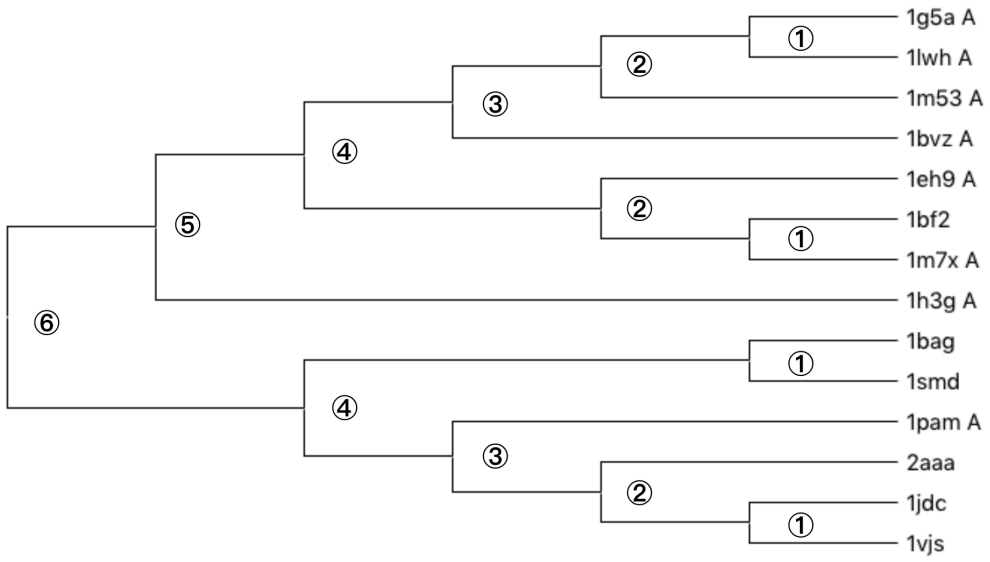

Figure 15 The order of progressive alignment in DLPAlign of family "BB11018" in BAliBASE.

Refinement

The purpose of refinement is to correct some errors that may have occurred in the alignment between previous sequences. In this specific implementation, we also used the iterative refinement step to divide all aligned sequences into two groups each time randomly and then used the profile-profile alignment to realign them again. However, we added accuracy judgment. Each refinement was valid only if the maximum sum described in Section 3.4 was larger than before.

3.5 Comparing DLPAlign with other MSA tools

3.5.1 Accuracy results

To determine the accuracy of DLPAlign implemented by the CNN + BiLSTM decision-making model and comparing this with other MSA tools with high

accuracy, three empirical benchmarks were selected –BAliBASE 3.0, OXBench 1.3 and SABmark 1.65 –and the newest versions of 10 popular MSA tools were chosen for comparison: QuickProbs, PnpProbs, GLProbs, MSAProbs, Probalign, ProbCons, PicXAA, MAFFT, MUSCLE and Clustal**Ω**. Of these 10 MSA tools, PicXAA adopted the non-progressive strategy, PnpProbs used both the non-progressive and progressive strategy, and the others used the progressive strategy. The TC-score and SP-score mentioned in Section 1.4 were the leading indicators in the comparison.

	All (386)		RV11 (38)		RV12 (44)	
	TC	SP	TC	SP	TC	SP
DLPAlign	**65.47**	**89.57**	**47.67**	**69.97**	87.32	**94.79**
MLProbs	**65.80**	**89.50**	**48.14**	**70.33**	**87.64**	**94.79**
QuickProbs	65.41	89.41	46.93	69.59	87.03	94.52
PnpProbs	62.46	88.75	45.15	68.80	87.25	**94.79**
GLProbs	62.09	88.84	44.68	69.27	**87.38**	**94.83**
MSAProbs	64.51	89.09	44.40	68.18	87.03	94.63
ProbCons	61.89	88.31	40.89	65.26	84.14	92.03
PicXAA	59.97	87.84	46.64	68.98	86.60	94.61
MAFFT	50.08	82.24	28.23	52.54	75.57	88.17
Muscle	53.17	84.33	32.06	57.15	58.90	90.26
Clustal**Ω**	56.20	83.97	36.22	59.01	79.38	90.60
ProbAlign	60.68	87.78	45.69	69.50	86.69	94.64

Table 18 Average TC-scores and SP-scores for BAliBASE (Chapter 3)

The alignments in BAliBASE were organized into reference sets that were designed to represent real multiple alignment problems. Table 18 shows the average TC-score of the whole benchmark with 386 families and the accuracy of its two divergent reference sets (say, RV11 and RV12). DLPAlign could also handle RV11, which is a very divergent subset, obtaining a 1.58% higher TC-score than the third-best MSA tool.

Table 19 shows the results of DLPAlign, as well as other MSA tools, for OXBench. In addition to the complete set of families, the table shows the alignment accuracy for families with average PID of less than and more than 30%. Note that OXBench did not divide the whole database into different subsets. We made the division here because we thought the two parts after

	All (395)		0 - 30% (63)		30% - 100% (332)	
	TC	SP	TC	SP	TC	SP
DLPAlign	**82.52**	**90.42**	**44.16**	65.59	**89.80**	**95.13**
MLProbs	**82.72**	**90.57**	**45.13**	**66.56**	**89.86**	**95.12**
QuickProbs	81.77	90.17	41.50	64.50	88.35	94.46
PnpProbs	82.06	90.19	43.96	**66.33**	89.54	95.01
GLProbs	81.93	90.13	43.34	66.11	89.55	94.99
MSAProbs	81.50	89.83	42.81	65.22	89.08	94.78
ProbCons	80.68	89.45	41.50	64.50	88.35	94.46
PicXAA	81.14	89.61	39.78	63.16	89.32	94.93
MAFFT	78.15	88.07	35.93	60.68	86.40	93.53
Muscle	80.67	89.50	40.95	63.96	88.21	94.34
ClustalΩ	79.99	88.91	37.39	60.78	88.08	94.25
ProbAlign	81.68	89.97	41.06	63.80	89.39	94.93

Table 19 Average TC-scores and SP-scores for OXBench (Chapter 3)

separation could respectively represent divergent families and high similarity families.

It can be seen that no matter which divergent set or high similarity set is used, DLPAlign can always produce some improvement, although the improvements are sometimes small.

Table 20 summarizes the accuracy of the SABMark benchmark, which was divided into two subset Twilight Zone and Superfamilies, depending on the SCOP classification. These subsets together covered the entire known fold space using sequences with very low to low, and low to intermediate similarity, respectively. DLPAlign improved both subsets and the whole benchmark. For Twilight Zone, except for DLPAlign, none of the MSA tools could get a TC-score of more than 25%. In this subset, DLPAlign's TC-score was 4.38% higher than the third-best MSA tool, PnpProbs. The results were very surprising.

Families with low similarity are always the most challenging part of the MSA task. It is worth noting that DLPAlign gets better performance on the low or medium similarity protein families, which other progressive methods are not

	All (423)		Superfamily (315)		Twilight Zone (108)	
	TC	SP	TC	SP	TC	SP
DLPAlign	**42.59**	**61.82**	**48.37**	**67.76**	**25.71**	**44.51**
MLProbs	**42.58**	**62.10**	**48.63**	**67.92**	**24.94**	**44.78**
QuickProbs	40.65	61.05	46.56	66.86	23.40	44.11
PnpProbs	41.48	61.25	47.03	66.84	24.63	43.94
GLProbs	41.22	61.30	46.95	66.97	23.86	43.74
MSAProbs	40.04	60.24	45.81	66.00	22.60	42.46
ProbCons	39.17	59.69	44.64	65.27	22.57	42.42
PicXAA	38.44	59.06	44.45	65.18	20.33	40.23
MAFFT	33.00	53.15	39.10	60.05	14.72	32.14
Muscle	33.47	54.51	39.13	61.30	16.96	34.70
ClustalΩ	35.47	55.02	41.43	61.70	18.10	35.55
ProbAlign	38.63	59.53	44.11	65.40	22.64	42.43

Table 20 Average TC-scores and SP-scores for SABMark (Chapter 3)

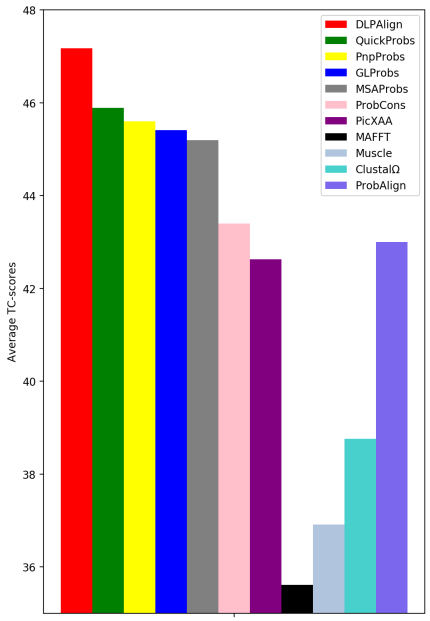

Figure 16 Over 711 families in BAliBASE, OXBench and SABmark with PID $\leq$ 30%. DLPAlign gets the best average TC-score 47.17 and improves about 2.8% over the second best one which is 45.89.

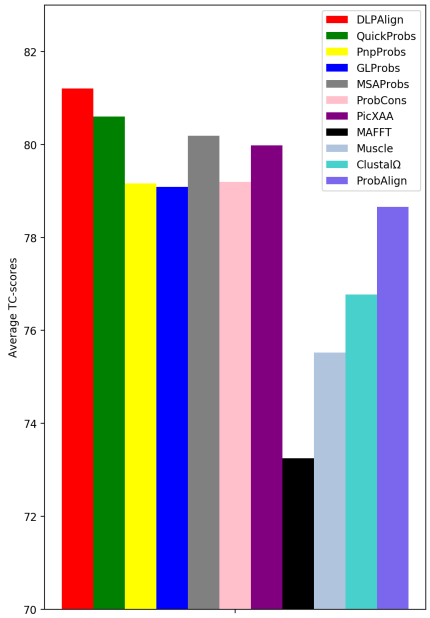

Figure 17 Over 352 families in BAliBASE, OXBench and SABmark with PID between 30% and 60%. DLPAlign gets the best average TC-score 81.21 and improves about 0.8% over the second best one which is 80.60.

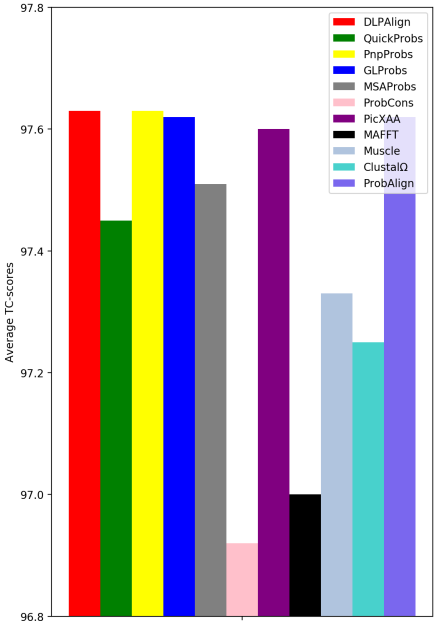

Figure 18 Over 141 families in BAliBASE, OXBench and SABmark with PID $> 60\%$. All the MSA tools could achieve more than 96.80 on average TC-score.

	Average running time (sec.)		
	BAliBASE	OXBench	SABMark
DLPAlign	38.10	0.84	0.40
QuickProbs	6.10	0.11	0.06
PnpProbs	13.30	0.27	0.21
GLProbs	13.63	0.23	0.15
MSAProbs	9.49	0.14	0.07
ProbCons	33.98	0.49	0.24
PicXAA	29.82	0.49	0.35
MAFFT	0.33	0.15	0.14
Muscle	1.69	0.04	0.08
ClustalΩ	0.97	0.03	0.04
ProbAlign	21.92	0.27	0.12

Table 21 Average running time (in seconds) of three benchmarks by DLPAlign and other MSA tools

good at. Figures 16 and 17 compare the average TC-scores on low similarity families (with PID $\leq$ 30%, 711 such families in all) and medium similarity (with 30% $<$ PID $\leq$ 60%, 352 such families in all) in BAliBASE, OXBench and SABmark. It can be seen from the figures that the improvement with DLPAlign was pronounced, especially in the low-similarity families set. Also, every MSA tool got a high TC-score for high-similarity protein families, as shown in Figure 18.

3.5.2 Efficiency results

All the tools were run on an HP desktop computer with four Intel Cores i5-3570 (3.40 GHz) and a main memory of size 19.4 GB. Table 21 shows the efficiency results.

It should be noted that ProbCons, Probalign, MSAProbs, GLProbs and PnpProbs all used the standard progressive alignment steps mentioned in Section 3.1, so there was not much difference in their running times. The running time of DLPAlign comprised mainly the running time of the decision-making model and the standard progressive alignment. As Table 21 shows, the time taken by the decision-making model was affected by the benchmark size (BAliBASE was largest benchmark while SABMark was the smallest), but in general, the

time was acceptable.

3.6 Applications of DLPAlign

3.6.1 Protein secondary structure prediction

Protein secondary structure prediction is an appropriate application of multiple sequence alignment [80].

We picked some protein sequences related to COVID-19, which could be found at PDB with these PDB ID: 6YI3, 6VYO, 6W9C, and 6W61 [1]. We then used the following process to evaluate the performance of DLPAlign with the other five highly accurate MSA tools

> Given a protein sequence S, we use Jpred 4 [87] to search protein sequences similar to this sequence. Then, we construct an MSA $\mathscr{M}$ for these sequences and S, and then use the secondary structure prediction tool provided on Jpred 4 to predict the secondary structure of S. Finally, comparing the predictions with reference secondary structures offered by Jpred 4.

Table 22 summarizes the number of wrongly aligned residues for each MSA tool. The table shows that DLPAlign always got the fewest wrong aligned residues of the leading MSA tools.

Figures 19 and 20 are two examples that show the predicted protein secondary structures by DLPAlign, QuickProbs, PnpProbs, GLProbs, MSAProbs and PicXAA on proteins with PDB ID 6W61 and 6YI3.

PDB ID	DLPAlign	QuickProbs	PnpProbs	GLProbs	MSAProbs	PicXAA
6YI3	**10**	16	12	12	15	16
6VYO	**5**	**5**	**5**	**5**	**5**	7
6W9C	**20**	31	24	22	23	32
6W61	**13**	17	**13**	**13**	15	20

Table 22 Number of wrongly aligned residues in the predicted secondary structures of proteins (Chapter 3)

[1] They are all available only from April this year, and relevant references have not been published.

```
      6W61:  MSSQAWQPGVAMPNLYKMQRMLLEKCDLQNYGDSATLPKGIMMNVAKYTQLCQYLNTLTL
 Reference:  --------------EEEEE----EEE---------------HHHHHHHHHHHHHHHHH---
  DLPAlign:  --------------EEEEE----EEE---------------EEHHHHHHHHHHHHH---E
QuickProbs:  --------------HEEEE----EEE---------------EEHHHHHHHHHHHHH---E
  PnpProbs:  --------------HEEEE----EEE---------------HHHHHHHHHHHHHHH---E
   GLProbs:  --------------HEEEE----EEE---------------HHHHHHHHHHHHHHH---E
  MSAProbs:  --------------HEEEE----EEE---------------HHHHHHHHHHHHHHH---E
    PicXAA:  --------------HEEEE----EEE---------------EEHHHHHHHHHHHHH---E

      6W61:  AVPYNMRVIHFGAGSDKGVAPGTAVLRQWLPTGTLLVDSDLNDFVSDADSTLIGDCATVH
 Reference:  -------EEE------------HHHHHHH-----EEEE-----------EEEEE------
  DLPAlign:  EE-----EEE------------HHHHHHH-----EEEE-----------EEEEEEE----
QuickProbs:  EE-----EEEE-----------HHHHHHH-----EEEE-----------EEEEEEE----
  PnpProbs:  EE-----EEEE-----------HHHHHHH-----EEEE-----------EEEEE------
   GLProbs:  EE-----EEEE-----------HHHHHHH-----EEEE-----------EEEEE------
  MSAProbs:  EE-----EEE------------HHHHHHH-----EEEE-----------EEEEEEE----
    PicXAA:  EE-----EEE------------HHHHHHH-----EEEE-----------EEEEE------

      6W61:  TANKWDLIISDMYDPKTKNVTKENDSKEGFFTYICGFIQQKLALGGSVAIKITEHSWNAD
 Reference:  -----EEEEEE---------------HHHHHHHHHHHHHHHH-----EEEEEE----HHH
  DLPAlign:  -----EEEEEE---------------HHHHHHHHHHHHHHHH----EEEEEEE----HHH
QuickProbs:  ----EEEEEEE----------------HHHHHHHHHHHHHHH----EEEEEEE----HHH
  PnpProbs:  ----EEEEEEE----------------HHHHHHHHHHHHHHH----EEEEEEE----HHH
   GLProbs:  ----EEEEEEE----------------HHHHHHHHHHHHHHH----EEEEEEE----HHH
  MSAProbs:  ----EEEEEEE----------------HHHHHHHHHHHHHHH----EEEEEEE----HHH
    PicXAA:  ----EEEEEEE----------------HHHHHHHHHHHHHHH----EEEEEEE----HHH

      6W61:  LYKLMGHFAWWTAFVTNVNASSSEAFLIGCNYLGKPREQIDGYVMHANYIFWRNTNPIQL
 Reference:  HHHHHH---EEEEEE-------HHEEEEE---------------------EEE--------
  DLPAlign:  HHHHHH---EEEEEE-------HEEEEEE---------------------EEE--------
QuickProbs:  HHHHHH---EEEEEE--------EEEEEEE---------------------EEE--------
  PnpProbs:  HHHHHH---EEEEEE-------HEEEEEE---------------------EEE--------
   GLProbs:  HHHHHH---EEEEEE-------HEEEEEE---------------------EEE--------
  MSAProbs:  HHHHHH---EEEEEE-------HEEEEEE-------------------EEEE--------
    PicXAA:  HHHHHHH--EEEEEE--------EEEEEEE---------------------EEE--------

      6W61:  SSYSLFDMSKFPLKLRGTAVMSLKEGQINDMILSLLSKGRLIIRENNRVVISSDVLVNN
 Reference:  ----------EEEE---EEEEE----HHHHHHHHHHH---EEE----EEEEE--EE---
  DLPAlign:  ---------EEEEE---EEEEE----HHHHHHHHHHH---EEE----EEEEE--EE---
QuickProbs:  ---------EEEEE---EEEEE----HHHHHHHHHHH---EEE----EEEEE--EEE--
  PnpProbs:  ---------EEEEE---EEEEE----HHHHHHHHHHH---EEE----EEEEE--EE---
   GLProbs:  ---------EEEEE---EEEEE----HHHHHHHHHHH---EEE----EEEEE--EE---
  MSAProbs:  ---------EEEEE---EEEEE----HHHHHHHHHHH---EEE----EEEEE--EE---
    PicXAA:  ----H-----------EEEEE----HHHHHHHHHHH---EEE----EEEEE--EE---
```

Figure 19 The predicted protein secondary structures by DLPAlign, QuickProbs, PnpProbs, GLProbs, MSAProbs and PicXAA on protein with PDB ID 6W61.

```
      6YI3:  GAMGLPNNTASWFTALTQHGKEDLKFPRGQGVPINTNSSPDDQIGYYRRATRRIRGGDGK
 Reference:  ----------EEEE-EE--------------------------EEEEE----------
  DLPAlign:  ----------EE---E--------------------------EEEEE--EEE-----
QuickProbs:  ------------------------------------------EEEEEEEEE------
  PnpProbs:  ------------------------------------------EEEEEEEEE------
   GLProbs:  ------------------------------------------EEEEEEEEE------
  MSAProbs:  ------------------------------------------EEEEEEEEE------
    PicXAA:  ------------------------------------------EEEE--EEEE-----

      6YI3:  MKDLSPRWYFYYLGTGPEAGLPYGANKDGIIWVATEGALNTPKDHIGTRNPANNAAIVLQ
 Reference:  --------EEEE----------------EEEEEE----------------------EEE
  DLPAlign:  --------EEEE----------------EEEEEE-----------------------
QuickProbs:  --------EEEE----------------EEEEEE-----------------------
  PnpProbs:  --------EEEE----------------EEEEE----------------------EEE
   GLProbs:  --------EEEE----------------EEEEE----------------------EEE
  MSAProbs:  --------EEEE----------------EEEEEE-----------------------
    PicXAA:  --------EEEE----------------EEEEE-----------------------

      6YI3:  LPQGTTLPKGFYAEGSRGGS
 Reference:  ----------EEE-------
  DLPAlign:  ----------EEEE------
QuickProbs:  -----------E--------
  PnpProbs:  ----------EEEE------
   GLProbs:  ----------EEEE------
  MSAProbs:  ----------EEEE------
    PicXAA:  ----------EEEE------
```

Figure 20 The predicted protein secondary structures by DLPAlign, QuickProbs, PnpProbs, GLProbs, MSAProbs and PicXAA on protein with PDB ID 6YI3.

3.7 Conclusions

The significant contribution of this paper is to use a deep-learning method as a decision-making model to determine which specific calculation method to use in the posterior probability matrix part of progressive alignment approaches and release a new progressive multiple protein sequence alignment tool based on this deep learning model. Our study showed that DLPAlign produced the most accurate decision-making model.

We did not optimize DLPAlign for efficiency, so in Chapter 4, we propose some research directions related to efficiency improvement in DLPAlign.

Though the current results of DLPAlign are not better than our previous research MLProbs, it is caused by a lack of training data, and since we keep accumulating data, the approach will keep improving.

Chapter 4

Discussion and Future works

In the Chapters 2 and 3, we explained how machine learning and deep learning can be applied to help construct better MSAs. In this chapter, we discuss the weaknesses in the previous two MSA methods and set the direction for future improvement.

4.1 Discussion

The research on the two data-centric MSA methods listed in this thesis shows that machine-learning and deep-learning methods can be used to improve the accuracy of existing MSA tools, which is also an extension of the application of machine learning and deep learning in bioinformatics.

Although the machine-learning and deep-learning methods applied in our research achieved relatively high accuracy, improvement is still possible. MLProbs just chose to use some shallow machine-learning algorithms. If enough data could be used for training some deep-learning models for it, the accuracy can be improved further. Besides, If we choose better features or better model training methods, it may further improve the accuracy of the models (not only in MLProbs, but also in DLPAlign) and even the accuracy of the final MSA. Another problem is efficiency, which is related mainly to DLPAlign. As can be seen from Table 21, the running time is significantly increased in DLPAlign because of multiple uses of the decision-making model.

Since our data-centric MSA methods have achieve good results on the alignment of small-scale protein families, the next step is to find out whether these data-centric methods can also be applied to large protein families.

In response to the two weaknesses and the next step, we propose some possible improvements in the next section.

4.2 Future works

Since two different MSA methods were investigated in this thesis, we propose future works from two perspectives.

4.2.1 Future works inspired by MLProbs

CNN, which has achieved great success in computer vision, is very suitable for training models like $\mathscr{P}^{\mathtt{Aln}}_{\mathtt{U,V}}$, which was mentioned in Chapter 2 for MSA construction.

An MSA is very similar to a 2D picture; they both have hierarchal structures. For example, an MSA has conserved columns, which form conserved regions, while a picture has points, which form lines, which in turn, form boxes. So determining whether an MSA should be aligned by `U` or `V` is similar to determining whether a picture is a dog or a cat. One major difficulty is that the input of our MSA problem is a protein family, not an MSA. One possible solution is that first, a quick MSA tool is used to construct an imperfect MSA, and then CNN is used to classify this imperfect MSA. CNN is known to be very good at recognizing imperfect pictures, allowing titling, and tolerating translation and other distortions and noises, so it will perform well at recognizing imperfect MSAs.

Instead of an MSA being treated as a computer vision problem, it may be treated as a natural language processing problem, with advanced tools, like LSTM and BERT [99], applied to help train the models.

For example, for a protein family, we can determine whether we should use `U` or `V` to align it as follows: for each pair of sequences in the family, we construct the optimal alignment of this pair, and then determine to which class it belongs. Then, we pick the class to which the majority of the pairs of sequences belong as the class of the family. The problem of determining to which class a pairwise alignment belongs can be treated as an NLP sentiment analysis problem; each column in the alignment is regarded as a word, the words comprise a sentence, and we need to determine the "emotion" (i.e., `U` or `V`) expressed by the sentence.

4.2.2 Future works inspired by DLPAlign

The efficiency of DLPAlign is reduced as the number of sequences increases, because the input of the decision model is sequence pairs. The more sequences, the more sequence pairs will be obtained, thus increasing the number of times the decision model runs.

One way to improve the efficiency is to reduce the number of times the decision model runs. In this study, we regard each sequence pair as a sentence to classify. If we take the combination of more than two sequences as the input

of the decision model, then the whole decision model will run less, which will significantly improve the efficiency of DLPAlign. This is one area for improvement in DLPAlign in the future.

Also, if we use a protein family simulation tool such as INDELible [100] to obtain enough protein family data, we can also consider the entire protein family, or the temporary MSA built by the fast MSA tool, as a training set of the decision model, so the decision model will run only once, thus further improving the efficiency.

Another direction related to DLPAlign is choosing better models for the protein sequence classification. In natural language processing field, except CNN and LSTM, BERT [99] is a very new and popular model for text classification or emotion classification problem. Since our protein sequence pair classification is similar to the classification tasks in natural language processing field, we can also consider using BERT to train the decision-making model in DLPAlign.

4.2.3 Future works on large-scale protein families

In Section 1.3.1, we described the general steps in constructing MSAs on large-scale protein families:

(1) divide massive data into different subsets according to specific rules;

(2) use MSA tools that achieve state-of-art effects on small-scale datasets to align each subgroup; and

(3) combine the MSAs of subsets to obtain the final MSA.

We can perform our MLProbs or DLPAlign in step (2) to get the MSAs on small datasets. For step (1), the difficulty is how to find a suitable method to divide the entire protein family into multiple subsets, while ensuring that the similarity within the group is high and the similarity between groups is low. We can divide the entire protein family into cluster [101] problems, which are more common in machine learning and deep learning. There are many high-accuracy models available for this.

References

[1] C. Grasso and C. Lee, "Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems," *Bioinformatics*, vol. 20, no. 10, pp. 1546–1556, 2004.

[2] C. Notredame and D. G. Higgins, "SAGA: Sequence alignment by genetic algorithm," *Nucleic acids research*, vol. 24, no. 8, pp. 1515–1524, 1996.

[3] J. Kim, S. Pramanik, and M. J. Chung, "Multiple sequence alignment using simulated annealing," *Bioinformatics*, vol. 10, no. 4, pp. 419–426, 1994.

[4] K. Katoh, K.-i. Kuma, H. Toh, and T. Miyata, "MAFFT version 5: Improvement in accuracy of multiple sequence alignment," *Nucleic acids research*, vol. 33, no. 2, pp. 511–518, 2005.

[5] J. S. Papadopoulos and R. Agarwala, "COBALT: Constraint-based alignment tool for multiple protein sequences," *Bioinformatics*, vol. 23, no. 9, pp. 1073–1079, 2007.

[6] J. Stoye, V. Moulton, and A. W. Dress, "DCA: An efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment," *Bioinformatics*, vol. 13, no. 6, pp. 625–626, 1997.

[7] O. Gotoh, "Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments," *Journal of molecular biology*, vol. 264, no. 4, pp. 823–838, 1996.

[8] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment," *science*, vol. 262, no. 5131, pp. 208–214, 1993.

[9] I. M. Wallace, O. O'sullivan, D. G. Higgins, and C. Notredame, "M-Coffee: Combining multiple sequence alignment methods with T-Coffee," *Nucleic acids research*, vol. 34, no. 6, pp. 1692–1699, 2006.

[10] P. Di Tommaso, S. Moretti, I. Xenarios, M. Orobitg, A. Montanyola, J.-M. Chang, J.-F. Taly, and C. Notredame, "T-coffee: A web server for the multiple sequence alignment of protein and rna sequences using structural information and homology extension," *Nucleic acids research*, vol. 39, no. suppl_2, W13–W17, 2011.

[11] D.-F. Feng and R. F. Doolittle, "Progressive sequence alignment as a prerequisitetto correct phylogenetic trees," *Journal of molecular evolution*, vol. 25, no. 4, pp. 351–360, 1987.

[12] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning," *Nature biotechnology*, vol. 33, no. 8, pp. 831–838, 2015.

[13] P. Inglese, J. S. McKenzie, A. Mroz, J. Kinross, K. Veselkov, E. Holmes, Z. Takats, J. K. Nicholson, and R. C. Glen, "Deep learning and 3D-DESI imaging reveal the hidden metabolic heterogeneity of cancer," *Chemical science*, vol. 8, no. 5, pp. 3500–3511, 2017.

[14] N. H. Tran, X. Zhang, L. Xin, B. Shan, and M. Li, "De novo peptide sequencing by deep learning," *Proceedings of the National Academy of Sciences*, vol. 114, no. 31, pp. 8247–8252, 2017.

[15] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu, "Accurate de novo prediction of protein contact map by ultra-deep learning model," *PLoS computational biology*, vol. 13, no. 1, e1005324, 2017.

[16] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature methods*, vol. 12, no. 10, pp. 931–934, 2015.

[17] V. Likic, "The Needleman-Wunsch algorithm for sequence alignment," *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, pp. 1–46, 2008.

[18] T. F. Smith, M. S. Waterman, *et al.*, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[19] F. Sievers and D. G. Higgins, "Clustal omega, accurate alignment of very large numbers of sequences," in *Multiple sequence alignment methods*, Springer, 2014, pp. 105–116.

[20] C. Notredame, D. G. Higgins, and J. Heringa, "T-Coffee: A novel method for fast and accurate multiple sequence alignment," *Journal of molecular biology*, vol. 302, no. 1, pp. 205–217, 2000.

[21] R. C. Edgar, "MUSCLE: Multiple sequence alignment with high accuracy and high throughput," *Nucleic acids research*, vol. 32, no. 5, pp. 1792–1797, 2004.

[22] K. Katoh and H. Toh, "PartTree: An algorithm to build an approximate tree from a large number of unaligned sequences," *Bioinformatics*, vol. 23, no. 3, pp. 372–374, 2007.

[23] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "ProbCons: Probabilistic consistency-based multiple sequence alignment," *Genome research*, vol. 15, no. 2, pp. 330–340, 2005.

[24] U. Roshan and D. R. Livesay, "Probalign: Multiple sequence alignment using partition function posterior probabilities," *Bioinformatics*, vol. 22, no. 22, pp. 2715–2721, 2006.

[25] Y. Liu, B. Schmidt, and D. L. Maskell, "MSAProbs: Multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities," *Bioinformatics*, vol. 26, no. 16, pp. 1958–1964, 2010.

[26] Y. Ye, D. W.-l. Cheung, Y. Wang, S.-M. Yiu, Q. Zhang, T.-W. Lam, and H.-F. Ting, "GLProbs: Aligning multiple sequences adaptively," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 12, no. 1, pp. 67–78, 2015.

[27] A. Gudys and S. Deorowicz, "QuickProbs-a fast multiple sequence alignment algorithm designed for graphics processors," *PloS one*, vol. 9, no. 2, e88901, 2014.

[28] A. Gudyś and S. Deorowicz, "QuickProbs 2: Towards rapid construction of high-quality alignments of large protein families," *Scientific reports*, vol. 7, no. 1, pp. 1–12, 2017.

[29] F. Armougom, S. Moretti, O. Poirot, S. Audic, P. Dumas, B. Schaeli, V. Keduas, and C. Notredame, "Expresso: Automatic incorporation of structural information in multiple sequence alignments using 3d-coffee," *Nucleic acids research*, vol. 34, no. suppl_2, W604–W608, 2006.

[30] J.-M. Chang, P. Di Tommaso, J.-F. Taly, and C. Notredame, "Accurate multiple sequence alignment of transmembrane proteins with psi-coffee," *BMC bioinformatics*, vol. 13, no. S4, S1, 2012.

[31] S. M. E. Sahraeian and B.-J. Yoon, "PicXAA: Greedy probabilistic construction of maximum expected accuracy alignment of multiple sequences," *Nucleic acids research*, vol. 38, no. 15, pp. 4917–4928, 2010.

[32] Y. Ye, T.-W. Lam, and H.-F. Ting, "PnpProbs: A better multiple sequence alignment tool by better handling of guide trees," *BMC bioinformatics*, vol. 17, no. 8, p. 285, 2016.

[33] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow, "Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees," *Science*, vol. 324, no. 5934, pp. 1561–1564, 2009.

[34] K. Liu, T. J. Warnow, M. T. Holder, S. M. Nelesen, J. Yu, A. P. Stamatakis, and C. R. Linder, "SATe-II: Very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees," *Systematic biology*, vol. 61, no. 1, p. 90, 2012.

[35] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, and T. Warnow, "PASTA: Ultra-large multiple sequence alignment for nucleotide and amino-acid sequences," *Journal of Computational Biology*, vol. 22, no. 5, pp. 377–386, 2015.

[36] D. N. Nam-phuong, S. Mirarab, K. Kumar, and T. Warnow, "Ultra-large alignments using phylogeny-aware profiles," *Genome biology*, vol. 16, no. 1, p. 124, 2015.

[37] O. Trelles, P. Prins, M. Snir, and R. C. Jansen, "Big data, but are we ready?" *Nature Reviews Genetics*, vol. 12, no. 3, pp. 224–224, 2011.

[38] N. H. Shah, "Translational bioinformatics embraces big data," *Yearbook of medical informatics*, vol. 21, no. 01, pp. 130–134, 2012.

[39] N. H. Shah and J. D. Tenenbaum, "Focus on translational bioinformatics: The coming age of data-driven medicine: Translational bioinformatics' next frontier," *Journal of the American Medical Informatics Association: JAMIA*, vol. 19, no. e1, e2, 2012.

[40] J. S. Conery, J. M. Catchen, and M. Lynch, "Rule-based workflow management for bioinformatics," *The VLDB journal*, vol. 14, no. 3, pp. 318–329, 2005.

[41] X. Liu, J. Wu, J. Wang, X. Liu, S. Zhao, Z. Li, L. Kong, X. Gu, J. Luo, and G. Gao, "Weblab: A data-centric, knowledge-sharing bioinformatic platform," *Nucleic acids research*, vol. 37, no. suppl_2, W33–W39, 2009.

[42] K. Dempsey, B. Currall, R. Hallworth, and H. Ali, "An intelligent data-centric approach toward identification of conserved motifs in protein sequences," in *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, 2010, pp. 398–401.

[43] C. Wang, B. B. Zhou, and A. Y. Zomaya, "Evolvingspace: A data centric framework for integrating bioinformatics applications," *IEEE Transactions on Computers*, vol. 59, no. 6, pp. 721–734, 2010.

[44] J. Frankovich, C. A. Longhurst, and S. M. Sutherland, "Evidence-based medicine in the emr era," *N Engl J Med*, vol. 365, no. 19, pp. 1758–1759, 2011.

[45] N. H. Tran, R. Qiao, L. Xin, X. Chen, C. Liu, X. Zhang, B. Shan, A. Ghodsi, and M. Li, "Deep learning enables de novo peptide sequencing from data-independent-acquisition mass spectrometry," *Nature methods*, vol. 16, no. 1, pp. 63–66, 2019.

[46] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in bioinformatics*, vol. 18, no. 5, pp. 851–869, 2017.

[47] K. Lan, D.-t. Wang, S. Fong, L.-s. Liu, K. K. Wong, and N. Dey, "A survey of data mining and deep learning in bioinformatics," *Journal of medical systems*, vol. 42, no. 8, p. 139, 2018.

[48] R. S. Olson, W. La Cava, Z. Mustahsan, A. Varik, and J. H. Moore, "Data-driven advice for applying machine learning to bioinformatics problems," *arXiv preprint arXiv:1708.05070*, 2017.

[49] Y. Li, C. Huang, L. Ding, Z. Li, Y. Pan, and X. Gao, "Deep learning in bioinformatics: Introduction, application, and perspective in the big data era," *Methods*, vol. 166, pp. 4–21, 2019.

[50] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.

[51] B. Tang, Z. Pan, K. Yin, and A. Khateeb, "Recent advances of deep learning in bioinformatics and computational biology," *Frontiers in genetics*, vol. 10, 2019.

[52] A. Serra, P. Galdi, and R. Tagliaferri, "Machine learning for bioinformatics and neuroimaging," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 5, e1248, 2018.

[53] L. Peng, M. Peng, B. Liao, G. Huang, W. Li, and D. Xie, "The advances and challenges of deep learning application in biological big data processing," *Current Bioinformatics*, vol. 13, no. 4, pp. 352–359, 2018.

[54] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, "BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark," *Proteins: Structure, Function, and Bioinformatics*, vol. 61, no. 1, pp. 127–136, 2005.

[55] G. Raghava, S. M. Searle, P. C. Audley, J. D. Barber, and G. J. Barton, "OXBench: A benchmark for evaluation of protein multiple sequence alignment accuracy," *BMC bioinformatics*, vol. 4, no. 1, p. 47, 2003.

[56] A. S. Siddiqui, U. Dengler, and G. J. Barton, "3Dee: A database of protein structural domains," *Bioinformatics*, vol. 17, no. 2, pp. 200–201, 2001.

[57] C. C. Minh, J. Chung, C. Kozyrakis, and K. Olukotun, "STAMP: Stanford transactional applications for multi-processing," in *2008 IEEE International Symposium on Workload Characterization*, IEEE, 2008, pp. 35–46.

[58] I. Van Walle, I. Lasters, and L. Wyns, "SABmark—a benchmark for sequence alignment that covers the entire known fold space," *Bioinformatics*, vol. 21, no. 7, pp. 1267–1268, 2004.

[59] A. G. Murzin, S. E. Brenner, T. Hubbard, C. Chothia, *et al.*, "SCOP: A structural classification of proteins database for the investigation of sequences and structures," *Journal of molecular biology*, vol. 247, no. 4, pp. 536–540, 1995.

[60] A. Andreeva, A. Prlić, T. J. Hubbard, and A. G. Murzin, "SISYPHUS —structural alignments for proteins with non-trivial relationships," *Nucleic acids research*, vol. 35, no. suppl_1, pp. D253–D259, 2007.

[61] K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington, "HOMSTRAD: A database of protein structure alignments for homologous families," *Protein science*, vol. 7, no. 11, pp. 2469–2471, 1998.

[62] P. D. Bank, "Protein data bank," *Nature New Biol*, vol. 233, p. 223, 1971.

[63] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S. R. Eddy, S. Griffiths-Jones, K. L. Howe, M. Marshall, and E. L. Sonnhammer, "The Pfam protein families database," *Nucleic acids research*, vol. 30, no. 1, pp. 276–280, 2002.

[64] N. Daniels, A. Kumar, L. Cowen, and M. Menke, "Touring protein space with Matt," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 9, no. 1, pp. 286–293, 2011.

[65] M. Menke, B. Berger, and L. Cowen, "Matt: Local flexibility aids protein multiple structure alignment," *PLoS computational biology*, vol. 4, no. 1, 2008.

[66] R. Edgar, *Bench*, `https://www.drive5.com/bench/`, Accessed Aug. 1, 2019.

[67] J. D. Thompson, F. Plewniak, and O. Poch, "BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs.," *Bioinformatics (Oxford, England)*, vol. 15, no. 1, pp. 87–88, 1999.

[68] *Qscore*, `https://www.drive5.com/qscore/`, Accessed Aug. 1, 2019.

[69] T. Wheeler and J. Kececioglu, *Opal: Software for aligning multiple biological sequences (version 2.1. 0)*,", 2012.

[70] J. Kececioglu and D. DeBlasio, "Accuracy estimation and parameter advising for protein multiple sequence alignment," *Journal of Computational Biology*, vol. 20, no. 4, pp. 259–279, 2013.

[71] D. DeBlasio and J. Kececioglu, "Boosting alignment accuracy by adaptive local realignment," in *International Conference on Research in Computational Molecular Biology*, Springer, 2017, pp. 1–17.

[72] G. Landan and D. Graur, "Heads or tails: A simple reliability check for multiple sequence alignments," *Molecular biology and evolution*, vol. 24, no. 6, pp. 1380–1383, 2007.

[73] O. Penn, E. Privman, H. Ashkenazy, G. Landan, D. Graur, and T. Pupko, "GUIDANCE: A web server for assessing alignment confidence scores," *Nucleic acids research*, vol. 38, no. suppl_2, W23–W28, 2010.

[74] I. Sela, H. Ashkenazy, K. Katoh, and T. Pupko, "GUIDANCE2: Accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters," *Nucleic acids research*, vol. 43, no. W1, W7–W14, 2015.

[75] T. H. Ogden and M. S. Rosenberg, "Multiple sequence alignment accuracy and phylogenetic inference," *Systematic biology*, vol. 55, no. 2, pp. 314–328, 2006.

[76] H. Li, A. Coghlan, J. Ruan, L. J. Coin, J.-K. Heriche, L. Osmotherly, R. Li, T. Liu, Z. Zhang, L. Bolund, *et al.*, "TreeFam: A curated database of phylogenetic trees of animal gene families," *Nucleic acids research*, vol. 34, no. suppl_1, pp. D572–D580, 2006.

[77] S. Kumar, G. Stecher, M. Li, C. Knyaz, and K. Tamura, "MEGA X: Molecular evolutionary genetics analysis across computing platforms," *Molecular biology and evolution*, vol. 35, no. 6, pp. 1547–1549, 2018.

[78] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Mathematical biosciences*, vol. 53, no. 1-2, pp. 131–147, 1981.

[79] J. Sukumaran and M. T. Holder, "DendroPy: A Python library for phylogenetic computing," *Bioinformatics*, vol. 26, no. 12, pp. 1569–1571, 2010.

[80] V. Simossis and J. Heringa, "Integrating protein secondary structure prediction and multiple sequence alignment," *Current Protein and Peptide Science*, vol. 5, no. 4, pp. 249–266, 2004.

[81] H.-M. Chu, R.-T. Guo, T.-W. Lin, C.-C. Chou, H.-L. Shr, H.-L. Lai, T.-Y. Tang, K.-J. Cheng, B. L. Selinger, and A. H.-J. Wang, "Structures of Selenomonas ruminantium phytase in complex with persulfated phytate: DSP phytase fold and mechanism for sequential substrate hydrolysis," *Structure*, vol. 12, no. 11, pp. 2015–2024, 2004.

[82] J. Eberhardt, A. G. McEwen, W. Bourguet, D. Moras, and A. Dejaegere, "A revisited version of the apo structure of the ligand-binding domain of the human nuclear receptor retinoic X receptor alpha," *Acta Crystallographica Section F: Structural Biology Communications*, vol. 75, no. 2, 2019.

[83] C. Wang, A. A. Aleksandrov, Z. Yang, F. Forouhar, E. A. Proctor, P. Kota, J. An, A. Kaplan, N. Khazanov, G. Boël, *et al.*, "Ligand binding to a remote site thermodynamically corrects the F508del mutation in the human cystic fibrosis transmembrane conductance regulator," *Journal of Biological Chemistry*, vol. 293, no. 46, pp. 17 685–17 704, 2018.

[84] M. G. Baud, E. Lin-Shiao, M. Zengerle, C. Tallant, and A. Ciulli, "New synthetic routes to triazolo-benzodiazepine analogues: Expanding the scope of the bump-and-hole approach for selective bromo and extra-terminal (BET) bromodomain inhibition," *Journal of medicinal chemistry*, vol. 59, no. 4, pp. 1492–1500, 2015.

[85] D. Nüss, P. Goettig, I. Magler, U. Denk, M. Breitenbach, P. B. Schneider, H. Brandstetter, and B. Simon-Nobbe, "Crystal structure of the NADP-dependent mannitol dehydrogenase from Cladosporium herbarum: Implications for oligomerisation and catalysis," *Biochimie*, vol. 92, no. 8, pp. 985–993, 2010.

[86] Q. Xu, D. Buckley, C. Guan, and H.-C. Guo, "Structural insights into the mechanism of intramolecular proteolysis," *Cell*, vol. 98, no. 5, pp. 651–661, 1999.

[87] A. Drozdetskiy, C. Cole, J. Procter, and G. J. Barton, "JPred4: A protein secondary structure prediction server," *Nucleic acids research*, vol. 43, no. W1, W389–W394, 2015.

[88] P. Sneath and R. Sokal, "Numerical Taxonomy WH Freeman and Co," *San Francisco*, pp. 1–573, 1973.

[89] S. Lai, K. Liu, S. He, and J. Zhao, "How to generate a good word embedding," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 5–14, 2016.

[90] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017.

[91] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, Presses universitaires de Louvain, vol. 89, 2015.

[92] R. Dey and F. M. Salemt, "Gate-variants of gated recurrent unit (GRU) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, IEEE, 2017, pp. 1597–1600.

[93] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 11–15.

[94] C. Guggilla, T. Miller, and I. Gurevych, "CNN-and LSTM-based claim classification in online user comments," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2740–2751.

[95] Q. Liu, F. Zhou, R. Hang, and X. Yuan, "Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification," *Remote Sensing*, vol. 9, no. 12, p. 1330, 2017.

[96] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A C-LSTM neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[97] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, Montreal, Canada, vol. 14, 1995, pp. 1137–1145.

[98] V. Van Asch, "Macro-and micro-averaged evaluation measures [[basic draft]]," *Belgium: CLiPS*, vol. 49, 2013.

[99] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[100] W. Fletcher and Z. Yang, "INDELible: A flexible simulator of biological sequence evolution," *Molecular biology and evolution*, vol. 26, no. 8, pp. 1879–1888, 2009.

[101] A. K. Mann and N. Kaur, "Review paper on clustering techniques," *Global Journal of Computer Science and Technology*, 2013.