Data-centric Approaches for better Multiple Sequence Alignment

by

Kuang Mengmeng

Supervised by

Dr. Hing-fung Ting

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 区盟盟

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
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

Declaration ......................................................... i
Acknowledgements ............................................... ii
Table of Contents ................................................. iv
List of Figures .................................................. vii
List of Tables ................................................... ix
List of Abbreviations ........................................... x

1 Introduction ................................................... 1
  1.1 Background .................................................. 1
  1.2 Problem definition .......................................... 2
  1.3 Previous research ........................................... 3
    1.3.1 Algorithm-centric approaches in MSA ................. 3
    1.3.2 Data-centric approaches in Bioinformatics .......... 5
  1.4 Benchmarks and measurements .............................. 6
  1.5 Main contributions ........................................ 9
    1.5.1 MLProbs: A Data-centric Pipeline for better Multiple
           Sequence Alignment ........................................ 9
    1.5.2 DLPAlign: A Deep Learning based Progressive Align-
           ment for Multiple Protein Sequences .................. 10
  1.6 Thesis organization ....................................... 11

2 MLProbs ....................................................... 12
  2.1 Overview .................................................... 12
  2.2 $\mathcal{P}_{\text{Aln}}^{UV}$: A data-centric method for choosing better tools ...... 13
  2.3 $\mathcal{P}_{\text{Ral}}^{U}$: A data-centric method for better realignment ....... 15
  2.4 Trainings and evaluations of the classifiers ............... 21
  2.5 An implementation of the pipeline $\mathcal{P}_{\text{Ral}}^{U} \circ \mathcal{P}_{\text{Aln}}^{P,\text{NP}}$ .......... 22
  2.6 Comparing MLProbs with other MSA tools .................. 23
    2.6.1 Accuracy results ......................................... 23
    2.6.2 Efficiency results ....................................... 28
  2.7 Applications of MLProbs ................................... 29
    2.7.1 Phylogenetic tree construction .......................... 29
    2.7.2 Protein secondary structure prediction ............... 33
2.8 Conclusions .................................................. 33

3 DLPAlign ............................................. 35

3.1 Overview .................................................. 35
3.2 Selection of promotion parts .............................. 37
3.3 Deep-learning-based decision-making method .......... 39
3.4 Decision-making model-based progressive alignment method .. 43
3.5 Comparing DLPAlign with other MSA tools .......... 48
  3.5.1 Accuracy results ......................................... 48
  3.5.2 Efficiency results ....................................... 53
3.6 Applications of DLPAlign ................................. 54
  3.6.1 Protein secondary structure prediction .............. 54
3.7 Conclusions ................................................ 56

4 Discussions and Future works ......................... 57

4.1 Discussion ................................................ 57
4.2 Future works ............................................. 57
  4.2.1 Future works inspired by MLProbs .................. 58
  4.2.2 Future works inspired by DLPAlign ................. 58
  4.2.3 Future works on large-scale protein families ...... 59

References .................................................. 60
List of Figures

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. ........................................... 16

Figure 2
An example of the calculation of score of a column. .................. 17

Figure 3
Average TC-scores on four empirical benchmarks (BAliBASE, OXBench, OXBench-X, SABMark) with different minimum realignment widths 22

Figure 4
Average TC-scores on four empirical benchmarks (BAliBASE, OXBench, OXBench-X, SABMark) of PnpProbs, P, NP and the pipeline $\mathcal{P}_{\text{Aln}}^{\text{P,NP}}$ 23

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 $\mathcal{P}_{\text{Q}}^{\text{Ral}}$ ..................... 24

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. ................................................................. 28

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. ............................................. 29

Figure 8
The phylogenetic tree of TF105063 constructed by MLProbs. ....... 31

Figure 9
The reference phylogenetic tree of TF105063. ........................... 32
Figure 10
The processes of a general progressive alignment. .......................... 36

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 $\mathcal{P}_A^i$ 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. ............................... 43

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 $512 \times 8$ 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 $4 \times 1$ matrix that represents the final category. ............................... 44

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 ............................... 46

Figure 14
The guide tree of family “BB11018” in BAliBASE calculated by DLPAlign. ................................................................. 47

Figure 15
The order of progressive alignment in DLPAlign of family “BB11018” in BAliBASE. ................................................................. 48

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. ...... 51
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.

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

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

Table 1
The number of families and the total number of sequences in each benchmark. ................................. 8

Table 2
The Information of three empirical benchmarks ............... 8

Table 3
Comparing $C_{p, np}^\text{Aln}$ and PnpProbs’s 18%-rule. ...................... 14

Table 4
Testing results for the classifiers $C_U^\text{Aln}$. ......................... 14

Table 5
TC scores obtained by the pipeline $P_{U, V}^\text{Aln}$. ....................... 16

Table 6
Testing results for the classifiers $C_U^\text{Ral}$. ......................... 20

Table 7
Average TC-scores of the pipelines $P_{U}^\text{Ral} \circ U$. .................... 20

Table 8
Average TC-scores and SP-scores for SABMark (Chapter 2) ....... 25

Table 9
Average TC-scores and SP-scores for BAliBASE (Chapter 2) ....... 26

Table 10
Average TC-scores and SP-scores for OXBench (Chapter 2) ....... 27

Table 11
Average TC-scores and SP-scores for OXBench-X (Chapter 2) ....... 27

Table 12
Average running time (in seconds) for constructing an MSA ...... 30

Table 13
The unweighted RF-distances for the phylogenetic trees constructed 31

Table 14
Number of wrongly aligned residues in the predicted secondary structures (Chapter 2) ..................... 33
Table 15
Average TC-scores of each tool on the three empirical benchmark databases ........................................ 39

Table 16
The macro average precision, recall and $F_1$-score on the test data. . . 42

Table 17
The precision, recall and $F_1$-score on the test data in different categories by CNN + BiLSTM structure. ......................... 42

Table 18
Average TC-scores and SP-scores for BAliBASE (Chapter 3) ....... 49

Table 19
Average TC-scores and SP-scores for OXBench (Chapter 3) ....... 50

Table 20
Average TC-scores and SP-scores for SABMark (Chapter 3) ....... 51

Table 21
Average running time (in seconds) of three benchmarks by DL-PAlign and other MSA tools ................................. 53

Table 22
Number of wrongly aligned residues in the predicted secondary structures of proteins (Chapter 3) ................................. 54
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Convolutional neural network</td>
<td>40</td>
</tr>
<tr>
<td>FFT</td>
<td>fast Fourier Transform</td>
<td>4</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Unit networks</td>
<td>40</td>
</tr>
<tr>
<td>HMM</td>
<td>hidden Markov model</td>
<td>4</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short Term Memory networks</td>
<td>40</td>
</tr>
<tr>
<td>MSA</td>
<td>Multiple Sequence Alignment</td>
<td>1</td>
</tr>
<tr>
<td>PDB</td>
<td>Protein Data Bank</td>
<td>7</td>
</tr>
<tr>
<td>PID</td>
<td>percentage identity</td>
<td>5</td>
</tr>
<tr>
<td>RF</td>
<td>Robinson-Foulds</td>
<td>30</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
<td>35</td>
</tr>
<tr>
<td>RNN</td>
<td>recurrent neural network</td>
<td>40</td>
</tr>
<tr>
<td>RRR</td>
<td>realign reliable regions</td>
<td>23</td>
</tr>
<tr>
<td>RUR</td>
<td>realign unreliable regions</td>
<td>23</td>
</tr>
<tr>
<td>SCOP</td>
<td>Structural Classification of Proteins</td>
<td>7</td>
</tr>
<tr>
<td>SP</td>
<td>sum-of-pairs</td>
<td>7</td>
</tr>
<tr>
<td>TC</td>
<td>total-column</td>
<td>7</td>
</tr>
<tr>
<td>UPGMA</td>
<td>unweighted pair group method with arithmetic mean</td>
<td>35</td>
</tr>
<tr>
<td>WPGMA</td>
<td>weighted pair group method with arithmetic mean</td>
<td>35</td>
</tr>
</tbody>
</table>
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 $S_i, i = 1, 2, \cdots, n$
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

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

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:

\[
S' := \begin{cases} 
S'_1 = (S'_{11}, S'_{12}, \cdots, S'_{1l}) \\
S'_2 = (S'_{21}, S'_{22}, \cdots, S'_{2l}) \\
\vdots \\
S'_n = (S'_{n1}, S'_{n2}, \cdots, S'_{nl}) 
\end{cases}
\]
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. Part-Tree [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.
1.3 Previous research

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 erythematos 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.
Chapter 1. Introduction

<table>
<thead>
<tr>
<th></th>
<th>Num. of families</th>
<th>Total num. of sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAliBASE</td>
<td>386</td>
<td>11082</td>
</tr>
<tr>
<td>OXBench</td>
<td>395</td>
<td>3292</td>
</tr>
<tr>
<td>SABmark</td>
<td>423</td>
<td>2418</td>
</tr>
<tr>
<td>SISYPHUS</td>
<td>126</td>
<td>1772</td>
</tr>
<tr>
<td>HOMSTRAD</td>
<td>1030</td>
<td>3454</td>
</tr>
<tr>
<td>Mattbench</td>
<td>259</td>
<td>1698</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2619</strong></td>
<td><strong>23716</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Information</th>
<th>BAliBASE</th>
<th>OXBench</th>
<th>SABMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Families</td>
<td>386</td>
<td>395</td>
<td>423</td>
</tr>
<tr>
<td>Minimum Length of Sequences</td>
<td>36</td>
<td>42</td>
<td>30</td>
</tr>
<tr>
<td>Maximum Length of Sequences</td>
<td>7923</td>
<td>544</td>
<td>796</td>
</tr>
<tr>
<td><strong>Average Length of Sequences</strong></td>
<td><strong>344.23</strong></td>
<td><strong>124.82</strong></td>
<td><strong>173.87</strong></td>
</tr>
<tr>
<td>Minimum Number of Sequences</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Maximum Number of Sequences</td>
<td>142</td>
<td>122</td>
<td>25</td>
</tr>
<tr>
<td><strong>Average Number of Sequences</strong></td>
<td><strong>28.71</strong></td>
<td><strong>8.33</strong></td>
<td><strong>5.72</strong></td>
</tr>
</tbody>
</table>

**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. Tables 8, 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 ≤ 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 DLP Align: 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\% $< \text{PID} \leq 60\%$; 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 $\mathcal{P}_{U,V}^{\text{Aln}}$: 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 $C_{\text{Aln},U,V}$ on protein families:

$C_{\text{Aln},U,V}$ has two classes 0 and 1. A protein family $\mathcal{F}$ is in class 1 if the MSA constructed by $V$ is better than that constructed by $U$; otherwise, $\mathcal{F}$ is in 0.

We used the popular $TC$ score to measure the goodness of an alignment. The $TC$ score of an alignment $\mathcal{M}$ is the percentage of columns in $\mathcal{M}$ 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 $C_{\text{Aln},U,V}$, which can naturally lead to way to outperform $U$ and $V$: We use machine learning algorithm to construct a model (or classifier) for $C_{\text{Aln},U,V}$. Then, we construct the alignments using the pipeline $\mathcal{P}_{U,V}^{\text{Aln}}$, which, given an input family $\mathcal{F}$, it first uses the classifier to determine to which class in $C_{\text{Aln},U,V}$ $\mathcal{F}$ belongs, and it uses $V$ to construct an alignment for $\mathcal{F}$ if the family belongs to the class 1; otherwise, it uses $U$.

We have implemented the pipelines $\mathcal{P}_{U,V}^{\text{Aln}}$ for various tools $U$ and $V$. We have particular interest in the tool PnpProbs. To construct an MSA for input family $\mathcal{F}$, PnpProbs first computes the average PID of $\mathcal{F}$, 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 $C_{\text{Aln}}^{P,NP}$ has better precision and recall. This is indeed the case; as can be seen from Table 3, the precision and recall of $C_{\text{Aln}}^{P,NP}$ are significantly higher than those of the 18%-rule of PnpProbs.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{Aln}}^{P,NP}$</td>
<td>86.28</td>
<td>93.70</td>
</tr>
<tr>
<td>The 18%-rule</td>
<td>79.43</td>
<td>80.80</td>
</tr>
</tbody>
</table>

**Table 3** Comparing $C_{\text{Aln}}^{P,NP}$ and PnpProbs’s 18%-rule.

Based on $C_{\text{Aln}}^{P,NP}$, we have implemented the pipeline $P_{\text{Aln}}^{P,NP}$, and we have also implemented the pipelines $P_{\text{Aln}}^{Pnp,Q}$, $P_{\text{Aln}}^{GL,MSA}$, $P_{\text{Aln}}^{GL,Pic}$, and $P_{\text{Aln}}^{MSA,MAF}$ 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.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{Aln}}^{P,NP}$</td>
<td>86.28</td>
<td>93.70</td>
<td>89.84</td>
</tr>
<tr>
<td>$C_{\text{Aln}}^{P,NP}$</td>
<td>82.36</td>
<td>86.27</td>
<td>84.27</td>
</tr>
<tr>
<td>$C_{\text{Aln}}^{P,NP}$</td>
<td>84.66</td>
<td>87.97</td>
<td>86.28</td>
</tr>
<tr>
<td>$C_{\text{Aln}}^{P,NP}$</td>
<td>86.12</td>
<td>91.38</td>
<td>88.67</td>
</tr>
<tr>
<td>$C_{\text{Aln}}^{P,NP}$</td>
<td>84.23</td>
<td>86.88</td>
<td>85.53</td>
</tr>
</tbody>
</table>

**Table 4** Testing results for the classifiers $C_{U,V}^{\text{Aln}}$.

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

$C_{\text{Aln}}^{P,NP}$ are not exceptionally good, but Table 5 shows that $P_{\text{Aln}}^{P,NP}$ can still help improve the alignment accuracy of PnpProbs. In fact, in many cases, even
though the classifier \( C_{P, NP} \) advises a wrong tool for input family \( \mathcal{F} \), the TC scores of the alignments constructed by \( P \) and \( NP \) for \( \mathcal{F} \) 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 \( C_{P, NP} \).

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 \( \mathcal{P}_{Aln}^{\text{Pnp,Q}, \text{GL, MSA}} \) and \( \mathcal{P}_{\text{MSA, MAF}}^{\text{Aln}} \). 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 \( \mathcal{P}_{\text{MSA, MAF}}^{\text{Aln}} \) can make no improvement on BALiBASE. MSAProbs (with score 64.51) is dominantly 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 \( C_{\text{MSA, MAF}}^{\text{Aln}} \) that chooses MAFFT instead of MSA are serious enough to reduce the average TC score of the pipeline \( \mathcal{P}_{\text{MSA, MAF}}^{\text{Aln}} \) to one smaller than that of MSAProbs.

### 2.3 \( \mathcal{P}_{U}^{\text{Ral}} \): 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 \( \mathcal{M} \), we identify regions in \( \mathcal{M} \) (i.e., blocks of consecutive columns of \( \mathcal{M} \)) 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
<table>
<thead>
<tr>
<th></th>
<th>BAiLiBASE</th>
<th>OXBench-X</th>
<th>OXBench</th>
<th>SABMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>62.17</td>
<td>59.53</td>
<td>82.18</td>
<td>41.39</td>
</tr>
<tr>
<td>NP</td>
<td>60.74</td>
<td>57.77</td>
<td>82.05</td>
<td>41.28</td>
</tr>
<tr>
<td>P_{Aln}</td>
<td>63.30</td>
<td>60.10</td>
<td>82.43</td>
<td>42.13</td>
</tr>
<tr>
<td>P_{Aln}</td>
<td>62.46</td>
<td>59.54</td>
<td>82.06</td>
<td>41.48</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>65.41</td>
<td>59.44</td>
<td>81.77</td>
<td>40.65</td>
</tr>
<tr>
<td>P_{Aln}</td>
<td>64.31</td>
<td>60.07</td>
<td>82.31</td>
<td>41.81</td>
</tr>
<tr>
<td>GLProbs</td>
<td>62.09</td>
<td>59.34</td>
<td>81.93</td>
<td>41.22</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>64.51</td>
<td>59.37</td>
<td>81.50</td>
<td>40.04</td>
</tr>
<tr>
<td>GL_{Aln,MSA}</td>
<td>63.83</td>
<td>59.89</td>
<td>82.19</td>
<td>41.56</td>
</tr>
<tr>
<td>GLProbs</td>
<td>62.09</td>
<td>59.34</td>
<td>81.93</td>
<td>41.22</td>
</tr>
<tr>
<td>PicXAA</td>
<td>60.97</td>
<td>58.85</td>
<td>81.14</td>
<td>38.44</td>
</tr>
<tr>
<td>GL_{Aln,Pic}</td>
<td>62.81</td>
<td>59.66</td>
<td>82.34</td>
<td>41.67</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>64.51</td>
<td>59.37</td>
<td>81.50</td>
<td>40.04</td>
</tr>
<tr>
<td>MAF</td>
<td>50.08</td>
<td>56.90</td>
<td>78.15</td>
<td>33.00</td>
</tr>
<tr>
<td>GL_{Aln,MAF}</td>
<td>64.38</td>
<td>60.03</td>
<td>81.91</td>
<td>40.19</td>
</tr>
</tbody>
</table>

**Table 5** TC scores obtained by the pipeline $\mathcal{P}_{U,V}$.

![Alignment Example](alignment_example.png)

**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.

---

\[\text{The score of column } i \text{ is } 0.667:\]

\[CS[i] = \frac{B[F_1, A_2] + B[F_1, F_3] + B[A_2, F_3]}{C_3^2} = \frac{(-2) + (-2) + 6}{3} = 0.667\]

---

**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 $C_{U}^{Ral}$ on all possible MSAs, where $U$ is the alignment tool used to do the realignment:

$C_{U}^{Ral}$ has two classes $1$ and $0$, and an MSA $M$ is in class $1$ if it is better to realign the reliable regions (i.e., if the MSA constructed by realigning $M$’s reliable regions has a TC score higher than the one constructed by realigning $M$’s unreliable regions); otherwise, $M$ 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 $C_{U}^{mw}$, where $U$ is the tool used to realign the unreliable regions.

Given an MSA $M$, let $M_i$ denote the alignment obtained by using $U$ to realign all the unreliable regions with width no smaller than $i$. The classification $C_{U}^{mw}$ has four classes $2$, $10$, $20$, $30$, and $M$ is in class $i$ if $M_i$ has the maximum TC score among those of $M_2$, $M_{10}$, $M_{20}$ and $M_{30}$.

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

18
To evaluate our realignment approach, we have implemented the pipelines $\mathcal{P}_U^{\text{Ral}}, \mathcal{P}_Q^{\text{RL}}, \mathcal{P}_{\text{NSA}}^{\text{MAF}}$ and $\mathcal{P}_{\text{MAF}}^{\text{MAF}}$. 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 $\mathcal{P}_U^{\text{Ral}}$, we compare the alignment accuracy of $U$ and that of the combination of $\mathcal{P}_U^{\text{Ral}}$ and $U$, denoted by $\mathcal{P}_U^{\text{Ral}} \circ U$, which works as follows:

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

We have implemented the pipeline $\mathcal{P}_U^{\text{Ral}} \circ U$ for the tools QuickProbs ($Q$), GL-Probs ($GL$), 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, $\mathcal{P}_Q^{\text{Ral}} \circ Q$ has the highest TC scores for all four databases.
Chapter 2. MLProbs

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{C}_Q$</td>
<td>87.35</td>
<td>93.64</td>
<td>90.39</td>
</tr>
<tr>
<td>$\mathcal{C}_G$</td>
<td>88.95</td>
<td>84.07</td>
<td>86.44</td>
</tr>
<tr>
<td>$\mathcal{C}_M$</td>
<td>81.27</td>
<td>84.65</td>
<td>82.93</td>
</tr>
<tr>
<td>$\mathcal{C}_M$</td>
<td>90.85</td>
<td>87.65</td>
<td>89.22</td>
</tr>
</tbody>
</table>

Table 6  Testing results for the classifiers $\mathcal{C}_U$

<table>
<thead>
<tr>
<th></th>
<th>BAliBASE</th>
<th>OXBench-X</th>
<th>OXBench</th>
<th>SABMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuickProbs</td>
<td>65.41</td>
<td>59.44</td>
<td>81.77</td>
<td>40.65</td>
</tr>
<tr>
<td>$\mathcal{C}_Q$</td>
<td>65.42</td>
<td>59.74</td>
<td>81.95</td>
<td>40.73</td>
</tr>
<tr>
<td>GLProbs</td>
<td>62.09</td>
<td>59.34</td>
<td>81.93</td>
<td>41.22</td>
</tr>
<tr>
<td>$\mathcal{C}_G$</td>
<td>62.12</td>
<td>59.49</td>
<td>82.24</td>
<td>41.41</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>64.51</td>
<td>59.37</td>
<td>81.50</td>
<td>40.04</td>
</tr>
<tr>
<td>$\mathcal{C}_M$</td>
<td>64.52</td>
<td>59.42</td>
<td>81.74</td>
<td>40.08</td>
</tr>
<tr>
<td>MAFFT</td>
<td>50.08</td>
<td>56.90</td>
<td>78.15</td>
<td>33.00</td>
</tr>
<tr>
<td>$\mathcal{C}_M$</td>
<td>50.42</td>
<td>56.99</td>
<td>78.30</td>
<td>33.07</td>
</tr>
</tbody>
</table>

Table 7  Average TC-scores of the pipelines $\mathcal{P}_U$.
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 \( \ln(s_1,s_2) \) denote the length of the optimal (pairwise) alignment \( \mathcal{M} \) of \( s_1 \) and \( s_2 \), and define \( \text{pid}(s_1,s_2) \), the percentage identity of \( s_1 \) and \( s_2 \), to be the percentage of columns of \( \mathcal{M} \) with identical amino acids, and \( \text{sc}(s_1,s_2) \), 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 \( \mathcal{M} \).

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

Consider any multiple sequence alignment \( \mathcal{M} \). Recall that the column score of any column \( C \) of \( \mathcal{M} \) 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 \( \mathcal{M} \), denoted as \( \text{pl}_\delta(\mathcal{M}) \) or simply \( \text{pl}(\mathcal{M}) \), to be the ratio between the total number of columns of \( \mathcal{M} \) with column scores greater than \( \delta \) and the total number of columns of \( \mathcal{M} \).

We use the following features for trainings the classifications \( \mathcal{C}_{\text{aln}}^{\text{U,V}}, \mathcal{C}_{\text{Ral}}^{\text{U}} \) and \( \mathcal{C}_{\text{mw}}^{\text{V}} \):

- \( \mathcal{C}_{\text{aln}}^{\text{U,V}} \): \( \text{av}(\text{pid}), \text{av}(\text{sc}), \text{av}(\ln), \text{pl} \) and \( \text{sz} \).
- \( \mathcal{C}_{\text{Ral}}^{\text{U}} \): \( \text{av}(\text{pid}), \text{av}(\text{sc}), \text{sd}(\text{sc}) \) and \( \text{pl} \).
- \( \mathcal{C}_{\text{mw}}^{\text{V}} \): \( \text{av}(\text{pid}), \text{sd}(\text{pid}), \text{av}(\ln) \) and \( \text{sz} \)

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\textsubscript{1}-score of C\textsubscript{Aln}\textsubscript{V} and C\textsubscript{Ral}\textsubscript{U} have been mentioned in the previous two sections. Here we only show the comprehensive performance of C\textsubscript{mw}\textsubscript{V} on four benchmarks in Figure 3.

![Figure 3](image)

**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 $P\textsubscript{Q}^{Ral} \circ P\textsubscript{Aln}$

Tables 5 and 7 show that $P\textsubscript{Aln}$ and $P\textsubscript{Ral}$ 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.
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 $\mathcal{Q}$.
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%.

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Superfamily</th>
<th>Twilight Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC-score</td>
<td>SP-score</td>
<td>TC-score</td>
</tr>
<tr>
<td>MLProbs</td>
<td>42.58</td>
<td>62.10</td>
<td>48.63</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>40.65</td>
<td>61.05</td>
<td>46.56</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>41.48</td>
<td>61.25</td>
<td>47.03</td>
</tr>
<tr>
<td>GLProbs</td>
<td>41.22</td>
<td>61.30</td>
<td>46.95</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>40.04</td>
<td>60.24</td>
<td>45.81</td>
</tr>
<tr>
<td>ProbCons</td>
<td>39.17</td>
<td>59.69</td>
<td>44.64</td>
</tr>
<tr>
<td>PicXAA</td>
<td>38.44</td>
<td>59.06</td>
<td>44.45</td>
</tr>
<tr>
<td>MAFFT</td>
<td>33.00</td>
<td>53.15</td>
<td>39.10</td>
</tr>
<tr>
<td>Muscle</td>
<td>33.47</td>
<td>54.51</td>
<td>39.13</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>35.47</td>
<td>55.02</td>
<td>41.43</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>38.63</td>
<td>59.53</td>
<td>44.11</td>
</tr>
</tbody>
</table>

| 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
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
2.6 Comparing MLProbs with other MSA tools

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

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>0 - 30%</th>
<th>30% - 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC-score</td>
<td>SP-score</td>
<td>TC-score</td>
</tr>
<tr>
<td>MLProbs</td>
<td>82.72</td>
<td>90.57</td>
<td>45.13</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>81.77</td>
<td>90.17</td>
<td>41.50</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>82.06</td>
<td>90.19</td>
<td>43.96</td>
</tr>
<tr>
<td>GLProbs</td>
<td>81.93</td>
<td>90.13</td>
<td>43.34</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>81.50</td>
<td>89.83</td>
<td>42.81</td>
</tr>
<tr>
<td>ProbCons</td>
<td>80.68</td>
<td>89.45</td>
<td>41.50</td>
</tr>
<tr>
<td>PicXAA</td>
<td>81.14</td>
<td>89.61</td>
<td>39.78</td>
</tr>
<tr>
<td>MAFFT</td>
<td>78.15</td>
<td>88.07</td>
<td>35.93</td>
</tr>
<tr>
<td>Muscle</td>
<td>80.67</td>
<td>89.50</td>
<td>40.95</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>79.99</td>
<td>88.91</td>
<td>37.39</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>81.68</td>
<td>89.97</td>
<td>41.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Improved %</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.80%</td>
<td>0.42%</td>
<td>2.66%</td>
<td>0.35%</td>
<td>0.35%</td>
<td>0.12%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>0 - 30%</th>
<th>30% - 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC-score</td>
<td>SP-score</td>
<td>TC-score</td>
</tr>
<tr>
<td>MLProbs</td>
<td>59.80</td>
<td>66.14</td>
<td>41.68</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>59.44</td>
<td>65.86</td>
<td>40.72</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>59.54</td>
<td>65.95</td>
<td>40.97</td>
</tr>
<tr>
<td>GLProbs</td>
<td>59.34</td>
<td>65.81</td>
<td>41.00</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>59.37</td>
<td>65.82</td>
<td>40.60</td>
</tr>
<tr>
<td>ProbCons</td>
<td>58.93</td>
<td>65.62</td>
<td>39.41</td>
</tr>
<tr>
<td>PicXAA</td>
<td>58.85</td>
<td>65.39</td>
<td>39.00</td>
</tr>
<tr>
<td>MAFFT</td>
<td>56.90</td>
<td>64.20</td>
<td>37.22</td>
</tr>
<tr>
<td>Muscle</td>
<td>56.83</td>
<td>64.39</td>
<td>36.32</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>58.05</td>
<td>64.81</td>
<td>39.10</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>59.27</td>
<td>65.71</td>
<td>39.80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Improved %</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.43%</td>
<td>0.29%</td>
<td>1.66%</td>
<td>0.73%</td>
<td>0.05%</td>
<td>-</td>
</tr>
</tbody>
</table>
Chapter 2. MLProbs

and OXBench-X with PID ≤ 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](image)

**Figure 6** Over 1356 families in BAliBASE, OXBench, OXBench-X and SABmark with PID ≤ 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
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

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

<table>
<thead>
<tr>
<th>Tool</th>
<th>BAiiBASE</th>
<th>OX Bench</th>
<th>OX Bench-X</th>
<th>SABMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLProbs</td>
<td>16.7325</td>
<td>0.8689</td>
<td>43.4129</td>
<td>0.8171</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>5.8985</td>
<td>0.1053</td>
<td>15.9758</td>
<td>0.0604</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>11.2323</td>
<td>0.3065</td>
<td>31.7131</td>
<td>0.2106</td>
</tr>
<tr>
<td>GLProbs</td>
<td>12.0890</td>
<td>0.2808</td>
<td>32.1543</td>
<td>0.1492</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>8.6230</td>
<td>0.1404</td>
<td>30.7912</td>
<td>0.0738</td>
</tr>
<tr>
<td>ProbCons</td>
<td>27.2583</td>
<td>0.4576</td>
<td>96.3181</td>
<td>0.2263</td>
</tr>
<tr>
<td>PicXAA</td>
<td>23.5338</td>
<td>0.4050</td>
<td>106.6262</td>
<td>0.2973</td>
</tr>
<tr>
<td>MAF FT</td>
<td>0.3191</td>
<td>0.1226</td>
<td>0.2345</td>
<td>0.1056</td>
</tr>
<tr>
<td>Muscle</td>
<td>1.5939</td>
<td>0.0339</td>
<td>2.7322</td>
<td>0.0715</td>
</tr>
<tr>
<td>ClustalOmega</td>
<td>1.1442</td>
<td>0.0313</td>
<td>0.8776</td>
<td>0.0466</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>16.1930</td>
<td>0.2618</td>
<td>73.7960</td>
<td>0.1112</td>
</tr>
</tbody>
</table>

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 $F$, we construct an MSA $M$ for $F$ using $U$. Then, we use the phylogenetic tree construction tool MEGA X [77] to construct a phylogenetic tree from $M$.

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.
2.7 Applications of MLProbs

<table>
<thead>
<tr>
<th>TreeFam ID</th>
<th>MLProbs</th>
<th>QuickProbs</th>
<th>PnpProbs</th>
<th>GLProbs</th>
<th>MSAProbs</th>
<th>ProbCons</th>
<th>PicXAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF105063</td>
<td>128</td>
<td>130</td>
<td>132</td>
<td>132</td>
<td>134</td>
<td>140</td>
<td>130</td>
</tr>
<tr>
<td>TF105073</td>
<td>112</td>
<td>112</td>
<td>112</td>
<td>114</td>
<td>116</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>TF105311</td>
<td>94</td>
<td>92</td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>92</td>
<td>102</td>
</tr>
<tr>
<td>TF105313</td>
<td>18</td>
<td>22</td>
<td>18</td>
<td>18</td>
<td>22</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>TF105629</td>
<td>108</td>
<td>112</td>
<td>110</td>
<td>108</td>
<td>114</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>TF105801</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>144</td>
<td>150</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>TF3133227</td>
<td>208</td>
<td>212</td>
<td>228</td>
<td>228</td>
<td>230</td>
<td>224</td>
<td>220</td>
</tr>
</tbody>
</table>

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 \( M \) 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.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>MLProbs</th>
<th>QuickProbs</th>
<th>PupProbs</th>
<th>GLProbs</th>
<th>MSAProbs</th>
<th>ProbCons</th>
<th>MAFFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1U24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5TFD</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>6HN6</td>
<td>11</td>
<td>12</td>
<td>23</td>
<td>22</td>
<td>19</td>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>5DFD</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2GDF</td>
<td>50</td>
<td>50</td>
<td>52</td>
<td>52</td>
<td>50</td>
<td>60</td>
<td>53</td>
</tr>
<tr>
<td>3GDF</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9GAF</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>13</td>
</tr>
</tbody>
</table>

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.
Chapter 2. MLProbs

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) MSAProsbs 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.
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 $P^i_A, i = 1, 2, 3, 4$, which respectively represents the different options of Part A mentioned above. We implemented $P^i_B, i = 1, 2$, where $i$ denoted the different clustering methods for guide tree generation in Part B, and $P^i_C, i = 1, 2$, 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 $P^i_A$, $P^i_B$ and $P^i_C$ 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.
3.3 Deep-learning-based decision-making method

<table>
<thead>
<tr>
<th></th>
<th>BAiLABASE</th>
<th>OXBenCh</th>
<th>SABMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}_A^1$</td>
<td>62.17</td>
<td>80.95</td>
<td>39.22</td>
</tr>
<tr>
<td>$\mathcal{P}_A^2$</td>
<td>61.06</td>
<td>81.73</td>
<td>39.00</td>
</tr>
<tr>
<td>$\mathcal{P}_A^3$</td>
<td>64.42</td>
<td>81.57</td>
<td>40.11</td>
</tr>
<tr>
<td>$\mathcal{P}_A^4$</td>
<td>64.27</td>
<td>81.56</td>
<td>40.62</td>
</tr>
<tr>
<td>Upper Bound $^1$</td>
<td>66.50</td>
<td>82.93</td>
<td>42.84</td>
</tr>
<tr>
<td>Max. Improvement $^2$</td>
<td>3.23%</td>
<td>1.47%</td>
<td>5.47%</td>
</tr>
<tr>
<td>$\mathcal{P}_B^1$</td>
<td>62.17</td>
<td>80.95</td>
<td>39.22</td>
</tr>
<tr>
<td>$\mathcal{P}_B^2$</td>
<td>62.42</td>
<td>80.93</td>
<td>39.20</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>62.74</td>
<td>80.96</td>
<td>39.28</td>
</tr>
<tr>
<td>Max. Improvement</td>
<td>0.51%</td>
<td>0.01%</td>
<td>0.15%</td>
</tr>
<tr>
<td>$\mathcal{P}_C^1$</td>
<td>62.17</td>
<td>80.95</td>
<td>39.22</td>
</tr>
<tr>
<td>$\mathcal{P}_C^2$</td>
<td>62.95</td>
<td>81.00</td>
<td>39.09</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>63.65</td>
<td>81.21</td>
<td>39.69</td>
</tr>
<tr>
<td>Max. Improvement</td>
<td>1.11%</td>
<td>0.26%</td>
<td>1.20%</td>
</tr>
</tbody>
</table>

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 $\mathcal{P}_A$.

3.3 Deep-learning-based decision-making method

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

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

$\mathcal{C}^{\text{Aln}}_{\mathcal{P}_A^i}$ has four classes $\mathcal{P}_A^1$, $\mathcal{P}_A^2$, $\mathcal{P}_A^3$ and $\mathcal{P}_A^4$. A protein family $\mathcal{F}$ is in class $\mathcal{P}_A^i$ if the MSA constructed by $\mathcal{P}_A^i$ 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 $\alpha$ ($\alpha = 512$ in our structure). When the length of a pair was insufficient, it was filled by gaps at the end to increase the length to $\alpha$. When the length of the sequence exceeded $\alpha$, only the leading fragments of length $\alpha$ 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].
3.3 Deep-learning-based decision-making method

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 $P_i$ is the precision and $R_i$ is the recall rate of class $i$, where $i = 1, 2, 3$ or 4 in this paper, the macro average precision ($P_{\text{macro}}$), recall ($R_{\text{macro}}$) can be calculated by Formula (1).

\[
\begin{align*}
P_{\text{macro}} &= \frac{1}{N} \sum_{i=1}^{N} P_i \\
R_{\text{macro}} &= \frac{1}{N} \sum_{i=1}^{N} R_i
\end{align*}
\]

(1)

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

The macro average $F_1$-score ($F_{\text{macro}}$) is equal to the harmonic average of $P_{\text{macro}}$ and $R_{\text{macro}}$.

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.


<table>
<thead>
<tr>
<th>Models</th>
<th>$P_{\text{macro}}$</th>
<th>$R_{\text{macro}}$</th>
<th>$F_{\text{macro}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>87.13</td>
<td>86.69</td>
<td>86.91</td>
</tr>
<tr>
<td>CNN + RNN</td>
<td>86.90</td>
<td>88.13</td>
<td>87.51</td>
</tr>
<tr>
<td>CNN + LSTM</td>
<td>87.13</td>
<td>87.71</td>
<td>87.41</td>
</tr>
<tr>
<td><strong>CNN + BiLSTM</strong></td>
<td><strong>87.70</strong></td>
<td><strong>88.68</strong></td>
<td><strong>88.19</strong></td>
</tr>
<tr>
<td>CNN + GRU</td>
<td>87.93</td>
<td>86.71</td>
<td>87.32</td>
</tr>
<tr>
<td>CNN + BiGRU</td>
<td>88.23</td>
<td>87.89</td>
<td>88.06</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P^1_A$</td>
<td>82.97</td>
<td>96.90</td>
<td>89.39</td>
</tr>
<tr>
<td>$P^2_A$</td>
<td>79.46</td>
<td>92.83</td>
<td>85.62</td>
</tr>
<tr>
<td>$P^3_A$</td>
<td>95.44</td>
<td>77.81</td>
<td>85.73</td>
</tr>
<tr>
<td>$P^4_A$</td>
<td>92.93</td>
<td>87.18</td>
<td>89.96</td>
</tr>
<tr>
<td>Macro Avg.</td>
<td>87.70</td>
<td>88.68</td>
<td>88.19</td>
</tr>
</tbody>
</table>

**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 $\mathcal{P}_A^i$, 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.

![Confusion Matrix Example](image)

**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 $\mathcal{P}_A^i$ 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-
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 $512 \times 8$ 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 $4 \times 1$ matrix that represents the final category.
3.4 Decision-making model-based progressive alignment method

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

**Decision-making of the posterior probability matrix calculation**

Each pair \( x, y \) in \( \mathcal{F} \) is inseted into the model with the highest accuracy described in the previous section to get a label (say, \( \text{label}_{x,y} \)). 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 \( \mathcal{F} \) 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 \( \mathcal{C}^{Aln}_{\mathcal{F}} \), where \( i = 1, 2, 3 \) or 4, the dominate proportion of predicted labels can be calculated using the following Formula (2).

\[
\text{dominate\_proportion} = \arg\max_i \left( \frac{PLP_i}{TLP_i} \right)
\]

(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 \( \mathcal{F} \) and the maximum sum is saved as \( \text{Prob}(x, y) \). 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.

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

(3)

where \(x.\text{Length}, y.\text{Length}\) 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.

\[
\text{Distance}[A \cup B][C] = \frac{|A| \times \text{Distance}[A][C] + |B| \times \text{Distance}[A][C]}{|A| + |B|}
\]

(4)

where \(|A|, |B|\) and \(|C|\) represent the weight of clusters A, B and C.
3.4 Decision-making model-based progressive alignment method

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](image)

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).

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

(5)

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

47
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 $\mathcal{F}$.

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

![Figure 15](image)

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
Comparing DLPAlign with other MSA tools

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 ClustalΩ. 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.

<table>
<thead>
<tr>
<th></th>
<th>All (386)</th>
<th>RV11 (38)</th>
<th>RV12 (44)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>SP</td>
<td>TC</td>
</tr>
<tr>
<td>DLPAlign</td>
<td>65.47</td>
<td>89.57</td>
<td>47.67</td>
</tr>
<tr>
<td>MLProbs</td>
<td>65.80</td>
<td>89.50</td>
<td>48.14</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>65.41</td>
<td>89.41</td>
<td>46.93</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>62.46</td>
<td>88.75</td>
<td>45.15</td>
</tr>
<tr>
<td>GLProbs</td>
<td>62.09</td>
<td>88.84</td>
<td>44.68</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>64.51</td>
<td>89.09</td>
<td>44.40</td>
</tr>
<tr>
<td>ProbCons</td>
<td>61.89</td>
<td>88.31</td>
<td>40.89</td>
</tr>
<tr>
<td>PicXAA</td>
<td>59.97</td>
<td>87.84</td>
<td>46.64</td>
</tr>
<tr>
<td>MAFFT</td>
<td>50.08</td>
<td>82.24</td>
<td>28.23</td>
</tr>
<tr>
<td>Muscle</td>
<td>53.17</td>
<td>84.33</td>
<td>32.06</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>56.20</td>
<td>83.97</td>
<td>36.22</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>60.68</td>
<td>87.78</td>
<td>45.69</td>
</tr>
</tbody>
</table>

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
Chapter 3. DLPAlign

<table>
<thead>
<tr>
<th></th>
<th>All (395)</th>
<th>0 - 30% (63)</th>
<th>30% - 100% (332)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>SP</td>
<td>TC</td>
</tr>
<tr>
<td>DLPAlign</td>
<td>82.52</td>
<td>90.42</td>
<td>44.16</td>
</tr>
<tr>
<td>MLProbs</td>
<td>82.72</td>
<td>90.57</td>
<td>45.13</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>81.77</td>
<td>90.17</td>
<td>41.50</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>82.06</td>
<td>90.19</td>
<td>43.96</td>
</tr>
<tr>
<td>GLProbs</td>
<td>81.93</td>
<td>90.13</td>
<td>43.34</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>81.50</td>
<td>89.83</td>
<td>42.81</td>
</tr>
<tr>
<td>ProbCons</td>
<td>80.68</td>
<td>89.45</td>
<td>41.50</td>
</tr>
<tr>
<td>PicXAA</td>
<td>81.14</td>
<td>89.61</td>
<td>39.78</td>
</tr>
<tr>
<td>MAFFT</td>
<td>78.15</td>
<td>88.07</td>
<td>35.93</td>
</tr>
<tr>
<td>Muscle</td>
<td>80.67</td>
<td>89.50</td>
<td>40.95</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>79.99</td>
<td>88.91</td>
<td>37.39</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>81.68</td>
<td>89.97</td>
<td>41.06</td>
</tr>
</tbody>
</table>

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
### 3.5 Comparing DLPAlign with other MSA tools

<table>
<thead>
<tr>
<th></th>
<th>All (423)</th>
<th>Superfamily (315)</th>
<th>Twilight Zone (108)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>SP</td>
<td>TC</td>
</tr>
<tr>
<td>DLPAlign</td>
<td>42.59</td>
<td>61.82</td>
<td>48.37</td>
</tr>
<tr>
<td>MLProbs</td>
<td>42.58</td>
<td>62.10</td>
<td>48.63</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>40.65</td>
<td>61.05</td>
<td>46.56</td>
</tr>
<tr>
<td>PnpProbs</td>
<td>41.48</td>
<td>61.25</td>
<td>47.03</td>
</tr>
<tr>
<td>GLProbs</td>
<td>41.22</td>
<td>61.30</td>
<td>46.95</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>40.04</td>
<td>60.24</td>
<td>45.81</td>
</tr>
<tr>
<td>ProbCons</td>
<td>39.17</td>
<td>59.69</td>
<td>44.64</td>
</tr>
<tr>
<td>PieXAA</td>
<td>38.44</td>
<td>59.06</td>
<td>44.45</td>
</tr>
<tr>
<td>MAFFT</td>
<td>33.00</td>
<td>53.15</td>
<td>39.10</td>
</tr>
<tr>
<td>Muscle</td>
<td>33.47</td>
<td>54.51</td>
<td>39.13</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>35.47</td>
<td>55.02</td>
<td>41.43</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>38.63</td>
<td>59.53</td>
<td>44.11</td>
</tr>
</tbody>
</table>

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

**Figure 16** Over 711 families in BAliBASE, OXBench and SABmark with PID ≤ 30%. DLPAlign gets the best average TC-score 47.17 and improves about 2.8% over the second best one which is 45.89.
Chapter 3. DLPAlign

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.
3.5 Comparing DLPAlign with other MSA tools

<table>
<thead>
<tr>
<th></th>
<th>BAliBASE</th>
<th>OXBench</th>
<th>SABMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLPAlign</td>
<td>38.10</td>
<td>0.84</td>
<td>0.40</td>
</tr>
<tr>
<td>QuickProbs</td>
<td>6.10</td>
<td>0.11</td>
<td>0.06</td>
</tr>
<tr>
<td>PnPProbs</td>
<td>13.30</td>
<td>0.27</td>
<td>0.21</td>
</tr>
<tr>
<td>GLProbs</td>
<td>13.63</td>
<td>0.23</td>
<td>0.15</td>
</tr>
<tr>
<td>MSAProbs</td>
<td>9.49</td>
<td>0.14</td>
<td>0.07</td>
</tr>
<tr>
<td>ProbCons</td>
<td>33.98</td>
<td>0.49</td>
<td>0.24</td>
</tr>
<tr>
<td>PicXAA</td>
<td>29.82</td>
<td>0.49</td>
<td>0.35</td>
</tr>
<tr>
<td>MAFFT</td>
<td>0.33</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td>Muscle</td>
<td>1.69</td>
<td>0.04</td>
<td>0.08</td>
</tr>
<tr>
<td>ClustalΩ</td>
<td>0.97</td>
<td>0.03</td>
<td>0.04</td>
</tr>
<tr>
<td>ProbAlign</td>
<td>21.92</td>
<td>0.27</td>
<td>0.12</td>
</tr>
</tbody>
</table>

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 ≤ 30%, 711 such families in all) and medium similarity (with 30% < PID ≤ 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
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. 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 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.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>DLPAlign</th>
<th>QuickProbs</th>
<th>PnpProbs</th>
<th>GLProbs</th>
<th>MSAProbs</th>
<th>PicXAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>6YI3</td>
<td>10</td>
<td>16</td>
<td>12</td>
<td>12</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>6VYO</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6W9C</td>
<td>20</td>
<td>31</td>
<td>24</td>
<td>22</td>
<td>23</td>
<td>32</td>
</tr>
<tr>
<td>6W61</td>
<td>13</td>
<td>17</td>
<td>13</td>
<td>13</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

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

1They are all available only from April this year, and relevant references have not been published.
Figure 19 The predicted protein secondary structures by DLPAlign, QuickProbs, PnpProbs, GLProbs, MSAProbs and PicXAA on protein with PDB ID 6W61.
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. ML-Probs 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 $P_{U,V}^{\text{Aln}}$, which was mentioned in Chapter 2 for MSA construction.

An MSA is very similar to a 2D picture; they both have hierarchical 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


References


