MLProbs: A Data-Centric Pipeline for Better Multiple Sequence Alignment

Mengmeng Kuang, Yong Zhang, Tak-Wah Lam, and Hing-Fung Ting

Abstract—In this paper, we explore using the data-centric approach to tackle the Multiple Sequence Alignment (MSA) construction problem. 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 explores using classification models trained from existing benchmark data to guide the construction. We identified two simple classifications to help us choose a better alignment tool and determine whether and how much to carry out realignment. We show that shallow machine-learning algorithms suffice to train sensitive models for these classifications. Based on these models, we implemented a new multiple sequence alignment pipeline, called MLProbs. Compared with 10 other popular alignment tools over four benchmark databases (namely, BAliBASE, OXBench, OXBench-X and SABMark), MLProbs consistently gives the highest TC score. More importantly, MLProbs shows non-trivial improvement for protein families with low similarity; in particular, when evaluated against the 1,356 protein families with similarity $\leq$ 50%, MLProbs achieves a TC score of 56.93, while the next best three tools are in the range of [55.41, 55.91] (increased by more than 1.8%). We also compared the performance of MLProbs and other MSA tools in two real-life applications – Phylogenetic Tree Construction Analysis and Protein Secondary Structure Prediction – and MLProbs also had the best performance. In our study, we used only shallow machine-learning algorithms to train our models. It would be interesting to study whether deep-learning methods can help make further improvements, so we suggest some possible research directions in the conclusion section.

Index Terms—Multiple sequence alignment, protein family, machine learning, classification

1 Introduction

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

MSA construction is common in a lot of biological analyses and post-genomic research. Biologists sometimes need to construct MSA for hundreds of sequences, each with hundreds or more residues. To help handle these daunting and tedious tasks, a lot of research effort has been devoted to automating the construction process. Since the early 80s, the problem has been tackled using the algorithm-centric approach: algorithms are designed to solve a combinatorial optimization problem, in which every possible column of residues is associated with a column score, and the objective is to find the alignment with the largest sum of column scores. 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], [6], the divide-and-conquer method [7], iterative refinement [8], Gibbs sampling [9], consensus [10], and the progressive method [11]).

We now have reliable MSA software tools that can construct good alignments for protein families with high similarity. However, for those with low similarity, no existing tools can consistently construct satisfactory alignments, so for them, biologists usually need some external information, such as the 3D crystal structure of the sequences, to determine their correct alignments. It has always been a challenge for the research community to develop new tools that can construct better MSAs for them. Even a small improvement can have a significant impact because correct positioning for even a few more key residues in the alignment can save biologists considerable time and effort by shifting their attention quickly to the correct regions for downstream analysis.

It seems that after decades of research, we have exhausted almost all the mathematical and algorithmic techniques that are applicable to the MSA problem, and in the last few years, no fundamentally new technique has been proposed. In striving for a breakthrough, we noted that the algorithm-centric approach was not suitable to tackle the MSA problem because the problem is data-centric in nature. An MSA is basically a statement of homologue, 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

- *Mengmeng Kuang, Tak-Wah Lam, and Hing-Fung Ting are with The University of Hong Kong, Pok Fu Lam, Hong Kong. E-mail: {mmkuang, twlam, hfting}@cs.hku.hk.*
- *Yong Zhang is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: zhangyong@siat.ac.cn.*

Manuscript received 11 Apr. 2020; revised 12 Oct. 2021; accepted 27 Jan. 2022. Date of publication 4 Feb. 2022; date of current version 3 Feb. 2023. This work was supported by Hong Kong under Grant GRF-17208019. (Corresponding author: Hing-fung Ting.) Digital Object Identifier no. 10.1109/TCBB.2022.3148382

thousands or even millions of years. The algorithm-centric approach attempts to capture these events with abstract models, from which it formulates feasible computational problems. 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 the 190 possible pairs of the 20 amino acids, and the score given to a pair estimates the prior probability that the pair has the same ancestor. Obviously, these 190 scores are far from enough to capture the long evolutionary history of tens of thousands of protein families.

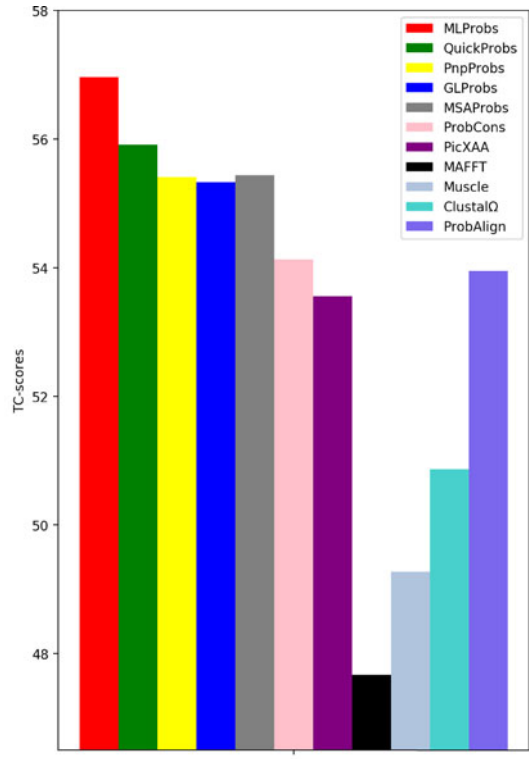
This paper explores a different approach, namely the data-centric approach, to tackle the MSA construction problem. Instead of relying on abstract models, we study how to apply machine-learning algorithms to learn models from protein family data and use them to help construct better MSAs. Using data-centric methods to tackle difficult problems in computer science is rapidly gaining popularity because of recent advances in machine learning. Machine learning has also been applied in bioinformatics, and we have witnessed many successes [12], [13], [14], [15], [16].

There is still no notable research on applying machine learning to the MSA construction problem, though some works have explored using existing data and knowledge to improve alignment accuracy. For example, the knowledge-base multiple sequence alignment technique uses existing knowledge databases, such as SWISSPROT, GENBANK or HOMSTRAD, to construct more reliable sequence alignments [17], [18]. Homology extension is another strategy, which uses database searches to collect homologous sequences to gain evolutionary information for improving alignment accuracy [19], [20].

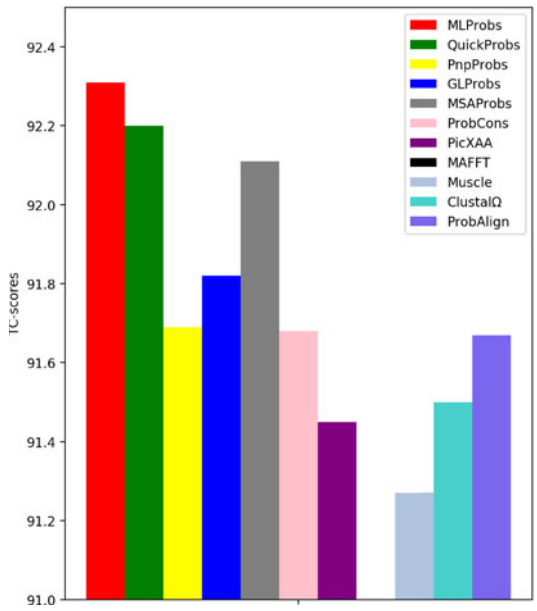
1.1 Our Contributions

The critical factor for the success of a machine-learning application is whether there are enough high-quality data to train effective models. The training data used in this paper are the 6000+ protein families obtained from the BENCH website (https://www.drive5.com/bench) [21]. Since this number of families may not be enough for training deep machine-learning models, such as CNN or LSTM, we applied shallow machine learning algorithms, like Random Forest and SVM, for the training and used the resulting models to help construct MSAs. We evaluated the alignment accuracy of our methods using the latest versions of the golden benchmark databases: SABmark v1.65 [22], BAliBASE v3 [23], and OXBench v1.3 and its extension OXBench-X [24].

We provide details of our methods and their implementation in Section 2 and Section 3, respectively. Using our methods, we built a pipeline, called MLProbs, which applies our methods to help construct MSAs. We compared the alignment accuracy of MLProbs and 10 other popular MSA tools: (1) PnpProbs[25], (2) QuickProbs [26], (3) GLProbs [27], (4) PicXAA [28], (5) ProbCons [29], (6) MSAProbs [30], (7) MAFFT [4], (8) Muscle [31], (9) ClustalΩ [32], and (10) ProbAlign [33]. Tables 6, 7, 8 and 9 (in Section 4) show the alignment accuracy of the MSAs constructed using the tools for families in SABMark, BAliBASE,



(a) Over 1,356 families in BAliBASE, OXBench, OXBench-X and SABMark with PID$\leq$50%. MLProbs has the best TC score 56.93, and it improves the 2nd best one, which is 55.91, by more than 1.65%.



(b) Over all families with PID$>$ 50%.

Fig. 1. Comparison of the TC scores over families in SABMark, BAliBASE, OXBench and OXBench-X.

OXBench and OXBench-X, respectively. In all four databases, MLProbs achieved the highest TC score of all the tools. More importantly, MLProbs performed particularly well for families with low similarity. To provide an overall picture, Fig. 1a compares the TC scores of the tools for all families in SABMark, BAliBASE, OXBench and OXBench-X with PID $\leq$ 50% (there are 1,356 such families in total). MLProbs had the highest TC score, and its improvement

over the second-best tool was more than double that of the second-best tool's improvement over the third-best tool. As we mentioned above, obtaining better alignments for families with low similarity is always a challenge for research in MSA construction, so MLProbs' improvement should have a significant impact.

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

The rest of the paper is organized as follows. In Section 2, we describe our methods, and in Section 3, we provide details of their implementation. In Section 4, we describe the implementation of MLProbs and compare its performance with 10 other popular MSA tools. We also applied MLProbs to two real-life applications, namely a phylogenetic tree construction and protein secondary structure prediction, and evaluated its effectiveness. Finally, in Section 5, we provide a conclusion and propose future research directions.

2 Methods

In this section, we explore how to use machine learning to help answer the following two questions: (i) how to choose better tools to align a family, and (ii) given an MSA for the input family, whether and how much to carry out realignment to improve its accuracy. We describe our methods in this section and provide details of their implementation in the following section.

2.1 How to Choose Better Tools

Over the past few decades, many MSA construction tools have been designed and implemented, and each has its own strengths and weaknesses. For example, progressive alignment tools work better in general, but 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. For example, the web-based MSA service SeqAna [34] allows users submit a small alignment sample as a reference for SeqAna to automatically identify the best tool to align their large set of sequences. In this paper, we propose applying machine learning to help make the right decision.

Our study focuses on a case in which there are only two tools to choose from. Consider any two MSA tools: U and V. We define the following binary classification $\mathcal{C}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$ for protein families:

> *$\mathcal{C}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$ has two classes* 0 *an* 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 use the popular *TC score* [35] to measure the suitability of an alignment. The TC score of the 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 benchmark databases). We note that different implementations may have slightly different ways of computing the TC scores (e.g., some do not consider columns with gapped entries). In this paper, we use qscore (http://www.drive5.com/qscore) to compute all the TC scores.

The key concern is how to build an accurate model for $\mathcal{C}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$, which can naturally outperform U and V. We use a machine-learning algorithm to construct a model (or classifier) for $\mathcal{C}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$. Then we construct the alignments using the pipeline $\mathcal{P}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$, which, given an input family $\mathcal{F}$, first uses the classifier to determine to which class in $\mathcal{C}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$ $\mathcal{F}$ 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 implemented $\mathcal{P}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$ for various MSA tools U and V and evaluated their performance. We report our results in Section 3.

2.2 How to Get a 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 [36], [37]: 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. Many algorithmic techniques have been proposed for determining unreliable regions [36], [37], [38], [39], [40], [41]. Ours is based on column scores. The *score of a column* is defined to be the average score of the amino acid pairs[1] in the column; a column is unreliable if its score is smaller than a predetermined threshold. Unreliable regions are simply blocks of maximal consecutive of unreliable columns. Intuitively, we should realign them. However, we observed that there are two decisions that need to be made correctly to make our realignment procedure effective.

2.2.1 To Realign or Not to Realign?

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

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

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

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

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

2.2.2 How Wide Should an Unreliable Region be?

Realigning an unreliable region with only one or two columns is unlikely to improve the whole alignment. Therefore, we should skip unreliable regions that are too narrow, and realign only 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 determine the best one, we define the following classification $\mathcal{C}^{\texttt{mw}}_{\texttt{U}}$, where U is the tool used to realign the unreliable regions.

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

The classifications $\mathcal{C}^{\texttt{mw}}_{\texttt{U}}$ and $\mathcal{C}^{\texttt{Ral}}_{\texttt{U}}$ suggest a natural pipeline $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}}$ for the realignment step. Given any MSA $\mathcal{M}$, we first determine the class in $\mathcal{C}^{\texttt{Ral}}_{\texttt{U}}$ to which $\mathcal{M}$ belongs. If $\mathcal{M}$ 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 $\mathcal{C}^{\texttt{mw}}_{\texttt{U}}$ to which $\mathcal{M}$ belongs, and then return the alignment obtained using U to realign all the unreliable regions of $\mathcal{M}$ with a width no smaller than i.

2.3 Training of the Classifiers

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

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

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

Consider any multiple sequence alignment $\mathcal{M}$. 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 $\delta = 1.0$). Define the *peak-length ratio* of $\mathcal{M}$, denoted as $\texttt{pl}_\delta(\mathcal{M})$ or simply $\texttt{pl}(\mathcal{M})$ to be the ratio between the total number of columns of $\mathcal{M}$ with column scores greater than δ and the total number of columns of $\mathcal{M}$.

We use the following features to train the classifications: $\mathcal{C}^{\texttt{aln}}_{\texttt{U,V}}$, $\mathcal{C}^{\texttt{Ral}}_{\texttt{U}}$ and $\mathcal{C}^{\texttt{mw}}_{\texttt{U}}$:

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

We use the Random Forest algorithm in the Python machine learning library scikit-learn [42] to train the classifiers, with all the parameters set to default in all our training (e.g., the number of trees in the forest is 100, the minimum number of samples required to be a leaf node is 1, etc.). The data we use are obtained from the BENCH website (https://www.drive5.com/bench), which has a total of 6,592 protein families. For the training, we use five-fold cross-validation to avoid overfitting.[2] Note that BENCH contains both DNA and protein families; our experiments use only the protein families. Following is a summary of the data we use.

- We used 6,592 protein families, 4214 of which have more than two sequences.
- There were 151,340 protein sequences in total.
- The minimum, average and maximum length of the sequences were 24, 210.7 and 7,923, respectively.

We provide details of the training and testing of the classifiers, as well as the implementation of the pipelines, in the next section.

3 Implementation

To test the effectiveness of our methods, we implemented $\mathcal{P}^{\texttt{Aln}}_{\texttt{U,V}}$ and $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}}$ for various MSA tools and tested how much improvement they made. We also tried to find the best $\mathcal{P}^{\texttt{Aln}}_{\texttt{U,V}}$ and $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}}$ for implementing MLProbs.

3.1 Finding the Best $\mathcal{P}^{\texttt{Aln}}_{\texttt{U,V}}$

We implemented the pipelines $\mathcal{P}^{\texttt{Aln}}_{\texttt{U,V}}$ for various tools U and V. We had particular interest in the tool PnpProbs. To construct an MSA for input family $\mathcal{F}$, PnpProbs first computes the average PID of $\mathcal{F}$; 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 tried very hard to find other statistical conditions and algorithmic methods to help us make a better decision about the choice of P or NP, but all our efforts were in vain. We thought it would be interesting to find out whether our data-centric method could help us make better decisions, or more precisely, whether our trained classifier $\mathcal{C}^{\texttt{Aln}}_{\texttt{P,NP}}$ had better precision and sensitivity. This was indeed the case. As can be seen from Table 1, the precision and sensitivity of $\mathcal{C}^{\texttt{Aln}}_{\texttt{P,NP}}$ are significantly higher than those of the 18%-rule of PnpProbs.

Based on $\mathcal{C}^{\texttt{Aln}}_{\texttt{P,NP}}$, we implemented the pipeline $\mathcal{P}^{\texttt{Aln}}_{\texttt{P,NP}}$, and we also implemented the pipelines $\mathcal{P}^{\texttt{Aln}}_{\texttt{Pnp,Q}}$, $\mathcal{P}^{\texttt{Aln}}_{\texttt{GL,MSA}}$, $\mathcal{P}^{\texttt{Aln}}_{\texttt{GL,Pic}}$,

2. In our evaluation of MLProbs, the TC and SP scores are obtained by averaging results from five cross-validation runs.

TABLE 1
Comparing $\mathcal{C}^{\texttt{Aln}}_{\texttt{P,NP}}$ and PnpProbs's 18%-Rule

	Precision	Sensitivity
$\mathcal{C}^{\texttt{Aln}}_{\texttt{P,NP}}$	86.41	92.78
The 18%-rule	79.43	80.80

TABLE 2
Testing Results for the Classifiers $\mathcal{C}^{\texttt{Aln}}_{\texttt{U,V}}$

	Precision	Sensitivity	F-index
$\mathcal{C}^{\texttt{Aln}}_{\texttt{P,NP}}$	86.41	92.78	89.48
$\mathcal{C}^{\texttt{Aln}}_{\texttt{Pnp,Q}}$	81.89	88.74	85.18
$\mathcal{C}^{\texttt{Aln}}_{\texttt{GL,MSA}}$	80.89	91.36	85.81
$\mathcal{C}^{\texttt{Aln}}_{\texttt{GL,Pic}}$	86.09	91.22	88.58
$\mathcal{C}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$	80.27	91.45	85.49

TABLE 3
TC Scores Obtained by the Pipeline $\mathcal{P}^{\texttt{ALn}}_{\texttt{U,V}}$

	BAliBASE	OXBench-X	OXBench	SABMark
P	62.17	59.53	82.18	41.39
NP	60.74	57.77	82.05	41.28
$\mathcal{P}^{\texttt{Aln}}_{\texttt{P,NP}}$	63.22	59.57	82.33	42.11
PnpProbs	62.46	59.54	82.06	41.48
QuickProbs	65.41	59.44	81.77	40.65
$\mathcal{P}^{\texttt{Aln}}_{\texttt{Pnp,Q}}$	64.86	59.93	82.26	41.69
GLProbs	62.09	59.34	81.93	41.22
MSAProbs	64.51	59.37	81.50	40.04
$\mathcal{P}^{\texttt{Aln}}_{\texttt{GL,MSA}}$	64.07	59.69	82.16	41.82
GLProbs	62.09	59.34	81.93	41.22
PicXAA	60.97	58.85	81.14	38.44
$\mathcal{P}^{\texttt{Aln}}_{\texttt{GL,Pic}}$	62.74	59.68	82.24	41.99
MSAProbs	64.51	59.37	81.50	40.04
MAFFT	50.08	56.90	78.15	33.00
$\mathcal{P}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$	64.44	59.95	81.70	40.13

and $\mathcal{P}^{\texttt{Aln}}_{\texttt{MSA,MAF}}$ for PnpProbs(Pnp), QuickProbs (Q), GLProbs (GL), MSAProb (MSA), PicXAA (Pic), and MAFFT (MAF). Table 2 summarizes the accuracy of the classifiers, and Table 3 compares the TC scores of the tools and pipelines for the four benchmark databases.

The TC scores obtained by $\mathcal{P}^{\texttt{Aln}}_{\texttt{P,NP}}$ were consistently and significantly higher than those obtained by PnpProbs. Moreover, among the 15 tools and pipelines in the table, $\mathcal{P}^{\texttt{Aln}}_{\texttt{P,NP}}$ had the highest TC scores for OXBench-X, OXBench and SABMark. MLProbs is based on $\mathcal{P}^{\texttt{Aln}}_{\texttt{P,NP}}$, and with additional help from the realignment methods given in the next subsection, it achieved the best alignment accuracy for all four databases.

3.2 Finding the Best $\mathcal{P}^{\texttt{Ral}}_{T}$

To evaluate our realignment approach, we implemented the pipelines $\mathcal{P}^{\texttt{Ral}}_{\texttt{Q}}$, $\mathcal{P}^{\texttt{Ral}}_{\texttt{GL}}$, $\mathcal{P}^{\texttt{Ral}}_{\texttt{MSA}}$ and $\mathcal{P}^{\texttt{Ral}}_{\texttt{MAF}}$. Our preliminary study showed that to get better results, we needed 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 as a maximal consecutive of reliable columns. We define a fuzzy region, an unreliable region and a confusing region similarly. Our refinement identifies some of the regions in which we do not have much confidence, namely messy regions (because the subsequences in these regions are so different that we have little hope of making any improvement) and the fuzzy regions (because it is hard to decide whether they are reliable or not). Our realignment procedure ignores the fuzzy and the messy regions, focusing instead on the reliable and unreliable regions.

To measure the effectiveness of a realignment pipeline $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}}$, we compare the alignment accuracy of U and that of the combination of $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}}$ and U, denoted by $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}} \circ \texttt{U}$, 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 $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}}$ to realign the reliable/unreliable regions of $\mathcal{M}$.*

We implemented the pipeline $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}} \circ \texttt{U}$ for the tools QuickProbs (Q), GLProbs (GL), MSAProbs (MSA) and MAFFT(MAF). Table 4 summarizes the accuracy of the classifiers, and Table 5 summarizes the TC scores obtained by the pipelines for BAliBASE, OXBench-X, OXBench and SABMark. The realignment step improved the scores in all cases. Furthermore, $\mathcal{C}^{\texttt{Ral}}_{\texttt{Q}} \circ \texttt{Q}$ had the highest TC scores for all four databases.

TABLE 4
Testing of $\mathcal{C}^{\texttt{Ral}}_{\texttt{U}}$

	Precision	Sensitivity	F-index
$\mathcal{C}^{\texttt{Ral}}_{\texttt{Q}}$	88.21	91.71	89.93
$\mathcal{C}^{\texttt{Ral}}_{\texttt{GL}}$	87.50	81.31	84.29
$\mathcal{C}^{\texttt{Ral}}_{\texttt{MSA}}$	84.79	89.92	87.28
$\mathcal{C}^{\texttt{Ral}}_{\texttt{MAF}}$	91.82	83.91	87.69

TABLE 5
TC-Scores of the Pipelines $\mathcal{P}^{\texttt{Ral}}_{\texttt{U}} \circ \texttt{U}$

	BAliBASE	OXBench-X	OXBench	SABMark
QuickProbs	65.41	59.44	81.77	40.65
$\mathcal{P}^{\texttt{Ral}}_{\texttt{Q}} \circ \texttt{Q}$	65.50	59.74	81.89	40.73
GLProbs	62.09	59.34	81.93	41.22
$\mathcal{P}^{\texttt{Ral}}_{\texttt{GL}} \circ \texttt{GL}$	62.12	59.49	82.24	41.41
MSAProbs	64.51	59.37	81.50	40.04
$\mathcal{P}^{\texttt{Ral}}_{\texttt{MSA}} \circ \texttt{MSA}$	64.32	59.39	81.74	40.34
MAFFT	50.08	56.90	78.15	33.00
$\mathcal{P}^{\texttt{Ral}}_{\texttt{MAF}} \circ \texttt{MAF}$	50.22	56.76	78.22	32.83

4 Experiments and Evaluation of MLProbs

Tables 3 and 5 show that $\mathcal{P}^{\texttt{Aln}}_{\texttt{P,NP}}$ and $\mathcal{P}^{\texttt{Ral}}_{\texttt{Q}}$ were the top performers. We thought it would be interesting to find out how well their combination performed, so, we implemented and

TABLE 6
TC-Scores and SP-Scores on SABMark

	All		Superfamily		Twilight Zone	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**42.07**	**61.75**	**47.96**	**67.61**	**24.89**	**44.64**
QuickProbs	40.65	61.05	46.56	66.86	23.40	**44.11**
PnpProbs	**41.48**	**61.25**	**47.03**	66.84	**24.63**	43.94
GLProbs	41.22	61.30	46.95	**66.97**	23.86	43.74
MSAProbs	40.04	60.24	45.81	66.00	22.60	42.46
ProbCons	39.17	59.69	44.64	65.27	22.57	42.42
PicXAA	38.44	59.06	44.45	65.18	20.33	40.23
MAFFT	33.00	53.15	39.10	60.05	14.72	32.14
Muscle	33.47	54.51	39.13	61.30	16.96	34.70
ClustalΩ	35.47	55.02	41.43	61.70	18.10	35.55
ProbAlign	38.63	59.53	44.11	65.40	22.64	42.43
Improved %	1.42%	0.82%	2.00%	0.96%	1.06%	1.20%

tested the pipeline $\mathcal{P}_{\mathtt{Q}}^{\mathtt{Ral}} \circ \mathcal{P}_{\mathtt{P,NP}}^{\mathtt{Aln}}$, which, given any input family $\mathcal{F}$, first uses $\mathcal{P}_{\mathtt{P,NP}}^{\mathtt{Aln}}$ to construct an alignment of $\mathcal{F}$, and then uses $\mathcal{P}_{\mathtt{Q}}^{\mathtt{Ral}}$ to improve this alignment. We call this pipeline MLProbs.

4.1 Comparing MLProbs With Other MSA Tools

This section compares the performance of MLProbs and 10 other popular MSA tools: (1) PnpProbs, (2) QuickProbs, (3) GLProbs, (4) PicXAA, (5) ProbCons, (6) MSAProbs, (7) MAFFT, (8) Muscle, (9) ClustalΩ and (10) ProbAlign. We compare the accuracy of their alignments for families in the four benchmark databases SABMark, OXBench, OXBench-X and BaliBASE.

Table 6 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 of no more than 50%, and Twilight Zone contains different SCOP subsets with PID of no more than 25%. In addition to TC scores, we compared the SP scores of the alignments, which also measure the quality of alignments, though they are less commonly used than TC scores. For all three sets of families, no tools can obtained TC scores greater than 50%, so it is very difficult to construct good MSAs for them. Note that MLProbs had the best TC and SC scores in all cases. The last row of Table 6 shows the improved percentage of MLProbs' scores over the second best scores.

Table 7 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 11 tools can construct good alignments for its families; they all have TC scores greater than 85%. However, for RV11, the TC scores for 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, so their sizes are much larger. In fact, the average size of the families in OXBench is 8.33, while it is 122.49 for OXBench-X.

Table 8 shows our results for OXBench. Besides the complete set of families, the table shows the alignment accuracy for families with PID of no less than 30% and for those above 30%. Note in 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 second-best tool is nearly 2.7%.

TABLE 7
TC-Scores and SP-Scores for BAliBASE

	All		RV11		RV12	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**65.84**	**89.52**	**48.14**	**70.33**	**87.60**	**94.68**
QuickProbs	**65.41**	**89.41**	**46.93**	**69.59**	87.03	94.52
PnpProbs	62.46	88.75	45.15	68.80	87.25	94.79
GLProbs	62.09	88.84	44.68	69.27	**87.38**	**94.83**
MSAProbs	64.51	89.09	44.40	68.18	87.03	94.63
ProbCons	61.89	88.31	40.89	65.26	84.14	92.03
PicXAA	59.97	87.84	46.64	68.98	86.60	94.61
MAFFT	50.08	82.24	28.23	52.54	75.57	88.17
Muscle	53.17	84.33	32.06	57.15	58.90	90.26
ClustalΩ	56.20	83.97	36.22	59.01	79.38	90.60
ProbAlign	60.68	87.78	45.69	69.50	86.69	94.64
Improved %	0.67%	0.12%	2.5%	1.00%	0.2%	-0.16%

TABLE 8
TC-Scores and SP-Scores for OXBench

	All		0 - 30%		30% - 100%	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**82.67**	**90.55**	**45.13**	**66.56**	**89.86**	**95.13**
QuickProbs	81.77	90.17	41.50	64.50	88.35	94.46
PnpProbs	**82.06**	**90.19**	**43.96**	**66.33**	89.54	**95.01**
GLProbs	81.93	90.13	43.34	66.11	**89.55**	94.99
MSAProbs	81.50	89.83	42.81	65.22	89.08	94.78
ProbCons	80.68	89.45	41.50	64.50	88.35	94.46
PicXAA	81.14	89.61	39.78	63.16	89.32	94.93
MAFFT	78.15	88.07	35.93	60.68	86.40	93.53
Muscle	80.67	89.50	40.95	63.96	88.21	94.34
ClustalΩ	79.99	88.91	37.39	60.78	88.08	94.25
ProbAlign	81.68	89.97	41.06	63.80	89.39	94.93
Improved %	0.72%	0.38%	2.66%	0.18%	0.35%	0.12%

TABLE 9
TC-Scores and SP-Scores on OXBench-X

	All		0 - 30%		30% - 100%	
	TC-score	SP-score	TC-score	SP-score	TC-score	SP-score
MLProbs	**59.60**	**66.08**	40.95	**50.87**	**68.55**	73.36
QuickProbs	59.44	65.86	40.72	**50.74**	68.41	73.11
PnpProbs	**59.54**	**65.95**	**40.97**	50.56	**68.44**	73.32
GLProbs	59.34	65.81	**41.00**	50.53	68.14	73.14
MSAProbs	59.37	65.82	40.60	50.39	68.37	73.22
ProbCons	58.93	65.62	39.41	49.43	68.29	**73.39**
PicXAA	58.85	65.39	39.00	49.19	68.37	73.15
MAFFT	56.90	64.20	37.22	47.25	66.33	72.32
Muscle	56.83	64.39	36.32	47.70	66.66	72.40
ClustalΩ	58.05	64.81	39.10	48.67	67.14	72.55
ProbAlign	59.27	65.71	39.80	49.65	68.60	**73.41**
Improved %	0.10%	0.20%	-	0.26%	0.16%	-

The results for OXBench-X were quite different. As Table 9 shows, no tools can construct satisfactory alignments even for families with high similarity. This is not surprising because the families in OXBench-X are much larger. Even though MLProbs can still obtain better TC scores, its improvement is not as significant as we saw in the other benchmarks. One reason might be that 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).

Finally, for each of the four benchmark databases, we compared the average running time of MLProbs and other MSA tools for constructing an alignment. All the tools were run on a Dell desktop computer with four i5-6500 (3.20GHz) Intel cores and 7.6GB main memory. Table 10 shows the results.

TABLE 10
Average Running Time (in Seconds) for Constructing an MSA

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

Note that the running time of MLProbs comprises three main parts: (1) the time to get an alignment using PnpProbs, (2) the time to realign some regions using QuickProbs, and (3) the running times of the classifiers. The total running time of the classifiers is negligible (corroborated by the column for BAliBASE in Table 10, which shows that the running time of MLProbs roughly equals the sum of those of PnpProbs and QuickProbs). The difference between the running time of MLProbs and the sum of that of PnpProbs and QuickProbs is 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 size of the families in OXBench-X is very large, so those constructed by PnpProbs are not very reliable.

TABLE 11
The Unweighted RF-Distances for the Phylogenetic Trees Constructed

TreeFam ID(Num. of Seq.)	MLProbs	QuickProbs	PnpProbs	GLProbs	MSAProbs	ProbCons	PicXAA
TF105063(133)	**126**	130	132	132	134	140	130
TF105073(99)	**112**	112	112	112	114	116	124
TF105311(70)	**92**	92	94	94	94	92	102
TF105313(67)	**18**	22	18	18	22	22	18
TF105629(115)	**106**	112	110	110	108	114	120
TF105801(102)	**140**	140	140	140	144	150	148
TF313227(192)	**212**	212	228	228	230	224	220

4.2 Applications

4.2.1 Phylogenetic Tree Construction Analysis

A popular way to construct a phylogenetic tree for a protein family is first to construct an MSA for the family, and then to convert it to a phylogenetic tree. As remarked in [43], the quality of the constructed phylogenies depends to a large extent on the accuracy of the MSAs. Since our experiments show that MLProbs has the best alignment accuracy of all the tools, we expect the phylogenetic trees constructed from its alignments to be good. To verify this, we compared the quality of the phylogenetic trees constructed from the MSAs obtained by MLProbs, QuickProbs, PnpProbs, GLProbs, MSAProbs, ProbCons and PicXAA for families in TreeFam [44]. The phylogenetic trees were constructed as follows:

For each tool $\mathtt{U}$ *and each family* $\mathcal{F}$*, we construct an MSA* $\mathcal{M}$ *for* $\mathcal{F}$ *using* $\mathtt{U}$*. Then, we use the phylogenetic tree construction tool MEGA X [45] to construct a phylogenetic tree from* $\mathcal{M}$*.*

We measured the quality of a phylogenetic tree by its unweighted Robinson-Foulds (RF) distance [46] between the tree and the reference tree given in TreeFam; the smaller the distance the better the tree. We used the package DendroPy [47] to compute the distances. Table 11 summarizes our results. Note that for all families, the phylogenetic trees constructed by MLProbs alignments had the smallest unweighted RF distance.

4.2.2 Protein Secondary Structure Prediction

Another common application of MSA construction is to predict the secondary structure of proteins [48], and again the quality of the MSAs affects the accuracy of the predictions. We used MLProbs and other MSA tools to predict the secondary structure of the following protein sequences: 1U24 [49], 6HN6 [50], 5TFD [51], 5DFD [52], 2GDF, 3GDF [53] and 9GAF [54] as follows:

Given a protein sequence s*, we use Jpred 4 [55] to search protein sequences similar to this sequence. Then, we construct an MSA* $\mathcal{M}$ *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 12 shows the number of wrongly aligned residues made by the various tools. In all cases, MLProbs has the smallest number of wrongly aligned residues.

5 Conclusion

This paper explored using the data-centric approach to improve the accuracy of MSA construction. We identified two classification problems that may help improve alignment construction and used the shallow machine learning algorithm Random Forest to train models for them. Then we build a pipeline for MLProbs that make use of these models to help construct MSAs. An empirical evaluation showed that MLProbs' alignment accuracy was significantly better than that of many popular MSA tools.

An interesting question is whether we can make further improvements if we use deep-learning algorithms for the training. We propose some research directions in the following.

Convolutional neural networks (CNNs) [56], which have achieved great success in computer vision, are very suitable for training models like $\mathcal{P}^{\mathtt{Aln}}_{\mathtt{U},\mathtt{V}}$ for MSA construction. We note that an MSA is very similar to a 2D picture; both have hierarchical structures (e.g., an MSA has conserved columns, which form conserved regions, while a picture has points, which form lines, which in turn form boxes), and determining whether an MSA should be aligned by $\mathtt{U}$ or $\mathtt{V}$ is similar to recognizing whether a picture is a dog or a cat, for example. One major difficulty is that the input of our MSA problem is a protein family, not an MSA. One possible solution is that we first use some quick MSA tools to construct an

TABLE 12
Number of Wrongly Aligned Residues in the Predicted Secondary Structures

PDB ID(Length)	MLProbs	QuickProbs	PnpProbs	GLProbs	MSAProbs	ProbCons	MAFFT
1U24(337)	1	1	1	2	2	4	5
5TFD(229)	6	6	6	6	7	6	7
6HN6(282)	11	12	23	22	19	13	24
5DFD(114)	2	2	2	6	7	4	2
2GDF(237)	50	50	52	52	50	60	53
3GDF(267)	4	5	6	6	5	7	8
9GAF(295)	10	12	10	14	13	14	13

imperfect MSA, and then use CNN to classify this imperfect MSA. CNN is known to be very good at recognizing imperfect pictures, allowing titling, tolerating translation and other distortions and noises, and it performs well at recognizing imperfect MSAs.

Instead of treating an MSA as a computer vision problem, we may treat it as a natural language processing (NLP) problem and apply advanced tools like LSTM and BERT [57] to help train the models. For example, given 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 the 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 of 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.

REFERENCES

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

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

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

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

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

[6] Y. Zhang *et al.*, "Constrained pairwise and center-star sequences alignment problems," *J. Combinatorial Optim.*, vol. 32, no. 1, pp. 79–94, 2016.

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

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

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

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

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

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

[13] P. Inglese *et al.*, "Deep learning and 3D-DESI imaging reveal the hidden metabolic heterogeneity of cancer," *Chem. Sci.*, vol. 8, no. 5, pp. 3500–3511, 2017.

[14] N. H. Tran, X. Zhang, L. Xin, B. Shan, and M. Li, "De novo peptide sequencing by deep learning," in *Proc. Nat. Acad. Sci. USA*, 2017, pp. 8247–8252.

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

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

[17] K. D. Nguyen and Y. Pan, "A knowledge-based multiple-sequence alignment algorithm," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 10, no. 4, pp. 884–896, Jul./Aug. 2013.

[18] K. Nguyen, X. Guo, and Y. Pan, *Multiple biological sequence alignment: scoring functions, algorithms and evaluation*. Hoboken, NJ, USA: Wiley, 2016.

[19] P. Di Tommaso *et al.*, "T-coffee: A web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension," *Nucleic Acids Res.*, vol. 39, no. suppl_2, pp. W13–W17, 2011.

[20] E. W. Floden, P. D. Tommaso, M. Chatzou, C. Magis, C. Notredame, and J.-M. Chang, "PSI/TM-Coffee: A web server for fast and accurate multiple sequence alignments of regular and transmembrane proteins using homology extension on reduced databases," *Nucleic Acids Res.*, vol. 44, no. W1, pp. W339–W343, 2016.

[21] R. C. Edgar, "Quality measures for protein alignment benchmarks," *Nucleic Acids Res.*, vol. 38, no. 7, pp. 2145–2153, 2010.

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

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

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

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

[26] A. Gudyś and S. Deorowicz, "QuickProbs 2: Towards rapid construction of high-quality alignments of large protein families," *Sci. Rep.*, vol. 7, 2017, Art. no. 41553.

[27] Y. Ye *et al.*, "GLProbs: Aligning multiple sequences adaptively," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 12, no. 1, pp. 67–78, Jan./Feb. 2015.

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

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

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

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

[32] F. Sievers and D. G. Higgins, "Clustal omega," *Curr. Protoc. Bioinf.*, vol. 48, no. 1, pp. 3–13, 2014.

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

[34] K. D. Nguyen, "On the edge of web-based multiple sequence alignment services," *Tsinghua Sci. Technol.*, vol. 17, no. 6, pp. 629–637, 2012.

[35] J. D. Thompson, F. Plewniak, and O. Poch, "A comprehensive comparison of multiple sequence alignment programs," *Nucleic Acids Res.*, vol. 27, no. 13, pp. 2682–2690, 1999.

[36] T. J. Wheeler and J. D. Kececioglu, "Multiple alignment by aligning alignments," *Bioinformatics*, vol. 23, no. 13, pp. i559–i568, 2007. [Online]. Available: https://bioinformaticshome.com/tools/msa/descriptions/OPAL.html

[37] J. Kececioglu and D. DeBlasio , "Accuracy estimation and parameter advising for protein multiple sequence alignment," *J. Comput. Biol.*, vol. 20, no. 4, pp. 259–279, 2013.

[38] D. DeBlasio and J. Kececioglu, "Boosting alignment accuracy by adaptive local realignment," in *Proc. Int. Conf. Res. Comput. Mol. Biol.*, 2017, pp. 1–17.
[39] G. Landan and D. Graur, "Heads or tails: A simple reliability check for multiple sequence alignments," *Mol. Biol. Evol.*, vol. 24, no. 6, pp. 1380–1383, 2007.
[40] O. Penn, E. Privman, H. Ashkenazy, G. Landan, D. Graur, and T. Pupko, "GUIDANCE: A web server for assessing alignment confidence scores," *Nucleic Acids Res.*, vol. 38, no. suppl_2, pp. W23–W28, 2010.
[41] I. Sela, H. Ashkenazy, K. Katoh, and T. Pupko, "GUIDANCE2: Accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters," *Nucleic Acids Res.*, vol. 43, no. W1, pp. W7–W14, 2015.
[42] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[43] T. H. Ogden and M. S. Rosenberg, "Multiple sequence alignment accuracy and phylogenetic inference," *Systematic Biol.*, vol. 55, no. 2, pp. 314–328, 2006.
[44] H. Li *et al.*, "TreeFam: A curated database of phylogenetic trees of animal gene families," *Nucleic Acids Res.*, vol. 34, no. suppl_1, pp. D572–D580, 2006.
[45] S. Kumar, G. Stecher, M. Li, C. Knyaz, and K. Tamura, "MEGA X: Molecular evolutionary genetics analysis across computing platforms," *Mol. Biol. Evol.*, vol. 35, no. 6, pp. 1547–1549, 2018.
[46] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Math. Biosci.*, vol. 53, no. 1/2, pp. 131–147, 1981.
[47] J. Sukumaran and M. T. Holder, "DendroPy: A Python library for phylogenetic computing," *Bioinformatics*, vol. 26, no. 12, pp. 1569–1571, 2010.
[48] V. Simossis and J. Heringa, "Integrating protein secondary structure prediction and multiple sequence alignment," *Curr. Protein Peptide Sci.*, vol. 5, no. 4, pp. 249–266, 2004.
[49] H.-M. Chu *et al.*, "Structures of Selenomonas ruminantium phytase in complex with persulfated phytate: DSP phytase fold and mechanism for sequential substrate hydrolysis," *Structure*, vol. 12, no. 11, pp. 2015–2024, 2004.
[50] J. Eberhardt, A. G. McEwen, W. Bourguet, D. Moras, and A. Dejaegere, "A revisited version of the apo structure of the ligand-binding domain of the human nuclear receptor retinoic X receptor ," *Acta Crystallogr. Sect. F: Struct. Biol. Commun.*, vol. 75, no. 2, pp. 98–104, 2019.
[51] C. Wang, *et al.*, "Ligand binding to a remote site thermodynamically corrects the F508del mutation in the human cystic fibrosis transmembrane conductance regulator," *J. Biol. Chem.*, vol. 293, no. 46, pp. 17 685–17 704, 2018.
[52] M. G. Baud, E. Lin-Shiao , M. Zengerle, C. Tallant, and A. Ciulli, "New synthetic routes to triazolo-benzodiazepine analogues: Expanding the scope of the bump-and-hole approach for selective bromo and extra-terminal (BET) bromodomain inhibition," *J. Med. Chem.*, vol. 59, no. 4, pp. 1492–1500, 2015.
[53] D. Nüss *et al.*, "Crystal structure of the NADP-dependent mannitol dehydrogenase from Cladosporium herbarum: Implications for oligomerisation and catalysis," *Biochimie*, vol. 92, no. 8, pp. 985–993, 2010.
[54] Q. Xu, D. Buckley, C. Guan, and H.-C. Guo, "Structural insights into the mechanism of intramolecular proteolysis," *Cell*, vol. 98, no. 5, pp. 651–661, 1999.
[55] A. Drozdetskiy, C. Cole, J. Procter, and G. J. Barton, "JPred4: A protein secondary structure prediction server," *Nucleic Acids Res.*, vol. 43, no. W1, pp. W389–W394, 2015.
[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[57] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

Mengmeng Kuang received the BEng degree in computer science from Harbin Institute of Technology in 2018 and the MPhil degree in computer science from the University of Hong Kong in 2020. His research interests include bioinformatics.

Yong Zhang received the PhD degree from the Department of Computer Science and Engineering, Fudan University in 2007. He is currently a professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include the design and analysis of algorithms, bioinformatics, and combinatorial optimization.

Tak-wah Lam received the PhD degree in computer science from the University of Washington in 1988. He is currently a professor with the University of Hong Kong. His research interests include algorithms and bioinformatics.

Hing-fung Ting received the PhD degree in computer science from Princeton University in 1992. He is currently an associate professor with the University of Hong Kong. His research interests include the design and analysis of algorithms and bioinformatics.