Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Sequence Analysis

LE TRECO: an Lstm-based Embedding method for the TREe COnstruction step in Multiple Sequence Alignment

Jing-Teng Hwang [1], Yi-Chang Lu [1,2]*

[1]Department of Electrical Engineering, National Taiwan University, Taipei 106319, Taiwan
[2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106319, Taiwan

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Multiple Sequence Alignment (MSA) has been proven useful in many bioinformatics areas, including protein structure prediction, protein clustering. However, as protein databases become larger and larger, existing MSA methods are no longer efficient when dealing with millions of protein sequences. In addition, language models are appearing as a new way to embed amino acids into high-dimensional vector representations, which can be further used to finetune downstream tasks. So far, although there are numerous approaches on accelerating the workflow of MSA, they often reduce execution time by multi-threading or divide-and-conquer. No one has tried incorporating machine learning (ML) into MSA pipelines.

Results: In this work, we propose a new guide tree construction algorithm, LE TRECO. The algorithm is similar to the mBed guide tree construction method used in Clustal Omega and Kalign3 for they both divide sequences into small clusters before constructing the final guide tree. Different from mBed, the sequence is embedded by a pre-trained LSTM model and the distances between protein sequences are computed by Needleman-Wunsch algorithm in LE TRECO. The result shows that our guide tree algorithm, combined with existing MSA programs, attains SOTA accuracy on many large MSA benchmarks with subtle extra overhead. In some cases, our algorithm even executes faster than traditional methods.

Availability and implementation:: https://github.com/JeterHwang/guide-tree

Contact: yiclu@ntu.edu.tw

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Multiple Sequence Alignments (MSA) are essential elements in bioinformatics for they hold abundant phylogenetic information. Now it is widely used in active binding site prediction, protein secondary and tertiary structure analysis (Daugelaite *et al.*, 2013), protein function prediction analysis (Kemena and Notredame, 2009).Since many biomedical procedures rely on MSA results, people have been pursuing for high accuracy MSA for decades.

In the past two decades, several MSA methods have been proposed, such as MAFFT (Katoh *et al.*, 2002), MUSCLE (Edgar, 2004), Clustal Omega (Sievers *et al.*, 2011), FAMSA (Deorowicz *et al.*, 2016), Kalign3 (Lassmann, 2020), T-Coffee (Di Tommaso *et al.*, 2011). They can be classified into 2 main categories, *Progressive Alignment* and *Iterative Alignment*, with the former being less accurate but more efficient. In fact, most of the MSA methods fall into the category of progressive alignment, which typically includes three stages:

1. **Distance Matrix Calculation:** In this stage, mutual distances between every pair of protein sequences are calculated. To be more specific, supposed there are N sequences to be aligned, the program will have to calculate a $N * N$ matrix, which can be quite time-consuming when N becomes large. Mutual distance is computed by different pairwise alignment algorithms, including Needleman-Wunsch algorithm, Smith-Waterman algorithm, k-mer algorithm.

2. **Guide Tree Construction:** In this stage, a phylogenetic tree is constructed by Unweighted Pair Group Method with Arithmetic Mean (UPGMA) (Gronau and Moran, 2007) or Neighbor-Joining (NJ) (Saitou and Nei, 1987)
3. **Alignment along the Guide Tree:** After the guide tree construction is completed, the final MSA is aligned in a greedy heuristic such that the closer sequences are aligned first. Sequences are added to the profile in the branching order of the guide tree. Different alignment algorithms are used in different MSA methods. For instance, MAFFT and T-Coffee use normal dynamic programming, Clustal Omega uses hidden Markov models (HMM), Kalign3 uses a bit-parallel version of Gene Myers' approximate string matching algorithm in the alignment process.

Among the three stages, the first stage (i.e. distance matrix calculation) is suggested consume a large proportion of total execution time by (Oliver *et al.*, 2005a,b). In order to overcome this bottleneck, several techniques are proposed, such as k-mer distance (Sievers *et al.*, 2011; Edgar, 2004), FFT approximation (Katoh *et al.*, 2002), multi-threading (Nakamura *et al.*, 2018), bit-parallel instruction (Lassmann, 2020; Deorowicz *et al.*, 2016). Although they successfully boost the scalability on large protein benchmarks, their accuracy still has rooms for improvement.

With the advent of ML, neural networks (NN) are used in many areas like computer vision, speech recognition, and signal processing. The advantage of NN lies in that they can learn and identify hidden features of natural phenomena. In the past, these areas required prior knowledge of experts to extract encoded information within the natural world, which was undoubtedly inefficient. By contrast, NNs can predict unmeasured properties by inference on unknown datasets, which is much more accurate and efficient than human labor. Recently, Natural Language Processing (NLP) has become a popular application of machine learning. Due to the advancement in computation hardware and availability of large corpus, AI engineers are able to train deep language models with millions of parameters. RNN-based models are first proposed to deal with NLP tasks, including Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung *et al.*, 2014). After that, transformer-based models (Vaswani *et al.*, 2017) like Bidirectional Encoder Representations from Transformers (BERT) supersedes RNN-based models for higher accuracy. Since the hidden layers of these models are found to contain semantic meanings of sentences, they are often used as an encoder, transforming raw sequences into vector representations that contain semantic information of the whole context. Various downstream tasks can fine tune based on these extracted vector representations, greatly improving final performance and reducing the training data in that the vectors already contain useful semantic and contextual information.

The success in language models draw attention to transfer learning on protein sequences since natural language and protein sequences are both sequential data. Since the number of unmeasured protein sequences is far more than that of measured protein sequences, many works train their models through self-supervised manner. That is, they mask out each token in the sequence and train the generative model to predict the token being masked out. For instance, (Yang *et al.*, 2018; Villegas-Morcillo *et al.*, 2021; Heinzinger *et al.*, 2019) point out that simplistic models pre-trained on large protein database (e.g. Uniprot) can produce high-quality vector representations that improve downstream tasks such as protein-protein interaction prediction and 3D structure prediction. The vector representations are rich in biochemical or physiochemical properties of molecules, which facilitates the process of feature extraction, phenomena also observed in NLP case. (Rives *et al.*, 2021; Meier *et al.*, 2021) even trains complex transformers like BERT and RoBERTa with 250 million protein sequences from UniRef protein database, broadening the application to secondary and tertiary structure prediction and contact prediction. Despite the big success in transformer models, some prefer choosing recurrent neural networks (e.g. LSTMs) as the language model due to the fact that they require less training time and training data to get satisfactory results, a merit for budget-tight laboratories. For example, (Sledzieski *et al.*, 2021; Bepler and Berger, 2021) use a bidirectional LSTM (biLSTM) as the embedding layer, encoding protein sequences into vector representations that can be used for different downstream tasks through transfer learning. However, no one has used these meaningful representations in the area of MSA, where massive phylogenetic information is needed when building a guide tree. Therefore, our goal would be incorporating pre-trained language models into the workflow of guide tree construction, leveraging the learned bio information.

In this work, a novel guide tree construction algorithm, LE TRECO, is proposed (fig. 1). The proposed method first transforms protein sequences into vector representations by a pre-trained 3-layer biLSTM model. Then we adopt divide-and-conquer technique similar to mBed algorithm used in Clustal Omega, splitting protein sequences into small sub-clusters, where sub guide trees are built using accurate pairwise alignment algorithm. The final guide tree is constructed by merging all sub guide trees through the UPGMA algorithm. Owing to the computation power of GPU and the divide-and-conquer heuristic, the execution time falls into acceptable range even when the number of protein sequences is large. Finally, the guide tree is fed into existing MSA methods to produce final alignment. We evaluate LE TRECO on three large MSA benchmarks, including HomFam (Yamada *et al.*, 2016), extended HomFam version 2 (extHomFam-v2) (Deorowicz *et al.*, 2016), and ContTest (Fox *et al.*, 2016). The result shows that LE TRECO outperforms most existing MSA programs on alignment accuracy.

The main contribution of this work is the integration of pre-trained language models with MSA tasks. As far as we know, no one has ever used language models to embed protein sequences within the guide tree construction process. We also test different MSA algorithms in each of the four stages to acquire the best trade-off between speed and alignment accuracy. Compared to the legacy, LE TRECO boosts MAFFT and Clustalo on extHomFam-v2 by 8% and 6%. Combined with T-Coffee, it achieves SOTA accuracy on multiple large MSA benchmarks.

The rest of this article is organized as follows. The language model architecture, clustering algorithm, guide tree construction algorithm, and alignment algorithm are described in Section 2. Benchmark selections, hardware settings, and scoring metrics of our experiments are stated in Section 3. The qualitative evaluations for LE TRECO and several MSA programs are presented in Section 4, concluding the article.

2 Approach and Method

This section introduces the pipeline of LE TRECO. We will give detailed description on the design of every stage of our algorithm, along with analysis of hyperparameter choice.

2.1 The Choice between Embedding Schemes

In this stage, input protein sequences (often in FASTA format) are transformed into vector representations as described in section 1. With the language models, we can convert a sequence of amino acids into a sequence of vector representations (embeddings) within a second. Several LSTM-based and transformer-based models have made a success in this task. The most representative LSTM-based model is *ProSE*, proposed by Bepler and Berger (2021), training a 3 layer biLSTM on both supervised and self-supervised training tasks. On the self-supervised training task, they train the biLSTM model on the Uniref90 protein database,

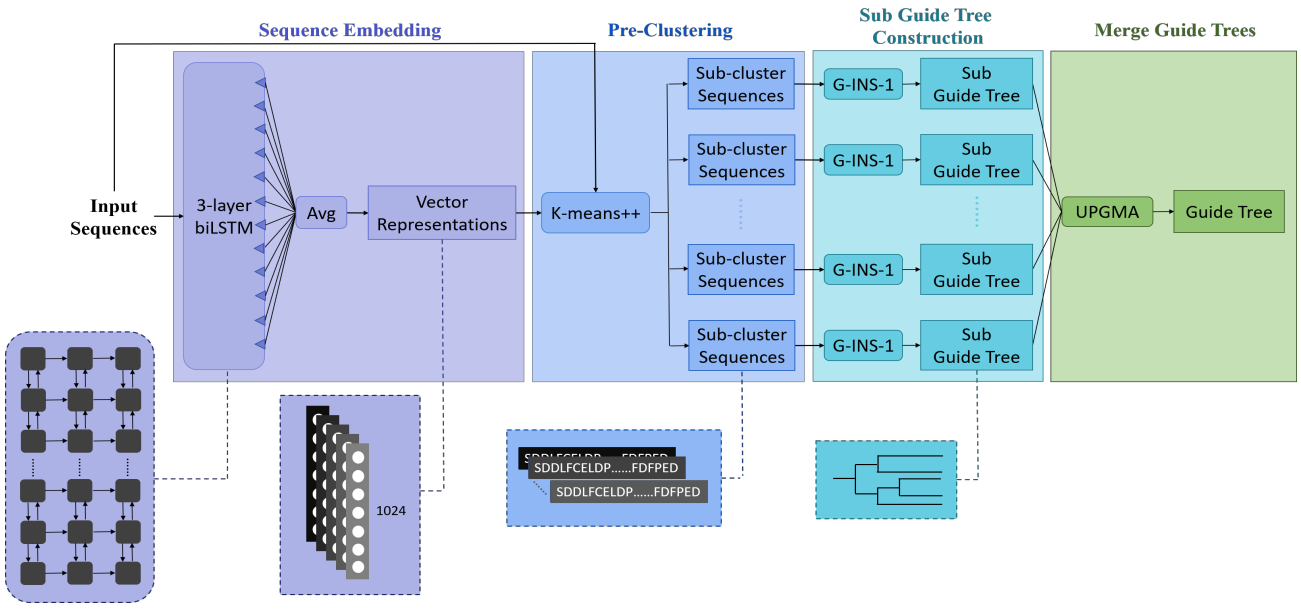

Figure 1: An overview of the guide tree construction process of LE TRECO. It takes protein sequences as input and outputs a guide tree that will be inputted to existing MSA programs along with raw sequences, generating the final MSA alignment. The main functional blocks are sequence embedding, pre-clustring, sub guide tree construction, and merge guide trees from left to right.

which contains 76 millions of unlabeled protein sequences. By masking out each token in the natural protein sequence, the model is trained to predict the vanished tokens. Though the model can learn semantic information following this training heuristic, it has little knowledge about the physiochemical properties of the corresponding protein. To complement this, they perform supervised training on 28,000 labeled protein sequences from Structural Classification of Proteins extended (SCOPe) database. The model is trained by 2 supervised tasks, respectively *residue-residue contact prediction* and *structural similarity prediction*. The results of embeddings are outstanding in distinguishing protein structure similarity. Another protein embedding model is *ESM-1* proposed by Rives *et al.* (2021), which uses a transformer-based language model. The model is trained on Uniref50 dataset with self-supervised mask training heuristic same as that used in *ProSE*. The pre-trained embeddings are used in various downstream tasks such as *contact prediction* and *protein classification*. (Lin *et al.*, 2022) releases the newest model, *ESM-2* this year, which broadens and deepens the transformer and expands the training data to Uniref50 plus sampled Uniref90.

After careful evaluation (detailed in Section 4.4.1), we select *ProSE*, the LSTM-based model as our protein embedding model for the following reasons. First of all, though having much bigger model size (up to 3 billion parameters), *ESM-1* and *ESM-2* do not perform well on predicting protein similarity. It's probably because they are only trained on self-supervised mask training, learning only semantic information in protein sequences. Second of all, the model sizes of transformers are a big burden to our GPU memory even if we only perform model inference. What's worse, due to heavy model size and self-attention mechanism, the inference time of transformers is 2x longer than that of LSTM-based models, which becomes unacceptable when the sequence number grows.

ProSE supports 2 types of models, *ProSE-DLM* and *ProSE-MT*. Their difference lies in that *ProSE-DLM* only undergoes self-supervised training while *ProSE-MT* is further trained on residue-residue contact prediction and structural similarity prediction task. In our analysis, *ProSE-MT* performs better than *ProSE-DLM*, so we choose *ProSE-MT* as the final embedding model in this stage. However, we make some small changes to the original model to make it fit our scenario. First we remove the structural similarity prediction head in the model, with 3-layer biLSTM left untouched. Then we change the output dimension of 3-layer biLSTM from Eq. (1):

$$Dimension_{out} = N_{in} + num\_layers * H_{out} * D, \quad (1)$$

to Eq. (2):

$$Dimension_{out} = H_{out} * D, \quad (2)$$

where N_{in} is the input dimension of the first LSTM layer, which is twenty-one since one-hot encoding is used to encode amino acids. num_layers is equal to three, which represents the number of LSTM layers. H_{out} is the output dimension of each LSTM layer, which is 1,024 in our case. D is set to two when the LSTM is bidirectional, or set to one otherwise. In other words, we remove the shortcuts between the output of the intermediate layers to the final output, which is used to increase converging speed during training. In our experiment, high dimensional embeddings tend to fit certain protein families, and thus fail to generalize to large datasets containing various protein families. Besides, the speed of K-means++ algorithm in the next stage is influenced by the dimensionality of the embeddings. Specifically, higher dimensionality will slow down the next stage. Therefore, lower the dimensionality of the embeddings not only enhances the overall alignment accuracy but also reduces execution time.

2.2 Pre-clustering

Subsequently, the protein sequences are clustered by their mutual distances in the vector space generated in the previous stage. Similar to (Sievers *et al.*, 2011; Lassmann, 2020), we adopt bi-secting K-means++ to divide protein sequences into small sub-clusters, whose sizes are lower than certain threshold. After evaluation (detailed in Section 4.4.2), we choose 500 for the maximum sub-cluster size. Here we make a small modification from the original algorithm. After the bi-secting K-means++, we redo K-means++ on all protein sequence embeddings, with K centers initialized

Table 1. The alignment commands used in the final alignment stage.

Program	Alignment Command
MAFFT	mafft –anysymbol –thread 16 –treein tree.dnd seqs.fasta > output.fasta
Clustalo	clustalo –threads=16 –in seqs.fasta –guidetree-in tree.dnd –out output.fasta –force
FMASA	famsa -keep-duplicates -t 16 -gt import tree.dnd seqs.fasta output.fasta
T-Coffee	t_coffee -reg -thread 16 -child_thread 0 -seq seqs.fasta -nseq 200 -tree tree.dnd -method mafftgins1_msa -outfile output.fasta

Algorithm 1 PRE-CLUSTERING

Input: N Vector representations $V = \{V_1, V_2, \ldots, V_N\}$
Output: K clusters $C = \{C_1, C_2, \ldots, C_K\}$

$C \leftarrow \{(V, None)\}$
$threshold \leftarrow 500$
while $size(C[0]) > threshold$ **do**
  $C_{max} \leftarrow C[0]$
  $C \leftarrow C - C[0]$
  $k \leftarrow 2$
  $\{(Vec_1, Cen1), (Vec_2, Cen_2)\} \leftarrow$ Kmeans++$(C_{max}[0], k)$
  $C \leftarrow C \cup \{(Vec_1, Cen1), (Vec_2, Cen_2)\}$
  sort C by decreasing cluster size
end while
$K \leftarrow$ number of clusters in C
Initialize Kmeans++ with K cluster centers
$C \leftarrow$ Kmeans++(V, K)

to be the centers found by bi-secting K-means++ algorithm in the hope of alleviating the bias introduced by the bi-secting heuristic. The details of the clustering algorithm is provided in Alg. 1

2.3 Sub Guide Tree Construction

Once the sequences are clustered into small sub-clusters, guide tree construction process starts in each sub-cluster. As mentioned in Section 1, traditional guide tree construction relies on a distance matrix, which stores the mutual distances of the protein sequences. There are several existing distance (similarity) metrics for protein sequences. K-mer distance used in Sievers *et al.* (2011) and FFT approximation used in Katoh *et al.* (2002) are fast but not accurate. Edit distance used in (Deorowicz *et al.*, 2016) is very fast and more accurate than K-mer and FFT. The L-INS-1(i) and G-INS-1(i) option used in MAFFT (Katoh *et al.*, 2005) calculate sequence similarity with Smith-Waterman (SW) and Needleman-Wunsch (NW) alignment scores, respectively. These methods are the most accurate but become time-consuming when the sequence number grows huge since NW and SW algorithms are hard to accelerate on CPU. How to strike a balance between accuracy and speed has been an important issue for previous MSA algorithms.

In general, G-INS-1(i) is not suitable for sequences larger than 1,000, which means it can hardly be used in large protein benchmarks. However, the pre-clustering done in the previous stage makes G-INS-1(i) a possible choice to build distance matrices. It reduces the estimated time complexity of building all distance matrices from Eq. (3):

$$T_{dist\_mat} = N^2 * T_{avg}, \tag{3}$$

to Eq. (4):

$$\begin{aligned} T_{dist\_mat} &= M^2 * \frac{N}{M} * T_{avg} \\ &= M * N * T_{avg}, \end{aligned} \tag{4}$$

where M is the maximum sub-cluster size, N is the number of sequences, T_{avg} is the average time for mutual distance calculation. Therefore, the total execution time will be affordable even if the sequence number N grows extremely large. That is, we can use L-INS-1(i) or G-INS-1(i) for distance calculation to get accurate results within acceptable execution time.

2.4 Complete Guide Tree Construction

In this stage, we merge the sub guide trees generated in the previous stage by hierarchical clustering method. There are several existing hierarchical clustering method, where Unweighted Pair Grouping Method with Arithmetic-mean (UPGMA) is most widely used. However, since UPGMA is of $O(n^3)$ time complexity (n denotes the number of sequences), it would be unfeasible to implement UPGMA on large protein datasets. Hence, plenty of acceleration schemes are proposed to address the issue. Clustal Omega and MUSCLE adopt Fast UPGMA algorithm, which can attain $O(n^2)$ time complexity. Nevertheless, their correctness is doubted for they assume that if either C_i or C_j is the nearest neighbor (has minimal distance) to K, then $C_i \cup C_j$ is also the nearest neighbor of K. The authors of (Gronau and Moran, 2007) point out the fallacy and propose a revised version, *LCP UPGMA*, which not only attains $O(n^2)$ complexity but also produces the same result as original UPGMA. They prove that local closest pair (LCP) and global closest pair (GCP) are equivalent in the case of UPGMA. To be more specific, if C_i is closest to C_j, or that to say, C_i is the nearest neighbor (NN) of C_j, and vice versa is true, we can assume $d(C_i, C_j)$ is the global minimum in the distance matrix. The reduction formula of UPGMA will make this assumption holds for every iteration. That way, we don't have to scan through the whole distance matrix to find the global minima, which requires $O(n^2)$ time; instead, we only have to maintain and scan through the nearest neighbor of each cluster, which requires only $O(n)$ time.

In our work, we construct a temporary guide tree with LCP UPGMA based on the cluster centers found in the pre-clustering stage, where the distance matrix is calculated by the Euclidean distances between cluster centers. Then the positions of each cluster center on the temporary guide tree (which must be the leaves), are replaced by corresponding sub guide trees built in the previous stage, forming the complete guide tree. The detailed implementation is provided in Alg. 2, where the LCP UPGMA is based on the algorithm stated in (Gronau and Moran, 2007).

2.5 Final Alignment

In this stage, the complete guide tree, along with the raw protein sequences are inputted to existing MSA programs, producing the final MSA. Four MSA programs, MAFFT, Clustal Omega (Clustalo), FAMSA, T-Coffee, are selected to perform the task since they all support user-defined guide trees. The detailed linux commands to run the alignment are listed in Table 1.

3 Evaluation

3.1 Benchmark Selection

We select 3 publicly available benchmarks, HomFam (Yamada *et al.*, 2016), extHomFam-v2 (Deorowicz *et al.*, 2016), ContTest (Fox *et al.*, 2016) for evaluating large MSA. HomFam consists of 89 families, constructed by extending HOMSTRAD data with structural information and

Algorithm 2 LCP UPGMA

Input: $C = \{C_1, C_2, \ldots, C_K\}$
Output: Guide tree T

$Q \leftarrow \varnothing$
$T \leftarrow \{node(C_1), node(C_2), \ldots, node(C_K)\}$
for $i = 1, 2, \ldots, K-1$ **do**
  if Q is empty **then**
    $C_{left} \leftarrow$ any cluster left in C
    $Q \leftarrow Q \cup \{C_{left}\}$
  end if
  while NN(NN($Q[-1]$))$\neq Q[-1]$ **do** ▷ NN: Nearest Neighbor
    $C_{NN} \leftarrow$ NN($Q[-1]$)
    $Q \leftarrow Q \cup C_{NN}$
  end while
  $C_1 \leftarrow Q[-1]$
  $C_2 \leftarrow$ NN($Q[-1]$)
  $Q \leftarrow Q - C_1$
  $C \leftarrow C - C_1 - C_2$
  $C_{new} \leftarrow C_1 \cup C_2$
  for each cluster $C_i \in C$ **do**
    $dist(C_i, C_{new}) = (dist(C_i, C_1) + dist(C_i, C_2))/2$
  end for
  $C \leftarrow C \cup C_{new}$
  Create a new node $node(C_{new})$ on T
  $node(C_{new}).left = node(C_1)$
  $node(C_{new}).right = node(C_2)$
end for

corresponding families from PFam database. The families are splitted into Small ($0 < N \leq 3000$), Medium ($3000 < N \leq 10000$) and Large ($10000 < N$), where N is the number of sequences in that family. To show the superiority of LE TRECO on extremely large benchmarks, we select the biggest MSA benchmark, extHomFam-v2, which is constructed by the same manner of HomFam except that the selection threshold is lower. Since PFam and HOMSTRAD database have increased since HomFam was first created, the number of sequences in extHomFam-v2 is much bigger than HomFam, with total 393 families. The dataset is firther splitted into 5 subsets, Small ($N \in [200, 10k)$), Medium ($N \in [10k, 40k)$), Large ($N \in [40k, 100k)$), Xlarge ($N \in [100k, 250k)$), Huge ($N \in [250k, 3M)$). Due to server memory limitation, we only perform experiment on Small, Medium, Large division. In practice, HomFam and extHomFam-v2 rely heavily on the availability of reference alignment in each family. However, Due to the scarcity of known reference alignment, we can only compare the predicted MSA with small fraction of sequences that appear in reference alignment, which leads to some inaccuracy. Hence, we include ContTest in our benchmarks in that it uses contact map prediction accuracy to evaluate MSA. Following (Yamada *et al.*, 2016), we split ContTest into 3 subsets with the same manner as HomFam.

3.2 Hardware Settings

We run all experiments on a workstation with one 8-core Intel i7-11700K processor (clocked at 3.60GHz), two Nvidia RTX 3090 GPUs (24GB RAM), and 64 GB memory. To fairly compare the execution time, all MSA programs are run with 16 computing threads, and all neural network inferences are performed on a single GPU with batch size 128. Unless otherwise stated, the maximum sub-cluster size is set to 500 in our method.

3.3 Scoring Metrics

For HomFam and extHomFam-v2, we use sum-of-pairs (SP) and total-column (TC) to evaluate the predicted MSA. SP is the fraction of correctly aligned pairs in predicted MSA and aligned pairs in the reference alignment. TC is the fraction of correctly aligned columns in predicted MSA and aligned columns in reference alignment. Following PASTA (Mirarab *et al.*, 2015), MAGUS (Smirnov and Warnow, 2021), MAFFT (Yamada *et al.*, 2016), we use FastSP (Mirarab and Warnow, 2011) to compute SP and TC scores. For ContTest, we use built-in PSICOV version 2.1 (Jones *et al.*, 2012) to calculate precision score for contact predictions.

4 Result

This section presents the benchmark results of LE TRECO and other MSA programs. In section 4.1, we show SP, TC scores and execution time of HomFam dataset. In section 4.2, the result of extHomFam-v2 is presented. In section 4.3, the contact prediction accuracy measured in ContTest dataset is presented. In section 4.4, we experiment on language model types as well as the maximum sub-cluster size.

4.1 HomFam

This test is meant to show the superiority of LE TRECO over traditional methods. As described in section 3, HomFam is divided into 3 groups with different sequence numbers, and SP/TC scores are acquired by the FastSP program (Mirarab and Warnow, 2011). The result is shown in Table 4.1. The proposed guide tree algorithm, LE TRECO, combined with existing MSA tool, attains higher SP, TC score than that using the tool alone. In the small division, Clustalo+LE TRECO has the highest SP score, while T-coffee has the highest TC score. In the medium and large division, T-Coffee+LE TRECO gets the best SP and TC score. As

Table 2. Comparison of algorithms on HomFam dataset

	Small			Medium			Large			All		
Algorithm	SP	TC	Time	SP	TC	Time	SP	TC	Time	SP[1]	TC[1]	Time[2]
MAFFT	84.9	63.6	37	81.8	55.3	3:54	69.8	41.1	26:14	78.8	53.3	30:45
MAFFT+LE TRECO	86.3	65.6	5:21	86.2	63.4	21:02	76.7	48.4	1:08:53	83.1	59.1	1:35:16
Clustalo	85.5	64.0	6:46	81.2	58.0	52:42	65.6	37.9	1:22:54	77.4	53.3	2:22:22
Clustalo+LE TRECO	**87.6**	66.9	12:04	87.6	67.0	57:09	72.1	45.1	1:51:44	82.4	59.7	3:00:57
FAMSA	85.1	62.6	**12**	86.3	62.6	**35**	73.4	42.9	**5:47**	81.6	56.0	**6:34**
FAMSA+LE TRECO	85.8	62.4	5:05	87.1	63.6	19:30	75.2	45.0	39:15	82.7	57.0	1:03:50
T-Coffee	86.4	65.8	16:36	87.7	66.5	1:09:13	77.5	49.9	1:53:58	83.9	60.7	3:19:47
T-Coffee+LE TRECO	87.5	**67.1**	17:20	**88.7**	**68.0**	1:02:30	**79.5**	**53.2**	1:21:42	**85.2**	**62.8**	2:41:32

[1] The value is computed by averaging small, medium, large values.
[2] The value is computed by summing small, medium, large values.

Table 3. Comparison of algorithms on extHomFam-v2 dataset

Algorithm	Small			Medium			Large			All		
	SP	TC	Time	SP	TC	Time	SP	TC	Time	SP[1]	TC[1]	Time[2]
MAFFT	81.7	67.6	4:15	74.2	57.4	1:58:53	64.2	45.7	14:26:42	73.4	56.9	16:29:50
MAFFT+LE TRECO	83.7	70.3	22:51	81.0	64.5	4:04:12	73	54.3	15:51:07	79.2	63.0	20:18:10
Clustalo	85.2	72.5	22:02	74.7	58.0	4:10:03	61.5	43.7	17:00:21	73.8	58.1	21:32:26
Clustalo+LE TRECO	86.7	73.9	42:10	80.9	65.5	7:18:15	67.8	49.9	21:35:52	78.5	63.1	29:36:17
FAMSA	85.0	71.6	**47**	82.3	65.9	**18:31**	75.9	57.8	**2:45:40**	81.1	65.1	**3:04:58**
FAMSA+LE TRECO	85.0	71.5	20:19	82.8	65.5	2:53:56	75.8	56.7	8:39:36	81.2	64.6	11:53:51
T-Coffee	86.7	74.4	47:34	81.7	65.6	6:48:25	73.7	56.5	18:59:32	80.7	65.5	26:35:31
T-Coffee+LE TRECO	**87.7**	**75.5**	51:27	**84.9**	**69.7**	6:42:23	**77.4**	**60.0**	15:01:28	**83.3**	**68.4**	22:35:18

[1] The value is computed by averaging small, medium, large values.
[2] The value is computed by summing small, medium, large values.

for the execution speed, FAMSA executes the fastest for its bit-parallel feature. On the other hand, T-Coffee has the slowest execution speed since it uses the iterative alignment scheme (MAFFT G-INS-i option) when constructing alignment. However, the iterative alignment scheme also lets T-Coffee achieve the highest SP, TC score. Figure 2 plots the overall SP and TC score on HomFam dataset, computed by averaging values of small, medium, large division. The accuracy enhancement is most profound for MAFFT and Clustalo, while smaller on FAMSA and T-Coffee because their baselines have already achieve high SP/TC scores.

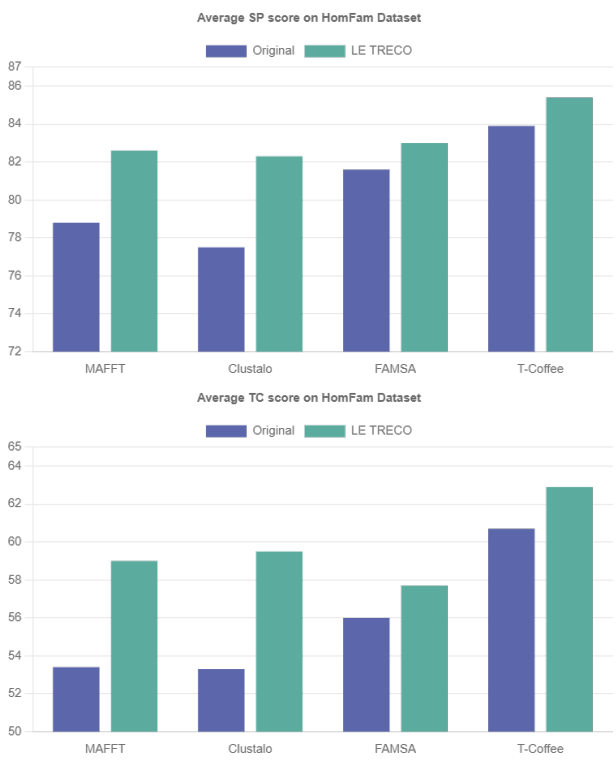

Figure 2: The overall SP and TC scores of 4 traditional MSA methods and their variants after adding LE TRECO on the HomFam dataset. The top subplot is for SP scores, while the bottom subplot is for TC score.

4.2 extHomFam-v2

We also compare our algorithm with other methods on the extHomFam-v2 dataset, which is formed by enlarging the HomFam dataset. The SP/TC scores as well execution time of all algorithms are compared in table 3. LE TRECO achieves a huge enhancement for MAFFT, Clustalo, T-Coffee, increasing up to 8% accuracy. While in FAMSA, it achieves comparable results with the original program in small and medium division, but becomes slightly inferior on the large division. Since we do not observe this phenomenon in other datasets such as HomFam and ContTest, we suspect that the additional sequences added by (Deorowicz *et al.*, 2016) are more favorable for the Kalign LCS distance metric used in FAMSA's distance matrix construction. Figure 3 compares the overall SP/TC scores for traditional methods and their LE TRECO variants. The score of LE TRECO is significantly higher in MAFFT, Clustalo, T-Coffee, while comparable in FAMSA.

4.3 ContTest

To avoid HomFam-specific results, we test our algorithm on ContTest, a benchmark evaluating MSA by contact prediction, as stated in section 4. The score is acquired by PSICOV version 2.1 (Jones *et al.*, 2012). As shown in Table 4, LE TRECO boosts contact prediction score except for FAMSA on medium division. The overall average score enhancement for MAFFT, Clustalo, T-Coffee are respectively 2.01%, 3.52%, 5.35%, while FAMSA+LE TRECO produces comparable results with FAMSA. Different from HomFam, where T-Coffee+LE TRECO achieves the best SP/TC scores on almost all divisions, MAFFT+LE TRECO achieves the highest score for all divisions in ContTest. In addition, we observe that FAMSA no longer holds superior accuracy among the three progressive alignment methods (MAFFT, Clustalo, FAMSA). The overall average score shows that FMASA+LE TRECO even has the lowest score among all MSA programs plus LE TRECO. Nevertheless, FAMSA still executes the fastest, while T-Coffee remains the slowest.

4.4 Ablation Study

4.4.1 Model Type

To better understand the influence of pre-trained language models on alignment accuracy, we experiment transformer-based models and LSTM-based models on HomFam dataset. The transformer-based models are proposed by (Rives *et al.*, 2021; Lin *et al.*, 2022), named ESM, and we choose the latest version (ESM-2) with 3 different parameter sizes (43M, 150M, 650M). The LSTM-based model is the same model used elsewhere in this paper, proposed by (Bepler and Berger, 2021), named Prose-MT. Each protein sequence is processed by these models, where the outputs

Table 4. Comparison of algorithms on ContTest dataset

Algorithm	Small		Medium		Large		All	
	Score	Time	Score	Time	Score	Time	Score[1]	Time[2]
MAFFT	0.4129	14	0.5038	6:42	0.6008	42:00	0.5058	48:56
MAFFT+LE TRECO	**0.4172**	1:48	**0.5278**	29:45	**0.6328**	1:32:08	**0.5259**	2:03:41
Clustalo	0.3103	1:48	0.4869	39:43	0.5235	2:09:06	0.4402	2:50:39
Clustalo+LE TRECO	0.3609	3:39	0.5077	1:00:49	0.5576	3:05:16	0.4754	4:09:45
FAMSA	0.3166	2	0.4902	1:22	0.5513	5:10	0.4527	6:34
FAMSA+LE TRECO	0.3210	1:38	0.4707	26:00	0.5626	1:05:35	0.4514	1:33:14
T-Coffee	0.3063	5:17	0.4824	1:20:24	0.5607	3:15:10	0.4498	4:40:51
T-Coffee+LE TRECO	0.3613	6:46	0.5222	1:18:15	0.6264	2:36:39	0.5033	4:01:40

[1] The value is computed by averaging small, medium, large values.
[2] The value is computed by summing small, medium, large values.

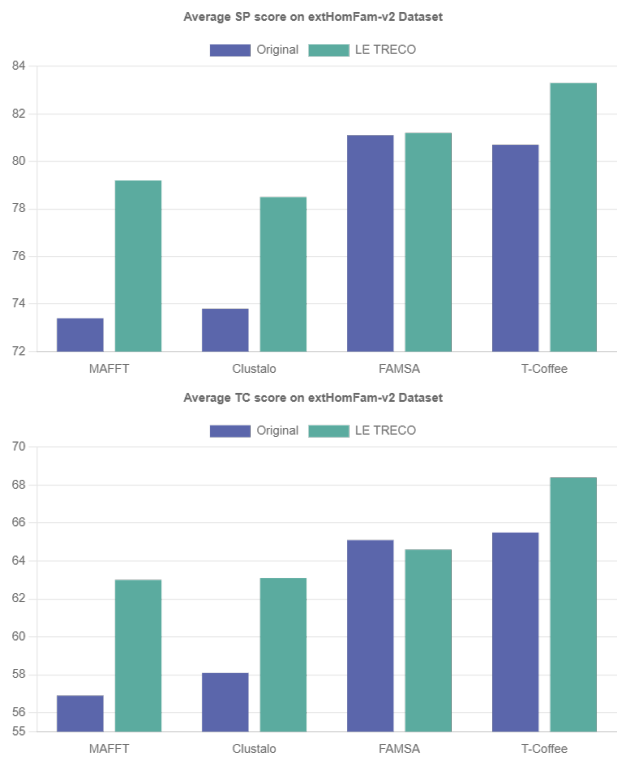

Figure 3: The overall SP and TC scores of 4 traditional MSA methods and their variants after adding LE TRECO on the extHomFam-v2 dataset. The top subplot is for SP scores, while the bottom subplot is for TC score.

of each amino acid token are averaged to form a vector representation. After that, all workflows are the same as stated in Section 2. To be fair, the hardware settings remain unchanged as stated in Section 3.2, except that the inference batch sizes of ESM models are reduced to 32 for their large model sizes.

The result is shown in Table 5. Prose-MT outperforms three ESM models in average SP, TC and average total execution time. The superior performance of Prose-MT probably results from its finetuning for 2 supervised tasks (residue-residue contact prediction, structural similarity prediction). The supervised training enables the Prose-MT model to better extract latent structural information , thus fitting better to our scenario. Therefore, even though ESM models are trained on a much larger dataset (Uniref 50 + sampled Uniref90) compared with that of Prose-MT (Uniref 90), the generated embeddings don't work well in our task due to lack of downstream finetuneing. As for the execution time, transformer-base models are intrinsically more complex and heavy, which reasons for their long inference time compared with LSTM-based models.

Table 5. Comparison of different NN models

Model Type	Avg. SP[1]	Avg. TC[1]	Tot. Time[2]
ESM-43M	78.0	52.9	4:57:42
ESM-150M	80.7	55.5	4:32:11
ESM-650M	79.0	52.7	4:35:41
Prose-MT	**83.3**	**59.8**	**2:05:24**

[1] Computed by averaging the average score (average over small, medium, large division) of 4 MSA programs (MAFFT, Clustalo, FAMSA, T-Coffee)
[2] Computed by averaging the total time (sum over small, medium, large division) of 4 MSA programs (MAFFT, Clustalo, FAMSA, T-Coffee)

4.4.2 Max Sub-cluster Size

We also perform experiment on the value selection for maximum sub-cluster size in bi-secing K-means++ algorithm. For simplicity, we use HomFam dataset for this experiment, and test 10 cluster sizes $(100, 200, \ldots, 1000)$ for each of the 4 MSA programs. We plot the result in figure 4, where the dot size is proportional to the cluster size. As one can see, the execution time increases for larger cluster sizes, which is plausible in that it must construct a bigger distance matrix in each sub-cluster using slow NW algorithm. On the other hand, SP scores seem not related to the cluster size. The variation of SP scores is within 4%, and it doesn't follow any specific pattern. Although we only plot SP score, the TC score also shows the similar pattern. Therefore, we conclude that the selection of max sub-cluster size does not greatly impact the alignment accuracy. In other place of this paper, we simply set this value to 500.

5 Conclusion

In this work, we propose a novel MSA workflow, LE TRECO, for high alignment accuracy on large protein datasets. Different from traditional MSA methods, we introduce pre-trained biLSTM model into our pipeline. Each protein sequence is embedded by pre-trained biLSTM model to form a vector representation. Then the sequences are divided into sub-cluster by modified bi-secting K-means++ algorithm. Sub guide trees are constructed in every sub-cluster and merged to form the complete guide tree.

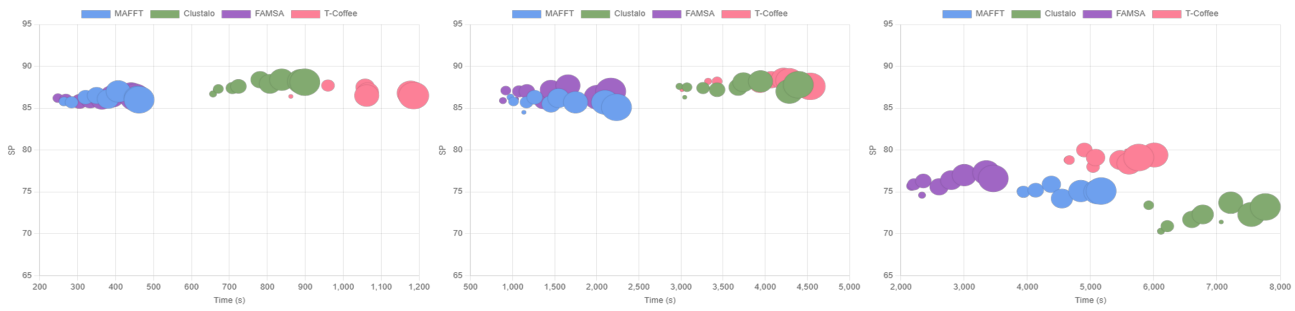

Figure 4: The detailed analysis of the influence of maximum sub-cluster size used in bi-secting K-means++ algorithm on execution time and SP scores. The dot size is proportional to the maximum sub-cluster size. The dot color blue, green, purple, red represents MAFFT, Clustalo, FAMSA, T-Coffee repectively.

At last, MSA is aligned along the guide tree by existing MSA programs. The proposed LE TRECO provides distinctive enhancement for 4 tested MSA programs (MAFFT, Clustalo, FMASA, T-Coffee) on HomFam, extHomFam-v2, and ContTest, with subtle extra overhead. Furthermore, the combination of T-Coffee and LE TRECO achieves SOTA performance on all 3 benchmarks.

References

Bepler, T. and Berger, B. (2021). Learning the protein language: Evolution, structure, and function. *Cell systems*, **12**(6), 654–669.

Chung, J. *et al.* (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Daugelaite, J. *et al.* (2013). An overview of multiple sequence alignments and cloud computing in bioinformatics. *International Scholarly Research Notices*, **2013**.

Deorowicz, S. *et al.* (2016). Famsa: Fast and accurate multiple sequence alignment of huge protein families. *Scientific reports*, **6**(1), 1–13.

Di Tommaso, P. *et al.* (2011). T-coffee: a web server for the multiple sequence alignment of protein and rna sequences using structural information and homology extension. *Nucleic acids research*, **39**(suppl_2), W13–W17.

Edgar, R. C. (2004). Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics*, **5**(1), 1–19.

Fox, G. *et al.* (2016). Using de novo protein structure predictions to measure the quality of very large multiple sequence alignments. *Bioinformatics*, **32**(6), 814–820.

Gronau, I. and Moran, S. (2007). Optimal implementations of upgma and other common clustering algorithms. *Information Processing Letters*, **104**(6), 205–210.

Heinzinger, M. *et al.* (2019). Modeling aspects of the language of life through transfer-learning protein sequences. *BMC bioinformatics*, **20**(1), 1–17.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Jones, D. T. *et al.* (2012). Psicov: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, **28**(2), 184–190.

Katoh, K. *et al.* (2002). Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic acids research*, **30**(14), 3059–3066.

Katoh, K. *et al.* (2005). Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucleic acids research*, **33**(2), 511–518.

Kemena, C. and Notredame, C. (2009). Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, **25**(19), 2455–2465.

Lassmann, T. (2020). Kalign 3: multiple sequence alignment of large datasets.

Lin, Z. *et al.* (2022). Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*.

Meier, J. *et al.* (2021). Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems*, **34**, 29287–29303.

Mirarab, S. and Warnow, T. (2011). Fastsp: linear time calculation of alignment accuracy. *Bioinformatics*, **27**(23), 3250–3258.

Mirarab, S. *et al.* (2015). Pasta: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. *Journal of Computational Biology*, **22**(5), 377–386.

Nakamura, T. *et al.* (2018). Parallelization of mafft for large-scale multiple sequence alignments. *Bioinformatics*, **34**(14), 2490–2492.

Oliver, T. *et al.* (2005a). Multiple sequence alignment on an fpga. In *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, volume 2, pages 326–330. IEEE.

Oliver, T. *et al.* (2005b). Using reconfigurable hardware to accelerate multiple sequence alignment with clustalw. *Bioinformatics*, **21**(16), 3431–3432.

Rives, A. *et al.* (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, **118**(15), e2016239118.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, **4**(4), 406–425.

Sievers, F. *et al.* (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, **7**(1), 539.

Sledzieski, S. *et al.* (2021). Sequence-based prediction of protein-protein interactions: a structure-aware interpretable deep learning model. *bioRxiv*.

Smirnov, V. and Warnow, T. (2021). Magus: multiple sequence alignment using graph clustering. *Bioinformatics*, **37**(12), 1666–1672.

Vaswani, A. *et al.* (2017). Attention is all you need. *Advances in neural information processing systems*, **30**.

Villegas-Morcillo, A. *et al.* (2021). Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, **37**(2), 162–170.

Yamada, K. D. *et al.* (2016). Application of the mafft sequence alignment program to large data—reexamination of the usefulness of chained guide trees. *Bioinformatics*, **32**(21), 3246–3251.

Yang, K. K. *et al.* (2018). Learned protein embeddings for machine learning. *Bioinformatics*, **34**(15), 2642–2648.