Predicting Sequence Alignment Distances via Approximate String Matching

Jian Chen [1*], Le Yang [2*], Lu Li[3], Steve Goodison[5], Yijun Sun[1,2,4 †]

[1]Department of Computer Science and Engineering
[2]Department of Microbiology and Immunology
[3]Department of Oral Biology
[4]Department of Biostatistics
The State University of New York at Buffalo, Buffalo, NY 14203
[5]Department of Health Sciences Research
Mayo Clinic, Jacksonville, FL 32224

Abstract

Motivation: Sequence comparison is the foundation of sequence analysis in bioinformatics. Large-scale comparisons are facilitated by efficient alignment-free methods, which only compute rough approximations to the alignment distances. Recently, two data-dependent methods using neural networks were proposed for high accuracy alignment-free comparison by using embedding vectors. However, their performance on long, varying length sequences is still limited by the structure of their embedding function.

Results: In this paper, we propose a new method combining the ASM (**A**pproximate **S**tring **M**atching) with neural networks to learn embedding vectors that is robust to insertions and deletions for biological sequences. Furthermore, we designed an efficient gradient computation algorithm for the proposed structure. The results of large-scale experiments on alignment distance estimation and taxonomy assignment proved that the proposed method significantly outperformed the state-of-the-art methods in terms of accuracy and efficiency on five real-world datasets.

Availability: Open-source software for the proposed method is developed and freely available at `https://www.acsu.buffalo.edu/yijunsun/lab/ASM.html`.

Contact: `yijunsun@buffalo.edu`

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

Bioinformatics
submitted on xx xx, 2022

*Equal contribution
†Please address all correspondence to Dr. Yijun Sun (`yijunsun@buffalo.edu`).

1 Introduction

In bioinformatics, sequence analysis is a significant area of research. Sequence comparison is one of the fundamental analysis procedures, where biological sequences are compared either globally or locally to examine their functional and evolutional relationships. In this paper, we will discuss the global comparison. Traditionally, it is done through pairwise alignment using the gold standard Needleman-Wunsch (NW) algorithm [1]. However, it cannot be used for large-scale analysis due to its quadratic complexity. During the past 20 years, the sequencing capacity has been increasing at a super-exponential speed [2,3]. Hence, many large sequence datasets are available and in dire need of efficient approaches for the analyses.

In order to enable the large-scale analysis, many alignment-free approaches were developed as efficient alternatives of the global alignment [4–7]. Commonly used methods are mainly based on word frequency (k-mer [8] and FFP [9]) or sub-strings (ACS [10], Kr [11], kmacs [12]). The word frequency methods estimate alignment distance using similarities between frequency distributions of fixed-length sub-sequences where the length parameter is usually determined empirically. The substring methods do not specify a word length and are aware of similarities and differences between words. They use more complex data structures to process varying-length fragments of a sequence and are generally more accurate than the methods relying on fixed-length words. These methods have found many applications including database search [13, 14], sequence annotation [15], metagenomics [16–19] and comparative genomics [5]. Nowadays, with the advent of the third-generation sequencing technologies [20] and sophisticated denoising algorithms [21,22], accurate long sequences are becoming available in a high-throughput manner. These sequences could be used for more reliable and detailed analyses. For example, in metagenomics, analysis on species and strain levels using accurate, full-length 16S sequences could lead to a better understanding of the diversity of microbiomes [23]. However, the alignment-free methods mentioned above only provide rough approximations to the alignment distances [24]. Therefore, more accurate algorithms are demanded.

Later, machine learning techniques were adopted to improve the accuracy of alignment-free comparison by building data-dependent models for sequences. In contrast to data-independent methods like k-mer, the model optimized by training can predict the alignment distances for sequences of the same type with much higher accuracy at speed close to the k-mer method. Two methods have been developed for pairwise sequence comparison, namely SENSE [24] and NeuroSEED [25]. They both use neural networks to map sequences into a high dimension embedding space, such that the embedding distance between a pair of embedding vectors is close to the alignment distance between their sequences. The mapping functions are learned by training a Siamese network [26]. They have achieved great accuracy and efficiency on short, uniform length sequences. However, our experiment in section 3.3 shows that their performance on long, varying length sequences is still inadequate due to two weaknesses. First, their embedding vectors are not robust to indels (*insertions and deletions*) that are prevalent in biological sequences. As early attempts of using deep learning for sequence comparison, they adopted network structures used in other domains, such as computer vision and natural language processing [27–30], which are unsuitable for processing biological sequences. Although gated recurrent units and attention transformers have found great success in processing sequential natural language data [31, 32], NeuroSEED has proved they can not compete with con-

volutional neural networks (CNN) on alignment distance estimation between biological sequences. Second, biological sequences usually have varying lengths due to small indels, even for similar pairs. However, SENSE only takes sequences of a constant length, and NeuroSEED uses zero padding to extend sequences to a uniform length, which is not biologically meaningful and often leads to bad performance. These shortcomings greatly limit their application.

We developed a novel neural network architecture to learn the mapping function to address these issues. Specifically, we introduced the approximate string matching (ASM) [33] into the artificial neural networks framework to handle the indels. Because ASM measures the similarity between a pattern and a sub-sequence using edit-distance, the embeddings are robust to indels. The patterns can be learned automatically through training, while the patterns used in traditional ASM are hand-crafted. Furthermore, the network can handle varying length sequences without modifying the sequences (*padding and trimming*). We developed an efficient algorithm to compute the gradients for the parameters, such that the parameters could be optimized by training a Siamese network using backpropagation. We then performed a large-scale benchmark experiment for alignment distance estimation on four curated datasets and one uncurated dataset of amplicon sequences. We also performed a taxonomy assignment experiment on the uncurated dataset. The results demonstrated that our method outperformed other state-of-the-art alignment-free methods with a large gap across all the datasets on the two experiments in terms of the mean relative error (MRE) and taxonomy prediction accuracy, respectively, with comparable efficiency. Furthermore, our model achieved much lower MRE than NeuroSEED on the datasets of full-length 16S rRNA and 23S rRNA sequences which proved the advantage of our model in handling varying length sequences and the robustness to indels. We believe our method could be used to develop a general model that can quickly and reliably predict the pairwise alignment distance of *any* pair of full-length 16S rRNA sequences with high accuracy.

2 Methods

We propose a novel neural network for approximate string matching to map sequences to embedding vectors to address the weaknesses mentioned in the previous section. The network uses a set of patterns and gap penalties as parameters to extract features. These parameters can be optimized by training a Siamese network [26] using the gradient descent algorithm. We start with a brief introduction to the siamese neural network and the ASM algorithm. Then we will give a detailed explanation of the network structure and the gradient computation method.

2.1 Siamese Neural Network

We start by giving a brief introduction to the Siamese neural network. Fig. 1 presents an overview of the proposed method. Siamese neural network is a class of network architectures that consist of two identical networks, taking a sequence pair as one training sample. Given a pair of sequences $(\mathcal{S}_1, \mathcal{S}_2)$, each network takes one sequence as input and outputs $f(\mathcal{S}_1|\Theta)$ and $f(\mathcal{S}_2|\Theta)$ as the embedding vectors of the two sequences, respectively. Here, f is the mapping function, and Θ is the parameters of the network to be optimized. Then, we compute the alignment distance $d_a(\mathcal{S}_1, \mathcal{S}_2)$ derived from

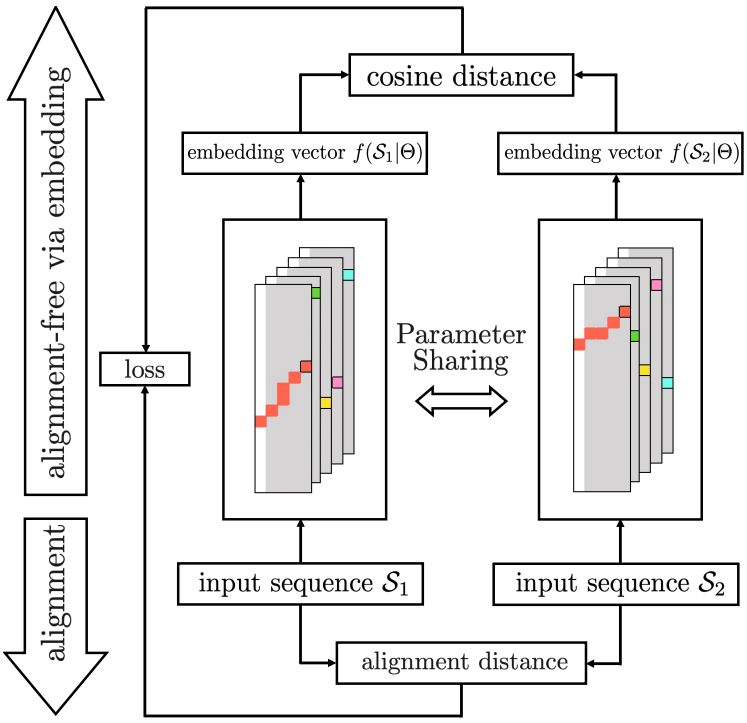


Figure 1: Overview of the proposed ASM method and its training process. Siamese network structure and the approximate string matching structure with a set of patterns.

the global alignment by using the Needleman-Wunsch (NW) algorithm. We compute the cosine distance [34] $d_e(f(\mathcal{S}_1|\Theta), f(\mathcal{S}_2|\Theta))$ as the embedding distance. To train the network parameters, we use the gradient descent method to minimize the mean squared error between the alignment distance and the embedding distance, given by:

$$\mathcal{L}(\Theta) = \sum_{i,j}(d_e(f(\mathcal{S}_i|\Theta), f(\mathcal{S}_j|\Theta)) - d_a(\mathcal{S}_i, \mathcal{S}_j))^2. \tag{1}$$

The two networks are forced to share parameters in the training process, including both initialization and gradient descent updates. Once the model is optimized, one of the networks can be used for sequence comparison.

2.2 Approximate String Matching

The main novelty of our network structure is that we use approximate string matching (ASM) to extract features that are robust to insertions and deletions. In this section, we give a brief description of the ASM algorithm. Let $\mathcal{S} = s_1 \dots s_L$ be a sequence and $\mathcal{P} = p_1 \dots p_M$ be a pattern. We aim to identify a sub-sequence in $\mathcal{S}$ that has the minimum edit distance to pattern $\mathcal{P}$ among all subsequences of $\mathcal{S}$. Dynamic programming provides the optimal solution for this purpose [33]. Specifically, we first construct a scoring matrix $\mathbf{F}$ of size $(M+1) \times (L+1)$, where the (m, ℓ)-th entry $F_{m,\ell}$ is the minimum edit distance between the first m characters of $\mathcal{P}$ and any sub-sequence

$\mathcal{S}_{\ell',\ell} = s_{\ell'} \dots s_\ell$ of $\mathcal{S}$ that ends at position ℓ. The scoring matrix can be constructed recursively:

$$F_{m,\ell} = \begin{cases} 0 & \text{if } m = 0 \\ m, & \text{if } \ell = 0 \\ \min\left(F_{m-1,\ell-1} + c, F_{m-1,\ell} + 1, F_{m,\ell-1} + 1\right) & \text{otherwise,} \end{cases} \tag{2}$$

where c is the substitution cost that takes a value of 0 if $p_m = s_\ell$ and 1 otherwise. Once $\mathbf{F}$ is computed, the minimum value in the last row is the optimal score, and the best-matched sub-sequence can be identified through backtracking. The computational complexity is on the order of $\mathcal{O}(ML)$. Since pattern $\mathcal{P}$ is usually much shorter than sequence $\mathcal{S}$, the scoring matrix can be computed efficiently. Moreover, unlike CNN, approximate string matching uses the edit distance to measure the similarity between a pattern and a sub-sequence, *allowing deletions and insertions*. Thus, it is well suited for extracting pertinent information from biological sequences.

2.3 Novel Neural Network for Approximate String Matching

A major issue with approximate string matching for our application is that patterns of interest are generally unknown and can only be pre-determined heuristically. To address the issue, we propose a novel neural network for approximate string matching that enables us to learn patterns from data *automatically*. To cast the problem into a continuous optimization problem, some modifications are merited. First, we use one-hot coding to transform input sequence $\mathcal{S}$ into an $L \times d$ matrix $\mathbf{S} = [\mathbf{s}_1^T, \cdots, \mathbf{s}_L^T]^T$. Here, each letter is represented by a d-dimensional row vector and d is the size of an alphabet ($d = 4$ for DNA sequences and $d = 20$ for protein sequences). Similarly, pattern $\mathcal{P}$ is transformed into an $M \times d$ matrix $\mathbf{P} = [\mathbf{p}_1^T, \cdots, \mathbf{p}_L^T]^T$, where the value of each element can be learned through training (detailed below). Second, to accustom the ASM to the activation functions in neural networks, we modify the recursion to maximize the alignment score. We use $\mathbf{p}_m \mathbf{s}_\ell^T$ as a score of matching the m-th vector of the pattern with the ℓ-th vector of the input sequence, and we use negative values to penalize indels. Third, while in the original approximate string matching algorithm, the gap penalties is set to 1, following the work of [35], we assign a learnable parameter to each pattern as its gap penalty to make the constructed neural network more generally applicable. With the above modifications, Eq. (2) is re-formulated as:

$$F_{m,l} = \begin{cases} 0, & \text{if } m = 0 \\ -mg, & \text{if } \ell = 0 \\ \max\left(F_{m-1,\ell-1} + \mathbf{p}_m \mathbf{s}_\ell^T, F_{m-1,\ell} - g, F_{m,\ell-1} - g\right), & \text{otherwise,} \end{cases} \tag{3}$$

where g is the attached gap penalty. Using the above-modified approximate string matching, we constructed a neural network to map input sequences into embedding vectors. Specifically, given N patterns $\mathbf{P}_1, \dots, \mathbf{P}_N$, the model generates N scoring matrices $\mathbf{F}_1, \dots, \mathbf{F}_N$. The model then uses the global max-pooling on the last row of each scoring matrix $\mathbf{F}_n$, $1 \leqslant n \leqslant N$ to extract the maximum value v_n. These values are then concatenated as a vector $\mathbf{v}$ to compute the embedding vector $\mathbf{u} = \text{ReLU}(\mathbf{v} + \mathbf{b})$ by using the ReLU activation function [36], where $\mathbf{b}$ is the associated bias terms. To learn the patterns and gap penalties, the neural networks are trained using back-propagation.

Because the patterns are independent from each other, the derivative of the error defined in Eq. (1) with respect to the patterns and gap penalties could be computed as: $\frac{\partial \mathcal{L}(\Theta)}{\partial \mathbf{P}_n} = \frac{\partial \mathcal{L}(\Theta)}{\partial v_n}\frac{\partial v_n}{\partial \mathbf{P}_n}$ and $\frac{\partial \mathcal{L}(\Theta)}{\partial g_n} = \frac{\partial \mathcal{L}(\Theta)}{\partial v_n}\frac{\partial v_n}{\partial g_n}$, $1 \leqslant n \leqslant N$. However, v_n is not differentiable as a function of $\mathbf{P}_n$ and g_n. Therefore, we developed an algorithm to compute approximate alternatives of the terms $\partial v_n/\partial \mathbf{P}_n$ and $\partial v_n/\partial g_n$, $1 \leqslant n \leqslant N$ and implemented it to fit in the automatic gradient computation in the machine learning framework PyTorch [37].

2.4 Gradient Computation

We note that, although the computation of the scoring matrix $\mathbf{F}$ corresponds to a given pattern $\mathbf{P}$ depends on its entire input sequence, only the pattern and the best-matched sub-sequence is involved in the computation of the output of the global max-pooling v. Because v is equal to the global alignment score [1] between the pattern and its best-matched sub-sequence by definition [33], we can use the differentiable Needleman-Wunsch algorithm [35] to approximate the value v and compute the partial derivatives with respect to the pattern and the gap penalty. Specifically, we replace the max function with the generalized maximization operator [38]: $\max^\gamma(\mathbf{x}) = \gamma \log(\sum_j \exp(x_j/\gamma))$, where $\gamma > 0$ is a smoothing parameter. Suppose the best-matched sub-sequence of pattern $\mathbf{P}$ has a length of $\tilde{L}$, we index it as $\tilde{\mathbf{S}} = [\tilde{\mathbf{s}}_1^T, \cdots, \tilde{\mathbf{s}}_{\tilde{L}}^T]^T$ for simplicity. We construct an $(M+1) \times (\tilde{L}+1)$ matrix $\tilde{\mathbf{F}}$ as follows:

$$\tilde{F}_{m,\ell} = \begin{cases} -mg, & \text{if } \ell = 0 \\ -\ell g, & \text{if } m = 0 \\ \max^\gamma\left(\tilde{F}_{m-1,\ell-1} + \mathbf{p}_m \tilde{\mathbf{s}}_\ell^T, \tilde{F}_{m-1,\ell} - g, \tilde{F}_{m,\ell-1} - g\right), & \text{otherwise.} \end{cases} \tag{4}$$

By definition, we have $\tilde{F}_{M,\tilde{L}} \approx v$, the approximation of the partial derivative with respect to each row of the pattern $\mathbf{P}$ is given by:

$$\frac{\partial v}{\partial \mathbf{p}_m} \approx \frac{\partial \tilde{F}_{M,\tilde{L}}}{\partial \mathbf{p}_m} = \sum_{\ell=1}^{\tilde{L}} \frac{\partial \tilde{F}_{M,\tilde{L}}}{\partial \tilde{F}_{m,\ell}} \frac{\partial \tilde{F}_{m,\ell}}{\partial \tilde{F}_{m-1,\ell-1}} \cdot \tilde{\mathbf{s}}_\ell, \forall m \in [1, \cdots, M]. \tag{5}$$

Where the terms $\partial \tilde{F}_{M,\tilde{L}}/\partial \tilde{F}_{m,\ell}$ are computed recursively as:

$$\frac{\partial \tilde{F}_{M,\tilde{L}}}{\partial \tilde{F}_{m,\ell}} = \frac{\partial \tilde{F}_{M,\tilde{L}}}{\partial \tilde{F}_{m+1,\ell}} \frac{\partial \tilde{F}_{m+1,\ell}}{\partial \tilde{F}_{m,\ell}} + \frac{\partial \tilde{F}_{M,\tilde{L}}}{\partial \tilde{F}_{m,\ell+1}} \frac{\partial \tilde{F}_{m,\ell+1}}{\partial \tilde{F}_{m,\ell}} + \frac{\partial \tilde{F}_{M,\tilde{L}}}{\partial \tilde{F}_{m+1,\ell+1}} \frac{\partial \tilde{F}_{m+1,\ell+1}}{\partial \tilde{F}_{m,\ell}}. \tag{6}$$

Likewise, for $\partial \tilde{F}_{m,\ell}/\partial g$, the partial derivative with respect to g is calculated as:

$$\frac{\tilde{F}_{m,\ell}}{\partial g} = \frac{\partial \tilde{F}_{m,\ell}}{\partial \tilde{F}_{m-1,\ell}} \frac{\tilde{F}_{m-1,\ell}}{\partial g} + \frac{\partial \tilde{F}_{m,\ell}}{\partial \tilde{F}_{m,\ell-1}} \frac{\tilde{F}_{m,\ell-1}}{\partial g} + \frac{\partial \tilde{F}_{m,\ell}}{\partial \tilde{F}_{m-1,\ell-1}} \frac{\tilde{F}_{m-1,\ell-1}}{\partial g}. \tag{7}$$

The partial derivative of an entry with respect to one of its left, upper, and upper-left neighbors is computed using Eq. (4).

3 Experiments

We performed a large-scale benchmark study to demonstrate the effectiveness and efficiency of the proposed method.

3.1 Sequence Datasets

Table 1 summarizes the five sequence datasets that we used in the study. The Qiita dataset was generated from 66 skin, saliva, and feces microbiome samples collected from Yanomami, the uncontacted Amerindians [39]. It consists of 6,734,572 sequences of 151 bps in length covering the V4 hyper-variable region of the 16S rRNA gene. The RT988 dataset contains 4,119,942 sequences from 90 oral plaque microbiome samples [40]. The sequences have 464-465 bps covering the V3-V4 hyper-variable region of the 16S rRNA gene. Both datasets were generated by Illumina MiSeq, and before the analysis, pre-processing was performed, including pair-end joining, quality filtering, and length filtering. The Greengenes dataset was extracted from the Greengenes database [41] and consisted of 925,718 unique full-length 16S rRNA gene sequences with a length ranging from 1,325 to 1,500 bps. The Silva-23S dataset was downloaded from the Silva database [42], containing 39,176 full-length 23S rRNA gene sequences with a length ranging from 2,750 to 3,150 bps. The Zymo dataset was generated from a Zymo mock community sequenced by PacBio CCS sequencing technology [43]. We used the *removePrimers* function obtained from the DADA2 R package [22] to remove primers and orient all the sequences in the forward direction. After pre-processing, the Zymo dataset contains 69,367 sequences with a length ranging from 1,187 to 1,518 bps.

Table 1: Summary of the five sequence datasets used in the study.

Dataset	# samples	# reads	sequence length
Qiita	66	6,734,572	151
RT988	90	4,119,942	464-465
Greengenes	N/A	925,718	1,325-1,500
Silva-23S	N/A	39,176	2,750-3,150
Zymo	N/A	69,367	1,187-1,518

3.2 Experimental settings

We compared our method with k-mer and six other state-of-the-art alignment-free methods, namely, ACS [10], Kr [11], FFP [9], kmacs [12], SENSE [24], and NeuroSEED. When evaluating an alignment-free method, estimation accuracy and computational efficiency are two major considerations. Thus, we calculated the mean relative error (MRE) between alignment distances and distances estimated by an alignment-free method and recorded its computational time. Compared to the mean squared error, the MRE measures the extent to which estimated distances deviate from alignment distances, *regardless of* the values of alignment distances. Since it is computationally infeasible to align all the sequence pairs in a dataset, we randomly sampled $1,000$ sequences without replacement and calculated the alignment distances of all $499,500$ possible sequence pairs. The Needleman-Wunsch algorithm [1] with the default setting was used for sequence alignment (match score $= 2$, mismatch score $= -3$, gap opening score $= -5$, gap extension score $= -2$). To minimize random variations, the above sampling process was repeated ten times. Therefore, we generated ten test datasets from each sequence dataset to evaluate the seven competing methods.

For kmacs, FFP (V3.19), Kr (V2.0.2), and NeuroSEED, we used the source code downloaded from the paper websites. Since kmacs is an extension of ACS, we used the kmacs binary by setting $k = 0$ as the implementation of the ACS method. For k-mer, we used our C++ implementation, which is optimized for amplicon sequence data by using a sparse k-mer count representation. For k-mer, kmacs and FFP, different parameters can be applied. Since there is no principal way to estimate the optimal parameter, we tested ten parameters for each method and recorded the best result. For SENSE, we used the default parameters. For NeuroSEED, We tested the settings that achieved the best results in the original paper. And for ASM, we set the number of pattern filters to 300 and the pattern length to 20. For ASM, SENSE, and NeuroSEED, we need to train a model for each sequence dataset. Since SENSE can only be applied to sequences of roughly equal length, it was trained and tested only on the Qiita and RT988 datasets. To form a training dataset, we randomly sampled sequences from a dataset without replacement and calculated the alignment distances of 5×10^6 sequence pairs. To prevent information leakage, we verified that none of the sequences used for training were in test data. The Adam optimizer [44] was employed to train the Siamese neural networks in SENSE and ASM, where we set the learning rate to 1e-4 and the number of training epochs to 200. All the experiments were performed on a 3.3 GHz Quad-Core Intel Core i5 with 16GB memory.

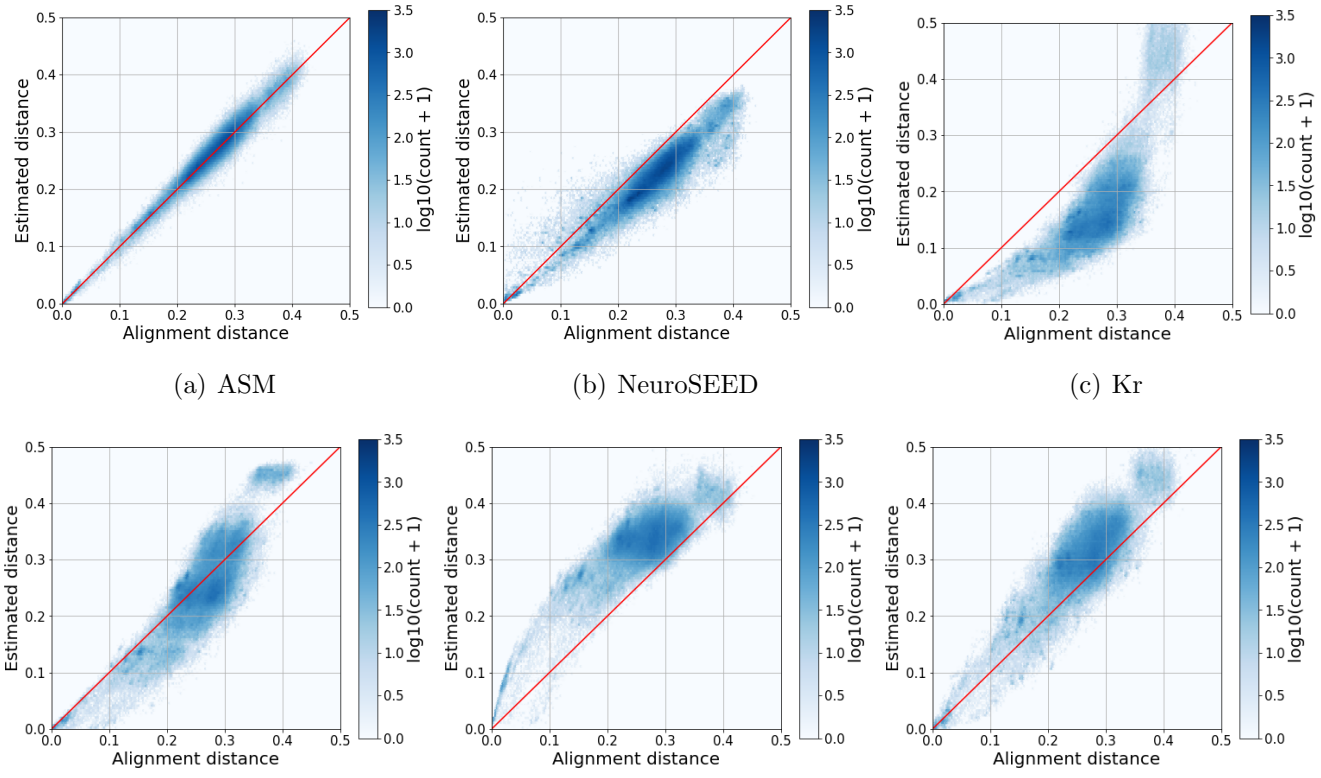
Table 2: CPU time (in second) and MRE (%) of seven methods performed on Qiita and RT988 datasets. The numbers in parentheses are standard deviations. When different parameters were used for a method, the best result was boldfaced. A p-value was computed by comparing the MRE result of ASM with the best result of a method.

		Qiita			RT988		
Method	**Parameter**	**CPU time**	**MRE**	***p*-value**	**CPU time**	**MRE**	***p*-value**
ASM	default	16.1 (0.3)	**4.5 (0.1)**	–	21.6 (0.7)	**2.8 (0.1)**	–
SENSE	default	9.7 (0.3)	**5.2 (0.1)**	5.2e-21	18.6 (0.1)	**3.9 (0.2)**	8.0e-12
NeuroSEED	default	6.2 (0.4)	**7.3 (0.1)**	4.5e-23	10.8 (0.5)	**8.7 (0.1)**	3.4e-30
ACS	default	14.7 (0.2)	**51.4 (0.6)**	5.9e-33	44.1 (1.3)	**69.5 (0.7)**	4.7e-34
Kr	default	61.6 (4.3)	**189.1 (8.1)**	3.0e-23	113.8 (1.6)	**22.4 (0.4)**	9.6e-30
kmacs	$k = 1$	22.1 (0.6)	21.2 (0.3)		62.7 (1.5)	27.2 (0.2)	
	$k = 2$	24.2 (0.5)	**10.2 (0.1)**	5.8e-27	69.8 (1.2)	**16.4 (0.1)**	1.8e-34
	$k = 3$	25.8 (0.2)	14.4 (0.2)		76.5 (1.6)	21.6 (0.4)	
	$k = 4$	27.6 (0.2)	23.7 (0.2)		82.5 (1.7)	31.2 (0.5)	
	$k = 5$	29.5 (0.4)	31.2 (0.2)		88.2 (2.5)	39.0 (0.6)	
	$k = 6$	31.6 (0.4)	37.0 (0.2)		93.4 (2.7)	45.0 (0.5)	
	$k = 7$	33.1 (0.9)	41.7 (0.2)		96.5 (1.7)	50.1 (0.5)	
	$k = 8$	34.5 (0.8)	45.7 (0.2)		100.0 (1.3)	54.1 (0.5)	
	$k = 9$	35.9 (1.0)	49.2 (0.2)		104.4 (2.2)	57.1 (0.4)	
	$k = 10$	37.0 (0.8)	52.2 (0.2)		108.1 (2.9)	59.6 (0.4)	
***k*-mer**	$k = 3$	3.1 (0.1)	**14.9 (0.3)**	3.1e-26	3.3 (0.1)	**36.8 (0.6)**	2.6e-27
	$k = 4$	5.3 (0.2)	65.8 (0.6)		8.3 (0.3)	62.7 (1.6)	
	$k = 5$	6.4 (0.3)	129.5 (1.1)		14.0 (0.1)	147.8 (1.5)	
	$k = 6$	6.7 (0.3)	167.3 (1.4)		16.9 (0.1)	219.1 (1.8)	
	$k = 7$	6.6 (0.2)	188.4 (1.7)		17.9 (0.2)	267.6 (2.2)	
	$k = 8$	6.5 (0.2)	200.7 (1.9)		18.0 (0.1)	301.4 (2.7)	
	$k = 9$	6.5 (0.2)	201.2 (2.0)		18.0 (0.1)	302.1 (2.7)	
	$k = 10$	6.5 (0.2)	201.4 (2.0)		17.9 (0.1)	301.0 (2.6)	
	$k = 11$	6.5 (0.2)	201.4 (2.0)		17.9 (0.1)	300.0 (2.6)	
	$k = 12$	6.4 (0.2)	201.4 (2.0)		17.9 (0.1)	299.0 (2.5)	
FFP	$l = 6$	2.9 (0.1)	87.9 (0.2)		2.9 (0.01)	93.0 (0.1)	
	$l = 7$	5.0 (0.1)	90.0 (0.1)		4.9 (0.05)	85.4 (0.1)	
	$l = 8$	9.8 (0.2)	38.7 (0.3)		9.9 (0.02)	83.2 (0.1)	
	$l = 9$	17.4 (0.7)	**38.5 (0.5)**	4.3e-32	18.0 (0.05)	78.0 (0.2)	
	$l = 10$	33.2 (1.1)	97.4 (1.0)		35.5 (0.04)	**18.4 (0.5)**	3.1e-25
	$l = 11$	61.2 (2.4)	137.5 (1.4)		64.9 (0.05)	95.6 (1.1)	
	$l = 12$	121.6 (4.5)	162.8 (1.8)		123.7 (0.4)	169.8 (1.7)	
	$l = 13$	233.1 (6.4)	178.9 (2.0)		219.8 (1.0)	220.1 (2.2)	
	$l = 14$	428.3 (5.9)	189.4 (2.1)		353.4 (3.4)	251.0 (2.6)	
	$l = 15$	692.5 (12.0)	196.4 (2.2)		493.4 (7.0)	274.2 (3.0)	

3.3 Benchmark Study on Accuracy and Efficiency

We first applied the eight methods to the Qiita and RT988 datasets, for which sequences cover only a sub-region of the 16S rRNA gene and have nearly the same lengths. Table 2 reports the MREs and CPU time of the eight methods, obtained by averaging over ten runs. Supplementary Fig. 1 and 2 present the estimated distances against the alignment distances for the two datasets, respectively. In terms of prediction accuracy, ASM performed slightly better than SENSE and NeuroSEED and significantly outperformed the best results of all other methods by a large margin on both datasets. In terms of running time, while k-mer is the fastest algorithm, its performance is not comparable to our method.

We next applied the competing methods to the Greengenes and Silva-23S datasets, where the sequences have varying lengths. Since there is no biologically meaningful way to trim sequences to an equal length, SENSE cannot be applied. Table 3 reports the MREs and CPU time of the seven methods. Fig. 2 and Supplementary Figs. 3, 4 present the plots of the estimated distances against the alignment distances for the seven methods on the two datasets. Similar to the results obtained on the Qiita and RT988 datasets, the prediction accuracies of ASM were significantly better than the best results of all other competing methods. We noted NeuroSEED performed even worse than kmacs, which is data-independent. It is likely because the CNN structure cannot handle indels.



(a) ASM (b) NeuroSEED (c) Kr

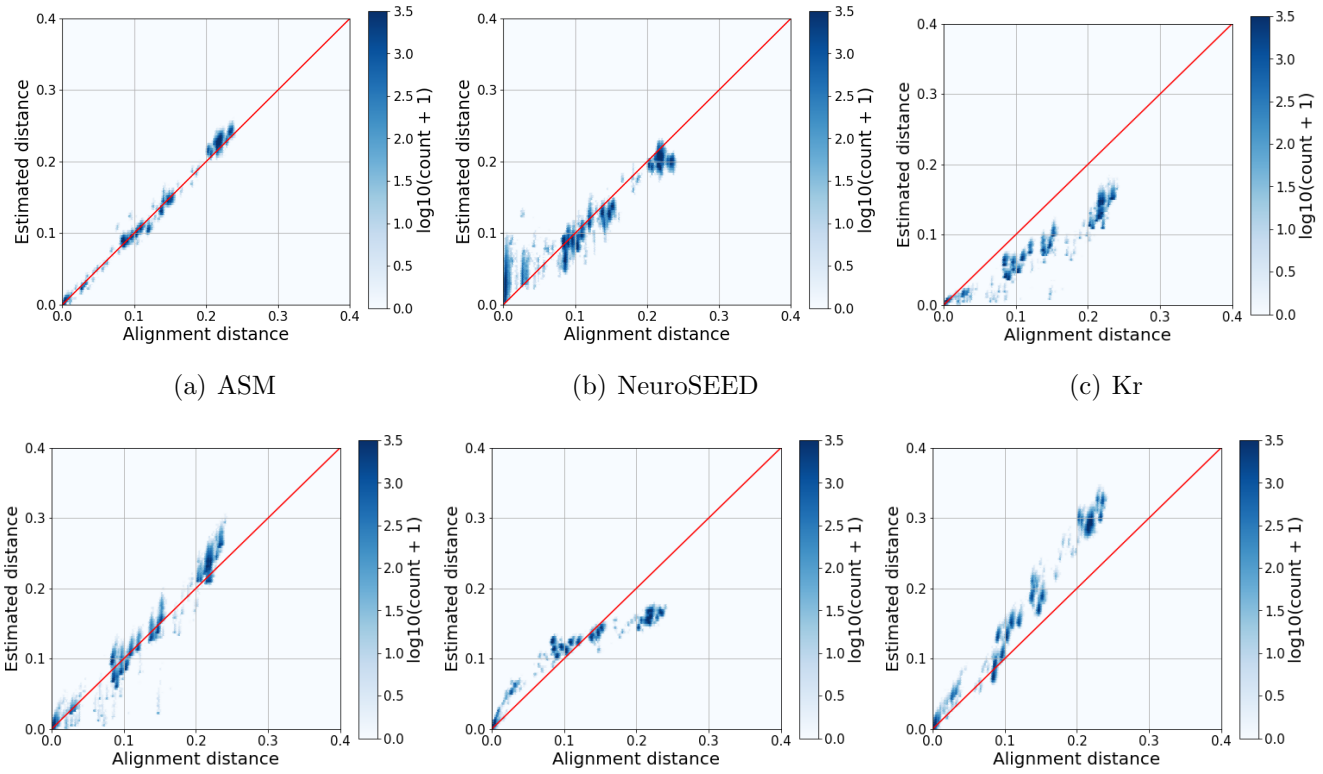
(d) kmacs (k=2) (e) k-mer (k=5) (f) FFP (l=12)

Figure 2: Visualization of alignment distances versus estimated distances computed by six methods performed on the Greengenes dataset with lower MREs. Each dot represents a sequence pair, and the color of a hex bin represents the number of sequence pairs in the bin.

Table 3: CPU time (in second) and MRE (%) of six methods performed on Greengenes, Silva-23S, and Zymo datasets. (∗The model used in the Greengenes study was tested on Zymo dataset)

		Greengenes			Silva-23S			Zymo*		
Method	**Parameter**	**CPU time**	**MRE**	***p*-value**	**CPU time**	**MRE**	***p*-value**	**CPU time**	**MRE**	***p*-value**
ASM	default	54.0 (0.9)	**3.6 (0.2)**	–	97.8 (2.3)	**4.0 (0.1)**	–	57.3 (1.0)	**10.6 (0.3)**	–
NeuroSEED	default	22.2 (1.2)	**17.0 (0.2)**	1.8e-29	65.2(1.1)	**21.2(0.3)**	1.1e-29	23.1 (0.9)	**134.0 (3.4)**	1.1e-26
ACS	default	129.0 (3.8)	**43.1 (0.5)**	2.8e-32	277.0 (5.0)	**58.5 (0.4)**	3.3e-36	133.1 (4.6)	**86.9 (0.9)**	2.8e-33
Kr	default	140.2 (8.3)	**41.7 (0.4)**	8.7e-33	162.3 (4.3)	**41.9 (2.1)**	2.1e-21	173.5 (12.3)	**42.0 (0.3)**	3.2e-32
kmacs	$k = 1$	189.1 (0.9)	16.8 (0.3)		419.5 (13.1)	30.7 (0.3)		191.3 (0.8)	36.4 (0.7)	
	$k = 2$	210.1 (1.7)	**12.7 (0.2)**	1.6e-29	464.2 (7.4)	18.2 (0.3)		214.7 (1.0)	**24.4 (0.4)**	6.5e-27
	$k = 3$	229.7 (1.0)	18.0 (0.3)		510.7 (11.7)	**13.0 (0.4)**	2.0e-22	231.9 (0.8)	28.3 (0.3)	
	$k = 4$	249.1 (2.8)	25.4 (0.3)		553.5 (9.6)	14.0 (0.4)		248.2 (0.3)	37.6 (0.4)	
	$k = 5$	269.7 (4.7)	31.8 (0.3)		594.7 (10.8)	19.8 (0.4)		263.6 (0.3)	44.1 (0.4)	
	$k = 6$	294.3 (9.6)	36.8 (0.3)		642.5 (6.0)	25.5 (0.4)		280.7 (0.8)	48.3 (0.4)	
	$k = 7$	303.2 (1.9)	41.0 (0.3)		676.2 (8.5)	30.3 (0.4)		293.0 (0.5)	51.9 (0.4)	
	$k = 8$	317.3 (2.1)	44.3 (0.3)		708.5 (10.8)	34.5 (0.4)		304.4 (0.2))	55.0 (0.4)	
	$k = 9$	342.6 (16.0)	47.2 (0.3)		749.7 (20.8)	38.2 (0.4)		316.8 (0.6)	57.6 (0.4)	
	$k = 10$	362.6 (25.1)	49.6 (0.2)		773.2 (9.6)	41.3 (0.4)		327.4 (0.3)	59.7 (0.4)	
***k*-mer**	$k = 3$	3.4 (0.01)	67.7 (0.3)		5.84 (0.1)	78.9 (0.4)		3.5 (0.0)	53.7 (0.4)	
	$k = 4$	9.9 (0.1)	35.7 (0.3)		14.1 (0.2)	66.6 (0.4)		10.0 (0.4)	**40.1 (0.4)**	2.9e-39
	$k = 5$	26.7 (0.2)	**30.1 (0.5)**	8.5e-29	32.2 (0.4)	49.3 (0.4)		26.8 (0.3)	107.0 (1.4)	
	$k = 6$	44.0 (0.1)	103.5 (0.7)		46.4 (0.8)	**25.7 (0.5)**	2.5e-28	43.3 (0.1)	200.7 (1.7)	
	$k = 7$	51.2 (0.7)	154.2 (0.8)		62.8 (0.9)	41.4 (1.1)		51.0 (0.1)	271.7 (1.9)	
	$k = 8$	52.6 (0.5)	181.0 (0.8)		85.3 (3.0)	101.9 (1.4)		53.0 (0.2)	318.9 (2.4)	
	$k = 9$	52.3 (0.3)	181.1 (0.8)		85.5 (1.9)	101.9 (1.4)		53.0 (0.2)	318.7 (2.4)	
	$k = 10$	52.1 (0.1)	181.1 (0.8)		85.5 (3.1)	101.9 (1.4)		53.0 (0.2)	318.7 (2.4)	
	$k = 11$	52.0 (0.0)	181.1 (0.8)		84.7 (2.0)	102.0 (1.4)		53.1 (0.3)	318.6 (2.4)	
	$k = 12$	52.1 (0.1)	181.2 (0.8)		84.9 (3.7)	102.0 (1.4)		53.1 (0.3)	318.8 (2.4)	
FFP	$l = 6$	3.2 (0.02)	98.1 (0.05)		3.6 (0.8)	98.3 (0.7)		3.1 (0.3)	98.0 (0.0)	
	$l = 7$	5.2 (0.2)	96.0 (0.05)		5.9 (0.8)	97.0 (0.7)		5.1 (0.4)	95.6 (0.0)	
	$l = 8$	11.0 (0.3)	91.6 (0.1)		12.2 (0.8)	94.6 (0.7)		10.4 (0.8)	90.3 (0.0)	
	$l = 9$	19.9 (0.7)	84.7 (0.1)		21.7 (0.8)	90.3 (0.7)		19.2 (1.6)	80.9 (0.1)	
	$l = 10$	40.7 (1.0)	93.9 (0.1)		44.0 (0.8)	84.4 (0.7)		39.1 (3.0)	86.9 (0.1)	
	$l = 11$	76.1 (0.7)	56.4 (0.2)		84.8 (0.8)	92.5 (0.7)		74.1 (6.4)	57.0 (0.3)	
	$l = 12$	149.2 (0.8)	**18.0 (0.3)**	2.6e-29	170.8 (0.8)	55.4 (0.7)		143.1 (11.8)	**42.8 (0.2)**	5.1e-31
	$l = 13$	274.1 (0.9)	77.3 (0.4)		311.8 (0.8)	**24.3 (0.7)**	5.3e-27	250.0 (12.1)	128.7 (0.7)	
	$l = 14$	527.2 (2.1)	120.0 (0.5)		590.0 (0.8)	96.1 (0.7)		382.4 (2.5)	199.1 (0.6)	
	$l = 15$	1031.8 (3.5)	147.1 (0.6)		1126.2 (0.8)	147.6 (0.7)		548.5 (15.4)	274.2 (3.0)	

We finally tested the generalization capability of ASM and NeuroSEED by training a model on the Greengenes dataset and applying it to the Zymo dataset. The results were reported in Table 3, Fig. 3, and Supplementary Fig. 5. Again, ASM significantly outperformed other approaches. We noted that, comparing to the result obtained from the Greengenes dataset, the prediction accuracies of all the methods declined on the Zymo dataset. A possible explanation is that, the distribution of alignment distances is different in these two datasets, since the species richness of Zymo is much lower than Greengenes. We noted that NeuroSEED's prediction accuracy declined dramatically among all the methods. As demonstrated in Fig. 3(b), it performed particularly bad in distinguishing subtle differences between similar sequences. We believe it is because its embedding function is too sensitive to indels. In contrast to NeuroSEED, ASM still maintained a high level of accuracy in predicting alignment distances. This suggests that it is possible to develop and publish a trained model (e.g., for full-length 16S rRNA gene sequences) that other researchers can use to process their datasets.



(a) ASM (b) NeuroSEED (c) Kr

(d) kmacs (k=2) (e) k-mer (k=5) (f) FFP (l=12)

Figure 3: Visualization of alignment distances versus estimated distances computed by the six methods performed on the Zymo dataset.

3.4 Taxonomy Assignment

To further demonstrate the effectiveness and utility of the proposed method, we conducted an experiment where we used our approach to perform taxonomy assignment of a given sequence dataset by searching against a reference database. For the purpose of this study, we constructed a query dataset by randomly sampling 3,000 sequences from the Zymo dataset. We used as the reference database the GG97 dataset, a subset of the Greengenes database [41], which is the default closed-reference database used by QIIME [45] and contains 30,592 sequences with complete taxonomy annotations at the genus level. We performed a database search using the NW algorithm and annotated each query sequence using the genus of the best-matched reference sequence. We used the result obtained by the NW algorithm as the ground truth and compared the performance of our method with the six competing methods used in the Zymo study. For kmacs, k-mer and FFP, the parameters were set to be those that achieved the best performance in the Zymo study. All the methods were performed in parallel using 4 threads. Fig. 4 presents the annotation accuracy and CPU time of the seven methods. Our method achieved 98.4% prediction accuracy, outperforming k-mer, ACS, Kr, and kmacs by about 20% and FFP, NeuroSEED by about 13%. We noticed that NeuroSEED, kmacs, and FFP rank differently from the results in the Zymo study. This is because for sequence alignment we concern about the accuracy of distance estimation for *all* sequence pairs, while for sequence annotation we are only interested in sequence pairs that are similar. In terms of

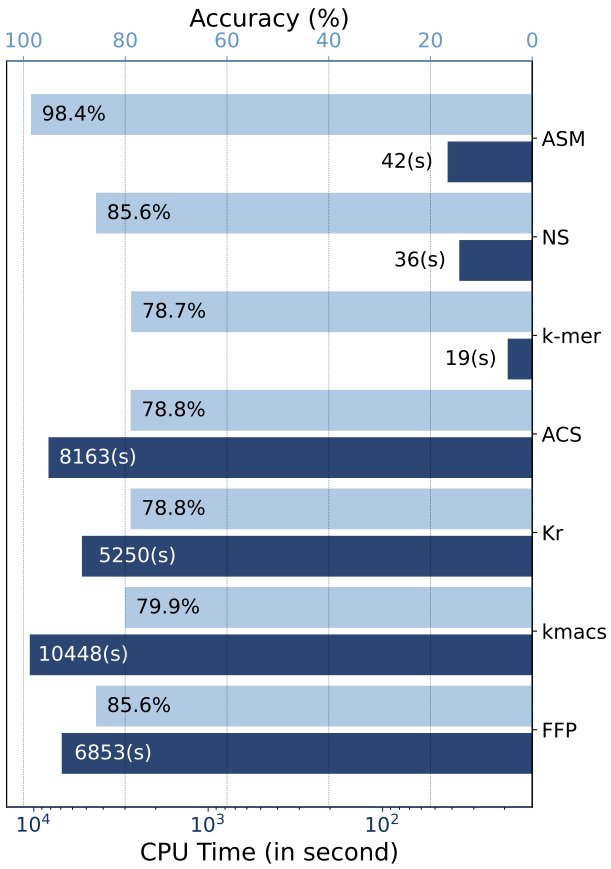


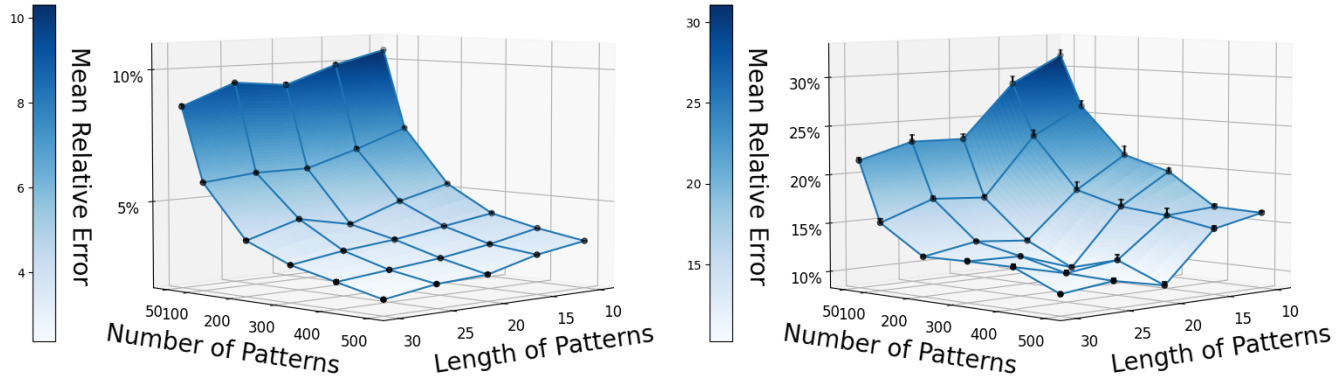
Figure 4: Comparison of taxonomy prediction accuracy and CPU time of six methods applied to the Zymo dataset. Where NS is short for NeuroSEED. The parameter settings for each method are the best case in the Greengenes study.

computational efficiency, our method performed slightly worse than k-mer, but ran two orders of magnitude faster than Kr, ACS, kmacs, and FFP. It is worth noting that, compared to the results reported in Tables 2, our method and k-mer ran much faster than the other methods. This is because, for k-mer and our method, the embedding vectors of the reference sequences can be pre-computed, while for Kr, ACS, and kmacs, the pairwise sequence comparison can only be performed in the presence of both query and reference sequences. Although FFP can pre-process the reference sequences, it is computationally expensive to compute the Kullback-Leibler distances between the query and reference sequences. In contrast, both k-mer and our method can take advantage of fast sparse matrix computation.

3.5 Parameter Sensitivity Analysis

The proposed method has two parameters, namely, the pattern number and the pattern length. We performed a parameter sensitivity analysis to investigate how the method performs with respect to the two parameters. We applied the method to the Greengenes and Zymo datasets and reported in Fig. 5 the MRE results obtained using various pattern numbers and lengths. We can see that, with the increase of the pattern length, the prediction errors dropped quickly and then flattened when

the pattern length was longer than 300. We also observed that our method achieved similar results when the pattern length was larger than 20. For the balance between accuracy and efficiency, we set the pattern number to 300 and the pattern length to 20 as the default parameters.



(a) Greengenes

(b) Zymo

Figure 5: Parameter sensitivity analysis of the proposed method performed on (a) Greengenes and (b) Zymo datasets.

4 Conclusion

We developed a neural network structure based on ASM to build a data-dependent model for accurate and efficient sequence comparison. We demonstrated the outstanding performance of our method on long, varying-length sequences through large-scale benchmark experiments. Moreover, the taxonomy assignment experiment results further proved the value of our method in practical application. We believe that our method can construct a general model for each type of widely studied homologous sequence, which could exploit the discriminatory capabilities of high-precision full-length sequence data in large-scale analysis. (e.g., for more accurate annotations of sequence databases and a better understanding of microbial phylogeny.) Possible directions for future research could be using ASM for local sequences comparison, aggregating the ASM structure to form multi-layer structures, and interpreting learned patterns.

References

[1] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[2] Andrea Sboner, Xinmeng Jasmine Mu, et al. The real cost of sequencing: higher than you think! *Genome Biology*, 12(8):1–10, 2011.

[3] Zachary D Stephens, Skylar Y Lee, et al. Big data: astronomical or genomical? *PLOS Biology*, 13(7):e1002195, 2015.

[4] Andrzej Zielezinski, Hani Z Girgis, et al. Benchmarking of alignment-free sequence comparison methods. *Genome Biology*, 20:144, 2019.

[5] Andrzej Zielezinski, Susana Vinga, et al. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology*, 18(1):186, 2017.

[6] Oliver Bonham-Carter, Joe Steele, et al. Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Briefings in Bioinformatics*, 15(6):890–905, 2014.

[7] Kai Song, Jie Ren, et al. New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, 15(3):343–353, 2014.

[8] Samuel Kariin and Chris Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends in Genetics*, 11(7):283–290, 1995.

[9] Gregory E Sims, Se-Ran Jun, et al. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106(8):2677–2682, 2009.

[10] Igor Ulitsky, David Burstein, et al. The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 13(2):336–350, 2006.

[11] Mirjana Domazet-Lošo and Bernhard Haubold. Efficient estimation of pairwise distances between genomes. *Bioinformatics*, 25(24):3221–3227, 2009.

[12] Chris-Andre Leimeister and Burkhard Morgenstern. Kmacs: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, 30(14):2000–2008, 2014.

[13] Stephen F Altschul, Warren Gish, et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[14] Robert C Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 2010.

[15] Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, 15(3):1–12, 2014.

[16] Yijun Sun, Yunpeng Cai, et al. ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences. *Nucleic Acids Research*, 37(10):e76, 2009.

[17] Yunpeng Cai and Yijun Sun. ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time. *Nucleic Acids Research*, 39(14):e95, 2011.

[18] Wei Zheng, Qi Mao, et al. A parallel computational framework for ultra-large-scale sequence clustering analysis. *Bioinformatics*, 35(3):380–388, 2019.

[19] Yunpeng Cai, Wei Zheng, et al. ESPRIT-Forest: parallel clustering of massive amplicon sequence data in subquadratic time. *PLOS Computational Biology*, 13(4):e1005518, 2017.

[20] Eric E Schadt, Steve Turner, et al. A window into third-generation sequencing. *Human Molecular Genetics*, 19(R2):R227–R240, 2010.

[21] A Murat Eren, Hilary G Morrison, et al. Minimum entropy decomposition: unsupervised oligotyping for sensitive partitioning of high-throughput marker gene sequences. *The ISME Journal*, 9(4):968–979, 2015.

[22] Benjamin J Callahan, Paul J McMurdie, et al. DADA2: high-resolution sample inference from Illumina amplicon data. *Nature Methods*, 13(7):581, 2016.

[23] Jethro S Johnson, Daniel J Spakowicz, et al. Evaluation of 16s rrna gene sequencing for species and strain-level microbiome analysis. *Nature Communications*, 10(1):1–11, 2019.

[24] Wei Zheng, Le Yang, et al. SENSE: Siamese neural network for sequence embedding and alignment-free comparison. *Bioinformatics*, 35(11):1820–1828, 2019.

[25] Gabriele Corso, Zhitao Ying, et al. Neural distance embeddings for biological sequences. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] Jane Bromley, Isabelle Guyon, et al. Signature verification using a "siamese" time delay neural network. *Advances in Neural Information Processing Systems*, 6:737–744, 1993.

[27] Tomas Mikolov, Martin Karafiát, et al. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, pages 1045–1048, 2010.

[28] Shaoqing Ren, Kaiming He, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99, 2015.

[29] Franco Scarselli, Marco Gori, et al. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[30] Antonia Creswell, Tom White, et al. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[31] Martin Sundermeyer, Ralf Schlüter, et al. Lstm neural networks for language modeling. In *INTERSPEECH*, 2010.

[32] Qiang Wang, Bei Li, et al. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

[33] Peter H Sellers. The theory and computation of evolutionary distances: pattern recognition. *Journal of Algorithms*, 1(4):359–373, 1980.

[34] Dengsheng Zhang and Guojun Lu. Evaluation of similarity measurement for image retrieval. In *International Conference on Neural Networks and Signal Processing*, pages 928–931, 2003.

[35] Satoshi Koide, Keisuke Kawano, et al. Neural edit operations for biological sequences. In *Advances in Neural Information Processing Systems*, pages 4960–4970, 2018.

[36] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pages 807–814, 2010.

[37] Adam Paszke, Sam Gross, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

[38] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International Conference on Machine Learning*, pages 894–903. PMLR, 2017.

[39] Jose C Clemente, Erica C Pehrsson, et al. The microbiome of uncontacted Amerindians. *Science Advances*, 1(3):e1500183, 2015.

[40] RJ Genco, MJ LaMonte, et al. The subgingival microbiome relationship to periodontal disease in older women. *Journal of Dental Research*, 98(9):975–984, 2019.

[41] Daniel McDonald, Morgan N Price, et al. An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. *The ISME Journal*, 6(3):610, 2012.

[42] Elmar Pruesse, Christian Quast, et al. Silva: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. *Nucleic Acids Research*, 35(21):7188–7196, 2007.

[43] Benjamin J Callahan, Joan Wong, et al. High-throughput amplicon sequencing of the full-length 16S rRNA gene with single-nucleotide resolution. *Nucleic Acids Research*, 47(18):e103, 07 2019.

[44] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–13, 2014.

[45] J Gregory Caporaso, Justin Kuczynski, et al. Qiime allows analysis of high-throughput community sequencing data. *Nature Methods*, 7(5):335–336, 2010.