KMer-Node2Vec: Learning Vector Representations of K-mers from the K-mer Graph

***Abstract*—Current alignment-free DNA sequence comparison tools, such as DNA2Vec [1] and Seq2Vec [2], have become ubiquitous and are widely used in the computational biology. However, these approaches do not widely apply to the large-scale DNA sequential retrieval due to their extremely long training time. This paper proposes a graph-based DNA embedding algorithm, called KMer-Node2Vec, which converts the large DNA corpus into a k-mer co-occurrence graph, then takes the k-mer sequence samples from this graph by randomly traveling and finally trains the k-mer embedding on this sampling corpus. We posit that because of the stable size of k-mers, the size and density of the k-mer co-occurrence graph change slightly with the increase in the training corpus. Therefore, KMer-Node2Vec has a stable runtime on the large-scale data set, and its performance advantage becomes more and more obvious with the growth of the training corpus. Extensive experiments conducted on real-world data sets demonstrate that KMer-Node2Vec outperforms DNA2Vec by 29 times in terms of efficiency in 4G data set, while the error generated by the random walk sampling is small.**

***Index Terms*—K-mer Representation, K-mer Co-occurrence, K-mer Graph, Random Walk, Sampling**

I. INTRODUCTION

DNA/RNA sequence comparison technique has been widely used in many fields, such as pathogen detection [3], species traceability [4], etc. Let us take pathogen detection as an example. After the S16 rRNA next-generation sequencing (NGS), genomic sequences are fragmented into millions of reads composed of a series of sequential bases (A-adenine; G-guanine; T-thymine; C-cytosine). Then, those DNA/RNA reads are clustered together, so that the pathogen's abundance and label can be identified for these clusters if some sequences in these clusters match well with the already known sequence in the genome sequence database using the sequence comparison technique.

The sequence comparison technique can be divided into two groups: (i) alignment-based approaches [4]–[7], and (ii) alignment-free approaches [8]–[13]. The alignment-based approaches are more accurate than alignment-free approaches, but do not scale with the very large data sets due to most of them are dependent on dynamic programming with high computational complexity. That explains why many alignment-free approaches have been proposed as alternatives to make faster sequence comparisons over the past few decades. However, most of the alignment-free approaches, which rely on heuristic rules, work at a cost of not guaranteeing the identification of the optimal and highest scoring alignment, and so often produce lower accuracy than the alignment-based approaches. Recent alignment-free research works [1], [14]–[17] inspired by the Natural Language Processing (NLP) area achieve much better accuracy. Such approaches first divide the long DNA sequence into k-mer components and then combine these k-mer vectors to present the DNA sequence. Therefore, learning continuous distributed vector representations of k-mers is very important for DNA sequence comparison. However, it is yet to be affordable to straightly apply these methods to learning DNA sequence vectors for sequence comparison, because most of these methods suffer from a huge time consumption in training k-mer embedding, especially for large-scale data sets. For example, DNA2Vec [1] needs 4.5 days to train the model on 4G DNA sequence data set.

In this paper, we address the problem of learning k-mer embedding on large-scale data sets. Our motivation comes from the word co-occurrence graph. Many researches [18], [19] in the NLP area have tried to map text to the word co-occurrence graph and capture the implicit structure of text data through the methods proposed by various graph models. Inspired by these approaches, our solution is also established on a k-mer co-occurrence graph, but different from the prior approaches, our approach do the k-mer sampling by random walk traveling on this graph. First, KMer-Node2Vec uses k-mer co-occurrence information in a DNA sequence to construct a k-mer graph whose each node as a k-mer, whose edges as co-occurrences between a k-mer and its adjacency k-mer and whose edge direction represents the order of k-mers. Second, KMer-Node2Vec performs fast and efficient random walk traveling on this k-mer graph and samples k-mer sequences. Third, the Skip-Gram model [20] has been applied to these sampled k-mer sequences for learning the final k-mer embedding.

The major advantages of KMer-Node2Vec include: (i) the k-mer graph is better to grasp the informative connection pattern of k-mers within and between sequences as it relaxes a strong constraint in locality using a sliding window to learn k-mer co-occurrence relation. (ii) the k-mer graph reduces the data redundancy for contributing to a fast k-mer sampling and vector training. (iii) the k-mer graph will not explode in size with the drastic increase in sequence data since the maximum number of nodes in the k-mer graph is 4^k and adding more training corpus only results in adjusting the edge weights, which means the size of the training samples is a constant and do not change with the increasing size of the training DNA corpus.

Contributions:

- We offer a detailed study on the k-mer co-occurrence graph with consideration of different edge weights and nodes weights.
- We present the KMer-Node2Vec approach, which is based on the k-mer co-occurrence graph to calculate the k-mer embedding. An effective sampling strategy is developed to generate the k-mer sequences, and the Skip-Gram algorithm is used to calculate the k-mer embedding on these k-mer sequences.
- Extensive experiments are conducted to evaluate the efficiency and the accuracy of the proposed algorithm. Experimental results show that the KMer-Node2Vec algorithm is capable of outperforming DNA2Vec [1] by 29 times on 4G DNA/RNA sequence data with the high accuracy (99.5%) in DNA retrival task.

Road Map: The rest of this paper is organized as follows. Section II first review the classic work on alignment-based/free approaches. Section III explains the detailed study of KMer-Node2Vec model. Section IV shows the experimental settings and empirical research. Section V summarizes the paper.

II. RELATED WORK

A. alignment-based approach

The biological sequence alignment-based search techniques have been developed over the past few decades. BLAST [4] finds similar sequences by locating the short match between two sequences. It sacrifices a certain sensitivity in exchange for a huge speed improvement. BLAST+ [6] improves the performance of BLAST by dividing the long query sequence into blocks and allows only the sequence region strongly related to the query sequence to be retrieved for the long sequence. Clustal W [21] and Cluster X [22] build alignments according to the topology of the guide tree to obtain quick matches. However, the Clustal W and Clustal X's high complexity $O(N^2)$ of the guide tree construction makes alignments inefficient. Though Clustal Omega [23] introduced an $O(NlogN)$ method called mBed to support alignments for hundreds of thousands of sequences, it is still difficult to extend to alignments for millions of sequences. FASTA [24] provides a comprehensive set of rapid similarity search tools for DNA sequence alignment, but is even slower than BLAST. Ssearch [25] uses the best local alignment search tool based on the Smith-Waterman algorithm [5], which ensures finding the best alignment score for given parameters. Glsearch [25] calculates the global alignment of the query sequence and the local alignment in the library.

However, most of these tools are inefficient to address large-scale DNA sequences for their relatively slow running time in the alignment process. Our proposed DNA sequence vector retrieval approach based on k-mer embedding reaches a Top-20 accuracy of 99.5% at a speed 3.3 times faster than BLAST+.

B. alignment-free approach

Many alignment-free (feature-based) approaches have been proposed as alternatives to alignment-based approaches. Specifically, DNA/RNA sequences are converted into real-valued vectors, and then the similarity score between these vectors is calculated for sequence similarity measurement. At present, graphics-based methods [26]–[28], chaos theory-based methods [29], [30], word frequency-based methods [9]–[12], and etc, have been used for sequence vector representations. Although the alignment-free approaches above sacrifice accuracy in exchange for speed improvement, their disadvantages are also obvious that their accuracy of sequence similarity comparison drops significantly in the big data scenario.

Fortunately, the Word2Vec technique, which achieves a good performance in NLP area, brings new opportunities to biological sequence modeling and retrieval. Asgari et al. [15] first fragment protein sequences into sub-sequences by n-gram modeling [31] and use Skip-Gram [31] for learning vector representations of these sub-sequences. Seq2Vec [16] extends BioVec's approach by Doc2Vec [14] to consider the order of k-mers that co-occur in the same sequence. Meanwhile, DNA2Vec embedding [1] can represent variable-length k-mers in a consistent fashion via nucleotide concatenation experiments. Min et al. [17] presented us k-mer embedding with GloVe [32], which is based on factorizing a matrix of word co-occurrence statistics on a local and global level.

However, these methods suffer from a huge training time on large-scale DNA data and witness their training time increase with a surge in data. For example, DNA2Vec, the prior k-mer training state of the art, costs 4.5 days to train the model only on 4G DNA sequences. To overcome this problem, we propose a graph-based method called KMer-Node2Vec, which produces a fast and efficient training of k-mer embedding for large-scale DNA sequences. The training process of KMer-Node2Vec on 4G DNA sequences needs only 3.8 hours, which outperforms DNA2Vec 29 times in terms of efficiency with the 99.5% Top-20 accuracy.

III. KMER-NODE2VEC ALGORITHM

A. Motivation

Our motivation comes from the word co-occurrence graph, which have been successfully used in the NLP tasks, such as information retrieval [33] and text classification [34]. The impact of the term order has been a popular issue, and relationships between the terms, in general, are claimed to play an important role in text processing. For example, the sentence "Lily is more beautiful than Lucy" is totally different from the sentence "Lucy is more beautiful than Lily". Thus, we can use a word co-occurrence graph representation that captures these word relationships instead of the sliding window for traditional word embedding approaches [1]. Similarly, we can extract k-mer co-occurrence information from the DNA sequences to construct a k-mer co-occurrence graph whose each k-mer as a node, whose edges represent co-occurrences between a k-mer and its adjacency k-mer, and whose edge direction represents k-mer order.

There are two prominent advantages of the word graph for representing textual data sets. (1) First, the word graph is effective in delivering textual relations. In the word graph, each node represents a unique word, and word co-occurrence information well reflects the "closeness" between words within and between documents [19]. Zuckerman et al. [18] presented one potential advantage of using a directed word graph model over context-dependent representations (i.e., Word2Vec) to preserve the word order information. (2) Second, the word graph shows scalability in representing the large-scale text: (i) because of the stable vocabulary, the size of the word graph change slightly with the increase in the training corpus; (ii) as corpus grows larger, the word's distribution (i.e., word co-occurrences) in the word graph becomes closer to that of the real text data sets.

So, training the k-mer embedding on a k-mer co-occurrence graph is an effective and efficient approach. (1) In terms of effectiveness, the k-mer graph can capture the k-mer co-occurrences within and between sequences and the k-mer order information. Meanwhile, the k-mer's distribution is very close to that of the real big data sets. (2) In terms of scalability, three points are highlighted. First, the number of nodes in this k-mer co-occurrence graph is not large since the maximum number of graph nodes is 4^k. Second, these k-mer graphs are highly sparse. Noted that the maximum edges for each node are four since there are only four kinds of base-pair (A, T, C, G). Hence, a more efficient sparse graph embedding technique [35] can be applied to these data sets. The last and most important issue is that as the size of the training corpus increases, the edge weights will change a lot, but the size of nodes and graph density do not have significant alterations. Thus, this pushes us to propose our approach, KMer-Node2Vec, to address the k-mer embedding issue of inefficiency and unscalability.

B. Framework of KMer-Node2Vec

Fig. 1 shows the framework of KMer-Node2Vec, consisting of three steps: (i) the k-mer co-occurrence graph is constructed from DNA data sets; (ii) the random walk technique has been applied to sample sequences of nodes from the k-mer graph; (iii) the sampled sequences are used as the training corpus to learn k-mer embedding by Skip-Gram algorithm.

1) K-mer Graph Construction: A DNA sequence is presented as a k-mer co-occurrence graph that corresponds to a weighted directed graph whose nodes represent unique k-mers, whose edges represent co-occurrences between the k-mers with edge direction representing the order of k-mers. An example of graph creation is given in Fig. 2. The source DNA sequence is "GAACTTCGAAGTGCCGTGAAGC" and let $k = 3$, the sequence will be represented as a series of 3-mer = {GAA, AAC, ACT, CTT, TTC, TCG, CGA, GAA, AAG, AGT, GTG, TGC, GCC, CCG, CGT, GTG, TGA, GAA, AAG, AGC}. Each 3-mer is a node of a 3-mer graph (see Fig 2(b)). An edge is drawn between a k-mer and its following k-mer. For clarity purposes, the edges are created from the node corresponding to "GAA" (see Fig. 2(a)). Fig. 2(b) corresponds to the resulting weighted directed graph where each node

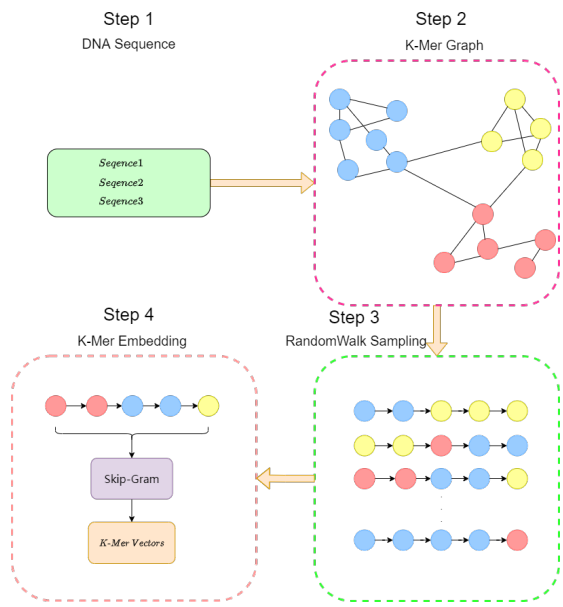


Fig. 1. Overview of KMer-Node2Vec

represents a unique 3-mer, and each edge is a co-occurrence of the two 3-mers.

Weight on edges: The number of simultaneous occurrences of two k-mers in the sequences is used as their edge weight. A weighted adjacency matrix W is used to store the edge weights, and W_{vx} is the weight from node v to node x. Note that disconnected edge weight is set as 0.

$$W_{vx} = \sum co\text{-}occurrence(v, x) \tag{1}$$

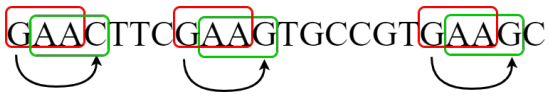
where $co\text{-}occurrence(v, x)$ is the number of times that the k-mer v, x appear together from left to right in the same order. For example, in Fig. 2(a), the number of occurrences of 3-mer pairs "GAA" and "AAG" is 2, so $W_{GAA,AAG}$ is equal to 2. In the end, we get the relationship between each k-mer in the whole sequence, and the weighted k-mer graph is shown in Fig. 2(b).

Weight on nodes: The weight of the node is used to determine the sampling times in the random walk process. We take probability PW_v as a representation of the importance of the k-mer v in the whole corpus, that is, the node weight in the graph. Intuitively, the k-mer frequency can be applied to measure the importance score.

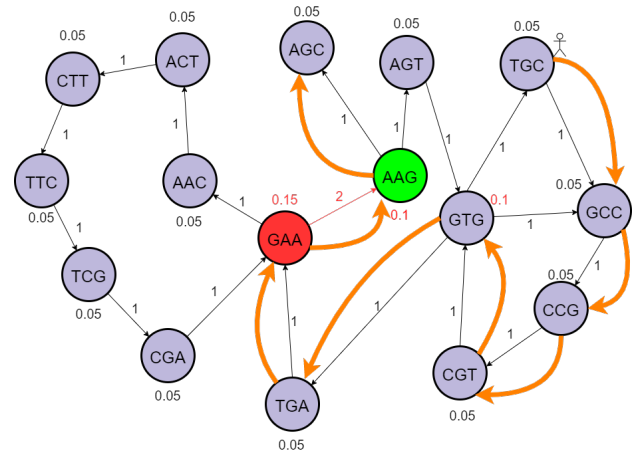
$$PW_v = \frac{n_v}{\sum_k n_k} \tag{2}$$

where PW_v is the important score for k-mer v, n_v is the number of occurrences of the k-mer v, and $\sum_k n_k$ is the number of occurrences of all k-mers. Taking Figure 2(b) as an example, the total number k-mer is equal to 20, then $PW_{GAA} = \frac{3}{20} = 0.15$.

2) K-mer Node Sampling: This random walk sampling process [35] iteratively explores the global structure network of the object to estimate the proximity between two nodes.



(a) A graph edge is directed from 3-mer node v_i to v_j if they overlap by two continuous nucleotides with v_i's suffix being the v_j's prefix, i.e., "GAA" (the red node in Fig. 2(b)) to "AAG" (the green node in Fig. 2(b)).



(b) The 3-mer graph built from the DNA sequence in Fig. 2(a). The solid line with arrows in yellow describes the path of a random walk with "TGC" as the starting node and "AGC" as the ending node. Noted that the pair of "GAA" to "AAG" occurs twice in the sequence, so that the weight of the graph edge from "GAA" to "AAG" is 2.

Fig. 2. Construction of the K-Mer Graph

Generate a random walk from the current node, and select the random neighbors of the current node as a candidate based on the transition probability on the k-mer co-occurrence graph.

Take a hypothetical case in Fig. 2(a) as an example to explain how we use our graph-based sampling strategy to sample one sequence from the k-mer graph. For example, we select TGC as the starting node for executing a random walk of walk length 8; the random walk started from TGC, passed through GCC, CCG, ..., and finally arrived at AGC (see Fig. 2(b) for the path of this random surfing). The sample of the sequence obtained by the random walk is exactly $TGCCGTGAAGC = TGC - GCC - CCG - CGT - GTG - TGA - GAA - AAG - AGC$. Thus, random walk sampling k-mer sequences can be collected in this way as the training corpus for learning the k-mer embedding. Two points are worth highlighting in this sampling process.

First, each starting node should execute sufficient random walks to obtain a relatively large and qualified sample for k-mer training. Particularly, a node's number of walks should be in line with its role in the graph, as important nodes should enjoy more random walks than the others. That is why we assign each graph node a node weight (see Eq. 2) that reflects a node's importance to affect each node's number of walks. The sampling times of node v as the starting node are shown in Eq. 3:

$$number_walks(v) = \lfloor total_walks \times PW_v \rfloor \tag{3}$$

where $total_walks$ is the total number of random walk samplings. For example, in the Fig. 2, let $total_walk = 200$, $number_walks(AAG)$ should be $200 \times 0.15 = 30$. That means there should be 30 sampled sequences of k-mer nodes that start from the node AAG.

Second, to capture both homophily and structural similarity between k-mer nodes from the graph, we employ Node2Vec [35] as our sampling method. Node2Vec devises a second order random walk as the sampling strategy. Consider a random walk that just traversed edge (t, v) and now resides at node v. We denote x as the potential next node, v as the current node, and t as the previous node. Every starting node has its biased walks ruled by the following transition probabilities:

$$P(x \mid v, t) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } W_{vx} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, W_{vx} denotes the edge weight between node v and node x from Eq. 1, and Z is the normalizing constant. π_{vx} is defined as:

$$\pi_{vx} = \alpha_{pq}(t, x) W_{vx} \tag{5}$$

where W_{vx} is the weight between edge (v, x) from Eq. 1, and the bias factor $\alpha_{pq}(.)$ is defined as:

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } t = x \\ 1 & \text{if } t \neq x \text{ and } W_{tx} \neq 0 \\ \frac{1}{q} & \text{if } t \neq x \text{ and } W_{tx} = 0 \end{cases} \tag{6}$$

where p is the return parameter, q is the in-out parameter and W_{tx} represents the edge weight between node t and node x from Eq. 1. By controlling p and q [35], the random walks can interpolate between BFS and DFS in the graph. The Alias table [36] is used to sample nodes, which can reduce the sampling time complexity to $O(1)$.

3) Training K-mer embedding with the Skip-Gram model: The Skip-Gram Model is used to train the k-mer embedding on the sampling corpus. It is obvious that if two different k-mers have very similar "contexts" (that is, what k-mers are

likely to appear around them), then our model needs to output very similar vector representations for such two k-mers. For example, "CCTAYGGGRBGCASCA" and "CTAYGGGRBGCASCAG" always appear together since these 16-mers are the bacterial conserved region, so that they should have similar vector representations. Similarly, those related k-mers, which always appear in the conserved region, would probably have similar vector representations as well.

4) KMer-Node2Vec Algorithm: The algorithm is given in Algorithm 1. This algorithm first constructs the k-mer graph by scanning the DNA data sets. Then, it samples a set of k-mer sequences from the k-mer graph by random walk traverling. Finally, k-mer embedding is learned from these sequences using the Skip-Gram model.

Algorithm 1: *KMer-Node2Vec Algorithm*

input : A processed DNA corpus: C, The number of sampling n, Walk length l, Return parameter p, In-out parameter q, Context size k, Dimension d

output: K-mer vector representations Φ

1 $W \leftarrow \emptyset$ `// Adjacency Matrix ;`
2 $PW \leftarrow \emptyset$ `// Weight on nodes ;`
3 **for** $i = 1$ **to** $Size(C)$ **do**
4 **for** $k\text{-}mer$ **to** C_i **do**
5   $W_{k\text{-}mer,k\text{-}mer+1} \leftarrow W_{k\text{-}mer,k\text{-}mer+1} + 1$
  $PW_{k\text{-}mer} \leftarrow PW_{k\text{-}mer} + 1$
6 $\pi \leftarrow$ PreprocessModifiedWeights (W, p, q) based on Eq 5 ;
7 $G' = (V, E, \pi, PW)$;
8 $S_{sampled} \leftarrow \emptyset$ `//` $S_{sampled}$ `is the result set of random walk sampling.`
9 **for** *all nodes* $u \in V$ **do**
10 **for** $iter = 1$ **to** $n\frac{PW_u}{\sum_k PW_k}$ **do**
11   $k\text{-}mer_sequence \leftarrow$ SamplingProcess (G', u, l);
12   Append $k\text{-}mer_sequence$ to $S_{sampled}$
13 c Φ = Skip-Gram$(k, d, S_{sampled})$;
14 **return** Φ;
15 **function** SamplingProcess(Graph $G' = (V, E, \pi, PW)$, Start node u, Length l);
16 Initialize $k\text{-}mer_sequence$ to $[u]$;
17 **for** $walk_iter = 1$ **to** l **do**
18 $curr \leftarrow walk\,[-1]$;
19 $V_{curr} \leftarrow$ GetNeighbors$(curr, G')$;
20 $s \leftarrow$ AliasSample(V_{curr}, π);
21 Append s to $k\text{-}mer_sequence$;
22 **return** $k\text{-}mer_sequence$;

The algorithm first initializes variables (lines 1-2). We build the k-mer co-occurrence graph (Lines 3-7). Then, we use the Eq. 5 to calculate the transitive matrix π in line 6. Line 8-12 introduces using a random walk-based sampling strategy on the k-mer co-occurrence graph to generate the k-mer sequences $S_{sampled}$. In the end, in line 13, the k-mer embedding Φ is calculated by the Skip-Gram model based on $S_{sampled}$.

C. Time and Space Complexity Analysis.

The KMer-Node2Vec algorithm is composed of three steps: (i) scanning once all the sequences to construct the k-mer graph (Lines 1-2). The time complex is $O(|N|)$ time. After scanning, the k-mer graph is constructed (Lines 3-7), which takes $O(m|V|)$ memory, where m is the average degree of the graph; (ii) generating of n random walk traveling samples with walk length l (Lines 8-12) takes $O(nl)$ for both time and memory complexity; (iii) we run Skip-Gram (Line 13) on these sampling k-mer sequence corpus. The sampling of corpus whose size is $n \times l$, requires $O(nllog(|V|))$ for time and $O(nl)$ for memory. The space complexity of storing the final k-mer embeddings is $O(d|V|)$. In summary, the KMer-Node2Vec's time complex is $O(|N| + nl + nllog(|V|))$ and space complexity is $O(m|V|+nl+d|V|)$. It is clear that KMer-Node2Vec needs to scan all the DNA/RNA sequences once to construct the k-mer co-occurrence graph.

D. DNA Sequence Embedding

A DNA sequence is viewed as a sentence with its fragmented k-mers as words so that we transform the problem of DNA sequence embedding into a problem of sentence embedding. The sequence vector is the average vector of these k-mer vectors together (see Fig. 3).

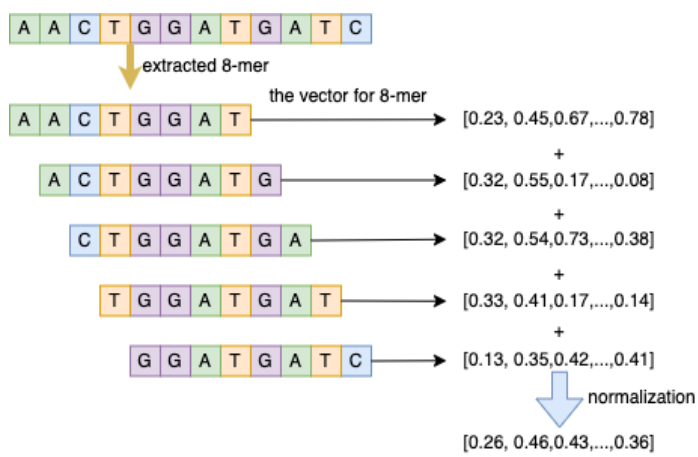


Fig. 3. An example of DNA sequence embedding

IV. Experimental Results

A. Experimental Settings

We use different data sets to test our approaches (see Table I).

Virus data set: Because of the next-generation sequencing error, any characters other than A, T, C, and G are removed from the sequences. After pre-processing, the virus data set contains 13,274 virus sequences, and the average length of

TABLE I
THE STATISTIC OF DATA SETS

data set	Total size	Total sequence number	Average sequence length
Fungi	142MB	2,007	7,1670bp
Virus	380MB	13,274	29,563bp
Bacteria	706MB	28,115	25,858bp
1G data set	1GB	43,145	26,644 bp
2G data set	2GB	65,614	28,947 bp
3G data set	3GB	95,991	30,599 bp
4G data set	4GB	129,999	34,247 bp

the virus sequence is 29,536bp. The size of the virus data set is around 380M.

Fungi data set: With the same pre-processing, the Fungi data set contains 2,007 fungi sequences, and the average length of fungi sequences is around 71,670 bp. The total size of the fungi data set is around 148M.

Bacteria data set: After pre-processing, the bacteria data set contains 28,115 bacteria sequences, and the average length of bacteria sequences is around 25,858 bp. The total size of the bacteria data set is about 700M.

iG data set: To test the efficiency and scalability of KMer-Node2Vec, we also process the different size of gene sequences from the gene bank. In Table. I, 1G data set means the data size is about 1GB.

All our experiments are conducted on a PC with a 3.80GHz Intel Core I7 Processor 8 Cores, 57GB memory, and running Win10, and all algorithms are implemented in Python and C++. Meanwhile, the Pecanpy algorithm [37], which uses cache optimized compact graph data structure and parallelization to improve the performance of Node2vec, is applied in our experiment. We choose 8-mers for running KMer-Node2Vec and DNA2Vec. Because the 8-mer graph we constructed is relatively sparse (the density is no more than 0.01% ($\frac{65536\times4}{65536\times(65536-1)/2}$)), according to Pecanpy's distinction of operation modes of networks with different densities, $SparseOTF$ mode is selected, which is suitable for networks that are large and sparse.

B. Accuracy Evaluation Criteria

We adopted the sequence retrieval task to evaluate the accuracy and tested them with the baseline approach, BLAST+ [6].

The DNA sequence retrieval system using KMer-Node2Vec are described here: (i) the raw DNA sequences are split into non-overlapping segments with 150bp. (ii) the 150bp segments are transformed to vectors and loaded into the Faiss [38] system. In our experiment, HNSW [39] index are used as the vector index in the Faiss system. (iii) Top K vectors of sequences can search in the Faiss system to find K vectors of DNA sequences that are highly similar to the query. To make it easy to record, we call this alignment-free vector search system the Faiss approach in our experiment.

All the hyperparameters in the experiments are adjusted to achieve the best performance. The hyperparameters of Faiss and BLAST+ are shown in Table. II.

TABLE II
PARAMETERS OF FAISS AND BLAST+

Method	Parameters		Value
Faiss	index		HNSW
	d		128
	vertex_connection		100
	ef_search		2000
	ef_construction		128
BLAST+	Create database	dbtype	nucl
		hash_index	Null
		parse_seqids	Null
	Retrieve sequence	max_target_seqs	20
		dust	no
		outfmt	6 qseqid sseqid

[1] The official documentation of Faiss parameters is shown in https://github.com/facebookresearch/faiss/wiki.
[2] The official documentation of BLAST+ parameters is shown in https://www.ncbi.nlm.nih.gov/books/NBK279684/.

The retrieval task: We randomly selected 1,000 DNA sequences with random sizes from 15bp to 75bp as the query sequence and this species label corresponding to this sequence is the ground-truth. The query sequence is inputted into the system and Top-K similar DNA sequences, denoted as S_K, are returned. Note that K means the system returns not just the most similar sequence, but also 2nd most similar, ..., K-th most similar sequence to the query sequence.

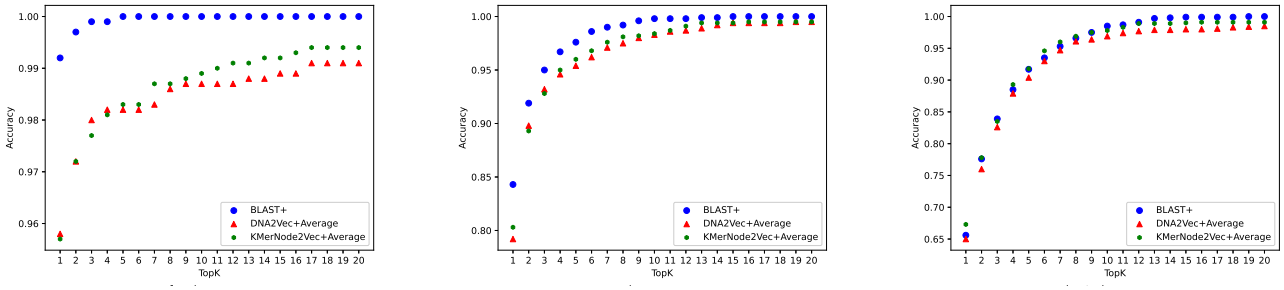
So, Top-K accuracy takes the K model predictions with higher probability. If one of them is a true label (correct space for this DNA sequence), it classifies the prediction as correct. The equation of Top-k accuracy is defined as follows:

$$accuracy(k) = \frac{\sum_{i=1}^{u} |s_i \in Top(i,k)|}{u} \tag{7}$$

where k denotes the number of query's nearest neighbors to be found, u represents the number of queries, and s_i is the exact space label as the ground-truth of the query.

C. Model evaluation

1) Efficiency of KMer-Node2Vec algorithm: Fig. 5 and Fig. 6 summarize the runtimes and memory usage for DNA2Vec [1] and KMer-Node2Vec on three data sets described in Table I. The experimental results show that: (i) KMer-Node2Vec is much faster than DNA2Vec in terms of training speed on all data sets, with the former **27 times faster** than the latter, on average. (ii) KMer-Node2Vec's runtime increases slower than that of DNA2Vec with training data growing, which means the runtime gap between KMer-Node2Vec and DNA2Vec will increase if data becomes larger. For example, because of the stable size of the k-mer graph, the runtime of KMer-Node2Vec increases slightly from **80min** to **230min**, while DNA2Vec's runtime rises from **1780min** to **6600min** on the same data sets. (iii) Compared with DNA2Vec, KMer-Node2Vec consumes more memory usage. This is because KMer-Node2Vec employs a one-shot k-mer sampling strategy, which generates all sampled sequences at once for training while DNA2Vec adopts the art of generator technique, which produces samples one at a time and only



(a) Fungi query accuracy (b) Virus query accuracy (c) Bacteria query accuracy

Fig. 4. (a), (b), (c) are the sequence retrieval accuracy comparisons on three species data sets in Table I using different portfolios. Note that "XXX2Vec+Average" means that the sequence vectors involved in the comparisons are computed within XXX2Vec's pre-trained k-mer vectors (see Section III-D)

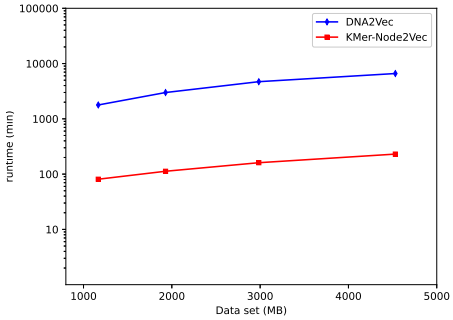


Fig. 5. Runtimes for KMer-Node2Vec and DNA2Vec on four data sets (1G-4G). Notice that the ordinate scale is arranged in logarithmic base 10.

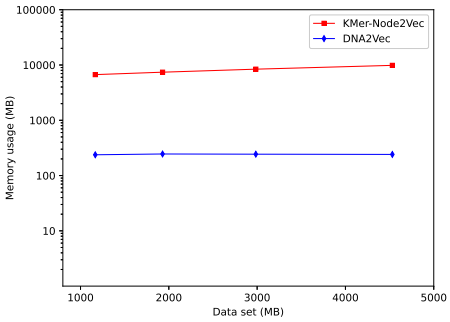


Fig. 6. Memory usage for KMer-Node2Vec and DNA2Vec on four data sets (1G-4G). Notice that the ordinate scale is arranged in logarithmic base 10.

when required. Fortunately, the memory usage of KMer-Node2Vec intend to remain unchanged once reaching the memory usage peek (see Fig. 6), since the size of the k-mer graph and sampled sequences can be viewed as constant.

2) Sequence retrieval using Faiss and BLAST+: We compared the sequence search performance between our proposed k-mer vector search in Faiss and BLAST+ using three species data sets described in Table I. Unless stated-otherwise, we set the number of query's nearest neighbors to be found as 20 and the number of queries as 1,000. Fig. 4 and Fig. 7 show the

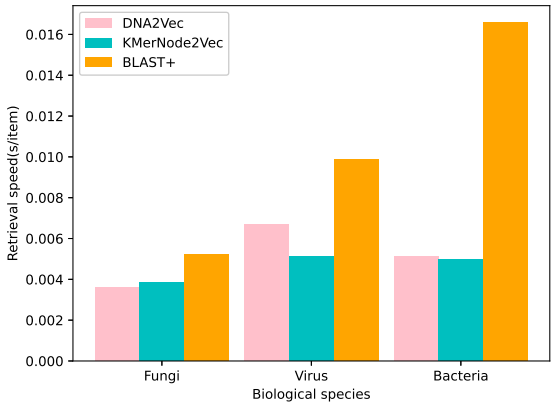


Fig. 7. Sequence retrieval speed comparisons between BLAST+ and k-mer vector search in Faiss. Here we used DNA2Vec and KMer-Node2Vec to produce k-mer vectors for sequence vectors, so that Faiss can make sequence retrieval within such sequence vectors.

runtimes and accuracy on these three data sets, respectively.

Overall, we found that BLAST+ sightly outperforms Faiss in terms of accuracy, but Faiss significantly took a much shorter search time than BLAST+. In terms of search time, Faiss is about **1.4 times** and **3.3 times** faster than BLAST+ using the minimum data set (Fungi) and the maximum one (Bacteria), respectively. In fact, as data set increases, the retrieval time gap between Faiss and BLAST+ will become larger and larger, so Faiss is a good alternative to BLAST+ on large-scale data sets. Meanwhile, the Top-K accuracy of Faiss is very similar to that of BLAST+ in most cases (see Fig. 4), which verifies the effectiveness of the Faiss approach and KMer-Node2Vec's k-mer embedding. Since Faiss typically achieves high search accuracy of **99.5%** at the Top-20 level, it is adequate to meet the requirement of real application.

V. Conclusion

We propose, the KMer-Node2Vec algorithm, to improve the performance of k-mer embedding, which converts the

large corpus into a k-mer co-occurrence graph, then takes the k-mer sequence samples from this graph by randomly traveling and trains the k-mer embedding on these sampling corpora in the end. We argue that due to the stable size of k-mer graph, the size and density of the k-mer co-occurrence graph do not change with the increase of training corpus. Thus, KMer-Node2Vec has a stable runtime on the large-scale data set, and its performance advantage becomes more and more obvious with the growth of the training corpus. Experimental results show that the proposed algorithm outperforms traditional DNA2Vec by 29 times in terms of efficiency with 99.5% retrieval accuracy on 4G data set.

References

[1] P. Ng, "dna2vec: Consistent vector representations of variable-length k-mers," *arXiv preprint arXiv:1701.06279*, 2017.

[2] H. J. Kim, S. E. Hong, and K. J. Cha, "seq2vec: analyzing sequential data using multi-rank embedding vectors," *Electronic Commerce Research and Applications*, vol. 43, p. 101003, 2020.

[3] C.-M. Leung, D. Li, Y. Xin, W.-C. Law, Y. Zhang, H.-F. Ting, R. Luo, and T.-W. Lam, "Megapath: sensitive and rapid pathogen detection using metagenomic ngs data," *BMC genomics*, vol. 21, no. 6, pp. 1–9, 2020.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

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

[6] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. L. Madden, "Blast+: architecture and applications," *BMC bioinformatics*, vol. 10, no. 1, pp. 1–9, 2009.

[7] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez *et al.*, "Clustal w and clustal x version 2.0," *bioinformatics*, vol. 23, no. 21, pp. 2947–2948, 2007.

[8] H. Wang, Z. Xu, L. Gao, and B. Hao, "A fungal phylogeny based on 82 complete genomes using the composition vector method," *BMC evolutionary biology*, vol. 9, no. 1, pp. 1–13, 2009.

[9] S.-R. Jun, G. E. Sims, G. A. Wu, and S.-H. Kim, "Whole-proteome phylogeny of prokaryotes by feature frequency profiles: An alignment-free method with optimal feature resolution," *Proceedings of the National Academy of Sciences*, vol. 107, no. 1, pp. 133–138, 2010.

[10] G. E. Sims and S.-H. Kim, "Whole-genome phylogeny of escherichia coli/shigella group by feature frequency profiles (ffps)," *Proceedings of the National Academy of Sciences*, vol. 108, no. 20, pp. 8329–8334, 2011.

[11] B. E. Blaisdell, "A measure of the similarity of sets of sequences not requiring sequence alignment." *Proceedings of the National Academy of Sciences*, vol. 83, no. 14, pp. 5155–5159, 1986.

[12] L. Wan, G. Reinert, F. Sun, and M. S. Waterman, "Alignment-free sequence comparison (ii): theoretical power of comparison statistics," *Journal of Computational Biology*, vol. 17, no. 11, pp. 1467–1490, 2010.

[13] G. E. Sims, S.-R. Jun, G. A. Wu, and S.-H. Kim, "Alignment-free genome comparison with feature frequency profiles (ffp) and optimal resolutions," *Proceedings of the National Academy of Sciences*, vol. 106, no. 8, pp. 2677–2682, 2009.

[14] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.

[15] E. Asgari and M. R. Mofrad, "Continuous distributed representation of biological sequences for deep proteomics and genomics," *PloS one*, vol. 10, no. 11, p. e0141287, 2015.

[16] D. Kimothi, A. Soni, P. Biyani, and J. M. Hogan, "Distributed representations for biological sequence analysis," *arXiv preprint arXiv:1608.05949*, 2016.

[17] X. Min, W. Zeng, N. Chen, T. Chen, and R. Jiang, "Chromatin accessibility prediction via convolutional long short-term memory networks with k-mer embedding," *Bioinformatics*, vol. 33, no. 14, pp. i92–i101, 2017.

[18] M. Zuckerman and M. Last, "Using graphs for word embedding with enhanced semantic relations," in *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, 2019, pp. 32–41.

[19] P. Sen, D. Ganguly, and G. Jones, "Word-node2vec: improving word embedding with document-level non-local word co-occurrences," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 1041–1051.

[20] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks, "A closer look at skip-gram modelling." in *LREC*, vol. 6. Citeseer, 2006, pp. 1222–1225.

[21] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994.

[22] J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins, "The clustal_x windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools," *Nucleic acids research*, vol. 25, no. 24, pp. 4876–4882, 1997.

[23] F. Sievers and D. G. Higgins, "Clustal omega for making accurate alignments of many protein sequences," *Protein Science*, vol. 27, no. 1, pp. 135–145, 2018.

[24] W. R. Pearson, "Finding protein and nucleotide similarities with fasta," *Current protocols in bioinformatics*, vol. 53, no. 1, pp. 3–9, 2016.

[25] F. Madeira, Y. M. Park, J. Lee, N. Buso, T. Gur, N. Madhusoodanan, P. Basutkar, A. R. Tivey, S. C. Potter, R. D. Finn *et al.*, "The embl-ebi search and sequence analysis tools apis in 2019," *Nucleic acids research*, vol. 47, no. W1, pp. W636–W641, 2019.

[26] W. Chen, B. Liao, and W. Li, "Use of image texture analysis to find dna sequence similarities," *Journal of Theoretical Biology*, vol. 455, pp. 1–6, 2018.

[27] E. Delibaş and A. Arslan, "Dna sequence similarity analysis using image texture analysis based on first-order statistics," *Journal of Molecular Graphics and Modelling*, vol. 99, p. 107603, 2020.

[28] M. Curilem Saldías, F. Villarroel Sassarini, C. Muñoz Poblete, A. Vargas Vásquez, and I. Maureira Butler, "Image correlation method for dna sequence alignment," *PloS one*, vol. 7, no. 6, p. e39221, 2012.

[29] J. S. Almeida, J. A. Carrico, A. Maretzek, P. A. Noble, and M. Fletcher, "Analysis of genomic sequences by chaos game representation," *Bioinformatics*, vol. 17, no. 5, pp. 429–437, 2001.

[30] H. J. Jeffrey, "Chaos game representation of gene structure," *Nucleic acids research*, vol. 18, no. 8, pp. 2163–2170, 1990.

[31] J. K. Vries and X. Liu, "Subfamily specific conservation profiles for proteins based on n-gram patterns," *BMC bioinformatics*, vol. 9, no. 1, pp. 1–13, 2008.

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[33] R. Blanco and C. Lioma, "Graph-based term weighting for information retrieval," *Information retrieval*, vol. 15, no. 1, pp. 54–92, 2012.

[34] S. Hassan, R. Mihalcea, and C. Banea, "Random walk term weighting for improved text classification," *International Journal of Semantic Computing*, vol. 1, no. 04, pp. 421–439, 2007.

[35] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[36] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola, "Reducing the sampling complexity of topic models," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 891–900.

[37] R. Liu and A. Krishnan, "Pecanpy: a fast, efficient and parallelized python implementation of node2vec," *Bioinformatics*, vol. 37, no. 19, pp. 3377–3379, 2021.

[38] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[39] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.