Learning Genomic Sequence Representations using Graph Neural Networks over De Bruijn Graphs

Abstract

The rapid increase of genomic sequence data requires new methods for creating robust sequence representations. Existing techniques often neglect detailed structural information, focusing mainly on contextual information. We addressed this issue by developing k-mer embeddings that combine contextual and structural string information, by enriching De Bruijn graphs with structural similarity connections. We also crafted a self-supervised method using Contrastive Learning, employing a heterogeneous Graph Convolutional Network encoder and constructing positive pairs based on node similarities. Our embeddings consistently outperform prior methods for Edit Distance Approximation and Closest String Retrieval tasks.

1 Introduction

Genomic sequence data is growing at an unprecedented rate, requiring the development of novel methods that can provide both accurate and scalable sequence representations. These representations are essential for various computational biology tasks, including gene prediction and multiple sequence alignment. Methods from Natural Language Processing (NLP), such as Word2Vec and Transformers, have been adopted to improve the representation of genomic sequences. These NLP-based approaches are effective at capturing the context within a sequence, which is important because the semantics of words often outweigh their precise letters.

Character-level n-gram models might be used to capture structural nuances. However, a uniform representation of each n-gram across all sequences can oversimplify the problem. Applying techniques like transformer-based models on n-grams can escalate computational demands. Consequently, these methods may overlook nuanced k-mer variations important for understanding single-nucleotide polymorphisms and other minor sequence changes. These SNPs can influence disease susceptibility, phenotypic traits, and drug responses.

Therefore, we developed a k-mer embedding approach that combines metagenomic context and string structure. In our method, contextual information refers to the relationships between k-mers closely situated within sequences, and structural information examines nucleotide patterns within a k-mer and their relations to other k-mers. We constructed a metagenomic graph that builds upon the De Bruijn Graph to capture k-mer transitions and structural similarities.

Given the advances in Graph Neural Networks (GNNs), we grounded our method in GNNs but designed for heterogeneous graphs. This approach effectively recognizes and uses both contextual and structural connection types. Drawing from the success of self-supervised pre-training in NLP and Computer Vision, we designed a self-supervised objective for genomic graph data. We employed contrastive loss aiming to align k-mers with similar context and structure in representation space.

Finally, we tested our technique on two downstream tasks: Edit Distance Approximation and Closest String Retrieval. The former estimates the minimum changes needed to transform one genomic sequence into another, avoiding quadratic computational complexity. The latter task, Closest String Retrieval, involves finding sequences similar to a query.

.

2 Related Work

2.1 Genomic Sequence Representation

Machine learning methods have emerged in computational biology to represent genomic sequences. A key component is the k-mer: a continuous nucleotide sequence of length k. The Word2Vec method, which represents words as vectors using their context, treats overlapping k-mers in genomic sequences as words in sentences. Building on this, kmer2vec was introduced to apply Word2Vec to genomic data for Multiple Sequence Alignment. Another strategy is to use the De Bruijn graph, where k-mers are nodes and their overlaps are edges, in conjunction with Node2Vec, which derives node features from the contextual information of biased random walks. This method underpins GRaDL for early animal genome disease detection. K-mers also pair well with transformer-based models: DNABERT leverages a BERT-inspired objective and k-mer tokenization to predict genome-wide regulatory elements. Metagenome2Vec blends Node2Vec with transformers to analyze metagenomes with limited labeled data. Given the high computational demands of these transformer-based approaches, they are outside the scope of our benchmarks in this study.

2.2 Graph Neural Networks

Graph Convolutional Networks (GCNs) are foundational to several innovations in graph-based machine learning. In genomics, GNNs have been applied in metagenomic binning. Because we aim to enhance our node embeddings with structural similarity, both heterogeneity and heterophily are key considerations. Recognizing the ubiquity of heterogeneity in real-world graphs, Relational GCNs (R-GCNs) were developed. These networks expand upon GCNs by generalizing the convolution operation to handle different edge types. To tackle heterophily, where distant nodes in a graph may bear similar features, Geom-GCN maps nodes to a latent space, while another approach suggests a distinct encoding approach for node embeddings and neighborhood aggregations.

2.3 Self-Supervised Learning

Self-supervised learning (SSL) enables effective use of unlabeled data and reduces dependence on annotated labels. Among SSL methods, contrastive learning has made a significant impact. At its core, contrastive learning seeks to bring similar data instances closer in the embedding space while pushing dissimilar ones apart. When applied to graph data, several techniques have been proposed for obtaining positive pairs, including uniform sampling, node dropping, and random walk sampling.

3 Methodology

3.1 Metagenomic Graph

The De Bruijn Graph, which is created from metagenomic sequences, forms the basis of our method. In this graph, each k-mer, a substring of length k from the sequences, is represented by a different node. An edge from node v_i to node v_j in the graph indicates that the k-mer at node v_i directly precedes the k-mer at node v_j in one of the sequences of the metagenome.

When used, edge weights represent the frequency of these transitions, capturing genomic structures within the graph.

Although Node2Vec captures the sequential context in De Bruijn graphs, it overlooks structural k-mer similarities. To address this, we expand the graph to include connections based on these similarities. We formulate two edge types for our graph, where nodes $v_i, v_j, ...$ represent k-mers.

De Bruijn Graph's edges The first edge type is designed to capture contextual information. Let $T(v_i, v_j)$ be the count of transitions between k-mers within a dataset of genomic sequences. The weight of an edge connecting nodes v_i and v_j, $w_{ij}^{(dBG)}$, is defined by,

$$w_{ij}^{(dBG)} = \frac{T(v_i,v_j)}{\sum_{v_k \in \delta^+(v_i)} T(v_i,v_k)}$$

where $\delta^+(v_i)$ denotes nodes adjacent to v_i via outgoing edges.

Sub-k-mer Frequency edges To capture the structural similarity between strings, we introduce a method using sub-k-mer frequency vectors, denoted as $y^{(KF_{sub\_k})}$. This vector quantifies the occurrences of each sub-k-mer of length sub_k within a given k-mer. The i-th entry indicates the frequency of the i-th sub-k-mer,

$$y^{(KF_{sub\_k})}[i] = \sum_{j=1}^{k-sub\_k+1} I[kmer[j : j + sub\_k - 1] = s_i] \forall s_i, s \in \sum^{sub\_k}$$

The k-mer similarity is determined by the cosine similarity between the sub-k-mer frequency vectors,

$$w_{ij}^{(KF_{sub\_k})} = \frac{y_i^{(KF_{sub\_k})^T} y_j^{(KF_{sub\_k})}}{||y_i^{(KF_{sub\_k})}||_2 ||y_j^{(KF_{sub\_k})}||_2}$$

This method, scaling linearly with the frequency vector size per weight, provides a computational advantage over the direct Edit Distance calculation for k-mers. We apply edge-filtering at threshold t, retaining only the links with the highest similarity. The filtered set of weights is then,

$$W^{(KF_{sub\_k})} = \{w_{ij}^{(KF_{sub\_k})} | w_{ij}^{(KF_{sub\_k})} \geq t\}$$

To accommodate graphs for larger k values, we have developed a more scalable approximation of the above approach. It utilizes approximate nearest neighbor search on the sub-k-mer frequency vectors, which replaces the computationally demanding pairwise cosine similarity calculations.

The metagenomic graph is defined as $G = (V, E, W)$. Nodes V correspond to individual k-mers. The edges E can be categorized into two sets: De Bruijn Graphs's edges $E^{(dBG)}$ and Sub-k-mer Frequency edges $E^{(KF)}$. Edges in $E^{(KF)}$ may be further subdivided based on various sub_k values. Edge weights W can contain $W^{(dBG)}$ and several $W^{(KF_{sub\_k})}$.

3.2 Encoder

We tailored GNNs for a heterogeneous metagenomic graph to capture nuanced k-mer relationships. The design employs varying depths of message passing: deeper for De Bruijn edges to capture broader context and shallower for similarity measures. Central to this GNN is the adapted Graph Convolutional Layer, formulated as:

$$H^{(l+1)} = \sigma(\tilde{D}^{(edge\_type)^{-\frac{1}{2}}} \tilde{W}^{(edge\_type)} \tilde{D}^{(edge\_type)^{-\frac{1}{2}}} H^{(l)} \Theta^{(l)})$$

where $\tilde{W}^{(edge\_type)}$ includes added self-loops and $\tilde{D}_{ii}$ is its diagonal degree matrix. The term edge_type refers to either dBG or KF_{sub_k}. The GCN layout consists of multiple layers, each characterized by a unique edge feature type and the number of channels.

3.3 Self-Supervised Task

We use a contrastive learning method for k-mer representations. Graph nodes are initialized using a sub-k-mer frequency vector. Positive and negative pairs are sampled and, along with the k-mer representations from the encoder, are used to compute the loss.

Biased Random Walk Sampling We employ Biased Random Walk Sampling to capture k-mer contextual information. This approach uses $w^{(dBG)}$ edges to conduct walks, implemented exactly as in Node2Vec. Given a walk of a set length, we extract positive pairs by applying a window of size m. Using a shrink factor δ, drawn uniformly from $1, ..., m$, we determine the range $i \pm \delta$ within which nodes are considered positive pairs to node v_i. Repeating this across multiple random walks, we gather a comprehensive set of positive pairs.

Structural Similarity Sampling To capture the structural notion of k-mers, we sample pairs with probability proportional to sub-k-mer frequency similarity, $w^{(KF_{sub\_k})}$. The goal is for k-mers linked by higher similarity to have similar representations. The probability of sampling is given by,

$$P(v_i, v_j) \propto w_{ij}^{(KF_{sub\_k})}$$

Negative Sampling We randomly select negative pairs from all node pairs in the graph, leveraging the assumption that most pairs lack a high similarity edge. This approach ensures diversity in learned representations.

Loss Function Having established both positive (P_{pos}) and negative (P_{neg}) pair types, we apply the contrastive loss function. Using $\sigma(x)$ as the sigmoid function, the loss function is:

$l_{ij} = -log(\sigma(z_i^T z_j)) - \sum_{(v_i,v_l)\in P_{neg}} log(1 - \sigma(z_i^T z_l))$

To reduce memory usage, we employed Neighborhood Sampling for mini-batching during training.

4 Bioinformatics Tasks

4.1 Edit Distance Approximation

The task is to calculate the edit distance without quadratic complexity. The NeuroSEED framework offers a solution using sequence representations trained on a ground truth set of edit distances. In our approach, we began with sequence representations derived from k-mer embeddings and fine-tuned them with a single linear layer. Our experiments were tested against One-Hot encoding (for k = 1), Word2Vec, and Node2Vec. To find optimal hyperparameters, we executed a grid search on the validation set. Based on previous work, we used the hyperbolic function. Our primary metric for evaluation was the percentage Root Mean Squared Error (percent RMSE), where l denotes the dataset's maximum sequence length, h represents the hyperbolic distance function, and f_θ indicates the downstream model,

$\%RMSE(D) = \frac{100}{l}\sqrt{\sum_{s_1,s_2\in D}(EditDistance(s_1,s_2) - h(f_\theta(s_1), f_\theta(s_2)))^2}$

4.2 Closest String Retrieval

The task is to find the sequence from a reference set that is closest to a query. We assessed embeddings fine-tuned on the edit distance approximation task using Convolutional Neural Networks (CNNs). These embeddings were contrasted with ones directly derived from our Self-supervised method, One-Hot, Word2Vec, or Node2Vec, through concatenation or taking the mean of k-mer embeddings. For performance assessment, we used top-n percent accuracies, measuring how often the actual sequence appears within the top n percent of positions based on the closeness of embedding vectors in hyperbolic space. We selected the optimal model for the embeddings based on the validation loss observed for the previous Edit Distance task.

5 Results and Analysis

In all our experiments, the memory requirements of the One-Hot method increased exponentially, leading to its exclusion from our results for k > 7. When pre-training exclusively on the training set, our method, thanks to the GCN encoder, can generalize beyond k-mers present in the training set. In contrast, Node2Vec and Word2Vec can only handle k-mer sizes up to the diversity of the training dataset. Therefore, for k > 6, where the test set introduces new k-mers, we excluded these methods.

5.1 Edit Distance Approximation

Table 1 presents the results obtained by using our pre-trained embeddings to estimate edit distances between sequences on the RT988 and Qiita datasets. For the RT988 dataset, our Contrastive Learning (CL) and Node2Vec techniques surpassed Word2Vec and One-Hot. The increased losses in Qiita highlight its greater complexity. In this context, our method's integration of k-mer structural similarity becomes even more beneficial, outperforming all other tested methods. This benefit becomes more evident as k increases, underscoring our embedding's capability to adapt to new nodes.

5.2 Closest String Retrieval

Tables 2a and 2b present the performance of our zero-shot sequence embeddings, directly derived from the aggregation of our k-mer embeddings, in retrieving the nearest sequences in the Qiita dataset. The tables also showcase a comparison with the embeddings that were specifically fine-tuned for the Edit Distance Task.

For direct k-mer aggregation, our Contrastive Learning (CL) embeddings are obtained through concatenation, while for k-mer aggregation with One-Hot, Word2Vec, and Node2Vec, we report the results of the better performing method, either concatenation or averaging. The superior zero-shot non-parametric retrieval performance of our CL method emphasizes the combined utility of both context and structural similarity during self-supervised pre-training. Notably, while k-mers of size around three are optimal for Top 1 percent retrieval, larger k-mers excel in the Top 10 percent metrics. This suggests that smaller k-mers are better at discerning local sequence distances, while larger ones capture broader sequence distances.

For embeddings fine-tuned using CNNs for Edit Distance Approximation, the complexity of CNNs obscures differences between the embeddings. Our method based solely on zero-shot concatenated k-mer embeddings outperforms this complex fine-tuning. This shows the advantage of our embeddings over the method by previous work.

6 Conclusion

In our study, we introduced a novel k-mer embedding technique that seamlessly integrates metagenomic contextual and structural nuances, achieved through the enhancement of the De Bruijn graph and the use of contrastive learning. In the Edit Distance Approximation task, our technique consistently demonstrated superior performance compared to One-Hot, Word2Vec, and Node2Vec. Moreover, without requiring any downstream fine-tuning, our aggregated k-mer embeddings outperformed the prior method in the Closest String Retrieval task. These findings suggest potential broader uses in computational biology.