Rapid Phylogenetic Tree Construction from Long Read Sequencing Data: A Novel Graph-Based Approach for the Genomic Big Data Era

Harisankar Sadasivan, Luke Ross, Chih-Yu Chang, Kushantha Upulanga Attanayake

Computer Science and Engineering, University of Michigan Ann Arbor, MI 48109, USA
Email: hariss@umich.edu (H.S), lukeross@umich.edu (L.R), chiyu@umich.edu (C.C), kushan@umich.edu (K.U.A)

Abstract

Genomics is the largest producer of big data, with an expected 40 EB of data every year. The rapid growth of genomic data necessitates efficient methods for analysis and classification. We present a novel, automated pipeline for swift phylogenetic tree construction from long-read sequencing data. Our approach addresses computational challenges by utilizing compact repeat graphs instead of full genome assemblies. We integrate advanced graph embedding techniques, combining structural and content-based approaches, to capture genomic relationships efficiently. Demonstrating our method on 20 bacterial genomes across 5 classes, we achieve a cophenetic correlation of 0.53 with the ground truth phylogenetic tree. Our pipeline reconstructs meaningful evolutionary relationships directly from sequencing reads without requiring complete assemblies or time-consuming alignments. This work represents a significant advancement towards rapid pathogen classification during outbreaks and offers a scalable solution for analyzing the expanding universe of sequenced organisms. By bridging graph theory, machine learning, and genomics, our method paves the way for more efficient phylogenetic analysis in the era of big data biology.

Introduction

The large volume of genetic collected over time is not useful until it is analyzed and classified properly. Phylogenetics is the study of finding relationships among species or genes with the combination of molecular biology and mathematics. A phylogenetic tree or an evolutionary tree, as shown in Figure 1, is a graph that shows the evolutionary relationships among various biological species based on their evolutionary history and shared ancestors.

Conventionally, phylogenetic trees are constructed using expert crafted features including specific marker genes [15,16,17]. Typical work in automating the construction of this tree obtains similarities between raw genomes, requiring one to perform pairwise alignments among genetic sequences. A huge amount of work has been dedicated to speeding up the process and improving the quality of multiple sequence alignment on sequences within a given dataset [4,5,18].

However, so far, the most well-known algorithms that construct high-quality phylogenetic trees requires sequences of species which have been assembled from sampled reads beforehand. In addition, the use of dynamic programming in solving sequence alignment problem with time complexity of $O(mn)$, where$m$ and n are lengths of two genomes, makes the entire process fairly time consuming. Genome assembly is the process of taking a large number of DNA fragments, known as reads, from DNA sequencer and putting them back together to create a representation of the original chromosomes from which the DNA originated. Nowadays, several popular assembly algorithms [9,19,20,22] have been developed to assemble short, 100-200 base-pair long, or long, 10,000-100,000 base-pair long, reads. These techniques incorporate graph-based algorithms by first building assembly graphs, as shown in Figure 2, with their edges representing k-mers or overlapping sequences and convert the entire assembly into a graph traversal problem.

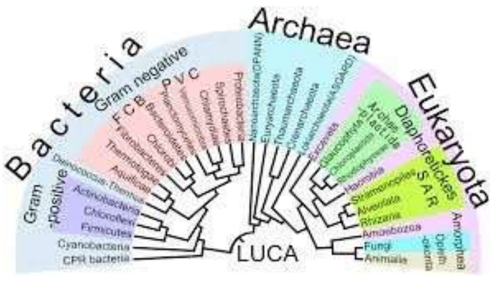

Figure 1: An example of phylogenetic tree

It is understood that these generated assembly graphs encode sufficient information for one to reconstruct genomes. We propose a method to use this type of graph as a more informative representation of a single genome. Therefore, it becomes reasonable that we can obtain similarities between genomes via comparing graph similarity. Moreover, it is possible that we build phylogenetic trees using a reasonable amount of time compared to conventional approaches given only sequencing reads, while exploiting the rich, untapped graphical structure of genomes.

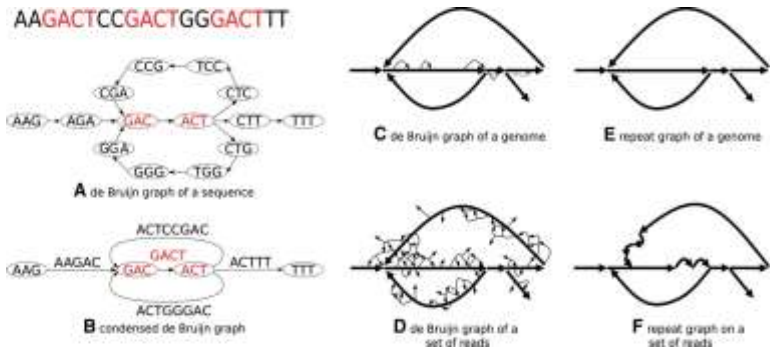
AAGACTCCGACTGGGACTTT
A de Bruijn graph of a sequence
B condensed de Bruijn graph
C de Bruijn graph of a genome
D de Bruijn graph of a set of reads
E repeat graph of a genome
F repeat graph on a set of reads
ACTCCGAC
AAGAC
GACT
ACTTT
ACTGGGAC

Figure 2: Various types of assembly graphs

In this work, we design and implement a complete pipeline to fully automate the process of phylogenetic trees construction. Given a large number of reads of a single genome, we generate and preprocess the desired assembly graph into a more condensed format with the help of powerful, open-sourced software, Flye Assembler [9]. We implement various types of graph embedding schemes, e.g. node2vec [6], struc2vec [14] to derive the representation of a graph that best encodes its structural and content information. Thereafter, similarity between two genomes may be obtained by comparing graph similarity using Pyramid Match Kernel in [13]. The final dendrogram is then built using hierarchical clustering.

The designed automated pipeline performs considerably well after evaluation, a correlation coefficient of 0.53 with the ground truth phylogenetic tree. In addition, from the final dendrogram, one can find out that bacteria belonging to the same category are classified to the same cluster most of the time.

Although we have identified several causes which may be limiting us from obtaining a higher correlation, our work demonstrates that biological relationships can be automatically established for any new species without needing complete whole genomes. We believe that this pipeline may be used to quickly characterize new pathogens via clustering during disease outbreaks.

In this paper, we organize the contents as follows: In section 2, we talk about the source of the data and how we generate the ground truth. Details of each phase of the pipeline are discussed in section 3. Experiments we conducted and evaluation methods are explained in section 4. We then describe prior works and the conclusion in sections 5 and 6, respectively.

Datasets Used

NCBI GenBank database [23] is home to whole genome assemblies of every bacterium known to the public. We identify 5 different classes of bacteria and pick 4 closely related bacteria from each class:

- *Alpha-proteobacteria*: E coli, Salmonella typhi, Pseudomonas aeruginosa and Salmonella enterica.
- *Firmicutes*: Enterococcus faecalis, Streptococcus pneumoniae, Listeria monocytogenes and Bacillus subtilis.
- *Actinobacteria*: Mycobacterium bovis, Mycobacterium tuberculosis, Streptomyces avermitilis and Mycobacterium paratuberculosis.
- *Beta-proteobacteria*: Neisseria meningitidis A, Chromobacterium violaceum, Bordetella pertussis and Nitrosomonas europaea.
- *Epsilon-proteobacteria*: Helicobacter pylori, Helicobacter hepaticus, Wolinella succinogenes and Campylobacter jejuni.

From NCBI GenBank database, the whole assembly for all the bacteria are downloaded. Artificial Oxford Nanopore technologies (ONT) MinION sequencer reads are randomly generated ensuring 50X coverage using BBMap, a bioinformatics tool.

Assembly graphs constructed from these datasets have node numbers in the order of their genome sizes. The genome assembly graphs contain anywhere from 4.3M to 14M nodes in each of them, resulting in roughly 180M nodes across all 20 assembly graphs in total and an average of 9M nodes per bacterial assembly graph.

We constructed the ground truth phylogenetic tree of the selected 20 bacteria following Hug et.al [7]. The ground truth tree in Figure 3 clearly displays the genetic relationships between bacteria. Here, we define the dendrogrammatic distance between a pair of bacteria as the number of hops between them in the ground truth evolutionary tree of life for the 20 bacteria under consideration. How we evaluate the performance of our work will be discussed in section 3 in detail.

PROPOSED METHOD

Figure 4 shows the complete pipeline of our work. In this section, we discuss our proposed method for automatically generating the phylogenetic tree from long read genomes in detail.

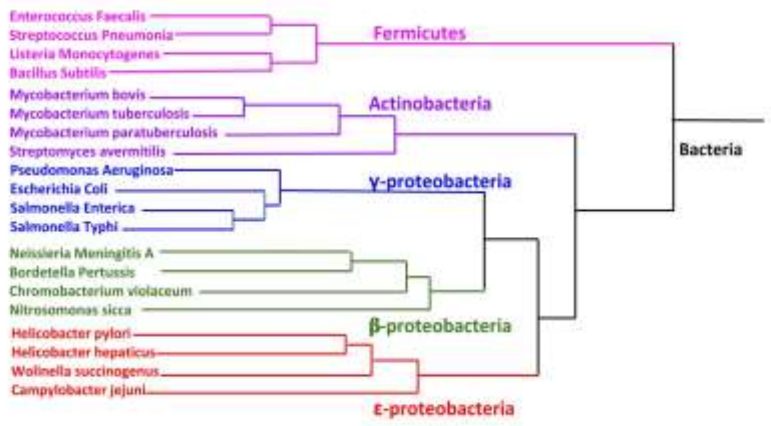

Figure 3: The ground truth phylogenetic tree

Graph compression: From Assembly to Repeat Graphs

Bacterial genomes under consideration range from 4.3M to 14M base-pairs long. Reads generated by the sequencers have 40-50X coverage which means that we have roughly 400M+ base-pairs of information per bacteria and there are 20 bacteria in total. In real life scenario, we have 330M+ species of bacteria and storing information can be a significant task, given that long-read assemblers are still in the growing stages. Hence, we try to innovate this step by making use of a compact representation of the assembly graph called repeat graph.

Repeat graphs are built using approximate sequence matches and can tolerate higher noise level of the input reads. Each edge is labeled with its index, length, coverage, orientation, number of repetition and the corresponding contigs. Using repeat graphs, several millions of nodes from the original assembly graph get converted into a much smaller repeat graph in which the edges contain sufficient information that could represent a complete sequence of a bacterium, as shown in Figure 5. In order to generate repeat graphs for genomes, we utilize an open-sourced long-read assembler called Flye [9] and tweak its parameters to obtain the desired repeat graphs instead of using the draft genome assembly.

As described above, each edge embeds rich information representing part of the genome while all nodes are simply junctions in the genome segments without any meaning, unlike common graphs where nodes represent meaningful entities and edges represent relationships. Furthermore, embedding schemes we use are mainly node-based. Therefore, we further convert the original repeat graphs into line graphs with steps depicted in Figure 6, in which each node corresponds to an edge of the original graph while structure of the original graph is still retained.

Graph Embeddings

The decision to use node embeddings was motivated primarily by the structure of the repeat graphs. For most bacteria, the generated repeat graphs would be composed of a relatively small number of nodes and edges, but each edge will have a large amount of metadata associated with it. This includes a long sequence of DNA base pairs, called a contig, the number of times this sequence was repeated in the original input sample, the number of input reads that contributed to this sequence etc. The length of these contigs would vary wildly between nodes, with some being composed of only several thousand base-pairs, while others being made up of millions of base pairs. We propose using embeddings to compare pairwise similarity between graphs while utilizing all this information.

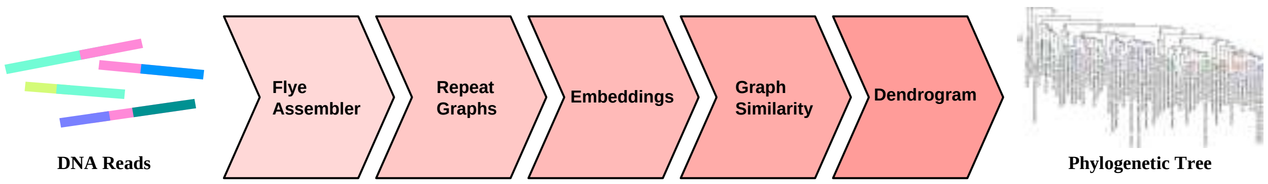

Figure 4: Pipeline for automatic phylogenetic tree generation

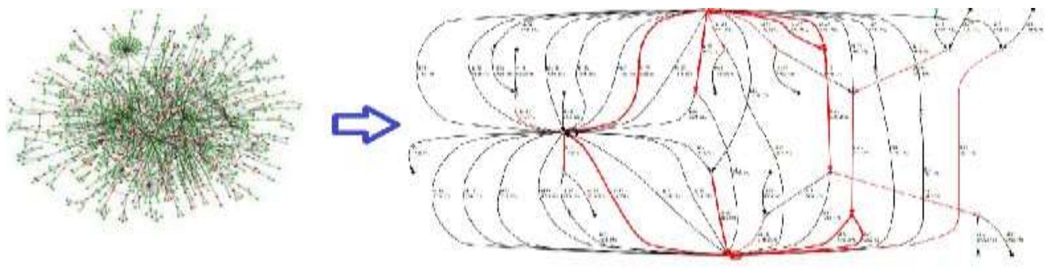

Figure 5: An example of compressing an assembly graph into a repeat graph

To get a representative set of embeddings, for each graph we generated a set of structural embeddings and a set of content embeddings, which were then concatenated together to generate one large set of embeddings.

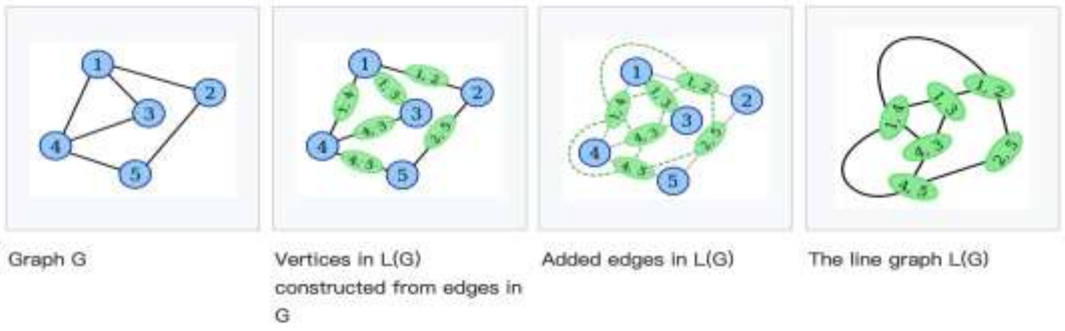

Figure 6: Illustration of a graph and its line graph

Structural Embeddings

To encode the structural information about the graphs, we tried use node2vec [6] and struc2vec [14] for our embeddings. For both methods, default values were used for all parameters except for the dimension of the embeddings.

Content Embeddings

For generating embeddings for the contigs, we used BioVec [1], which is an implementation of Word2Vec [10] for DNA sequences, where a contig is treated as a sentence, and small fixed length sequences of the contig is treated as words. Biovec is limited to dealing only with sequences of length 3, with any sequence longer than that being broken into multiple sequences of 3 and then embedded independently. This method is fast to both train and for generating embeddings.

The second method we opted to use was dna2vec [12], which is similar to Biovec, but can be generalized to sequences of any length, and supports variable length sequences. This method takes much longer to train to generate embeddings with and has a much higher memory footprint.

Both methods were trained using bacterial genomes as a corpus, and embeddings generated by both methods were scaled by each node's repeat number to accurately reflect the DNA structure of the original genome.

Graph Similarity

In order to generate a phylogenetic tree, we compute the similarity of each graph using the node embeddings generated from the repeat graphs. Here we employ the Pyramid Match Graph Kernel from [13] to measure similarity using node embeddings.

The Pyramid Match Graph Kernel operates on normalized embeddings in their latent Euclidean space and partitions this space into bins. By comparing the bins in which the embeddings fall for a pair of graphs, we can get a well-defined kernel similarity function. Varying the size of these bins and weighting accordingly allows the algorithm to more accurately capture the Euclidean similarity of the node embeddings. Finally, the algorithm can be easily extended to graphs with discrete node labels by only comparing nodes with the same labels and summing up similarity scores across label classes.

While this is an attractive extension of the Pyramid Match Kernel, our data fit the continuous, latent structure of graph embeddings much better than discrete labels, so our final pipeline does not discriminate based on discrete node labels. We carefully vectorize the implementation of the Pyramid Match Kernel since we require the distance between all pairs of graphs in order to cluster them. We use vectorized numpy functions in addition to a just in time compiler for python known as "numba" in order to optimize the performance of the similarity calculation

We opt not to use the Earth Mover's Distance similarity measure, also presented in [13]. The Earth Mover's Distance method has $O(n^3 \log(n))$ time complexity for a pair of graphs with n nodes, which is prohibitive even for reasonably sized graphs. In contrast, Pyramid Match Kernel has time complexity $O(dnL)$ for embedding dimension d, n nodes, and L different levels of bin sizing. This method scales linearly with the size of the graphs, which is much more manageable. Additionally, while the Pyramid Match Graph Kernel is truly a kernel method (that is, it defines a positive semidefinite kernel function which affords us some nice theoretical guarantees [13]), the Earth Mover's Distance method requires solving an additional complicated linear program in order to transform the similarity measure from an indefinite kernel into a true kernel method. Using a non-exhaustive heuristic search, we found that $L = 4$ gave the best results for our data.

Dendrogram Construction

After generating the distance between all pairs of graphs using the Pyramid Match Kernel as described above, we build the phylogenetic tree as a dendrogram using hierarchical clustering. Section 4.2 outlines in more detail our experimental results comparing across different types of linkages at this stage in the pipeline, but Ward's method was found to yield consistently better results than other linkage techniques.

Note that most hierarchical clustering algorithms have time complexity between $O(n^2)$ and $O(n^3)$ depending on the linkage criterion. In our case this was not prohibitive since we have selected a fairly small subset of bacterial genomes on which to create the dendrogram. Nonetheless, it does challenge the extension of our method to larger datasets. More efficient methods of performing hierarchical clustering is an area with a significant body of literature [3] [2] [11] and may be treated as a promising area for future extensions of our method.

Evaluation Method

The use of Ward linkage completes the pipeline for phylogenetic tree generation, but an additional step is needed for comparing our dendrogram to the ground truth tree. For this we used cophenetic correlation:

$$c = \frac{\Sigma_{i<j}(x(i,j)-\bar{x})(t(i,j)-\bar{t})}{\sqrt{(\Sigma_{i<j}(x(i,j)-\bar{x})^2)(\Sigma_{i<j}(t(i,j)-\bar{t})^2)}}$$

where $x(i, j)$ measures the dendrogramatic distance of elements i, j of the ground truth, $t(i, j)$ measures the dendrogramatic distance of elements i, j in our model, and $\bar{x}$, $\bar{t}$ are their averages, respectively. This correlation coefficient is simply a Pearson correlation of the dendrogrammatic distance in our model with the dendrogrammatic distance in the ground truth. This gives us a quantitative way to compare our results with the ground truth.

It is important to note that we are using dendrogrammatic distance in a different context unlike typical methods. Linkage defines the distance between clusters given all pairwise distances between original observations and yields a real-valued dendrogrammatic distance between any pair of observations in the dendrogram. The format of our ground truth data however restricts us to the discrete dendrogrammatic distance measure described at the end of Section 2. The ground truth tree given by NCBI specifies only the splits in the tree without the real-valued distance between them. To deal with this, we use the graph-hop counting strategy to calculate the dendrogrammatic distance in both the ground truth and our model.

Experiments

Evaluation and Results

Our final cophenetic correlation coefficient was 0.53 using embeddings from node2vec and dna2vec, Pyramid Match Kernel with $L = 4$, and ward linkage. The generated dendrogram is given in figure 7.

We hypothesize that the primary reason this correlation coefficient is not closer to 1 is the discretization of our dendrograms. Figure 3 and our ground truth tree do not encode any continuous notion of similarity, but nonetheless the content and structure of the genomes we use have similarity and distances which are much closer to continuous than discrete. Bacteria within the same class can be very closely related to one another while still being a handful of hops away, implying that important information is lost at this stage which makes our evaluation metric rougher that it might be in the continuous case. Nonetheless, given the complexity of our problem space, we are pleased to see reasonable correlation with the ground truth.

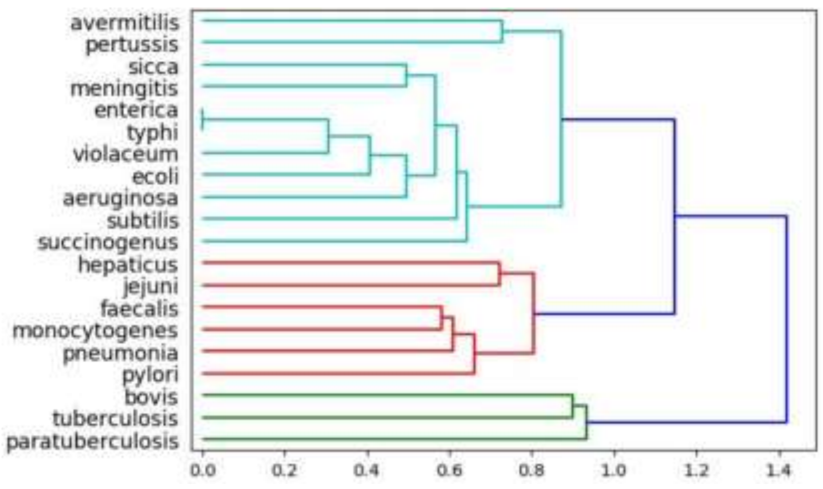

Figure 7: Generated phylogenetic tree as output from our pipeline.

Varying Linkage Criteria

In exploring the hyperparameters for our pipeline, one interchangeable piece we optimized over was the linkage criteria. Figure 8 summarizes our findings.

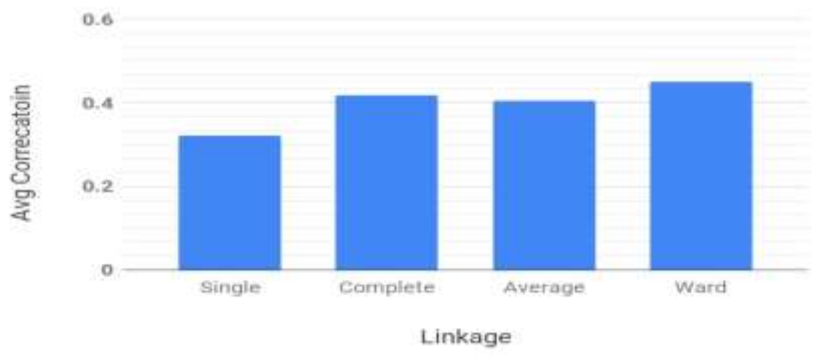

Figure 8: Correlation values for Linkage types, averaged over differing valuesfor embedding dimensions and *L* for Pyramid Match Kernel.

The single linkage criteria use the following formula to assign distance measures:

$$d(v,u) = min(dist(u[i], v[j]))$$

For all points I in cluster u and j in cluster v. Intuitively, this linkage technique leads to chains of points which are individually close but as a group can be far. Since the Pyramid Match Kernel looks at embeddings in a Euclidean space in all dimensions, it makes sense why the single method would perform the worst since it can result in spread out chains as clusters.

The complete linkage criteria uses the following formula to assign distance measures:

$$d(v,u) = max(dist(u[i], v[j]))$$

for all points i in cluster u *and* j in cluster v. This has the opposite effect as the single link method and aims to keep the most dissimilar elements as close as possible, effectively pushing the clusters towards higher dimensional spheres. This aligns well with our similarity metric, which is why complete linkage performs second best out of those tested.

The average linkage criteria uses the following formula to assign distance measures:

$$d(v,u) = \Sigma_{i,j} \frac{dist(u[i], v[j])}{|u||v|}$$

for all points i in cluster u *and* j in cluster v, where $|u|$ is the cardinality of cluster u and likewise for v. This method is subject to some of the same weaknesses as the single link method although less so since it takes all data points into account. For this reason, it is the second worst performing method of those tested.

The Ward linkage criteria use the following formula to assign distance measures:

$$d(v,u) = \sqrt{\frac{|v| + |s|}{T} d(v,s)^2 + \frac{|v| + |t|}{T} d(v,t)^2 - \frac{|v|}{T} d(s,t)^2}$$

where u is the newly joined cluster consisting of clusters s and t , v is an unused cluster in the forest, $T = |v| + |s| + |t|$, and $|*|$ is the cardinality of its argument. This method can intuitively be thought of as minimizing the variance within a given cluster. This aligns very well with our similarity metric since it takes all elements of a cluster into account while minimizing the ways in which they differ in the Euclidean space. This method is the most sophisticated and is the best performing of all the linkage criteria we tested

Varying Embedding Types

For embeddings, we experiment with various combinations of structural and content embeddings.

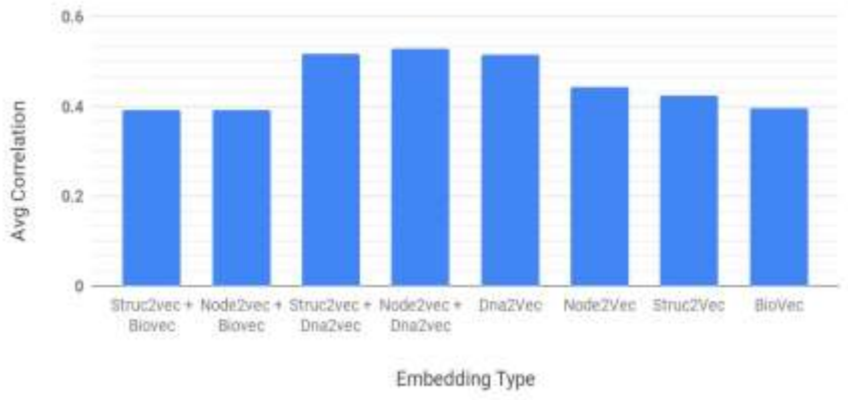

Figure 9: Correlation values for Embedding methods

Our results show that there is not much difference in correlation value between the two structural embedding methods, node2vec and struc2vec, but there is a noticeable difference between the two content embedding methods dna2vec and biovec. This is to be expected as most of the information regarding the original genome is encoded in the metadata for of each node, thus their embeddings will contribute the most towards calculating similarity. The final repeat graphs are also small and sparse, and as such the embeddings generated by the two methods might not be noticeably different from each other as they do not have much information to work with. Dna2Vec having better results than BioVec is also to be expected. Because Dna2Vec can handle sequences of greater length than BioVec, the embeddings generated by it might also have more information encoded in them due to the larger context. Dna2vec is also better suited to handling inputs of varying length, which would be a considerable advantage in this situation because, as mentioned earlier, then length of contigs could vary wildly even within a single graph.

Furthermore, our results also show that combining both structural and content embeddings produce better results than using either in isolation. This shows that considering the structure of the graph and the contigs of each node are important for accurately evaluating similarity between genomes.

Related Work

Comparing genome similarities for phylogenetic tree construction has been a bioinformatician's problem in the past few decades [4,5,8,18]. These tools often provide reasonably accurate results and overcome several limitations, and some of them are very popular and widely used nowadays.

However, all these previous works require complete sequences of species in the datasets. Our proposed pipeline incorporating various graph mining techniques mentioned in former sections, is the very first automated approach, to the best of our knowledge, that builds phylogenetic trees solely from sampling reads.

Conclusion

Our innovative pipeline demonstrates significant potential for revolutionizing phylogenetic tree construction in the era of big data genomics. By leveraging compact repeat graphs, advanced graph embeddings, and efficient similarity measures, we've shown it's possible to rapidly generate meaningful evolutionary relationships directly from sequencing reads. Our method's ability to achieve a 0.53 cophenetic correlation with ground truth, without full genome assembly or alignment, marks a substantial advancement in the field. This approach not only accelerates pathogen identification during outbreaks but also offers a scalable solution for analyzing the ever-expanding universe of sequenced organisms. As genomic data continues to grow exponentially, our work paves the way for more efficient, accurate, and timely phylogenetic analysis.

References

[1] Ehsaneddin Asgari, Mohammad R. K. Mofrad, and Firas H. Kobeissy. 2015. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. In *PloS one*.

[2] Ziv Bar-Joseph, David K. Gifford, and Tommi S. Jaakkola. 2001. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics* 17, suppl$_1$(062001),$\mathcal{S}$22 – –$\mathcal{S}$29.

[3] Pedro Contreras and Fionn Murtagh. 2011. Fast, Linear Time Hierarchical Clustering using the Baire Metric. Journal of Classification, July 2012, Volume 29, Issue 2, pp
118-143. (2011). https://doi.org/10.1007/s00357-012-9106-3 arXiv:arXiv:1106.2229

[4] Florence Corpet. 1988. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research* 16, 22 (1988), 10881–10890.

[5] Robert C. Edgar. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* 32, 5 (2004), 1792–1797.

[6] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *KDD : proceedings. International Conference on Knowledge Discovery Data Mining* 2016 (2016), 855–864.

[7] Laura A Hug, Brett J Baker, Karthik Anantharaman, Christopher T Brown, Alexander J Probst, Cindy J Castelle, Cristina N Butterfield, Alex W Hernsdorf, Yuki Amano, Kotaro Ise, et al. 2016. A new view of the tree of life. *Nature microbiology* 1, 5 (2016), 16048.

[8] Kazutaka Katoh and Daron M. Standley. 2013. MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. *Molecular Biology and Evolution* 30, 4 (2013), 772–780.

[9] Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel Pevzner. 2018. Assembly of Long Error-Prone Reads Using Repeat Graphs. *bioRxiv* (2018).

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

[11] Daniel Müllner et al. [n. d.]. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. ([n. d.]).

[12] Patrick Ng. 2017. dna2vec: Consistent vector representations of variable-length k-mers. *CoRR* abs/1701.06279 (2017).

[13] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Matching Node Embeddings for Graph Similarity. (2017).

[14] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *KDD*.

[15] Satoh, Soichirou, Mamoru Mimuro, and Ayumi Tanaka. "Construction of a phylogenetic tree of photosynthetic prokaryotes based on average similarities of whole genome sequences." *PloS one* 8.7 (2013): e70290.

[16] Qin, Ling, et al. "A novel approach to phylogenetic tree construction using stochastic optimization and clustering." *Bmc Bioinformatics* 7 (2006): 1-12.

[17] Shaktawat, Priyanka, and Parvati Bhurani. "A Review: Phylogeny Construction Methods."

[18] Chatzou, Maria, et al. "Multiple sequence alignment modeling: methods and applications." *Briefings in bioinformatics* 17.6 (2016): 1009-1023.

[19] Lin, Yu, et al. "Assembly of long error-prone reads using de Bruijn graphs." *Proceedings of the National Academy of Sciences* 113.52 (2016): E8396-E8405.

[20] Kamath, Govinda M., et al. "HINGE: long-read assembly achieves optimal repeat resolution." *Genome research* 27.5 (2017): 747-756.

[22] Chin, Chen-Shan, et al. "Phased diploid genome assembly with single-molecule real-time sequencing." *Nature methods* 13.12 (2016): 1050-1054.

[23] National Center for Biotechnology Information (NCBI) [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – Accessed Dec 2019.