NOVEL APPROACHES TO MARKER GENE REPRESENTATION LEARNING USING TRAINED TOKENIZERS AND JOINTLY TRAINED TRANSFORMER MODELS

by

Alexander Manuele

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2021

This thesis is dedicated to front-line workers, healthcare and otherwise, who kept our world turning in this tumultuous year.

Table of Contents

List of Tables . . . . . **vi**

List of Figures . . . . . **vii**

Abstract . . . . . **ix**

List of Symbols and Abbreviations . . . . . **x**

Acknowledgements . . . . . **xii**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 Marker Genes and Microbial Community Analysis 2
1.2.1 Ribosomal RNA Genes 3
1.2.2 Amplicon Sequence Data 4
1.2.3 Challenges and shortcomings of amplicon sequence data 7

1.3 Machine-learning applications with amplicon sequence data 8
1.3.1 Considerations in machine learning: Model selection 9
1.3.2 Considerations in machine learning: Feature selection 9

1.4 Segmentation of DNA sequences 10
1.4.1 k-mer based segmentation 11
1.4.2 Statistically driven segmentation of DNA sequences 12

1.5 Representation learning of text data 12
1.5.1 Word embedding 13
1.5.2 Sub-word tokenization 14
1.5.3 Transformers 15
1.5.4 Sentence transformer 18

1.6 Representation learning of DNA sequence data 18
1.6.1 k-mer embedding 20

1.7 Contributions 22

1.8 Thesis outline 22

Chapter 2 Statistically Driven Segmentation of DNA Sequences . **24**

2.1 Data acquisition and preprocessing 24

2.2 Byte-pair encoding . . . 25

2.3 Unigram tokenization . . . 27

2.4 16S Sequence Embedding . . . 28

2.5 Results . . . 30
2.5.1 Properties of encoded sequences . . . 30
2.5.2 Machine-learning assignment of taxonomy . . . 36
2.5.3 Sequence embedding assessment . . . 42

2.6 Conclusions . . . 46

Chapter 3 Bi-Directional Transformers Produce High-Quality Sequence Embeddings with Multiple Use Cases . . . 50

3.1 Masked Language Model . . . 51
3.1.1 Transformer model architecture . . . 51
3.1.2 Data set preparation . . . 55
3.1.3 Training . . . 55

3.2 Sequence embedding: Siamese network architecture . . . 56
3.2.1 Sequence transformer architecture . . . 56
3.2.2 Sequence transformer loss and optimization . . . 56
3.2.3 Data set preparation . . . 58
3.2.4 Training . . . 58

3.3 Quality of sequence transformer embeddings . . . 58
3.3.1 Spearman correlation of pairwise-alignment scores and cosine similarities . . . 59
3.3.2 Clustering of embedded sequences . . . 60
3.3.3 Classification of embedded sequences . . . 60
3.3.4 Nearest sequence lookup . . . 62

3.4 Classification of host phenotype from amplicon sequence data . . . 63
3.4.1 Data set preparation . . . 64
3.4.2 Sample-level embedding . . . 65
3.4.3 Classification of samples . . . 68

3.5 Conclusions . . . 68

Chapter 4 Conclusion . . . 71

4.1 Implications . . . 71

4.2 Limitations . . . 72

4.3 Future directions 75

Bibliography **78**

List of Tables

2.1 Clustering metrics for SIF embeddings calculated using different tokenization strategies at different taxonomic ranks. 47

2.2 Time in seconds to complete 5 epochs of Word2Vec training using different tokenization strategies. 47

3.1 Spearman correlation coefficient values between cosine similarity of embedded sequences and pairwise alignment distance of raw sequences. 60

3.2 Clustering metrics for sequence embedding at different taxonomic ranks using both SIF embedding and SRoBERTa embedding. 61

3.3 Performance metrics of 5-fold cross validated classification of embedded 16S sequences to different taxonomic ranks using random forest classifiers. 61

3.4 Summary of data sets used for microbiome sample classification. 64

3.5 Classification of disease state from stool microbiota using center-log ratio normalized abundance values and SRoBERTa embedding aggregation for data sets from five studies. 69

List of Figures

1.1 Structure of the prokaryotic ribosomal rRNA. 5

1.2 Segmentation of DNA into 3-mers. 11

1.3 Overview of Word2Vec architectures. 13

1.4 Overview of the transformer model architecture. 16

1.5 Overview of the cosine similarity sentence embedding objective. 19

1.6 Overview of the smooth inverse-frequency (SIF) sequence embedding approach. 21

2.1 Distribution of token lengths as they appear in encoders' vocabularies. 32

2.2 Distribution of token frequencies using various token encoding methods. 33

2.3 Inverse rank-frequency plots for token frequencies in SILVA data set. 35

2.4 Length distributions in number of tokens of 16S sequences encoded using various tokenization strategies. 36

2.5 Performance of models assigning taxonomic labels to KEGG database 16S sequences. 39

2.6 Pearson correlation of features selected in the 16S genus classification task. 41

2.7 Scree plot showing proportion of variance explained by top 20 principal coordinates from the 16S genus classification task. . . 42

2.8 Mean accuracy of random forest classification of genus using dense vector embeddings of DNA sequences using different tokenization strategies and parameters. 44

3.1 Left-right encoder-decoder transformer architecture vs bidirectional encoder-only architecture. 53

3.2 Overview of the SRoBERTa training process. 57

3.3 Distribution of overlaps of top five results between BLAST homology search and cosine similarity nearest-neighbor lookup. . 63

3.4 Overview of the microbiome sample embedding procedure. . . 66

Abstract

Next-generation DNA sequencing technologies have made marker-gene DNA sequence data widely available. Analysis of microbiome data has many challenges, including sparsity, high cardinality, and intra-study dependencies during feature engineering. Language-modelling techniques may provide the means to overcome these challenges. The first step in sequence modelling is dividing the sequence into sensible tokens. We show that trained tokenization strategies, byte-pair encoding and unigram language modelling can replace traditional sliding-window based segmentation techniques for DNA marker genes in classification, clustering, and language-modelling tasks. We then propose a novel approach for feature representation of DNA marker genes, proposing a training scheme to learn dense vector representations of DNA sequences using transformer language models optimized using DNA sequence pair-wise alignment scores. We demonstrate that our representations match or exceed previously published approaches for treatment of individual marker genes and of microbiome samples while providing fixed-length, low-cardinality representations of each.

List of Symbols and Abbreviations Used

$\in$ Denotes set membership.

$\mathbb{R}$ The set of real-valued numbers.

$\mathbb{Z}$ The set of all integers.

AMI Adjusted Mutual Information.

ANOVA Analysis of Variance.

API Application Programming Interface.

ARI Adjusted Rand Index.

ASV Amplicon Sequence Variant.

BERT Bidirectional Encoder Representations from Transformers.

BLAST Basic Local Alignment Search Tool.

BPE Byte Pair Encoding.

CBOW Continuous Bag-of-Words.

CLR Centered Log-Ratio.

CPU Central Processing Unit.

DADA2 Divisive Amplicon Denoising Algorithm 2.

DNA Deoxyribonucleic acid.

GPU Graphics Processing Unit.

KEGG Kyoto Encyclopedia of Genes and Genomes.

LSTM Long Short-Term Memory.

MLM Masked Language Model.

MSE Mean Squared Error.

NCBI National Center for Biotechnology Information.

NGS Next-Generation Sequencing.

OTU Operation Taxonomic Unit.

PCoA Principal Co-ordinates Analysis.

QIIME2 Quantitative Insights Into Microbial Ecology 2.

RESTful Representatonal State Transfer.

RF Random Forest.

RoBERTa Robustly Optimized BERT Approach.

rRNA Ribosomal Ribonucleic Acid.

SIF Smooth Inverse Frequency.

SSU Small Subunit.

SVD Singular Value Decomposition.

Acknowledgements

There are several people whom I would like to take this opportunity to thank. First and foremost, my supervisor Dr. Rob Beiko, for providing endless opportunities for me both academic and professional, for his tireless effort in shaping this research project and document into what it is now, and for being generally a good boss and supervisor, even through remote working conditions.

I would like to thank my committee members, Dr. Sageev Oore and Dr. Vlado Keselj, not only for taking the time and effort to act as readers but also for their excellent instruction through coursework.

I would also like to thank the members of our lab for forming a community nurturing of curiosity and research. Particular thanks go to Fin, Mike, and Diana for always being available to answer questions, bounce ideas off of, and provide research papers.

Finally, I would like to thank my loved ones, for consistently supporting my ambitions and goals.

Chapter 1

Introduction

1.1 Motivation

DNA sequences are frequently used as proxies for biodiversity, and important ecological questions are now being asked using machine learning. For example, diagnostic machine-learning algorithms are being developed to predict human disease state given a collection of microbial DNA sequences taken from a body site on the subject [47, 80, 101, 96, 99]. One of the most significant challenges of DNA machine learning tasks is the question of feature representation. DNA must be represented numerically before it can be processed by a machine-learning algorithm. When choosing a feature representation, one must consider that DNA sequences are typically variable length and comprised of a character set that is comprised of only four characters. A typical approach for DNA sequence representation is to segment DNA into overlapping chunks, called k-mers, count the k-mers in a sequence, and use the resulting count vector as a feature. While this approach is sufficient for some use cases, the process destroys contextual information and string-based distance measures, limiting their applications.

Current practices for the treatment of DNA proxies for ecological samples prevent the development of production-ready machine learning algorithms. The most common approach for feature representation for these data is to construct features from a collection of samples. An undesirable consequence of this that a corresponding feature vector is entirely dependant on properties of the data set from which the sample was taken. This prevents pre-trained ML models from being used with test data from outside of the study used for training.

Recent research has investigated the application of techniques popular in the natural language processing (NLP) domain with DNA sequence data [17, 55, 32, 92, 49]. Of note, researchers have investigated the use of language modelling and representation learning techniques to develop novel feature representations for DNA sequences,

with the goals of creating dense, fixed-length representations of sequences that meaningfully capture distances among them [55, 92]. These studies were successful, but were limited by their use of normalization processes involving statistics calculated over an entire data set.

We propose that adapting contemporary language-modelling techniques for use with DNA sequences will allow us to develop novel DNA feature representations which can overcome the limitations of previous work. The development of robust fixed-length dense vector representations of DNA will allow researchers to create feature representations of collections of DNA sequences by performing vector arithmetic over the vector embeddings of the sequences. As language-modelling tasks generally scale in compute complexity, memory demand, or both as the length of sequences increases, it is important to identify viable methods to reduce the lengths of DNA input representations for such tasks.

1.2 Marker Genes and Microbial Community Analysis

Over the last 30 years or so, the tools and processes available to microbial ecologists have changed drastically. Historically, researchers interested in the diversity and composition of microbes living in a given environment had to resort to culturing samples in a wet-lab setting and directly observing the microbes using microscopy, characterizing them by morphological traits. While studies employing such methods have advanced microbial life sciences, they faced a major limitation: The majority of microbial organisms cannot survive and grow in a laboratory setting and thus fail to be represented in experiments that rely on culturing [30]. In light of this, researchers proposed that un-culturable microbes might be compared and categorized by differences in their DNA sequence [24].

With the advent of high-throughput next-generation sequencing (NGS) techniques, researchers have acquired the ability to extract genetic material directly from an environment, forgoing the need for culturing and thereby allowing them to better characterize microbial diversity. Differences in DNA sequence allow different organisms from a given environment to be distinguished, thus allowing detailed characterization of a community of microorganisms (i.e., the *microbiome*).

There are multiple strategies for extracting microbial genetic information from an

environment. "Metagenomic" methods aim to characterize microorganisms by randomly sampling from all the genetic information present in an environmental sample. Other, more-focused methods target specific DNA sequences corresponding to individual genes. It is common to be interested in the taxonomy of microbes in a sample. In the context of living organisms, taxonomy is a classification schema that places organisms into increasingly comprehensive hierarchical groups [69]. If researchers target a *ubiquitous* gene that is present in all organisms, they may perform taxonomic analyses of organisms by comparing sequence differences among the recovered genes. In order to detect and characterize specific DNA sequences from a sample, researchers must have *a priori* knowledge of the composition of some part of the target sequences. This is because the DNA sequencing takes place by first amplifying small regions of DNA using a series of biochemical reactions. To perform this amplification, researchers must design molecular anchors (called "primers") that flank the target sequence, which requires knowledge of the flanking DNA sequence.

The process of sequencing specific genes and using the sequences to differentiate organisms is called marker-gene analysis. When selecting a marker gene, researchers will usually select a gene that contains both sequence regions that are highly similar across organisms, and regions that are more variable. The conserved regions are used to design "universal" primers that can detect and sequence the marker genes from all organisms of interest. Differences in variable gene regions are then used as diagnostic markers to distinguish different organisms in the sample. The presence and relative abundance of unique marker-gene sequences can then serve as a proxy for the relative quantities of microbes in a sample.

1.2.1 Ribosomal RNA Genes

The most common targets in marker-gene analyses are the genes that encode ribosomal RNA (rRNA). Ribosomes are ubiquitous cellular components with functions that are critically essential for all life, and thereby all living cells contain ribosomal genes. All ribosomes comprise large and small subunits, each of which in turn comprises proteins and small RNA molecules, called rRNAs (figure 1.1a). In the case of prokaryotes, the rRNA genes found in the ribosome are referred to as 23S, 16S, and 5S. The 23S and 5S rRNA are localized to the ribosome's large subunit and the 16S

rRNA to the ribosome's small subunit.

16S is a suitable candidate for marker gene analysis for several reasons. Firstly, it is found in every prokaryotic cell, meaning that 16S amplification can theoretically detect all bacteria and archaea in a sample. Additionally, 16S rRNA genes must maintain a specific three-dimensional structure to function properly. This property has had the evolutionary result of the 16S containing several highly conserved regions; mutations in these regions very rarely survive and spread in a population as they typically result in the death of the organism. These conserved regions flank regions of DNA with highly variable sequences which mutate easily and generally vary from species to species. Thus, 16S sequences can be easily detected and amplified via the design of primers that target one of the conserved regions, then compare the differences in the variable regions to identify the unique species in a sample (figure 1.1 (b)).

1.2.2 Amplicon Sequence Data

One of the most common instruments for performing marker-gene analysis is the Illumina MiSeq [90]. Given a sample, these sequencing machines returns thousands of "short read" sequences of length 150-250 nucleotides with associated quality scores. Although this level of DNA sequencing can in principle allow the identification of even rare organisms, the sequencing process is imperfect and can generate infrequent errors in sequences that might otherwise be identical. Therefore, it is not feasible to assume that all unique short reads correspond exactly to unique organisms. In light of this, researchers have developed approaches to approximating the "true" unique sequences from a set of short-read sequences.

An early and still widely used approach to mitigate the errors of sequencing instruments is to cluster sequences based on their sequence identity into "operational taxonomic units" (OTUs). Several OTU clustering methods group sequences with at least 97% sequence identity into a cluster, perhaps motivated by a proposal in the late 1990s that a sequences with a 16S percent identity of 97% or greater could be used to delineate species [21, 78, 79]. After clustering, each OTU will have an associated "representative sequence", an unambiguous nucleotide sequence which represents all 16S sequences in that OTU and is used for subsequent taxonomic assignment and

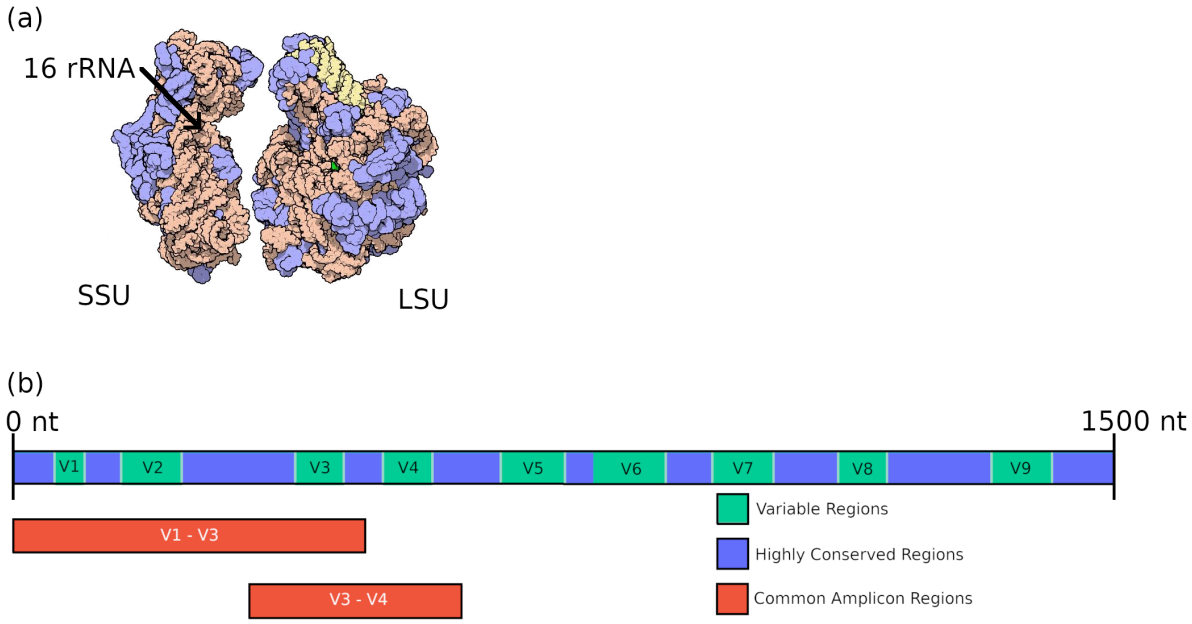

Figure 1.1: Prokaryotic ribosomal rRNA. (a) Space-filling model of prokaryotic ribosome. The ribosome exists as a complex of proteins and RNA molecules. The left piece shows the small subunit, comprising 16S rRNA complexed with several proteins. Proteins are shown as purple, and rRNA molecules are shown as pink and yellow. The right piece shows the large subunit, comprising 23S and 5S rRNA complexed with several proteins. Adapted from lecture slides courtesy of Robert Beiko. (b) Approximate layout of the 16S prokaryotic small subunit rRNA gene. The 16S gene is approximately 1500 nucleotides in length and consists of several highly conserved regions and nine hypervariable regions. Variable regions are shown in green, conserved regions are shown in blue. Shown in red are two examples of commonly used targets for amplicon sequencing, the V1-V3 region and the V3-V4 region. Modified from Ricci et al. [72].

analyses. While the 97% sequence identity threshold is still commonly used, contemporary OTU clustering algorithms allow users to specify the percent identity as a parameter and various studies use different values.

While OTU clustering has provided a means to make analysis of high-throughput amplicon sequencing data tractable, it is not without shortcomings. Researchers have shown that minor adjustments in the OTU clustering procedure can significantly affect downstream analyses and interpretations. For example, He et al. [29] have shown that small changes to the number of sequences or samples included in OTU clustering calculations will significantly affect the resultant OTUs, showing that OTU clustering procedures are unstable. Others have shown that the 97% identity threshold is a sensible value for delineating certain species, but is too sensitive for some (i.e. separates species which should not be considered separate) and not sensitive enough for others (i.e. collapses species which ought to be considered separate) [56, 51]. Additionally, the distance metrics used to calculate similarity between sequences for OTU clustering may result in poor-quality clusters [56]. In spite of these shortcomings, OTUs remain widely used in studies of microbial diversity and ecology. Readers should understand that these data should be considered noisy and imperfect.

Quality scores indicate the level of confidence that the sequencing machine has identified the correct nucleotide. This information can be used to mitigate the impacts of sequencing errors, and are often used as a prefiltering step to remove reads of very low quality. In 2016, Callahan et al. [12] proposed an alternative algorithm to OTU clustering which uses the quality scores in conjunction with the short-read sequences to produce "Amplicon Sequence Variants" (ASVs) in a process generically referred to as "denoising". This work was published under the name DADA2. The process of denoising reads into ASVs replaces the process of clustering reads into OTUs: Instead of clusters, sequences are denoised and then assigned IDs based on uniqueness. Since the development of DADA2, additional denoising algorithms have been introduced, such as Deblur [3]. ASVs exhibit higher fidelity and more consistent representations of sample ecology than OTU clustering, and ASV-based analyses are becoming increasingly common in the literature [34, 13, 62].

1.2.3 Challenges and shortcomings of amplicon sequence data

Though amplicon sequencing has been a tremendous boon to microbial ecology research, it is not without shortcomings. A major limitation of amplicon sequence data is that it is inherently compositional [28, 27]. Since there is an intrinsic upper bound to the amount of reads a DNA sequencer can produce, the number of reads produced in a run cannot be considered reflective of the total number of DNA molecules, and consequently the total number of organisms, in a sample. As the magnitude of count values is reflective of the capacity of the sequencing machine rather than the magnitude of the organisms in a sample, it is best practice for researchers to treat the sequence count values as compositional data [27]. Compositional data are characterised by bias towards negative correlations and laden with spurious correlations, which can have serious implications for ordination analyses, clustering, network analysis, and statistical tests of differential abundance [27, 28]. An increasingly common approach is to project the count data to a simplex using a log-ratio transformation [27, 28, 46, 60]. While these recommendations are well-motivated by mathematical first principles, comparative analyses and machine-learning methods based on simple counts or proportions are still in widespread use.

Another significant challenge with amplicon sequencing data, and one which we aim to address in this work, is the difficulty of comparing samples across studies in the context of machine learning. Most published works that describe an amplicon-based machine-learning workflow are based on relative abundance or count vectors of either OTUs or ASVs [47, 84]. The algorithms which produce OTUs and ASVs accept raw read information as input and produce representative sequences and count tables as output. These count tables are essentially $S \times D$ matrices, where S is the number of samples from the study and D is the number of OTUs/ASVs detected. OTU/ASV count vectors generated from the set of samples are then used as feature vectors $v \in \mathbb{R}^D$ that are used to train models. While useful, models trained using these feature vectors cannot be extended to previously unseen test data that contain OTUs/ASVs that are absent from the training and validation sets.

As an illustrative example, consider two hypothetical studies in which separate teams of researchers wish to train a model to detect colorectal cancer using microbiome data from human stool samples collected in house. Suppose the first team

performs DNA sequencing and ASV denoising and acquires a $S_\alpha \times D_\alpha$ ASV count matrix. They then train their model to predict a class label using vectors $v \in \mathbb{R}^{D_\alpha}$ as input. Concurrently, the second team also performs sequencing and denoising using their samples and acquire a $S_\beta \times D_\beta$ ASV matrix. They train their model to output a label given some vector $v \in \mathbb{R}^{D_\beta}$. Because D_α and D_β were calculated in two separate studies, they will only be equal in the highly unlikely scenario that both studies detected precisely the same set of unique ASVs. In this context, then, two teams must both create machine-learning models to perform the same task using the same type of data, but each model will be incompatible with the other team's data.

1.3 Machine-learning applications with amplicon sequence data

Machine learning can be applied to amplicon sequence data at different steps in the analysis process. An important step in microbial community analysis pipelines is the assignment of taxonomic information to short reads, i.e. predicting the specific organism of interest for a given representative sequence. There are multiple approaches to these assignments, but the most popular include sequence similarity searches, machine-learning approaches trained with reference databases, and combinations of the two [8, 33, 88].

In addition to classification of individual sequences, there has been increased interest in the ability to use microbiome marker-gene data sets to predict environmental traits as a machine learning classification problem, particular in the prediction of human disease states from biopsies and/or stool samples [15, 97, 23, 101, 80, 67, 96, 99, 57, 99, 20]. Such investigations range in methodology and aim: some use data mining methods to identify whether specific species and species distributions are associated with specific host traits, whereas others frame prediction of a disease state as a machine learning classification problem using the microbiome data as input. Each approach offers advantages and shortcomings: The data-mining approaches may reveal helpful insights into the biological roles of specific microbes in their respective communities. The classification approaches may serve as inductive research tools, allowing researchers to inspect which features were weighted as important by their classifier and to inspect the traits of samples which confound their model. Microbiome sequencing may someday play a role in diagnostics, providing incentives to develop

effective machine-learning tools given the complexity of microbial communities. Currently, most machine learning studies use some transformation of OTU or ASV data as feature vectors, preventing the use of these models in production (as discussed in section 1.2.3).

1.3.1 Considerations in machine learning: Model selection

When performing machine learning analyses, it is important to consider the choice of model. It is widely accepted that there is no single machine learning model which will achieve the best possible performance for all possible data sets [93]. Model performance will be significantly affected by the factors such as the dimensionality of the sample data (i.e. number of features), the distribution of class labels, and the number and proportion of relevant features in the data set. For example, Ng [54] has shown that sample complexity of rotationally invariant models, such as support vector machines and artificial neural networks, scales linearly with the number of irrelevant features in the sample data whereas rotationally variant models such as Naïve Bayes classifiers and decision trees scale log-linearly. Additionally, some problems cannot be solved by learners which act as combinations of linear functions; these problems may require non-linear functions such as radial basis kernel methods [83].

With the availability and ease of use of contemporary machine learning programming libraries and APIs such as scikit-learn [58], it is a reasonably straightforward exercise to trial multiple machine learning models when attempting to design a good estimator for a given problem. Therefore, we consider it to be good practice to repeat experiments with multiple models where computing resources allow.

1.3.2 Considerations in machine learning: Feature selection

The microbiome data used in host phenotype prediction are typically high dimensional and sparse [57, 38]. Additionally, the sampling and sequencing procedures involved in acquiring microbiome data generally result in data sets with n samples on the order of tens or a few hundreds [84, 20]. The inevitable result is a data set in which $n_{features} \gg n_{samples}$. Most estimators require some ratio of $n_{samples}/n_{features}$ to successfully learn a discriminator function which can generalize to data outside their training set and generally perform better as the ratio of samples to features

increases [53, 54]. Therefore, applying methods which reduce the number of features in a data set can result in improved estimator model performance for microbiome data sets.

In addition to being used to improve model performance, feature-selection algorithms can be used to investigate which features in a data set are most informative or important. However, as with machine learning models, feature-selection approaches differ in their motivation and efficacy. Broadly, feature-selection algorithms can be divided into univariate approaches, which consider individual features, and multivariate approaches, which consider feature subsets [35]. Each approach has advantages and drawbacks; univariate approaches are tractable and interpretable but may fail to identify key interactions among features, whereas multivariate approaches can identify higher-order features but must rely on heuristics to perform their searches [35]. The ability to identify these higher-order feature interactions may be important in microbial community analysis, as microbes are known to interact significantly in their environment in both competitive and mutualistic manners [74, 81]. The effectiveness of different feature-selection algorithms can be data set specific, therefore single feature-selection models should not be used to draw conclusions without a thorough understanding of the distribution, variance, and correlation structure of a data set and its features.

1.4 Segmentation of DNA sequences

An essential first step in the machine learning analysis of DNA data is the choice of representation. As text data, DNA cannot be directly passed to a machine-learning model; it must first be converted to some numerical representation.

Recall that a molecule of DNA is comprised of a series of nucleotides, of which there are four types. Therefore, any DNA sequence S is composed of characters $c \in C, C = \{A, G, C, T\}$. One possible numerical representation of a DNA sequence of n characters is to convert characters to one-hot vectors. Since $|C| = 4$, a one-hot representation of a DNA sequence of length n would become a $n \times 4$ matrix.

Segmentation strategies, in which DNA sequences of length n are decomposed to form some list of tokens, are widely used. This section describes k-mer segmentation, perhaps the most common form of DNA segmentation, as well as previous

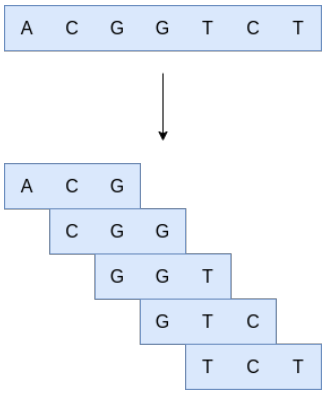

Figure 1.2: Segmentation of a short DNA sequence into overlapping 3-mers. A sequence of length n will produce $n - k + 1$ k-mers.

work describing a statistically motivated DNA segmentation approach. Finally, we describe contemporary statistical segmentation approaches that have been used in non-biological text processing fields.

1.4.1 *k*-mer based segmentation

A k-mer refers to a DNA sequence of length k. In the context of DNA segmentation, DNA sequences are usually segmented into k-mers that overlap with an overhang of one. For a DNA sequence of length n and a k-mer size k, this results in a list of k-mers of size $n - k + 1$ (figure 1.2).

While k-mer segmentation of DNA is somewhat naïve, it is effective in several applications. Taxonomic classifiers based on machine learning represent their sequences as k-mers prior to feeding them to their model [8]. There are multiple ways to represent lists of k-mers as numeric data. Similar to raw nucleotide sequences, it is possible to one-hot encode a list of k-mer sequences. However, a one-hot encoded approach will typically result in a highly sparse feature space, which results in analysis difficulties. Instead, researchers may opt to represent lists of k-mers as vectors $v \in \mathbb{Z}+^{4^k}$, such that $v[i]$ represents the ith value in a sorted list of possible k-mers. An individual sequence can then be represented as either a presence/absence vector in which $v[i] = 1$ if k-mer i is present in the sequence, 0 otherwise, or as a count vector, in which $v[i]$ will be assigned the count of the number of times k-mer i appears in the sequence.

When segmenting DNA into k-mers, the choice of k can have a significant impact

on the downstream analysis. Larger sizes of k provide the opportunity to detect rare sequence combinations, which may provide valuable information to an analysis. However, the cardinality of the feature vector scales exponentially with the size of k, resulting in increasingly sparse data that requires large amounts of memory. This is a trade-off which researchers much carefully navigate for their specific application.

1.4.2 Statistically driven segmentation of DNA sequences

While k-mer segmentation strategies are often effective, it is possible to choose a DNA segmentation strategy that is based on calculated principles rather than based on a fixed-length sliding window. For example, Liang [44] showed that it is possible to segment DNA using a maximum likelihood n-gram model.

Similar to a k-mer, an n-gram refers to a series of characters of length n, with the distinction being that k-mers are typically overlapping and n-grams are not. The model described by Liang [44] is simple: Given a sequence of n-grams X, predict token x_i given $x_{i-(n-1)}, ..., x_{i-1}$ by choosing $max(P(x_i|x_{i-(n-1)}, ..., x_{i-1}))$. The authors reduced the search space of the probabilities by restricting the size of n to an interval and by solving using the Viterbi algorithm rather than brute force. Their approach remains a proof of concept and has not been used to explore downstream applications, though they showed that approaches previously reserved for natural language processing such as n-gram segmentation are viable for DNA.

1.5 Representation learning of text data

Text analysis in computational tasks using tokenized input has been historically successful, but has significant limitations. For example, in the examples described above (k-mer and n-gram tokenization of sequences), tokens are represented as indices in some vocabulary. While such representations can be used successfully in many tasks, by nature they do not encode any distance measurement among tokens. In the field of natural language processing, the inability to quantify distance between n-gram or word tokens resulted in stagnation in tasks such as machine translation and speech recognition [50, 7]. As such, researchers have developed techniques for training machine-learning models to project tokens to a meaningful vector representation, a

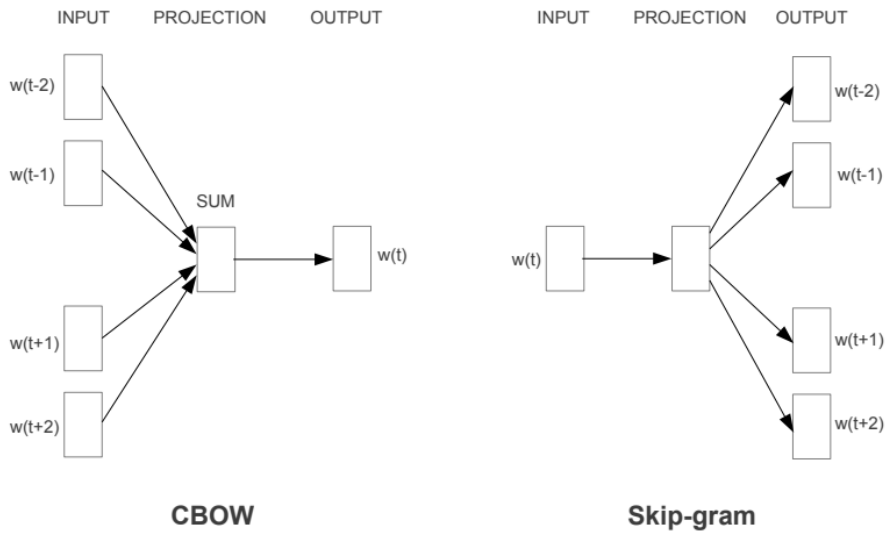

Figure 1.3: Overview of Word2Vec architectures. CBOW architecture uses words in a surrounding context to predict the target word. Skip-gram uses a current word to predict the words in the surrounding context. From Mikolov et al. [50]

process broadly defined as representation learning. Here, we discuss major developments in representation learning of text data and describe how they have been applied to DNA sequence data.

1.5.1 Word embedding

A major development in representation learning for text data was the development of word vector models. The motivation of a word vector model is to train some machine learning model to associate words from a vocabulary to a distributed real-valued vector space, $\mathbb{R}^d$ [6, 50, 7]. The process of projecting words from a vocabulary to a real-valued vector space is termed "word embedding". Some of the most successful such models were introduced by Mikolov et al. [50], who introduced two novel neural network architectures for learning word vectors, collectively termed Word2Vec.

In their paper, Mikolov et al. [50] described two related word embedding approaches, which they called Skip-gram and CBOW (figure 1.3). Both models are simple log-linear classifiers using a hierarchical softmax "projection layer" and trained with stochastic gradient descent. In the CBOW (Continuous Bag-of-Words) model, given a target word w in a sentence, n words upstream and n words downstream of the

target word are passed to the projection layer and averaged. The averaged projection layer is then used to predict the target word w using the hierarchical softmax operation. Conversely, in the Skip-gram model, a target word w is passed to the projection layer and then used to predict the values of n upstream and n downstream context words. The authors demonstrated that both approaches can be used to project words onto a real valued vector space by discarding the softmax layer after training (i.e. word vectors are the value produced by the projection layer). Famously, the vectors seem to encode semantically meaningful information, allowing vector arithmetic to produce sensible results (e.g. $vector(\text{“}biggest\text{”}) - vector(\text{“}big\text{”}) \approx vector(\text{“}smallest\text{”})$).

Subsequent to the development of Word2Vec, alternative word embedding models built on similar principles emerged, such as FastText, which incorporates the use of sub-word n-grams to enrich the word vectors [7], and GloVe, which incorporates global information in addition to the local context information used in Word2Vec [59].

1.5.2 Sub-word tokenization

When creating numerical representations for DNA sequence data, it is sensible to use k-mer representations as described above. As DNA is composed of a fixed set of four possible nucleotide characters, k-mer representations will have fixed vocabularies of size 4^k. When modelling natural languages, however, the question of how to define a vocabulary is not so simple. One way of addressing the problem of the open vocabularies found in natural languages is the introduction of sub-word tokenization, in which texts are tokenized into segments shorter than words rather than by dictionary-defined or white space separated words [11]. In addition to providing more flexible language modelling capabilities, researchers have found that sub-word tokenization improves representation learning outcomes and makes models more robust and generalizable than using words as tokens [11, 40, 45, 39].

Here, we briefly summarize two popular sub-word tokenization algorithms, byte pair encoding (BPE) and unigram language modelling. We propose that these approaches may offer sensible alternatives to k-mer segmentation for DNA sequences.

Byte pair encoding is a greedy iterative algorithm which trains an encoder to build a vocabulary with which to segment text. Byte pair encoding was originally proposed in 1994 as a data compression algorithm [25]. The algorithm proceeds by scanning

a training corpus for common sub-string pairs and then replacing those pairs with symbolic tokens, which are added to a vocabulary that is initialized to contain the character set of the corpus.

BPE has the desirable traits of allowing user-specified vocabulary size and being able to handle previously unseen sub-strings by tokenizing text using the character set as tokens. The simple, effective, and flexible characteristics of BPE have made it the *de facto* tokenization strategy for language modelling and neural machine translation [76, 11, 39, 19, 45, 40].

Although BPE has become the dominant tokenization strategy in language modelling tasks, some research has indicated that an alternative strategy, unigram language modelling, may be preferable [11, 39]. Unigram language modelling is an approach which attempts to calculate the most likely tokenization of a text using expectation maximization, thereby considering multiple sub-word candidates. Unlike BPE, which iteratively grows a vocabulary, unigram language modelling proceeds by initializing a large vocabulary to contain common sub-strings found in a training corpus, calculating the most likely tokenization for each document in the corpus, and then iteratively eliminating tokens with low likelihoods until a target vocabulary size is met. Some research has shown that by considering multiple sub-word candidates, language models may become more robust to noise and may be more inclined to learn semantically meaningful text segmentations [11, 39].

1.5.3 Transformers

The field of natural language processing has shifted from application specific modelling to pre-training large language models and later fine tuning those models for application-specific tasks [91]. The advent of this paradigm shift can be attributed to a multitude of factors, not the least of which is the development of novel effective architectures. In particular, a machine learning model architecture referred to as the transformer [85] has seen wide-spread adoption. The paradigm of fine tuning a large, pre-trained transformer model has been used to achieve state-of-the art results in a wide range of tasks, including but not limited to text classification, machine translation, text summarizing, and commonsense inference [45, 86, 87, 94, 41, 10, 43].

The transformer is an architecture designed for sequence transduction tasks [85].

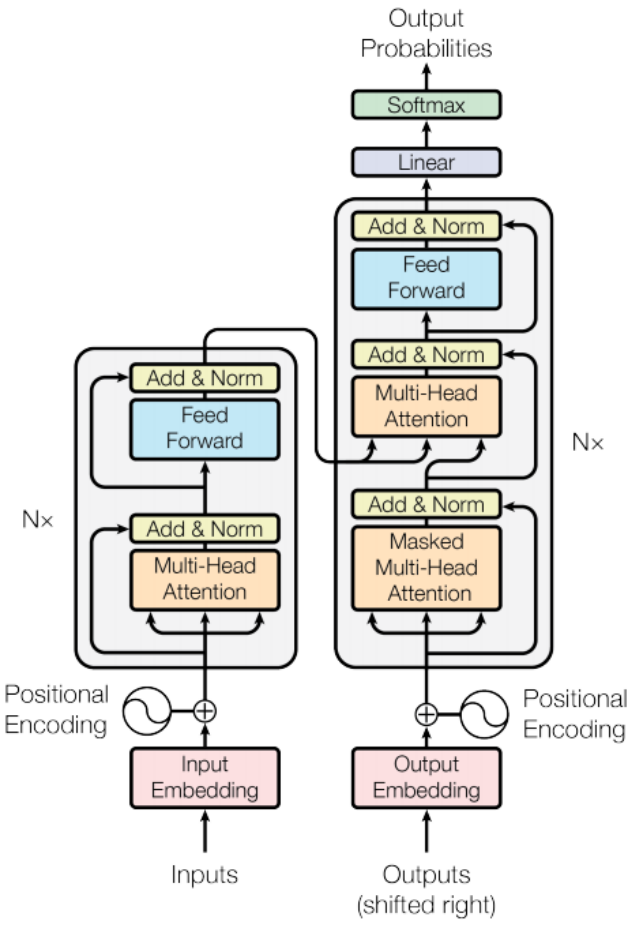

Figure 1.4: Overview of the transformer model as originally described in Vaswani et al. [85]. Transformer architectures are encoder-decoder architectures characterized by stacks of layers containing multi-headed self-attention layers followed by fully connected feed-forward layers. Each layer is followed by a residual connection and batch normalization ("Add & Norm"). The transformer represents input position using a positional encoding operation applied to the inputs. Figure from Vaswani et al. [85].

Prior to the development of the transformer, state-of-the-art sequence modelling was achieved through the use of recurrent, encoder-decoder sequence models such as long short-term memory (LSTM) networks and gated recurrent networks [82, 16, 5]. Recurrent model architectures typically function by performing successive computations along steps of the input and output sequence. While these methods have been successful in a wide variety of applications, the sequential manner of their calculations is a massive limitation as it prevents parallelisation. Some researchers proposed mechanisms to reduce or the amount of sequential calculations required to model sequences, such as purely convolutional sequence modelling [26]. However, the number of computations required to model intra-sequence dependencies scales with the distance between those dependencies [85]. Instead of modelling sequences as states-over-time using auto-recurrent hidden states, Transformer architectures use attention mechanisms to model global dependencies between input and output sequences [85].

The transformer architecture is an encoder-decoder architecture characterized by stacks of identical layers in both the encoder and decoder. Each stack contains a multi-headed attention mechanism as well as a fully connected feed-forward network. To allow the network to model positional dependencies in input data, both the encoder and decoder have a positional embedding layer prepended (figure 1.4). By using attention instead of recurrence or convolution to model sequences (both of which scale in compute costs with the distance of intra-sequence dependencies), transformer is able to compute long-term dependencies in constant time [85]. In addition to the constant time computation of sequence dependencies, attention is parallelizable, allowing Transformer models to take advantage of GPU accelerated parallel processing. We discuss the mechanisms of attention and Transformer models in more detail in chapter 3.

Though the transformer was originally designed specifically for sequence transduction tasks (specifically neural machine translation), researchers soon applied variations of the architecture on the paradigm of pre-training large models for general language "understanding" and later fine tuning those models for task specific performance [65, 19]. For example, the BERT model uses bidirectional encoders in a transformer architecture, pre-trained using a masked language model objective in which tokens in an input sequence are randomly ablated and the model is tasked

with predicting the token ID of the original tokens [19]. This practice of pre-training large language models and subsequently tuning them with some supervised learning objective has largely replaced other language modelling techniques, achieving state of the art performance in several natural language processing tasks [91].

1.5.4 Sentence transformer

One interesting and relevant application of pre-trained transformer models is the ability to fine-tune them to generate semantically meaningful continuous vector representations $v \in \mathbb{R}^d$ of full sentences. Reimers and Gurevych [71] showed that pre-trained transformer language models can be tuned to learn sentence-level representations by simply adding a pooling layer to the network's output and fine-tuning the resulting model in a jointly-trained configuration using annotated sentence pairs (figure 1.5). For the task of computing sentence-level vector embeddings, the model is trained using sentence pairs annotated with a similarity score between 0 and 1 and minimizes the distance between the annotated score and the cosine similarity of the embedding vectors of the input sentences, using mean-squared error loss to update the network weights via back-propagation [71].

1.6 Representation learning of DNA sequence data

The development and use of vector representations for text data in the natural language processing domain has not gone unnoticed by biologists and bioinformaticians. In 2017, a researcher published a proof-of-concept pre-print showing that a Word2Vec style approach could be applied to DNA k-mers [55]. That is, they trained a skip-gram network architecture and training objective to project DNA fragments of length k to a continuous vector representation $v \in \mathbb{R}^d$. They showed that cosine similarity in the projected vector space correlated with DNA sequence alignment scores, suggesting that the embedding approach may preserve semantic information for DNA sequences in the same way observed in language text. Following this work, several other researchers have gone on to apply representation learning techniques to biological sequence data [31].

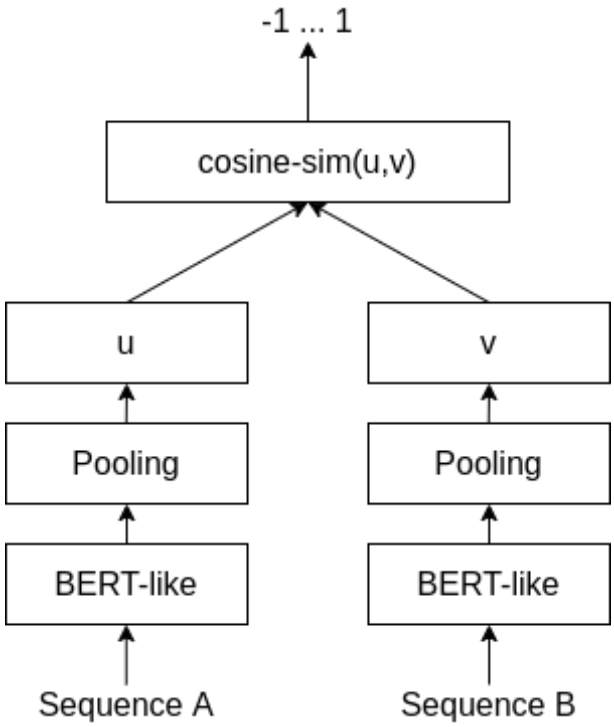

Figure 1.5: Overview of the cosine similarity sentence embedding objective proposed by Reimers and Gurevych [71]. A pre-trained bidirectional transformer has a pooling layer added after the final, d-dimensional hidden layer of the model and is loaded twice in parallel (annotated as "BERT-like" in the figure). The conjoined models are fed sentence pairs with a corresponding similarity score between (0, 1). Each sentence pair is passed through the models to produce d-dimensional vector representations, u and v. The cosine similarity between u and v is calculated, and the mean squared error (MSE) between the cosine similarity and the annotated score is calculated. The model then adjusts its weights via back-propagation gradient descent using the MSE as loss. Adapted from Reimers and Gurevych [71].

1.6.1 *k*-mer embedding

Representing DNA as k-mer count vectors or k-mer one-hot vectors are functional approaches, but these approaches have significant drawbacks, including high dimensionality of the representations and a lack of sensible distance metrics between values in the representation space. These issues are similar to those faced by natural language processing researchers when creating representations for words and text. Therefore, researchers began to apply natural language processing paradigms to biological sequence data.

Multiple researchers have now shown that k-mers can be projected to a continuous dense vector representation using neural language models [55, 49, 92]. In the specific domain of 16S marker gene analysis, Woloszynek et al. [92] trained a Word2Vec model to learn k-mer embeddings using 16S sequences as the training corpus. Their research showed that the resultant embeddings could be aggregated to create sequence level embeddings using a procedure called Smooth Inverse Frequency embedding (SIF).

SIF is a procedure proposed by Arora et al. [4] as a means of producing sentence-level dense vectors. Following the success of word embedding algorithms like Word2Vec, researchers became interested in producing dense vector representations of higher-order text units like sentences or documents. While several supervised and unsupervised learning based approaches have been created, Arora et al. [4] showed that a relatively simple, non-parametric approach could create high-quality sentence embeddings which outperformed several contemporary approaches (figure 1.6). The approach relies on a trained word-embedding model, though it is agnostic to the specifics of that model. The procedure is to first segment a collection of sequences into tokens (e.g. words for language, k-mers for DNA) (figure 1.6a). Next, use a pre-trained word embedding model to project the tokens from each sequence to a continuous vector representation (figure 1.6b). Then, take the geometric mean of the token embeddings comprising a sequence and create a matrix of the resultant vectors (figure 1.6c). Finally, calculate the first principal component of the matrix via singular value decomposition and subtract it from the matrix (figure 1.6d). The result is a matrix with columns containing vector embeddings for the sequences in the corpus.

With previous literature showing the robustness of k-mer embedding and Arora et al. [4] creating a compelling sentence-level embedding approach, Woloszynek et al. [92]

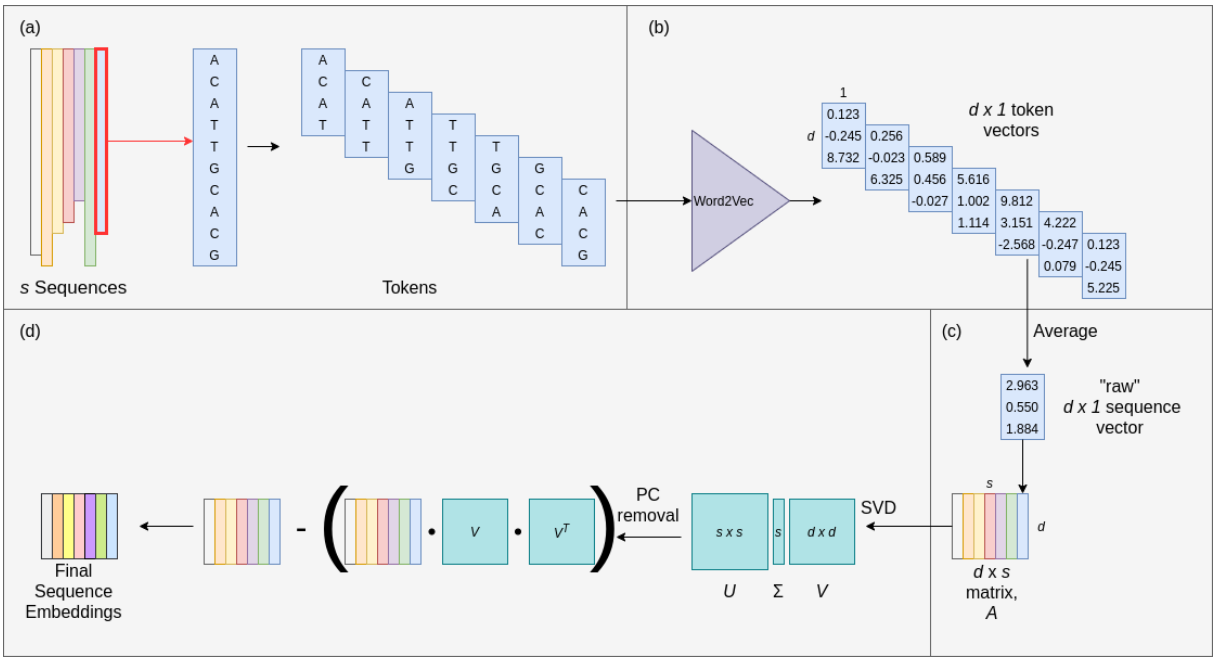

Figure 1.6: Overview of the smooth inverse-frequency (SIF) sequence embedding approach. (a) A body of s sequences is segmented into tokens using some tokenization strategy. Here, sequences are segmented into 4-mers. (b) The tokens from each sequence are embedded as d-dimensional vectors using a trained word embedding model. Here, $d = 3$. (c) The geometric mean of the token vectors comprising each sequence is taken. The resulting d-dimension vectors are used to form the columns of a matrix A. These columns represent the "raw" sequence embeddings for each sequence. (d) The matrix A is decomposed using singular value decomposition (SVD). The results of SVD are used to remove the first principal component from A. The result is a $d \times s$ matrix where each column is a d-dimensional sequence embedding.

showed that the SIF approach could be used to create dense vector representations of 16S sequences. They showed that by training a Word2Vec model to learn k-mer embeddings and then applying the SIF approach, the resultant 16S vectors could be used to classify taxonomy, perform phylogenetic clustering, and approximate cluster consensus sequences. They also showed that there was a correlation between the pairwise cosine similarity of their sequence embeddings and pairwise sequence alignment scores, suggesting that their embeddings approximate distance similar to sequence alignment metrics.

1.7 Contributions

While there has been some work investigating approaches for tokenization of DNA sequences as an alternative to k-mers, we have not found work which investigates the effects of using these sequences in downstream work flows. We train two statistically driven text encoders using a large corpus of 16S marker genes and analyze the resultant tokens. We show that our segmentation strategies produce shorter sequences than k-mer encoding with no loss of performance in sequence taxonomic assignment tasks. We also show that the tokens from our trained encoders can replace k-mers when training DNA segment vectors as per Woloszynek et al. [92], resulting in dramatically faster training time and no loss of vector quality.

While several studies have used transformer-based methods for the analysis of protein sequences, very few have applied these techniques to DNA sequence data [31]. We show that we can use encoded 16S marker genes as training data for a transformer model learning a masked language modelling task. We introduce a novel method for producing gene-level DNA sequence embeddings by introducing a fine-tuning loss function to our pre-trained transformer model that minimizes the difference between cosine similarity of output vectors and pairwise alignment scores of raw DNA sequences. We show that these embeddings outperform previously published embedding strategies in the criteria defined by their authors. Finally, we show that vector arithmetic of the resultant marker gene embeddings can be used to create continuous representations of entire microbiome samples, opening the possibility of production-ready machine learning models for those data.

1.8 Thesis outline

The remainder of this thesis is divided into three chapters. In chapter 2, we describe our approaches to training tokenizers for 16S marker gene data. We train byte-pair encoding and unigram tokenizers to create vocabularies of sizes matching k-mer vocabularies of size $k = 6$ and $k = 8$. We investigate the properties of our encoded DNA sequences and show that our encoding strategies can replace k-mers for tokenization in taxonomic classification tasks. Next, we use our encoding strategy to learn DNA token vectors using CBOW Word2Vec. We produce sequence-level vectors

using the approach described by Woloszynek et al. [92] and show that our encoding strategies can be used in place of k-mers in representation-learning problems.

In chapter 3, we describe our implementation of a transformer model trained on the masked language modelling task using 16S marker gene data as input. Subsequently, we describe how we use a novel sequence pair objective function to fine tune our transformer to produce high quality gene-level sequence embeddings. We compare our embeddings against those created by the approach described in Woloszynek et al. [92] and show that our approach results in ostensibly higher-quality vectors. Finally, we propose a novel approach for creating sample-level vector representations of microbiome data and investigate their application in the use of host phenotype classification tasks.

In chapter 4, we discuss the implications and limitations of our work and propose future research directions.

Chapter 2

Statistically Driven Segmentation of DNA Sequences

In this chapter we describe our investigation into alternative approaches to k-mer segmentation for tokenization of 16S DNA sequences. We trained BPE and unigram language model tokenization algorithms using non-redundant sequences from the SILVA database, a large database of 16S gene sequences, to learn token strategies for 16S genes. Our trained tokenizers resulted in far fewer features than did k-mer strategies. This is a major advantage for language modelling tasks, as such tasks scale linearly or worse in either compute or memory demand as a function of sequence length. We show that both BPE and unigram tokens can be used in place of k-mer tokens in sequence classification tasks with no loss in classification performance. We then show that BPE and unigram tokens can be used to learn dense vector representations of DNA "words", performing similarly to k-mers with a dramatic reduction in model training time.

2.1 Data acquisition and preprocessing

To train the Byte-Pair-Encoding and unigram tokenization algorithms, we downloaded the SILVA NR-Ref version 138.1 database of small subunit rRNA (SSU) sequences using the SILVA web portal [64]. We selected all 453,395 prokaryotic reference sequences belonging to Bacteria and Archaea, leaving out the eukaryotic reference sequences.

Some nucleotides from the DNA database are annotated as ambiguous due to low sequencing quality. We used only the unambiguous SSU sequences from the SILVA data set by removing all sequences with characters that did not refer precisely to a specific nucleotide, i.e. 'A' for Adenine, 'T' for Thymine, 'G' for Guanine, or 'C' for Cytosine. 410,078 sequences remained after filtering.

For taxonomic-prediction tasks, we used a smaller data set of 16S sequences curated by the KEGG database [36]. Note that there is a non-zero intersection between

the SILVA and KEGG data sets; some sequences appear in both sets. We do not consider there to be data set cross-contamination as the SILVA data set is used in unsupervised tasks and the KEGG set is used in supervised tasks. We retrieved 16,800 KEGG curated 16S sequences from the data repository shared by Woloszynek et al. [92], who also performed analysis on 16S sequence data. The KEGG sequences are not annotated with taxonomy, but rather with database IDs. We wrote a script to query the KEGG database RESTful API to acquire the taxonomic information for each sequence in our data set. We were able to successfully retrieve taxonomic information for 16,254 sequences, which we annotated with hierarchical labels at the levels of Kingdom, Phylum, Class, Order, Family, Genus, and Species. Some sequences are fully resolved to these taxonomic ranks and are assigned the label of *Candidatus* at the rank subsequent to their most granular resolved rank; this label refers to organisms which may be well characterized but are yet to be cultured (i.e. grown in laboratory) [77]. When performing clustering or classification, we removed targets with a label of *Candidatus*.

2.2 Byte-pair encoding

Several contemporary high-performance language models use a strategy called Byte Pair Encoding to tokenize their input sequences [19, 45, 66]. Byte Pair Encoding (BPE) is a language agnostic text compression algorithm first described by Sennrich et al. [76]. In BPE, an encoder is trained to learn a tokenization strategy given a corpus of text. The BPE algorithm is an iterative greedy algorithm which compresses text with the following strategy (detailed in algorithm 1):

1. Given a corpus S with character set C and a target vocabulary size v :
2. Initialize a vocabulary V s.t. $C \in V$
3. Parse the corpus S and determine the most-frequent pair of adjacent tokens (s_1, s_2)
4. Create a new token s_n to represent the most-frequent token pair. Append it to the vocabulary V
5. Replace all pairs (s_1, s_2) in the corpus with s_n

6. Repeat until $|V| = v$

The result of training is an encoder which can map a sequence with character set C to a sequence of tokens in the set V as well as a decoder which can map a sequence of tokens back to its corresponding set of characters. BPE has the useful property of not requiring special treatment for white space, e.g. word gaps. As there are no such delineations in DNA sequences, a DNA tokenization strategy requires this property.

A trained BPE encoder segments text by dividing the text into individual characters and then iteratively merging the characters into the tokens stored in the BPE vocabulary. The priority of token merges is driven by the order in which tokens were added to the vocabulary [76].

The deterministic and greedy nature of BPE is generally viewed as a shortcoming of the approach, as the deterministic segmentation approach can limit language models in learning morphemes [63, 39]. This shortcoming of the determinism of BPE encoding can be overcome using a simple drop-out approach: when segmenting texts, randomly prevent merge operations from occurring with a probability p [63]. By randomly preventing merge operations, stochastic text segmentation can occur. Exposing language models to these stochastically segmented texts has been shown to make them more robust [63, 11].

BPE tokenization of DNA sequences could result in properties which are favorable over the k-mer representation. A major drawback of k-mer tokenization is the property of overlapping k-mers: By using overlapping tokens, k-mer tokenization strategies result in sequences of length n being represented as lists of tokens of length $(n - k + 1)$, where typically $k << n$. Thus, k-mer tokenized sequences are approximately as long as un-tokenized DNA sequences. This can make k-mer representations prohibitively expensive for language models that use full sequence information such as the transformer, Seq2Seq, or even Word2Vec, all of which scale in compute and/or memory demand as the length of input sequences increases [85, 82, 50]. Unlike k-mer tokenization, BPE tokenization converts sequences to non-overlapping series of tokens. Indeed, at least one work that we are aware of use BPE tokenization for DNA sequences, though this processing was peripheral to their main work and they do not investigate the properties of the tokenization strategy against k-mer tokenization [98].

Shorter sequences are desirable for many sequence-analysis tasks; shorter sequences result in faster token counting for count vector methods and are more tractable for sequence modelling tasks as described above. This makes BPE appealing especially for marker gene analyses: As marker genes are characterised by highly conserved regions and BPE is designed to collapse common sub strings, it is likely that a BPE encoder trained using marker genes will result in a compression of the marker gene sequences.

Here, we investigate the efficacy of BPE encoding to replace k-mer encoding in tasks using 16S marker genes. To compare byte pair encoded sequences against k-mer encoder sequences, we trained BPE encoders using 410,078 unambiguous 16S marker gene sequences until their vocabularies reached size 4^k; that is, until the vocabulary size matched the number of possible k-mers of size k. We used Google's SentencePiece [40] library to train BPE encoders with vocabulary sizes of 4^6 and 4^8. Our choice of BPE implementation includes sub-word regularization and BPE dropout for robustness. For brevity, we will refer to these encoders as BPE6 and BPE8, respectively. To allow the encoder to compress long, conserved regions of the marker genes, we configured the BPE algorithm to allow a maximum token length of 256 (i.e. a single token can replace 256 characters), up from the default value of 8.

Algorithm 1 Train Byte-Pair Encoder

Input: A set of sequences S, final target vocabulary size v

$V \leftarrow$ set of unique characters in S

while $|V| < v$ **do**

$s_1, s_2 \leftarrow$ Tokens comprising most frequent bigram

$s_n \leftarrow s_1 + s_2$

$V \leftarrow V + s_n$

Replace all s_1, s_2 bigrams with s_n

end while

2.3 Unigram tokenization

While BPE tokenizers have seen wide adoption in the NLP field, some recent work suggests that BPE is a sub-optimal tokenization strategy [11]. This work shows that

an alternative tokenization strategy called unigram tokenization results in tokens which are more morphologically meaningful than BPE tokens. The work is focused on language modelling, and also shows that the use of tokenization strategy can affect the performance of downstream language-modelling tasks.

The unigram tokenization algorithm was proposed by Kudo [39]. Unlike BPE, which is a greedy algorithm, unigram is a maximum-likelihood model that considers multiple sub-word candidates. The unigram model begins by parsing the set of training sequences and creating a heuristic seed vocabulary V composed of the set of characters occurring in the training data and the most common sub-strings in the data, such that $|V|$ is much larger than the target vocabulary size. The algorithm iteratively ranks the sub-strings (tokens) in V and removes the lowest scoring tokens until $|V|$ is equal to the target vocabulary size (algorithm 2).

Algorithm 2 Train Unigram Language Model

Input: A set of sequences S, final target vocabulary size v, decay rate η

$V \leftarrow$ heuristic seed vocabulary: The set of unique characters in S and the most common sub-strings in S, s.t. $|V| \gg v$

while $|V| > v$ **do**

Compute $\mathcal{L} = \sum_{s=1}^{|D|} log(\sum_{x \in S(X^{(s)})} P(x))$; maximized via EM algorithm

For each token $x_i \in V$, compute l_i, the reduction of $\mathcal{L}$ if x_i is removed from V

Sort V by x_i values. Remove $(100 - \eta)\%$ tokens from V.

end while

To investigate whether unigram tokenization is better for 16S marker gene tokenization than BPE or k-mer, we trained unigram tokenizers using the same parameters as our BPE tokenizers: we trained two unigram tokenizers using 410,078 unambiguous 16S marker gene sequences with vocabulary sizes 46 and 48. Hereafter, we will refer to these models at Unigram6 and Unigram8. We set the same token length limit of 256 as we used in our BPE tokenizers.

2.4 16S Sequence Embedding

A major motivation in the decision to use BPE or unigram tokenization strategies in place of k-mer tokenization is the desire to use shorter, non-overlapping token

sequences in language modelling tasks. To our knowledge, there is only one published work investigating the projection of 16S marker genes to a continuous vector space, described by Woloszynek et al. [92].

The procedure describe by Woloszynek et al. [92] for producing continuous vector representations of 16S sequences is a two-step procedure: First, use a 16S corpus to train a Word2Vec model to produce k-mer embeddings. Next, use a procedure described by Arora et al. [4] that uses word embeddings to produce sentence embeddings and apply this to the k-mer embeddings.

The procedure described by Arora et al. [4] is referred to as Smooth Inverse Frequency (SIF) embedding and was proposed as a simple but effective baseline for the evaluation of sentence-level embedding models. SIF embedding requires the use of a trained word/token embedding model. The procedure is fairly simple (further described in algorithm 3):

1. Given a corpus S of tokenized sequences and a token embedding model W which projects tokens to a vector $v \in \mathbb{R}^d$

2. For each sequence s_i in S containing l_i tokens, apply model W to all tokens in s_i s.t. S becomes a matrix of shape $l \times d$

3. Average the token embeddings in each sequence matrix such that the result is a single d-dimensional vector for each sequence s. When calculating the average, down-weight tokens with high frequency in the corpus.

4. Place the resultant sequence vectors into a matrix M s.t. M is a $|C| \times d$ matrix.

5. Perform singular value decomposition (SVD) on the matrix M to acquire the first principal component of M.

6. Subtract the first principal component from M. The projection of the data along the first principle component is thought to encode semantic material of the entire corpus, and therefore removing this projection acts as a de-noising step for the individual sequence embeddings. The resultant matrix now contains the final sequence embeddings.

To investigate the viability of BPE and unigram representations in a language modelling context, we replicated the methods described by Woloszynek et al. [92]

using k-mer, BPE, and unigram representations. We trained Word2Vec models using the CBOW training objective and the same parameters described by Woloszynek et al. [92], implemented using the Gensim Python library [70]. We used an embedding dimension of 128, a look-ahead/look-behind window of 50 tokens, negative sampling value of 10, a high-frequency down-sampling rate of $1e^{-4}$, and a minimum word count value of 100, with all other parameters set to default.

Algorithm 3 Simple Average with PC Removal Sequence Embedding

Input: Token embeddings $\{v_k \in \mathbb{R}^d | k \in V\}$, a set of sequences S, down-weighting term a

Output: Sequence embeddings $\{v_s : s \in S\}$

$n \leftarrow$ total number of tokens in S

for all tokens $k \in V$ **do**

$f_k \leftarrow$ frequency of k in S

$p(k) \leftarrow f_k/n$

end for

for all sequences s in S **do**

$v_s \leftarrow \frac{1}{|s|}\Sigma_{k \in s} v_k \frac{a}{a+p(k)}$

end for

Form matrix X with columns $\{v_s : s \in S\}$, let u be the first singular vector of X

for all sentences s in S **do**

$v_s \leftarrow v_s - uu^T v_s$

end for

2.5 Results

2.5.1 Properties of encoded sequences

Prior to assessing the usefulness of BPE and unigram encoding strategies in downstream applications, we encoded our training corpus of 16S marker genes using BPE, unigram, and k-mer approaches and investigated the distribution properties of tokens in the various encoding strategies. We investigate the effect of encoding strategy on sequence length, token frequency distributions, and feature importance rankings. We found that using BPE or unigram encoding resulted in dramatically shorter sequences

than k-mer encoding. We also found that token distributions and feature importance rankings changed depending on which tokenization strategy is used, suggesting that researchers may wish to adjust their analyses if switching from one strategy to another.

Compression rate of tokens

To approximate the compression rate of the two tokenization strategies, we investigated the distribution of token lengths in the trained encoders' vocabularies. Each encoder has an upper bound on the number of characters a single token can represent of 256. We refer to the number of characters encoded by a single token as that token's token length. For each possible token length $i \in [0, 256]$, we counted the number of tokens in each encoder's vocabulary with length i (figure 2.1). We found that for all encoders, the majority of tokens have a token length $i < 50$. Interestingly, we found that the most common token length for BPE6 and BPE8 are 6 and 8 respectively, which matches the token length of k in their defined vocabulary sizes of 4^k. We do not see this with unigram encoding; both Unigram6 and Unigram8 have a most-common token length of 6.

We found that with both BPE and unigram encoding, a larger vocabulary size allows for larger token lengths to occur in the vocabulary. This matches our expectations; we would expect frequent strings of length l_i to be less frequent than sub-strings found within those strings. Using a larger vocabulary size allows the encoder to add longer strings to the token vocabulary.

We found that BPE6 and Unigram6 learned tokens of similar lengths, having similar distributions of token lengths in their vocabularies. However, we observed that Unigram8 contains far more long tokens than BPE8 (red plots, figure 2.1), suggesting that the maximum-likelihood model used in the unigram tokenization process is more likely to use large tokens in its vocabulary than the strictly greedy approach used by BPE. We also note a spike of the number of tokens with a length of 256 in Unigram8's vocabulary, which is the upper bound. This may indicate that Unigram8 trained with a larger maximum token size would add still larger tokens to its vocabulary.

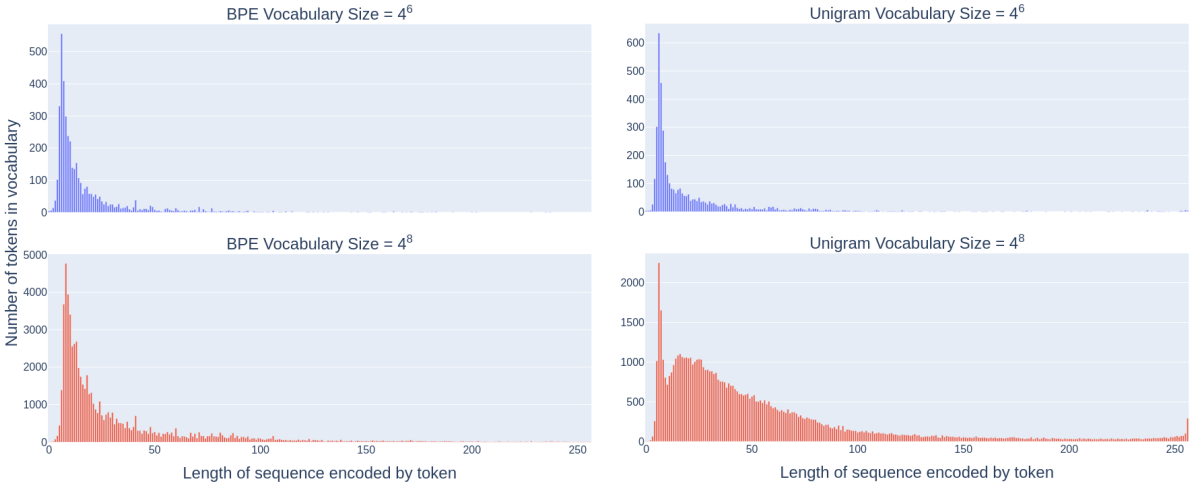

Figure 2.1: Distribution of token lengths as they appear in encoders' vocabularies. x-axis indicates the number of characters represented by a token, i.e. the length of a token. y-axis indicates the number of tokens in the encoder vocabulary with a given token length. Unigram models are represented in the right hand column. BPE models are represented in the left hand column.

Distribution of tokens

Next, we investigated the distribution of token lengths that appear in the set of 410,078 training sequences after tokenization (figure 2.2). When inspecting the *in situ* distributions of token lengths, we find that the apparent differences in token length distributions among models seen in their vocabulary distributions is no longer apparent. BPE6, BPE8, and Unigram8 all have most frequent token sizes of 5 and 6. Curiously, Unigram6 has a most frequent token size of 1, representing single nucleotide tokens.

We find that in spite of BPE8 having far fewer tokens with length > 100 than Unigram8 in its vocabulary (figure 2.1), the distribution of large tokens in real data occurred with similar frequency (figure 2.2, red plots). Thus, in spite of BPE and Unigram encoders producing differently distributed vocabularies, their encoding in practice resulted in similar token distributions. This discrepancy can be explained by the difference in vocabulary building strategies used by BPE and unigram encoders. Where BPE is purely greedy and frequency based, it is unlikely to maintain large common strings, especially as the vocabulary size becomes smaller. In the context of DNA sequence data where strings are contiguous, for any string s with length $l_s > 1$

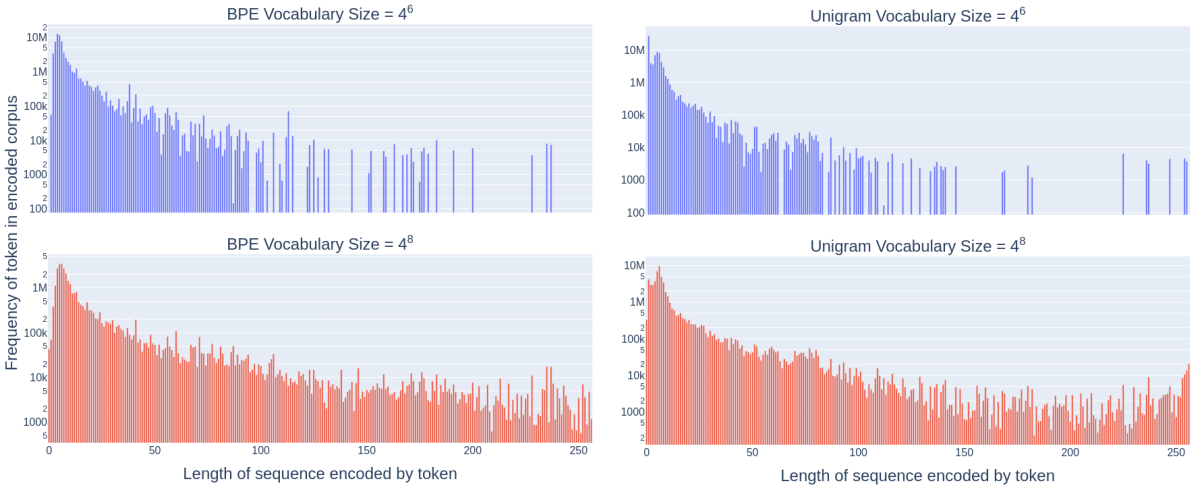

Figure 2.2: Distribution of token frequencies using various token encoding methods, binned by encoded token length. Tokens are counted over 410,078 full length 16S sequences retrieved from SILVA [64]. Top row shows tokenizers trained to a vocabulary size of 4096 tokens. Bottom row shows tokenizers trained to a vocabulary of 65536 tokens. Left column shows tokenizers trained using the BPE algorithm. Right column shows tokenizers trained using the Unigram Language Model algorithm. Note that the y-axis has a log scale.

that occurs with frequency k in a corpus, it is extremely likely that there is some sub-string of s that occurs with frequency $f > k$. Therefore, large strings will be ousted from the limited vocabulary by their more frequent sub-strings. Unigram, however, uses a loss function approach to score strings when deciding whether to keep them in the vocabulary, so it is not strictly necessary that long strings with more frequent sub-strings will be discarded. In practice, however, both methods prefer using relatively short tokens to encode sequences.

Similar to what we observed when inspecting vocabulary distributions, we note that BPE6 and Unigram6 have far fewer large tokens than BPE8 and Unigram8. It is apparent that larger vocabulary sizes allow for the addition of larger tokens to the vocabulary, resulting in increased compression of sequences. Certain applications require increased compute and memory for increased sequence length, while others do so for increased vocabulary size, and the best choice of vocabulary size may be use-case dependent.

Inverse rank frequency of tokens

We investigated the relationship between token frequency in the corpus and token rank. Token rank refers to the index of a token in a list of tokens sorted descending by their frequency of occurrence. We investigated whether any or all of the tokenization strategies (BPE, Unigram, k-mer) produced approximately log-linear inverse rank frequency plots, which is a characteristic widely observed in computational linguistics data sets [61]. We encoded our large 16S corpus using BPE, unigram, and k-mer representations with vocabulary sizes $\{4^6, 4^8\}$. Next, we calculated the frequency of all tokens in each encoding strategy, sorted by descending frequency, and plotted frequency vs rank on a log-log plot (figure 2.3). We performed simple linear regression on the log-log plots and used the resultant R^2 value as a measure of linearity.

We find that both choice of vocabulary size and tokenization strategy affect the frequency-rank distributions of tokens. Of the three representations, Unigram tokens resulted in the closest to linear inverse rank frequency. We note that for k-mer and unigram encoding strategies, a larger vocabulary size resulted in closer to log-linear inverse rank frequency distributions, whereas for BPE the converse was true. We notice that the frequency of tokens was more concentrated in BPE and Unigram tokenization strategies than in k-mer strategies (figure 2.3). Based on the variance between frequency distributions of the models, we suggest that parameterized, frequency-driven analysis methods may need to be modified based on the representation used.

Length of encoded sequences

We investigated the resultant sequence length from encoding 16S sequences with our tokenization methods. Again, we tokenize our SILVA data set of sequences using BPE, Unigram, and k-mer representations with vocabulary sizes of $4^k, k \in 6, 8$. We recorded the length (i.e. number of tokens) of all encoded sequences and plotted them (figure 2.4). The average length of a 16S sequence is ≈ 1500 nucleotides. As discussed in chapter 1, the length of a k-mer encoded DNA sequence with n nucleotide will always be $(n-k+1), k << n$, so k-mer encoded sequences will be approximately equal in length to the raw sequences. The average 16S sequence was ≈ 1500, with some significant variability in the data set.

Both unigram and BPE encoding reduced the number of tokens in a sequence

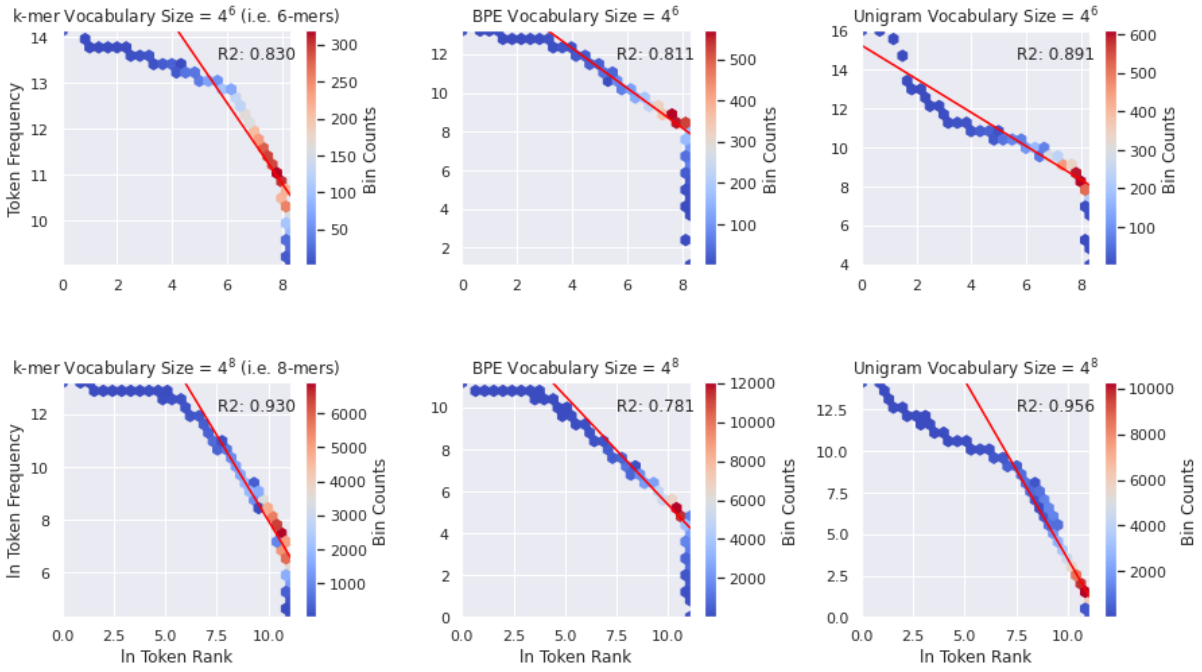

Figure 2.3: Inverse rank-frequency plots for token frequencies in SILVA data set. Columns show k-mer, BPE, and unigram representation. Rows show vocabulary sizes of 4^6 and 4^8. Frequency values are hex-binned for legibility. The x-axis shows the log value of the token rank. The y-axis shows the log value of the token frequency in the corpus.

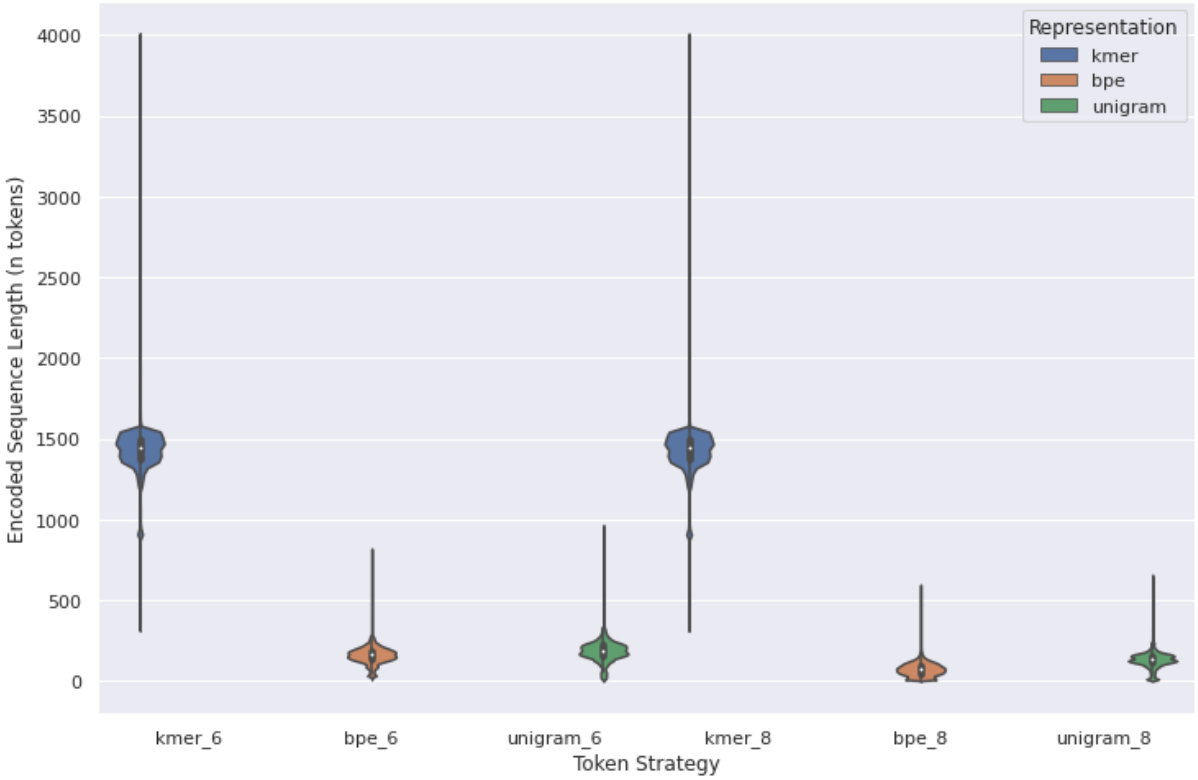

Figure 2.4: Length distributions in number of tokens of 16S sequences encoded using various tokenization strategies. Distributions were measured by measuring the number of tokens in each tokenized sequence from the 410,087 SILVA 16S sequences. y-axis indicates the number of tokens in the encoded sequence.

by approximately 7-fold compared to k-mer methods. As expected from our results discussed above, BPE8 and Unigram8 produced shorter sequences (i.e. fewer tokens) than their BPE6 and Unigram6 counterparts. Interestingly, despite unigram encoding resulting in vocabularies containing more large tokens than BPE, BPE representations resulted in shorter sequences. This is likely due to the high frequency of very short tokens found in unigram encoding (figure 2.2).

2.5.2 Machine-learning assignment of taxonomy

It is common practice to use machine-learning classifiers to assign taxonomic labels to 16S sequences, often using some k-mer derived representation as features [8, 33, 89]. To assess the efficacy and quality of BPE and unigram tokens in 16S segmentation, we implemented machine-learning models to classify 16S sequences using inputs derived

from k-mer, BPE, and unigram tokens and compared the performance of each model.

Experimental Design

As a feature representation, we used a count-vector representation, whereby given a sequence s and vocabulary of tokens $V, |V| = d$, the sequence s is represented as a vector $v \in \mathbb{Z}^d$ where $v[i]$ is the number of times the ith token in V occurs in s. We justify our choice to use count vectors as research has shown that count vectors of k-mers are effective for classification of 16S sequences [33]. We created count vectors using k-mer tokens, BPE tokens, and unigram tokens.

Taxonomic ranks are assigned hierarchically. Therefore, we formulated 3 classification tasks: Given a count vector of a sequence s, predict that sequence's taxonomic assignment at the ranks of Phylum, Order, and Genus (i.e. 3 classification problems with increasing granularity). For our classification tasks, we used our 16,254 KEGG database 16S gene sequences.

We used the SciKit-learn machine learning library [58] to implement two machine-learning models to predict a sequence's taxonomic class label given the count vector embedding as input. We produced count vectors using *6*-mer, BPE6, and Unigram6 representations. We implemented a Multinomial Naïve Bayes classifier and a Random Forest (RF) classifier, each using the SciKit learn default hyperparameters.

We also investigated the effect of feature selection on classification performance. We implemented statistical univariate feature selection using ANOVA F-score to select the n best features with $n \in 1024, 512, 256, 128$. We also implemented a tree-based stochastic mutual information feature selection strategy, selecting the n best features from a RF model based on impurity with $n \in 1024, 512, 256, 128$. When using RF feature selection and RF classification in the same trial, we used different models for feature selection and classification.

We performed machine-learning trials using each machine learning model, each feature selection strategy, and each count vector embedding (BPE6, Unigram6, 6-mer) (figure 2.5). For our classification tasks, we used seeded train-test splits with a test set size of 20%. We measured the success of each permutation by measuring the model's weighted F1 score over the test set. We calculated the F1 score as $F1 = 2\frac{P*R}{P+R}$ where P and R are precision and recall, respectively. Precision is defined

as $P = \frac{T_p}{T_p+F_p}$, where T_p and F_p are the number of true positive and false positive predictions in a test set, respectively. Recall is defined as $R = \frac{T_P}{T_p+F_n}$, where F_n is the number of false negative predictions in the test set.

Classification results

There were no significant differences in the best results for each feature representation (i.e. tokenization strategy). However, there were major differences among performances of models using the different representations (figure 2.5). Of note, univariate feature selection resulted in significant degradation of model performance for unigram and BPE representations, but not for 6-mer representation. In the genus classification task, for example, BPE token representations classified with RF classifiers achieved an F1 score of 0.980 using 128 features selected via RF mutual information, but only achieved an F1 score of 0.307 when using 128 features selected using ANOVA F score. The unigram representations resulted in nearly identical scores. 6-mer representations, however, had almost no change in score, achieving 0.988 with RF feature selection and 0.986 with ANOVA feature selection.

Naïve Bayes classifiers performed worse than RF classifiers in every trial. This difference was more pronounced when using BPE or unigram tokens than with k-mer tokens. For example, the best RF classifier for BPE-encoded sequences in genus prediction, using 1024 features, achieved an F score of 0.990, whereas the best Naïve Bayes classifier for the same achieved 0.943, with a similar difference in results for unigram. With k-mer tokens, the difference in performance between RF and Naïve Bayes performance was near constant, with RF achieving 0.01 - 0.04 higher F scores in all trials. For BPE and unigram, however, the difference in performance became more pronounced with smaller feature sets. Using 128 features selected via RF, BPE-trained RF classifiers achieved an F1 score of 0.980, whereas the Naïve Bayes classifier trained with the same features achieved a score of only 0.771.

Overall, we found that all the highest scoring models used RF classifiers with separate RF mutual information feature selection. While token strategy choice did not affect the highest scores of the models, the 6-mer representation was more robust to feature selection with small n features and to model choice.

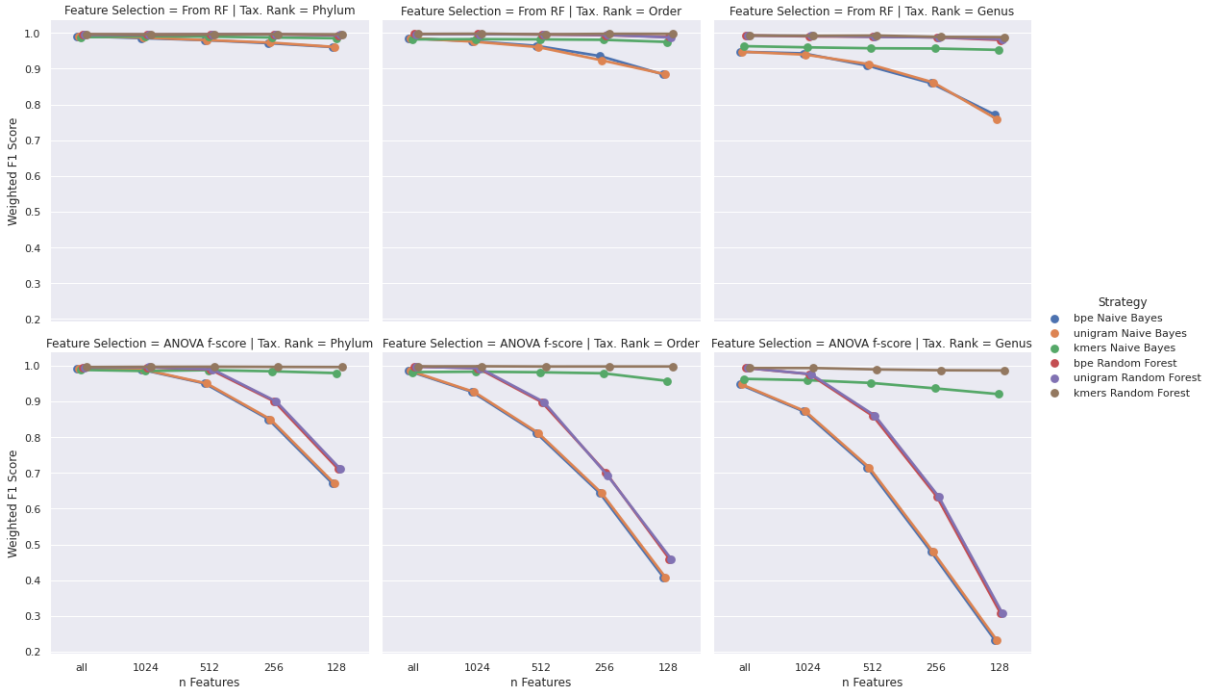

Figure 2.5: Performance of models assigning taxonomic labels to KEGG database 16S sequences. Sequences were encoded as token count vectors using different tokenization strategies (k-mer, BPE, Unigram). All tokenization strategies used a vocabulary size of 4096. Our trials compared the performance impacts of statistical univariate feature selection using ANOVA F-score and stochastic mutual information feature selection using random forest. We repeated our experiments using both random forest and Naïve Bayes classifiers. The top row shows results using random forest feature selection. The bottom row shows results using ANOVA F-score feature selection. Columns show performance of models at different taxonomic ranks, with increasing granularity from left to right.

Feature Correlations

To learn more about the apparent differences between tokenization methods, we investigated the internal correlation structure of features selected using our two feature selection strategies and our three tokenization strategies. We chose to investigate results from the genus classification task, as these results showed the greatest discrepancy among methods (figure 2.5). We calculated the Pearson correlation co-efficient among the 128 selected features for each feature selection strategy and each token count representation (figure 2.6).

For both feature-selection strategies, 6-mer token features have a higher rate of intra-correlation than do unigram or BPE features. This is an intuitive finding: due to the overlapping nature of k-mers, it is natural to expect them to be correlated with one another. When using ANOVA F score univariate feature selection, BPE and unigram features had very little intra-feature correlation and almost no negative intra-feature correlation. The features selected by RF mutual information had more intra-feature correlation than did ANOVA F score for BPE and unigram tokens. It appears that some internal correlation structure is important to the success of classifiers using these token representations: ANOVA feature selection with BPE and unigram resulted in features with little internal correlation and produced a classification F1 score of $\approx$0.30 for both token strategies, whereas RF feature selection resulted in a higher magnitude internal correlation and produced a classification F1 score of $\approx$0.98.

To further investigate the differences between feature importance and correlations among features, we performed ordination analysis on the 128 features selected from each feature selection algorithm for each tokenization strategy in the genus classification task. We performed principal coordinates analysis (PCoA) over the correlation matrices of each feature set, calculating the eigenvectors and eigenvalues of each set. We ranked the principal co-ordinates by their percentage of variance explained and plotted the top 20 principal co-ordinates (figure 2.7). We found that k-mer representations had a large amount of variance explained by the first principal co-ordinate, with 28.0% from the RF selected features and 45.4% with the ANOVA F-score selected features. In comparison, BPE's first principal co-ordinate explained only 7.59% of variance with RF selected features and 3.19% with ANOVA selected features. Unigram's first principal co-ordinate explained 6.31% of variance with RF feature selection and

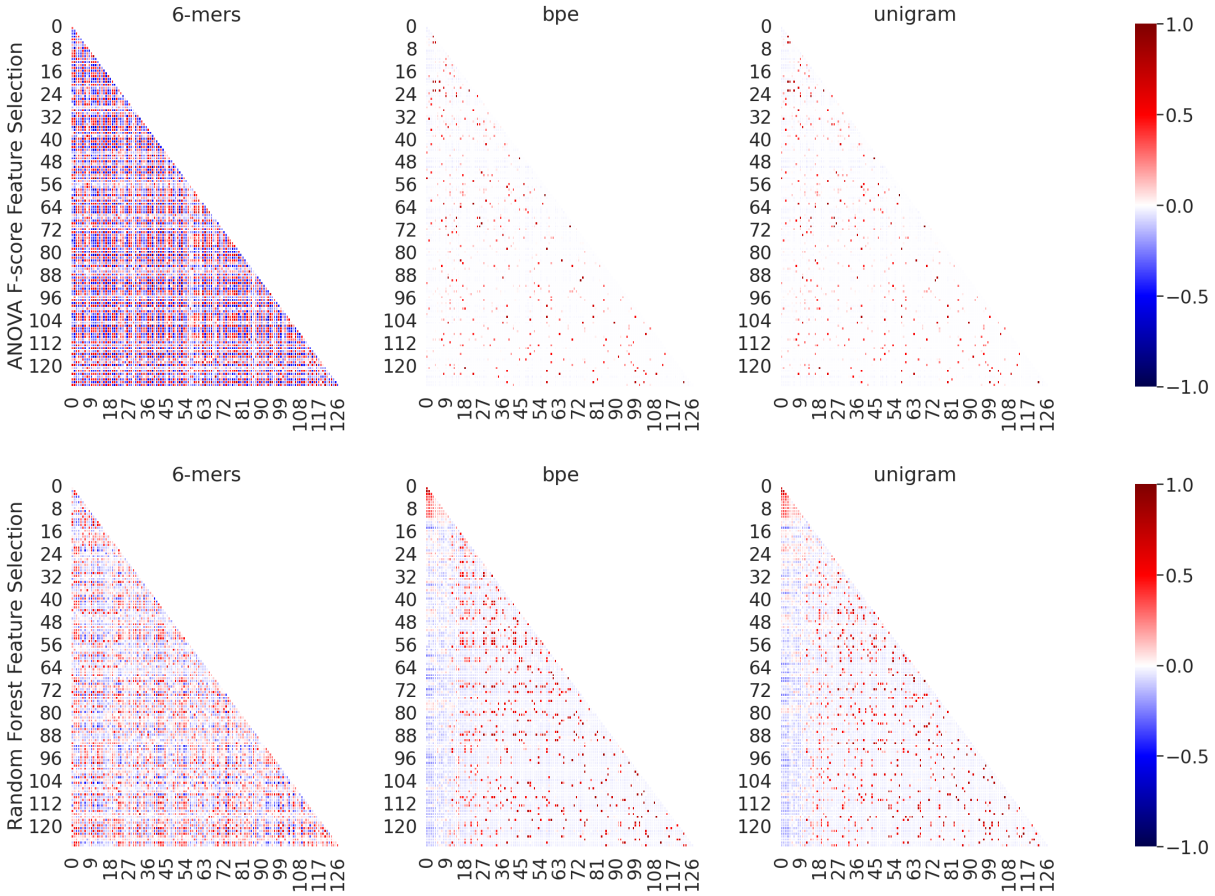

Figure 2.6: Pearson correlation of 128 most important features selected by univariate feature selection and stochastic mutual information feature selection in the 16S genus classification task using multiple tokenization strategies. Each panel shows the pairwise Pearson correlation coefficient of the features selected by a given algorithm for a given tokenization strategy. Top row shows univariate feature selection by ANOVA F-score. Bottom row shows stochastic mutual information feature selection by RF. Columns show results from tokenization strategy of k-mer, BPE, and unigram, from left to right. Strong negative correlations are shown as dark blue, strong positive correlations are shown as dark red, and no correlation is shown as white.

Figure 2.7: Scree plot showing the proportion of variance explained by the top 20 principal coordinates extracted from the 128 most important features selected by univariate feature selection and stochastic mutual information feature selection in 16S genus classification task using multiple tokenization strategies. The top row shows univariate feature selection by ANOVA F-score. The bottom row shows stochastic mutual information feature selection by random forest. Columns show results from tokenization strategy of k-mer, BPE, and Unigram, from left to right.

3.20% with ANOVA selected features. We found that stochastic mutual information feature selection produced more significant individual principal co-ordinates for BPE and unigram than statistical univariate feature selection, whereas univariate statistical feature selection produced more significant individual principal co-ordinates for k-mers.

2.5.3 Sequence embedding assessment

As described in section 2.4, Woloszynek et al. [92] proposed a method for learning dense vector representations of 16S sequences which involved first training a word embedding model using DNA k-mers and then aggregating the vectors comprising the k-mers in a sequence via the SIF embedding procedure. We performed sequence embedding with this method using 6-mers, BPE6 tokens, and Unigram6 tokens to

train the word embedding model. Of note, we observed an approximately 10x speed-up of training time for BPE and Unigram representations vs k-mer representations as a result of the shorter sequences produced by these strategies (table 2.2).

To compare the quality of sequence embeddings that result from each token representation, we embedded the 16,254 KEGG database 16S sequences that had labels available. We then perform classification and clustering experiments with the embedded sequences to determine their utility in these tasks.

Classification of embedded sequences

To compare the different sequence embeddings, we trained a RF classifier to predict the genus of the sequence using the sequence embedding as input. The SIF procedure has a user-specified hyperparameter, a, which down-weights the vector embeddings of more frequent tokens (3). We tested the effects of choice of a value by repeating the sequence embedding procedure using $a \in \{$None, 100, 10, 1, 0.1, 0.001, 0.00001, 0.0000001, 0.000000001$\}$ using 6-mers, BPE6, and Unigram6. Where $a =$ None, we did not perform the a term normalization, instead directly averaging the vector embeddings of tokens. We then trained a RF classifier to predict the genus of a sequence using the final sequence embedding as input. We performed 5-fold cross validation and reported the mean accuracy (figure 2.8).

BPE and k-mer representations achieved ≈ 0.01 higher mean accuracy than the unigram representation. There was no consistent trend in the effects of the down-weighting parameter a; each representation showed fluctuations within 1% mean classification accuracy with changing a value. As the choice of a value does not significantly affect model performance using the resulting representation, we concluded that token down-weighting was unimportant for our purposes. Note that Woloszynek et al. [92] reported improvements in Adjusted Rand Index (ARI) clustering metrics with $a = 0.00001$, but did not report effect size.

Clustering of embedded sequences

To further compare the quality of sequence embeddings, we performed K-means clustering on the embedded sequences, modifying K to reflect different hierarchical taxonomic assignments. For each target of {Kingdom, Phylum, Class, Order, Family,

Figure 2.8: Mean accuracy of random forest classification of genus using dense vector embeddings of DNA sequences using different tokenization strategies and parameters. "a value" indicates the value of the a term in the modified SIF embedding procedure proposed by Woloszynek et al. [92]. y-axis indicates the mean validation accuracy from 5-fold cross validation trials, with error bars showing the standard deviation of the accuracies. Note that the y-axis is truncated to emphasize error bars.

Genus}, we performed K-means with K set to the number of labels of each target found in the KEGG data set.

We assessed the quality of clusters using four metrics: homogeneity, completeness, ARI, and adjusted mutual information (table 2.1). Each score is bound between $[0.0, 1.0]$, with a larger value indicating a higher score. Homogeneity is a measure of the proportion of instances in a cluster that are a measure of a single class, with a score of 1.0 being assigned if all instances in all clusters are of the same class. Completeness measures the proportion of data points in a single class that are assigned to a single cluster, with a score of 1.0 being assigned if all data points from each class are assigned to the same clusters. The Rand Index is a metric for assessing the quality of clusters that is calculated by considering all pairs of samples and comparing the true labels and predicted labels of the samples. It is calculated as:

$$RI = \frac{(n \text{ agreeing pairs})}{n \text{ pairs}}$$

The Adjusted RI modifies the RI score such that randomly assigned labels will result in a score close to 0.0. Given the above definition of RI, ARI is defined as:

$$ARI = \frac{(RI - \text{Expected}RI)}{max(RI) - \text{Expected}RI}$$

The mutual information score measures the similarity between two labellings of the same data; in this context, between the predicted cluster labels and the true cluster labels. If $|U_i|$ is the number of samples in cluster U and $|V_j|$ is the number of samples in cluster $|V|$, then the mutual information score between U and V is given as:

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

Similar to the ARI, the adjusted mutual information score (AMI) modifies the MI score to account for random chance. The AMI adjustment is more complex than the ARI adjustment and requires further definitions. Given a set of samples S clustered to U, the probability that a random sample selected from S will be placed in cluster U_i is

$$P_U(i) = \frac{|U_i|}{N}$$

and the entropy of U is then

$$- \sum_{i=1}^{n \text{ clusters}} P_U(i) \log P_U(i)$$

Given these definitions and the definition of MI, AMI is then defined as:

$$AMI(U,V) = \frac{MI(U,V) - \text{Expected}(MI(U,V))}{avg(H(U),H(V)) - \text{Expected}(MI(U,V))}$$

Since Woloszynek et al. [92] reported using ARI to assess their choice of a value during sequence embedding, we performed preliminary investigations into the effect of a value on ARI. Modifying the a value during sequence embedding resulted in inconsistent fluctuations on the order of magnitude of 0.001 for subsequent ARI scores. Coupled with our findings in section 2.5.3, we therefore decided that token down-weighting was unimportant for our purposes and performed our detailed cluster analysis using sequences embedded without the use of down-weighting.

The clustering metrics among the representations were similar, with k-mer method clusters achieving scores 0.0 - 0.05 higher in most metrics than the nearest trained token method, with the exception of the ARI scores for family and genus, in which the k-mer method outperformed the next best methods by 0.10 and 0.12, respectively (table 2.1). There was a general trend of BPE method clusters achieving metric values approximately 0.1 higher than unigram, with some exceptions. For every metric and every feature representation, we observe that metrics improve with finer resolution taxonomic groupings.

2.6 Conclusions

Given the results of our experiments, we believe that BPE tokenization and unigram language modelling are viable representation strategies for for marker-gene analysis.

BPE and unigram representations have major advantages over k-mers in applications which scale in computational complexity with the length of sequences, as the length of BPE or unigram encoded token sequences is nearly an order of magnitude shorter than k-mer tokenizations. We observed such an advantage when training our Word2Vec models, noting a ≈10x speedup in absolute run time when training Word2Vec models using BPE and unigram sequences vs k-mer sequences 2.2. Shorter sequences also result in faster counting operations when creating count vectors or presence/absence vectors, as such representations require parsing the tokens in a sequence to create the counts.

Table 2.1: Clustering metrics for SIF embeddings calculated using different tokenization strategies at different taxonomic ranks. We performed K-means clustering for each taxonomic rank, setting K to equal the number of distinct classes at each rank. We clustered sequences at the ranks of Kingdom, Phylum, Class, Order, Family, and Genus. Each tokenization strategy had a vocabulary size of 4096. ARI indicates the adjusted Rand Index. AMI indicates the Adjusted Mutual Information metric.

Rank		K	P	C	O	F	G
n Classes		2	36	99	198	428	666
Homogeneity	k-mer	0.100	0.852	0.917	0.940	0.954	0.961
	BPE	0.101	0.813	0.906	0.929	0.945	0.954
	Unigram	0.062	0.826	0.889	0.914	0.934	0.944
Completeness	k-mer	0.013	0.368	0.501	0.637	0.746	0.813
	BPE	0.016	0.340	0.485	0.609	0.712	0.774
	Unigram	0.012	0.348	0.473	0.597	0.698	0.758
ARI	k-mer	-0.013	0.129	0.124	0.199	0.307	0.382
	BPE	-0.016	0.090	0.100	0.142	0.206	0.261
	Unigram	-0.027	0.097	0.088	0.131	0.185	0.233
AMI	k-mer	0.028	0.508	0.631	0.732	0.792	0.822
	BPE	0.028	0.473	0.614	0.704	0.756	0.778
	Unigram	0.020	0.483	0.599	0.689	0.738	0.755

Table 2.2: Time in seconds to complete 5 epochs of Word2Vec training using different tokenization strategies. Models were trained using an Advanced Micro Devices Ryzen 7 2700X CPU with 16 threads.

Token Strategy	Runtime (s)
k-mer	1763.42
BPE	188.79
Unigram	192.49

In addition to scaling with sequence length, certain applications scale with the size of the vocabulary, including count or presence/absence approaches and transformer-based language modelling [19, 45, 85]. BPE and unigram representations each offer the benefit of allowing user defined vocabulary sizes, allowing far more flexibility in designing vocabulary size dependant analyses than k-mer representations, which require vocabulary sizes to be 4^k.

We find that BPE and unigram can each be used for direct classification and representation approaches with little or no loss in performance compared with k-mers. However, our findings suggest that BPE and unigram representations are more sensitive to model choice and have less-correlated features than k-mer representations. The latter is an intuitive finding; due to the overlapping nature of k-mers it is natural that they would be highly correlated. The former is less intuitive and suggests that parameterized models which perform well with k-mers will likely require different parameter choices to perform well with BPE or unigram representations.

Our trials with feature selection suggest that while BPE and unigram token models can match k-mer representations in various applications, they will likely require more features to do so. This finding is corroborated by our findings using progressively stricter feature selection as well as by our ordination analysis — our PCoA analysis suggests that BPE and unigram representations require more features to explain the variance in a data set.

We find that BPE and unigram models are both viable, but there are important differences between the two and we do not consider them interchangeable. Unigram representations appear to be more sensitive to parameter choices than BPE representations, as seen in our sequence embedding, classification, and ordination experiments. These differences can likely be attributed to the differences in token distributions between the two models. Unigram-based approaches produce a very large number of tokens representing single characters, a phenomenon that does not occur in our BPE models. Single-character tokenization tends to result in poor conservation of information in DNA sequences, which has partially motivated the use of k-mer representations. We therefore believe that the high incidence of single-token characters as tokens in our unigram representations is a plausible explanation for those representations being more sensitive to parameter choices in downstream tasks.

Given that both BPE and unigram perform well in all downstream applications that we tested, we believe each approach to be viable for replacing k-mer representation in marker gene analysis tasks. However, our findings show that BPE strategies result in shorter sequences with preference to longer tokens which are more robust to parameter choice in down stream applications. Therefore, we suggest that researchers should consider using BPE tokenization to replace k-mer tokenization in their DNA sequence modelling.

Chapter 3

Bi-Directional Transformers Produce High-Quality Sequence Embeddings with Multiple Use Cases

In the previous chapter, we investigated the viability of BPE and unigram tokenized encodings of 16S marker genes in the application of sequence embeddings. To our knowledge, the only previously published approach to projecting 16S sequences onto a continuous vector space is the approach published by Woloszynek et al. [92], which maps tokens in a sequence s to a continuous vector space $\mathbb{R}^d$ and then aggregates the resultant vectors to represent a sequence (algorithm 3). While this approach proved effective, contemporary transformer-based models make it possible to pre-train a model to learn the "language" of a corpus of sequences [91, 19, 45, 98, 41] and subsequently fine-tune that model to produce high quality sequence-level vector embeddings which preserve semantic information from the raw texts [71].

There has been some previous work applying transformers to biological sequences [31]. These models have been applied to application areas such as protein tertiary structure prediction, protein sub-cellular localization prediction, bacterial transcription start site annotation, and human genome annotation [73, 22, 68, 17, 32, 98]. As described in chapter 1, each of these papers follows the paradigm of pre-training a large transformer architecture ("language" modelling of the biological sequence corpora) followed by fine tuning the model for some supervised learning objective.

Here, we pre-train a transformer language model to learn to model prokaryotic small subunit rRNA genes (16S marker genes) and subsequently fine tune our model to produce continuous vector fixed-length representations of 16S marker genes. To our knowledge, we are the first to apply transformer language modelling to the task of marker gene vector embedding. First, we trained a RoBERTa style transformer architecture [45] using our corpus of 410,078 16S marker genes from the SILVA database [64]. Then, we created a corpus of annotated sequence pairs by calculating the pairwise alignment scores of 16,254 16S sequences from the KEGG database

using the VSEARCH algorithm [36, 75]. We used a sub-sampling of the resulting 132,088,131 sequence pairs to fine tune our model to produce sequence level vector embeddings using the procedure described by Reimers and Gurevych [71]. We compared the quality of our sequence embedding approach against the approach used in chapter 2 as described by Woloszynek et al. [92] using clustering metrics and taxonomic performance. We show that our approach's sequence vector embeddings can be used to perform a GPU accelerated similarity search. We selected 1000 random sequences from our KEGG 16S data set and use BLAST+ [14] to identify the closest 5 sequences from our SILVA data set for each sequence. We then performed sequence embedding and similarity search using the same sequences and found that searching the vector space can approximate local alignment search in a small fraction of compute time. Finally, we investigate the application of our trained embedding model to short-read microbiome data, producing sample-level embeddings of human stool microbiota and using the resultant vectors to classify the host phenotype.

3.1 Masked Language Model

The premise of this chapter is that transformer models pre-trained on language modelling tasks can subsequently be fine-tuned for task specific purposes in several areas, including when using biological sequence data [91, 19, 71, 45, 17, 32, 73, 22, 68, 31]. Therefore, we propose that a transformer specifically trained to model 16S gene sequences can subsequently be fine-tuned to produce high quality 16S sequence embeddings.

The notion of a language model is defined by the task of the model during learning. Several contemporary language models have seen great success with a training objective known as masked language modelling (MLM) [19, 42, 45, 91]. The MLM objective is simple: Models are fed segments of text as training data in which tokens in the text are randomly ablated and replaced with a "mask" token. The model is tasked with predicting the identities of the masked tokens.

3.1.1 Transformer model architecture

As discussed earlier, the transformer is an encoder-decoder model. The general premise of an encoder-decoder architecture is that inputs are passed through some

"encoder" model architecture such that an encoded state is produced. The resultant state of is then passed to the "decoder" model, which maps the state vector to some output with fixed shape.

The Transformer family of architectures is characterized by stacks of layers containing a multi-headed self attention layer followed by a fully connected feed-forward layer in both encoder and decoder [85]. Canonical transformer architectures perform language modelling by using the states of tokens $i_0, ...i_{n-1}$ to predict token i_n. Here, we use bidirectional transformers pre-trained using the masked-language modeling task, as bidirectional pre-training has been shown to improve the robustness of transformer-based language model [19, 45]. Unlike canonical or "left-to-right" transformer architectures, bidirectional transformers consist only of the transformer encoder, which is used to predict the identities of masked tokens in the masked language-modelling task (figure 3.1). As discussed in chapter 1, the transformer encoder comprises a series of N stacks, where each stack consists of a multi-headed self-attention layer followed by a fully connected feed-forward layer. In the process of learning the masked-language modelling task, the model learns context-dependant, d-dimensional vector embeddings for each token in the vocabulary used to train the model.

The mechanism which allows transformer models to model intra-sequence dependencies without recurrence or convolution is the attention mechanism coupled with positional encoding. In general, attention describes a mapping of query, key, and value vectors to some output (referred to as Q, K, and V for brevity from here). There are multiple ways to compute attention, but transformer architectures use scaled dot-product attention, computed as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the cardinality of the key vector [85].

Transformer is also characterized by the use of multi-headed attention, which allows the model to attend to sequence dependencies at different ranges. Multi-head attention is a procedure in which the vectors Q, K, and V, where $X \in Q, K, V, |X| = d_x$ are linearly projected h times to projections with dimensions d_q, d_k, d_v in parallel using different, learned projections over which the attention operation is applied in

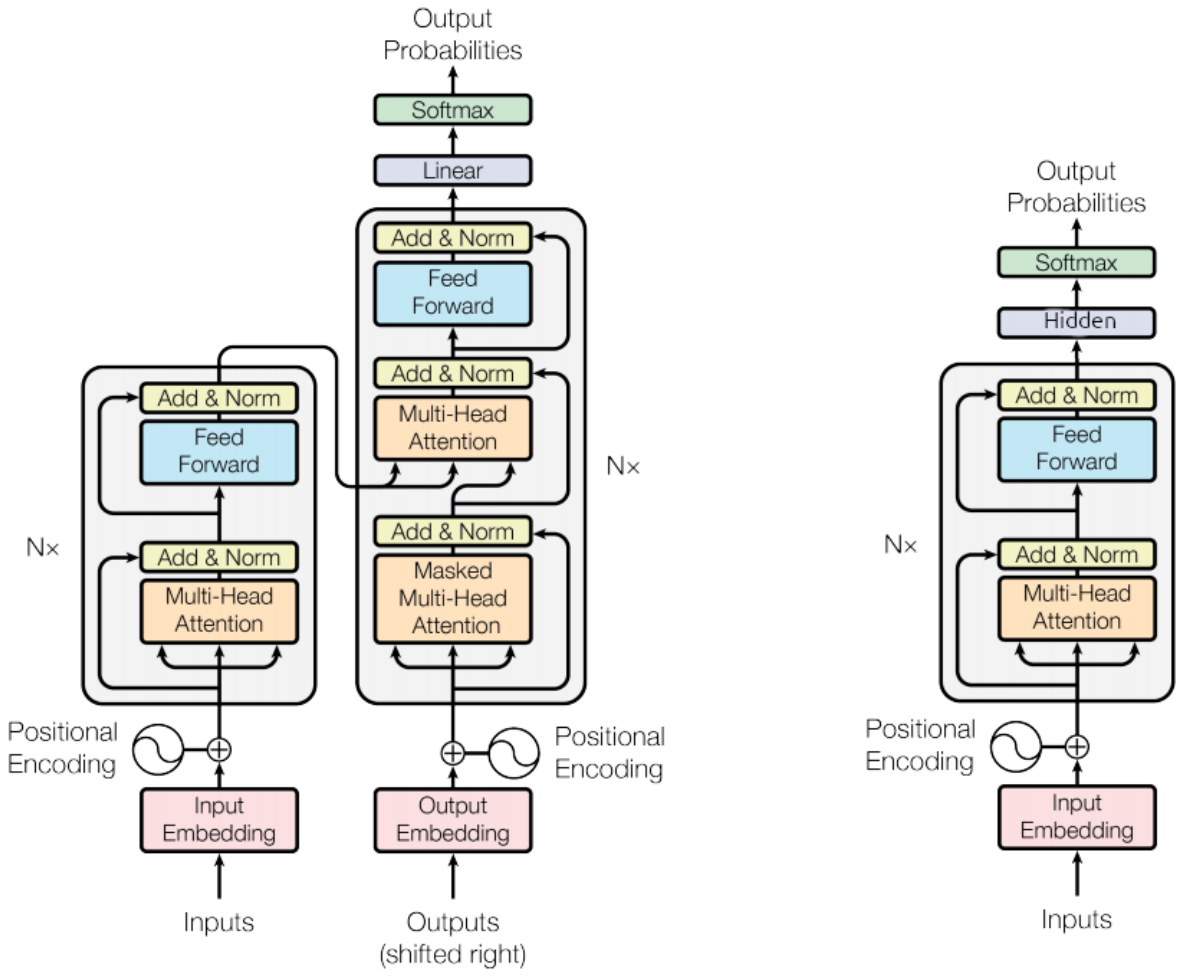

Figure 3.1: Left-right encoder-decoder architecture (left) vs bidirectional encoder-only architecture (right). The encoder only architecture uses bidirectional context to predict the identities of randomly masked tokens in the input sequence. The tokens are identified by performing softmax of the hidden layer over the vocabulary. Modified from Vaswani et al. [85].

parallel. Subsequently, the resultant d_v dimensional output values are concatenated and projected again with a learned linear projection, resulting in a final output value. This operation is summarized as

$$MultiHead(Q, K, V) = Concat(head_1, ...head_h)W^O$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

and W_i^X are the projection parameters for the vectors [85].

While the full encoder-decoder transformer architecture applies multi-head attention in multiple ways, the encoder only architecture uses only self-attention. In self-attention, the Q, K, and V vectors are all derived from the output of the layer, i.e. Q, K, and V are equivalent.

The positional encoding layer is what allows attention-based models to capture positional dependencies in input data without the use of recurrence or convolution. Transformer models use a pre-defined positional encoding function to inject positional information which is computed and added to each token vector in the input. Positional encoding functions in general may be either learned or pre-defined. In practice, Vaswani et al. [85] found that pre-defined positional encoding functions performed nearly identical to learned ones.

To perform the positional encoding for an input representation $X \in \mathbb{R}^{n \times d}$, where d is the dimension of the token vectors and n is the number of tokens, one first constructs a positional embedding matrix $P \in \mathbb{R}^d$. The canonical transformer approach uses the fixed positional encoding strategy in which elements of P are populated using sine and cosine functions,

$$p_{i,2j} = sin(\frac{i}{10000^{\frac{2j}{d}}})$$

$$p_{i,2j+1} = cos(\frac{i}{10000^{\frac{2j}{d}}})$$

[85]. This results in each dimension in P corresponding to a sinusoid, the wavelengths of which form a geometric progression from 2π to $10000 * 2\pi$. The frequencies of the wavelengths are monotonically decreasing along the embedding dimension, allowing absolute positions to be inferred by frequency. This representation also allows the model to learn relative position-wise dependencies with offset length k in constant

time, as for any offset k P_{i+k} can be represented as a linear function of P_i [85, 100]. As the positional encoding matrix P is the same shape as the input matrix X, P can be added to X, thereby embedding positional information into the input data X.

Some bidirectional encoders are optimized using the joint objectives of masked language modelling and next-sentence prediction [19]. However, in the context of marker gene modelling, there is no sensible analogy to next-sentence prediction. Additionally, Liu et al. [45] has shown that masked language modelling alone is enough to robustly train a bidirectional transformer model. Therefore, we based our transformer model architecture on the one described by Liu et al. [45].

We constructed our transformer model using the Python library HuggingFace Transformers [91]. We constructed a RoBERTa style transformer with 6 hidden layers, 12 attention heads, and a maximum sequence length of 258. Transformers require a pre-defined vocabulary size in order to construct the token embedding layer. The computational complexity of the model increases with vocabulary size. Knowing this and consulting our results from chapter 2, we decided to use BPE tokenization with a vocabulary size of $4^6 = 4096$. The hidden layer size must be divisible by the number of attention heads. To match the dimensionality of our previous embedding experiments as closely as possible while satisfying that criterion, we used a hidden layer size of 132.

3.1.2 Data set preparation

To model 16S sequences, we prepared our SILVA database 16S corpus of 410,087 unambiguous prokaryotic SSU sequences [64]. Using our byte-pair encoding algorithm trained in chapter 2, BPE6, we encoded each sequence into non-overlapping tokens with a vocabulary size of 4096.

3.1.3 Training

We trained our transformer model on the masked language modeling task. We trained our model using a $P(mask) = 0.15$, i.e. the probability of a token being randomly replaced with a mask token is 0.15. We randomly selected 10% of the training data to be used as a validation set. We trained our model using an NVIDIA RTX3090 GPU. We assessed the loss calculated over the validation set at each epoch to implement

an early-stopping callback; we configured the model to train for either 200 epochs or until the validation loss stops improving for 15 consecutive epochs, restoring the best-scoring model after training. This resulted in our model being trained for 145 epochs, approximately 65 hours on the single GPU.

3.2 Sequence embedding: Siamese network architecture

Our ultimate motivation in training our transformer language model was to fine-tune the model to produce high-quality sequence-level vector embeddings. This can be accomplished by adding a pooling layer to the pre-trained transformer and fine-tuning using annotated sequence pairs. Here we describe the model architecture, training procedure, and our proposed sequence annotation procedure.

3.2.1 Sequence transformer architecture

The sequence embedding architecture we used is described in Reimers and Gurevych [71]. The architecture is straight-forward: Simply append a mean-pooling layer to the hidden state layer of a pre-trained transformer model. For an input sequence of n tokens and hidden state dimension of d such that an input sequence x is of the shape $x \in \mathbb{R}^{n \times d}$, this results in an output vector of shape $v \in \mathbb{R}^d$. This resultant vector acts as the sequence embedding.

3.2.2 Sequence transformer loss and optimization

Simply pooling the hidden vectors of the pre-trained transformer is not enough to produce high quality sequence embeddings. However, the pre-trained model can be fine-tuned using the cosine similarity loss objective described in Reimers and Gurevych [71]. To do so, the model must be trained with annotated pairs of inputs in which each pair has an associated similarity score between 0 and 1. The model encodes each sequence in the pair to produce a final output through the pooling layer with dimensionality $\mathbb{R}^h$, where h is the cardinality of the hidden layer of the network. The model then computes the cosine similarity of the resultant sequence vectors, which also by definition will be a value between 0 and 1. Next, the model calculates the mean squared error between the cosine similarity and the annotated

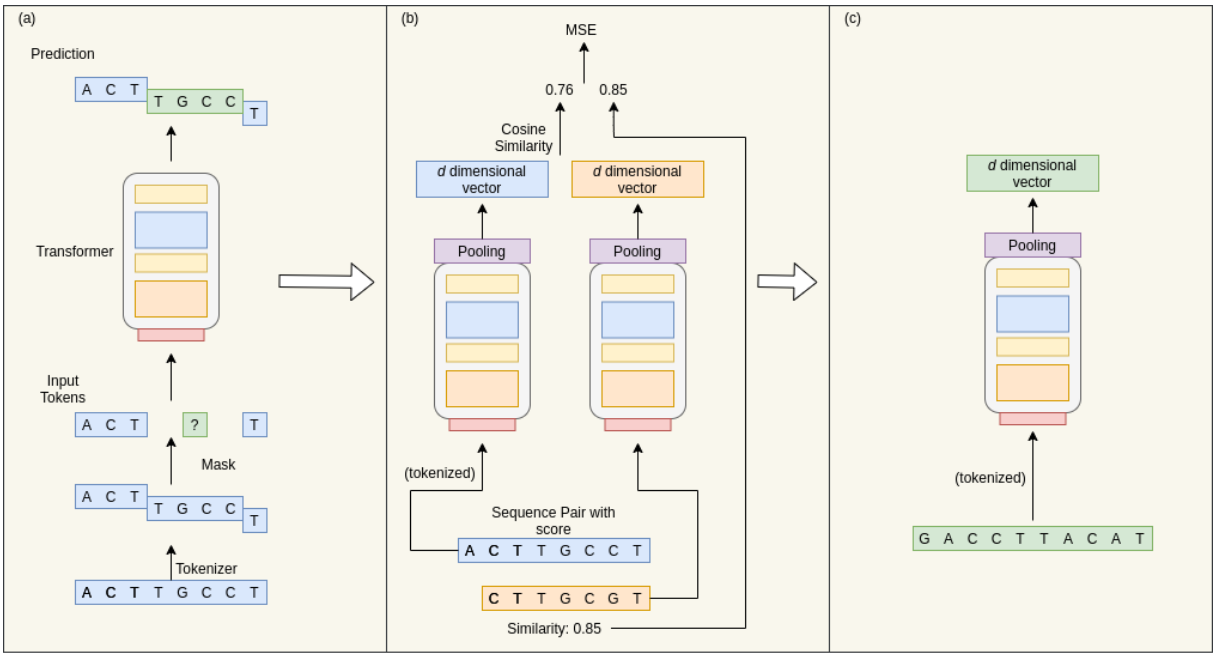

Figure 3.2: Overview of the SRoBERTa training process. (a) A transformer encoder is pre-trained using the masked language modelling task. Input sequences are tokenized and tokens are randomly replaced with a $<$ mask $>$ token during training. The model objective is to predict the identity of the masked tokens. (b) After pre-training, a pooling layer is added to the transformer model. The model is then fine-tuned using sequence pairs annotated with a similarity score. The model outputs a dense vector corresponding to each sequence in a pair and calculates the cosine similarity between them. Then, it computes the MSE between cosine similarity and the annotated similarity score and updates its weights via back-propagation. (c) Once fine tuned, the transformer encoder can produce dense vector representations of DNA sequences.

similarity score and adjusts its weights via back-propagation [71].

For our sequence embedding task, we needed to devise a measure which can be used to describe the similarity of two DNA sequences. We considered that historical research has used Spearman correlation between sequence pairwise-alignment score and sequence embedding cosine similarity as a way of assessing the quality of embeddings [55, 92]. We therefore decided to use pairwise-alignment scores as assigned by the open-source algorithm VSEARCH as our similarity score for pairs of 16S sequences. VSEARCH uses an adaptation of the dynamic programming Needleman-Wunsch algorithm [52] to assign alignment scores between 0 and 100 for all $\binom{n}{2}$ sequences in a corpus containing n sequences [75]. See figure 3.2 for an overview of the sequence embedding training process.

3.2.3 Data set preparation

As our SILVA data set is too large to feasibly compute pairwise alignment scores for all $\binom{n}{2}$ sequences within, we used our KEGG data set comprising 16,254 16S sequences to create annotated sequence pairs [36]. We used the VSEARCH version 2.17.1 command line tool to calculate pairwise alignment scores for our KEGG data set with default parameters, resulting in 132,088,131 annotated sequence pairs [75]. We divided the pairwise alignment scores by 100 to produce scores in $[0, 1]$.

We proceeded by selecting subsets of our annotated sequence pairs for training, validation, and testing. We calculated the distribution of pairwise alignment scores in our pairs data set. We then sampled from the data set using the distribution of scores as sampling weights with the goal that our sub-sets would have similar score distributions as our full data set. We sampled a training set size of 1,000,000 pairs and validation and test set sizes of 100,000 pairs without replacement.

3.2.4 Training

We used the SentenceTransformers Python library to add pooling to our pre-trained transformer model and to implement cosine similarity loss [71]. We trained our model for 35 epochs using our training set of 1,000,000 pairs. We used our validation set of 100,000 pairs to record validation loss at each epoch. After training, we restored the model parameters which achieved the best validation loss score.

3.3 Quality of sequence transformer embeddings

We performed several experiments to investigate the quality of 16S embeddings produced by our fully trained model. Where relevant, we compared the embeddings from our approach against the embeddings produced by the methods described in Woloszynek et al. [92], which we described in chapter 2. We will refer to our embeddings as SRoBERTa embeddings and the latter embeddings as SIF embeddings.

Our model produced sequence embeddings whose cosine similarities correlate with pairwise alignment scores much more strongly than those produced by SIF embedding. We repeated the clustering experiments described in chapter 2 and found that SRoBERTa sequence embeddings clustered with better metrics than SIF embeddings

at all taxonomic ranks. We than performed taxonomic classification with both embeddings and found no significant difference, as both representations achieved near-perfect scores.

One proposed application of sentence embedding architectures is fast semantic similarity search [71]. We investigated whether our embeddings can be used to quickly identify the most closely related sequence in a database and compared our results to BLAST+, the *de facto* gold standard for sequence homology search [14]. We found that our embeddings can be used to modestly approximate homology search with extreme improvements in speed, offering a trade-off of less accurate, much faster sequence look-ups.

Finally, we investigated whether sequence embeddings in microbiome data samples can be aggregated and subsequently used to classify host phenotype. We classified human stool samples from multiple studies using OTU data and one study using ASV data. Our feature representation provides a fixed-length feature vector that did not significantly decrease the performance of the classifier vs OTU/ASV center log-ratio transformed abundances. Our fixed-length representation reduces the dimensionality of the data set and makes it possible to apply trained models to data from outside the training set study, a feature which is not possible with traditional OTU/ASV methods.

3.3.1 Spearman correlation of pairwise-alignment scores and cosine similarities

In their study, Woloszynek et al. [92] showed that there was a Spearman correlation between the cosine similarities of their sequence embeddings and the VSEARCH pairwise alignment scores of the raw sequences. We used our 100,000 test set pairs to calculate pair-wise cosine similarities for sequences embedded using both our SRoBERTa method and the SIF method described by Woloszynek et al. [92]. We calculated the Spearman correlation co-efficient between the cosine similarities and the VSEARCH pairwise alignment scores. As expected, our method's embeddings have dramatically stronger correlation between cosine similarity and VSEARCH alignment score (table 3.1). Though our model was trained directly to maximize this correlation, it is still worth showing that it outperforms the SIF model in this metric.

Table 3.1: Spearman correlation coefficient values between cosine similarity of embedded sequences and pairwise alignment distance of raw sequences. Note that SRoBERTa directly optimizes the correlation between cosine similarity and pairwise alignment score, whereas SIF does not. Correlation was calculated over 100,000 VSEARCH annotated KEGG database 16S sequence pairs reserved for test purposes.

	Spearman r
SRoBERTa	**0.9310**
SIF	0.1956

3.3.2 Clustering of embedded sequences

We continued our assessment of sequence embedding quality by repeating our clustering experiments from section 2.5.3. Using our 16,254 KEGG 16S sequences, we performed sequence embedding using the SIF procedure with BPE6 tokens and our SRoBERTa sequence embedding procedure. We then clustered the embedded sequences with K-means clustering multiple times, again modifying K to reflect the number of distinct taxa at each target of {Kingdom, Phylum, Class, Order, Family, Genus}. We assessed the clusters using the same metrics as earlier: Homogeneity, completeness, ARI, and AMI. Our embeddings produced higher quality clusters nearly universally, with improvements in nearly every metric at every taxonomic rank (table 3.2).

3.3.3 Classification of embedded sequences

We assessed whether our embedded sequences can be used to classify the taxonomy of the organism of origin. We created three label sets for the data: Phylum, Order, and Genus, representing taxonomic labels at increasingly granular levels in the hierarchical taxonomy scheme. During each trial, we removed data with targets which appeared less than five times in the data set. We then used the SciKit-learn Python library to train RF classifiers to predict the labels of sequences with each label set. We performed 5-fold cross validation and report the average balanced accuracy and F1 score as well as standard deviations for each (table 3.3). We calculated balanced accuracy as the macro average of class-wise recall scores, where recall is calculated as $R = \frac{T_P}{T_P+F_N}$, with T_P indicating true positive predictions and F_N indicating false negative predictions.

Table 3.2: Clustering metrics for sequence embedding at different taxonomic ranks using both SIF embedding and SRoBERTa embedding. We performed K-means clustering for various taxonomic ranks, setting K to equal the number of distinct classes at each rank. We clustered sequences at the ranks of Phylum, Order, and Genus. ARI indicates the adjusted Rand index score. AMI indicates adjusted mutual information.

Rank		P	O	G
n Classes		36	198	666
Homogeneity	SRoBERTa	0.865	0.952	0.962
	SIF	0.813	0.929	0.954
Completeness	SRoBERTa	0.369	0.643	0.799
	SIF	0.340	0.609	0.774
ARI	SRoBERTa	0.122	0.196	0.324
	SIF	0.090	0.142	0.261
AMI	SRoBERTa	0.512	0.740	0.809
	SIF	0.473	0.704	0.778

Table 3.3: Performance metrics of 5-fold cross validated classification of embedded 16S sequences to different taxonomic ranks using random forest classifiers.

Rank	Model	Balanced Accuracy	F1
Phylum	SRoBERTa	0.930 ± 0.026	0.989 ± 0.003
	SIF	0.911 ± 0.039	0.989 ± 0.003
Order	SRoBERTa	0.962 ± 0.009	0.992 ± 0.001
	SIF	0.967 ± 0.010	0.994 ± 0.001
Genus	SRoBERTa	0.976 ± 0.004	0.985 ± 0.002
	SIF	0.983 ± 0.006	0.989 ± 0.001

Both embeddings performed with near perfect performance with small variance in validation metrics. We therefore can't conclude that either representation is consistently better suited to taxonomic assignment using direct classification over the sequence embedding.

3.3.4 Nearest sequence lookup

One interesting suggested use of sentence embedding is the ability to perform GPU accelerated semantic similarity search of text data [71]. Users can identify the closest matches to a query sequence by producing embeddings of a target search corpus, embedding the query sequence, computing the cosine similarities between the query and targets, and then selecting the highest scores. We investigated whether we can apply this method to our DNA sequence embeddings to approximate homology search.

The gold-standard sequence homology search tool in bioinformatics is BLAST [2, 14, 48]. BLAST is widely used to identify homologous sequences by comparing queries to large databases of biological sequence data. While BLAST searches are known to be relatively fast, compute costs scale with the size of queries, the size of the database, and the number of queries. Here, we investigate how closely a nearest-embedding search can approximate a BLAST search.

To perform nearest-sequence look-up, we randomly selected 1000 sequences from our KEGG data set to use as queries. We then used NCBI BLAST+ software, version 2.12.0, to construct a local database comprising our 410,078 SILVA database 16S sequences and executed a BLAST nucleotide homology search for each of the 1000 randomly selected KEGG database sequences against the SILVA database sequences, recording the IDs and alignment scores of the top 5 best hits for each sequence [14]. Besides adjusting the number of hits returned to five, we used default parameters for our BLAST search. Next, we trained our SRoBERTa model to produce a database of vector embedded 16S sequences from our SILVA data set. Finally, we embedded the 1000 randomly selected KEGG sequences and calculated the cosine similarity scores between the query sequences and database sequences, again recording the top 5 results for each sequence. For each sequence, we compared how many of the top 5 results from the BLAST search overlap with the top 5 results from the cosine similarity search. We found that 76% of the sequences investigated had at least one

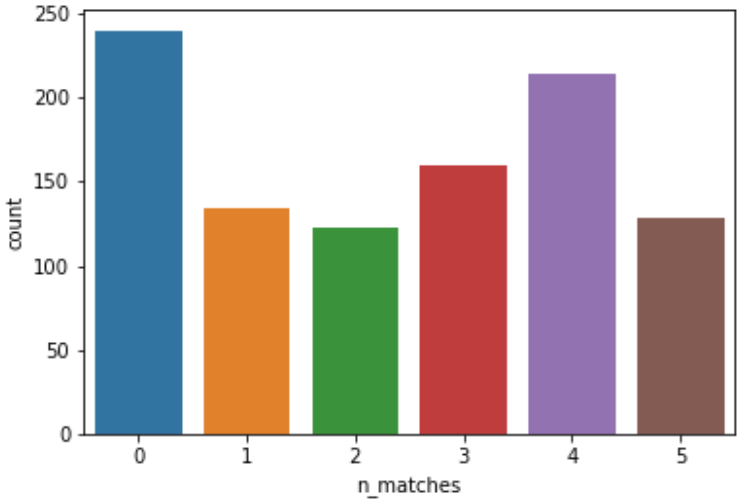

Figure 3.3: Distribution of overlaps of top five results between BLAST homology search and cosine similarity nearest-neighbor lookup. We used BLAST+ to identify the top five closest sequences for 1000 randomly selected KEGG 16S queries against a database of 410,087 SILVA 16S sequences. We then encoded all query and database sequences to a continuous vector representation using our fully trained SRoBERTa model and calculated the top five closest sequences for each query using cosine similarity. For each sequence, we count how many matches were found in the top five results of both methods.

result in common, with 13% of sequences producing exactly the same top 5 matches (figure 3.3).

3.4 Classification of host phenotype from amplicon sequence data

One of our motivations for developing fixed-length vector embeddings for 16S sequences is the promise of developing a fixed-length representation for compositional microbiome data. As discussed in section 1.2.3, the feature vectors of microbiome data are usually count vectors, with each element corresponding to the count of an OTU or ASV that was identified in the study. Because different studies (and indeed, the same studies with different sample denoising protocols) will result in different OTUs/ASVs being identified, different studies will produce data with variable feature vector shapes. This difference precludes the ability to develop models which can perform classification across studies, which is an essential ability for production machine learning applications and therefore a major barrier to deploying microbiome machine learning classification in clinical settings.

Table 3.4: Summary of data sets used for microbiome sample classification.OTU data were cleaned by removing OTUs which appeared in less than 15% of samples. As part of the DADA2 denoising process, ASVs with frequency below the first quartile of frequencies were removed, therefore this cleaning step was not applicable for our ASV data set.

Study	Type	n samples	n feat., init	n feat., post	Task.
Gevers	OTU	164	109,869	1787	Inflammatory bowel disease
Lozupone	OTU	53	10,105	887	HIV
Zeller	OTU	118	82,666	8147	Colorectal cancer
Son	OTU	105	17,983	888	Austism Spectrum Disorder
Zackular	ASV	60	3,113	N/A	Colorectal cancer

To address this limitation, we propose a strategy which uses a relative-abundance weighted average of the sequence embeddings comprising a microbiome sample to produce a single fixed-length feature vector. We used publicly available human stool sample data from five studies and compared the ability of a RF classifier to learn to predict host disease state from our embedding strategy against a classical relative abundance approach. While our approach did not improve performance, it never performed significantly worse than the relative abundance approach and offers the advantage of a fixed-length, continuous-valued representation.

3.4.1 Data set preparation

We prepared five data sets for our sample-level embedding classification trials. We retrieved four data sets from MicrobiomeHD, which hosts data and metadata for several microbiome classification tasks [20]. We retrieved data sets with binary classification tasks of disease state vs healthy for various disease states (table 3.4). We retrieved only data sets which used Illumina MiSeq 16S sequencing, ignoring data sets curated from studies using older instruments. The data provided from MicrobiomeHD come pre-processed as OTU counts clustered at 100% sequence identity using the USEARCH algorithm along with corresponding representative sequences; raw sequences were not provided, preventing analysis of the impact of denoising strategy on the final result [20]. All samples were human stool samples. The data sets we used are described in table 3.4.

OTU data sets are sparse with high cardinality (i.e. number of features). These

characteristics typically result in increased compute cost and decreased efficacy for machine learning models. We therefore subjected all our OTU count tables to the same data cleaning procedure. First, we perform simple data cleaning by removing any OTUs or samples that had either missing values or were comprised of all zeros. To reduce the high cardinality, We also removed all OTUs which appear in less than 15% of samples (table 3.4).

For our fifth data set, we retrieved data from a study investigating microbiome signatures in patients with colorectal cancer [96]. We retrieved this data as paired-end raw reads and processed them using a denoising procedure rather than an OTU clustering one. We used the QIIME2 bioinformatics software package to de-multiplex the sequences and investigate their quality scores [9]. We then assigned the sequences to ASVs using the DADA2 denoising algorithm [12].

3.4.2 Sample-level embedding

Microbiome data sets are presented as matrices $T \in \mathbb{Z}+^{n \times m}$, where the n rows correspond to individual samples and the m columns correspond to OTU or ASV IDs. Therefore, a single microbiome sample is represented as a count vector in $\mathbb{Z}+^{m}$, where several of the entries may be zero. Separately, microbiome data are accompanied by a table mapping each of the m ASVs or OTUs to representative DNA sequences. These m-dimensional vectors can be used to successfully classify several host-phenotype prediction tasks, however, their cardinality is dependent on the OTU/ASVs identified in a set of samples. This intra-data set dependency prevents the use of models trained with one data set from being used with others.

Here, we propose a procedure to create fixed-length, low-cardinality dense vector representations of microbiome samples. Given a trained sequence embedding model which projects sequences to a d-dimensional vector space, our sample-embedding procedure is a simple weighted average of the sequence embeddings for the sequences found in a sample. For each sample, we retrieved the frequencies of OTUs or ASVs with non-zero frequency and embedded the corresponding representative sequences to dense-vector representation using our trained model. In designing our sample-level embedding procedure, we considered microbiome data to be fundamentally compositional, bounded by the upper limit of sequencing capabilities of the high throughput

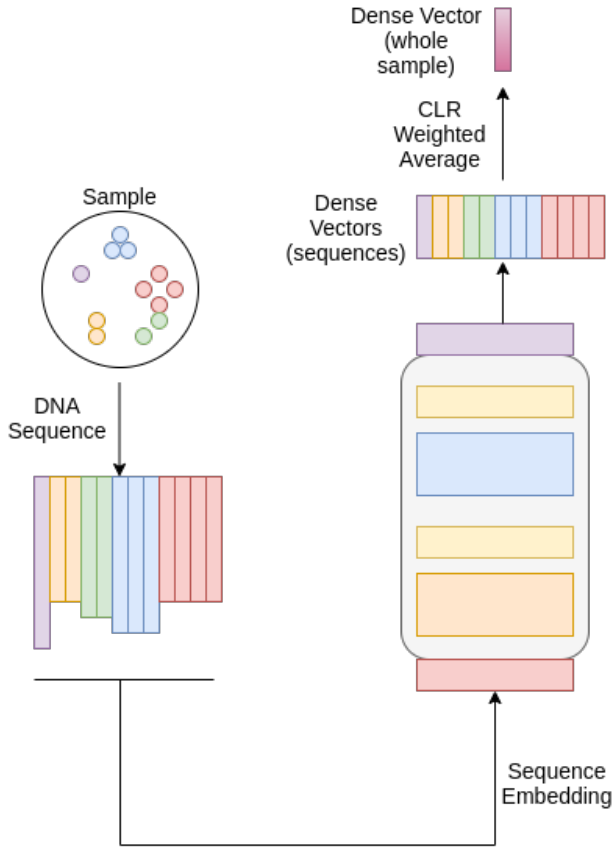

Figure 3.4: Overview of the microbiome sample embedding procedure. Representative sequences are retrieved from a microbiome sample and embedded to a dense vector representation using the SRoBERTa model. The count values of the representative sequences are transformed using a centered log-ratio transformation. The corresponding transformed values are then used as weights to produce a weighted average of the representative sequences, producing a fixed-length dense vector representation of the microbiome sample.

sequencing instruments, as per [28, 27]. Therefore, rather than using OTU or ASV frequencies or relative abundances as weights in our weighted average, we transformed the non-zero OTU/ASV frequencies to an Aitchison geometry using a centered log-ratio (CLR) transformation. We then used the transformed count values as weights to produce a weighted average of the sequence embeddings comprising a sample (algorithm 4). In so doing, we projected samples to a d-dimensional dense vector representation using only data pertaining to the individual sample (see figure 3.4 for an overview).

Algorithm 4 Count-table sample embedding procedure

Input: An OTU or ASV relative abundance table $T \in \mathbb{Z}+^{n \times m}$, a set of sequences S, $|S| = m$, comprising the representative sequences of table T, a trained sentence transformer model with pooling dimension $p \in \mathbb{R}^d$

Output: n Sample embeddings $v_t \in \mathbb{R}^d$

for all sequences s in S **do**

$e_s \in \mathbb{R}^d \leftarrow$ model.encode(s)

end for

for all $i \in [0, n]$ **do**

$r \leftarrow$ non-zero entries in T_i

$r_t \leftarrow$ centered log-ratio transform of r

$v_r \leftarrow d$-dimensional vector with all entries initialized to 0

for all $j \in [0, |r_t|]$ **do**

$e_j \leftarrow$ sequence embedding corresponding to OTU or ASV at $r_t[j]$

$v_r \leftarrow v_r + r_i[j](e_j)$

end for

$v_r \leftarrow v_r \div |r_i|$

end for

3.4.3 Classification of samples

Our goal is to demonstrate that our sample embedding procedure can replace canonical microbiome classification procedures without a reduction in performance. To do so, we trained a RF classifier on five binary classification tasks; four OTU based classification tasks and one ASV based classification task as described in table 3.4.

To classify OTU/ASV sample data, we followed best practice recommendations from the literature. As described above, we removed low-frequency OTUs by removing those OTUs which appear in less than 15% of samples. Additionally, we used centered log-ratio transforms of the OTU and ASV count vectors rather than the raw counts or relative abundances. According to several authors, this is the best practice when treating these data [80, 101, 28, 20, 27]. We also performed trials using raw counts and relative abundance vectors but found that the centered log-ratio transformed data consistently performed the best across all data sets, so we omit those results.

We trained a SciKit-learn RF classifier with 5000 estimators and otherwise default parameters to classify each of our five data sets [58]. We performed five-fold cross validation and recorded the area-under-curve for the receiver operating characteristic curve (ROC score). We trained models using both CLR transformed count vectors and our dense vector embeddings. We used the mean and standard deviation of ROC scores from the cross-validated trials to calculate the significant difference between the results of each feature representation using the student's t-test. We consider results to be significantly different if the t-test produces a p-value less than or equal to 0.05. Our embedding method's performance was not significantly different from the CLR transformed count vectors in four of the five studies and its performance was significantly better in the fifth (table 3.5).

3.5 Conclusions

Our findings show that our sequence embedding method creates dense vector embeddings that meet or exceed the quality of previously published methods in all metrics.

We have shown clearly that our method can be used to effectively perform taxonomic classification of 16S sequences, providing a fixed-length and low cardinality representation of variable length sequences without resorting to token counts. Our

Table 3.5: Classification of disease state from stool microbiota using center-log ratio normalized abundance values and SRoBERTa embedding aggregation for data sets from five studies. AUC indicates the mean area-under-curve for the receiver operating characteristic curve calculated over a five-fold cross validation. p-value is calculated via two-sample t-test, results are considered significantly different when $p < 0.05$.

Study	AUC (Abundance CLR)	AUC (Embedding)	p-value	Sig. Diff.
Gevers (OTU)	0.770 ± 0.134	0.753 ± 0.202	0.843	No
Lozupone (OTU)	0.957 ± 0.055	0.930 ± 0.058	0.475	No
Zeller (OTU)	0.754 ± 0.143	0.606 ± 0.207	0.230	No
Son (OTU)	0.581 ± 0.058	0.755 ± 0.027	0.001	Yes
Zackular (ASV)	0.456 ± 0.099	0.511 ± 0.125	0.466	No

representation also preserves relative distances between sequences, allowing for interpretable analyses such as clustering that would not be sensible on count based methods.

Our embeddings can be aggregated in a way such that microbial community samples can be represented as weighted averages of the sequences they contain. The resulting embeddings don't perform significantly worse than count table representations, implying that they maintain approximately the same amount of information with significant reduction of feature cardinality as well as the benefits of a fixed-length representation. Additionally, our method is the first DNA sequence embedding procedure which does not rely on data that do not pertain to individual samples. This has major implications in the development of scalable machine learning models for phenotype prediction tasks from host microbiome signatures, as our embedding procedure may be applied across studies and to unseen test data.

The success of our sample embedding procedure implies that our sequence embedding method is robust to variable sequence lengths, since the representative sequences found in microbiome samples are much shorter than the full length 16S sequences we used to train the embedding model (≈250 nucleotides vs ≈1500 nucleotides).

We have shown that the distance-preserving nature of our sequence embeddings allows for fast approximate nearest-sequence search using cosine similarities. Though our results were not a high fidelity approximation of BLAST searching, there may be useful applications for such searches where low sensitivity results are an acceptable

trade-off for speed. Given the apparent robustness of our approach to variable sequence lengths as inputs, an interesting avenue of research could be to investigate the ability of our model to perform asymmetric sequence lookups (i.e. lookup operations in which query sequences are significantly shorter than target sequences).

Some researchers use a handful of specific OTUs or ASVs in an ensemble fashion with other host meta data to create better machine learning models [96, 20]. The fixed-length, study-independent nature of our method makes it feasible to pre-train models across several data sets if curating them becomes possible. Given that our method fares no worse than direct OTU or ASV feature representations, we propose that such a pre-trained model could contribute to the success of predictive ensemble models.

Given that our embedding procedure outperforms the previously published method in all tested metrics while forgoing the requirement to use data set wide meta-statistics in their calculations, we suggest that our model has achieved state of the art performance in production of both 16S dense vector embeddings and microbiome sample level dense vector embeddings.

Chapter 4

Conclusion

Our research investigating novel tokenization strategies and sequence-embedding approaches for marker genes was successful and naturally lends itself to future research directions and applications. Here we discuss the implications of our findings, the limitations of our work, and propose future research directions.

4.1 Implications

Our findings from chapter 2 show that using BPE or unigram language modelling to learn tokenization strategies for 16S DNA sequences is viable and results in compression of sequences without loss of performance in downstream tasks. The ability to use an approach like BPE or unigram tokenization in lieu of k-mer segmentation becomes important in tasks such as those we performed in chapter 3 in which sequence length and vocabulary size become important variables for computational complexity.

Although unigram language modelling may be a more robust choice for natural language tasks such as neural machine translation [11, 39], our findings here suggest that BPE tokenization results in greater compression of marker-gene DNA sequences and increased robustness to parameter choice in downstream applications. We therefore believe that BPE is the best choice for tokenization strategy in future studies performing language modelling tasks over DNA sequences, particularly in the context of marker-gene analyses.

Contemporary language modelling paradigms which involve pre-training large models using unlabelled data followed by fine-tuning using supervised objectives allow users to implement creative objectives and loss functions. Previous DNA sequence-embedding work has pointed to Spearman correlation between embedding space cosine similarities and sequence space pairwise alignment distances as evidence that their embedding procedures were meaningful [55, 92]. We have shown that using contemporary language modelling paradigms we can directly optimize this correlation,

ensuring that our embeddings maintain the so-called semantic meaningfulness of the raw sequences. As research in this area continues, researchers may propose additional metrics and tests for assessing the biological meaningfulness and/or usefulness of sequence embeddings. Using procedures related to ours, they may directly optimize these criteria and ensure the quality of their embeddings.

In chapter 3, we showed that it is possible to rapidly approximate similar sequence searching using cosine similarity in the sequence embedding space. Though such searches are not likely to replace large-scale biological sequence-searching algorithms such as BLAST, it is possible that this sequence search task could be further optimized and applied in situations where fast heuristic searches are helpful.

One of our most important findings is the demonstration that one may aggregate the sequence embeddings from a microbiome sample to create sample-level dense vectors. To the best of our knowledge, our approach is the first to create fixed-length feature representations of microbiome samples without relying on statistics calculated over a set of samples. We therefore propose that our approach is the first microbiome feature representation that allows models trained with a given data set to be universally applied to any microbiome test set data. The development of such approaches is important towards the development of machine-learning models with clinical applicability [20, 47, 80, 101].

4.2 Limitations

An immediately apparent limitation of our work is that we have limited the scope of our analyses to 16S DNA sequences, raising the question as to how our methods would perform using other sources of DNA. Limiting the scope of our work to a single gene made several decisions trivial which become difficult when extending to broader DNA sequences. For example, researchers using transformer-based language models for DNA sequences have had to carefully design how to chunk their data for input to their model, as canonical transformer models do not scale well when maximum sequence length exceeds $\approx$500 tokens [17, 32, 85], whereas this was not a consideration for our model as each input could correspond to a single gene sequence. Similarly, the limitations of byte-pair encoding and similar learned tokenization models remain unknown when used with broader genomic data. We found that learned tokenization

strategies resulted in compression of 16S sequences through a combination of non-overlapping tokens and collapsing of conserved regions into single tokens. We have not tested whether a trained tokenization encoder could achieve compression of sequences if trained using non-homologous DNA sequences, which would have an increased diversity of substrings as compared to a corpus of marker genes.

Creating sample level microbiome dense vectors remains challenging due to the substantial variance involved in the pre-processing of raw read data into OTUs or ASVs. As we discussed earlier, OTU clustering methods are sensitive to parameter choice and data set composition and are generally unstable and noisy [51, 51, 56]. Similarly, ASV generation requires several user defined parameters and quality control decisions which affect the output. Therefore, using OTU and ASV data as input for feature engineering is likely to produce noisy outputs. In our work here, we do not investigate how robust our sample embedding model is to the noisy process of OTU and ASV generation so we cannot speculate how sensitive the model is to upstream parameter choices.

In our research, we have performed some parameter space searching but were limited due to the high compute demand for the methods we tested. For example, our choices of BPE and unigram language model vocabulary sizes were limited only to 4^6 and 4^8, but it is possible that other vocabulary sizes could produce more desirable results. Similarly, we do not repeat our experiments with variations in the tokenization algorithm's maximum token lengths. We also do no parameter searching with our transformer model, instead using default values from optimized language-modelling papers (save for our vocabulary size and embedding dimension, which we set to equal our BPE vocabulary and approximately equal to our previously tested embeddings, respectively) [45, 91]. It is possible that adjusting the transformer model's vocabulary size, embedding dimension, or other hyperparameters could significantly affect the model's downstream performance. Due to the significant compute demand required to train large transformer models and our satisfactory results, we do not explore transformer hyperparameter optimization in this work.

Along with leaving transformer hyperparameters largely unexplored, we do not try different training procedures for our transformer model. While some researchers

randomly inject short sentence fragments into the training of their bidirectional transformers [19], other researchers have shown that the use of fragment injection does not improve the training speed or robustness of the model [45]. For simplicity, we used only full-length DNA sequences while training our transformer, but it remains untested whether the DNA language model would benefit from fragment injection during training. As we do apply our model to embedding of short DNA reads during microbiome sample embedding, it is possible that the model would be improved by injection of fragments or other data augmentation procedures.

When fine-tuning our transformer to learn DNA sequence embeddings, we only performed experiments using our cosine similarity/pairwise alignment score mean squared error loss function. It is possible that different fine-tuning approaches could result in higher quality sequence embeddings. For example, one could modify the fine-tuning procedure to use triplet networks, ranking the distance between sequence triplets [71]. One could also investigate the use of alternative scoring metrics for annotating sequence pairs with distance scores; we used the VSEARCH pairwise alignment score due to its use in previously published DNA sequence embedding work, but it is possible that authors could devise other scoring metrics for distance between homologous DNA sequences.

One major limitation of our work, common to many feature-embeddings, is the loss of interpretability when using sample embedding methods as we describe. While researchers gain the ability to project microbiome data to fixed-dimension, data set-independent representations, they lose the ability to analyse the importance of individual microbes in their classification tasks. This is a fundamental limitation of the bidirectional transformer architecture; as it is composed only of the transformer encoder, it does not have the ability to reconstruct inputs from the dense vector space. It is possible that alternative models using transformer encoder-decoders would be able to produce high quality DNA sequence embeddings via a generative model, providing the ability to decode the resultant dense vectors, but such a model is outside the scope of this work and would require different training objectives. In addition to the inability of our model to decode dense-vector sequence representations, important sequences would be obfuscated by our sample embedding procedure of weighted averaging. Creating a fixed-length vector representation of sample vectors from which

feature importance of individual organisms could be recovered remains an open problem.

4.3 Future directions

An obvious future direction for this work is to investigate the applicability of our approaches on entire genomes rather than a single set of homologous marker genes. Transformer models do better with data sets orders of magnitude larger than what we used here [45]. Extending the DNA language modelling task to genomic data is an interesting proposition which is becoming more feasible as researchers develop transformer models capable of handling large inputs [98, 18, 37]. Indeed, the authors of one of these transformer models claimed that their model could use human genomic data and learn to annotate promoter sequences with high fidelity [98]. If researchers wished to investigate whether such large models would be able to be used to produce fixed-length embeddings of genomic DNA, they would have to decide on sensible chunks of genomes to use. For example, researchers could investigate the feasibility of embedding protein coding regions of DNA, individual chromosomes, viral genomes, or smaller bacterial genomes.

Transformers in natural language processing tasks are generally trained to model a single human language, with some exceptions. Interestingly, [44] suggested that DNA sequences from different organisms seem to statistically resemble distinct languages. Their method uses maximum likelihood to determine n-gram segmentation of DNA sequences. They write that genomic data from different organisms result in distinct probability distributions for their n-gram segmentation strategy, resulting in differentiated segmentation strategies for different genomes. Indeed, it is well known that genomic composition varies widely from species to species. An interesting experiment could be to train transformers to model genomic DNA from a specific phylogenetic group (e.g., species or genus) and investigate the efficacy of transfer learning to other groups, measuring the relationship between performance metrics in the transfer learning task and phylogenetic distance. It would be interesting to determine boundaries at which transfer learning begins to fail; for example, presumably transfer learning would be successful among closely related bacterial species, but it is unlikely that a model trained using bacterial genomics would perform well with human genomic test

data.

Our results showing the ability to perform fast nearest sequence queries are promising, especially where our model was not trained to specifically perform well at that task. It is likely that some models will be better suited to this type of query than others. While we saw some modest success comparing BLAST sequence similarity search results using similar sequences (16S only), it remains to be seen whether the search results would perform well given a model that produces vectors of a wider range of genomic material. It is possible that an asymmetric search strategy such as those seen in question answering models (where query is typically shorter much shorter than the target) could result in better look-up results. Optimizing these searches as a heuristic for use with large genomic database searches could be an interesting use case.

An interesting avenue for research is the idea of replacing OTU or ASV picking algorithms with some aspect of the transformer language model. A simple version of this would be to investigate whether clustering embedded reads using cosine similarity threshold or other clustering approach could replace OTU clustering. A more involved investigation would involve training transformer language models to use raw short-read data, i.e. paired end read data which have not been processed at all, including ambiguous nucleotides and quality scores. The paramaterized denoising or OTU picking algorithms could be directly replaced with a sequence embedding procedure. While the representative sequences from such a denoising algorithm would not be recoverable, we have shown that taxonomic assignment can be performed using sequence embeddings as input, which would allow downstream microbial ecology analyses to proceed as normal.

A relatively simple but compelling research idea is investigating how additional data could be implicitly encoded into our pairwise sequence distance optimization task. In our work, we only use VSEARCH pairwise alignment scores as the gold-standard pairwise score. Because the score is meant to indicate the semantic relatedness of the sequences, the choice of score will impact the resulting embeddings greatly. It is plausible that a motivated author could devise a more biologically meaningful scoring metric to implicitly encode additional information into the sequence embeddings. For example, an author could construct a corpus of 16S sequences annotated

with distances derived from the organisms' whole genomes. This could be an interesting way of incorporating genomic-level information into the sequence embeddings.

Creating 16S embeddings and short-read embeddings results in the creation of uniquely structured data sets in the case of microbiome data. Each sample in the data set can be represented as a variable-length "bag-of-vectors", in which the sample contains n count values corresponding to the number of d dimensional vectors contained in the sample. In our work, we collapse these "bag-of-vector" data by producing weighted averages of the vectors. Devising novel ways of aggregating this data could be an interesting research direction with both practical and theoretical implications. If the data are treated as compositional as suggested by [28, 27], one may consider scouring the compositional data analysis literature to propose some means of aggregating the vectors. For example, [1] suggested using the log-contrast as a way of representing compositional mixtures. Given some mixture X with n objects compositional co-efficients A, you can represent the mixture as $a_0 \log(x_0) + , ..., + a_n \log(x_n)$, where $\sum_{i=0}^{n} a_i = 1$. Of course, this approach cannot work where x_i is a real valued vector, as $\log(x)$ is undefined where $x < 0$. Investigating alternative approaches based on the principles of the log-contrast may be of interest.

Researchers may also consider geometrically motivated solutions to the task of aggregating "bag-of-vector" representations. For example, [95] suggest that pre-trained word embeddings can be combined to create sentence embeddings in a parameter free way using matrix algebra. Investigating similar approaches may allow 16S embeddings to be aggregated to create sample embeddings in a way that is competitive with the approach we describe in our work.

We consider our work here to be an important development in the research of biological sequence language modelling and embedding, particularly in the realm of marker gene and microbiome analysis. We hope that our work and proposed future research directions serve to inspire researchers to continue developing this interesting and relatively nascent area of research.

Bibliography

[1] J. Aitchison and J. Bacon-Shone. Log contrast models for experiments with mixtures. *Biometrika*, 71(2):323–330, 1984. doi: 10.1093/biomet/71.2.323. URL https://doi.org/10.1093/biomet/71.2.323.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215 (3):403–410, October 1990. doi: 10.1016/s0022-2836(05)80360-2. URL https://doi.org/10.1016/s0022-2836(05)80360-2.

[3] Amnon Amir, Daniel McDonald, Jose A. Navas-Molina, Evguenia Kopylova, James T. Morton, Zhenjiang Zech Xu, Eric P. Kightley, Luke R. Thompson, Embriette R. Hyde, Antonio Gonzalez, and Rob Knight. Deblur rapidly resolves single-nucleotide community sequence patterns. *mSystems*, 2(2), April 2017. doi: 10.1128/msystems.00191-16. URL https://doi.org/10.1128/msystems.00191-16.

[4] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003. ISSN 1532-4435.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

[8] Nicholas A. Bokulich, Benjamin D. Kaehler, Jai Ram Rideout, Matthew Dillon, Evan Bolyen, Rob Knight, Gavin A. Huttley, and J. Gregory Caporaso. Optimizing taxonomic classification of marker-gene amplicon sequences with QIIME 2's q2-feature-classifier plugin. *Microbiome*, 6(1), May 2018. doi: 10.1186/s40168-018-0470-z. URL https://doi.org/10.1186/s40168-018-0470-z.

[9] Evan Bolyen, Jai Ram Rideout, Matthew R. Dillon, Nicholas A. Bokulich, Christian C. Abnet, Gabriel A. Al-Ghalith, Harriet Alexander, Eric J. Alm, Manimozhiyan Arumugam, Francesco Asnicar, Yang Bai, Jordan E. Bisanz, Kyle Bittinger, Asker Brejnrod, Colin J. Brislawn, C. Titus Brown, Benjamin J. Callahan, Andrés Mauricio Caraballo-Rodríguez, John Chase, Emily K. Cope,

Ricardo Da Silva, Christian Diener, Pieter C. Dorrestein, Gavin M. Douglas, Daniel M. Durall, Claire Duvallet, Christian F. Edwardson, Madeleine Ernst, Mehrbod Estaki, Jennifer Fouquier, Julia M. Gauglitz, Sean M. Gibbons, Deanna L. Gibson, Antonio Gonzalez, Kestrel Gorlick, Jiarong Guo, Benjamin Hillmann, Susan Holmes, Hannes Holste, Curtis Huttenhower, Gavin A. Huttley, Stefan Janssen, Alan K. Jarmusch, Lingjing Jiang, Benjamin D. Kaehler, Kyo Bin Kang, Christopher R. Keefe, Paul Keim, Scott T. Kelley, Dan Knights, Irina Koester, Tomasz Kosciolek, Jorden Kreps, Morgan G. I. Langille, Joslynn Lee, Ruth Ley, Yong-Xin Liu, Erikka Loftfield, Catherine Lozupone, Massoud Maher, Clarisse Marotz, Bryan D. Martin, Daniel McDonald, Lauren J. McIver, Alexey V. Melnik, Jessica L. Metcalf, Sydney C. Morgan, Jamie T. Morton, Ahmad Turan Naimey, Jose A. Navas-Molina, Louis Felix Nothias, Stephanie B. Orchanian, Talima Pearson, Samuel L. Peoples, Daniel Petras, Mary Lai Preuss, Elmar Pruesse, Lasse Buur Rasmussen, Adam Rivers, Michael S. Robeson, Patrick Rosenthal, Nicola Segata, Michael Shaffer, Arron Shiffer, Rashmi Sinha, Se Jin Song, John R. Spear, Austin D. Swafford, Luke R. Thompson, Pedro J. Torres, Pauline Trinh, Anupriya Tripathi, Peter J. Turnbaugh, Sabah Ul-Hasan, Justin J. J. van der Hooft, Fernando Vargas, Yoshiki Vázquez-Baeza, Emily Vogtmann, Max von Hippel, William Walters, Yunhu Wan, Mingxun Wang, Jonathan Warren, Kyle C. Weber, Charles H. D. Williamson, Amy D. Willis, Zhenjiang Zech Xu, Jesse R. Zaneveld, Yilong Zhang, Qiyun Zhu, Rob Knight, and J. Gregory Caporaso. Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nature Biotechnology*, 37(8):852–857, July 2019. doi: 10.1038/s41587-019-0209-9. URL `https://doi.org/10.1038/s41587-019-0209-9`.

[10] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction, 2019.

[11] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining, 2020.

[12] Benjamin J Callahan, Paul J McMurdie, Michael J Rosen, Andrew W Han, Amy Jo A Johnson, and Susan P Holmes. DADA2: High-resolution sample inference from illumina amplicon data. *Nature Methods*, 13(7):581–583, May 2016. doi: 10.1038/nmeth.3869. URL `https://doi.org/10.1038/nmeth.3869`.

[13] Benjamin J Callahan, Paul J McMurdie, and Susan P Holmes. Exact sequence variants should replace operational taxonomic units in marker-gene data analysis. *The ISME Journal*, 11(12):2639–2643, July 2017. doi: 10.1038/ismej.2017.119. URL `https://doi.org/10.1038/ismej.2017.119`.

[14] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason

Papadopoulos, Kevin Bealer, and Thomas L Madden. BLAST+: architecture and applications. *BMC Bioinformatics*, 10(1):421, 2009. doi: 10.1186/1471-2105-10-421. URL https://doi.org/10.1186/1471-2105-10-421.

[15] Giovanni Cammarota, Gianluca Ianiro, Anna Ahern, Carmine Carbone, Andriy Temko, Marcus J. Claesson, Antonio Gasbarrini, and Giampaolo Tortora. Gut microbiome, big data and machine learning to promote precision medicine for cancer. *Nature Reviews Gastroenterology & Hepatology*, 17(10):635–648, July 2020. doi: 10.1038/s41575-020-0327-3. URL https://doi.org/10.1038/s41575-020-0327-3.

[16] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[17] Jim Clauwaert and Willem Waegeman. Novel transformer networks for improved sequence labeling in genomics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2020. doi: 10.1109/TCBB.2020.3035021.

[18] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[20] Claire Duvallet, Sean M. Gibbons, Thomas Gurry, Rafael A. Irizarry, and Eric J. Alm. Meta-analysis of gut microbiome studies identifies disease-specific and shared responses. *Nature Communications*, 8(1), December 2017. doi: 10.1038/s41467-017-01973-8. URL https://doi.org/10.1038/s41467-017-01973-8.

[21] Robert C Edgar. Updating the 97% identity threshold for 16s ribosomal RNA OTUs. *Bioinformatics*, 34(14):2371–2375, February 2018. doi: 10.1093/bioinformatics/bty113. URL https://doi.org/10.1093/bioinformatics/bty113.

[22] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *bioRxiv*, 2020. doi: 10.1101/2020.07.12.

199554. URL https://www.biorxiv.org/content/early/2020/07/21/2020.07.12.199554.

[23] Jessica D. Forbes, Chih yu Chen, Natalie C. Knox, Ruth-Ann Marrie, Hani El-Gabalawy, Teresa de Kievit, Michelle Alfa, Charles N. Bernstein, and Gary Van Domselaar. A comparative study of the gut microbiota in immune-mediated inflammatory diseases—does a common dysbiosis exist? *Microbiome*, 6(1), December 2018. doi: 10.1186/s40168-018-0603-4. URL https://doi.org/10.1186/s40168-018-0603-4.

[24] G. E. Fox, L. J. Magrum, W. E. Balch, R. S. Wolfe, and C. R. Woese. Classification of methanogenic bacteria by 16s ribosomal RNA characterization. *Proceedings of the National Academy of Sciences*, 74(10):4537–4541, October 1977. doi: 10.1073/pnas.74.10.4537. URL https://doi.org/10.1073/pnas.74.10.4537.

[25] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788.

[26] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning, 2017.

[27] Gregory B. Gloor and Gregor Reid. Compositional analysis: a valid approach to analyze microbiome high-throughput sequencing data. *Canadian Journal of Microbiology*, 62(8):692–703, August 2016. doi: 10.1139/cjm-2015-0821. URL https://doi.org/10.1139/cjm-2015-0821.

[28] Gregory B. Gloor, Jean M. Macklaim, Vera Pawlowsky-Glahn, and Juan J. Egozcue. Microbiome datasets are compositional: And this is not optional. *Frontiers in Microbiology*, 8, November 2017. doi: 10.3389/fmicb.2017.02224. URL https://doi.org/10.3389/fmicb.2017.02224.

[29] Yan He, J Gregory Caporaso, Xiao-Tao Jiang, Hua-Fang Sheng, Susan M Huse, Jai Ram Rideout, Robert C Edgar, Evguenia Kopylova, William A Walters, Rob Knight, and Hong-Wei Zhou. Stability of operational taxonomic units: an important but neglected property for analyzing microbial diversity. *Microbiome*, 3(1), May 2015. doi: 10.1186/s40168-015-0081-x. URL https://doi.org/10.1186/s40168-015-0081-x.

[30] Philip Hugenholtz, Brett M. Goebel, and Norman R. Pace. Impact of culture-independent studies on the emerging phylogenetic view of bacterial diversity. *Journal of Bacteriology*, 180(18):4765–4774, September 1998. doi: 10.1128/jb.180.18.4765-4774.1998. URL https://doi.org/10.1128/jb.180.18.4765-4774.1998.

[31] Hitoshi Iuchi, Taro Matsutani, Keisuke Yamada, Natsuki Iwano, Shunsuke Sumi, Shion Hosoda, Shitao Zhao, Tsukasa Fukunaga, and Michiaki Hamada. Representation learning applications in biological sequence analysis. *bioRxiv*,

2021. doi: 10.1101/2021.02.26.433129. URL https://www.biorxiv.org/content/early/2021/02/27/2021.02.26.433129.

[32] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 02 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab083. URL https://doi.org/10.1093/bioinformatics/btab083. btab083.

[33] Jethro S. Johnson, Daniel J. Spakowicz, Bo-Young Hong, Lauren M. Petersen, Patrick Demkowicz, Lei Chen, Shana R. Leopold, Blake M. Hanson, Hanako O. Agresta, Mark Gerstein, Erica Sodergren, and George M. Weinstock. Evaluation of 16s rRNA gene sequencing for species and strain-level microbiome analysis. *Nature Communications*, 10(1), November 2019. doi: 10.1038/s41467-019-13036-1. URL https://doi.org/10.1038/s41467-019-13036-1.

[34] Lisa Joos, Stien Beirinckx, Annelies Haegeman, Jane Debode, Bart Vandecasteele, Steve Baeyen, Sofie Goormachtig, Lieven Clement, and Caroline De Tender. Daring to be differential: metabarcoding analysis of soil and plant-related microbial communities using amplicon sequence variants and operational taxonomical units. *BMC Genomics*, 21(1), October 2020. doi: 10.1186/s12864-020-07126-4. URL https://doi.org/10.1186/s12864-020-07126-4.

[35] A. Jović, K. Brkić, and N. Bogunović. A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205, 2015. doi: 10.1109/MIPRO.2015.7160458.

[36] M. Kanehisa and S. Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, Jan 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.27. URL https://pubmed.ncbi.nlm.nih.gov/10592173. 10592173[pmid].

[37] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

[38] Rob Knight, Alison Vrbanac, Bryn C. Taylor, Alexander Aksenov, Chris Callewaert, Justine Debelius, Antonio Gonzalez, Tomasz Kosciolek, Laura-Isobel McCall, Daniel McDonald, Alexey V. Melnik, James T. Morton, Jose Navas, Robert A. Quinn, Jon G. Sanders, Austin D. Swafford, Luke R. Thompson, Anupriya Tripathi, Zhenjiang Z. Xu, Jesse R. Zaneveld, Qiyun Zhu, J. Gregory Caporaso, and Pieter C. Dorrestein. Best practices for analysing microbiomes. *Nature Reviews Microbiology*, 16(7):410–422, May 2018. doi: 10.1038/s41579-018-0029-9. URL https://doi.org/10.1038/s41579-018-0029-9.

[39] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *CoRR*, abs/1804.10959, 2018. URL http://arxiv.org/abs/1804.10959.

[40] Taku Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*, 2018.

[41] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining, 2019.

[42] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. Flaubert: Unsupervised language model pre-training for french, 2020.

[43] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[44] Wang Liang. Segmenting DNA sequence into words based on statistical language model. *Nature Precedings*, February 2012. doi: 10.1038/npre.2012.6939.1. URL `https://doi.org/10.1038/npre.2012.6939.1`.

[45] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

[46] David R Lovell, Xin-Yi Chua, and Annette McGrath. Counts: an outstanding challenge for log-ratio analysis of compositional data in the molecular biosciences. *NAR Genomics and Bioinformatics*, 2(2), 06 2020. ISSN 2631-9268. doi: 10.1093/nargab/lqaa040. URL `https://doi.org/10.1093/nargab/lqaa040`. lqaa040.

[47] Laura Judith Marcos-Zambrano, Kanita Karaduzovic-Hadziabdic, Tatjana Loncar Turukalo, Piotr Przymus, Vladimir Trajkovik, Oliver Aasmets, Magali Berland, Aleksandra Gruca, Jasminka Hasic, Karel Hron, Thomas Klammsteiner, Mikhail Kolev, Leo Lahti, Marta B. Lopes, Victor Moreno, Irina Naskinova, Elin Org, Inês Paciência, Georgios Papoutsoglou, Rajesh Shigdel, Blaz Stres, Baiba Vilne, Malik Yousef, Eftim Zdravevski, Ioannis Tsamardinos, Enrique Carrillo de Santa Pau, Marcus J. Claesson, Isabel Moreno-Indias, and Jaak Truu. Applications of machine learning in human microbiome studies: A review on feature selection, biomarker identification, disease prediction and treatment. *Frontiers in Microbiology*, 12, February 2021. doi: 10.3389/fmicb.2021.634511. URL `https://doi.org/10.3389/fmicb.2021.634511`.

[48] S. McGinnis and T. L. Madden. BLAST: at the core of a powerful and diverse set of sequence analysis tools. *Nucleic Acids Research*, 32(Web Server):W20–W25, July 2004. doi: 10.1093/nar/gkh435. URL `https://doi.org/10.1093/nar/gkh435`.

[49] Romain Menegaux and Jean-Philippe Vert. Continuous embeddings of DNA sequencing reads and application to metagenomics. *Journal of Computational Biology*, 26(6):509–518, 2019. doi: 10.1089/cmb.2018.0174. URL https://doi.org/10.1089/cmb.2018.0174. PMID: 30785347.

[50] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL http://arxiv.org/abs/1301.3781.

[51] Mohamed Mysara, Peter Vandamme, Ruben Props, Frederiek-Maarten Kerckhof, Natalie Leys, Nico Boon, Jeroen Raes, and Pieter Monsieurs. Reconciliation between operational taxonomic units and species boundaries. *FEMS Microbiology Ecology*, 93(4), March 2017. doi: 10.1093/femsec/fix029. URL https://doi.org/10.1093/femsec/fix029.

[52] 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, March 1970. doi: 10.1016/0022-2836(70)90057-4. URL https://doi.org/10.1016/0022-2836(70)90057-4.

[53] Andrew Y. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 404–412, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

[54] Andrew Y. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015435. URL https://doi.org/10.1145/1015330.1015435.

[55] Patrick Ng. dna2vec: Consistent vector representations of variable-length k-mers, 2017.

[56] Nam-Phuong Nguyen, Tandy Warnow, Mihai Pop, and Bryan White. A perspective on 16s rRNA operational taxonomic unit clustering using sequence similarity. *npj Biofilms and Microbiomes*, 2(1), April 2016. doi: 10.1038/npjbiofilms.2016.4. URL https://doi.org/10.1038/npjbiofilms.2016.4.

[57] Mai Oudah and Andreas Henschel. Taxonomy-aware feature engineering for microbiome classification. *BMC Bioinformatics*, 19(1), June 2018. doi: 10.1186/s12859-018-2205-3. URL https://doi.org/10.1186/s12859-018-2205-3.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[59] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

[60] Mariana Buongermino Pereira, Mikael Wallroth, Viktor Jonsson, and Erik Kristiansson. Comparison of normalization methods for the analysis of metagenomic gene abundance data. *BMC Genomics*, 19(1), April 2018. doi: 10.1186/s12864-018-4637-6. URL https://doi.org/10.1186/s12864-018-4637-6.

[61] David M. W. Powers. Applications and explanations of Zipf's law. In *New Methods in Language Processing and Computational Natural Language Learning*, 1998. URL https://www.aclweb.org/anthology/W98-1218.

[62] Andrei Prodan, Valentina Tremaroli, Harald Brolin, Aeilko H. Zwinderman, Max Nieuwdorp, and Evgeni Levin. Comparing bioinformatic pipelines for microbial 16s rRNA amplicon sequencing. *PLOS ONE*, 15(1):e0227434, January 2020. doi: 10.1371/journal.pone.0227434. URL https://doi.org/10.1371/journal.pone.0227434.

[63] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. Bpe-dropout: Simple and effective subword regularization. In *ACL*, 2020.

[64] Christian Quast, Elmar Pruesse, Pelin Yilmaz, Jan Gerken, Timmy Schweer, Pablo Yarza, Jörg Peplies, and Frank Oliver Glöckner. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Research*, 41(D1):D590–D596, 11 2012. ISSN 0305-1048. doi: 10.1093/nar/gks1219. URL https://doi.org/10.1093/nar/gks1219.

[65] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[66] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019. URL https://openai.com/blog/better-language-models/.

[67] Timothy W. Randolph, Sen Zhao, Wade Copeland, Meredith Hullar, and Ali Shojaie. Kernel-penalized regression for analysis of microbiome data. *The Annals of Applied Statistics*, 12(1), March 2018. doi: 10.1214/17-aoas1102. URL https://doi.org/10.1214/17-aoas1102.

[68] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John F. Canny, Pieter Abbeel, and Yun S. Song. Evaluating protein transfer learning with TAPE. *CoRR*, abs/1906.08230, 2019. URL http://arxiv.org/abs/1906.08230.

[69] Jane B. Reece and Neil A. Campbell. *Campbell biology*. Pearson Benjamin Cummings, San Francisco, 2010.

[70] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *IN PROCEEDINGS OF THE LREC 2010 WORKSHOP ON NEW CHALLENGES FOR NLP FRAMEWORKS*, pages 45–50, 2010.

[71] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. URL `http://arxiv.org/abs/1908.10084`.

[72] Veronica Ricci, Davide Carcione, Simone Messina, Gualtiero I. Colombo, and Yuri D'Alessandra. Circulating 16s rna in biofluids: Extracellular vesicles as mirrors of human microbiome? *International Journal of Molecular Sciences*, 21 (23), 2020. ISSN 1422-0067. doi: 10.3390/ijms21238959. URL `https://www.mdpi.com/1422-0067/21/23/8959`.

[73] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 2020. doi: 10.1101/622803. URL `https://www.biorxiv.org/content/early/2020/08/31/622803`.

[74] Henriette L. Røder, Jakob Herschend, Jakob Russel, Michala F. Andersen, Jonas S. Madsen, Søren J. Sørensen, and Mette Burmølle. Enhanced bacterial mutualism through an evolved biofilm phenotype. *The ISME Journal*, 12(11):2608–2618, July 2018. doi: 10.1038/s41396-018-0165-2. URL `https://doi.org/10.1038/s41396-018-0165-2`.

[75] Torbjørn Rognes, Tomáš Flouri, Ben Nichols, Christopher Quince, and Frédéric Mahé. VSEARCH: a versatile open source tool for metagenomics. *PeerJ*, 4: e2584, October 2016. doi: 10.7717/peerj.2584. URL `https://doi.org/10.7717/peerj.2584`.

[76] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL `https://www.aclweb.org/anthology/P16-1162`.

[77] E. Stackebrandt. Report of the ad hoc committee for the re-evaluation of the species definition in bacteriology. *INTERNATIONAL JOURNAL OF SYSTEMATIC AND EVOLUTIONARY MICROBIOLOGY*, 52(3):1043–1047, May 2002. doi: 10.1099/ijs.0.02360-0. URL `https://doi.org/10.1099/ijs.0.02360-0`.

[78] E. Stackerbrandt and B. M. Goebel. Taxonomic note: A place for DNA-DNA reassociation and 16s rRNA sequence analysis in the present species definition in bacteriology. *International Journal of Systematic and Evolutionary Microbiology*, 44(4):846–849, October 1994. doi: 10.1099/00207713-44-4-846. URL https://doi.org/10.1099/00207713-44-4-846.

[79] E. Stackerbrandt and B. M. Goebel. Taxonomic note: A place for DNA-DNA reassociation and 16s rRNA sequence analysis in the present species definition in bacteriology. *International Journal of Systematic and Evolutionary Microbiology*, 44(4):846–849, October 1994. doi: 10.1099/00207713-44-4-846. URL https://doi.org/10.1099/00207713-44-4-846.

[80] Alexander Statnikov, Mikael Henaff, Varun Narendra, Kranti Konganti, Zhiguo Li, Liying Yang, Zhiheng Pei, Martin J Blaser, Constantin F Aliferis, and Alexander V Alekseyenko. A comprehensive evaluation of multicategory classification methods for microbiomic data. *Microbiome*, 1(1), April 2013. doi: 10.1186/2049-2618-1-11. URL https://doi.org/10.1186/2049-2618-1-11.

[81] Reed M. Stubbendieck, Carol Vargas-Bautista, and Paul D. Straight. Bacterial communities: Interactions to scale. *Frontiers in Microbiology*, 7:1234, 2016. ISSN 1664-302X. doi: 10.3389/fmicb.2016.01234. URL https://www.frontiersin.org/article/10.3389/fmicb.2016.01234.

[82] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[83] Johan Suykens. Support vector machines: A nonlinear modelling and control perspective. *European Journal of Control - EUR J CONTROL*, 7:311–327, 12 2001. doi: 10.3166/ejc.7.311-327.

[84] Pajau Vangay, Benjamin M Hillmann, and Dan Knights. Microbiome learning repo (ML repo): A public repository of microbiome regression and classification tasks. *GigaScience*, 8(5), April 2019. doi: 10.1093/gigascience/giz042. URL https://doi.org/10.1093/gigascience/giz042.

[85] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[86] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://www.aclweb.org/anthology/W18-5446.

[87] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020.

[88] Qiong Wang, George M. Garrity, James M. Tiedje, and James R. Cole. Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, 73(16):5261–5267, August 2007. doi: 10.1128/aem.00062-07. URL `https://doi.org/10.1128/aem.00062-07`.

[89] Qiong Wang, George M. Garrity, James M. Tiedje, and James R. Cole. Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, 73(16):5261–5267, August 2007. doi: 10.1128/aem.00062-07. URL `https://doi.org/10.1128/aem.00062-07`.

[90] Chongqing Wen, Liyou Wu, Yujia Qin, Joy D. Van Nostrand, Daliang Ning, Bo Sun, Kai Xue, Feifei Liu, Ye Deng, Yuting Liang, and Jizhong Zhou. Evaluation of the reproducibility of amplicon sequencing with illumina MiSeq platform. *PLOS ONE*, 12(4):e0176716, April 2017. doi: 10.1371/journal.pone.0176716. URL `https://doi.org/10.1371/journal.pone.0176716`.

[91] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[92] Stephen Woloszynek, Zhengqiao Zhao, Jian Chen, and Gail L. Rosen. 16s rRNA sequence embeddings: Meaningful numeric feature representations of nucleotide sequences that are convenient for downstream analyses. *PLOS Computational Biology*, 15(2):1–25, 02 2019. doi: 10.1371/journal.pcbi.1006721. URL `https://doi.org/10.1371/journal.pcbi.1006721`.

[93] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, October 1996. doi: 10.1162/neco.1996.8.7.1341. URL `https://doi.org/10.1162/neco.1996.8.7.1341`.

[94] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[95] Ziyi Yang, Chenguang Zhu, and Weizhu Chen. Parameter-free sentence embedding via orthogonal basis, 2019.

[96] Joseph P. Zackular, Mary A.M. Rogers, Mack T. Ruffin, and Patrick D. Schloss. The human gut microbiome as a screening tool for colorectal cancer. *Cancer Prevention Research*, 7(11):1112–1121, August 2014. doi: 10.1158/1940-6207.capr-14-0129. URL https://doi.org/10.1158/1940-6207.capr-14-0129.

[97] Joseph P. Zackular, Nielson T. Baxter, Grace Y. Chen, and Patrick D. Schloss. Manipulation of the gut microbiota reveals role in colon tumorigenesis. *mSphere*, 1(1), February 2016. doi: 10.1128/msphere.00001-15. URL https://doi.org/10.1128/msphere.00001-15.

[98] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.

[99] Georg Zeller, Julien Tap, Anita Y Voigt, Shinichi Sunagawa, Jens Roat Kultima, Paul I Costea, Aurélien Amiot, Jürgen Böhm, Francesco Brunetti, Nina Habermann, Rajna Hercog, Moritz Koch, Alain Luciani, Daniel R Mende, Martin A Schneider, Petra Schrotz-King, Christophe Tournigand, Jeanne Tran Van Nhieu, Takuji Yamada, Jürgen Zimmermann, Vladimir Benes, Matthias Kloor, Cornelia M Ulrich, Magnus Knebel Doeberitz, Iradj Sobhani, and Peer Bork. Potential of fecal microbiota for early-stage detection of colorectal cancer. *Molecular Systems Biology*, 10(11):766, November 2014. doi: 10.15252/msb.20145645. URL https://doi.org/10.15252/msb.20145645.

[100] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[101] Yi-Hui Zhou and Paul Gallins. A review and tutorial of machine learning methods for microbiome host trait prediction. *Frontiers in Genetics*, 10:579, 2019. ISSN 1664-8021. doi: 10.3389/fgene.2019.00579. URL https://www.frontiersin.org/article/10.3389/fgene.2019.00579.