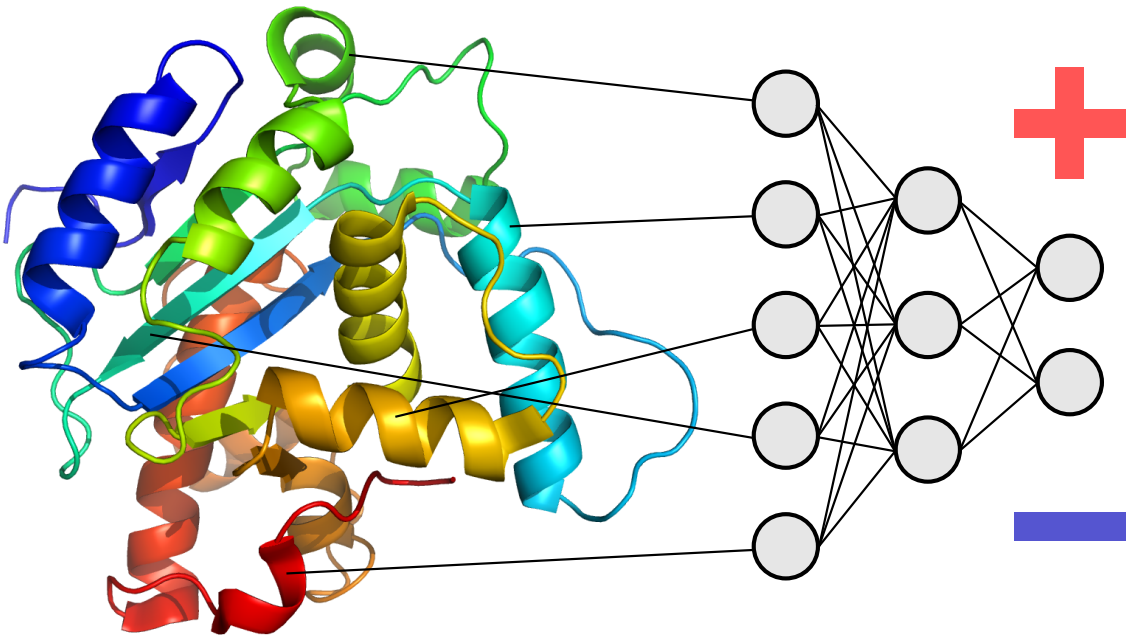

BL(B)ERT: Predicting Antibiotic Resistance with a Language AI Model

Applying a BERT Language Model to Predict Antibiotic Resistance Within Beta-lactamase/transpeptidase-like Proteins

Master's thesis in Complex Adaptive Systems

MATHIAS ÖRTENBERG TOFTÅS

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

BL(B)ERT: Predicting Antibiotic Resistance with a Language AI Model

Applying a BERT Language Model to Predict Antibiotic Resistance Within Beta-lactamase/transpeptidase-like Proteins

MATHIAS ÖRTENBERG TOFTÅS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2022

BL(B)ERT: Predicting Antibiotic Resistance with a Language AI Model
Applying a BERT Language Model to Predict Antibiotic Resistance Within Beta-lactamase/transpeptidase-like Proteins
MATHIAS ÖRTENBERG TOFTÅS

Supervisor: Erik Kristiansson, Chalmers, Mathematical Sciences
Examiner: Erik Kristiansson, Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Figure illustrating the classification of a protein with an AI model, the protein in the figure is used under a Creative Commons Attribution-Share Alike license. The figure has been modified, and the original can be found here [1].

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

BL(B)ERT: Predicting Antibiotic Resistance with a Language AI Model
Applying a BERT Language Model to Predict Antibiotic Resistance Within Beta-lactamase/transpeptidase-like Proteins
MATHIAS ÖRTENBERG TOFTÅS
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The effectiveness of utilizing a BERT language AI model to predict antibiotic resistance within beta-lactamase/transpeptidase-like proteins is tested. The performance of the model is compared to a traditional Hidden Markov Model (HMM) to verify its capabilities. To further ensure the models capabilities, several tests such as cross-validation and resistance towards sequence read errors are performed. We find that the BERT model outperforms the HMM model on amino acid sequences with lengths around 40-50 and shorter. These are the type of sequence lengths which we expect to encounter in our use case. For longer sequences, the HMM model is preferable as it requires less computation time. We also find that the BERT model and the HMM model has learned different aspects about the data, allowing the combination of the results of both methods to achieve even finer results.

Keywords: Antibiotic Resistance, BERT, Beta-lactamase/transpeptidase-like, Hidden Markov Model.

Acknowledgements

I would like to thank my supervisor Erik Kristiansson and his colleagues Fanny Berglund and Jaun Salvador Inda Diaz for their continuous support and help during this spring. Without their subject area expertise and insights into what would be interesting to delve deeper into, this project would likely not have gone further than the training of the model. I would also like to thank David Lund for his work with procuring the data used in this work.

Mathias Örtenberg Toftås, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BERT	Bidirectional Encoder Representations from Transformers
HMM	Hidden Markov Model
FFN	Feed Forward Neural Network
ROC	Receiver Operating Characteristic
CLS	Class Token
MSK	Mask Token
PAD	Padding Token
CPU	Central Processing Unit
GPU	Graphical Processing Unit
DNA	Deoxyribonucleic Acid
mRNA	Messenger Ribonucleic Acid
tRNA	Transfer Ribonucleic Acid

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Parameters

d_{model}	Hidden Dimension Size
l_{max}	Maximum Sequence Length
h	Number of Attention Heads
$d_{self-attention}$	Self-Attention Internal Size
d_{FFN}	FFN Internal Size
d_{depth}	Number of Stacked Encoders
r_{drop}	Dropout Rate

Contents

List of Acronyms **ix**

Nomenclature **xi**

List of Figures **xv**

List of Tables **xix**

1 Introduction **1**

2 Theory **3**
2.1 Proteins Synthesis and DNA Sequencing 3
2.2 Protein Function 5
2.3 The Model 6
2.3.1 Model Selection 6
2.3.2 Transformers 7

3 Methods **11**
3.1 The Data Sets 11
3.2 The BERT Model 12

4 Results **19**
4.1 Comparing with Previous Methods 19
4.2 Testing Training Transferability 25
4.3 Cross-Validating the Model 26
4.4 Timing the Model 27
4.5 Glimpsing into the Model 27
4.6 Testing on Metegenomic Reads 30
4.7 Discussion and Conclusion 32

Bibliography **35**

List of Figures

2.1 Graphic representing protein synthesis. 3
2.2 Translation of codons into amino acids. To arrive at the corresponding amino acid, begin at the center of the figure with the first nucleic acid of the codon, then follow the graphic outwards. 4
2.3 Figure demonstrating a proteins structure and its active site. The figure text font has been altered from the original, and a link to the original can be found here [8]. 5
2.4 Figure demonstrating how different amino acid sequences can be folded to have similar shapes. In the figure, the blue blocks represent the amino acids part of the active site. The 'folding' in the figure consists of taking segments of 3 amino acids and rotating them freely to achieve the same result. 6
2.5 A single Encoder block. The Encoder consists of two sub units, the Multi-Head Attention and the Feed Forward network. Each sub units output is subjected to dropout (not shown) added to the original input and then layer normalized before being feed forward. 8
2.6 The components within the Encoder. Not shown in the figure is the dropout that occurs on the output of the softmax layer before the final matrix multiplication in the Scaled Dot-Product Attention. . . . 9

3.1 Visual representation of the various data sets used in this work. **a** represents all bacterial DNA, **b** presents our specific family of bacteria, **c** represents DNA within this family which has been labelled by using a HMM model, and **d** represents our set of true labelled DNA. . 11
3.2 Visual representation grouping of data to perform the cross-validation. The dark orange blocks represent the original validation data, the yellow blocks represent the original training data. The pluses and minuses represent the sets of antibiotic resistant positive proteins and negative ones. 12
3.3 The embedding of sequences into vectors, the embedding takes both the token and the position into account. 13
3.4 The pre-training step. Every sequence gets modified by masking or changing some of the input tokens, the model is then tasked to predict the original token. 13

3.5 Pre-training runs performed whilst varying a single parameter, these runs ended up taking on the order of 1/20th the time it took for the final pre-training. 15
3.6 The loss function as it evolves over time during the pre-training, the shift from periodic to non periodic behavior around batch 300000 is due to a change from fixed to random batch order. 16
3.7 A graphic representation of the fine-tuning procedure. 17
3.8 The loss and accuracy of the fine-tuning. 17

4.1 **a)** ROC curve comparing the predictive capabilities of the BERT and HMM models. The curve shown is averaged across all sequence lengths (10-63). **b)** The area under the ROC curve as a function of the protein sequence length. 20
4.2 A variety of metrics comparing the BERT and HMM models for different sequence lengths. The optimal cutoff threshold used was obtained by selecting the threshold corresponding to the point of the ROC curve (Fig. 4.1) closest to the upper left corner. 21
4.3 The figures show the distribution of proteins in the 2D space of BERT and HMM scores. The dashed lines correspond to the cutoff thresholds of the two models, and the blue and red distributions correspond to the negative and positive proteins respectively. All figures are shown in the same linear scale. **a)** Shows sequence lengths between 10 and 17, **b)** shows between 18 and 35, **c)** shows between 36 and 53, and **d)** shows for all sequence lengths. 22
4.4 The corresponding 1D distributions to those shown in Fig. 4.3. The left column shows the distributions along the HMM axis and the right shows them along the BERT axis. As in the previous figure, **a)** shows the short sequences, **b)** the medium ones, **c)** the long ones, and **d)** shows all lengths. Again, the red distribution shows the positive proteins whilst the blue shows the negative proteins. 23
4.5 The degree of separation between the two distributions in the sense of measuring the area of overlap between the two distributions. For the case of a perfect classifier the separation would be 1, meaning no overlap, whilst a separation of 0 means that the two distributions overlap completely. 24
4.6 **a)** ROC showing the performance of the BERT and HMM models when the proteins have had random errors introduced. The ROC shown has been averaged across all chosen error rates. **b)** The area under the ROC for different error rates. 24
4.7 Various metrics to compare the BERT and HMM model when operating on proteins with introduced read errors. 25
4.8 ROC showing the difference between the standard model and the cross-validated one. The left figure shows the ROC curves for sequences of various amino acid lengths, and the right figure shows the area beneath the curves to show how the models perform for various sequence lengths. 26

4.9 Various metrics comparing the standard and cross-validated versions of the BERT model. 27
4.10 The graphs show how important each part of the protein is in order to classify it as antibiotic resistant. Each protein shown is a known antibiotic resistant protein. The dashed line shows the optimal threshold, and the graphs for each model are aligned such that their thresholds overlap. 28
4.11 PCA plots of the [CLS] token output of the BERT model. The left column shows the two distributions (red for positive and blue for negative proteins) when the model is correct, whilst the right column shows when the model is incorrect. The top row shows the BERT models classification and the bottom row shows the same for the HMM model. All plots are made in the same linear scale. 29
4.12 The corresponding distributions to the 2D distributions in Fig. 4.11. . 30
4.13 Graph showing the model output on unlabeled metagenomic reads. The insert shows the sequence length distribution. The dashed lines shows the cutoff thresholds obtained previously. The samples shown are three different metegenomic samples from the human gut. The vast majority of reads scored below the HMM cutoff, these are not included in the figure 31
4.14 More metegenomic reads, this time from waste water treatment plants from various countries, **a)** is from India, **b)** Senegal, and **c)** Sweden. . 32

List of Tables

3.1 The selected model parameters. 14

1
Introduction

According to the World Health Organization, "antibiotic resistance is one of the biggest threats to global health, food security, and development today" [2]. Further, the American government agency Centers for Disease Control and Prevention attribute nearly 50000 preventable deaths due to antibiotic resistance in 2019 [3]. We can extrapolate that figure to encompass the entire world and arrive at 1.2 million preventable deaths which agrees with values found by other studies [4]. Comparing this to the most common causes of death, such as stroke with roughly 6 million deaths per year, we find that it is within the same order of magnitude [5]. With this in mind it should be clear why any kind of research into antibiotic resistance is of value.

A particular problem when it comes to researching antibiotic resistance lies in acquiring samples of the bacterial genes that give rise to antibiotic resistance. The issue is two fold: firstly, it is not clear which part of a bacteria's DNA which gives rise to its antibiotic resistance, secondly, we wish to identify these genes before its corresponding bacteria becomes so wide spread that identification becomes trivial. There exists several traditional methods that perform this, but these usually depend on performing sequence alignments which limits the kind of genes they can discover.

In recent times, the application of various AI methods within the field of molecular biology has become both popular and widely successful. These methods have shown the capability of performing feats previously thought computationally nearly impossible such as AlphaFold's capabilities within protein folding [6]. They are also known for their abilities to learn general truths about the data which cannot be neatly written down in simple formulas. Thus, it seems prudent to see if any such AI method is capable of identifying antibiotic genes.

However, AI methods generally require large data sets to train. This is a big problem as we lack such large data sets and it is precisely for this reason why we wish to utilize AI methods; we wish to create these data sets by identifying new DNA sequences. Fortunately, there exists models which are capable of pre-training on unlabelled data which only require a small amount of subject matter specific data for the final fine-tuning.

A suitable AI model for our purposes are language models. These models are designed to take a sentence as input and perform some kind of prediction. They are suitable as one can imagine the similarities between sentences and the grammar of

language and amino acid sequences and the grammar of proteins. For our cases we have elected to use a recent language model known as the Bidirectional Encoder Representations from Transformers model (BERT) [11]. This model also offers the greatly needed capability of being pre-trainable on unlabelled data.

The purpose of this project is then to determine whether such an AI model is a suitable choice for the identification of antibiotic microbial DNA. We will compare its performance to that of a traditional method currently used by our supervisor known as a Hidden Markov Model. If the model proves successful, the hope is that our supervisor will be able to use said model in future work.

2

Theory

2.1 Proteins Synthesis and DNA Sequencing

In this work we mainly focus on the computational aspect of detecting certain proteins, thus the area specific knowledge regarding protein structures and similar is not required to understand this thesis. Therefore we will only cover the very basics required to understand this work. Proteins perform a vast array of different tasks such as catalysing reactions, transporting molecules, giving structure to cells, and much more. The function of a protein is determined by is structure which in turn is coded by DNA. Proteins are created in a process known as protein synthesis. This process begins when an RNA polymerase enzyme connects with the DNA at a transcription start site. Here it begins by unraveling the two strands making up the DNA helix and begins reading and transcribing the DNA onto a new strand known as mRNA. It continues doing this until it hits a stop codon, signaling that its has transcribed a complete sub unit of the DNA. The process continues later one once the mRNA has made contact with a ribosome, here the mRNA strand is read and each codon corresponds to one amino acid which is brought in by the tRNA. The ribosome connects each protein in order, building up the protein sequentially. This process can be seen in Fig. 2.1

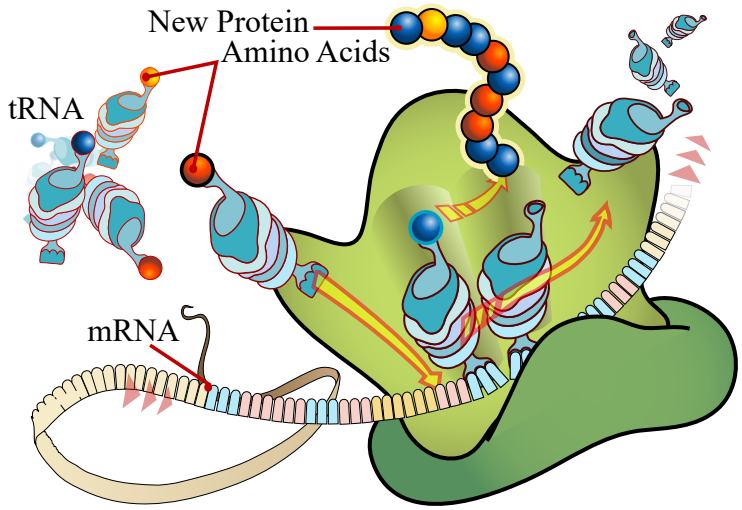

Figure 2.1: Graphic representing protein synthesis.

The most important parts of the above process for our use case is that proteins are coded into DNA in enclosed segments separated from the rest of the DNA by two stop codons. Thus if we wish to study specific proteins, it is possible to do so by looking at the underlying DNA which constructs the proteins. This has been quite successful in recent years with new AI techniques such as Alpha Fold able to reconstruct the protein structure from the amino acid sequence [6]. In order to do this, however, we must also translate the DNA sequence into the corresponding amino acid sequence. Thankfully, this translation is known and we can simple look the up in a table such as Fig. 2.2.

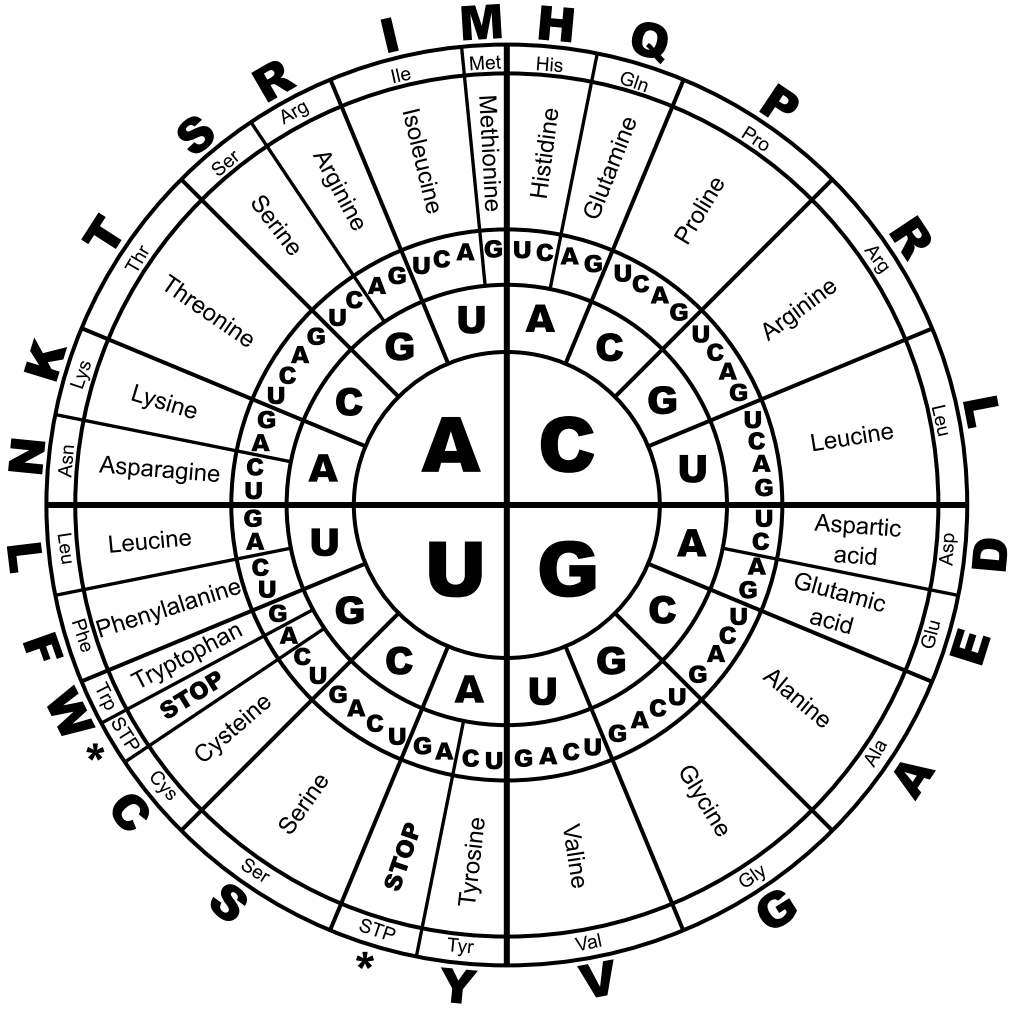

Figure 2.2: Translation of codons into amino acids. To arrive at the corresponding amino acid, begin at the center of the figure with the first nucleic acid of the codon, then follow the graphic outwards.

Unfortunately, since DNA exists in the real world, we must also contend with the practical issue of actually sequencing the DNA. There exists several methods that achieve this, but one thing that these methods all share in commons is that they introduce errors in the form of read errors. These errors are usually manageable, such as the popular method of Illumina sequencing which tends to produce error rates below 1% [9]. We must note however that these error rates pertain to the nucleic

acids and not the amino acids. For our use case, simply one nucleic acid being read incorrectly is enough to change an amino acid into another. Thus we find that the probability of all nucleic acids in a codon being read correctly is $0.99^3 \approx 0.97$, giving us an error rate of 3%. Because of this it would be prudent to keep in mind that any method proposed to be used in combination with proteins should be able to contend with such errors.

2.2 Protein Function

There exists several methods with which bacteria gain antibiotic resistance. Some of these include the bacteria developing new processes which avoid using the the antibiotics targets, creating enzymes that break down the antibiotic, changing the entry ways into the bacteria to stop the antibiotic from entering the cell, and even pumping the antibiotics out from the cell [7]. Of these methods we only tackle the creation of enzymes since this only requires a single gene to perform.

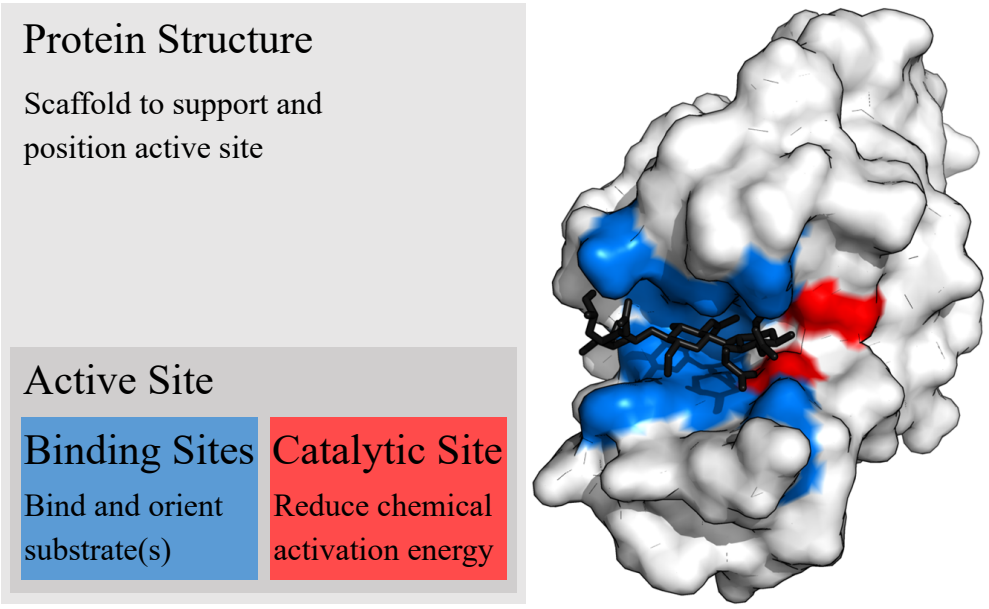

Figure 2.3: Figure demonstrating a proteins structure and its active site. The figure text font has been altered from the original, and a link to the original can be found here [8].

As with all proteins, it is the structure of enzymes which determines its function. Even more specifically for enzymes, it is the active site that determines what happens when an enzyme interacts with other molecules. So long as the active site retains the same or an equivalent shape, and the rest of the enzyme does not block the target molecule, the enzyme will continue to function even if the other parts of the enzyme are changed or mutated. This can be seen in Fig. 2.3.

2.3 The Model

2.3.1 Model Selection

To determine which AI model to apply, we must first understand our data. The data that we have consists of sequences of amino acids which represent our proteins. We know since earlier that one of the most important parts of the protein is the active site itself, therefore our model should somehow try to capture these features in the data. A naive method could try to identify the parts of the sequence that corresponds to the active site. The issue with such a method lies in the folding of proteins upon their creating. Because the proteins are folded into 3D space, the distance between amino acids in the 1D sequence does not correspond to the actual distance between them in 3D. Thus we expect the amino acids that make up the active site to be separated in the sequence. Even further, it is possible to construct the same or a similar active site with wildly varying amino acid sequences. A toy demonstration of this can be seen in Fig. 2.4 This informs us that our model needs to be able to take the entire sequence into account at once. Thus we can immediately rule out any method that traverses the sequence as these cannot capture relations between amino acids that are seemingly distant in the 1D sequence. Another issue to take into consideration is that the length of the sequences also varies, thus the model must also be able to handle this.

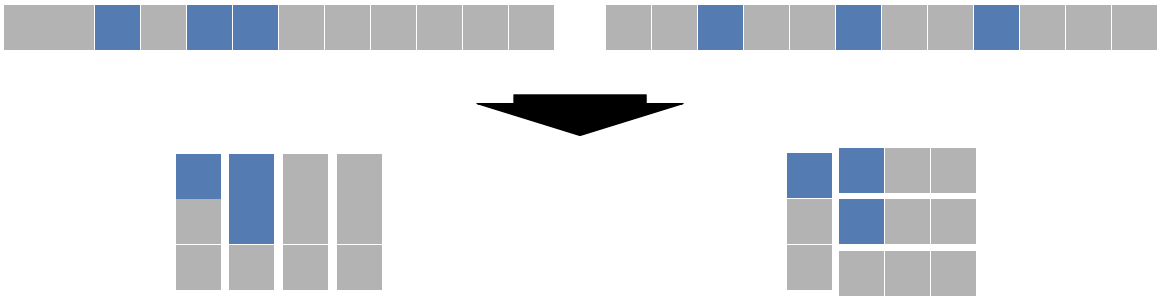

Figure 2.4: Figure demonstrating how different amino acid sequences can be folded to have similar shapes. In the figure, the blue blocks represent the amino acids part of the active site. The 'folding' in the figure consists of taking segments of 3 amino acids and rotating them freely to achieve the same result.

Taking these thoughts into consideration, we turn to language models. These models are built to understand human language. It might seem odd at first glance, but sentences share a lot in common with our amino acid sequences. They both contain tokens (words or amino acids) which are in some way connected to other tokens where the distance between the tokens in the sequence is not necessarily important, and they can both vary in length. Further, these models are often used for sentiment analysis, that is classifying sentences into two or more groups, which is similar to classifying proteins into antibiotic resistant or not. From this it should be clear why a language model would make a good candidate for dealing with proteins. Another aspect we should take into consideration is that the set of known antibiotic genes

within our family of proteins is rather small. Therefore, we require a model that somehow can contend with small data sets. A quick search into language AI model and DNA or proteins yields several methods but the most commonly represented one is the relatively new BERT model [11].

The BERT model is a natural language processing model developed by Google in 2018. This model is essentially a normal transformer but with a specific training regimen. The model meets our set out targets by employing what is known as self attention. This mechanism in essence performs a vector and transposed vector into matrix multiplication that creates a score between each pair of tokens in the input. This means that the model is distance agnostic and able to see distant connections in the input data. To account for the varying sequence length the model has a maximum input length and employs padding to fill out all inputs. This is of course one downside which we must accept since we cannot use methods that traverse the sequence. Finally, the BERT model is also trained in two steps, a pre-training step where it trains on unlabelled data to learn the grammar of language (or proteins in our case), and a fine-tuning step where we introduce the subject specific data to classify. The idea behind this is that with the pre-training, the model will learn general ideas about the language which helps it with classifying latter on. In layman's terms this entails that it requires less labelled training data. Since this model is otherwise highly acclaimed and suitable for our needs, we elected to go forward using it.

2.3.2 Transformers

The most integral part of the BERT model is the transformer. This machine learning model was first introduced in the paper *Attention Is All You Need* in 2017 [10]. The original paper does a good job of introducing and explaining their model, and for a deeper look into how it works it is a recommended read. For our case, we will lightly go other the model and its components to give name to and understanding of the different model parameters which we have to select and contend with. To begin let us view a single Encoder, this can be seen in Fig. 2.5. The input to the Encoder is copied three times creating the Keys, Values, and Queries, which are feed into the Multi-Head Attention. The output of this layer is then added to the original input and then layer normalized. The output of this operation is then feed to a Feed Forward Network, which also performs a similar residual connection and layer normalization. Since we keep a residual connection, the output of each sub unit must be same as the input, which is the Hidden Dimension Size or d_{model}. We will also note that we make use of dropout layers before each residual addition, but these are not shown in the figure. This gives an equation for each sub layer as such $\text{LayerNorm}(x + \text{DropOut}(\text{Sublayer}(x)))$.

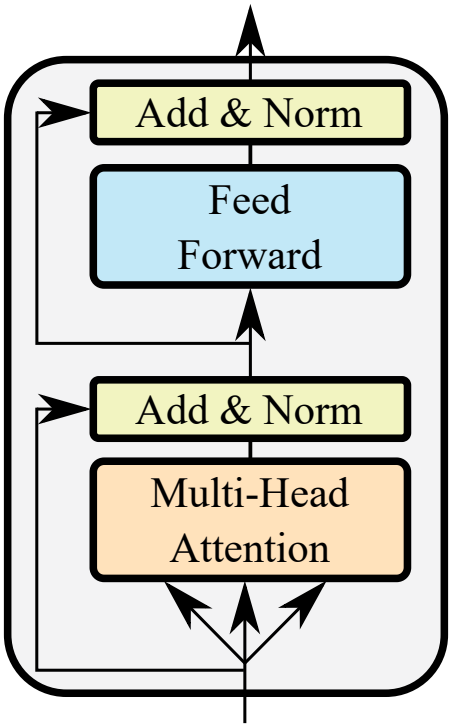

Figure 2.5: A single Encoder block. The Encoder consists of two sub units, the Multi-Head Attention and the Feed Forward network. Each sub units output is subjected to dropout (not shown) added to the original input and then layer normalized before being feed forward.

The most important part of the Encoder is the Multi-Head Attention block, which can be seen in Fig. 2.6. In the figure each head corresponds to the several partially see through layers and the number of heads is denominated by h. The Keys, Values, and Queries are all feed into individual linear layers which project the inputs into the Self-attention Internal Dimension Size $d_{self-attention}$ before being inputted into the Scaled Dot-Product Attention. This block can be seen to the left in the figure, in which we perform a series of simply operations. In essence the Keys and Queries are used to create an attention matrix which denotes how important each token in the input is to one another, finally this attention matrix is subjected to dropout before being multiplied with the unchanged Values input to produce the output. After this the output of all heads is concatenated and feed into a linear layer which projects the dimension down into the original input dimension.

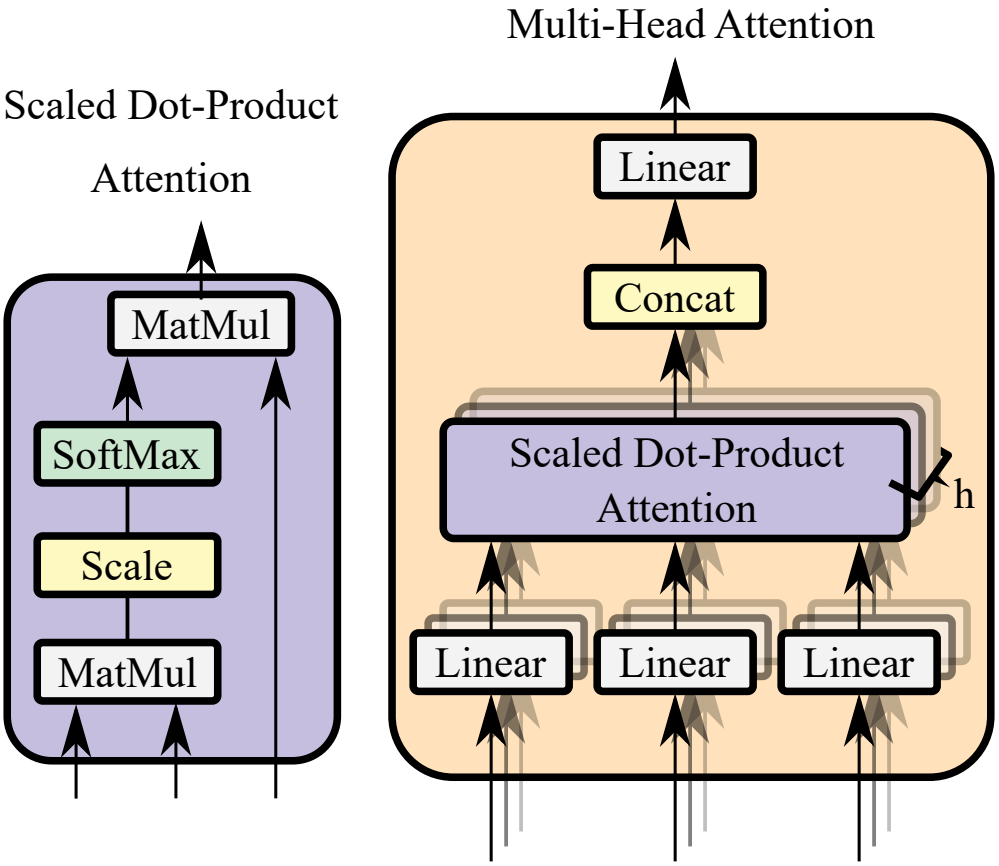

Figure 2.6: The components within the Encoder. Not shown in the figure is the dropout that occurs on the output of the softmax layer before the final matrix multiplication in the Scaled Dot-Product Attention.

With all of these various parts it is a good idea to clarify the parameters. The Hidden Dimension Size or d_{model} is the embedding size of the tokens and the consistent dimension to which the data is projected to through a linear layer after each sublayer in the model. The Maximum Sequence Length, or l_{max}, is the longest sequence which we can feed the model, and since one of these position will always be taken by the [CLS] token, the actual maximum length for the amino acid sequences is 63. The Number of Attention Heads, or h, is the number of parallel layers within the Multi-Head Attention sublayer. The Self-Attention Internal Size, or $d_{self-attention}$, is the dimension which the first linear layers inside the Multi-Head Attention sublayer project the input data to. The FFN Internal Size, or d_{FFN}, is the size which the first linear layer in the FFN network in the Encoder projects the input to. The Encoder Stack Depth, or d_{depth}, is the number of Encoders we stack together to create the BERT model. The Dropout Rate, or r_{drop}, is the rate of inputs that are ignored in the dropout layers.

3
Methods

3.1 The Data Sets

In this work we utilize different data sets for different parts of the training, thus we wish to elucidate these and present a visual picture to help with understanding. The primary data set consists of proteins belonging to the beta-lactamase/transpeptidase-like superfamily of proteins. In Fig. 3.1, we have the set of all bacterial DNA in **a**, within this set we have our family of interest in **b**, within that we have a set of HMM labelled data in **c**, and finally we have our true labelled data in **d**. We note that the true labelled data may have some overlap with the HMM labelled data. If we wished to pre-train a general BERT for proteins, we would pre-train it on the outer group **a**. However, since we wish to operate exclusively on our specific family of possibly antibiotic DNA, we can gain some extra performance by restricting our set to only **b**. For the fine-tuning part, if we had enough true labelled data, we would make exclusive use of **d**. This is unfortunately not the case, thus we fine-tune on **c** and use **d** as validation data.

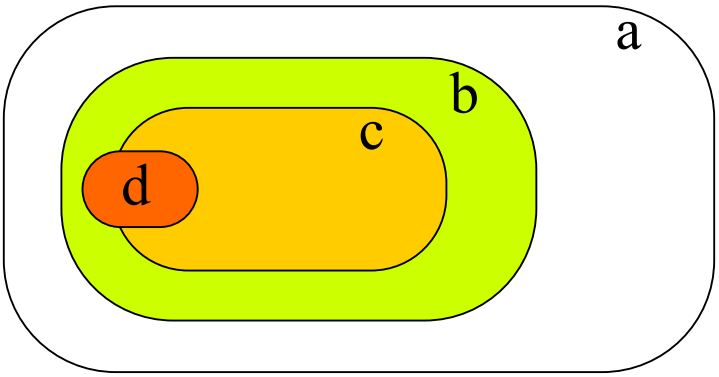

Figure 3.1: Visual representation of the various data sets used in this work. **a** represents all bacterial DNA, **b** presents our specific family of bacteria, **c** represents DNA within this family which has been labelled by using a HMM model, and **d** represents our set of true labelled DNA.

Another thing to note is that the reads in our data sets are generally not complete reads of the entire DNA sequence making up the protein, but snippets of them. Our reads tend to have a length of between 20-50 amino acids, whilst the complete sequences tend to have lengths around 300 amino acids long.

Later on we also wish to perform a cross-validation of the BERT model. We do this since it is possible that the HMM labelled data contains high correlations to the true labelled data since the HMM model is built upon using the true data. To determine if this is unfairly allowing the BERT model to somehow train on the validation data through the HMM data, we must somehow separate these dependencies and cross-validate the model. We do this by combining all training and validation data into one set, we then run this set through a clustering algorithm (UCLUST) with an identity threshold of 80% [12]. We then separate these clusters into six individual folds in a stratified manner, that is we try to keep the ratio between positive and negative and originally training and originally validation data consistent between the folds. A graphic representation of this can be seen in Fig. 3.2.

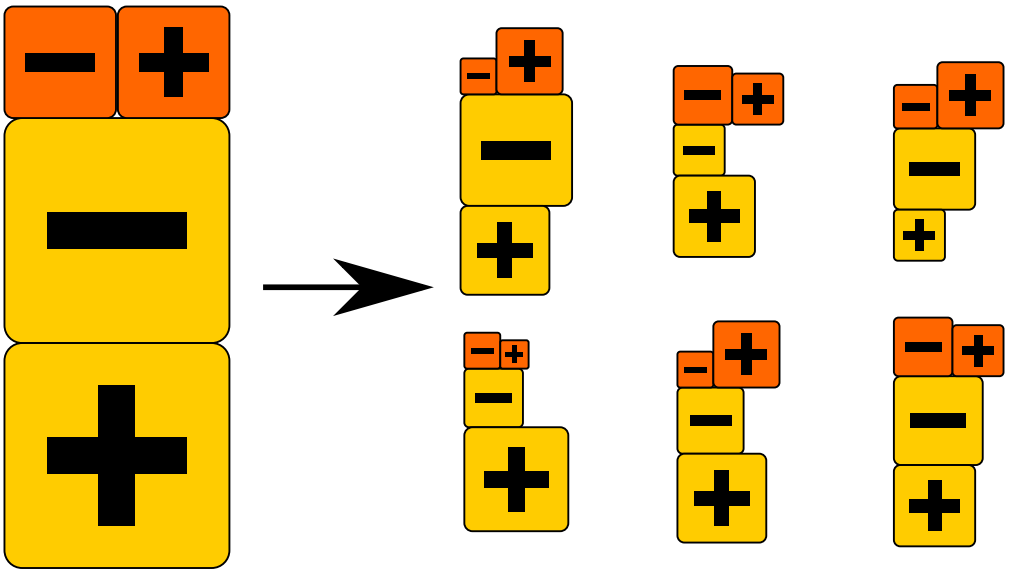

Figure 3.2: Visual representation grouping of data to perform the cross-validation. The dark orange blocks represent the original validation data, the yellow blocks represent the original training data. The pluses and minuses represent the sets of antibiotic resistant positive proteins and negative ones.

Finally we will also test the model on a set of metegenomic reads from waste water treatment plants located in Sweden, India, and Senegal, and a set of similar reads from the human gut.

3.2 The BERT Model

To construct the model, we stack a set of Encoders as described in the Theory section on top of one another. We then add a tokinization step at the input of the model. The tokenization is described by Fig. 3.3. We construct a vocabulary of accepted inputs of the model as the set of all amino acids and the special tokens [CLS], [PAD], and [MSK]. In order to transform these tags into vectors with which the model can operate, we construct a vector representation for each token. This vector representation is simple in the sense that we just assign each token a vector

in some space such that the vectors are evenly distributed. We also assign vectors to all possible positions in the input sequence. The final vector representation of a specific token at a specific position in the input is then achieved as the sum of the token and position embedding.

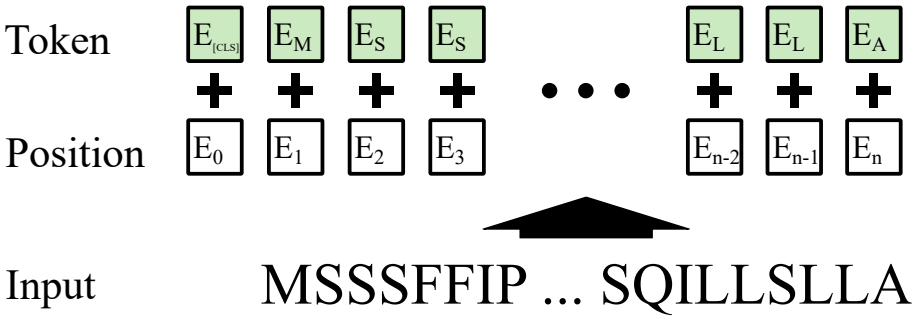

Figure 3.3: The embedding of sequences into vectors, the embedding takes both the token and the position into account.

Once the model has been constructed the next step is pre-training. The pre-training consists of taking unlabeled proteins, concealing or randomizing some of the tokens, and then asking the model to predict what these tokens should be. The advantage of doing this is that it allows the model to learn a general understanding of proteins, without requiring the use of labeled data. This is crucial as it will decrease the amount of labeled data we need in the fine tuning of which we have very little, only around 200 sequences. How this is done can be seen in Fig. 3.4.

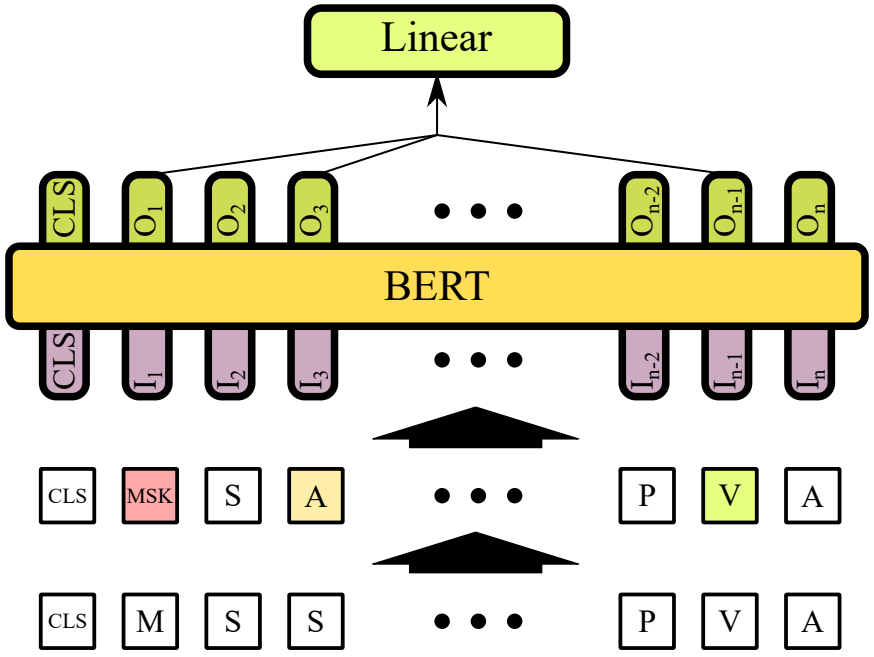

Figure 3.4: The pre-training step. Every sequence gets modified by masking or changing some of the input tokens, the model is then tasked to predict the original token.

For each sequence we select 15% of tokens to be modified. Of these 15%, 80% get

replaced with a [MSK] token, 10% gets replaced with a new random amino acid, and the remaining 10% are left unchanged. We feed these sequences then through the model and feed only the outputs corresponding to the modified tokens into a linear layer which classifies the output into one of the 22 amino acids. One important thing to note is that these modification are done every time a protein is presented to the model, thus we do not expect the model to suffer from overfitting from this step as the model nearly never sees the same input twice. We train the model until the loss enters a steady state. We also keep a set of validation data as a sanity check, but strictly speaking this should be unnecessary.

Table 3.1: The selected model parameters.

Model Feature	Value	Model Feature	Value
Hidden Dimension Size	512	FFN Internal Size	64
Maximum Sequence Length	64	Encoder Stack Depth	8
Number of Attention Heads	8	Dropout Rate	0.1
Self-Attention Internal Size	64		

Since the pre-training step takes a lot of time and computational power, we must perform the parameter selection at this step. We selected our initial parameters as roughly 80% the parameters from the original BERT model [11], these values can be seen in 3.1. Then we ran relatively short (on the order of 1/20th of the time the full pre-training would end up taking) pre-training steps whilst varying a single parameter. The result of these runs can be seen in Fig. 3.5. We wished to limit the total number of parameters as to not slow down the model to much, thus we ended up selecting the higher values for the hidden dimension size and the self-attention internal dimension size, but we kept the encoder stack depth at 8.

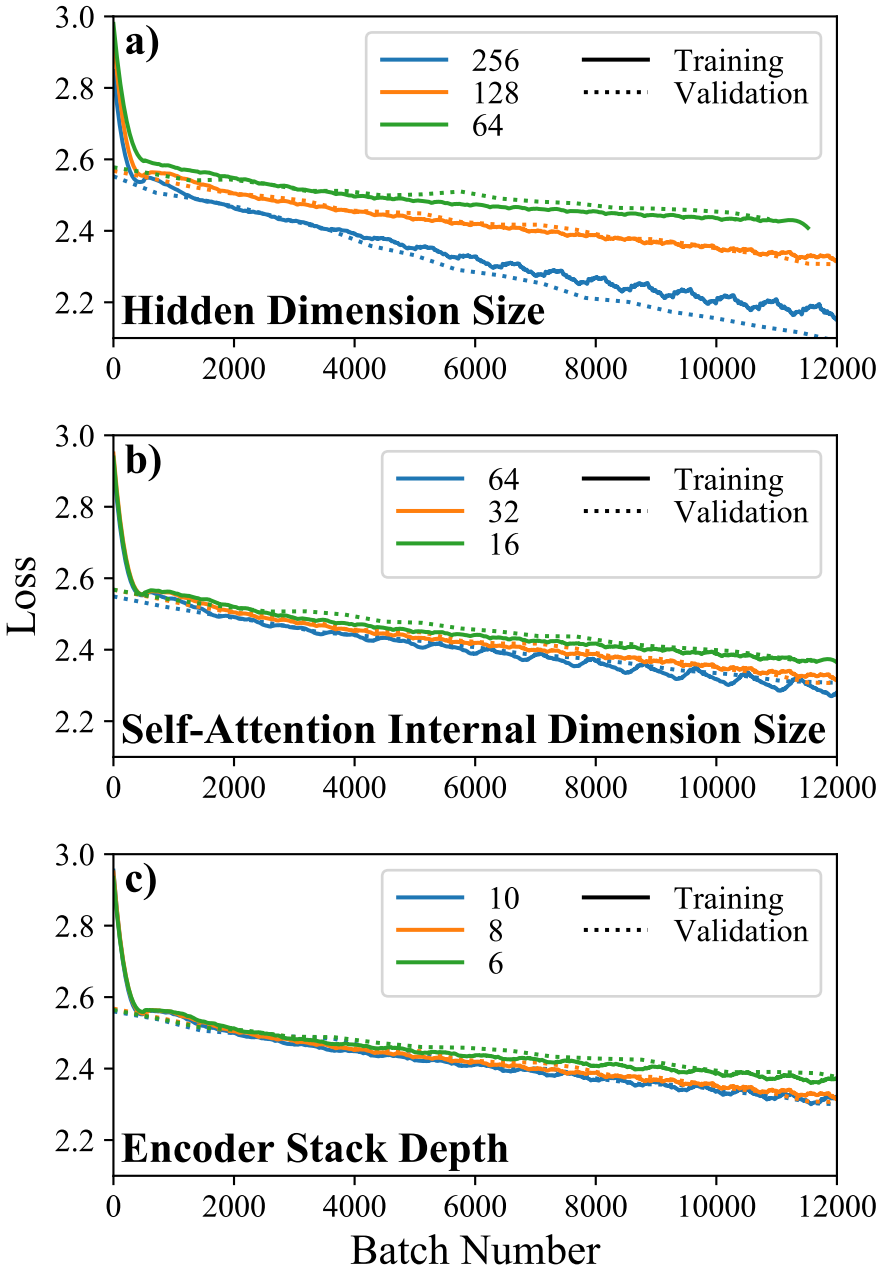

Figure 3.5: Pre-training runs performed whilst varying a single parameter, these runs ended up taking on the order of 1/20th the time it took for the final pre-training.

The full pre-training was run one a Nvidia GeForce RTX 2080 TI graphics card and took roughly 30h of computation time to complete. The evolution of the loss function can be seen in Fig. 3.6. We note that the sudden shift from periodic behaviour to the smother line at around batch 300000 is due to a change from fixed to random batch order. The corresponding accuracy of the pre-training ended up at roughly 43%, that is the model was capable of guessing correctly what amino acid had been either masked, changed, or left unchanged 43% of the time.

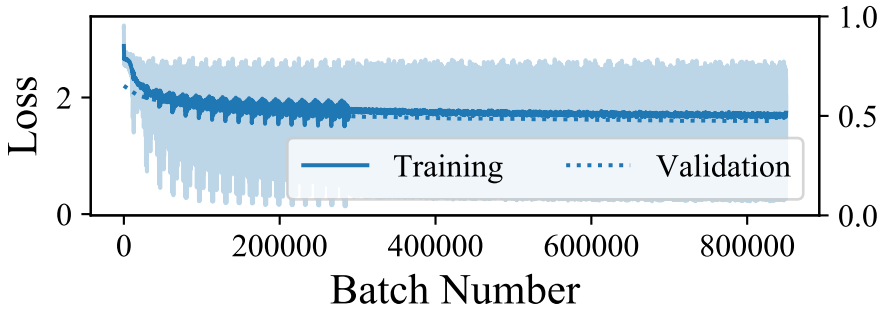

Figure 3.6: The loss function as it evolves over time during the pre-training, the shift from periodic to non periodic behavior around batch 300000 is due to a change from fixed to random batch order.

With the pre-training done we now turn to the fine-tuning. The fine-tuning is comparably an easier step now that all the heavy lifting has been performed by the pre-training. We simply feed an unmodified sequence into the model and pass the [CLS] token output into a linear model to classify it into either antibiotic resistant or not. We also introduce some noise into the training data by randomly taking as sub sequence of each sequence with a randomly length between 10 and 63. This does two things, it helps the model later on when we wish to predict on short sequences, and it also helps fight overfitting. A graphic of the fine-tuning can be seen in Fig. 3.7.

Since our version of BERT does not contain any Next Sentence Prediction or similar step in the pre-training as described in the original BERT paper [11], our [CLS] output is untrained, and has only access to the learned attention at each Encoder step. This entails that we cannot freeze the parameters of the BERT model during our fine-tuning as many schemes using a pre-trained BERT model usually does. Even now, with the pre-training that drastically lowers the amount of required labelled data, we still did not have enough data to fully train the BERT model. We circumnavigated this issue by using a HMM model to create a systematically labelled data set, and then we used our true labelled data set as our validation set.

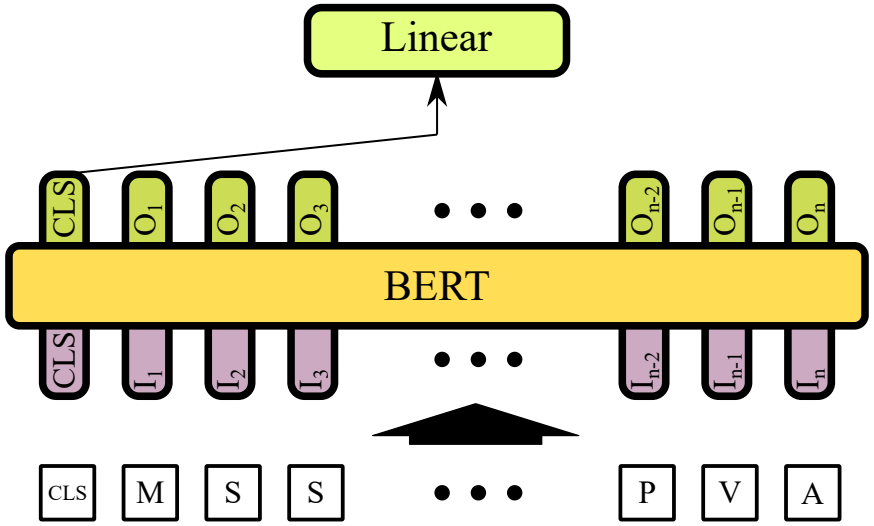

Figure 3.7: A graphic representation of the fine-tuning procedure.

Looking at Fig. 3.8, we see that the fine-tuning converges orders of magnitude faster as compared to the pre-training. Some back of the envelope calculations suggests that if we had not made use of the HMM labelled data, we would require about an order of magnitude more true labelled data than we have.

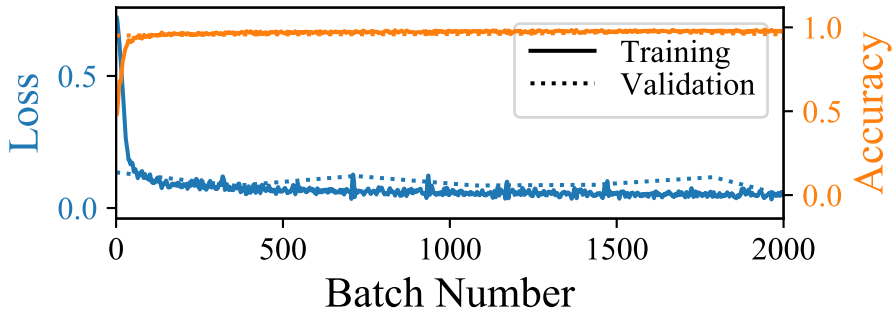

Figure 3.8: The loss and accuracy of the fine-tuning.

4

Results

4.1 Comparing with Previous Methods

To evaluate the performance of the BERT model, we compare its performance to a HMM model across various metrics. To start of we wish to note that HMM model only produces positive scores of how likely it believes that a protein is antibiotic resistant. The model also discards most proteins that fall beneath a specific score threshold, this is done to improve the time performance of the model. This all entails that proteins which we know to not exhibit antibiotic resistant may fail to show up in the output of the HMM model. To account for this we have elected to reintroduce all discarded proteins by assigning them a uniform random HMM score between 0 and the lower threshold score of the HMM model.

To begin with we study the receiver operator characteristic (ROC) curves as seen in Fig. 4.1 **a)**. This figure shows the ability of the two models to classify whether a protein is antibiotic resistant or not. For the case of a perfect classifier the ROC curve would connect the two ends of the dashed line forming an equilateral triangle, and a classifier that randomly guesses would correspond to a ROC curve that follows the dashed line. In the ROC curves shown we have neglected to differentiate the data with respect to the sequence length of the proteins, thus the curves are averaged across all sequence lengths (10-63). We see that the BERT model outperforms the HMM model. In the second figure Fig. 4.1 **b)**, we see the corresponding area under the ROC curves. In these graphs we have not averaged across the sequence length and thus can see its effect. A perfect classifier would have an ROC area of 1 whilst a random classifier would 0.5, since it would be the area beneath the dashed line in Fig. 4.1 **a)**. We see that for short sequence lengths the HMM model essentially guessing, whilst the BERT model performs admirably 0.9. We see that for longer sequence lengths both model perform similarly.

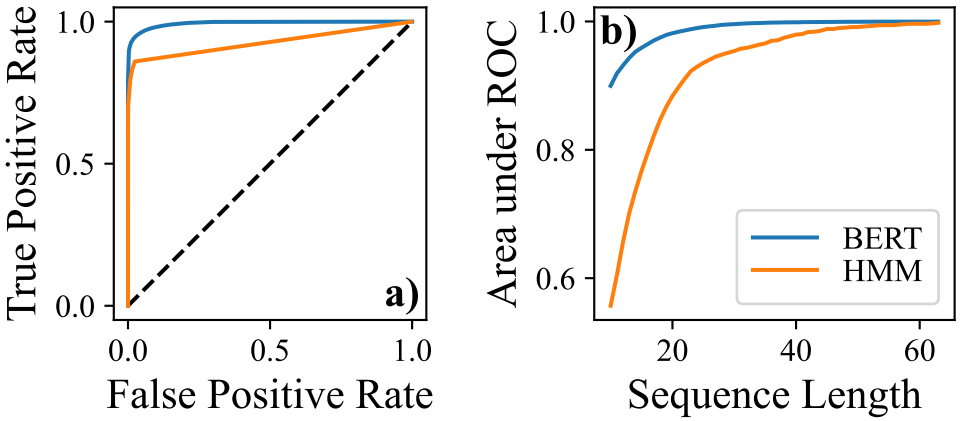

Figure 4.1: a) ROC curve comparing the predictive capabilities of the BERT and HMM models. The curve shown is averaged across all sequence lengths (10-63). **b)** The area under the ROC curve as a function of the protein sequence length.

Moving on we are also interested in more specific metrics to compare the models with, these can be seen in Fig. 4.2. These figures were produced by selecting a single cutoff threshold to be used for all sequence lengths. This was done since this how an HMM model is used normally, we do not vary the cutoff threshold with the sequence length of the proteins. This threshold was selected by taking the threshold which corresponds to the point on the ROC curve Fig. 4.2 **a)** closest to the upper left corner. The effects of our choices can be seen rather starkly in the figure as the apparent performance of the HMM model tappers off after a sequence length of 45. It is not the case that the HMM model performs worse on longer sequence lengths, we have all ready seen this to be the case in the previous figure Fig. 4.1 **b)**. Instead what this shows us that the the HMM model does separate the two distributions (positive and negative proteins regarding antibiotic resistance), thus the models capability is very dependent on the choice of cutoff threshold and suffers because of this since we select the best cutoff threshold averaged across all sequence lengths. On the other hand, we do not see this with the BERT model, implying it does separate the two classes more reasonably.

Taking a look at the actual curves in the figures Fig. 4.2, we again see that the HMM model is essentially guessing when it comes to proteins with a short sequence length. In the lower figure we see that for short sequences, the HMM model almost always guesses that the protein is negative. This is the case likely due to the fact, as discussed in more detail in the beginning of this chapter, that we reintroduced sequences which the HMM model did not produce a score for with a low HMM score. Since the HMM model struggles with short sequences, mos of these get reintroduced as negative proteins. We see that the BERT model struggles slightly with both the false positive rate and the false negative rate for short sequences, of which the false positive rate is of utmost importance. Still it performs much better than the HMM model which for all intents and purposes does not function for short sequence lengths.

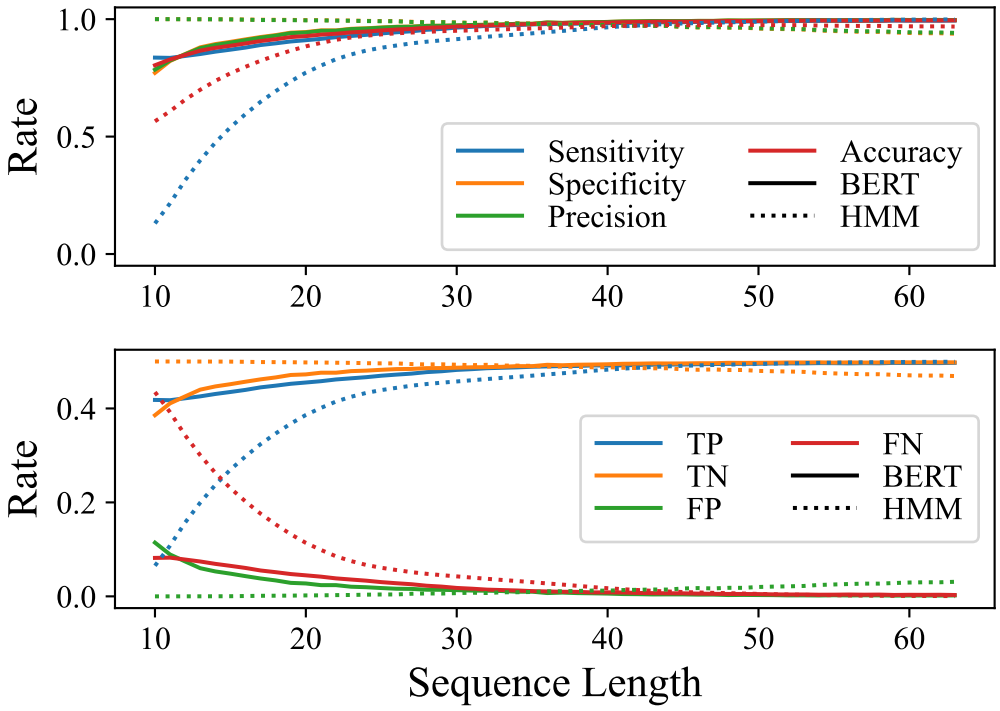

Figure 4.2: A variety of metrics comparing the BERT and HMM models for different sequence lengths. The optimal cutoff threshold used was obtained by selecting the threshold corresponding to the point of the ROC curve (Fig. 4.1) closest to the upper left corner.

We can further study how well the two methods perform in separating the two distributions of positive and negative proteins by plotting the distributions against one another in a 2D distribution plot. These plots can be seen in figure Fig. 4.3. In the figure we have separated the proteins again by sequence length, where **a)** shows the short proteins, **b)** shows the medium length ones, **c)** shows the long proteins, and finally **d)** shows all proteins. We again see that for the short proteins, the BERT model separates the two distributions well whereas the HMM model struggles, but for longer sequences this difference diminishes. We can now also see evidence of the earlier suggested instability in selecting the cutoff threshold for the HMM model. In each figure, it is clear that a small shift of the horizontal dashed line would cause a significant shift in the performance of the HMM model as the negative distribution consists of a sharp peak along the HMM axis.

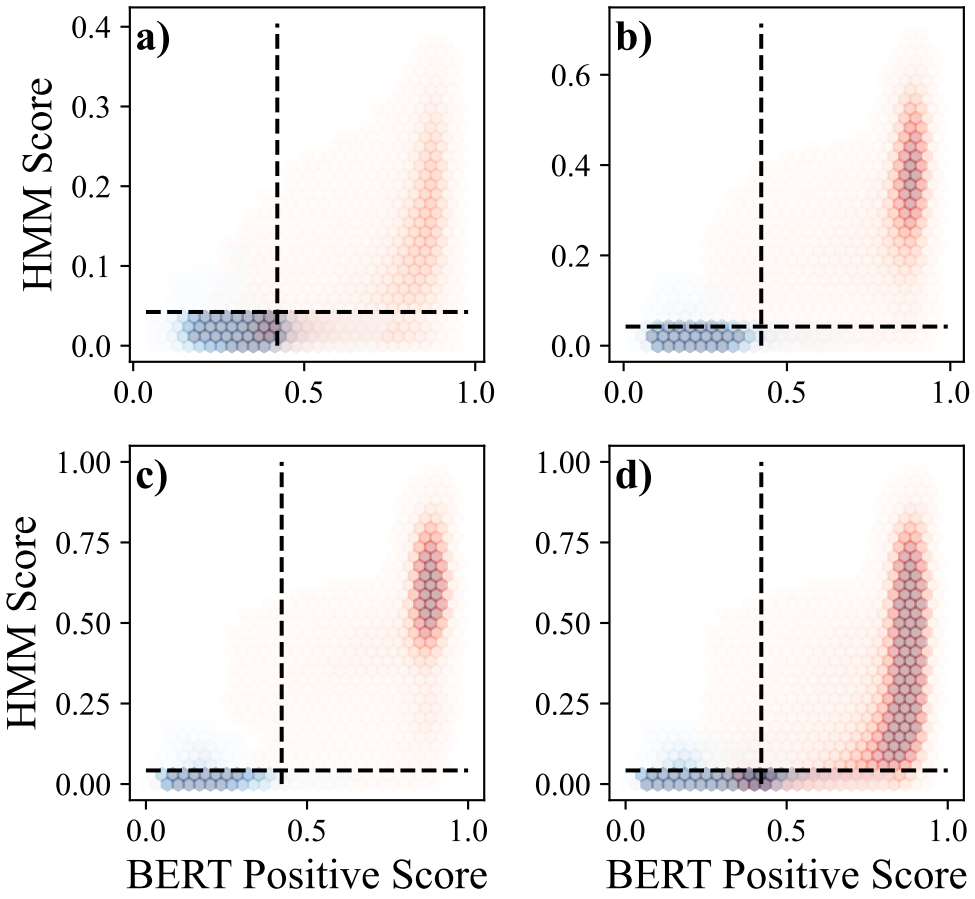

Figure 4.3: The figures show the distribution of proteins in the 2D space of BERT and HMM scores. The dashed lines correspond to the cutoff thresholds of the two models, and the blue and red distributions correspond to the negative and positive proteins respectively. All figures are shown in the same linear scale. **a)** Shows sequence lengths between 10 and 17, **b)** shows between 18 and 35, **c)** shows between 36 and 53, and **d)** shows for all sequence lengths.

To get a clearer view of what happens where the two distributions overlap we also study the corresponding 1D distributions along each axis, these can be seen in Fig. 4.4. We see that in row **a)**, which corresponds to the short proteins, the HMM model fails to separate a large share of the positive proteins from the negative ones, the same might also be said for the BERT model, but its share of overlap is much smaller. In row **b)** we see that the HMM model still struggles to separate the two groups whilst the BERT model does so almost perfectly. Row **c)** shows again that for the long sequences there is little difference between the two models.

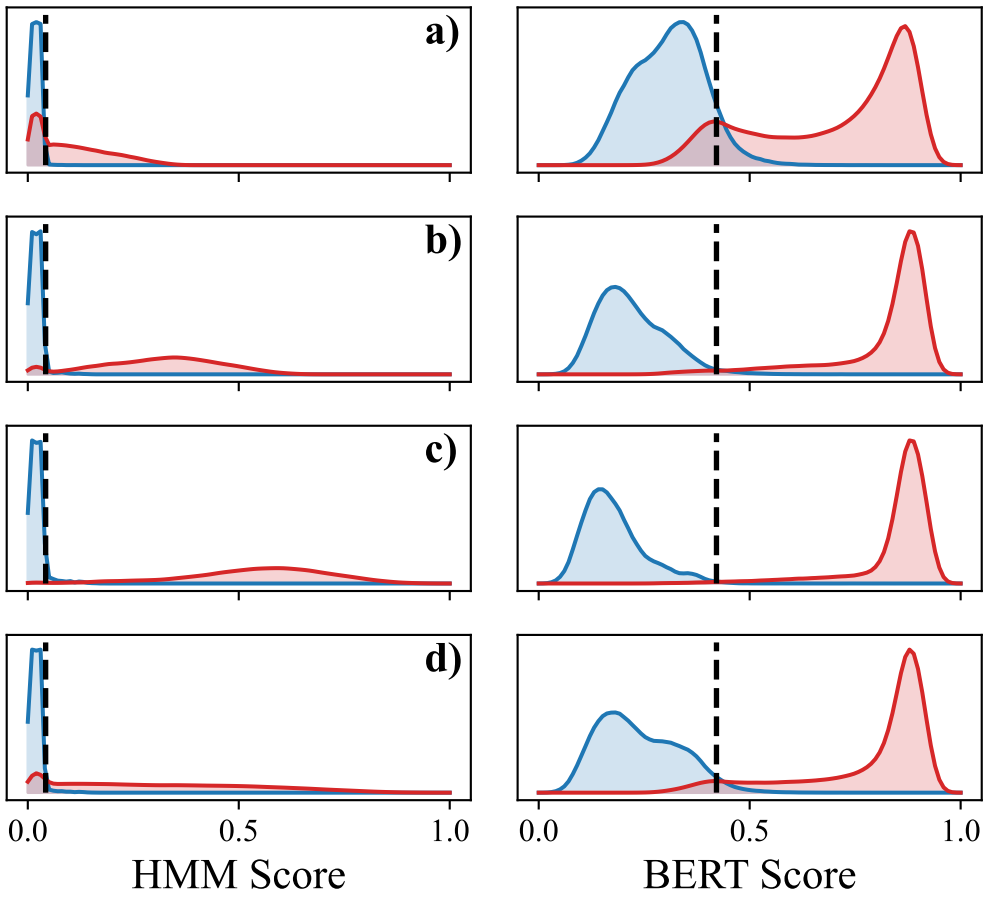

Figure 4.4: The corresponding 1D distributions to those shown in Fig. 4.3. The left column shows the distributions along the HMM axis and the right shows them along the BERT axis. As in the previous figure, **a)** shows the short sequences, **b)** the medium ones, **c)** the long ones, and **d)** shows all lengths. Again, the red distribution shows the positive proteins whilst the blue shows the negative proteins.

To get an understanding of how well the two models separate the distributions we can measure the overlapping area between them and compare between models, this can be seen in Fig. 4.5. In this figure, a separation of 1 means that the two distributions have no overlapping area, whilst 0 means that they overlap completely. We see that for very short sequence lengths (<15), the HMM model distributions have half or more of their area overlapping. Further we see that the BERT model performs with sequence lengths of 10 as well as the HMM model does with sequence lengths of 18. The two models equalize at roughly sequence lengths of 25, after which the difference is to small to confidently make any judgment.

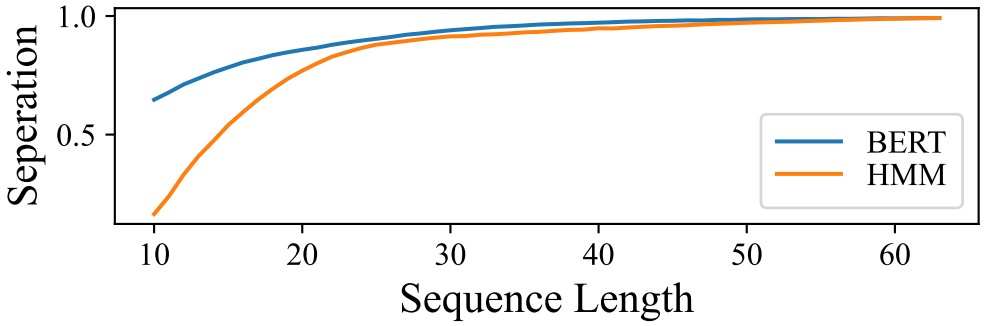

Figure 4.5: The degree of separation between the two distributions in the sense of measuring the area of overlap between the two distributions. For the case of a perfect classifier the separation would be 1, meaning no overlap, whilst a separation of 0 means that the two distributions overlap completely.

Since the HMM model is based upon, amongst other things, sequence alignment, it was expected to be highly resistant towards reading errors. That is, random errors in which a set of amino acids are read incorrectly and reported as another amino acid. These errors are expected and occur with varying error rate depending on the method, thus our models must be resistant towards such errors. To determine if such was the case we elected to test both models upon our validation data, but where we randomly introduced reading errors into said data at varying rates. The scheme used to do this similar to the pre-training step, with the only difference being that we only made use of randomly changing amino acids. In order to only see the effect of the read errors, we chose to only work with full length sequences, of length 63, since we all ready the effect of sequence length on both models.

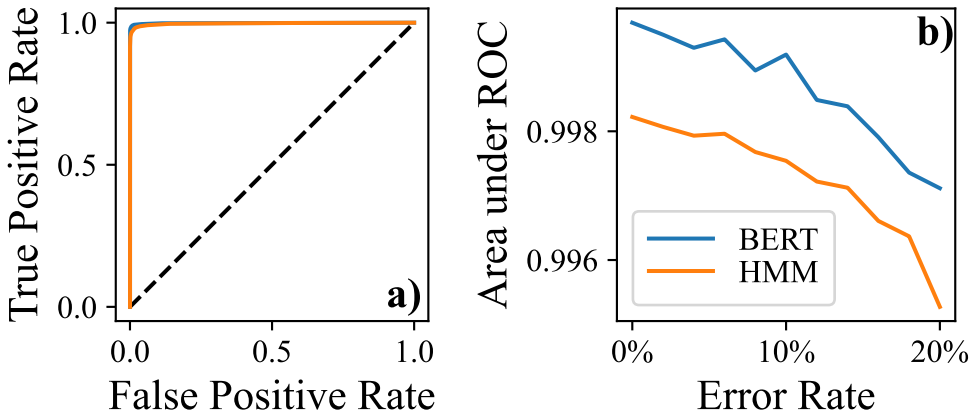

Figure 4.6: **a)** ROC showing the performance of the BERT and HMM models when the proteins have had random errors introduced. The ROC shown has been averaged across all chosen error rates. **b)** The area under the ROC for different error rates.

We can see in Fig. 4.6 that both models perform well for all error rates. The

difference we see between the two models is to small to make any conclusion. The reach the same verdict when studying the metrics in Fig. 4.7.

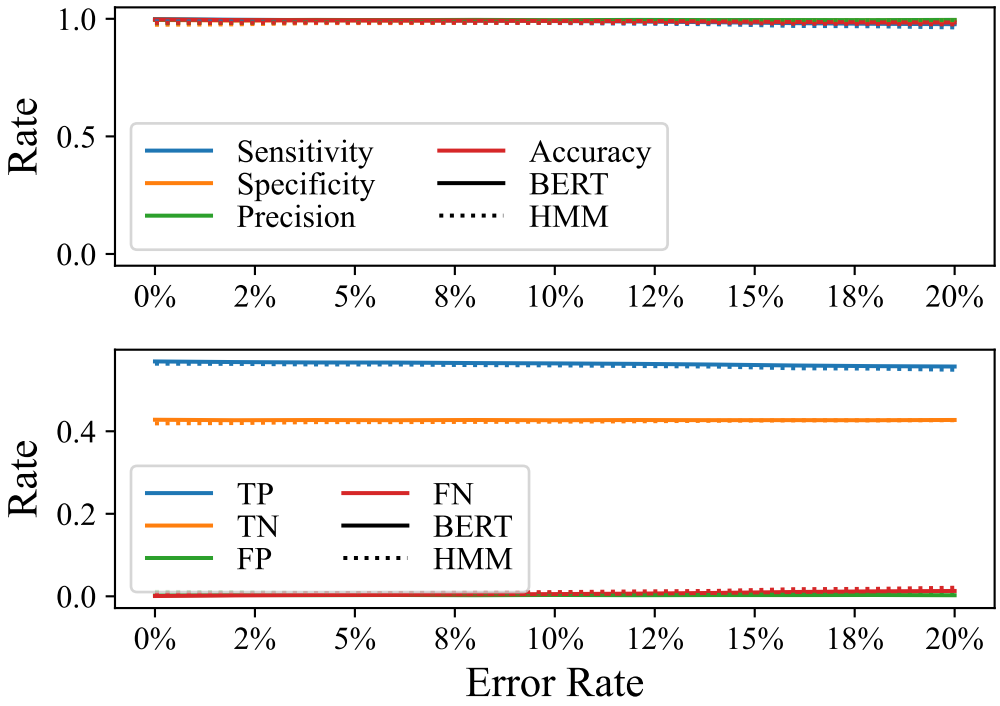

Figure 4.7: Various metrics to compare the BERT and HMM model when operating on proteins with introduced read errors.

4.2 Testing Training Transferability

As stated earlier, we chose to restrict our BERT to training upon an single super family of proteins in order to maximize performance. However, we believe that most of the learned information from the pre-training step is general to most proteins and thus transferable. To test this we selected another super family, known as methyltransferase, and re did the pre-training step upon this super family. However, instead of starting out from with an uninitialised BERT model, we made use of the one already trained on the betalactase super family. Doing this we found that the pre-training completed within less then a single epoch, and time wise the training took about a minute compared to the original 30 hours. This tells us that the pre-training was indeed quite general.

In conjugation with the above we also wished to test the BERT models performance on a data set containing sequences from both super families. The reason behind this is that a general BERT able to classify all protein families related to antibiotic resistance would be a much preferable model to have. To do this we created a balanced data set between the two families and performed the pre-training and fine-tuning on said set. As before the pre-training took minutes, and the same was

true of the fine-tuning. We found the final performance of the BERT model on this mixed data set to be slightly worse than when the model was trained on the two sets separately. When the model classified the two sets separately the average accuracy of the validation data was around 99%, when combining the sets the accuracy was around 98%. Thus we pay a small tax in accuracy to gain generality.

4.3 Cross-Validating the Model

Since we elected to fine-tune the model upon HMM labelled data, we had concerns that it was possible that the model was unfairly gaining insight into the validation data through the training data. What we mean by this is that the HMM model has been in essence 'trained' on our validation data. The HMM model works essentially by comparing new proteins with those in its bank. This means that it is possible that the HMM labelled data contains proteins that are identical or at least highly correlated to the validation data. In turn this could imply that the BERT model is not learning anything general but simply learning the entire validation set.

As stated in the Method segment, we avoid the above issues by performing a cross-validation of the fine tuning. How we separated the data into folds can be seen in the Method segment. The cross-validation helps ensure that the correlation between the training and validation data does not over inflate the performance of the BERT model. The comparison between the standard model and the cross-validated model can be seen in the figure Fig. 4.8. Here we see that the BERT model has truly learnt something general about the data and not simply managed to recreate the validation data. The various metrics in Fig. 4.9 also lend credence to this conclusion.

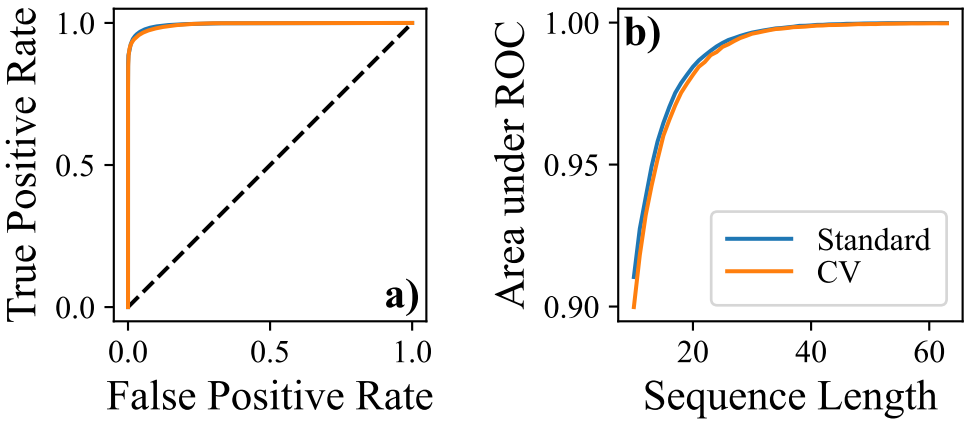

Figure 4.8: ROC showing the difference between the standard model and the cross-validated one. The left figure shows the ROC curves for sequences of various amino acid lengths, and the right figure shows the area beneath the curves to show how the models perform for various sequence lengths.

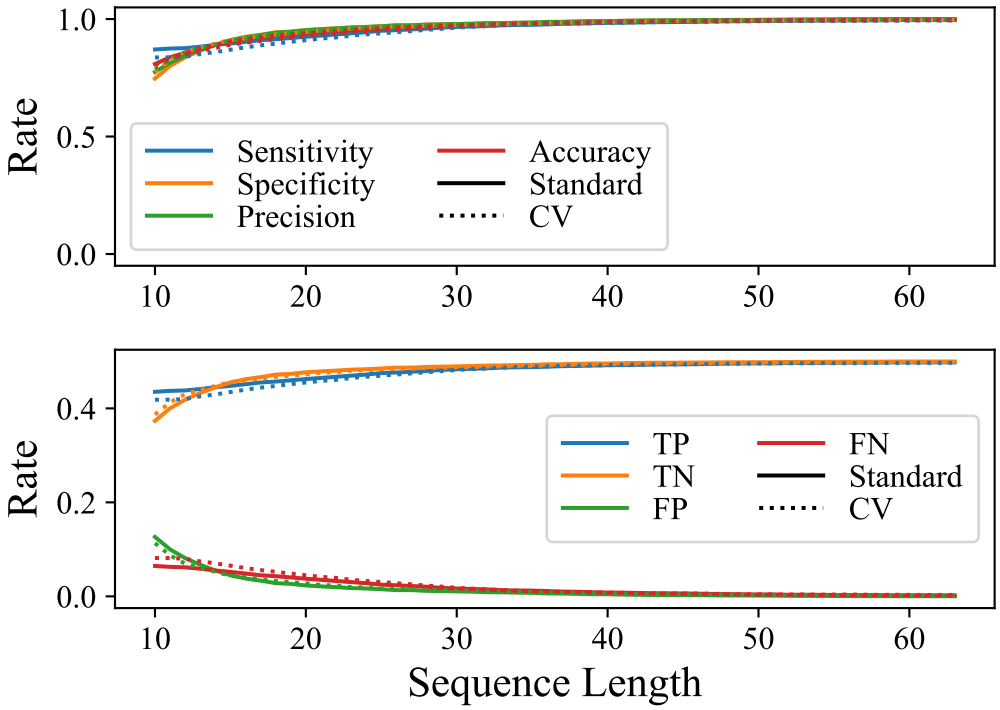

Figure 4.9: Various metrics comparing the standard and cross-validated versions of the BERT model.

4.4 Timing the Model

It is quite difficult to compare the two models in terms of computation times. The reason for this is that the models run on different architectures. The HMM model is run on the CPU whilst the BERT model is run on a GPU. However, for the use case intended for the BERT model, we can still make a qualitative comparison. Running on a GeForce RTX 2080 Ti, the BERT model can process 10 million sequences in roughly 30 minutes. The same sequences take roughly 2 minutes for the HMM model running on 40 CPUs. This tells us that for our use case, the HMM model is an order of magnitude faster then the BERT model. This difference can be made even bigger if we slacken the HMM somewhat which entails a less detailed result, but this increases the difference to two orders of magnitude.

4.5 Glimpsing into the Model

In order to understand what the model has learned, let us take a look at the which parts of a protein the model considers important when classifying a positive protein as such. We can do this by taking many snippets of a protein and running each snippet through our models, then we combine the scores for each snippet and thus we can get a score for each amino acid in the protein. We get the snippets by traversing the protein with varying window sizes. The result of such runs can be

seen in Fig. 4.10. Here we have chosen three positive proteins to take a look at. What we can note from these graphs is that the first peaks in them, tend to fall between amino acid 60 and 100, which is where we expect the active site to exists in our family of proteins [13]. Thus we can note that one possible thing the model has learned is in how to identify such sites, but we will not delve deeper into the analysis of this.

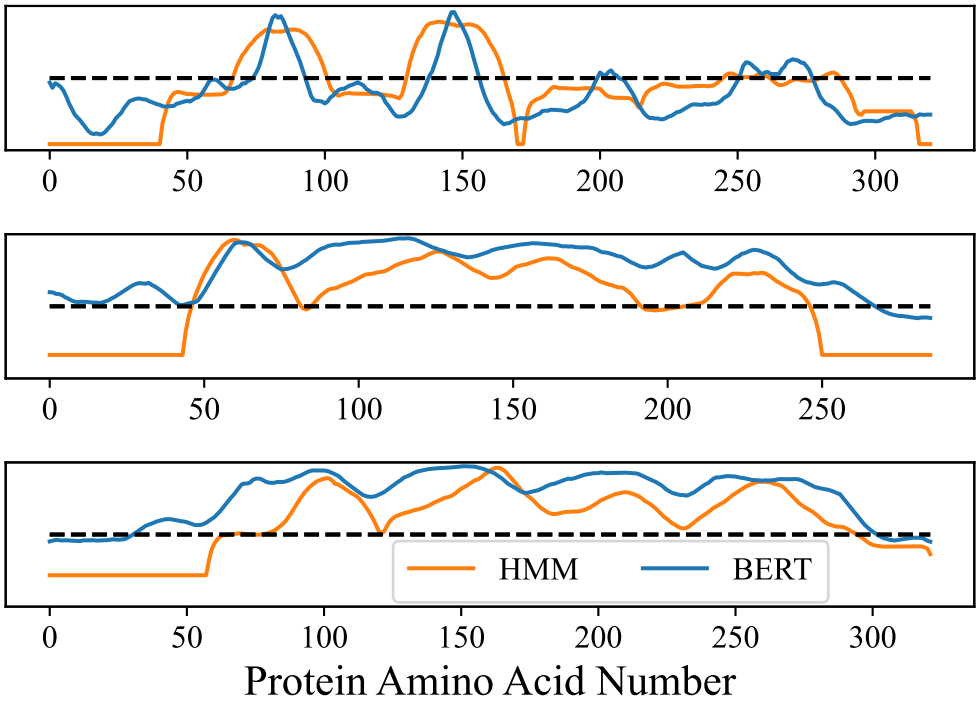

Figure 4.10: The graphs show how important each part of the protein is in order to classify it as antibiotic resistant. Each protein shown is a known antibiotic resistant protein. The dashed line shows the optimal threshold, and the graphs for each model are aligned such that their thresholds overlap.

Something else which we might take a look at is a Principle Component Analysis (PCA) of the [CLS] token output of the BERT model. In Fig. 4.11 we see the distributions of positive and negative proteins along the two most significant principle components. The left column shows the cases where the model was correct whilst the left shows when they miss classified. The top row shows the classification as done by the BERT model and the bottom row shows the HMM model. For the BERT model, we see what we expect. It has managed to separate the two distributions quite well and inserted a decision boundary between them.

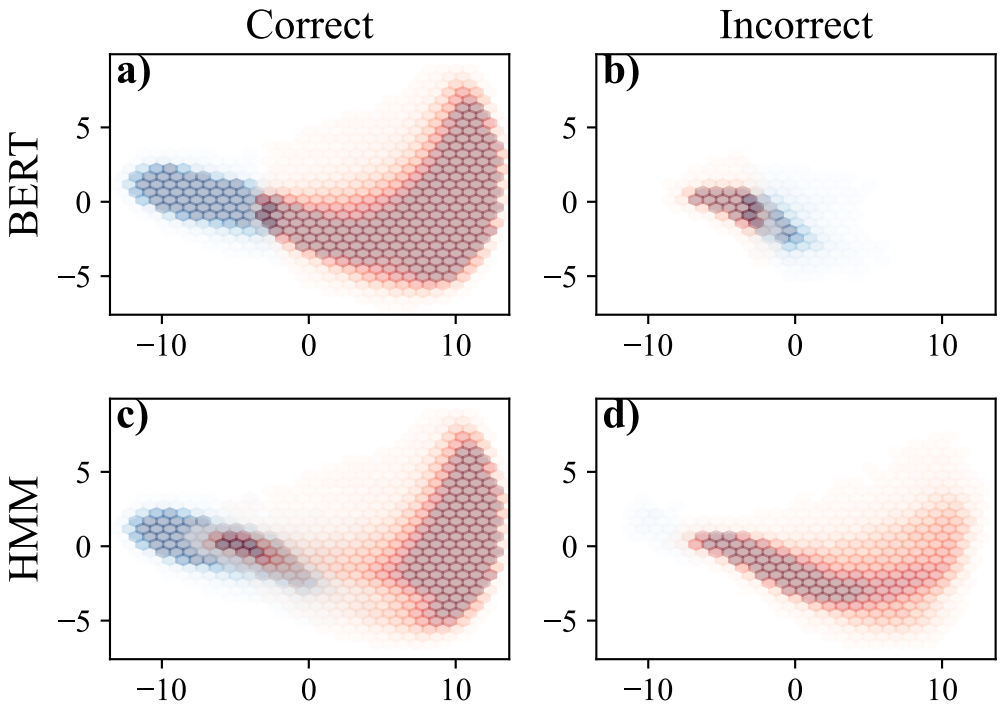

Figure 4.11: PCA plots of the [CLS] token output of the BERT model. The left column shows the two distributions (red for positive and blue for negative proteins) when the model is correct, whilst the right column shows when the model is incorrect. The top row shows the BERT models classification and the bottom row shows the same for the HMM model. All plots are made in the same linear scale.

We can see the same results but projected onto the axis in Fig. 4.12 for a cleared picture where the distributions overlap. The main purpose of studying these figure is to ensure that the AI model has learned something different than what the HMM model knows. We see this to be the case as the two models clearly differentiate between the two classes differently. This allows us to rest easy knowing that our choice of training the BERT model upon HMM labeled data has not merely created an AI model that models the HMM process, but rather a model that has learned something general about the data which is not captured by the HMM model.

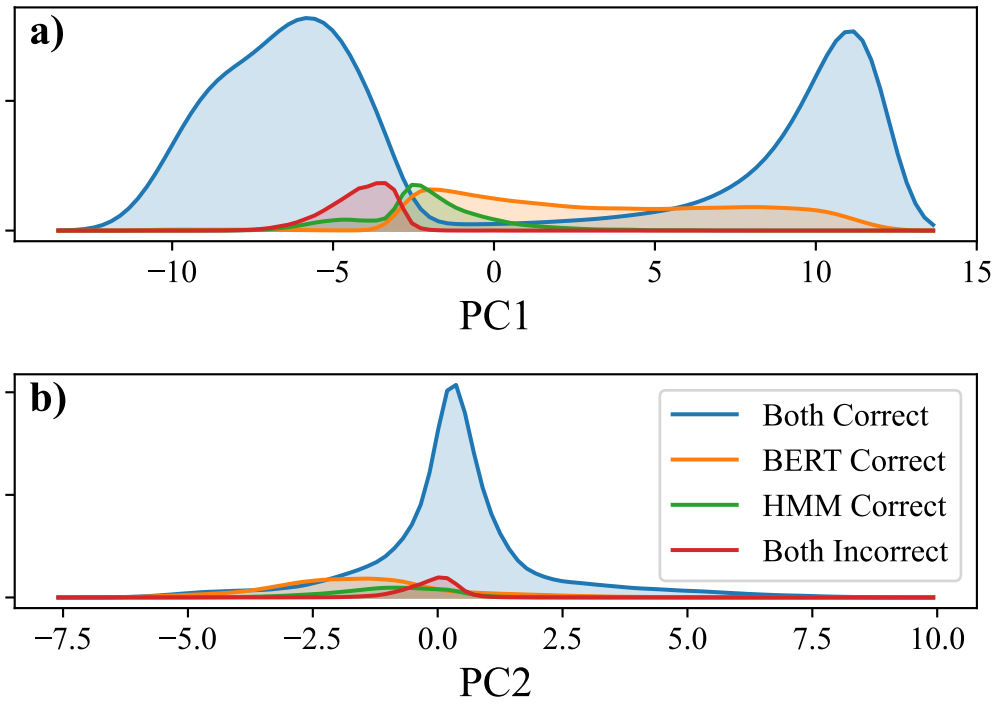

Figure 4.12: The corresponding distributions to the 2D distributions in Fig. 4.11.

4.6 Testing on Metegenomic Reads

Finally we can also test our model in its intended use case, searching through metagenomic reads. For this reason we have complied a set of samples, one set from the human gut and one from various waste water treatment plants from various countries. The results of running the samples through our models can be seen in Fig. 4.13 and Fig. 4.14. Since these reads are unlabeled, we can say little regarding the performance of each model. In these figures the vast majority of reads fall below the HMM cutoff threshold, so in order to make the figures readable we elected to not show these points in the figures. The breakdown of where the reads fall categorically is quite consistent between the samples and is as such: roughly 93.5% of all reads fall beneath both models cutoff threshold (lower left quadrant), roughly 6.5% fall in the lower right quadrant, and the remaining few reads are distributed roughly equally in the two quadrants shown in the figures. Each figure shown represents a total sample size of 10 million reads, but only a minuscule fraction of these fall within the quadrant of interest (the upper right one). Within each figure there exists on the order of 100 points or fewer in each upper right quadrant, and these reads are the ones most likely to be antibiotic resistant.

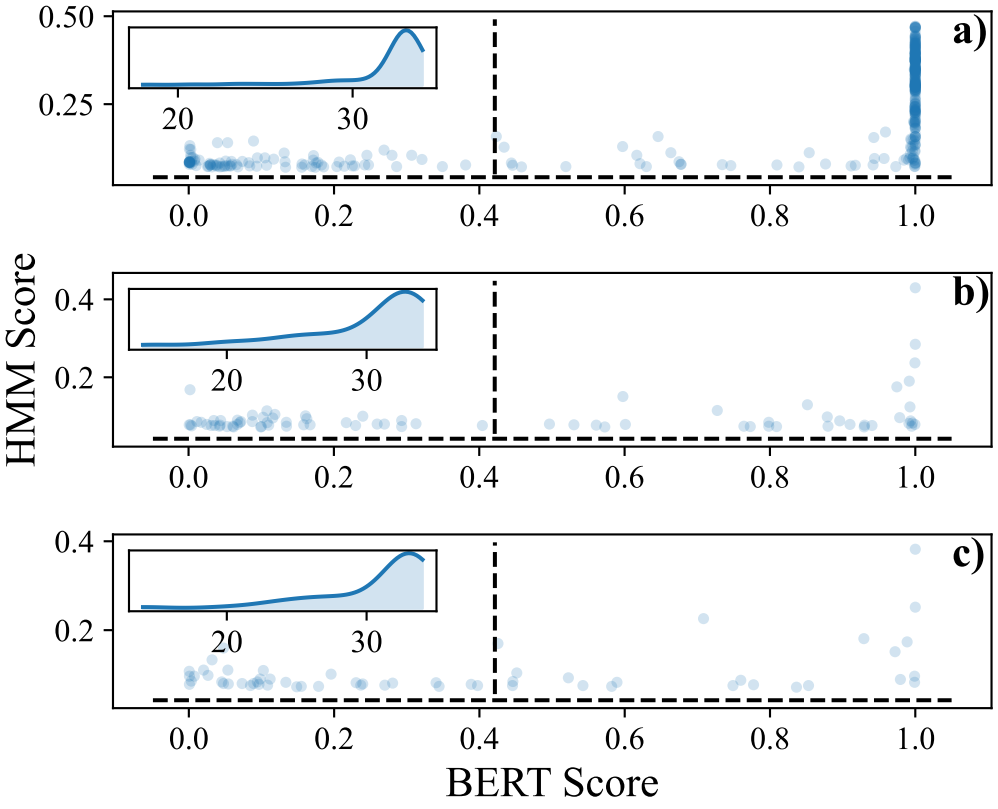

Figure 4.13: Graph showing the model output on unlabeled metagenomic reads. The insert shows the sequence length distribution. The dashed lines shows the cutoff thresholds obtained previously. The samples shown are three different metegenomic samples from the human gut. The vast majority of reads scored below the HMM cutoff, these are not included in the figure

In order to fully determine to performance of our model we would need to construct these proteins and test them in a wet lab, but this is to time consuming and would not fit within the time frame of this thesis work. One thing to note is that although the vast majority of points can be cast aside using only the BERT model, one is still left with roughly 6.5% of the points if we use the optimal cutoff threshold from the validation data. This number is to vast to be useful, thus we arrive at two possible ways to apply the model. The first is to use the HMM model as a pre-screening step. Since the HMM model is a couple orders of magnitude faster than the BERT model, this is not to cumbersome. The only negative to this strategy lies in that it would only allow us to further narrow down the output of the HMM model, and we know from the validation data that there exists proteins that are positive which the HMM model fails to capture but the BERT model can detect. The other option is to simply make use of a much higher cutoff threshold. There seems to exists evidence for this method to work well in the figures as one may note that each figure contains a line of points with nearly perfect BERT scores.

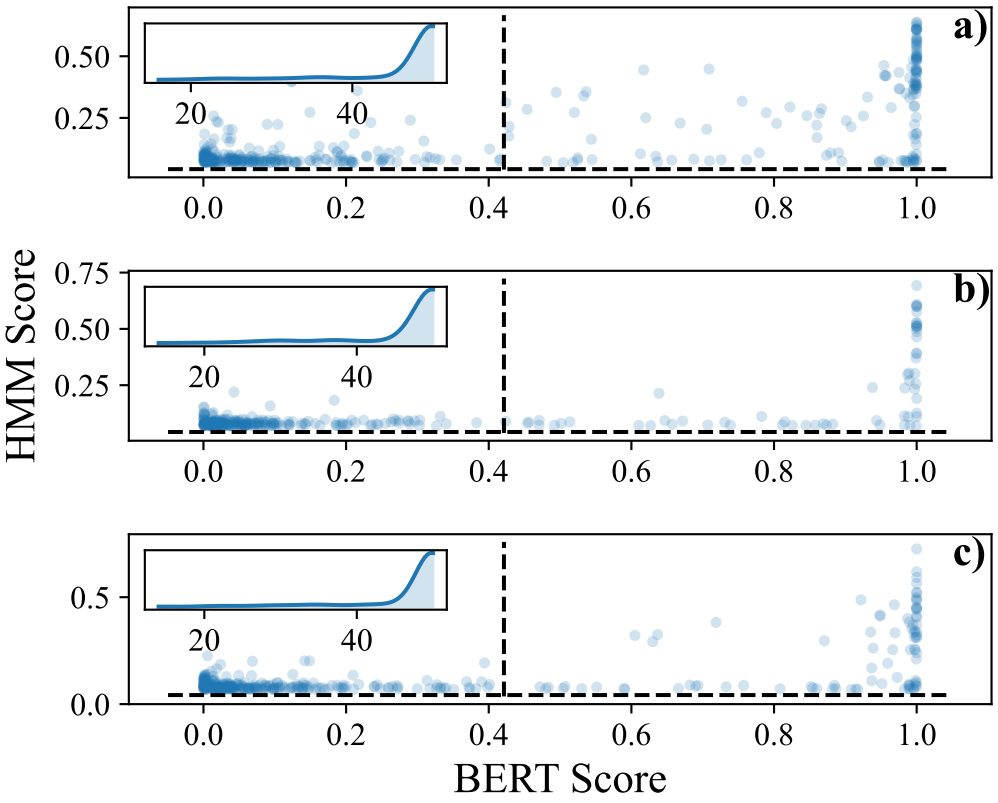

Figure 4.14: More metegenomic reads, this time from waste water treatment plants from various countries, **a)** is from India, **b)** Senegal, and **c)** Sweden.

4.7 Discussion and Conclusion

From the various results which we have presented, we can make a couple of statements. The most glaring aspect in which the BERT model outperformed the HMM model is the matter of sequence length. Depending on which of the various metrics shown in the results one selects to go by, the exact cutoff when the two models equalize in performance varies. The highest such value was found when comparing the area under the ROC in Fig. 4.1. Here the BERT model outshone the HMM model for sequences lengths of 50 and shorter, and the most striking difference was found at a sequence length of 10 where the BERT model was as good as the HMM model was at sequence lengths of 25. The smallest difference between the BERT and HMM model was when comparing the area overlap between the positive and negative distributions of sequences in each models predictive space as seen in Fig. 4.5. Here the BERT model outperformed the HMM model for sequence lengths of 25 and shorter, but it also had a small most likely insignificant lead up until sequence lengths of 50. From these and the other comparisons we find that the BERT model is capable of correctly classifying most short sequences whilst the HMM model gives random predictions for these sequences.

Another important aspect lies in the stability of each method. What we mean with this is the sensitivity of the model upon the selection of their cutoff threshold. We saw the effect of this in the ROC and metrics plots of Fig. 4.1 and Fig. 4.2. In these figures we saw that the HMM method seemingly performed worse with sequence lengths above the length of 45, and this is due to the optimal cutoff threshold being quite unstable and variable with changing sequence lengths. Thus, once a optimal threshold is selected by including all sequence lengths in the computation, this causes the model to miss classify the longer sequences which should be easier to classify. We do not see this behaviour with the BERT model, meaning it has a more stable optimal cutoff that does not vary to much with sequence length.

Unfortunately we also have to pay something for these increased performance figures, and in this case that payment lies in computation time. Of course, we are comparing apples to oranges, but for our supervisors use case, the HMM model has a computation time that is one to two orders of magnitude faster than the BERT model. However, the computation time of the BERT model was determined to be acceptable, capable of processing 10 million sequences in 30 minutes.

Looking into the future of this project, we found that our model was capable of becoming more general by training it on multiple protein super families. This could be the next step in order to create a general antibiotic resistance BERT model. Weather or not this is done, the model should be put to the final test by synthesizing the detected proteins from the metagenomic reads in a wet lab.

Going back to our stated goals, we have shown that the BERT model is indeed capable of identifying antibacterial microbial DNA. It also outperforms the HMM model, and especially so on the short reads which we tend to get with metagenomic reads. Further, we have also seen that the BERT model has learned something different than the HMM model, what this tells us is that we can gain even better results by utilizing both models at the same time and combining their results. This is not an issue computation time wise as the BERT model is already the bottle neck in that regard. Thus we recommend that the trained BERT model be used in combination with the HMM model for future use.

Bibliography

[1] Debstar, "Spombe Pop2p protein structure rainbow," 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Spombe_Pop2p_protein_structure_rainbow.png (accessed on 2022-05-12)

[2] World Health Organization, "Antibiotic resistance," 2022, [Online], Available: https://www.who.int/news-room/fact-sheets/detail/antibiotic-resistance (accessed on 2022-05-12)

[3] "ANTIBIOTIC RESISTANCE THREATS IN THE UNITED STATES," Centers for Disease Control and Prevention, USA, 2019. [Online]. Availale: https://www.cdc.gov/drugresistance/biggest-threats.html

[4] Antimicrobial Resistance Collaborators, "Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis," The Lancet, vol. 399, issue 10325, pp. 629-655, FEB. 2022, doi:10.1016/S0140-6736(21)02724-0

[5] World Health Organization, "The top 10 causes of death," 2020, [Online], Available: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death#:~:text=The%20top%20global%20causes%20of,birth%20asphyxia%20and%20birth%20trauma%2C (accessed on 2022-05-12)

[6] J. Jumper, R. Evans, A. Pritzel, et al., "Highly accurate protein structure prediction with AlphaFold," *Nature* 596, 583–589, 2021, doi:10.1038/s41586-021-03819-2

[7] "How Bacteria and Fungi Fight Back Against Antibiotics," Centers for Disease Control and Prevention, USA, 2021. [Online]. Availale: https://www.cdc.gov/drugresistance/about/how-resistance-happens.html#:~:text=Antibiotic%20resistance%20is%20accelerated%20when,resistant%20germs%20survive%20and%20multiply.

[8] T. Shafee, "Enzyme Structure," 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Enzyme_structure.svg (accessed on 2022-05-12)

[9] N. Stoler, A. Nekrutenko, "Sequencing error profiles of Illumina sequencing instruments," *NAR Genomics and Bioinformatics*, Volume 3, Issue 1, March 2021, doi:10.1093/nargab/lqab019

[10] A. Vaswani, et al., "Attention Is All You Need," 2017, doi:10.48550/arxiv.1706.03762

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018, doi:10.48550/ARXIV.1810.04805

[12] R. Edgar, "UCLUST algorithm," *usearch v11*, [Online]. Available: https://drive5.com/usearch/manual/uclust_algo.html (accessed on 2022-05-12)

[13] InterPro, "Beta-lactamase, class-A active site", [Online]. Available: https://www.ebi.ac.uk/interpro/entry/InterPro/IPR023650/structure/PDB/#table (accessed on 2022-05-12)

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY