Understanding the Human Genome Language with Natural Language Models

Piyush Mishra

Understanding the Human Genome Language with Natural Language Models

Turing Centre for Living Systems,
Aix-Marseille University
In collaboration with: Department of Biomedical Informatics,
School of Medicine and College of Engineering and Applied Sciences,
Stony Brook University

Piyush Mishra

Marseille, France
June 05, 2022

Submitted in partial fulfillment of the requirements for the Master degree in
Computational and Mathematical Biology (Bioinformatics)
Supervised by Dr. Ramana Davuluri

Declaration of Originality

I, Piyush Mishra, declare that the work presented in this text entitled "Understanding the Human Genome Language with Natural Language Models", which has been carried out as a collaboration with University of Aix-Marseille, France, and Stony Brook University, New York, USA, is entirely my own, to the best of my knowledge. All information derived from other sources has been duly acknowledged in the references, and due credit is given to the parties involved. No part of this work was previously presented for any other diploma at any other institution.

Piyush Mishra,
Turing Centre for Living Systems,
University of Aix-Marseille

Dr. Ramana Davuluri,
Dept. of Biomedical Informatics,
School of Medicine and College of Engineering and Applied Sciences,
Stony Brook University

Abstract

All genomes are made up of the letters `A, G, C`, and `T` signifying the different nucleotides in the DNA. About 500,000 human genomes have been sequenced, thereby creating a vast corpus consisting of the language of the genome, studying which will help uncover the different encoded messages, i.e., functions of the genome. We seek to make an automated mechanism to study this very language, taking inspiration from natural language models.

The four nucleotide bases in the DNA molecule carry information that determines when and where proteins are generated in the body. Transcription factors bind to specific regions and activate or inhibit gene activity. The human genome contains information about the tools needed to build and maintain an organism, as well as information about the individual's risk of developing common illnesses. If we know the rules to the grammar for a bio-linguistic system, we can use that grammar to create new functional proteins or molecular machines. We can also read and write the manufacturing instructions for the "body plan" of bacteria, plants, animals, and the human body.

These ideas are further established and discussed throughout the text and the transformer model is developed based on these ideas. It is tested on the first chromosome of the human genome for different number of layers in the transformer model.

Contents

List of Figures

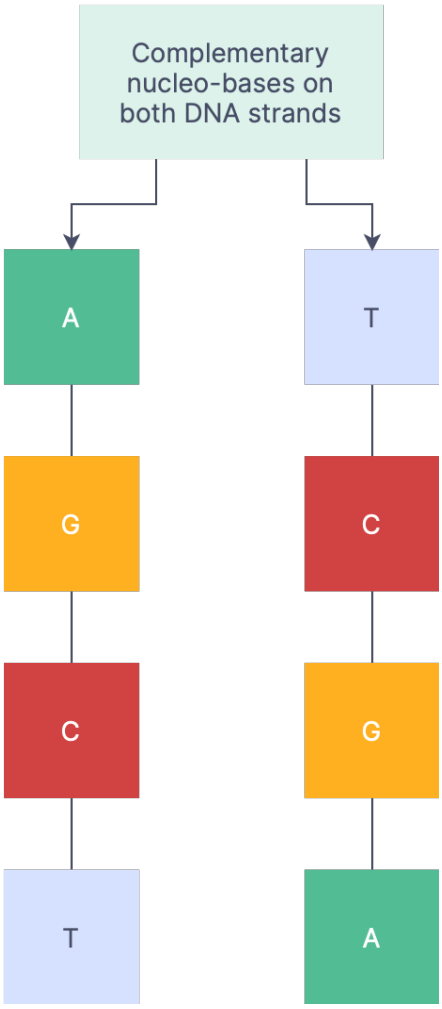

Figure 1: Basic vocabulary tokens of the genomic language

1 The Genome Language and Transfer Learning

The four nucleotide bases adenine (A), cytosine (C), guanine (G), and thymine (T), which might be thought of as the letters of the genomic language, carry information in the DNA molecule (Olson et al., 1989), (Gurard-Levin and Almouzni, 2014). DNA words are short sequences of letters that determine when and where proteins are generated in the body. Almost every cell in the human body has the letters arranged in the same order. Distinct genes are active (expressed) in different cell types, allowing them to perform their specialised functions, such as those of a brain cell or a muscle cell. The key to this gene control is transcription factors, which are DNA-binding proteins that bind to specific regions and activate or inhibit gene activity.

The genome not only gives us the information about the tools needed to build and maintain an organism, but it also contains details of the individual's risk of developing common illnesses such as diabetes, heart disease, and cancer. If the ability to read and understand the human genome is improved, the rapidly accumulating genomic information about

many diseases for medical purposes will be able to be utilised in a much more holistic and effective manner.

Pre-training of a model architecture helps the model in understanding the data before being "trained" for the task at hand (Kalyan et al., 2021). The objective, thus, here is that we generate a model that would already have a basic understanding of the human genomic language, so that a pre-trained model can be used for other fine-tuning tasks (Du et al., 2021). In a normal neural network training task, one starts by normally initialising the weights randomly, and as soon as training starts, the weights are changed as the model is optimised in order to reduce the difference between the actual ground truth and the outputs (Shang and Wah, 1996). Now, if one wants to train the model with another, more specific data, instead of repeating everything and starting with randomly initialised weights, one can start with where one left off with the previous, more general, training procedure. In doing so, the model can use the previous knowledge that it has accumulated to learn for specific tasks faster. This procedure is called transfer learning (Weiss et al., 2016), (Zhuang et al., 2020).

2 Is DNA a Language?

If we remove any cultural, historical and linguistic nuance from a natural language, it boils down to a sequence of words or tokens that have some meaning. The genome language, not unlike a natural language, has a sequence of tokens (nucleotides, codons, genes, based on the perspective from which one is observing). Necessarily, the sequence of these tokens has to have some meaning to it, since it is paramount in deciding the specific spatial and temporal phenotypic expressions seen in any organism.

However, is it justifiable to carry out computations on genome language based on these loose definitions? It is, thus, a subject of keen interest to justify using natural language models for understanding genome behaviour. As a consequence, one is interested in knowing if there is some niche characteristic that natural languages share in common with the genome language.

2.1 Zipf's Law

In any corpus written in the English language, the most commonly occurring word is "the", followed by "and", followed by "of", and so on. It is also observed that the frequency of the second-most frequent word, i.e., "and" is about half the frequency of the most frequent word "the"; and the frequency of the third-most frequent word, i.e. "of" is about a third of the frequency of the the most frequent word (Adamic, 2000),(Montemurro, 2001). If we generalise this behaviour, we can say that for words, the rank-frequency behaviour is an inverse power distribution as shown in Fig. 2. This is a remarkable property shared by any natural language that has ever existed. This is true even in case of languages that are undeciphered as well as constructed languages. Linguistically, the reason for this behaviour is not very well understood (Brillouin, 1959). However, with statistical and

mathematical points of view, this law can be corroborated.

2.1.1 Intuitive Statistical Understanding

We can statistically explain this phenomenon with an intuitive understanding of probability. If one considers a random sequence of characters in the Latin alphabet, one could consider that each alphabet cluster that is not separated by a space would be a word, denoted by w. The length of this word, i.e., the number of characters that w is made of, can be denoted by $|w|$. Since, there are 27 possibilities (all the alphabet characters along with the space),

$$\mathcal{P}(|w| = 1) = \frac{1}{27}$$

i.e., the probability of existence of a word with length 1 is $\frac{1}{27}$. Similarly,

$$\mathcal{P}(|w| = 2) = \frac{1}{27^2}$$
$$\mathcal{P}(|w| = 3) = \frac{1}{27^3}$$
$$\mathcal{P}(|w| = 4) = \frac{1}{27^4}$$
$$\dots$$
$$\mathcal{P}(|w| = n) = \frac{1}{27^n}$$

This shows that as the length of a word increases, the probability for that word to exist decreases exponentially, in case of random gibberish. Of course, natural language is not that straightforward, thus, these equalities don't hold. However, they become proportionalities.

So, while this approach does not explain the behaviour of natural languages in being "Zipfian", it helps in making statistical conclusions on them. For random gibberish, the criteria for the distribution was the length of the word, but it could be a myriad of different aspects for natural languages. For the scope of this study, the existence of this behaviour can aid us in comparing natural languages with the genome language: if such patterns can exist in random gibberish, then necessarily they should exist in evolutionarily conserved coding sequences of the genome.

2.2 Zipf like Behaviour with a Codonic Perspective

While there are several schools of thought debating on weather the comportment of DNA be considered as a natural language, which makes sense, because a DNA does not have any similarities with the natural linguistics, for most practical purposes. The genomic "language" is a physical code but not a real language – one can code a natural language in several codes but that does not make those codes languages. Our DNA has no implicit

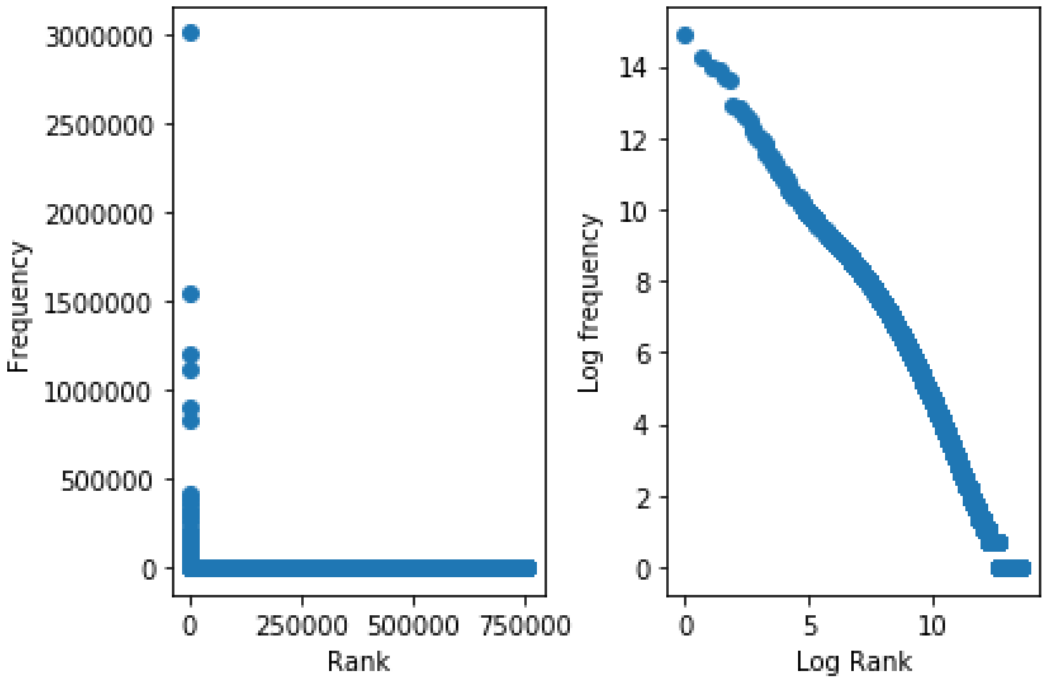

Figure 2: Frequency-rank distribution from the corpus of English language Wikipedia

literature. On the other hand, some others argue that "literature" does not really have a fixed definition, and the argument that a language needs to encode literature is a shallow one.

The code sequence (signal or text) is not a language. A language is a set of rules for constructing and interpreting (meaningful) representations of information, which consist of basic coding elements that represent basic concepts. The rules for expressing and interpreting information in natural language code can only be derived from text examples or told stories. DNA codons are a basic coding element and represent amino acids as a basic concept. The information expressed is information about how functional proteins and functional assemblies are constructed at a higher level. Perhaps there are rules for recognising or creating new, high-quality DNA codes that enable the production of functional proteins and the like. Like natural language, these bio-linguistic rules can only be derived from examples of DNA code. It is important to remember that the DNA code not only specifies amino acids and sequences proteins, but also how molecular machines and organisms are constructed.

Analogy of Grammar with Bio-Linguistic Systems
We know codons, so if we know the rules to the grammar for a bio-linguistic system as discussed above, we would know how to encode their structure into DNA code for newly designed functional proteins or molecular machines. In addition, one should be able to read and write the manufacturing instructions for the "body plan" of bacteria, plants, animals, and the human body (assuming the ribosome accepts our code).
For a working bio-linguistic system, not only the rules and code, but also the creator of the new code (which can be copied or invented by applying the rules) and the interpreter who can apply the rules to do something meaningful should have a comprehensive regulatory mechanism. In the case of this bio-language, interpreting machines are robot-like ribosomes that can use code as instructions for building components (such as tRNAs and proteins), and machines and organisms that are instructed to assemble from those components. This bio-linguistic system is similar to a language system that directs a robot complex, but is more than a robot that builds an assembly because it also directs the construction of the robot itself and directs it to copy the robot itself. The only missing element is the organism that invents a new machine and encodes its body plan.

While the argument for Zipf law is developed in natural languages, can this feature be also found in the genomic languages? This can be seen from a codonic perspective. When studying the frequencies of different codons observed in different genomes, it is found that the it shows a Zipfian pattern, as exhibited in Fig. 3. For the computational aspects of considering the DNA as a language, this fact suffices for us to move on with the the computations.

3 Theory of Language Models

3.1 Natural Languages

A **natural language** can be defined as a language that has evolved naturally over time, as a contrast to artificial or computer languages (Winograd, 1972). So, any language that we use to write or speak with is a natural language. For the sake of this discussion, only writable natural languages are considered since these languages are far easier to compare with the genomic languages than, say, sign languages.

In any written natural language, communication is done through a series of **tokens** which have a meaningful significance. For example, in the English language, each token is a word, i.e., a cluster of alphabets that is separated by a space. Without this basic understanding, one cannot begin to understand the nuances of, in this case, the English language. Thus, while analysing any natural language, it is paramount to identify and recognise what these tokens are.

In natural language processing, the primary motive is usually for the machine to be able

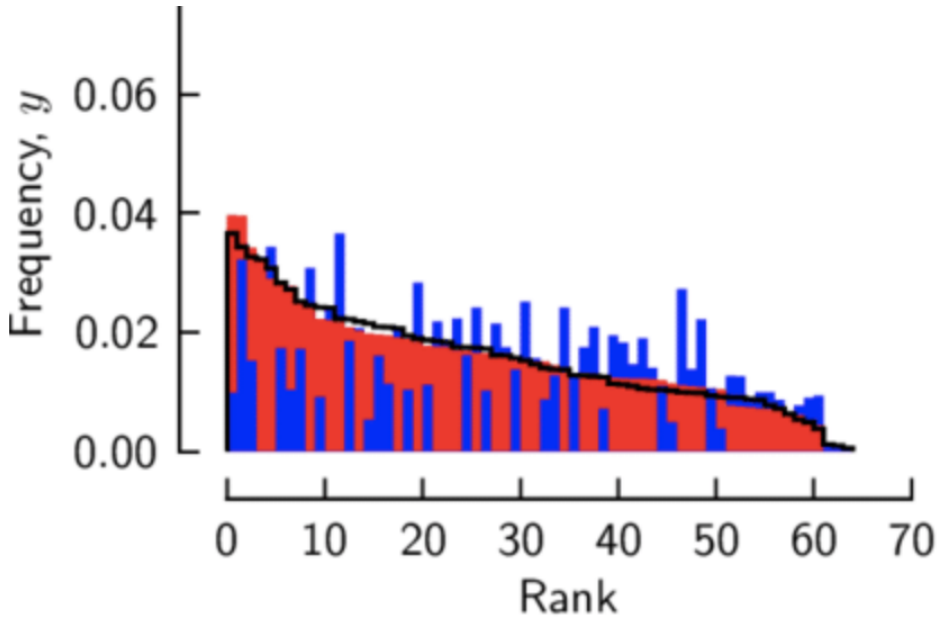

Figure 3: Zipfian patterns in the genome: frequency distribution of codons in humans (red) and *Arabidopsis thaliana* (blue); Source (Khomtchouk and Nonner, 2019)

.

to understand the language at hand, i.e., to make use of statistical analyses to essentially teach the language to the machine (Nadkarni et al., 2011). This is where the role of language models comes into play.

3.2 Language Models

Simply put, a language model (Zhai, 2008) is a system that predicts the next word in a sequence of words. It inadvertently means studying the probability distribution of several tokens that are present in the vocabulary so as to understand which token comes after the other. So, in other words, it gives us the probability of the next token, when a sequence of tokens is already given to us. A language model seeks to present whether the next possible token is, in some sense, "valid" or not – validity here does not necessarily refer to grammatical validity but empirical validity: how things are observed instead of how things should be. Broadly, there are two kinds of natural language models – probabilistic and neural network based. In more recent years, transformers have gained recognition. In this section, a sequential analysis for each of these methodologies is discussed and an argumentation is built for a critical appreciation of transformers.

3.2.1 Probabilistic natural language models

These are usually constructed using n-grams (hence n-gram models), which are sequences of n tokens, $n > 0$. Intuitively, the probability of the next word would be dictated by the

ratio of the number of times the entire sequence including the next word is observed in the textual corpus to the number of times the entire sequence without the next word is observed in the textual corpus. Mathematically, this gives –

$$\mathcal{P}(w_n|w_{n-1}, ..., w_0) = \frac{C(w_n, ..., w_0)}{C(w_{n-1}, ..., w_0)}$$

where C denotes the counts of the sequence in the corpus. This model has a Markovian assumption, in that, it assumes the context is depended on only the previous n tokens. It is evident why this kind of a model would not be very strong in the context of natural languages, since complicated texts have deeper contexts stretching far back than just a finite number of tokens. Further, from a computational standpoint, the scaling of such models would be extremely poor since as n increases, the number of possible permutations that can be done increases exponentially. Moreover, as vocabulary size increases, the problem of granularity starts to get introduced, i.e., the probability of all the tokens starts to converge at the same value, which is not really helpful.

3.2.2 Neural network based natural language models

These are computationally more complex than the aforementioned probabilistic language models – with Markovian assumptions, there is a caveat of memorylessness which is not useful when studying languages, since the entire basis of languages is memory Mikolov et al. (2012). The key idea is to create embeddings in the language space for each token in the vocabulary, pass a continuous vector of these embeddings through some matrix computations, and reach to what would be the most appropriate subsequent word.
The embedding vector e would be passed onto one or more hidden layers, from which an output distribution would be obtained by taking, say, the softmax of the computations. Mathematically, the hidden layer vector h would be given by,

$$h = f(\mathbf{W}e + b_1)$$

where $\mathbf{W}$ is the matrix containing the weights of the hidden layer and b_1 is the bias of this layer. Further, the output $\tilde{y}$, at this stage, would be the distribution of h,

$$\tilde{y} = softmax(\mathbf{U}h + b_2)$$

where $\mathbf{U}$ is the matrix containing the weights of the output layer and b_2 is the bias.
While this circumvents the problem of sparsity, it does not account for the problem of context. One can, however, account for context through the use of **recurrent neural networks** or RNNs. RNNs have an abstract concept of sequential memory (Petroni et al., 2019). So, instead of feeding the entire embedding vector to the hidden layer, only one token is fed at a time, and as the subsequent tokens are fed, the distributions from the previous tokens are simultaneously fed in. As one can imagine, RNN computations are extremely heavy and introduce the problem of vanishing gradients.

3.2.3 Transformers

The fundamental disadvantage of RNN-based architectures is that they are sequential. As a result, because there is no way to parallelise large sequences, training times skyrocket. The transformer architecture is the answer to this dilemma.
A transformer is a deep learning model that uses the **self-attention** mechanism to weight the importance of each element of the input data differently. Transformers, like recurrent neural networks (RNNs), are built to handle sequential input data like natural language for tasks like translation and text summarisation. Transformers, unlike RNNs, do not always process data in the same order. The attention mechanism, on the other hand, provides context for any point in the input sequence. If the input data is a natural language sentence, for example, the transformer does not need to process the first part of the sentence before the last. Rather, it detects the context that gives each word in the phrase its meaning. Because this feature enables for higher parallelisation than RNNs, training times are reduced.
In the experimentation for this study, computations are done, thus, by giving priority to transformer based mechanisms.

4 Pre-training Computational Mechanism

4.1 Attention is all you need

Any neural machine translation mechanism (sequence-to-sequence-model, that which takes a series of tokens as input and gives a series of tokens as outputs) uses two kinds of hidden structures – encoders and decoders. Each input token is processed by the encoder. The information captured by the encoder, called context, is sent to the decoder, which produces the output, one token at a time. **Context** is a vector for the case of machine translation, which invokes the concept of **attention**. Working with such contexts proved to be a limitation for most fundamental RNN based mechanisms, as discussed previously. Thus, the solution was proposed which introduced the idea of attention in the field of language models (Bahdanau et al., 2014), (Luong et al., 2015).

In an attention model (Vaswani et al., 2017), the encoder passes all the hidden states to the decoder, instead of just the last hidden state. Now, the decoder looks at the set of all hidden states (instead of just one, as in the non-attention models), gives scores to each of these hidden states, and multiplies each of these states with their respective scores (obtained from softmax) (Tetko et al., 2020). This very procedure helps in scaling the importance of different hidden states – it amplifies the priority of the hidden states with better scores while diminishing the priority of the hidden states with lower scores (Wolf et al., 2019). It is of significant interest to note that this machine translation is not just sequential aligning of translated words but because of the associated context, the mechanism actually learns the language as it is used in the corpus.

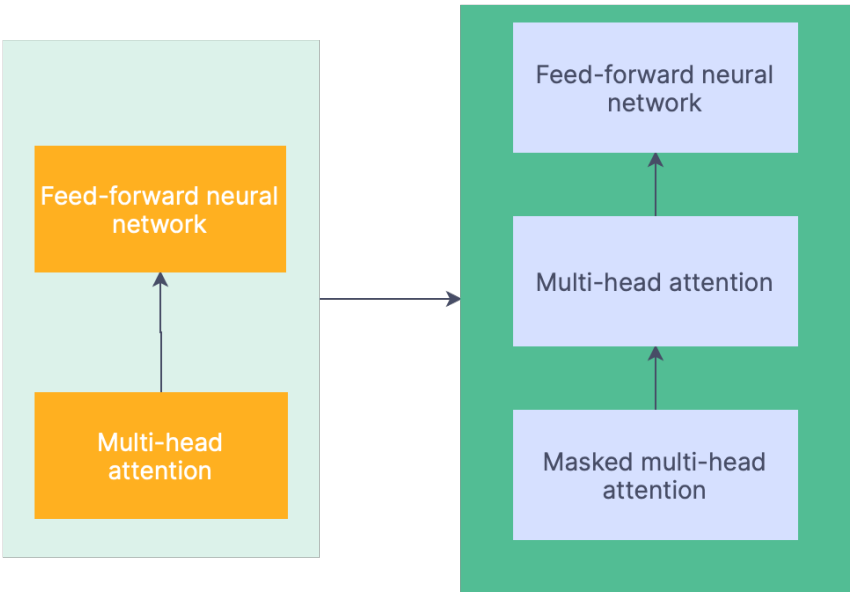

Figure 4: The encoder-decoder relationship of a simple transformer

4.2 Theory of Transformer

Fig. 4 shows a simplified version of the basic architecture of a transformer language model. In general, there are encoder and decoder stacks that are used subsequently. The input embeddings are passed into the first encoder, which are transformed and propagated to the next encoder, and finally, the output from the last encoder in the encoder-stack is passed to all the decoders in the decoder-stack. In addition to the self-attention and feed-forward layers, the decoders also have one more layer of Encoder-Decoder Attention layer. This helps the decoder focus on the appropriate parts of the input sequence.

The transformer building blocks are dot-product attention units that have been scaled. When a segment is fed into a transformer model, attention weights are calculated for each token at the same time. The attention unit generates embeddings for each token in context that include information about the token as well as a weighted combination of other relevant tokens, each weighted by its attention weight. A simple transformer-based language model learns three things – query weights W_Q, key weights W_K, and value weights W_V. One ought to multiply the input embedding x_i, for a token i, with each aforementioned matrix to produce three vectors – query vector $x_i W_Q = q_i$, key vector $x_i W_K = k_i$, and value vector $x_i W_V = v_i$. One ought to, then, calculate the attention weights using the query and the key vectors – the attention weight a_{ij} from token i to token j is the dot-product between q_i and k_j, divided by the square-root of the dimension of the key vector d_k. A softmax is taken of this vector to get the distributions of the same.

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

W_Q and W_K being different matrices permits the attention to be non-symmetric, which

means that if one token affects the emergence or the position of another token , that does not mean that this influence would be reciprocated by the other token for the first token taken into consideration. This helps in a much more nuanced understanding of the language.

Multi-head Attention
An attention head is referred to as one set of the three matrices required for the transformer to learn, (W_Q, W_K, W_V). On the one hand, whereas a specific attention head accounts for the tokens that are relevant to each token, the model can do this for different definitions of "relevance" with multiple attention heads. Furthermore, in successive layers, the influence field representing relevance can become progressively dilated. Many transformer attention heads encode meaningful relevance relations for humans. Attention heads, for example, can pay attention primarily to the next word, whereas others pay attention primarily to verbs and their direct objects. Each attention head's computations can be performed in parallel, allowing for fast processing. The attention layer's outputs are concatenated and fed into the feed-forward neural network layers.

4.3 DNABERT

The concept of DNABERT (Ji et al., 2021) stems from the idea of BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) which focuses on the idea of understanding the context of the language from both the directions, i.e., it argues that the presence of a token at any given position is influenced by tokens that are present, both before the given position as well as after. This, of course, thus, cannot "generate" text, but has proven to work remarkably well for other tasks. For this task at hand, since we don't necessarily need to generate the genomic language, but rather we need the mechanism to learn the human genomic language and find inferences, so we develop on this transformer mechanism.

Any language model technique aims to comprehend natural language. In the case of DNABERT, this usually entails predicting a codon in a blank. To accomplish this, models are typically trained using a large repository of specialised, labelled training data. This necessitates arduous manual data labelling by linguist teams, when it comes to natural language corpuses, and labelled genomic data when it comes to the genomic language – which in turn makes it paramount to brainstorm what necessitates the structural integrity of the data for the pipeline. Data pipelining is further discussed in subsequent sections.

The transformer is the principal model component that allows DNABERT to understand context and ambiguity in the genomic language. The transformer accomplishes this by processing any given k-mer (token) in relation to all other k-mers in a sequence, rather than one at a time. It is the transformer that allows the DNABERT model to understand the full context of the k-mer by looking at all surrounding k-mers (proximal context), allowing it to achieve better comprehension.

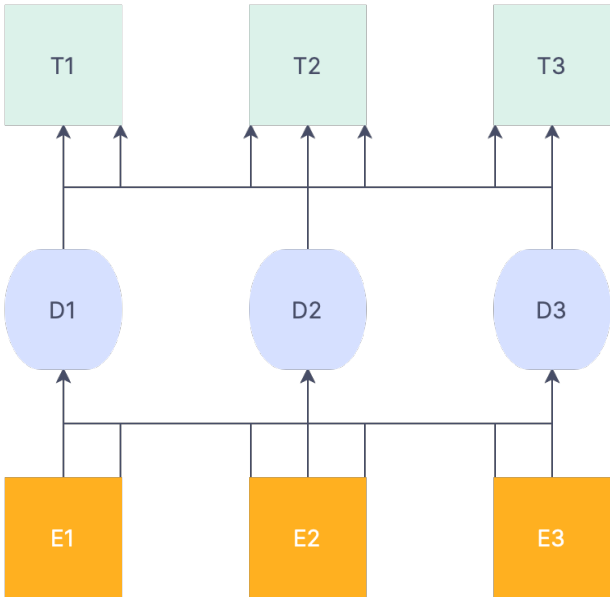

Figure 5: Basic transformer concept in BERT – each embedding E_i governs the presence of each hidden embedding D_i, and each hidden embedding influences each token T_i

This is in contrast to the traditional method of language processing known as word embedding, in which previous models would map every single word to a vector, which represents only one dimension of the meaning of that word. Large datasets of labelled data are required for these word embedding models. While they excel at many general natural language processing tasks, they fall short when it comes to the context-heavy, predictive nature of question answering, because all words are tied to a vector or meaning in some way. BERT (or in the case of genomic language, DNABERT) employs a masked language modelling technique to prevent the word in focus from "seeing itself" – that is, from having a fixed meaning independent of its context. BERT is then forced to identify the masked word solely on the basis of context.

4.3.1 Data Pipelining and Motivation

The entire human genome has recently been, finally, sequenced. In actuality, until very recently, around 8% of the genome was missing (Genovese et al., 2013), (Kidd et al., 2010), (Chaisson et al., 2015) – the coded part of the human genome consisted mostly of the euchromatin, the regions of the genome that sit together in a loose package, making it relatively easy for sequencing technology to unpack and have a look at it. It generally includes most actual genes or protein coding regions of DNA. Heterochromatin, however, are tightly packed regions of the DNA. This very fact, has historically made it difficult for the mechanisms to be able to encode these parts of the genome (Grewal and Jia, 2007). Euchromatin was easier to analyse with the tools available at the time, and though there were plenty of researchers interested in heterochromatin, the complexity of decoding it made it less of a priority.

The newest methods can read many longer sequences at once by virtue of using newer sequencing methods (Nurk et al., 2022). Most sequencing methods cannot read a whole chromosome from start to end, but read smaller sequences and paste them as and when they overlap. However, being able to read longer sequences means that there is less of a need to check for overlaps, and hence a lesser likelihood of errors arising. The result of these efforts is the 3.055 billion base-pair sequence. This also, henceforth, present certain new genes in the 8% of the human genome that was previously undeciphered. It is also a well-supported argument that heterochromatin isn't after all, junk DNA. It is highly probable that heterochromatin is heavily implicated in organising the genome, thereby, in many ways, controlling which genes are expressed and which are silenced, and even helping with DNA replication and repair.

A Naïve Segment Approach – DNA Sequence Chunks and Hexamers
At this point, it is of keen interest to understand what the tokens for this genomic language should be. A simple manoeuvre to this question would be to divide the genome sequences into codons – while in a perfect scenario, each nucleotide could be considered as a separate token, working with such a data would be extremely computationally heavy, and unnecessarily long. A good workaround would be to use codons, i.e., nucleotide pairs of three. This, however, raises the issue of loosing any kind of meaning when we divide the sequence into segment chunks. It might happen that just shifting the reading window by only one nucleotide, due to some computational error, would completely change the arrangement of all the upcoming subsequent codons. This would, inevitably, be counterintuitive to all the computation that is being carried out. We, hence, employ the concept of using hexamers. These are each tokens of six nucleotides with a sliding window of one nucleotide. First, the entire genome is divided into chunks – the definition of these chunks is open to interpretation, in most computations, a chunk is taken to be a chromosome, while in others, these are broken sequences of a specific number of nucleotides. From the perspective of natural language, a chunk can be considered as a paragraph. The segment-wise division of the genome language takes place in the structural hierarchy level of these chunks, i.e., each chunk consists of several segments chopped based on a specific number of nucleotides. Considering a genome sequence chunk as follows: `ATGGCCAATAGCCCATAAGGCCCACT`

The different segments, if we start from position 0, and consider the nucleotide length for each segment to be 7, would be `ATGGCCA, ATAGCCC, ATAAGGC, GCCCACT`. Notice that the last segment is `GCCCACT` instead of `CCACT` in order to conform to the segment length value. Thus, overlapping is incorporated in this method. Further, for data augmentation, we incorporate several other starting positions besides 0. Say, if we start from 2 in this genome chunk, we start with `GGCCAAT, AGCCCAT`, and so on. Now, for a segment, say, `ATGGCA`, the hexamers would be `ATGGCC` and `TGGCCA`. As is evident, there are several layers of overlapping when incorporating the pipelining of the data. These several layers help the model to learn better. It is a better approach than simple breaking the genome sequence, also because it improves the sensitivity of the model (Du et al., 2019).

For the pre-training of this model, ChIp-sequences (Marinov, 2017) of the human genomes are read, along with the upstream and the downstream sequence. The significance of this ideation is discussed in the further sections of the text. The set of three sequences – sequence upstream to a ChIp-seq read, the ChIp-seq read in question, and the sequence downstream to the particular ChIp-seq read are considered as an instance to the pipeline.

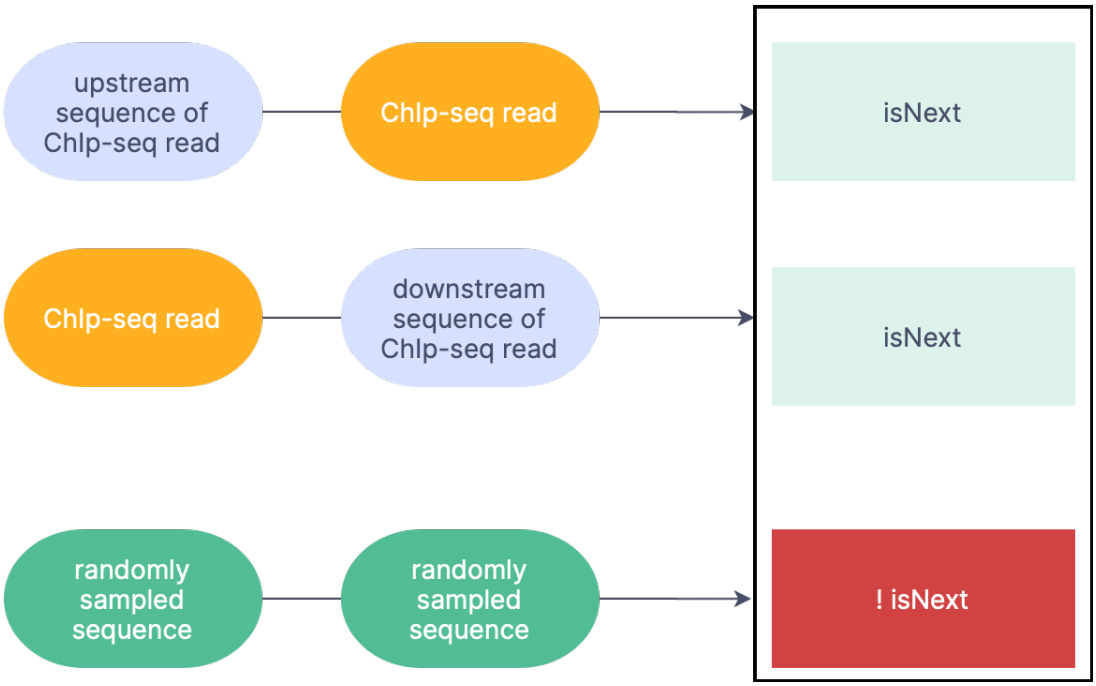

Figure 6: Classification mechanism for understanding distal context in the human genomic language – two consecutive sequences have a positive ground truth value while randomly sampled sequences have a negative ground truth value

4.3.2 Distal Context

Distal context is the context in this language of the human genome that is influenced by the base-pairs that are at a distance from the window of sequence in question, i.e., it helps in answering the question, "How does a sequence a few 500 base-pairs far away decide the functioning of the sequence that is being considered?" It is, thus, in this context that we require the sequences upstream and downstream of different ChIp-seq reads. One can consider this "sentences" to the pipeline. From the perspective of natural languages, this can be considered as the coherence that a group of sentences brings to the corpus of the text (Shi and Demberg, 2019), (Liu et al., 2019). The idea here, is to always be able to predict the next segment from the previous one.

The inevitable question here is, "How is one ought to characterise sentences in the genomic language?" The genomic language, albeit being considered as a language, does not have the concept of words and sentences/segments as one does in terms of natural languages. A naïve manoeuvre implemented was to break each instance randomly into segments and carry out the pre-training as one would have done for natural languages.

This, while working substantially well, gives rise to many problems – while doing this makes perfect sense from the perspective of informatics, computation and information theory (Deepakumara et al., 2001), biologically, it makes little sense. A segment in the DNA can be influential in many goings-on, and randomly chopping these sequences might hinder us from reaching the full potential of the model architecture. A good definition of

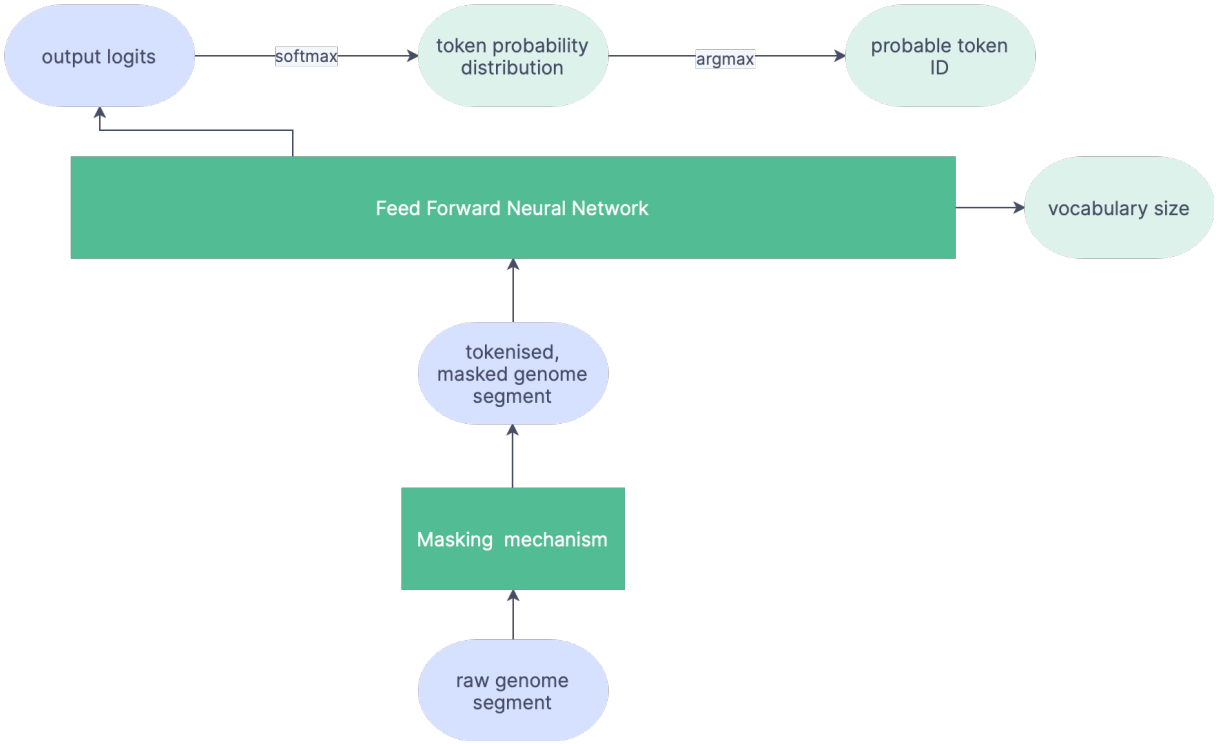

Figure 7: Reconstruction mechanism for proximal context

a sentence is a sequence of tokens which has a meaning of its own, largely independent of the tokens that are used in it. This, in terms of the genomic language, can be seen by considering annotated regions as sentences. Hence, we look into the ChIp-seq reads of the human genome.

In terms of computation, it is a classification problem – given two segments, the model ought to find if the second segment is indeed the next segment to the first one or not. So, as it happens with BERT, 50% of the time, two consecutive segments are taken, and the other 50% of the time, two completely random segments are taken. For former, the label `isNext` is 1, and for the latter, it is taken to be 0. With this information and the ground truth values, a binary classification scheme is set up for understanding the distal context, as shown in Fig. 6. While there still remains a level of ambiguity since one is essentially, either predicting annotated sequences from un-annotated ones or vice versa, the mechanism seems to work fairly well as is discussed in the subsequent sections.

4.3.3 Proximal Context

With distal context, the model architecture is able to understand the surface level context of the genomic language. To further enhance the comprehensibility of the architecture, the concept of proximal context is employed. In the perspective of natural languages, this can be considered as a word-level understanding of the language. As discussed above, a hexamer-level word embedding is used in the experimentations.

Computationally, this is a reconstruction problem, wherein, the model is given a segment and is trained to output the same segment, except that the input has a few tokens that are masked. The idea of the model is to learn what these masked tokens are and reconstruct what the given segment was. Theoretically, it is similar to the tasks of auto-encoders and self organising maps. The filling in the gaps is done for the model to understand the deeper context in the genomic language than just the distal context. Arguably, working with proximal context is more computationally expensive than with distal context.

Out of all the tokens in the segment, 15% of the tokens are selected to be "masked" which ought to be predicted. A simple manoeuvre to be able to predict the masked tokens is to delete that token from the original segment and replace it with `<mask>`. However, the issue with this manoeuvre is that when this model is used for fine-tuning, say, for promoter classification, the token `<mask>` will arguably not come up. In order to eschew this lack of congruence between the pre-training of the model and the fine-tuning for a specific task, there are three possibilities –

1. 80% of the time, the token is replaced by `<mask>`,
2. 10% of the time, the token remains unchanged,
3. 10% of the time, the token is replaced by a randomly sampled token – this noise makes the architecture to reduce its bias to the masked token during bidirectional context encoding. This prevents the model from overfitting (Lin et al., 2020), (Wu et al., 2019).

The entire mechanism is illustrated in Fig. 7. After the masking mechanism as discussed above, the masked segment tokens are sent in to a feed-forward network, which outputs the token logits. On taking the softmax of these logits, we get the probability distribution, of which, taking the argmax gives the most probable token.

5 Computation Results and Discussion

As pre-training entails training the model on the entire corpus of the text, it is extremely time consuming and energy demanding. For the sake of these results, we, thus, only look at the first chromosome of the human genome to have a dummy data, so as to have a basic understanding of the working of the model before actually making changes and manipulating the model based on the entire human genome. FIg. 8 shows the behaviour of the loss functions of the proximal context, the distal context, as well as the general model loss (which is an arithmetic addition of both the proximal and the distal context) for different number of hidden ayers of the model.

In Fig. 8 however, since the difference in the change of the convergence points is very low, one is unable to clearly decipher how changing the number of layers affects the model loss. It is, albeit, evident that loss for the distal context is always much lower than loss for the proximal context. This is coherent since, as discussed above, learning the distal

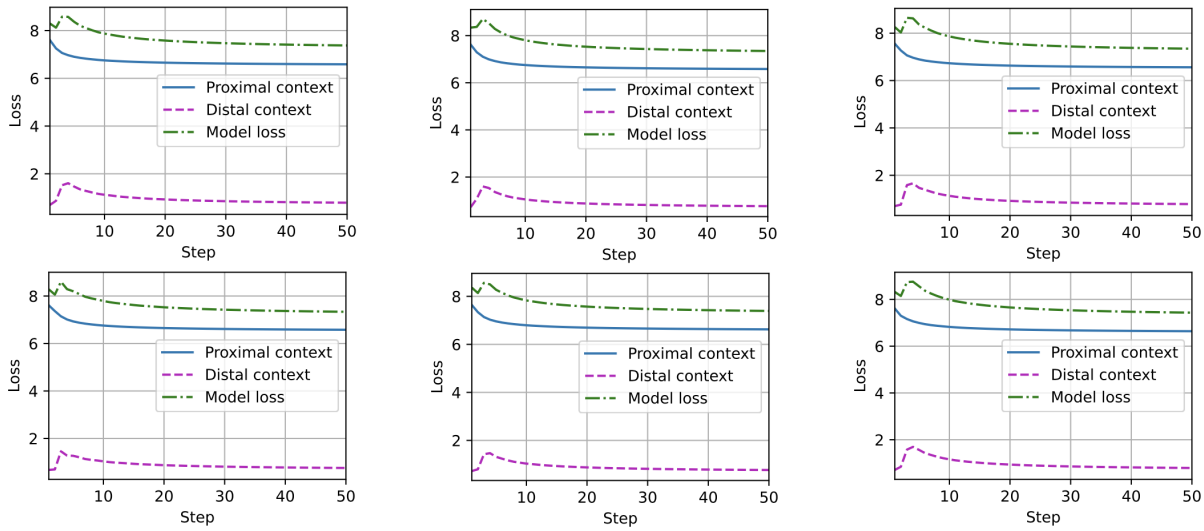

Figure 8: Behaviour of loss functions, both for the distal as well as the proximal context for 2, 3, 4, 5, 6 and 7 hidden layers of the neural network (row-wise)

context for the model is nothing but a binary classification. We also see the loss function first increase, attain a maximum and then steadily decrease to converge. For proximal context, however, the loss has a steady decreasing trend before attaining convergence.

Interpreting an Early Rise of the Loss
It is paramount to have a clear idea of what the significance of the loss function actually is – it is merely a function metric of error for each step of the training process. When we see an early rise in the loss function, it thus means that the model was actually learning something, but the effect of that learning was somehow detrimental to the model optimisation, i.e., it was learning the wrong thing (in the sense of soft computing, it was recognising the wrong patterns – it was perhaps recognising patterns that weren't necessarily important for the mapping of the function to the ground truths that a data point is associated with). The initial training, would thus, learn to associate a completely unrelated pattern to the variable `isNext` and evidently, it results in a spike in the loss. This is, however, quickly mitigated through the same process which penalises the model for associating wrong patterns to the output, and thus the loss quickly starts to decrease and converge.

FIg. 9 shows the trends of the losses of proximal and distal contexts. For proximal context, as the number of layers increases, the loss first decreases until the 4-layer model, after which it fluctuates a little bit before reaching a constant. Evidently, in this case, the 4-layer model works the best. For distal context, the model loss also decreases until the 4-layer model, after which, although it has a general decreasing trend, it still fluctuates before reaching a constant. While, rudimentarily it would make sense to choose a model with 8 or 9 layers in this case, the fluctuating trend does not make these models reliable. Further, since the distal context loss is significantly lower than the proximal context loss, it makes more sense to give more priority to a model which minimises the proximal context

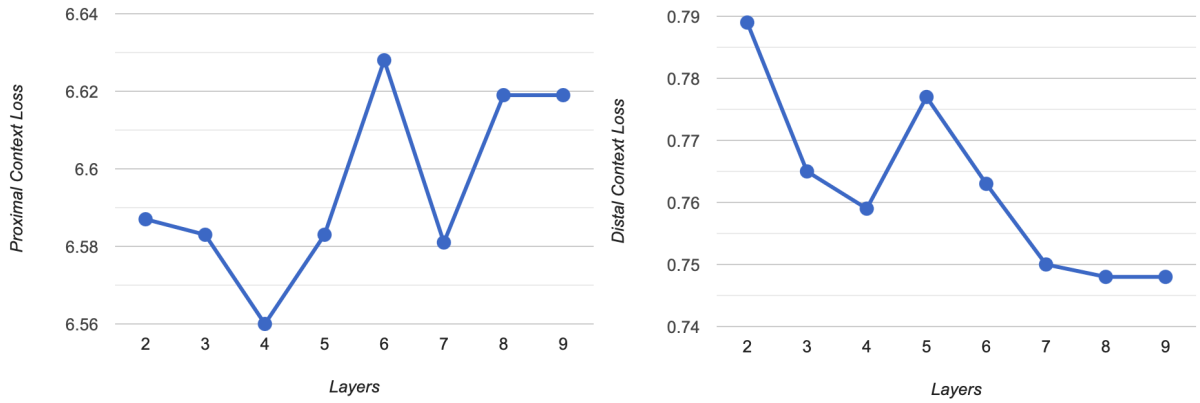

Figure 9: Proximal and Distal context loss trends as the number of hidden layers increases

loss better.

6 Conclusion

In these experimentations, a transformer-based language model was developed and pre-trained for understanding the human genome language in order to have a transfer learning-based mechanism for carrying out specific taks, like, say, promoter classification. A model based on both a high-level (distal) and a low-level (proximal) context is developed. In order to do that, it is established how the genome language can be compared and contrasted to natural languages, and the concept of natural language is discussed. Different layered models are subsequently tested out and an optimum model is chosen. For the subsequent steps, the pre-trained model will be tested out for carrying out further fine-tuning tasks to systematically corroborate the effectiveness of the developed model. Moreover, a multi-modal model will be developed with several other modalities of data so as to make the model even better than it already is.

References

Adamic, L. A. (2000). Zipf, power-laws, and pareto-a ranking tutorial, *Xerox Palo Alto Research Center, Palo Alto, CA, http://ginger. hpl. hp. com/shl/papers/ranking/ranking. html* .

Bahdanau, D., Cho, K. and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473* .

Brillouin, L. (1959). *La science et la théorie de l'information*, Masson.

Chaisson, M. J., Wilson, R. K. and Eichler, E. E. (2015). Genetic variation and the de novo assembly of human genomes, *Nature Reviews Genetics* **16**(11): 627–640.

Deepakumara, J., Heys, H. M. and Venkatesan, R. (2001). Fpga implementation of md5 hash algorithm, *Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No. 01TH8555)*, Vol. 2, IEEE, pp. 919–924.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* .

Du, N., Chen, J. and Sun, Y. (2019). Improving the sensitivity of long read overlap detection using grouped short k-mer matches, *BMC genomics* **20**(2): 49–62.

Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z. and Tang, J. (2021). All nlp tasks are generation tasks: A general pretraining framework, *arXiv preprint arXiv:2103.10360* .

Genovese, G., Handsaker, R. E., Li, H., Altemose, N., Lindgren, A. M., Chambert, K., Pasaniuc, B., Price, A. L., Reich, D., Morton, C. C. et al. (2013). Using population admixture to help complete maps of the human genome, *Nature genetics* **45**(4): 406–414.

Grewal, S. I. and Jia, S. (2007). Heterochromatin revisited, *Nature Reviews Genetics* **8**(1): 35–46.

Gurard-Levin, Z. A. and Almouzni, G. (2014). Histone modifications and a choice of variant: a language that helps the genome express itself, *F1000prime reports* **6**.

Ji, Y., Zhou, Z., Liu, H. and Davuluri, R. V. (2021). Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome, *Bioinformatics* **37**(15): 2112–2120.

Kalyan, K. S., Rajasekharan, A. and Sangeetha, S. (2021). Ammus: A survey of transformer-based pretrained models in natural language processing, *arXiv preprint arXiv:2108.05542* .

Khomtchouk, B. B. and Nonner, W. (2019). Gaussian-distributed codon frequencies of genomes, *G3: Genes, Genomes, Genetics* **9**(5): 1449–1456.

Kidd, J. M., Sampas, N., Antonacci, F., Graves, T., Fulton, R., Hayden, H. S., Alkan, C., Malig, M., Ventura, M., Giannuzzi, G. et al. (2010). Characterization of missing human genome sequences and copy-number polymorphic insertions, *Nature methods* **7**(5): 365–371.

Lin, C., Bethard, S., Dligach, D., Sadeque, F., Savova, G. and Miller, T. A. (2020). Does bert need domain adaptation for clinical negation detection?, *Journal of the American Medical Informatics Association* **27**(4): 584–591.

Liu, J., Cheung, J. C. and Louis, A. (2019). What comes next? extractive summarization by next-sentence prediction, *arXiv preprint arXiv:1901.03859* .

Luong, M.-T., Pham, H. and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation, *arXiv preprint arXiv:1508.04025* .

Marinov, G. K. (2017). Chip-seq for the identification of functional elements in the human genome, *Promoter Associated RNA*, Springer, pp. 3–18.

Mikolov, T. et al. (2012). Statistical language models based on neural networks, *Presentation at Google, Mountain View, 2nd April* **80**: 26.

Montemurro, M. A. (2001). Beyond the zipf–mandelbrot law in quantitative linguistics, *Physica A: Statistical Mechanics and its Applications* **300**(3-4): 567–578.

Nadkarni, P. M., Ohno-Machado, L. and Chapman, W. W. (2011). Natural language processing: an introduction, *Journal of the American Medical Informatics Association* **18**(5): 544–551.

Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A. V., Mikheenko, A., Vollger, M. R., Altemose, N., Uralsky, L., Gershman, A. et al. (2022). The complete sequence of a human genome, *Science* **376**(6588): 44–53.

Olson, M., Hood, L., Cantor, C. and Botstein, D. (1989). A common language for physical mapping of the human genome, *Science* **245**(4925): 1434–1435.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H. and Riedel, S. (2019). Language models as knowledge bases?, *arXiv preprint arXiv:1909.01066* .

Shang, Y. and Wah, B. W. (1996). Global optimization for neural network training, *Computer* **29**(3): 45–54.

Shi, W. and Demberg, V. (2019). Next sentence prediction helps implicit discourse relation classification within and across domains, *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 5790–5796.

Tetko, I. V., Karpov, P., Van Deursen, R. and Godin, G. (2020). State-of-the-art augmented nlp transformer models for direct and single-step retrosynthesis, *Nature communications* **11**(1): 1–11.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need, *Advances in neural information processing systems* **30**.

Weiss, K., Khoshgoftaar, T. M. and Wang, D. (2016). A survey of transfer learning, *Journal of Big data* **3**(1): 1–40.

Winograd, T. (1972). Understanding natural language, *Cognitive psychology* **3**(1): 1–191.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. et al. (2019). Huggingface's transformers: State-of-the-art natural language processing, *arXiv preprint arXiv:1910.03771* .

Wu, X., Lv, S., Zang, L., Han, J. and Hu, S. (2019). Conditional bert contextual augmentation, *International Conference on Computational Science*, Springer, pp. 84–95.

Zhai, C. (2008). Statistical language models for information retrieval, *Synthesis lectures on human language technologies* **1**(1): 1–141.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q. (2020). A comprehensive survey on transfer learning, *Proceedings of the IEEE* **109**(1): 43–76.