Cambridge Large Two (April282025), 1–5

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE TYPE

DNA Core Promoters identification through BERT

ID: 39603547

Abstract

Gene expression is one of the most recurrent investigation cases as genetic material is the origin of almost all known forms of life. In fact, understanding how gene expression is regulated and mediated is essential to control protein behavior inside the human body itself. Some sequences inside DNA itself are relevant in the initialization of transcription, which is the underlying step leading to protein synthesis. These fragments are called core promoters, and influencing them is critical in the functioning of gene expression. This report aims to analyze whether natural language processing tools such as BERT (Bidirectional Encoder Representation from Transformers) can discriminate between promoter and non-promoter fragments in complete human DNA genome sequences.

1. Introduction

Nowadays, the paradigm of genetic material is dominated by deoxyribonucleic acid (DNA). This polymer is at the origin of life. DNA is composed of a double-helical structure of chains that contain complementary bases. These bases are the elementary units of DNA, Citosine (C) is complementary to Guanine (G) and Adenine (A) to Timine (T) [1]. Transcription is a definitive step in protein coding; it is the production of a missager ribonucleic acid (mRNA), which is a complementary strand to DNA which is used for the protein translation process. Core promoters are specific sequences within DNA that are recognized by RNA polymerase II to mediate the transcription stage. These sequences support the initialization of gene transcription [2]. This is why developing mechanisms to detect which fragments are promoters or not is determinant to understand gene expression regulation.

The purpose of this study is to try to replicate natural language processing (NLP) strategies to DNA sequence classification task. The challenge arises in implementing a field that has been traditionally used in computer science, linguistics and social sciences [3] in the bioengineering field. Different methodologies will be used to adjust DNA sequences to corpus data standard requirements and build a model that predicts promoter sequences within raw genetic material sequences.

2. Dataset

A dataset of merged genomic sequences that meets the requirements of our analysis is not directly available. For that reason, extensive research is carried out that stores human genomes composed of a low number of base pairs (such as transmembrane proteins) and presents coding sequences. In *Table 1*, the different genomes selected to conform the dataset are detailed. This dataset merges the genomic sequences downloaded in FASTA format [4]. The most probable core promoters sites extracted from an algorithm created by the Department of Health Technology (DTU) of the Technical University of Copenhagen [5].

Every genomic sequence is added to the main dataset as well as the labels of every single nucleotide. The bases are tagged as promoters in all the positions going 100 indexes upstream from the main promoter site predicted. From the dataset, depending on the tokenization step stated, the tokens and labels associated with each token are extracted. It is then necessary to construct a vocabulary from all the tokens, labeling every different combination of nucleotides with a single identifier. The labels and identifier lists from the dataset are converted to tensors, which is the desirable structure to be included in the BERT analysis, which involves Pytorch library.

Table 1. Genomic sequences in the dataset.

Genomic Sequence	Brief Description
Human Globin	Human globin genes
Human Mitochondrial Genome	Mitochondrial DNA of Homo sapiens
C12orf24	Chromosome 12 open reading frame 24
C7orf25	Chromosome 7 open reading frame 25
C1orf198	Chromosome 1 open reading frame 198
C22orf23	Chromosome 22 open reading frame 23
Leukosialin	Leukocyte surface membrane protein
TMEM's	Different transmembrane proteins

3. Methodology

3.1 Tokenization techniques

The tokenizer is the tool that allows us to breakdown the genomic sequences to smaller and significant subsets, which are called tokens. Tokenization is an original technique from NLP which is based on compressing one piece of text to the smallest number of tokens that can represent the vocabulary contained in the text [6]. It is possible to extrapolate the use of this methodology to our problem of study. In fact, 3 main procedures are tested in DNA sequences in this article.

Single Tokenizer is the most simple approach to spread the sequences in elementary representative units. This alternative shows some positive points such as simplicity, uniformity and direct patterns transference. However, it is straightforward to understand why this solution is not suitable for a BERT analysis. Only 4 possible combinations are not enough to explain the

presence or absence of a promoter site, there are relevant combinations that could be explainable lost in the process. In the same line, this solution lacks biological context towards combinations of nucleotides indicative of a transcription starting site (TSS).

The second approach consists of tokenizing the sequences in k-mers, where k is the number of bases that make up each token. All elements of the sequence are introduced in at least k tokens from the total number of results of the tokenization. The main advantage of this method is the ability shown in capturing local patterns, which is important when it comes to detecting recurrent combinations that mark the beginning of a core promoter site. It is more probable that less dependencies occur in the long term, as each token also accounts for the biological context present in the whole sequence. Unfortunately, this procedure also presents some drawbacks. The loss of fine resolution occurs because of the difficulty in aligning the prediction of a promoter site with single nucleotides. The tokens overlapping introduced by this method add ambiguity to the classification problem, generating very similar or even equal tokens in close positions. The last and most important concern related to this alternative is the final extension of the vocabulary, generated by the exorbitant number of different tokens generated. This induces a higher computational expense from the model and considerably slows down the training time. Although there are complications, k-mer tokenization seems to be the most suitable and compatible to BERT.

The last method implemented is Byte-Pair Encoding (BPE). BPE is understood as a combinatorial optimization problem [7]. In our context, BPE is explained as an algorithm that grows the global vocabulary extracting iteratively the most frequent pair sequences. BPE runs until the desired number of iterations or vocabulary size is reached. This process ensures that the resulting vocabulary is made up of the most recurrent combinations. As in the other cases, some advantages and disadvantages arise after applying this method. BPE, as has been mentioned, generates tokenization based on the highest combinatorial frequencies, which highly optimizes the vocabulary. Moreover, it provides an interesting flexibility of token length, allowing different size tokens in the same subset. However, BPE is not compatible with biological problems. In other words, the frequency of tokens is not equal to the significance they have from a biological perspective. Additionally, it shows the same alignment difficulties as the k-mer tokenization.

The most suitable method that can be applied in our model is without any doubt the k-mer tokenization technique. The size of the vocabulary represents a definitive parameter when it comes to building and training the model. An analysis of the total number of tokens generated as a function of the tokenization procedure employed is depicted in *Figure 1*. As expected, the single tokenizer is the most simplistic method,while the k-mer tokenizer increases the vocabulary size in an almost exponential manner as the value of k increases. BPE is not taken into account in this partial analysis because the number of tokens is an hyperparameter itself of the method.

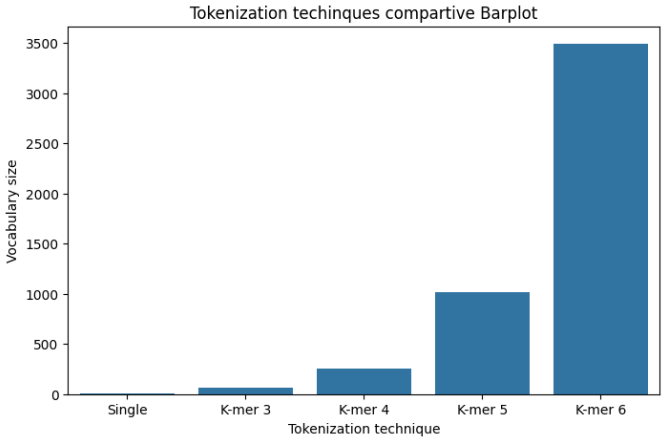

Figure 1. Vocabulary size against the tokenization approach.

3.2 DNA-BERT model

BERT is a transformer-based model that is normally applied in NLP to perform predictive tasks with corpus data. Transformers are structures that allow the processing of text sequences in parallel, which is differential compared to the other models [8]. The main challenge of this project is to apply BERT in the biological context of Genomics. Treating DNA sequences as corpus data and expecting that the model will be able to classify them as gene promoters is a very ambitious idea. This is consistent with the extrapolation of the knowledge captured by the model concerning semantic roles to DNA sequences [9].

As this model is not directly focused to linguistic problems, the model does not follow the conventional path BERT would follow in next sentence prediction algorithms. Self-supervised tasks such as masked language modeling are skipped. As there is no entire universe of words behind, the pre-training phase is not performed, whereas the typical fine-tuning step is carried out [10].

The number of tokens is very considerable. For this reason and due to the absence of a pre-training step, all the tokens identifiers and labels are distributed into equally sized churns. In this manner, the model could be trained sequentially without crashing the RAM memory on the process. A standard size accepted by the memory storage is selected for the length of the chunks and a certain overlapping span on every chunk with respect with the next is added too, just to ensure some consecutive sequence patters are retained.

The final dataset is created with the chunks containing the tokens identifiers and the labels for each token. This is an essential step to make sure that the data disposition follows the requirements of the Pytorch library. The data is randomly split into training and test sets, including all the chunks.

The model is initialized with the definition of some hyperparameters that are briefly explained in *Table 2*. Once all the chunks are created with the token identifiers and labels stored in tensors, the input sequences must be formatted to 3 different

embedding types. Token embedding reinforces what has been done before with the allocation of an identifier to each token, segment embedding provides the separation of the tokens based on the presence or absence of promoter site and position embedding delivers a position for every embedding in the sequences. These embedding layers convert the meaningless identifiers of tokens into continuous vectors compatible to neural networks.

Table 2. Hyperparameters of DNABERT model.

Hyperparameter	Description
`size_vocabulary`	Total number of tokens in the vocabulary.
`embedding_dim`	Dimension of the projected space.
`max_len`	Maximum input sequence length.
`n_heads`	Number of attention heads.
`n_layers`	Number of encoder layers.
`n_classes`	Number of output classes.
`dropout`	Dropout probability for regularization.

Table 3. Tuned parameters for the training step.

Parameter	Value	Description
`batch_size`	32	Samples per training update.
`num_epochs`	10	Full passes through the dataset.
`learning_rate`	1×10^{-4}	Step size for weight updates.
`num_classes`	2	Number of output categories.

To ensure the robustness of the model against the possibility of overfitting, dropout is used as a regularization technique. During training, some components of the input are set to 0. This is done to force the model to not be as reliable on single neurons or individual features. Then, right after that, a normalization layer is applied to the inputs to make sure that every single input follows a normal distribution and the scale is consistent across the layers.

The design of the encoder is the fundamental step in the model, as it is the basic unit which is repeated. The functioning inside an encoder consists of applying multi-head self-attention to the inputs and processing and transforming the upcoming attention information. The encoder layer attends every token in the sequence simultaneously. This is a way to capture relationships between tokens in parallel and understand which is the contextual place of every token inside the sequence. A simple feed-forward neural network is then applied to every token position to properly recover the information coming from the attention encoding. Finally, the resulting embeddings are converted into class scores. A summary of the model architecture and general workflow is exemplified in the diagram in *Figure 2*. Training the BERT model follows the classical approach of Deep Learning models. The main parameters that control the training process are described in *Table 3*. Every data point from the training set is divided in batches. During one epoch, the model accesses once every batch, which contains a smaller subset of the training data. The model starts to predict almost randomly the promoter site classes. In every case, going through every batch, the model better adjusts the weights and learns the patterns more closely.

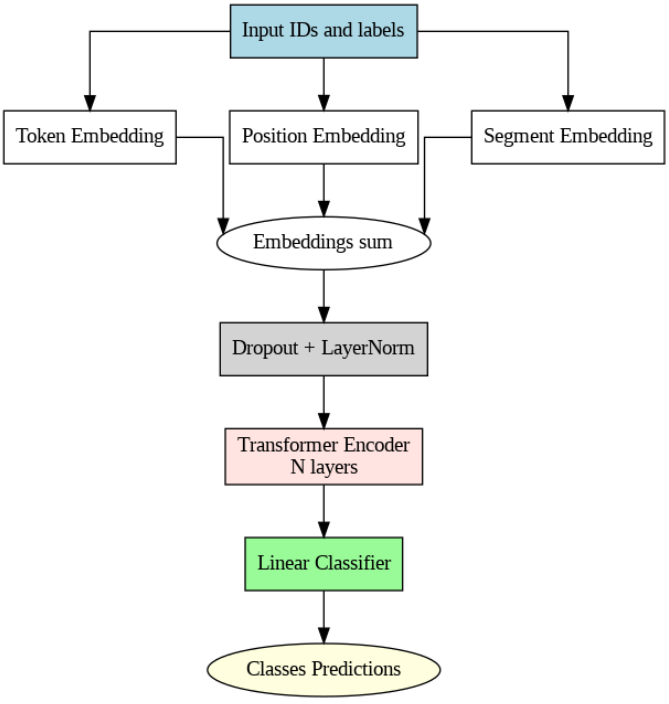

Figure 2. Pipeline of DNABERT.

The model is trained with sequences tokenized in 6 nucleotides and the same hyperparameters depicted in *Table 3*. *Figure 3* shows that training loss decreases drastically during the first two epochs and then experiences a plateau until the iterative process finishes. In terms of validation loss, it starts lower than the training loss and remains almost constant during all epochs. The number of epochs can be lowered to avoid some computational and temporal cost introduced in the model. Concerning the learning rate, a higher value could help to allow bigger steps in weight updates and reach convergence before. On the other side, the choice of the batch size seems to be consistent with the model.

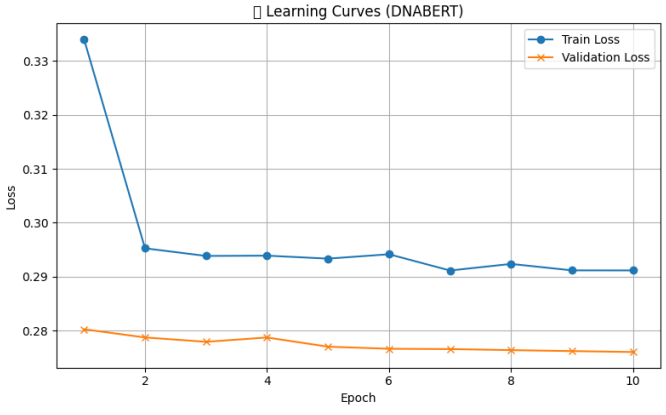

Figure 3. Training and validation loss representations for 6-mer tokenization.

The optimizer used for the model is the Adam optimizer, which is considered as the most suitable for transformer-based models. At every iteration, the model performs the predictions based on the inputs. The classification error is computed. The model is inspected backward to calculate the loss gradients towards the parameters and weights are updated based on that values to improve the predictive power. In every new step, the gradients are erased to be computed again with the current parameters.

The DNABERT instance created to perform model training is tuned with an embedding dimension of 128 and a maximum length of 1024 elements in the sequences. Apart from that, 4 transformer layers and attention heads are used to characterize the model. The classification criterion metric for distinguishing between the 2 classes is the cross-entropy loss (*Equation 1*). This is a metric assessing the difference between the predicted classes and the actual values.

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{C} \gamma_i \log(\hat{\gamma}_i) \tag{1}$$

4. Results & Discussion

Of all tokenization models that could be included in our approach, k-mer stands out as the most convenient one. If the aim is to prioritize the performance of the classifier, compromising computational efficiency and temporal cost, this solution is the one that captures the best the biological contextual patterns.

Figure 3 demonstrates that the model is working effectively during the training phase. In some cases, the validation can be significantly lower than the training loss due to a higher level of data complexity in the training. In contrast, the substantial difference between the training and validation loss curves could be explained by the strong dropout regularization applied to prevent overfitting.

BERT is an algorithm that is typically used to carry out Next Sentence Prediction (NSP) or similar tasks related to corpus data, employing Masked Language Modeling (MLM), which consists of the prediction of hidden words. The created model aims to benefit from the abilities demonstrated by BERT in the linguistics field in the prediction of genomic elements. DNABERT is coded as a classifier instead of a word predictor, that is why the vocabulary extracted from it is cleaner and more repetitive and limited than for the natural language. One of the main constraints that BERT shows when used for DNA sequences is the length sequences. These tend to be larger and more uniform. The last aspect affected by this genomics extrapolation of BERT is the semantic difficulty. Corpus data requires rich, complex and ambiguous language processing, taking into account all the existing disciplines and linguistic variations. In contrast, DNABERT has to adapt to repetitive patterns and capture them, as the semantic variability of DNA sequences is not large.

The decision about the optimal number of elements for every token depends on several factors. *Table 4* summarizes the performance of the model for different k values. From it can be extracted that the accuracy remains high for all the model configurations, although it stands out for k = 6. It is then computationally and temporally cheaper to select the 3-mer configuration, as it can be considered as the one that needs the lowest computational and temporal cost and delivers a similar performance than the other configurations.

The question is to understand whether 3-mer sequences are able to provide enough biological context to ensure trustful predictions. Deeper analysis on other metrics must be applied in further analyses to make sure that data imbalance worsens the prediction of promoter sites. The best option might be to select 6-mer tokenization, as it is the one achieving the highest accuracy from all the models, with a value of around 0.93.

Table 4. Accuracy of DNABERT model against the tokenization method.

Value of k	Test Accuracy
3	0.9123
4	0.9190
5	0.9179
6	0.9308

5. Conclusions

The main point that can be extracted is that natural language processing techniques such as Bidirectional Encoder Representation from Transformers (BERT) can be successfully implemented in promoter site classification tasks in DNA sequences. K-mer tokenization is the most suitable and biologically meaningful manner to create sequence subsets to be introduced in the models. A very similar architecture used to perform Next Sentence Prediction (NSP) can be extrapolated to design a BERT model to classify DNA sequences presence or absence of promoter sites. Hyperparameter tuning is determinant to create an appropriate model, training and validating the DNABERT instances.

In the trade-off between model performance and computational efficiency, the best predictive power is achieved on 6-mer tokenization technique, while the most efficient model with a reasonable high accuracy is 3-mer. It is mandatory to improve the model and ensure data imbalance is not conditioning the proper classification of DNA sequences. From a biological point of view, the model performance demonstrates that tokenizing DNA sequences into 6 nucleotides provides the greatest contextual information to certainly abroad promoter classification.

References

[1] Strömbergsson, H. (2015). Neural mechanisms of speech perception: Inferences from imaging and intervention studies. *The FEBS Journal*, 282(19), 3704–3717. https://doi.org/10.1111/febs.13307

[2] Angermueller, C., Pärnamaa, T., Parts, L., & Stegle, O. (2016). Deep learning for computational biology. *Molecu-*

lar Systems Biology, 12(7), 878. https://pmc.ncbi.nlm.nih.gov/articles/PMC6205604/

[3] Zhang, X., et al. (2024). DNABERT2: Efficient and accurate DNA sequence modeling with an enhanced BERT architecture. *arXiv preprint*. https://arxiv.org/abs/2409.19505

[4] National Center for Biotechnology Information (NCBI). https://www.ncbi.nlm.nih.gov/

[5] Knudsen, S. (1999). Promoter2.0: A neural network based algorithm for predicting promoters in structural genome projects. *Nucleic Acids Research*, 27(17), 4321–4325. https://services.healthtech.dtu.dk/services/Promoter-2.0/

[6] Wang, Y., et al. (2024). Transformer-based models for genomic sequence prediction: A survey. *arXiv preprint*. https://arxiv.org/abs/2402.18376

[7] Liu, F., et al. (2023). Improving DNA sequence understanding via pretraining with k-mer tokenization. *arXiv preprint*. https://arxiv.org/pdf/2306.16837

[8] Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A Primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8, 842–866. https://aclanthology.org/2020.tacl-1.54.pdf

[9] Warstadt, A., Singh, A., & Bowman, S. R. (2020). What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8, 34–48. https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00298/43535

[10] Vaswani, A., et al. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30. https://arxiv.org/abs/1706.03762