Bioinformatics, 2024, **40(4)**, btae196
https://doi.org/10.1093/bioinformatics/btae196
Advance Access Publication Date: 12 April 2024
Original Paper

Sequence analysis

Effect of tokenization on transformers for biological sequences

Edo Dotan[1,2], Gal Jaschek[3], Tal Pupko [2,*], Yonatan Belinkov[1,*]

[1]The Henry and Marilyn Taub Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa 3200003, Israel
[2]The Shmunis School of Biomedicine and Cancer Research, George S. Wise Faculty of Life Sciences, Tel Aviv University, Tel Aviv 69978, Israel
[3]Department of Genetics, Yale University School of Medicine, New Haven, CT 06510, United States

*Corresponding authors. The Henry and Marilyn Taub Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa 3200003, Israel. E-mail: belinkov@technion.ac.il (Y.B.); The Shmunis School of Biomedicine and Cancer Research, George S. Wise Faculty of Life Sciences, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: talp@tauex.tau.ac.il (T.P.)

Associate Editor: Alfonso Valencia

Abstract

Motivation: Deep-learning models are transforming biological research, including many bioinformatics and comparative genomics algorithms, such as sequence alignments, phylogenetic tree inference, and automatic classification of protein functions. Among these deep-learning algorithms, models for processing natural languages, developed in the natural language processing (NLP) community, were recently applied to biological sequences. However, biological sequences are different from natural languages, such as English, and French, in which segmentation of the text to separate words is relatively straightforward. Moreover, biological sequences are characterized by extremely long sentences, which hamper their processing by current machine-learning models, notably the transformer architecture. In NLP, one of the first processing steps is to transform the raw text to a list of tokens. Deep-learning applications to biological sequence data mostly segment proteins and DNA to single characters. In this work, we study the effect of alternative tokenization algorithms on eight different tasks in biology, from predicting the function of proteins and their stability, through nucleotide sequence alignment, to classifying proteins to specific families.

Results: We demonstrate that applying alternative tokenization algorithms can increase accuracy and at the same time, substantially reduce the input length compared to the trivial tokenizer in which each character is a token. Furthermore, applying these tokenization algorithms allows interpreting trained models, taking into account dependencies among positions. Finally, we trained these tokenizers on a large dataset of protein sequences containing more than 400 billion amino acids, which resulted in over a 3-fold decrease in the number of tokens. We then tested these tokenizers trained on large-scale data on the above specific tasks and showed that for some tasks it is highly beneficial to train database-specific tokenizers. Our study suggests that tokenizers are likely to be a critical component in future deep-network analysis of biological sequence data.

Availability and implementation: Code, data, and trained tokenizers are available on https://github.com/technion-cs-nlp/BiologicalTokenizers.

1 Introduction

Since the development of modern DNA sequencing technologies, there has been a rapid growth of available genomic data. While a relatively small bacterial genome such as *Escherichia coli* is roughly five million bases (Markowitz *et al.* 2012, https://doi.org/10.1093/nar/gkr1044), the complete sequence of a human genome is more than three billion bases long (Nurk *et al.* 2022, https://doi.org/10.1126/science.abj6987). Current large-scale protein datasets are growing at an exponential rate and already encompass hundreds of billions of amino acids (Steinegger and Söding 2018, https://doi.org/10.1038/s41467-018-04964-5). In light of the increasing size and length of biological sequence datasets, new processing methods are needed.

Deep-learning algorithms transformed many fields (LeCun *et al.* 2015, https://doi.org/10.1038/nature14539), including computer vision (Voulodimos *et al.* 2018, https://doi.org/10.1155/2018/7068349) and biomedical research (Rudas *et al.* 2023, https://doi.org/10.1371/journal.pdig.0000106). They were recently introduced to comparative genomics (Eraslan *et al.* 2019, https://doi.org/10.1038/s41576-019-0122-6, Koumakis 2020, https://doi.org/10.1016/j.csbj.2020.06.017, Talukder *et al.* 2021, https://doi.org/10.1093/bib/bbaa177, Alharbi and Rashid 2022, https://doi.org/10.1186/s40246-022-00396-x, Miller *et al.* 2022, https://doi.org/10.1038/s41467-022-33397-4). The use of deep learning for genomic analysis is a game-changer and gains momentum as neural network solutions usually outperform traditional algorithms (Kulmanov *et al.* 2018, https://doi.org/10.1093/bioinformatics/btx624, Jumper *et al.* 2021, https://doi.org/10.1038/s41586-021-03819-2). Both natural human languages and biological sequences are composed of discrete characters (letters and nucleotides, respectively). These characters are the building blocks of sophisticated structures, i.e. text and genomes, which include elements such as sentences and genes, respectively. Although natural language processing (NLP) architectures can be adapted to biological problems, considerable differences remain between human language and genomic data (List *et al.* 2016,

Received: 23 August 2023; **Revised:** 20 February 2024; **Editorial Decision:** 1 April 2024; **Accepted:** 11 April 2024

https://doi.org/10.1186/s13062-016-0145-2, Yu *et al.* 2019, https://doi.org/10.1073/pnas.1814684116, Dotan *et al.* 2023, https://openreview.net/forum?id=8efJYMBrNb). One major difference is that natural languages are typically composed of many different words, each composed of characters from a given alphabet, while DNA biological sequences are composed of long stretches of nucleotide characters and the definition of a word is not intuitive.

When analyzed using deep neural networks, long sequences raise memory consumption and run-time challenges. These challenges are held both when analyzing natural languages and biological sequence data. Different approaches to tackle these issues have emerged, including: (i) developing specific architectures for long sequences (Lin *et al.* 2021, http://arxiv.org/abs/2106.04554, Rao *et al.* 2021, https://proceedings.mlr.press/v139/rao21a.html); (ii) splitting the data into smaller segments (Dotan *et al.* 2023); (iii) *K*-mer representation of all possible nucleotides (Ji *et al.* 2021, https://doi.org/10.1093/bioinformatics/btab083).

In NLP, tokenization is the process of segmenting a running text into words or subword units (e.g. splitting "He's walking" into [He, 's, walk, ing]). Tokenization reduces the size of the vocabulary, which consists of a fixed set of items serving as atomic units. Tokenization may also help handling unknown words—e.g. if the word "walked" is unknown, splitting it to [walk, ed] may help associating it with the known unit "walk." Modern tokenization algorithms are data-driven and do not necessarily correspond to linguistically meaningful units. For example, the word "Bioinformatics," might be split into three subwords, "Bioin," "form," and "atics." Tokenization algorithms split such words into common subwords and thus enable NLP-based methods to put these tokens in context. This may result in increased ability to infer meanings. Subwords tokenizers include "Byte-Pair Encoding" (BPE), "WordPiece," and "Unigram" (Schuster and Nakajima 2012, https://doi.org/10.1109/ICASSP.2012.6289079, Sennrich *et al.* 2016, https://doi.org/10.48550/arXiv.1508.07909, Kudo 2018, http://arxiv.org/abs/1804.10959). The BPE and WordPiece tokenizers initialize a dictionary consisting of all the characters in the raw text, and progressively select pairs of tokens to merge and add them as a new token to the dictionary. BPE and WordPiece differ in how pairs are selected: while BPE adds the most frequent pair, WordPiece adds the pair that maximizes the frequency of the pair divided by the product of the frequencies of the two tokens (see Section 2). The Unigram methodology is different. It initializes a dictionary consisting of a very large number of relevant tokens. The dictionary is next trimmed by removing noncontributing tokens, which are inferred by applying a specific loss function (see Section 2).

While text tokenization of human languages such as English is a standard NLP methodology, when DNA sequences are analyzed, each nucleotide is typically considered a token. Thus, while human languages such as English contain very large dictionaries of thousands of tokens, genomic data contain considerably smaller dictionaries (four items). The number of tokens being processed is also different. Typical text in English can range from a few dozens to a few million tokens. This is in contrast to genomic data in which, using the dictionary of the four nucleotides, "A," "C," "G," and "T," the number of tokens representing the entire genome will be the number of nucleotides (Fig. 1a). One can think of different dictionaries, based on *K*-mers (Ji *et al.* 2021). For example, one that contains all the possible pairs: "AA," "AC," "AG," "AT," "CA" … "TT." This raises the size of the dictionary by a power of two (i.e. 16) and reduces the length of the sequences by approximately two folds (Fig. 1b). Of note, the conversion of nucleotides to codons has a similar impact as the dictionary size is 64 (61 sense codons and three stop codons) and the sequence length is reduced by a factor of three.

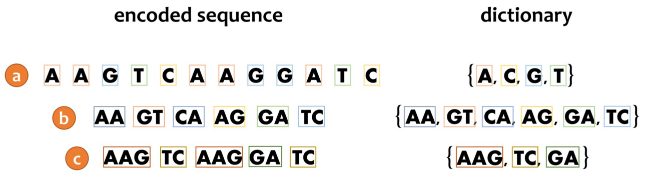


Figure 1. Different tokenization algorithms can be applied to biological sequences, as exemplified for the sequence "AAGTCAAGGATC." (a) The baseline "words" tokenizer assumes a dictionary consisting of the nucleotides: "A," "C," "G," and "T." The length of the encoded sequence is 12, i.e. the number of nucleotides; (b) the "pairs" tokenizer assumes a dictionary consisting of all possible nucleotide pairs. The length of the encoded sequences is typically halved; (c) a sophisticated dictionary consisting of only three tokens: "AAG," "TC," and "GA." Using this dictionary, the encoded sequence contains only five tokens.

The genome of each species contains various repetitive elements, which vary in type and length. We expect data-driven biological tokenizers to assign a token for such repetitive elements. Notably, repetitive elements comprise more than half of the human genome (Richard *et al.* 2008, https://doi.org/10.1128/MMBR.00011-08). The existence of such repetitive elements motivates the employment of data-driven tokenizers, which can substantially reduce the number of tokens to process without a substantial increase in the size of the dictionary. Of note, reducing the number of tokens partially alleviates the problems encountered with long sequences. However, too large dictionaries forbid capturing shared elements among sequences. Which tokenizer best balances between these two constraints is an open question. In this study, we focus on comparing transformers trained on data processed by different tokenizers, in terms of performance and input length.

2 Materials and methods

2.1 Outline

We aim to train and evaluate alternative tokenizers. The input is a set of biological sequences. Different biological tasks are considered, e.g. classifying the sequence into several categories. The tokenizers are applied to the input sequences creating a list of integers, which represent the different tokens. Such lists are used to train a deep-learning network model (in our case, a transformer). A single transformer (Vaswani *et al.* 2017, http://arxiv.org/abs/1706.03762) is trained for each biological task and each tokenizer. The performance is measured on transformer processing test data (which were processed with the same tokenizer as the training data). In addition, we report the effect on the number of input tokens, which is a proxy for memory and runtime consumption.

2.2 Tokenizers

We evaluated five different tokenizers on biological sequences: BPE, Unigram, WordPiece, "words," and "pairs." "Words" is a trivial tokenizer, in which the dictionary contains all possible amino-acids or nucleotides. In the "pairs" tokenizer, the dictionary contains all possible pairs of characters (amino-acids or nucleotides). Of note, while in "words" and "pairs" the dictionary size is fixed, it is a tunable parameter in the other three tokenizers. Thus, we tested various values for the dictionary size. For each computational task and for each combination of tokenizer and dictionary size, a different transformer was trained, and its performance evaluated (as described below). We would like to emphasize the differences in applying tokenizers to natural languages versus biological sequences. In most natural languages, there are three levels of text representation: characters, words, and sentences. In contrast, biological sequences have only two levels, as they lack the space character that separates words in natural languages.

2.2.1 BPE (byte-pair encoding)

Initially, BPE was a general-purpose compression algorithm (Gage 1994, https://dl.acm.org/doi/10.5555/177910.177914), but it has since been adopted for tokenizing textual data. It was first utilized for machine translation with recurrent neural networks (Sennrich *et al.* 2016) and later for transformers (Vaswani *et al.* 2017). The tokenizer creates a base vocabulary from unique characters in the pre-tokenized data and then gradually merges the most frequent pair, adding each new one to the vocabulary. This process stops when the vocabulary size reaches a hyperparameter that must be determined before training the tokenizer.

2.2.2 WordPiece

This tokenizer is similar to BPE (Schuster and Nakajima 2012). Like BPE, it uses the entire set of characters in pre-tokenized data to create a base vocabulary and progressively adds new tokens to it. Unlike BPE, which adds the most frequent pair, WordPiece selects the pair that maximizes a certain score calculated as:

$$\text{score} = \frac{f_{1,2}}{f_1 \times f_2}$$

Here, $f_{1,2}$ is the frequency of the pair of elements, while f_1 and f_2 are the frequencies of the two separate elements.

2.2.3 Unigram

Unigram (Kudo 2018) takes a different approach than BPE and WordPiece. It begins with a heuristic identification of an initialized vocabulary, which is later trimmed. There are different ways to create the initial vocabulary, e.g. selecting the most frequent sub-strings in the corpus or using a different tokenizer such as BPE with specific hyperparameters that yield a large vocabulary. Next, the Unigram tokenizer progressively removes tokens from the vocabulary by searching for tokens whose removal improves the model fit, as quantified using a loss function (detailed below). Usually, more than one token is removed at a time, since computing the loss for all tokens is a costly operation. Given a corpus of N words, $x_1, \ldots, x_i, \ldots, x_N$, the loss is the sum of the negative log-likelihood of the score of each word, denotated by $h(x_i)$ for the word i:

$$\text{loss} = -\sum_{i=1}^{N} \log(h(x_i))$$

where $h(x_i)$ is the maximum score of dividing the word x_i to tokens:

$$h(x_i) = \max_{z \in S(x_i)} (g(z))$$

$S(x_i)$ are all the possible options to split x_i to tokens, and g maps a specific set of tokens, i.e. option, $t_1, \ldots, t_j \ldots, \ t_M = z$ to a score.

$$g(z) = \prod_{j=1}^{M} p(t_j)$$

$p(t_j)$ is the unigram probability of token j, i.e. the number of occurrences of token j divided by the total number of tokens in the corpus.

2.3 Biological datasets

We compared the performance of the above tokenizers on eight datasets, described below, which vary in terms of their size, the learning task required (five classifications, two regressions, and one sequence alignment), and the type of sequence data (one dataset contains DNA sequences, while the others contain protein sequences).

Dataset1. Type III effectors (we will use the term effectors below) are proteins that are secreted by pathogenic bacteria from the bacterial cytoplasm into the host cell. In the host cell, they manipulate cellular processes to the benefit of the bacteria. The computational challenge is to classify bacterial proteins to those that are effectors and those that are not. The secretion signal that determines whether a protein is an effector or not was shown to reside in the 100 amino acids of the N-terminus of a protein (Notti and Stebbins 2016, https://doi.org/10.1128/microbiolspec.VMBF-0004-2015). We obtained a dataset of 641 effectors and 4544 noneffector proteins (Wagner *et al.* 2022, https://www.frontiersin.org/articles/10.3389/fpls.2022.1024405). From each protein, we only considered the 100 N-terminal amino acids. The true label (whether or not the protein is an effector) is known from experimental work. The computational task is to correctly classify each protein into its category. These data were divided into training, validation, and test, each containing 497, 60, and 84 effectors and 2034, 219, and 2291 noneffectors, respectively.

Dataset2. A superfamily is a group of proteins that share similar properties and functions. The second task is to classify proteins to superfamilies based on their amino-acid sequences. To this end, we downloaded sequences from the Pfam database (Mistry *et al.* 2021, https://doi.org/10.1093/nar/gkaa913). We randomly picked nine different superfamilies containing over 2000 protein sequences: (i) SSF100895 Kazal-type serine protease inhibitors; (ii) SSF110035 GDNF receptor-like; (iii) SSF109993 VPS9 domain; (iv) SSF101152 Mob1/phocein; (v) SSF110019 ERO1-like; (vi) SSF102546 RbsD-like; (vii) SSF101912 Sema domain; (viii) SSF100939 SPOC domain-like; (ix) SSF100879 Lesion bypass DNA polymerase (Y-family), little finger domain. For each superfamily we downloaded the first 2000 protein sequences, which we split into training, validation and test data, containing

1800, 100, and 100 sequences, respectively. The Pfam database includes information regarding the specific fragments issued with the superfamily. For each of the sequences, we extracted these fragments and concatenated them. The task is predicting the superfamily given the fragments.

Dataset3. Sequence alignment is one of the common tasks in bioinformatics (Van Noorden *et al.* 2014, https://doi.org/10.1038/514550a) as it provides a record of similarity between homologous sequences. One has to account for different evolutionary events such as insertions, deletions and substitutions to correctly infer the alignment. The third dataset contains pairwise homologous nucleotide sequences, and the task is to correctly align them. We have previously developed a deep-learning-based algorithm for such an alignment task, in which we train transformers to map pairs of unaligned sequences, i.e. source sentences, into a valid alignment, i.e. target sentences (Dotan *et al.* 2023). The average number of nucleotides is 429 and 434 for the source and target sentences, respectively. This dataset was simulated by SpartaABC (Loewenthal *et al.* 2021, https://doi.org/10.1093/molbev/msab266), and hence the correct alignment is known. The data include 395 000, 2000, and 3000 training, validation, and test alignments, respectively. To simulate those sequences, we have used the following parameters: (i) root length between 150 and 300 nucleotides; (ii) pairwise evolutionary distance between 0.05 and 0.15 substitutions per site; (iii) an insertion rate between 0.0 and 0.05 events per substitution and similarly for deletions; (iv) the "A parameter" dictates the length distribution of insertion and deletion events. The A parameter for insertions ranged between 1.01 and 2.0, and similarly for deletions. For each simulation of a pair of sequences, SpartaABC samples uniformly from those ranges and generates the alignment based on the sampled parameters.

Dataset4. Protein folds are characteristics of the protein 3D structure. Often, proteins evolve so that their sequence similarity becomes low, yet, they still share substantial structural similarity. Nevertheless, the folding information is encoded within the protein sequence. Here we analyzed 13 766 protein sequences, each of which is labeled by a specific fold (Hou *et al.* 2018, https://doi.org/10.1093/bioinformatics/btx780). There is a total of 1195 protein folds, and the computational task it to classify each protein to its correct fold based on its amino-acid sequence. These data were partitioned to 12 312, 736, 718 training, validation, and test pairs of sequence-fold.

Dataset5. The fluorescence intensity of a protein is determined by its sequence and structure. The general mapping form sequence space to fluorescence intensity is unknown in general. Here, we rely on previously established data (Sarkisyan *et al.* 2016, https://doi.org/10.1038/nature17995), which were partitioned to 21 446, 5363, and 27 217 training, validation, and test pairs of sequence-log-intensity values, respectively. Of note, unlike the previous tasks, here a regression model is needed from the sequence space to fluorescence intensities.

Dataset6. In stability landscape prediction, the challenge is to predict the concentration threshold from which the protein unfolds, given the sequence of amino-acids (Rocklin *et al.* 2017, https://doi.org/10.1126/science.aan0693). For this regression task, the data included 53 614, 2512, 12 851 training, validation, and test pairs of sequence-concentration, respectively.

Dataset7. This is a dataset for a fold prediction task, similar to dataset 4. However, here the classification is only to seven possible folds. These data were previously assembled (Andreeva *et al.* 2020, https://doi.org/10.1093/nar/gkz1064) and include 14 112, 1568, and 3921 training, validation, and test pairs, in which each pair includes a sequence and its associated fold. These data were taken from ProteinBERT (Brandes *et al.* 2022, https://doi.org/10.1093/bioinformatics/btac020). As these data did not contain a validation set, we randomly sampled 10% of the training data to serve as a validation data.

Dataset8. Neuropeptides are peptides that are used for communication between neural cells and their peripheral cells (Burbach and Peter 2010, https://doi.org/10.1016/j.ejphar.2009.10.015). The vast majority of neuropeptides are translated as precursor neuropeptides and undergo cleavage and maturation events. Ofer and Linial (2014) have previously assembled a database of precursor neuropeptides and nonprecursor neuropeptides from various animal species and developed a binary classification algorithm for predicting precursor neuropeptides given a set of protein sequences. We reanalyzed their data, which included 2727, 303, and 337 training, validation, and test pairs, in which each pair includes a sequence and whether or not it is a neuropeptide.

Of note, datasets 4, 5, and 6 were previously analyzed by Rao *et al.* (2019) and Brandes *et al.* (2022) and datasets 7 and 8 were previously analyzed by Brandes *et al.* (2022).

2.4 Tokenizer implementation

We compared five different tokenizers: BPE, WordPiece and Unigram (Schuster and Nakajima 2012, Sennrich *et al.* 2016, Kudo 2018) as well as the "words" and "pairs." The three first tokenizers can be trained for specific data, i.e. they are data-driven. These were trained (on the training data) with default parameters (e.g. parameters that control the trimming of the Unigram program). The output of this stage is a dictionary for each transformer and dataset. All datasets were next encoded using the obtained dictionaries. This step was achieved using the HuggingFace library (Wolf *et al.* 2020, https://doi.org/10.18653/v1/2020.emnlp-demos.6). For each tokenizer, we evaluate the following dictionary sizes: 100, 200, 400, 800, 1600, and 3200. The output of each tokenizer and sequence is a vector of tokens (represented as integer numbers). Different sequences are represented by a different number of tokens. In order for all vectors to be of the same length a maximum size was set. A fixed size is needed for applying positional embeddings (a modification step to the embedding matrix used to provide information regarding the ordering of the tokens) and for batching. Specifically, for dataset 1 the maximum size was set to 100 tokens. Similarly, for dataset 2, the maximum size was set to 512 tokens, for dataset 3 to 1024 tokens and for datasets 4–8, to 512 tokens. For all classification and regression tasks, proteins longer than this size were trimmed and proteins shorter than this size were padded with a special token. These encoded data were next used to train transformers (on the training data) for the specific computational task associated with each dataset.

2.5 Training the transformers

Two different transformer architectures were considered for all classification and regression tasks: BERT (Devlin *et al.* 2019, https://doi.org/10.18653/v1/N19-1423) and GPT

(Radford *et al.* 2018, https://www.mikecaptain.com/resources/pdf/GPT-1.pdf). As the former resulted in higher performance on the validation data, we only present results obtained with BERT. In each case, the models were randomly initialized and trained on the tokenized training data of each dataset. Using the validation data, we optimized several hyperparameters for each computational task: the number of layers, the number of attention heads, and the size of the hidden vector. The best performing configuration was with two hidden layers and two attention heads for all datasets. The size of the hidden vector was 128 for all datasets, except dataset 1, for which the optimal performance was with a vector of size 64. For each, dataset, tokenizer type, and dictionary size, we trained three transformers with different learning rates: 0.001, 0.0001, and 0.00001 and returned the one with the best performance. We used a constant scheduler in all analyses, i.e. the learning rate was fixed during the entire learning process. Transformer training and testing were implemented using the HuggingFace library (Wolf *et al.* 2020).

For dataset 3, which is associated with a sequence-to-sequence task, following Dotan *et al.* (2023), we relied on the "vaswani_wmt_en_de_big" architecture, with 6 hidden layers, 16 attention heads, and a hidden vector size of 1024. The training was conducted with the Fairseq library (Ott *et al.* 2019, http://arxiv.org/abs/1904.01038). The learning rate, warmup values, and max tokens were set to: 5×10^{-5}, 3000, and 4096, respectively.

2.6 Comparison with previous models

We compared the performance of the different trained models with those obtained in previous studies. Specifically, datasets 4, 5, and 6 were each previously analyzed by applying three different models: ProteinBERT (Brandes *et al.* 2022), the Tasks Assessing Protein Embeddings (TAPE) transformer (Rao *et al.* 2019, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7774645/), and a biological model that relied on the LSTM architecture (Rao *et al.* 2019). Datasets 7 and 8 were previously analyzed using ProteinBERT. These previous works all used the "words" tokenizer. As we did not pre-train our models, for a fair comparison, the performance of these previous models was evaluated without pre-training.

2.7 Evaluating the performance on the different tasks

For classification tasks, we report Accuracy (ACC), Area Under the Curve (AUC), and Matthew's Correlation Coefficient (MCC) [Matthews 1975, https://doi.org/10.1016/0005-2795(75)90109-9]. The latter is more suitable for unbalanced datasets as it considers the number of samples from each class:

$$\mathrm{MCC} = \frac{\mathrm{TN}\times\mathrm{TP} - \mathrm{FN}\times\mathrm{FP}}{\sqrt{(\mathrm{TP{+}FP})(\mathrm{TP{+}FN})(\mathrm{TN{+}FP})(\mathrm{TN{+}FN})}}$$

TN is the number of true negatives, TP true positives, FN false negatives, and FP false positives. We note that the range of MCC is from −1 to 1 while ACC and AUC range between 0 and 1.

For regressions tasks, we report the Spearman rank correlation, ranging from −1 to 1, where 1 reflects a perfect score.

$$\text{Spearman Correlation} = \frac{cov(R(X),\ R(Y))}{\sigma_{R(x)}\sigma_{R(Y)}}$$

X and Y are the predictions and the labels, respectively. $R(Z)$ is the ranking of Z, σ_Z is the standard deviation of Z, and $cov(Z,\ W)$ is the covariance of Z and W.

For the alignment task, we report the performance of the aligners with the Column Score (CS). The CS is measured by counting the number of columns in the inferred alignment that have a matching column in the true alignment, out of the total number of columns in the true alignment (Penn *et al.* 2010, https://doi.org/10.1093/molbev/msq066). The range of the CS is between zero and one. We also report the coverage score:

$$\text{Coverage} = \frac{VA}{TA}$$

where VA is the number of valid alignments and TA is the total number of alignments tested. When using transformers to align biological sequences, it is possible that the transformer erroneously creates invalid alignments. For example, if the input sequences are "AAG" and "AAC," the transformer may output a pairwise alignment in which "AA–G" is aligned to "AAC." This is clearly an invalid alignment as the number of characters in all alignment rows should be identical. In addition, each alignment row should be identical to the original corresponding (unaligned) sequence after removing all of its gaps. In rare cases, this is not the case, and these alignments are also considered invalid (Dotan *et al.* 2023). Of note, all alignments in the training data are valid alignments. Thus, higher coverage suggests better learning from the training data.

2.8 Hyperparameter optimization implementation

We conducted an analysis by evaluating each tokenizer across a grid of hyperparameter combinations. We focused on four key hyperparameters: layers (1, 2, or 4), attention heads (1, 2, or 4), hidden sizes (32, 64, or 128) and learning rate (0.00001, 0.0001, or 0.001). For every possible set of hyperparameters, we trained a transformer model, varying the tokenizer employed. Each model was trained for ten epochs. We evaluated the results on the superfamily classification task (dataset 2). Subsequently, we assessed the degree to which the selected hyperparameters influenced performance and investigated the broader implications of hyperparameter optimization on the comparative ranking of diverse tokenization methods. We compared the performance of a model in which one hyperparameter is fixed to an arbitrary value while the other hyperparameters are optimized to a model in which all four parameters are optimized. A few alternatives of this fixed value were evaluated.

2.9 Visualizing the signals within protein superfamilies

For the task of classifying sequences to superfamilies, the trained transformer allows highlighting the positions and signatures (tokens) that contribute most for distinguishing one superfamily from the others. We used the Captum library (Kokhlikyan *et al.* 2020, https://doi.org/10.48550/arXiv.2009.07896), which allows interpreting trained transformers regarding their decision making. To this end, the integrated gradients method (Sundararajan *et al.* 2017, http://arxiv.org/

abs/1703.01365) was used to calculate the importance of each of the input tokens (we used the default number of steps which is 50). For example, in the context of superfamily classification, how important each token is for the correct identification of a specific superfamily. Unlike standard motifs used in computational biology, here the algorithm can highlight both tokens whose inclusion in the protein sequence supports a classification to a specific superfamily and tokens whose exclusion supports the classification. To do this, we first identified the positions of tokens with high attribution scores in each sequence (absolute value above 0.2). Then, we created histograms for each family to see where these high score tokens were located. We next searched for specific amino-acids patterns by using a sliding-window approach. Moving along the sequences a window of size 15 amino acids, we searched for the presence of a specific token in at least eight out of 100 sequences within each superfamily. If so, we added a label for the token at that location. This information can be used to identify the locations of signals on protein sequences. The transformer used was the best performing one, trained on the BPE-tokenized data with 1600 vocabulary items.

2.10 Training on the BFD dataset

The BFD dataset is currently the largest public dataset of protein sequences, comprising over 2.2 billion sequences and 400 billion amino acids (Steinegger and Söding 2018). This dataset includes multiple sources of sequences that were aligned to longer sequences using MMseqs2 (Steinegger and Söding 2017, https://doi.org/10.1038/nbt.3988) and filtered based on sequence identity and number of sequences per cluster. After obtaining this dataset, we preprocessed the sequences by removing gap characters ("–"). This preprocessing phase resulted in a file of approximately 400 GB, containing pure proteins. We trained the BPE, WordPiece, and Unigram tokenizers on this dataset. Due to the large memory required for this task, we utilized the AMD EPYC 7H12 machines, with 256 cores and approximately 1 TB RAM. Due to memory and run time limitation, we trained the different tokenizers on subsamples with increasing sizes, ranging from 1000 to 10 000 000 sequences.

3 Results

3.1 Effectors and superfamilies classifications

We first evaluated the performance of the various tokenization algorithms for the task of classifying proteins to those that are effectors and those that are not (dataset 1). Figure 2a shows the performance, as measured by the MCC score, for the various tokenizers. It also shows the reduction in the length of the encoded proteins. The optimal performance, with an MCC score of 0.507, was obtained using the Unigram tokenizer with a dictionary size of 100 tokens. Compared with the default "words" tokenizer, it is both more accurate (the MCC of the "words" tokenizer was only 0.43), and it led to a 1.3-fold reduction in sequence length (i.e. the number of tokens). The highest fold reduction of 2.4 in length was obtained with the WordPiece tokenizer when using 3200 tokens, albeit with a reduction of 0.14 in the MCC score compared to the best tokenizer.

Figure 2b demonstrates the performance of applying the different tokenizers on the multiclass classification (dataset 2). The BPE tokenizer resulted in lower sequence length compared to Unigram. While the 3200 tokens Unigram dictionary led to only a 1.79-fold reduction in sequence length, BPE that has the same dictionary size, led to over 2.5 folds reduction in sequence length. The best performing transformer used the BPE tokenizer. It was trained on a dictionary containing 1600 tokens and had a very high performance, with an MCC score of 0.995. It led to a 2.2-fold reduction in the number tokens. Of note, the transformer trained with the "words" tokenizer had lower performance compared to transformers trained with alternative tokenizers. Similarly, most transformers using data-driven tokenizers outperformed the "pairs"-based dictionary.

3.2 Alignment

Next, we evaluated the impact of tokenizing nucleotide sequences on alignment accuracy and coverage (see Methods). Figure 3a shows the performance of the transformers on the alignment dataset. The accuracy of all transformers (measured by the CS) was very high (above 0.98), and the differences among the different transformers were very small (~0.007 difference in the CS between the best and the worst transformer). These high scores suggest that all transformers could reliably align the analyzed sequences.

Figure 3b shows the coverage of the transformers on the alignment dataset. The coverage is calculated as the number of valid alignments divided by the number of tested alignments (see Section 2). Large differences in coverage were observed between the worst and best transformers: the transformer using the "words" tokenizer obtained a coverage of 0.59, while the "pairs" had the highest coverage of 0.941. The best performing transformer using a data-driven tokenizer was the BPE transformer with a dictionary size of 400, resulting in a coverage of 0.924. However, this BPE tokenizer reduced the number of tokens by more than 4-fold, while the baseline "pairs" reduced the number of tokens by only 2-fold. Thus, the BPE with 400 tokens had only half as many tokens as the "pairs" baseline. As can be seen from the figure, there is a trade-off between reduction and performance. Careful examination of the BPE, WordPiece, and Unigram tokenizers shows that increasing the vocabulary size increases the coverage and reduction fold, but once the size reaches a few hundred (400, 200, 200 for BPE, WordPiece and Unigram, respectively), the coverage begins to decrease. Even dictionaries of 100 or 200 tokens have large impact on the length of the encoded sequences, slightly more than 3-fold.

3.3 Comparison of different models on additional classification and regression tasks

Figure 4a illustrates the results gained on the remote homology classification (dataset 4) (Hou *et al.* 2018, Rao *et al.* 2019). The results of the regression task of predicting the log-fluorescence of proteins (dataset 5) are demonstrated in Fig. 4b (Sarkisyan *et al.* 2016, Rao *et al.* 2019). Figure 4c shows the results of the proteins stability regression task (dataset 6) (Rocklin *et al.* 2017, Rao *et al.* 2019). The results of training the transformers on the tokenized data were compared to the previously published results of ProteinBERT (Brandes *et al.* 2022), TAPE (Rao *et al.* 2019), and LSTM (Rao *et al.* 2019) without their pretraining. Comparing the TAPE transformer with one of the transformer that used a data-driven tokenizer, specifically WordPiece with the 400 tokens, revealed that the latter was both more accurate and used less tokens in all three computational tasks (Fig. 4). Of

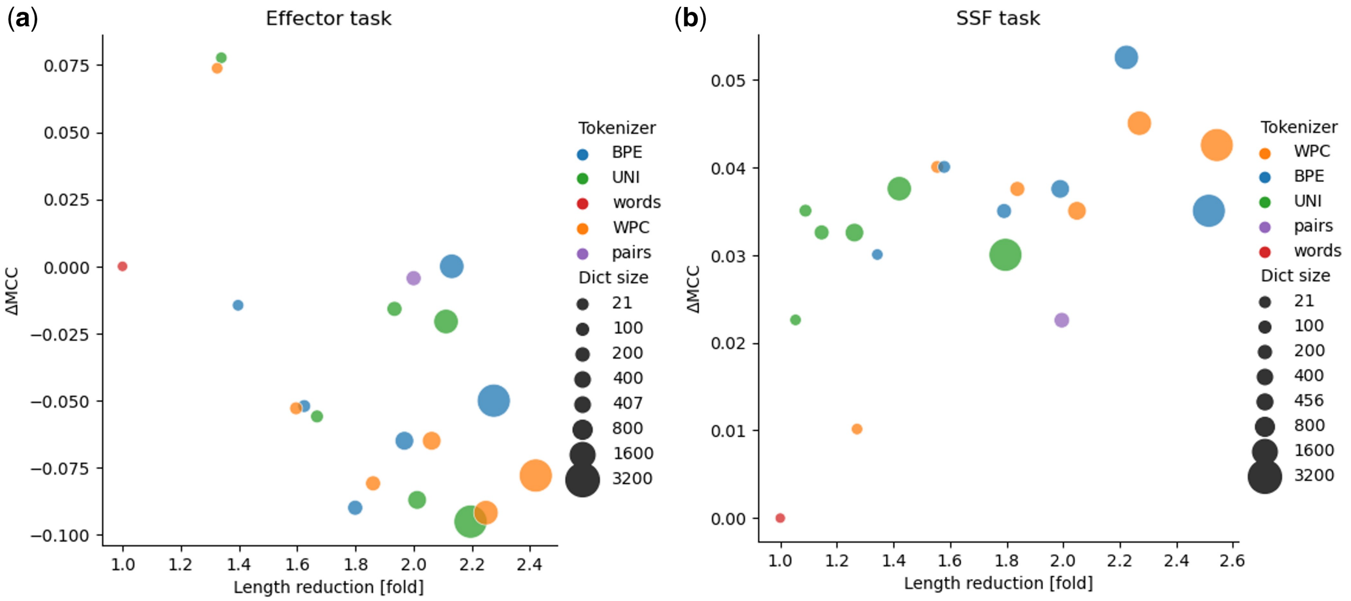


Figure 2. Panels (a) and (b) show the results of the trained transformers on the effector and superfamilies (SSF) classification tasks, respectively. A different color is assigned for each different tokenizer: BPE, WordPiece (WPC), Unigram (UNI), and the baseline tokenizers: "words" and "pairs." The dictionary size (Dict size) is demonstrated by the size of the circle. The *x*-axis indicates the fold-reduction in number of tokens used (i.e. the "length") relative to the "words" tokenizer. The *y* axis indicates the improvement in MCC score relative to the "words" tokenizer. The "word" tokenizer had MCC scores of 0.43 and 0.942 for the effector and SSF tasks, respectively. The closer the dictionary is to the right upper corner, the better it is, as it has higher length reduction and higher performance.

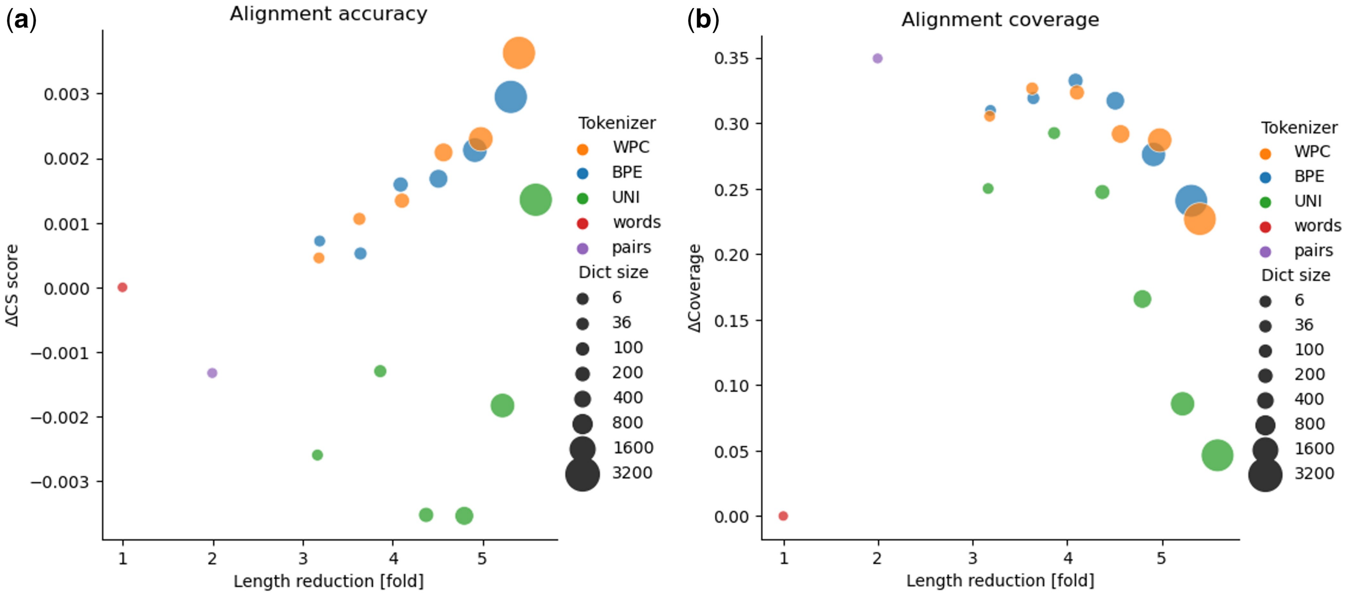


Figure 3. Results of the transformers trained on the alignment dataset preprocessed by different tokenizers: BPE, WordPiece (WPC), Unigram (UNI), and the baseline approaches: "words" and "pairs." Panels (a) and (b) report the performance as measured by the CS and the coverage, respectively. The "word" tokenizer had a CS of 0.99 (a), and a coverage of 0.59 (b). Of note, the CS is only computed on valid alignments.

note, TAPE includes seven times more free parameters than the transformer using WordPiece. Carefully examining the performance of ProteinBERT (Brandes *et al.* 2022) reveals it has the best performance on the fluorescence (Fig. 4b) and stability (Fig. 4c) tasks and the lowest performance on the remote homology task. Of note, while the ProteinBERT, LSTM and TAPE have 16 million (M), 38M, and 38M parameters, respectively, the remaining transformers studied have only 5M free parameters. One of the main advantages of using the optimized tokenizers was demonstrated in the fluorescence task (Fig. 4b), where using dictionaries with a small number of tokens (such as 3200 for BPE and WordPiece) was able to significantly reduce the length of the input sequences (by up to 20-fold) compared to the trivial character-based ("words") tokenizers used in ProteinBERT, LSTM, and TAPE.

We compared the various tokenization techniques on two additional tasks, previously analyzed in the ProteinBERT study (Brandes *et al.* 2022): fold prediction and neuropeptide classification (Fig. 5). For the fold prediction task (Fig. 5a), we observe a trade-off between the performance and the dictionary size. Of note, the transformer with the Unigram tokenizer (with 100 tokens) was both more accurate and

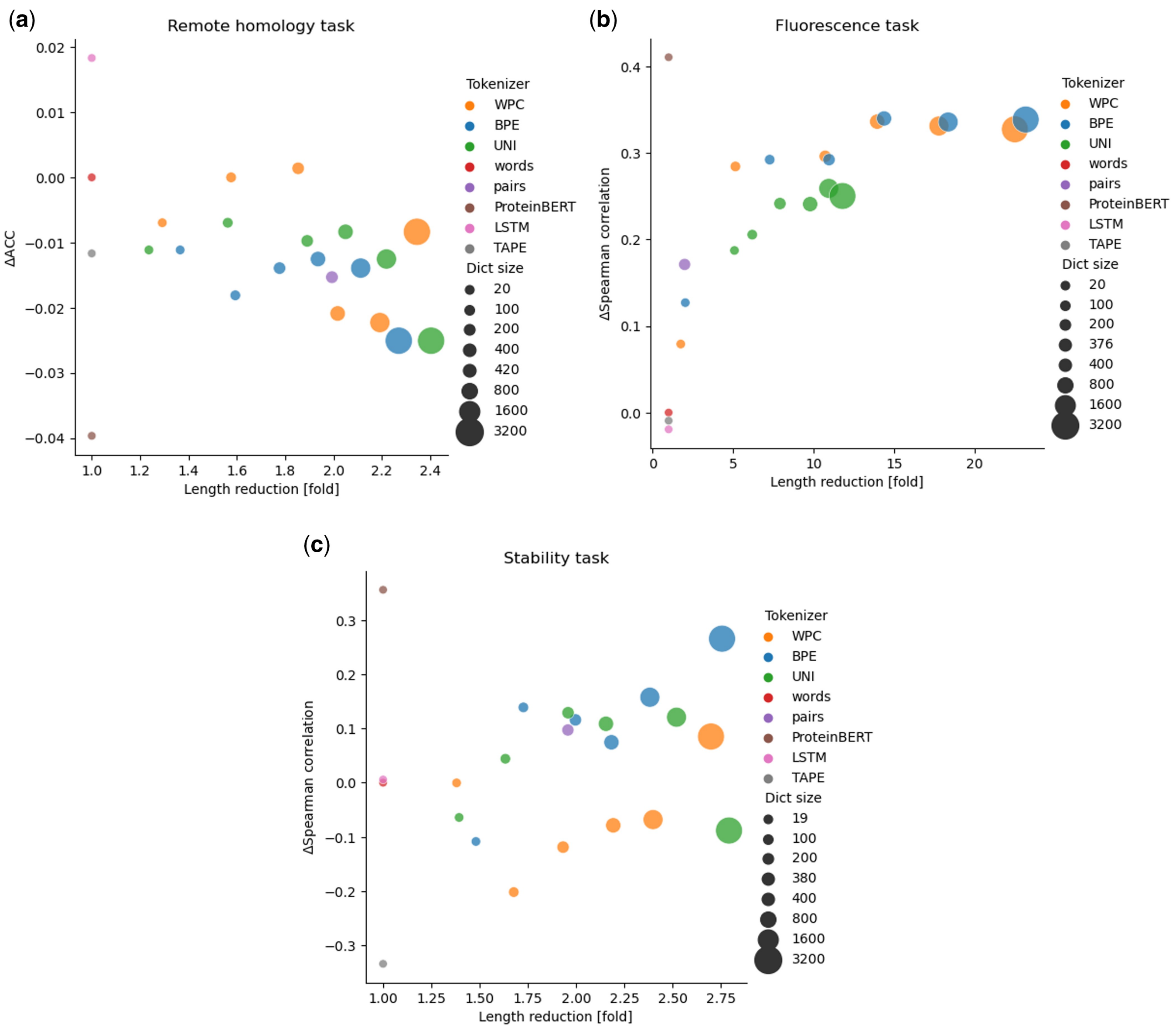


Figure 4. Evaluating the performance of the different tokenizers and comparing them to the previously tested models: ProteinBERT, LSTM, and TAPE. The *x*-axis is the length reduction, and the y-axis is the performance of the transformer trained on the same dataset. Panels (a), (b), and (c) display the results of datasets 4, 5, and 6, respectively. The "words" tokenizer had an accuracy (ACC) score of 0.101, spearman correlation scores of 0.23 and 0.274 for the remote homology task, fluorescence task and the stability task, respectively.

used less tokens than both "words" and ProteinBERT. For the neuropeptide prediction task, the ProteinBERT transformer performed worse (Fig. 5b) than all other transformers. A single transformer (the WordPiece with 3200 tokens) had the second-best performance and the highest length reduction compared to all other methods.

Our results suggest that for some datasets a tradeoff between accuracy and fold reduction exists, while for some tasks, using data-driven tokenizers can be beneficial in both aspects (accuracy and length reduction). In addition, our results suggest that the ProteinBERT architecture may be more suited for regression tasks, than to other tasks such as classification.

3.4 Hyperparameter optimization

We optimized four hyperparameters: the number of layers, the number of attention heads, the hidden vector size, and the learning rate. The importance of each hyperparameter and its influence on performance was tested on the superfamily classification task (dataset 2). Our results suggest that not optimizing a hyperparameter only marginally affects performance and the order of performance between the various tokenizers (Fig. 6). The "words" tokenizer consistently yielded the lowest accuracy in eleven out of twelve tests, and in the remaining case [Panel (a), layers = 2], it ranked second lowest. Conversely, three transformers consistently outperformed the others across all twelve tests. Specifically, transformers trained with BPE tokenizers with dictionary sizes of 1600 and 3200, along with the WPC tokenizer with a dictionary size of 3200, consistently secured top-three positions in nine tests.

3.5 Quantifying the difference in performance of tokenizers compared to the default

We quantified performance differences between the different tokenizers compared to the default, i.e. the "words"

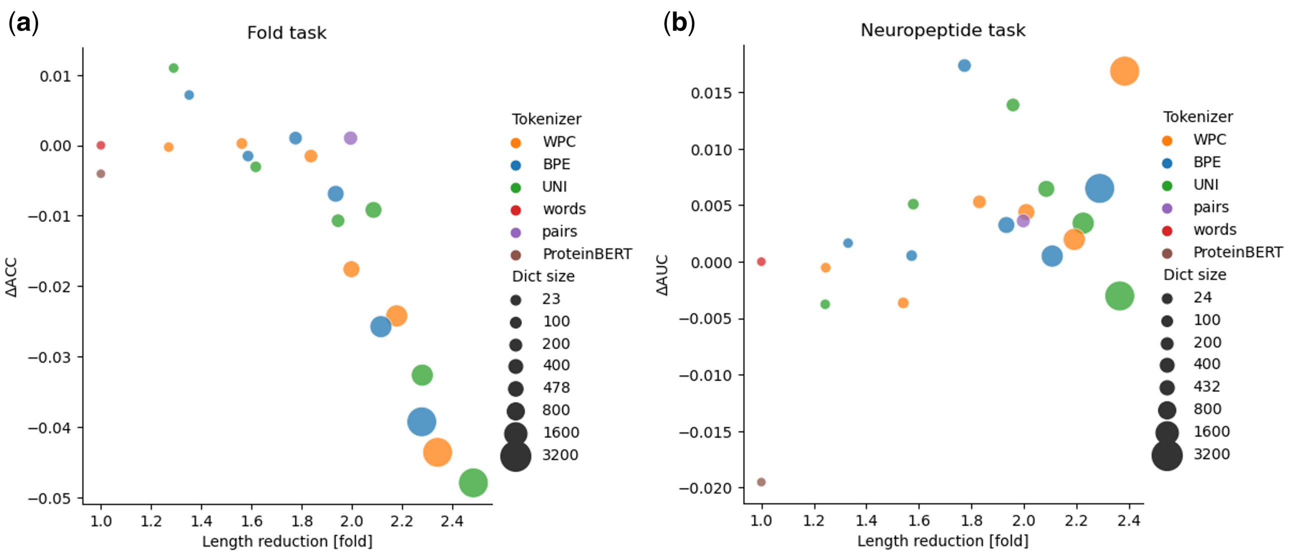


Figure 5. We evaluated the tokenization on two datasets proposed in ProteinBERT: (a) fold structure classification and (b) neuropeptide identification. The *x*-axis and the *y*-axis refer to the length reduction and the performance, respectively. The "words" tokenizer had an ACC score of 0.59, and an AUC score of 0.96 for the fold task, neuropeptide task, respectively.

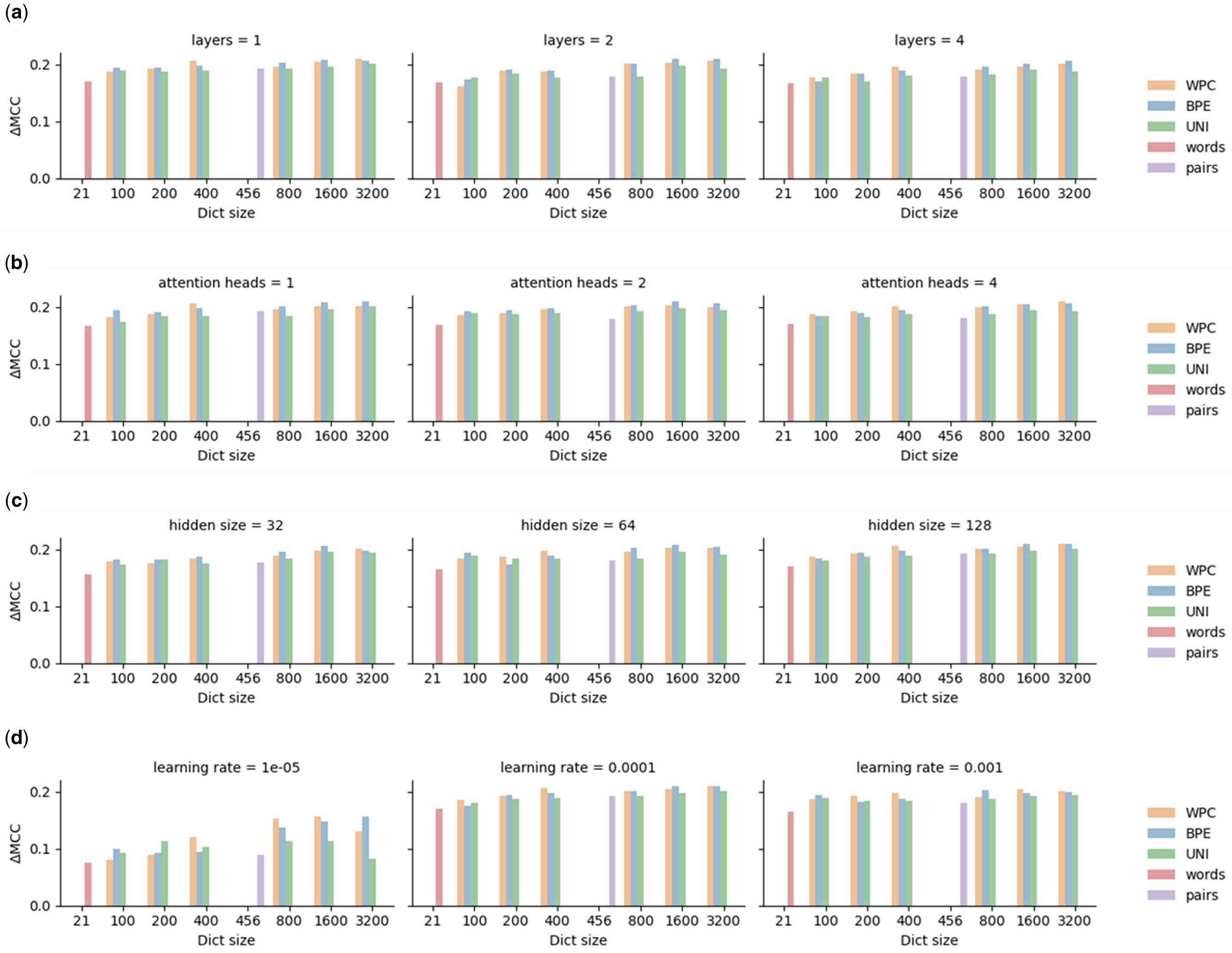


Figure 6. Evaluating the effect of hyperparameters optimization on the different tokenizer approaches. Panels (a), (b), (c), and (d) correspond to not optimizing one hyperparameter: the number of layers, the number of attention heads, the hidden vector size, and the learning rate, respectively. Within each panel, each subgraph depicts the effects of fixing the hyperparameter to a different value, while optimizing the other three. The color-coded bars in the panels represent distinct tokenizer methodologies, the *y*-axis is the delta MCC from the lowest-performing transformer, which achieved a score of 0.775, and the *x*-axis is the dictionary size.

tokenizer. Consider for example the UNI tokenizer. For dataset 1, we have performance with different dictionary sizes. We selected the dictionary size that resulted in the best performance. We repeated this procedure for all other datasets. Thus, we have nine performance values for UNI (eight different datasets, but for dataset 3, we have two performance values, one for coverage and one for alignment accuracy). We next compared the 9-tuple performance vector of UNI against the 9-tuple performance vector of "words" using the Wilcoxon test (Wilcoxon 1945, https://doi.org/10.2307/3001968). The results suggest that UNI significantly outperforms "words" ($P = .011$). Similar significant results were obtained for BPE and WPC (P = .035, .004, respectively). The "pairs" tokenizer was not significantly better than "words" ($P = .2$). These results highlight the benefit of using data-driven tokenizers.

3.6 Identification of contributing signals and their visualization

One of the key disadvantages of using transformers is that they are highly nonlinear and difficult to interpret. Biological sequences contain signals in specific locations that are important for determining their structure and function. Identifying these signals remains a challenge. Better understanding of these signals should result in a better mapping between a protein sequence and its structure and function, thus contributing to protein function prediction, classification, and design of novel proteins. By training a transformer to predict specific classes, we could apply interpretation tools to identify those signals. Figure 7 displays the resulting interpretation of each superfamily, specifically enrichment and depletion of specific tokens as a function of the protein length. For example, a "PKK" at the N terminus of a protein suggests it belongs to the superfamily SSF101152. One of the key differences from the signatures that appear in Pfam (Mistry *et al.* 2021) is that here we do not rely on a multiple sequence alignment, thus accounting for variability in the position of specific tokens along the length of the protein. In addition, dependencies among tokens are accounted for. Finally, depleted tokens can be identified, e.g. the presence of the token "ER" at the N terminus of proteins suggests it is not SSF100879.

3.7 Tokenizing the BFD dataset

In our previous experiments, we showed the effect of tokenizing the input on specific tasks, i.e. for each task we trained data-specific tokenizers. Here, we aimed to train tokenizers on a very large dataset, which may be important in cases where there are not enough sequences for a specific task, or as input for the next generation of biological pre-trained models. To this end, we trained BPE, WordPiece, and Unigram tokenizers on samples of proteins from the 2.2 billion protein sequences of the BFD dataset (Steinegger and Söding 2018). We evaluate the average sequences length as a function of the vocabulary size and number of sequences in the training data (Fig. 8). Increasing the size of the vocabulary resulted in a sharp decrease in the average numbers of tokens per protein, thus enabling processing longer biological sequences with the similar memory requirement. In addition, the increase of the training data resulted in smaller values of tokens per protein. The average number of tokens per protein converged after training on 1 000 000 samples. When using the largest dictionaries (51 200 tokens), the BPE, WordPiece, and Unigram reduced the average length by 15.6%, 16.8%, and 14.9%, respectively. Among all tokenizers, Unigram was most influenced by the training and vocabulary sizes: it has the highest number of tokens per proteins (182.4) when using small training size (1000 sequences) and 100 tokens in the vocabulary. Yet, when the training data was 10^7 sequences and the dictionary size higher than 50 000 tokens, it obtained the best average length (53.1 tokens for protein).

3.8 Comparing specific versus general data-driven tokenizers

Above we trained two types of data-driven tokenizers. The first type, which we term "specific," was trained on small datasets (e.g. effectors), while the second type, which we term "general" was trained on very large number of protein sequences, not related to a specific computational task. We next aimed to determine whether it is beneficial, for specific tasks, to use general versus specific data-driven tokenizers. To this end, we compared the performance between the specific and the general versions of the trained transformers BPE, Unigram, and WordPiece on seven computational tasks (the alignment task is based on nucleotide rather than protein sequences and was hence not included here). For all tasks, the specific type outperformed the general type, and the results were statistically significant for six out of the seven tasks (Fig. 9). For some tasks such as the remote homology task, using the general tokenizers resulted in very poor performance compared to the specific tokenizers. We hypothesize that tasks involving proteins with similar domains would be more affected by using task-specific tokenizers, while tasks encompassing proteins across the tree of life would likely show similar results to those using the BFD-trained tokenizers.

4 Discussion

Our results clearly indicate that data-driven tokenization of biological datasets can improve performance. This was demonstrated for all tested datasets and for all types of analyses (classification, regression, and sequence-to-sequence). However, no single tokenization method was optimal for all datasets, emphasizing the need to evaluate different tokenizers for each data and learning task.

K-mers were extensively used in bioinformatics and related machine-learning applications (Alam and Chowdhury 2020, https://doi.org/10.1371/journal.pone.0239381, ValizadehAslani *et al.* 2020, https://doi.org/10.3390/biology9110365, Orozco-Arias *et al.* 2021, https://doi.org/10.7717/peerj.11456). A biological sequence can be represented as a vector, in which each entry counts the number of occurrences of a specific *K*-mer (and the size of that vector corresponds to the total number of possible *K*-mers). Such a vector can be considered as a set of features that embeds the sequence. As the size of the vector may be large, when *K* is high, feature selection is usually applied to maintain only informative *K*-mers, e.g. Orozco-Arias *et al.* (2021). Representing a sequence as a vector of *K*-mer frequencies is inherently different from the process of embedding as used in current NLP research. In NLP-based embedding, additional information regarding both the position and the context of the different tokens is stored (Dufter *et al.* 2022, https://doi.org/10.1162/coli_a_00445). As shown in our work, NLP-based embedding can be accomplished using different tokenization methods, and one method to tokenize biological sequence is to use a dictionary comprising all *K*-mers of a fixed size (the "words" and "pairs" tokenization methods). It is thus important to

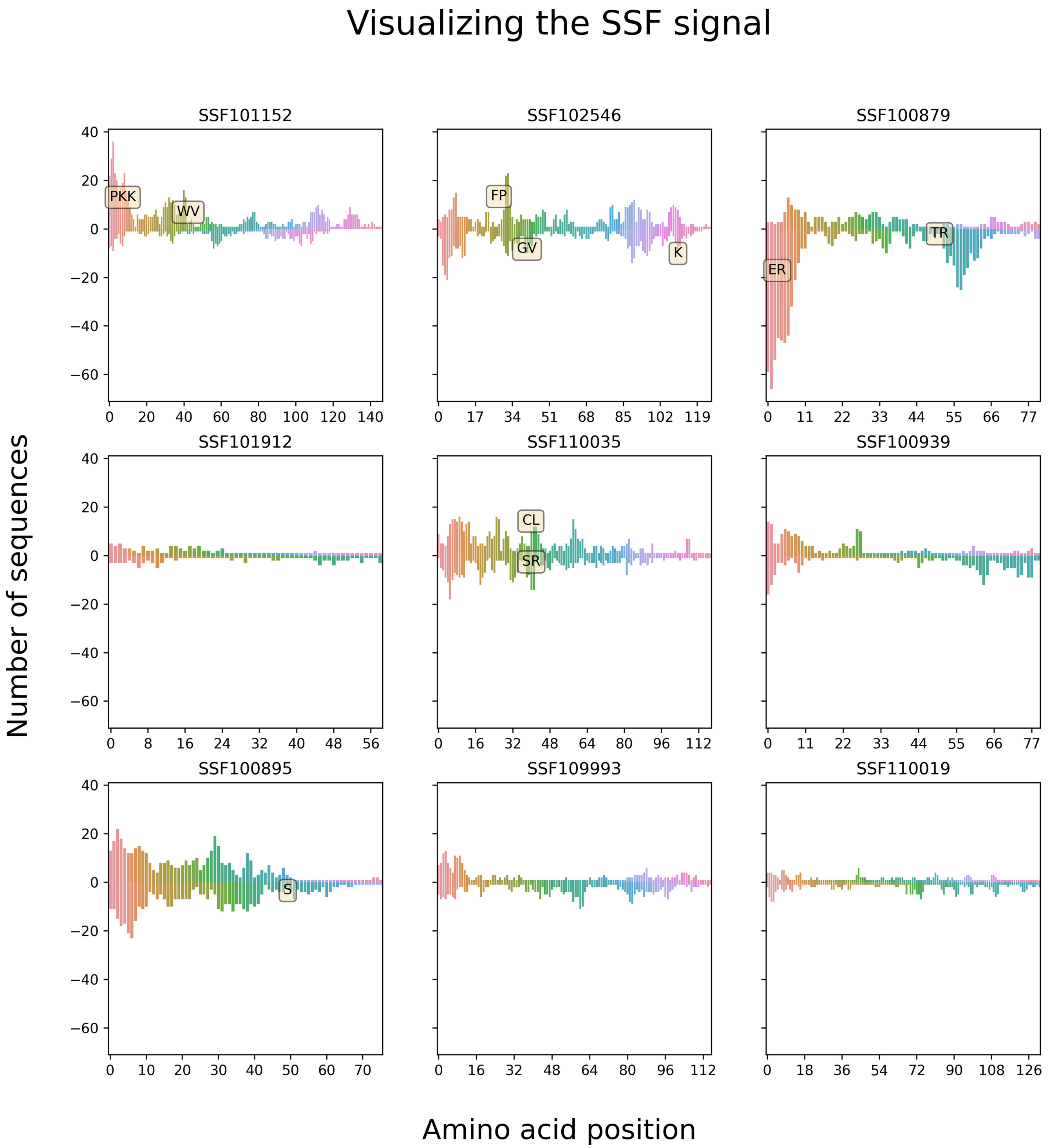


Figure 7. Visualizing the important features of each of the superfamilies. Each of the nine graphs correspond to a different family. In each graph, the *x*-axis is the amino-acid position, and the *y*-axis the number of sequences that had an important feature in this position. In addition, we applied distinct colors to the *x*-axis corresponding to the ascending values. We added labels with specific tokens if they are repeated in several different sequences. The graphs were created by 100 test protein sequences for each superfamily.

distinguish *K*-mer based feature representation of a biological sequence from *K*-mer based NLP-style embedding. Of note, our results show that using a fixed size *K*-mer for NLP-style embeddings is inferior to using data-driven tokenizers.

Transformers often cannot analyze sequences above a specific threshold length. It is common to segment longer sequences to subsequences shorter than this threshold, thus bypassing this restriction. However, this fragmentation prohibits the model from analyzing the entire input data, and can thus potentially decrease performance. Our results show that fragmentation can sometimes be avoided by tokenizing the data, i.e. tokenization allows architectures to expend their

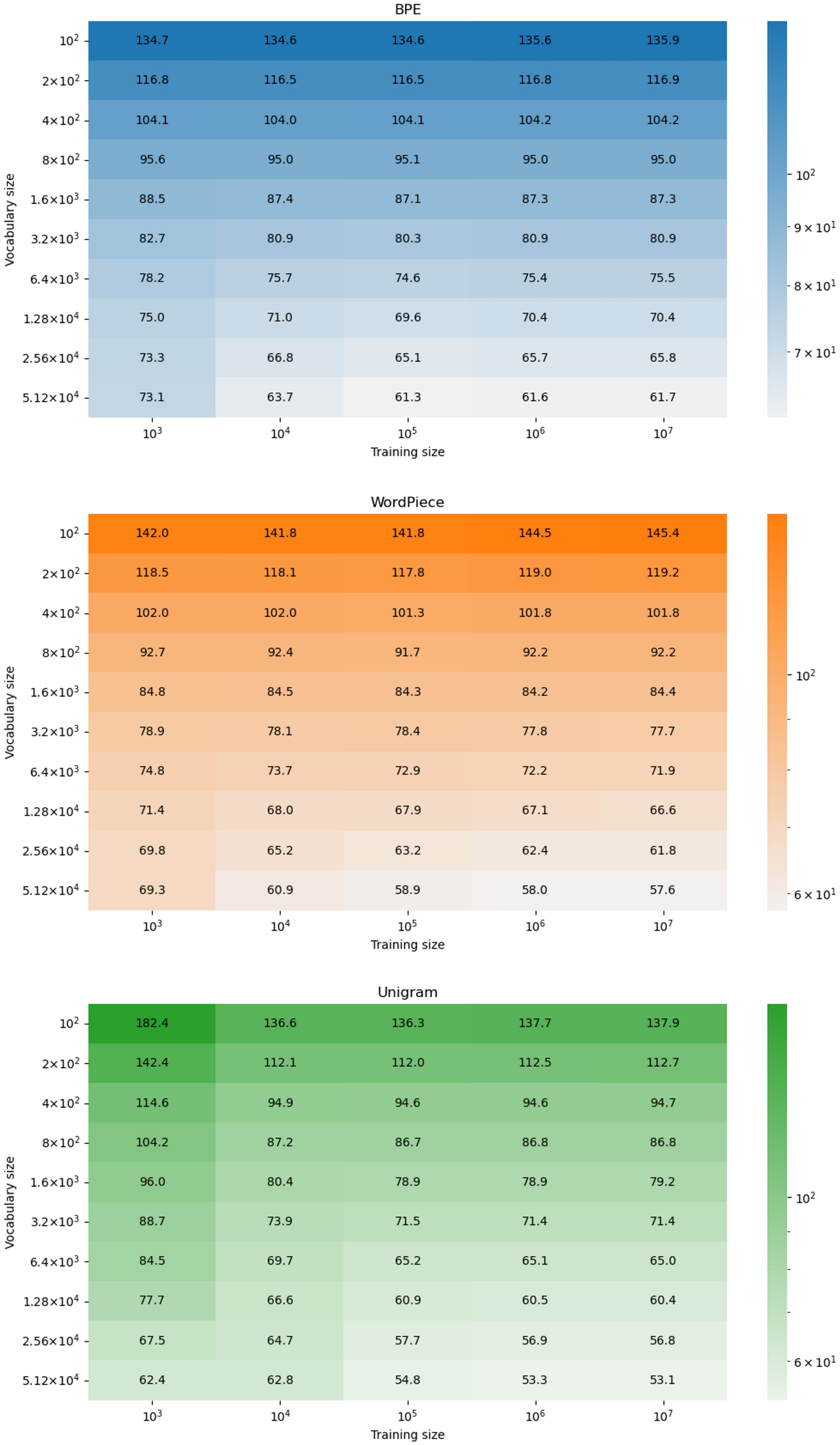
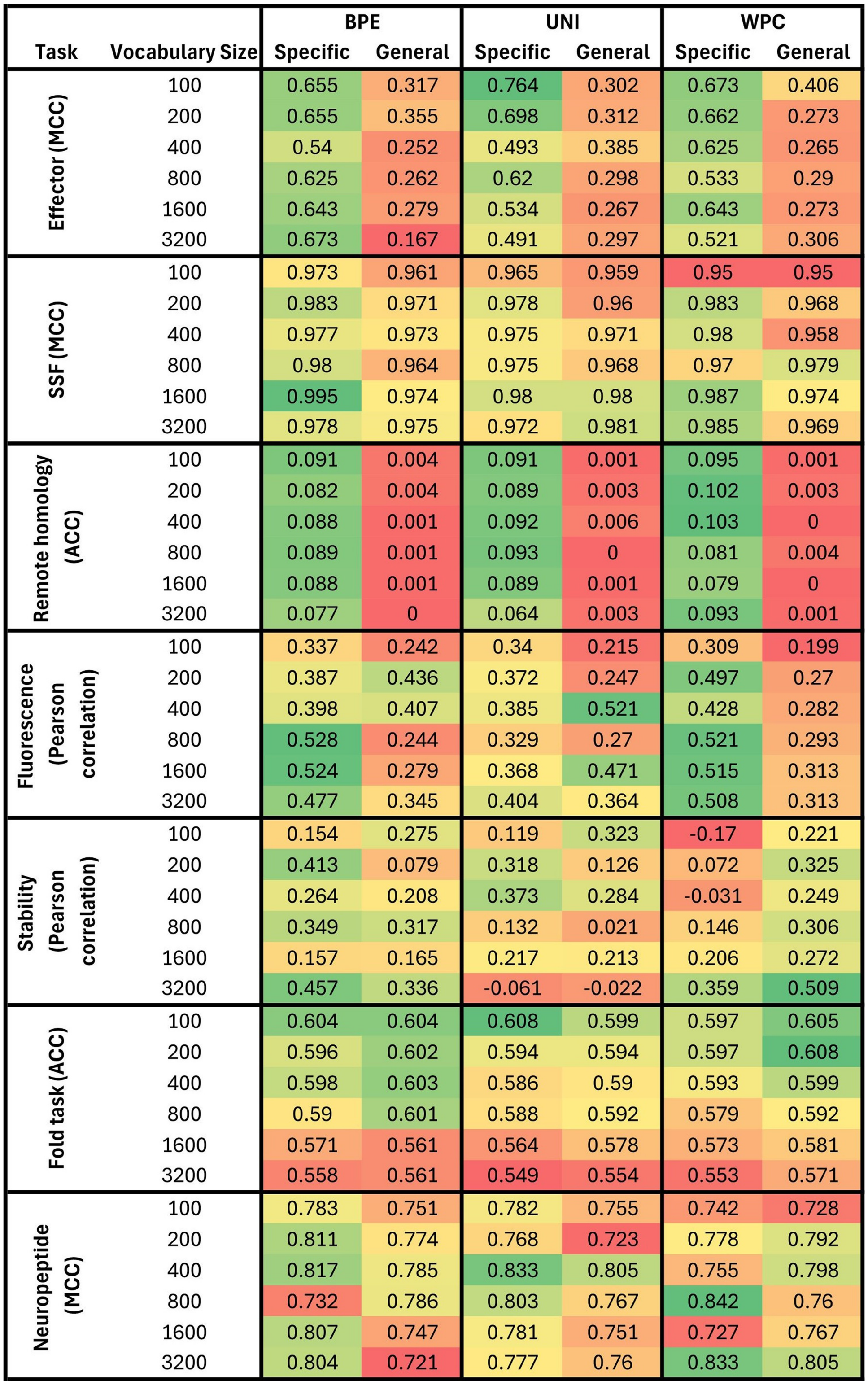


Figure 8. Effect of vocabulary size and number of training samples on the three tokenizers: BPE, WordPiece, and Unigram. The darker the color the higher the average number of tokens per protein. Increasing the vocabulary and the training size reduces the number of tokens per protein for all of the tested tokenizers.



		BPE		UNI		WPC	
Task	Vocabulary Size	Specific	General	Specific	General	Specific	General
Effector (MCC)	100	0.655	0.317	0.764	0.302	0.673	0.406
	200	0.655	0.355	0.698	0.312	0.662	0.273
	400	0.54	0.252	0.493	0.385	0.625	0.265
	800	0.625	0.262	0.62	0.298	0.533	0.29
	1600	0.643	0.279	0.534	0.267	0.643	0.273
	3200	0.673	0.167	0.491	0.297	0.521	0.306
SSF (MCC)	100	0.973	0.961	0.965	0.959	0.95	0.95
	200	0.983	0.971	0.978	0.96	0.983	0.968
	400	0.977	0.973	0.975	0.971	0.98	0.958
	800	0.98	0.964	0.975	0.968	0.97	0.979
	1600	0.995	0.974	0.98	0.98	0.987	0.974
	3200	0.978	0.975	0.972	0.981	0.985	0.969
Remote homology (ACC)	100	0.091	0.004	0.091	0.001	0.095	0.001
	200	0.082	0.004	0.089	0.003	0.102	0.003
	400	0.088	0.001	0.092	0.006	0.103	0
	800	0.089	0.001	0.093	0	0.081	0.004
	1600	0.088	0.001	0.089	0.001	0.079	0
	3200	0.077	0	0.064	0.003	0.093	0.001
Fluorescence (Pearson correlation)	100	0.337	0.242	0.34	0.215	0.309	0.199
	200	0.387	0.436	0.372	0.247	0.497	0.27
	400	0.398	0.407	0.385	0.521	0.428	0.282
	800	0.528	0.244	0.329	0.27	0.521	0.293
	1600	0.524	0.279	0.368	0.471	0.515	0.313
	3200	0.477	0.345	0.404	0.364	0.508	0.313
Stability (Pearson correlation)	100	0.154	0.275	0.119	0.323	-0.17	0.221
	200	0.413	0.079	0.318	0.126	0.072	0.325
	400	0.264	0.208	0.373	0.284	-0.031	0.249
	800	0.349	0.317	0.132	0.021	0.146	0.306
	1600	0.157	0.165	0.217	0.213	0.206	0.272
	3200	0.457	0.336	-0.061	-0.022	0.359	0.509
Fold task (ACC)	100	0.604	0.604	0.608	0.599	0.597	0.605
	200	0.596	0.602	0.594	0.594	0.597	0.608
	400	0.598	0.603	0.586	0.59	0.593	0.599
	800	0.59	0.601	0.588	0.592	0.579	0.592
	1600	0.571	0.561	0.564	0.578	0.573	0.581
	3200	0.558	0.561	0.549	0.554	0.553	0.571
Neuropeptide (MCC)	100	0.783	0.751	0.782	0.755	0.742	0.728
	200	0.811	0.774	0.768	0.723	0.778	0.792
	400	0.817	0.785	0.833	0.805	0.755	0.798
	800	0.732	0.786	0.803	0.767	0.842	0.76
	1600	0.807	0.747	0.781	0.751	0.727	0.767
	3200	0.804	0.721	0.777	0.76	0.833	0.805

Figure 9. Comparing the performance of transformer trained on data encoded by specific trained tokenizers ("Specific") and tokenizers trained on the BFD dataset ("General"). The evaluation was conducted on seven datasets, utilizing three tokenizer types: BPE, Unigram, and WordPiece. For each tokenizer, multiple vocabulary sizes were tested: 100, 200, 400, 800, 1600, and 3200. Each task is individually colored to facilitate comparison. To quantify the differences between the general and specific tokenizers, we performed paired *t*-tests and obtained the following *P*-values: 8.23×10^{-11}, 0.001, 5.07×10^{-18}, 0.0016, 0.356, 0.0004, 0.025, for datasets 1, 2, 4, 5, 6, 7, and 8, respectively. Of note, in this comparison, we only tested a single configuration of learning rate, with a value of 0.0001.

capacity to substantially longer proteins and DNA sequences, as was recently shown in DNABERT-2 (Zhou *et al.* 2023, https://doi.org/10.48550/arXiv.2306.15006).

In this study, we also demonstrated that important biological information can be extracted for post-analysis of trained models applied to specific learning tasks. For example, we could detect specific signatures for protein superfamily classification. One of the benefits of the proposed approach compared to motifs in the form of profile hidden Markov models is that it does not rely on a multiple sequence alignment, which may be unreliable, especially when highly diverged sequences are analyzed.

In this study, we examined the impact of tokenization at the molecular level. We hypothesize that tokenization has the potential to be applied to various forms of discrete biological data, such as genes (Miller *et al.* 2022). In addition, the incorporation of character compression into classical algorithms used in biology, such as Blast [Altschul *et al.* 1990, https://doi.org/10.1016/S0022-2836(05)80360-2] and Kraken-2 (Wood *et al.* 2019, https://doi.org/10.1186/s13059-019-1891-0), should be considered in order to decrease running times.

This work represents the initial phase of studying how tokenization impacts biological language models. Our study demonstrates that data-driven tokenizers should be considered, both for accuracy and for length reduction. Our work also shows that there is no single data-driven tokenizer that outperformed all the others. We demonstrate that the effect of tokenizing the sequence depends on the specific task, the data type and size, and the tokenization algorithm applied. In future work, it would be interesting to compare Large Biological Models (LBMs) performance which were pretrained with various tokenization algorithms, i.e. we speculate that in the future there will be several alternative LBMs, each pretrained with a different tokenization algorithm, and users can test which LBM is best suited to their computational task. Our study further suggests that future studies comparing the performance of new emerging transformer architectures on biological data, should include different tokenizers as a critical component in their evaluation.

Conflict of interest

None declared.

Funding

This work was supported by an Azrieli Foundation Early Career Faculty Fellowship to Y.B. This work was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel Aviv University to E.D. This work was also supported by the Israel Science Foundation [2818/21 to T.P., 448/20 to Y.B.].

Data availability

Code, data, and trained tokenizers are available on https://github.com/technion-cs-nlp/BiologicalTokenizers.

References

Alam MNU, Chowdhury UF. Short K-Mer abundance profiles yield robust machine learning features and accurate classifiers for RNA viruses. *PLoS One* 2020;**15**:e0239381.

Alharbi WS, Rashid M. A review of deep learning applications in human genomics using next-generation sequencing data. *Hum Genomics* 2022;**16**:26.

Altschul SF, Gish W, Miller W *et al.* Basic local alignment search tool. *J Mol Biol* 1990;**215**:403–10.

Andreeva A, Kulesha E, Gough J *et al.* The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic Acids Res* 2020; **48**:D376–82.

Brandes N, Ofer D, Peleg Y *et al.* ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics* 2022;**38**:2102–10.

Burbach J, Peter H. Neuropeptides from concept to online database www.neuropeptides.nl. *Eur J Pharmacol* 2010;**626**:27–48.

Devlin J, Chang M-W, Lee K *et al.* BERT: pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, MN: Association for Computational Linguistics, 2019, 4171–86.

Dotan E, Belinkov Y, Avram O *et al.* Multiple sequence alignment as a sequence-to-sequence learning problem. In: *The Eleventh International Conference on Learning Representations (ICLR), Kigali, Rwanda*. 2023.

Dufter P, Schmitt M, Schütze H. Position information in transformers: an overview. *Comput Linguist* 2022;**48**:733–63.

Eraslan G, Avsec Ž, Gagneur J *et al.* Deep learning: new computational modelling techniques for genomics. *Nat Rev Genet* 2019; **20**:389–403.

Gage P. A New Algorithm for Data Compression. *C Users J* 1994;**12** (2):23–38.

Hou J, Adhikari B, Cheng J. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics* 2018; **34**:1295–303.

Ji Y, Zhou Z, Liu H *et al.* DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. *Bioinformatics* 2021;**37**:2112–20.

Jumper J, Evans R, Pritzel A *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* 2021;**596**:583–9.

Kokhlikyan N, Miglani V, Martin M *et al.* Captum: a unified and generic model interpretability library for PyTorch. arXiv, arXiv:2009.07896v1, 2020, preprint: not peer reviewed.

Koumakis L. Deep learning models in genomics; are we there yet? *Comput Struct Biotechnol J* 2020;**18**:1466–73.

Kudo T. Subword regularization: improving neural network translation models with multiple subword candidates. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia*. Association for Computational Linguistics, 66–75. 2018.

Kulmanov M, Khan MA, Hoehndorf R *et al.* DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 2018;**34**:660–8.

LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature* 2015; **521**:436–44.

Lin T, Wang Y, Liu X *et al.* A survey of transformers. *AI Open* 2022; **3**:111–132.

List J-M, Sylvestre Pathmanathan J, Lopez P *et al.* Unity and disunity in evolutionary sciences: process-based analogies open common research avenues for biology and linguistics. *Biol Direct* 2016;**11**:39.

Loewenthal G, Rapoport D, Avram O *et al.* A probabilistic model for indel evolution: differentiating insertions from deletions. *Mol Biol Evol* 2021;**38**:5769–81.

Markowitz VM, Chen I-MA, Palaniappan K *et al.* IMG: the integrated microbial genomes database and comparative analysis system. *Nucleic Acids Res* 2012;**40**:D115–22.

Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim Biophys Acta* 1975; **405**:442–51.

Miller D, Stern A, Burstein D. Deciphering microbial gene function using natural language processing. *Nat Commun* 2022;**13**:5731.

Mistry J, Chuguransky S, Williams L *et al.* Pfam: the protein families database in 2021. *Nucleic Acids Res* 2021;**49**:D412–9.

Notti RQ, Stebbins CE. The structure and function of type III secretion systems. *Microbiol Spectr* 2016;**4**.

Nurk S, Koren S, Rhie A *et al.* The complete sequence of a human genome. *Science* 2022;**376**:44–53.

Ofer D, Linial M. NeuroP ID: a predictor for identifying neuropeptide precursors from metazoan proteomes. *Bioinformatics* 2014; **30**:931–940.

Orozco-Arias S, Candamil-Cortés MS, Jaimes PA *et al.* K-Mer-based machine learning method to classify LTR-retrotransposons in plant genomes. *PeerJ* 2021;**9**:e11456.

Ott M, Edunov S, Baevski A *et al.* fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), Minneapolis, Minnesota*, 2019, 48–53.

Penn O, Privman E, Landan G *et al.* An alignment confidence score capturing robustness to guide tree uncertainty. *Mol Biol Evol* 2010; **27**:1759–67.

Radford A, Narasimhan K, Salimans T *et al.* Improving language understanding by generative pre-training. OpenAI Research Papers. 2018.

Rao R, Bhattacharya N, Thomas N *et al.* Evaluating protein transfer learning with TAPE. *Adv Neural Inf Process Syst* 2019;**32**:9689–701.

Rao RM, Liu J, Verkuil R *et al.* MSA transformer. In: *Proceedings of the 38th International Conference on Machine Learning, virtual event*. PMLR. 2021, 8844–56.

Richard G-F, Kerrest A, Dujon B. Comparative genomics and molecular dynamics of DNA repeats in eukaryotes. *Microbiol Mol Biol Rev* 2008;**72**:686–727.

Rocklin GJ, Chidyausiku TM, Goreshnik I *et al.* Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science* 2017;**357**:168–75.

Rudas A, Chiang JN, Corradetti G *et al.* Automated large-scale prediction of exudative AMD progression using Machine-Read OCT biomarkers. *PLOS Digit Health* 2023;**2**:e0000106.

Sarkisyan KS, Bolotin DA, Meer MV *et al.* Local fitness landscape of the green fluorescent protein. *Nature* 2016;**533**:397–401.

Schuster M, Nakajima K. Japanese and Korean voice search. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan*. 2012, 5149–52.

Sennrich R, Haddow B, Birch A. Neural machine translation of rare words with subword units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics. 2016, 1715–25.

Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol* 2017;**35**:1026–8.

Steinegger M, Söding J. Clustering huge protein sequence sets in linear time. *Nat Commun* 2018;**9**:2542.

Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. In: *Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney, Australia*. 2017, 3319–28.

Talukder A, Barham C, Li X *et al.* Interpretation of deep learning in genomics and epigenomics. *Brief Bioinform* 2021;**22**:bbaa177.

ValizadehAslani T, Zhao Z, Sokhansanj BA *et al.* Amino acid K-Mer feature extraction for quantitative antimicrobial resistance (AMR) prediction by machine learning and model interpretation for biological insights. *Biology (Basel)* 2020;**9**:365.

Van Noorden R, Maher B, Nuzzo R. The top 100 papers. *Nature* 2014; **514**:550–3.

Vaswani A, Shazeer N, Parmar N *et al.* Attention is all you need. In: *31st Conference on Neural Information Processing Systems (NIPS), Long Beach, California, USA*. 2017.

Voulodimos A, Doulamis N, Doulamis A *et al.* Deep learning for computer vision: a brief review. *Comput Intell Neurosci* 2018; **2018**:7068349.

Wagner N, Alburquerque M, Ecker N *et al.* Natural language processing approach to model the secretion signal of type III effectors. *Front Plant Sci* 2022;**13**:1024405.

Wilcoxon F. Individual comparisons by ranking methods. *Biometrics Bull* 1945;**1**:80–3.

Wolf T, Debut L, Sanh V *et al.* Transformers: state-of-the-art natural language processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics. 2020, 38–45.

Wood DE, Lu J, Langmead B. Improved metagenomic analysis with kraken 2. *Genome Biol* 2019;**20**:257.

Yu L, Kumar Tanwar D, Penha EDS *et al.* Grammar of protein domain architectures. *Proc Natl Acad Sci USA* 2019;**116**:3636–45.

Zhou Z, Ji Y, Li W *et al.* DNABERT-2: efficient foundation model and benchmark for multi-species genome. In: *The Twelfth International Conference on Learning Representations (ICLR), Vienna Austria*. 2023.

Bioinformatics, 2024, 40, 1–15
https://doi.org/10.1093/bioinformatics/btae196
Original Paper