METHODOLOGY

genomicBERT and data-free deep-learning model evaluation

Tyrone Chen[1], Navya Tyagi[2,†], Sarthak Chauhan[3,†], Anton Y. Peleg[1] and Sonika Tyagi[4*]

*Correspondence:
sonika.tyagi@rmit.edu.au
[4]School of Computing Technologies, Royal Melbourne Institute of Technology, 3000, Melbourne, Australia
Full list of author information is available at the end of the article
†Equal contributor

Abstract

The emerging field of Genome-NLP (Natural Language Processing) aims to analyse biological sequence data using machine learning (ML), offering significant advancements in data-driven diagnostics. Three key challenges exist in Genome-NLP. First, long biomolecular sequences require "tokenisation" into smaller subunits, which is non-trivial since many biological "words" remain unknown. Second, ML methods are highly nuanced, reducing interoperability and usability. Third, comparing models and reproducing results are difficult due to the large volume and poor quality of biological data.

To tackle these challenges, we developed the first automated Genome-NLP workflow that integrates feature engineering and ML techniques. The workflow is designed to be species and sequence agnostic. In this workflow: a) We introduce a new transformer-based model for genomes called *genomicBERT*, which empirically tokenises sequences while retaining biological context. This approach minimises manual preprocessing, reduces vocabulary sizes, and effectively handles out-of-vocabulary "words". (b) We enable the comparison of ML model performance even in the absence of raw data.

To facilitate widespread adoption and collaboration, we have made *genomicBERT* available as part of the publicly accessible conda package called *Ziran*. We have successfully demonstrated the application of *Ziran* on multiple case studies, showcasing its effectiveness in the field of Genome-NLP.

Keywords: BERT; Deep Learning; Genomics; Genome-NLP; Machine Learning; Natural Language Processing; Random Forest; Tokenisation; XGBoost

Highlights

- We provide a comprehensive classification of genomic data tokenisation and representation approaches for ML applications along with their pros and cons.
- We infer k-mers directly from the data and handle out-of-vocabulary words. At the same time, we achieve a significantly reduced vocabulary size compared to the conventional k-mer approach reducing the computational complexity drastically.
- Our method is agnostic to species or biomolecule type as it is data-driven.
- We enable comparison of trained model performance without requiring original input data, metadata or hyperparameter settings.
- We present the first publicly available, high-level toolkit that infers the grammar of genomic data directly through artificial neural networks.
- Preprocessing, hyperparameter sweeps, cross validations, metrics and interactive visualisations are automated but can be adjusted by the user as needed.

Introduction

Biological sequence data contains abundant low-level information, but is often transformed into other data types as a part of analysis pipelines, with varying degrees of information loss. In biology, such analyses can involve predicting protein structure [1], analysing images [2], identifying genomic and epigenetic features [3], and functionally annotating sequences [4] [Figure 3]. In our study, we focus on the information richness in raw sequence data, and exploit this property to classify biological sequence collections with concepts borrowed from the field of Natural Language Processing (NLP), with the potential future application of a ChatGPT [5] or bioGPT-like [6] tool for users [Figure 1]. The other potential use case is to generate data-driven features from sequence data that can be combined with our categorical or numeric data for various data integration tasks involving ML modelling [7].

Decomposing a sequence into a series of numeric elements for machine-compatible processing is not a trivial process, both in natural languages and biological sequence [Figure 2]. In natural languages, tokenisation requires as prior knowledge the concept of a "word". While some languages such as English can be easily split into smaller elements (or words) based on space or punctuation separation, many languages do not have this exploitable property. In addition, some languages such as Mandarin have characters which change meaning depending on the context of neighbouring characters. A custom set of grammar and user-defined rules is often used, which is specific to the language of interest as well as the context of the study [8, 9]. At the same time, "stopwords" such as "the" that have low information content are also defined manually [10]. However, more recent methods developed on character-based languages allow "words" to be empirically derived from the data [11]. Similarly, in biology, sequences are often split into user-defined blocks of n-length known as k-mers, which are conceptually similar to n-grams in NLP [12]. Interestingly, the challenges faced in defining "words" in biology are similar to that of character-based languages, and the same preprocessing method to infer "words" from the data is compatible. The resulting tokens are subsequently transformed through ordinal encoding by assigning an integer value to each token. While one-hot encoding is an alternative, where tokens are transformed into a sparse vector, this consumes significantly higher compute resources in the context of NLP and is not always viable. [Table 1, 2]

The current state of the art of tokenisation in biology has not changed significantly since its original application. Biological tokenisation is limited to manually defined segmentation rules, for example breaking a sequence into subunits of n length called k-mers [13, 14, 15]. Unlike human languages, biological grammar and words are not fully understood, and using k-mers are sufficient to yield results, even though k-mers do not necessarily reflect biological reality or context of neighbouring sequences. Furthermore, the problem scales rapidly, as accounting for every possible combination of a k-mer, many of which may not necessarily be meaningful, is computationally expensive. Compounding this problem is the known issue of out-of-vocabulary words (OOV) in NLP, where novel k-mers or words absent from the initial dataset cannot be evaluated. Some methods attempt to mitigate the lack of biological meaning by including a range of overlapping k-mers [16], but the underlying issues of defining a "biological word" as well as computational complexity

remains. A method to empirically obtain biologically meaningful words or tokens from data in a computationally efficient way does not yet exist, and is required to fill this known gap in the field.

To obtain a quantitative data representation, the numeric representation of words is compiled after tokenisation [Figure 2]. Frequency-based [17] or embedding approaches [18] are commonly used strategies. Tokens are represented as count values or weighted count values in frequency-based approaches. With embeddings, tokens are represented as distributed vectors in n-dimensional space, allowing users to observe the semantic relationship between "words", which is a unique property among NLP data representations.

As a result, large matrices of values are obtained. In this context, machine learning (ML) and deep learning (DL) tools are well suited to extract meaningful biological information, and are becoming popular in both the biological or clinical sciences. Methods incorporating machine and deep learning are used to generate as well as interpret images, text, videos and other forms of data. In biology, ML has been used successfully to predict protein structure [1], analyse images [2], identify genomic as well as epigenetic features [3], and functionally annotate sequences [4] [Figure 3].

Commonly used algorithms both inside and outside language-based processing include decision tree-based models such as XGBoost [19] or Random Forest [20] and deep neural networks such as RNN (Recurrent Neural Network) [21] or Bidirectional Encoder Representations from Transformers (BERT) [22]. While the tree-based models are limited in scope to classification or regression, they have the useful property of being effective in cases where observations exceed samples, which is absent in most conventional machine learning algorithms. Thus, tree-based models are especially well suited to biological data which is commonly formulated with a few samples but a large number of features or observations [23]. Deep neural networks have a broader scope and can be adapted to most use cases if appropriately formulated data as well as metadata is present, but are more complex to interpret, although developing interpretable frameworks is an active area of research [24].

Multiple ways to formulate biological problems and represent biological data in the context of machine learning exist. Biomedical images such as Magnetic Resonance Imaging (MRI) and histology data require comparatively little preprocessing, while other modalities such as gene expression data or electronic health records (EHR) often require specific preprocessing to be reformatted into an appropriate data representation [25]. However, due to the sequential nature of biological sequence data such as nucleic acid or amino acid sequences, formulating biological sequence data as a NLP problem is often intuitive.

Underlying many of these methods is a large ecosystem of open-source user-friendly packages and tutorials which together significantly accelerates software development [26, 27, 28]. Developers of new methods can quickly test and release effective methods, or adapt existing ones. These packages are often highly abstracted, reducing the barrier of entry for developers or even users who want to apply a specific method to their own data. To our surprise, there is no machine learning specific analogue for this in the biological sciences [29], even though the biological sciences is the highest contributor to publicly accessible data growth [30] [Figure 3].

Paradoxically, the increased volume of data also contributes to irreproducibility. The large quantity of low quality data as well as metadata available [31] often

requires researchers to obtain data from its original source for re-inspection or re-training. In ML/DL, re-training a dataset is not a process that scales well, considering the volume of data, metadata, parameters as well as number of experiments which may need to be examined. Therefore, there is a need for a practical theory for pretrained models which can predict how, when, and why such models can be expected to perform well or poorly.

Therefore, we highlight three key problems. First, current biological sequence data segmentation approaches do not take biological context into account, as k-mers are arbitrarily generated, and creates an unsustainable vocabulary size of all their possible combinations. Second, the lack of high-level machine learning frameworks in genomics field amplifies the challenge of applying modern methods to biological data analyses. Third, the high degree of inaccessibility and irreproducibility in the biological sciences [31] limits the ability to compare models in the absence of raw data as well as metadata. We address these problems by implementing a biological sequence preprocessing method which takes biological context into account while reducing the amount of k-mers or tokens generated to just meaningful ones [11], and developing a ML framework surrounding this method which allows users to easily apply ML directly to biological sequence data [Figure 1]. Furthermore, our framework includes a module to compare DL models directly in the absence of raw data and metadata. We implement theory-driven empirical metrics to evaluate the quality of a trained model in the absence of training/testing data and without any detailed knowledge of the training/testing process [32]. This empirical metric is a useful addition to the existing list of quality control and interpretability metrics currently used in the community to predict the quality of pretrained models, design effective training procedures, improve network efficiency, quantify network robustness, improve ensemble learning techniques, and to analyze and improve large-scale contexts [33].

We incorporated state of the art methods as well as the main features used by conceptually similar packages in other domains, with a specific focus on user-friendliness, high-level abstraction, intuitive visualisations [34] and open-source licensing. In addition, we present a systematic review of multiple case studies using our package. This package, currently available through conda [35] and gitlab, is intended to be extended and built on, with any open source software developers welcome to contribute and add methods to the library.

Results

To demonstrate that ML can be applied to raw sequence data as well as our novel empirical tokenisation approach, we first developed a high-level computational classification framework capable of processing a range of input data formats using a range of algorithms and high-level strategies, including deep learning. We implement interactive and visual metrics, and provide extra features which are especially useful in the context of biological sequence data analyses. This is viewable alongside this publication for the user's reference [Supplementary Figure S1, Supplementary Table S1].

To show that our empirical tokenisation method and pipeline is agnostic to species as well as biological molecules, we obtained a prokaryotic and eukaryotic dataset and

processed them using the same computational framework [36, 37]. In addition, both DNA and protein sequences were used as input in their corresponding experiments.

To compare the effectiveness of our empirical tokenisation strategy with that of the conventional k-mer approach, we systematically performed both strategies and show the results of each. To obtain a realistic comparison, we considered the key variable of vocabulary size. A maximum vocabulary size of 32000 was set for our empirical tokenisation approach. Since the full vocabulary size of length 5 k-mers would be an order of magnitude larger, we performed feature selection to reduce this quantity of k-mers to a reduced subset of 32000 for all algorithms by selecting the most abundant k-mers.

For each case, hyperparameter sweeps were performed to identify the best non-overfit model [Supplementary Figure S1]. We then performed cross-validation and calculated the ROC (receiver operating characteristic) plot for each combination of algorithm and tokenisation strategy [Figure 4, Supplementary Table S1]. We make four important observations, within the scope of the tested parameters (1) while not ineffective, the empirical tokenisation approach is consistently outperformed by the conventional tokenisation approach in conventional machine learning, possibly due to vocabulary size limitations (2) classification score is improved in protein data relative to DNA, likely because of the enriched information content of proteins, (3) deep learning generally outperforms conventional machine learning on the same biomolecular type set and (4) a full deep learning hyperparameter sweep scales well and can be completed within a single day on our dataset, in contrast with some cases in conventional machine learning which take multiple days to complete.

To show that our empirical tokenisation strategy yields biologically meaningful information, we demonstrate that we can recapitulate highly important features directly [Figure 4]. Instead of manually selecting and rerunning multiple experiments with different combinations of k-mer lengths and sizes, we naturally derive the precise, variable-length sequences from the data. In many cases, these sequences map directly to known motifs, further demonstrating that our method is capable of preserving semantic relationships in both nucleic acid and protein sequence data regardless of species [Figure 4]. Some examples of the discovered DNA motifs include a known bacterial methylation motif [38] and a region implicated in activating human T-cells [39]. Protein motifs such as TRIP8B binding sites [40], a conserved motif in APOBEC2 [41], an immune response mediator [42] and a range of Zinc-Finger binding domains also were found [43]. At the same time, since our method is annotation-free, it is capable of discovering novel motifs. For instance, we discovered a motif which matches part of a promoter for a toxin regulating gene, but which has no known biological annotation [44]. In addition, we also detect features with interesting biochemical properties: a synthetic DNA sequence with a high sensitivity to miRNA used as a probe [45], and a synthetic protein sequence used in designing Zinc-Finger domains [46].

At the same time, we show that with our framework, we can achieve a fast turnaround time with most algorithms [Table S2]. In practice, we showed that a full hyperparameter sweep of up to 128 different iterations on the entire dataset can be carried out in a single day using either conventional ML or DL, with the exception of embedding approaches. The best result can then be manually selected for training and subsequent cross-validation.

Accounting for OOV words or tokens is a problem in any NLP workflow, as non-recorded tokens cannot be evaluated. However our method implicitly accounts for OOV. We achieve this by breaking down tokens to their simplest form, in this case individual nucleotides or amino acids. From these individual units, any novel nucleic acid or amino acid sequence can be reconstituted and therefore evaluated normally.

We considered the common challenge of comparing the performance of multiple trained deep learning models. In biology, two common problems are data as well as metadata unavailability [31], and long model retraining times due to biological data size. Without access to data, metadata and hyperparameters, we can only resort to examining information contained within a deep learning model itself, which are their layer weights and attributes [33]. Therefore, we implemented power law fitting to allow for a fast and intuitive comparison across multiple models [32]. Using this technique, power law exponents are calculated per weight layer and combined in a series of histograms. Here, we demonstrate that *Model 7* has the highest performance compared among eight deep learning models, as shown by its lowest median power law exponent score ($\tilde{6}$) and relatively narrow distribution [Figure 5].

Discussion

One of the novel concepts we introduce in our package is data-driven or empirical tokenisation applied to biological sequences. Conventional biological sequence analyses break sequences into arbitrary length blocks known as k-mers, since the natural language concept of a "word" is less well-defined in biology as well as many other natural languages. While effective in many applications, arbitrary tokenisation does not accurately represent many features in biological sequence data, and generates an exponentially larger set of tokens for downstream analysis. By repurposing a method originally developed for inferring "words" from character-based natural languages [11], we empirically derived k-mers from biological sequence data, which has never been done previously. This method was previously shown to recapitulate "words" in natural language from text corpora, and while many biological "words" remain unknown, its performance on natural language implies that meaningful short biological subsequences are being selected, including contextual information. Furthermore, one additional key feature is its handling of out of vocabulary "words" by decomposing a sequence into its smallest units to reconstruct any words not present in the vocabulary, which is a feature that conventional tokenisation methods lack.

It was not possible to rule out several technical reasons which may have affected our results [Figure 4]. First, a constraint was set during the empirical tokenisation process, which restricted the vocabulary size to 32000. While this constraint was equal for all experiments, it has an impact on the sequence segmentation process, where different words or tokens may arise from the data. However, in practice, the effect is usually mitigated as the most well-defined "words" will be detected as the most highly weighted segments. A second limitation is the hyperparameters in each experiment. While hyperparameters were optimised for all cases, it is not feasible to search every single combination of possible hyperparameters, and therefore it is theoretically possible to miss the true optimum hyperparameter set for each experiment. Similarly, when performing conventional tokenisation, the k-mer size

of 5 may not have been the optimal length, but it is equally infeasible to substitute in every possible combination of k-mer lengths. Third, while we selected our input data carefully, since biological grammar is not fully understood, we cannot rule out the possibility that there may be features in the data which limit the algorithms' effectiveness.

To implement our methods and to allow users to do the same, we created an open-source pipeline that addresses two closely related technical problems: (1) manual data preprocessing steps and (2) input data diversity. Our pipeline's scope is distinct from existing open-source machine learning frameworks or web interfaces which incorporate various combinations of APIs, model hubs, interactive visualisations and logging for reproducibility as well as interpretation [Table 2, Table S1]. Many of these have active communities where bugs and features are constantly discussed and resolved. However, these frameworks have specific use cases, are not domain specific and are usually assembled together into a pipeline which may not be modular for ease of use. Similarly, their tutorials are limited as it is not feasible to cover every possible use case, even though they are generally well documented. While each low-level Application Programmer Interface (API) streamlines model development, they require at least an intermediate knowledge of python [47] and theoretical knowledge of deep learning. Unlike these existing generic frameworks, our pipeline is intended specifically to provide a domain-specific, end-to-end pipeline by taking advantage of these multiple high-level frameworks. Importantly, we abstracted and automated the time-consuming data preprocessing step to ingest common biological sequence data formats directly.

The diversity of biological data modalities and analysis scopes further raises the barrier to performing data integration analyses, which is rapidly becoming the norm due to the increased system-level resolution it provides [29, 48]. Unlike well established non-biological data modalities such as text, video, image and audio, data modalities associated with biological sequence data alone is highly heterogeneous, with measurement types as well as scales varying per instrument, hypothesis and even within the scope of a single experiment. In addition, irreversible data transformation is commonly used to reduce features in conventional analysis pipelines. Simultaneous unification of these different measurements is an on-going problem. Thus, to be able to analyse any biological datasets in parallel, a common point before the many diverging data transformations has to be identified. Conveniently, the common unifying point in this context is the previously mentioned low-level biological sequence data. Given that data preprocessing is a significant bottleneck to all model development as well as testing, and that biological data is the fastest growing data type [30] [Figure 3], our pipeline lowers a critical barrier to machine learning in the clinical and biological sciences.

We also considered that biological data as well as metadata is not readily accessible or consistently formatted [31]. Therefore, we implemented a means to test model performance independent of data. While reproducibility is not a problem specific to the biological domain, it is amplified due to data inaccessibility for privacy reasons (especially clinical data), the diversity and size as well as scale of biological datasets relative to compute resource availability. Therefore, we applied power law fitting, which generates a series of interpretable quality control plots from model weights

[32]. Thus, different models can be compared without requiring the same input data and hyperparameters. Data-independent model testing is particularly important since data inaccessibility is expected to continue in the future, as past examples clearly show that it is easier to change core elements of a field than it is to change established workflows [49, 50, 51].

As part of our workflow design, we also considered real world situations, such as scalability and interoperability. Regarding scalability, in our workflow deep learning using the transformer architecture [52, 53] scales better than conventional machine learning, with a significantly lower memory requirement as well as sustainable runtime given the same data and similar parameters [Supplementary Table S2]. (We also note that the current k-mer state of the art results in a highly inflated vocabulary size and subsequently features due to the need to accommodate multiple sequence permutations). The deep learning workflow and case study data we demonstrated is capable of running on a single NVIDIA V100 GPU (graphics processing unit) within a day on our datasets, demonstrating that large scale compute is not required for every use case. In theory, it is even possible to reproduce our study on a laptop computer with an appropriate GPU and sufficient RAM (random access memory). However, when scaling to more data or different algorithms such as longformer [54], a high performance computing cluster or cloud compute will likely be required. In the context of interoperability, our deep learning workflow seamlessly integrates with any transformers [55] or pytorch-compatible API [26] while our conventional machine learning workflow integrates with scikit-learn [56] and XGBoost [19]. Connections to the high-level wandb platform are available [34]. Therefore, any methods that are compatible with a conventional pytorch workflow are applicable to the deep learning component of our pipeline.

We release our pipeline with the aim of starting a long-term community project providing a convenient toolkit for applying deep learning to biological sequence data. At this point in time, there are a few available algorithms, and the current scope is limited to biological sequence classification. Further expansions will add more algorithms, interpretability and increase the scope to tasks other than classification. Following the pattern of existing open-source software projects, our code is publicly accessible under a MIT license and we provide bug report templates as well as feature requests to streamline contributions.

Methods

Software

Git repository link and documentation: https://gitlab.com/tyagilab/ziran.

A python package and appropriate documentation is available for download and install via conda through:
`conda install -c tyronechen ziran`.

Data

Four sequence collections corresponding to two categories of proteins in *Streptomyces bingchenggensis* as well as *Homo sapiens*, along with their corresponding DNA sequence regions were obtained [36]. The raw fasta files were downloaded and sequences of interest were used as input to the pipeline.

Streptomyces bingchenggensis

- NAD(P)-binding Rossmann-fold domains; SCOP ID 51735
- P-loop containing nucleoside triphosphate hydrolases; SCOP ID 52540
- Corresponding genomic sequence; GCF_000092385.1_ASM9238v1

Homo sapiens

- Immunoglobulin; SCOP ID 48726
- beta-beta-alpha zinc fingers; SCOP ID 57667
- Corresponding genomic sequence; Homo_sapiens.GRCh38.76

To obtain the corresponding genomic DNA sequences, protein sequences were mapped back to the reference genome with BLASTp [57], and exons of the corresponding gene were subsequently extracted. Conventional reference mapping was not possible due to genomic annotation drift.

Workflow

A visual summary of the overall experimental design is presented for reference [Figure 6, Table 4].

Preprocessing

For data preprocessing, all sequences were first split into chunks of 512-length sequences, and treated as separate instances to accommodate technical and hardware limitations [Figure 6]. In conventional tokenisation, since "word" delimiters are not consistently defined in biology, we arbitrarily split the sequence into equal 5 length blocks known as k-mers on a sliding window. We then selected tokens based on frequency up to a vocabulary size of 32000 representing the most abundant tokens in the data. During empirical tokenisation, the stream of biological sequence data is passed in directly without any denoising steps using a user-defined vocabulary size of 32000, and as before we use this to recode the individual documents to a stream of ordinal tokens retaining positional information. In all cases, resulting tokens were ordinally encoded as a base step before further data transformations into the final input format which the model accepts for training. Data was assigned into training, testing, and validation buckets at a 90/5/5 split for future use and stored on disk as csv, json and parquet files. Each file type contains the same information, but are formatted in both human readable and machine readable formats for user convenience.

Data representation

Before entry into a machine learning model, data is transformed into a machine-compatible format [Figure 6]. One of the simplest forms of representing data is as low-level ordinal token encoding, which is achieved as part of standard preprocessing and is directly compatible with most deep learning architectures. The base ordinal encoding can be used to construct more sophisticated data representations such as token frequency tables or vectors representing a token in n-dimensional space. Biological sequence data are represented as each of these representations: embeddings, frequency as well as weighted frequency and low-level positional encodings. A vocabulary size of 32000 was used. We selected three well-established models to classify the corpora: Random Forest [20], XGBoost [19] and DistilBERT [53] [Figure 6].

Hyperparameter sweeps

A large combination of hyperparameters are available, and altering them may result in significant differences in performance. Therefore, we implemented hyperparameter sweeping through a combination of wandb [34], transformers [55] and scikit-learn [56] using random search, allowing us to specify a range of values per hyperparameter and search for the best combination. In addition, early stopping is possible in some cases and implemented with the hyperband approach [58]. The best model and set of hyperparameters were automatically saved. Cross-validation was then performed to assess the best model's performance.

Metrics and interactive visualisations

We captured all conventional metrics used in machine and deep learning, including: accuracy, F1 scores, ROC curves, precision and recall. Where applicable, interactive quality control plots are uploaded to wandb for interactive visualisation [34].

Comparing performance of multiple models in the absence of training data

In addition, we review a model independent of its input training or testing data by applying power law fitting directly on model weights, generating a range of summary metrics. Very broadly, consistently small alpha scores (*2¡x¡6*) is interpreted as an indicator of good model performance [33]. Overlaying multiple graphs allows one to easily select the best performing model. For a comprehensive explanation of this method and how to interpret the results, please refer to the original publication [32] and case studies [33].

Two key advantages of this technique exist. With independence from input data, it is now possible to assess model performance where input data is inaccessible or invalid, which is common in biological datasets. Furthermore, it is possible to compare performance across multiple models without relying on retraining a model with identical input data and hyperparameters. However, we note that this method is applicable to deep learning specifically and not to conventional machine learning.

Cross-validation

Standard k-fold cross-validation is performed to review the performance of the best model [59]. Similar to previous steps, metrics are collected in wandb [34] for visualisation [Supplementary Table S1].

Conclusion

Big genomic, transcriptomic and proteomic data analysis has always been complex, and with the exponential growth in data, modern methods capable of navigating these large datasets are necessary. In addition to providing a systematic overview of multiple machine learning algorithms on genome-NLP as a part of this study, we release our toolkit to facilitate and streamline the application of machine learning to biological sequence data. We show that machine learning is effective in classifying raw sequence data directly, and that our data-driven preprocessing is a viable option. Our programming framework provides the first end-to-end command line interface for applying machine learning to biological sequence data, and crucially, it automates the data preprocessing stage which is a bottleneck to all data analysis.

In addition to conventional metrics, we implement new methods which resolve three key issues in biology: automating time-consuming data preprocessing steps, empirically deriving k-mers from data while accounting for out-of-vocabulary "words" and reviewing multiple different deep learning models in the absence of raw data. Where applicable, metrics are provided as shareable interactive visualisations. We plan to continuously improve and add new features, and welcome contributions from the community, lowering the barrier for users to ask novel questions in large biological datasets through machine learning.

Acknowledgements
This work was supported by the MASSIVE HPC facility (www.massive.org.au). We acknowledge the helpful discussions and compute resources from the Monash eResearch Platform and Monash Bioinformatics Platform. We thank Yashpal Ramakrishnaiah for helpful suggestions on package management, code architecture and documentation hosting online. We thank Jasbir Dhaliwal for critically reviewing the manuscript. We thank Jane Hawkey for advice on recovering deprecated bacterial protein identifier mappings in NCBI. We thank Andrew Perry and Richard Lupat for helping resolve an issue with the package building process. Biorender was used to create many figures in this publication. We acknowledge and pay respects to the Elders and Traditional Owners of the land on which our four Australian campuses stand.

Funding
T.C. was supported by an Australian Government Research Training Program (RTP) Scholarship and Monash Faculty of Science Dean's Postgraduate Research Scholarship. S.T. acknowledges support from Early Mid-Career Fellowship by Australian Academy of Science and Australian Women Research Success Grant at Monash University. A.Y.P. and S.T. acnowledge MRFF funding for the SuperbugAI flagship.

Abbreviations
- API: Application programmer interface
- APOBEC2: Apolipoprotein B MRNA Editing Enzyme Catalytic Subunit 2
- AUC: Area under curve
- BERT: Bidirectional encoder representations from transformers
- BLASTp: Basic local alignment search tool protein
- BPE: Byte-pair encoding
- distilBERT: distilled Bidirectional encoder representations from transformers
- DL: Deep learning
- DNA: Deoxyribonucleic acid
- EHR: Electronic health records
- miRNA: microRNA
- ML: Machine learning
- MRI: Magnetic resonance imaging
- NLP: Natural language processing
- OOV: Out of vocabulary
- RF: Random forest
- RNA: Ribonucleic acid
- RNN: Recurrent neural network
- ROC: Receiver operating characteristic
- TRIP8B: Tetratricopeptide-repeat containing, Rab8b-interacting protein

Availability of data and materials
Source protein data was obtained from **SCOP IDs: 51735, 52540, 48726, 57667**. Corresponding DNA data was obtained from associated genomic assemblies for *Streptomyces bingchenggensis* **GCF_000092385.1_ASM9238v1** and *Homo sapiens* **Homo_sapiens.GRCh38.76**. Code associated with this project is located in gitlab https://gitlab.com/tyagilab/ziran. For full details on data and code please refer to Methods.

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Authors' contributions
T. C.: Conceptualization, Methodology, Software, Data Curation, Writing- Original draft preparation, Writing- Reviewing and Editing, Visualisation, Investigation, Validation. N. T., S. C.: Software, Writing- Original draft preparation. A. P.: Supervision. S. T.: Conceptualization, Methodology, Writing- Original draft preparation, Writing- Reviewing and Editing, Supervision.

Authors' information
Not applicable.

Author details
[1]Department of Infectious Diseases, Central Clinical School, Monash University, 3004, Melbourne, Australia.
[2]University of Delhi, 110021, New Delhi, India. [3]Indian Institute of Technology Delhi, 110016, New Delhi, India.
[4]School of Computing Technologies, Royal Melbourne Institute of Technology, 3000, Melbourne, Australia.

References

1. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., *et al.*: Highly accurate protein structure prediction with alphafold. Nature **596**(7873), 583–589 (2021)
2. Angermueller, C., Pärnamaa, T., Parts, L., Stegle, O.: Deep learning for computational biology. Molecular systems biology **12**(7), 878 (2016)
3. Li, Y., Shi, W., Wasserman, W.W.: Genome-wide prediction of cis-regulatory regions using supervised deep learning methods. BMC bioinformatics **19**(1), 1–14 (2018)
4. Yang, C., Yang, L., Zhou, M., Xie, H., Zhang, C., Wang, M.D., Zhu, H.: Lncadeep: an ab initio lncrna identification and functional annotation tool based on deep learning. Bioinformatics **34**(22), 3825–3834 (2018)
5. OpenAI: Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/ (2022)
6. Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., Liu, T.-Y.: Biogpt: generative pre-trained transformer for biomedical text generation and mining. Briefings in Bioinformatics **23**(6) (2022)
7. Acosta, J.N., Falcone, G.J., Rajpurkar, P., Topol, E.J.: Multimodal biomedical ai. Nature Medicine **28**(9), 1773–1784 (2022)
8. He, Y., Shen, Z., Zhang, Q., Wang, S., Huang, D.-S.: A survey on deep learning in dna/rna motif mining. Briefings in Bioinformatics **22**(4), 229 (2021)
9. Akiyama, M., Sakakibara, Y.: Informative rna base embedding for rna structural alignment and clustering by deep representation learning. NAR genomics and bioinformatics **4**(1), 012 (2022)
10. Fox, C.: A stop list for general text. In: Acm Sigir Forum, vol. 24, pp. 19–21 (1989). ACM New York, NY, USA
11. Kudo, T., Richardson, J.: Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226 (2018)
12. Waterman, M.S.: [52] computer analysis of nucleic acid sequences. In: Methods in Enzymology vol. 164, pp. 765–793. Elsevier, ??? (1988)
13. Crusoe, M.R., Alameldin, H.F., Awad, S., Boucher, E., Caldwell, A., Cartwright, R., Charbonneau, A., Constantinides, B., Edvenson, G., Fay, S., et al.: The khmer software package: enabling efficient nucleotide sequence analysis [version 1; referees: 2 approved, 1 approved with reservations]. F1000research (2015)
14. Brown, C.T., Irber, L.: sourmash: a library for minhash sketching of dna. Journal of open source software **1**(5), 27 (2016)
15. Dhaliwal, J., Wagner, J.: A novel feature extraction method based on highly expressed snps for tissue-specific gene prediction. Journal of Big Data **8**, 1–13 (2021)
16. Ng, P.: dna2vec: Consistent vector representations of variable-length k-mers. arXiv preprint arXiv:1701.06279 (2017)
17. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management **24**(5), 513–523 (1988)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
19. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
20. Breiman, L.: Random forests. Machine learning **45**, 5–32 (2001)
21. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science (1985)
22. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
23. Scornet, E., Biau, G., Vert, J.-P.: Consistency of random forests (2015)
24. Jung, H., Oh, Y.: Towards better explanations of class activation mapping. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1336–1344 (2021)
25. Wang, S., Celebi, M.E., Zhang, Y.-D., Yu, X., Lu, S., Yao, X., Zhou, Q., Miguel, M.-G., Tian, Y., Gorriz, J.M., *et al.*: Advances in data preprocessing for biomedical data fusion: An overview of the methods, challenges, and prospects. Information Fusion **76**, 376–421 (2021)
26. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
27. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., *et al.*: Tensorflow: a system for large-scale machine learning. In: Osdi, vol. 16, pp. 265–283 (2016). Savannah, GA, USA
28. MegEngine:A fast, scalable and easy-to-use deep learning framework. https://github.com/MegEngine/MegEngine (2020)
29. Chen, T., Tyagi, S.: Integrative computational epigenomics to build data-driven gene regulation hypotheses. GigaScience **9**(6), 064 (2020)
30. Stephens, Z.D., Lee, S.Y., Faghri, F., Campbell, R.H., Zhai, C., Efron, M.J., Iyer, R., Schatz, M.C., Sinha, S., Robinson, G.E.: Big data: astronomical or genomical? PLoS biology **13**(7), 1002195 (2015)
31. Choi, J., Pacheco, C.M., Mosbergen, R., Korn, O., Chen, T., Nagpal, I., Englart, S., Angel, P.W., Wells, C.A.: Stemformatics: visualize and download curated stem cell data. Nucleic acids research **47**(D1), 841–846 (2019)

32. Yang, Y., Theisen, R., Hodgkinson, L., Gonzalez, J.E., Ramchandran, K., Martin, C.H., Mahoney, M.W.: Evaluating natural language processing models with generalization metrics that do not need access to any training or testing data. arXiv preprint arXiv:2202.02842 (2022)
33. Martin, C.H., Peng, T., Mahoney, M.W.: Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. Nature Communications **12**(1), 4122 (2021)
34. Biewald, L.: Experiment tracking with weights and biases, software available from wandb. com (2020). URL https://www. wandb. com (2020)
35. Anaconda Software Distribution. Anaconda Inc. (2020). https://docs.anaconda.com/
36. Wheeler, D.L., Barrett, T., Benson, D.A., Bryant, S.H., Canese, K., Chetvernin, V., Church, D.M., DiCuccio, M., Edgar, R., Federhen, S., *et al.*: Database resources of the national center for biotechnology information. Nucleic acids research **35**(suppl1), 5–12 (2007)
37. Pandurangan, A.P., Stahlhacke, J., Oates, M.E., Smithers, B., Gough, J.: The superfamily 2.0 database: a significant proteome update and a new webserver. Nucleic acids research **47**(D1), 490–494 (2019)
38. Seong, H.J., Park, H.-J., Hong, E., Lee, S.C., Sul, W.J., Han, S.-W.: Methylome analysis of two xanthomonas spp. using single-molecule real-time sequencing. The plant pathology journal **32**(6), 500 (2016)
39. Soldevilla, M.M., Hervas, S., Villanueva, H., Lozano, T., Rabal, O., Oyarzabal, J., Lasarte, J.J., Bendandi, M., Inoges, S., López-Díaz de Cerio, A., *et al.*: Identification of lag3 high affinity aptamers by ht-selex and conserved motif accumulation (cma). PLoS One **12**(9), 0185169 (2017)
40. Bankston, J.R., Camp, S.S., DiMaio, F., Lewis, A.S., Chetkovich, D.M., Zagotta, W.N.: Structure and stoichiometry of an accessory subunit trip8b interaction with hyperpolarization-activated cyclic nucleotide-gated channels. Proceedings of the National Academy of Sciences **109**(20), 7899–7904 (2012)
41. Chen, K.-M., Harjes, E., Gross, P.J., Fahmy, A., Lu, Y., Shindo, K., Harris, R.S., Matsuo, H.: Structure of the dna deaminase domain of the hiv-1 restriction factor apobec3g. Nature **452**(7183), 116–119 (2008)
42. Schexnaydre, E.E., Gerstmeier, J., Garscha, U., Jordan, P.M., Werz, O., Newcomer, M.E.: A 5-lipoxygenase-specific sequence motif impedes enzyme activity and confers dependence on a partner protein. Biochimica et Biophysica Acta (BBA)-Molecular and Cell Biology of Lipids **1864**(4), 543–551 (2019)
43. Hoovers, J., Mannens, M., John, R., Bliek, J., Van Heyningen, V., Porteous, D., Leschot, N., Westerveld, A., Little, P.: High-resolution localization of 69 potential human zinc finger protein genes: a number are clustered. Genomics **12**(2), 254–263 (1992)
44. West, S., Kaye, S., Hamood, A., Iglewski, B.: Characterization of pseudomonas aeruginosa mutants that are deficient in exotoxin a synthesis and are altered in expression of rega, a positive regulator of exotoxin a. Infection and immunity **62**(3), 897–903 (1994)
45. Baker, M.B., Bao, G., Searles, C.D.: In vitro quantification of specific microrna using molecular beacons. Nucleic acids research **40**(2), 13–13 (2012)
46. Bhakta, M.S., Segal, D.J.: The generation of zinc finger proteins by modular assembly. Engineered Zinc Finger Proteins: Methods and Protocols, 3–30 (2010)
47. Van Rossum, G., *et al.*: Python programming language. In: USENIX Annual Technical Conference, vol. 41, pp. 1–36 (2007). Santa Clara, CA
48. Chen, T., Philip, M., Lê Cao, K.-A., Tyagi, S.: A multi-modal data harmonisation approach for discovery of covid-19 drug targets. Briefings in bioinformatics **22**(6), 185 (2021)
49. Zeeberg, B.R., Riss, J., Kane, D.W., Bussey, K.J., Uchio, E., Linehan, W.M., Barrett, J.C., Weinstein, J.N.: Mistaken identifiers: gene name errors can be introduced inadvertently when using excel in bioinformatics. BMC bioinformatics **5**(1), 1–6 (2004)
50. Abeysooriya, M., Soria, M., Kasu, M.S., Ziemann, M.: Gene name errors: Lessons not learned. PLoS Computational Biology **17**(7), 1008984 (2021)
51. Lewis, D.: Autocorrect errors in excel still creating genomics headache. Nature (2021)
52. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
53. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
54. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
55. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., *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, pp. 38–45 (2020)
56. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.*: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
57. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of molecular biology **215**(3), 403–410 (1990)
58. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research **18**(1), 6765–6816 (2017)
59. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the royal statistical society: Series B (Methodological) **36**(2), 111–133 (1974)
60. Breiman, L.: Classification and Regression Trees. Routledge, ??? (2017)
61. Shibata, Y., Kida, T., Fukamachi, S., Takeda, M., Shinohara, A., Shinohara, T., Arikawa, S.: Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical Report DOI-TR-161 (1999)
62. Marcus, M., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Journal of Computational Linguistics (1993)
63. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. " O'Reilly Media, Inc.", ??? (2009)
64. Loria, S., et al.: textblob documentation. Release 0.15 **2**(8) (2018)

65. Montani, I., Honnibal, M., Honnibal, M., Landeghem, S.V., Boyd, A., Peters, H., McCann, P.O., jim geovedi, O'Regan, J., Samsonov, M., Orosz, G., de Kok, D., Altinok, D., Kristiansen, S.L., Kannan, M., Bournhonesque, R., Miranda, L., Baumgartner, P., Edward, Bot, E., Hudson, R., Mitsch, R., Roman, Fiedler, L., Daniels, R., Phatthiyaphaibun, W., Howard, G., Tamura, Y., Bozek, S.: explosion/spaCy: V3.5.0: New CLI Commands, Language Updates, Bug Fixes and Much More. doi:10.5281/zenodo.7553910. https://doi.org/10.5281/zenodo.7553910
66. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, Malta (2010)
67. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., *et al.*: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, pp. 177–180 (2007)
68. Neubig, G., Nakata, Y., Mori, S.: Pointwise prediction for robust, adaptable japanese morphological analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 529–533 (2011)
69. Kudo, T., Yamamoto, K., Matsumoto, Y.: Applying conditional random fields to japanese morphological analysis. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 230–237 (2004)
70. Kudo, T.: Subword regularization: Improving neural network translation models with multiple subword candidates. arXiv preprint arXiv:1804.10959 (2018)
71. Schuster, M., Nakajima, K.: Japanese and korean voice search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5149–5152 (2012). IEEE
72. Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., *et al.*: Accelerating the machine learning lifecycle with mlflow. IEEE Data Eng. Bull. **41**(4), 39–45 (2018)

Figures

Figure 1 Graphical abstract summarising our *genomicBERT* workflow wrapped in the package *ziran*.

Figure 2 The upper half of the figure shows that the problem of defining words in non-space separated languages is identical to the problem of defining k-mers in biology. The left shows a tokenisation example in natural language, while the right shows a tokenisation example in biological sequence data. The second half of the figure describes a generic NLP processing pipeline. Compared to other data modalities, NLP has unique challenges. A n-dimensional matrix or vectors of numerical values is the core data representation used by any machine pipeline, instead of plain text. To convert text to numerical values requires a series of preprocessing steps [Figure 6]. In a conventional NLP preprocessing pipeline, text is first split into smaller elements in a step called 'tokenisation' and each token (that is, the smallest unit for preprocessing) is then assigned an integer value. [Table 1, 3] De-noising the data by 'removing punctuations or stopwords' may also be performed as part of this step. Next, a frequency transform or n-dimensional embedding is applied to obtain a 'data representation' of the matrix of values which is used as input for a machine or deep learning pipeline [Figure 6].

Figure 3 Machine learning has broad applications across the biomedical and clinical sciences. In part, this is powered by the fact that biological data has been the highest contributor to "big data" growth since 2015 [30], and data analyses are accelerated with increasing access to methods.

Figure 4 A boxplot showing the distribution of ROC-AUC (receiver operating characteristic-area under curve) scores for each combination of algorithm and data for (a) *Streptomyces bingchenggensis* and (b) *Homo sapiens*. Using the optimum set of hyperparameters from a sweep, each model was trained maximising the f1 score and eight k-fold cross-validations were performed to obtain the ROC-AUC scores shown. Feature importance scores were obtained using Gini importance (mean decrease in impurity) [60] and the top 50 were plotted for a (c) *Streptomyces bingchenggensis* DNA example and a (d) *Homo sapiens* protein example. A few prominent features with known biological attributes in the literature are highlighted.

Figure 5 A histogram of power law exponents for weight matrices from eight different deep learning models. Power law metrics are computed per weight layer of each deep learning model and overlaid to compare their performance in the absence of training data. Model 7 is shown to perform best due to its relatively low dispersion as well as median score.

Figure 6 A set of workflow diagrams. From top left (1): A description of input, output and steps throughout the pipeline. At minimum, fastq or fasta like files, along with a user specified scope and high-level parameters are required. Optional steps are coloured in light blue and marked by dashed arrows. (2) Top right: flow diagram showing how each preprocessing workflow was set up and how each dataset was preprocessed, with a small test example. "Case normalisation" and "Denoising" steps are theoretically optional for Natural Languages, but in practice implementing these steps improves the effectiveness of the resulting algorithm. (3) Bottom left: A flowchart describing the different ways tokenised data flows through a NLP pipeline. Tokenised data can be converted into frequency scores, dimensional representations or ordinal encodings. (4) Bottom right: A flowchart describing the flow of data through our specific experimental design. Frequency-based as well as dimensional representations were used in tree-based machine learning algorithms and ordinal encodings were used in deep learning algorithms

Tables

Additional Files

Additional file 1 — Supplementary Figure 1

Figure S1 Sample wandb interactive plots. Fully interactive plots are available in Supplementary Table S1.

Additional file 2 — Supplementary Table 1

Additional file 3 — Supplementary Table 2

Type	Strategy	Advantages	Limitations	Examples or published methods
Reduction	case conversion	Intuitive and fast	Only usable in languages where capitalisation is meaningful, irreversible character conversion	*"Hello world"* → *"hello world"*
Reduction	byte-pair encoding (BPE)	Intuitive and fast	Irreversible character conversion	BPE [61]
Reduction	stemming	Intuitive and fast	Dependent on a custom set of rules per language	*"happiness", "happier", "happy"]* → *"happ"*
Reduction	lemmatisation	More accurately represents language elements	Dependent on a custom set of rules per language	*"happiness", "happier", "happy"]* → *"happy"*
Denoising	stopword removal	Intuitive and fast	Dependent on a custom set of rules per language	*["hello", "world", "and", "mars"]* → *["hello", "world", "mars"]*
Tokenisation	segmentation (delimiter separation)	Intuitive and fast	Only usable in languages where language elements are separated by delimiters or language sub-elements (eg spaces, punctuation or contractions)	Penn TreeBank [62], TweetTokeniser [63], MWET [63], TextBlob [64], SpaCy [65], GenSim [66], Moses [67]
Tokenisation	segmentation (part of speech)	More accurately represents language elements	Dependent on a custom set of rules per language	KyTea [68], MeCab [69]
Tokenisation	n-gram	More accurately represents order of language elements	Inherits any limitations of the initial tokenisation approach	*["hello world", "and mars"]*
Tokenisation	segmentation (empirical)	More accurately represents language elements	Requires training for each corpus	Unigram [70], WordPiece [71], SentencePiece [11]
Tokenisation	segmentation (arbitrary length sequence)	Intuitive and fast	Does not accurately capture language elements	khmer [13], sourmash [14]

Table 1 Sequence preprocessing strategies commonly used in Natural Language Processing. Note that not every strategy is applicable to all languages, and a combination of strategies are commonly used.

Name	Scope	Reference
huggingface	High level API for transfer learning	[55]
MLflow	High level API for reproducible machine learning	[72]
MegEngine	Low level API for constructing models	[28]
Pytorch	Low level API for constructing models	[26]
scikit-learn	Low level API for constructing models	[56]
Tensorflow	Low level API for constructing models	[27]
wandb	High level API for reproducible machine learning	[34]
ziran	End-to-end pipeline from data processing to trained model for biological and clinical data	This paper

Table 2 Entries in this table include high-level APIs for developing, using and assessing models.

Type	Strategy	Advantages	Limitations	Examples or published methods
Token Encoding (one-hot)	Each token is represented as a sparse vector	Intuitive	Unrealistic compute time especially with large vocabulary sizes	*["hello", "world"] → [[0 0 0 0 1], [0 0 0 1 0]]*
Token Encoding (ordinal)	Each token is assigned an integer value	Intuitive, concise and fast	Base transformation which many other data representations are built on.	*["hello", "world"] → [0, 1]*
Token Encoding (positional)	Each token position is recorded	Positional information is accounted for during training	Compatible with specific architectures, sequence length is limited.	*["hello", "world"] → [0, 1] (0, 0)*
Token Encoding (type)	Each token type is recorded in the context of multiple group comparisons	Allows comparison of multiple token groups	Situational, used mostly in comparing two sentences in a question and answer context.	*["hello", "world"] → [0, 1] (0, 0); ["hello", "moon"] → [0, 2] (1, 1)*
Token Encoding (group)	A group of tokens or sequences (such as a sentence) is assigned a value.	Faster than individual tokens	Low level information is lost, situational usage.	*["hello", "world"] → [0]*
Dataset Encoding (frequency)	Encoded tokens are mapped to frequencies or weighted frequencies	Intuitive and fast, often sufficient even with information loss	Positional and contextual information is lost. Ineffective on tokens absent from vocabulary.	Count vectorisation [18], TF-IDF [17]
Dataset Encoding (embedding)	Encoded tokens are mapped as vectors onto n-dimensional space	Preserves semantic relationship between language elements	Ineffective on tokens absent from vocabulary.	CBOW [18], word2vec [18], dna2vec [16]

Table 3 Tokens are first recoded into a numeric form as a base. After this base step, more sophisticated transformations can be applied. Positional and group encodings are common in deep learning, while vector and frequency encodings are common in conventional machine learning.

Data type	Tokenisation method	Data representation
Biology	Arbitrary k-mer lengths	Frequency
Biology	Arbitrary k-mer lengths	Weighted frequency
Biology	Arbitrary k-mer lengths	Vectors in n-dimensional space
Biology	Arbitrary k-mer lengths	Ordinal positional encodings
Biology	Empirical tokenisation	Frequency
Biology	Empirical tokenisation	Weighted frequency
Biology	Empirical tokenisation	Vectors in n-dimensional space
Biology	Empirical tokenisation	Ordinal positional encodings

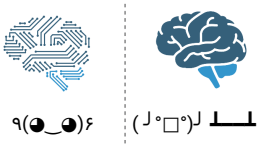
Table 4 A combination of each configuration of tokenisation method, data representation and algorithm compatible with our pipeline is run.

Table S1 Wandb interactive dashboard links for each deep learning experiment featured as part of this study.

Table S2 Total runtime of each method in human hours. Note that deep learning was run on GPUs while all other methods were run on CPUs as the non-deep learning methods cannot be run on GPUs with conventional machine learning packages.

MODERN DEEP LEARNING TOOLKIT FOR BIOLOGICAL DATA

1 Problem



High barrier for biologists

Existing high-level machine learning interfaces are tailored for machine learning experts and specific data types.

There is a lack of similar user-friendly machine learning kits for biologists and bioinformaticians.

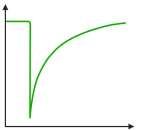
2 Solution

We introduce *ziran*

We solve this problem by providing a package which is designed for biological sequence data processing.

Our command line tool requires only the input sequence files and user-defined parameters.

3 Features



Highly visual and open source

Interactive visualisations with plots & tables of metrics and compute resources are generated.

Files are compatible with commonly used tools in the event where low-level customisation is needed.

4 Future

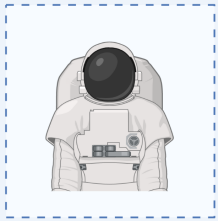
Extend to other methods

We will extend this package continuously with the latest state of the art methods.

Software is open-source and external contributions are welcome at https://gitlab.com/tyagilab/ziran

Challenges faced in preprocessing human language and biological sequence data

Human Language



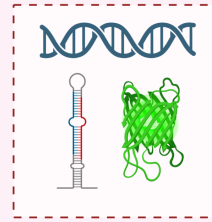
[Rule-based] Predefined words

```
EN: [Hello][world]
CN: [你][好][世][界]
```

[Data-driven] Learned words

```
EN: [Hello][world]
CN: [你好][世界]
```

Biological Sequence



[Rule-based] Predefined k-mer/n-gram

```
DNA: [ATCG][CGAT]
RNA: [AUCG][CGAU]
```

[Data-driven] Learned k-mer/n-gram

```
DNA: [AT][CG][CGAT]
RNA: [AUCGCG][A][U]
```

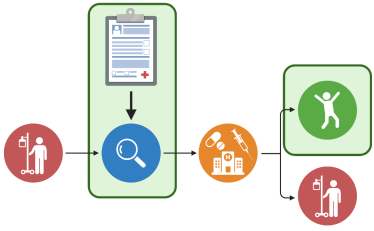
Data preprocessing for a conventional Natural Language Processing pipeline

Data flow from raw text to model training stage

Input sentence	Subword splitting	Denoising or stopword removal	Numerical transformation	Data representation for input into a machine learning model	A variety of methods are available for training
Hello The World	["Hello", "The", "world"]	["Hello", "The", "world"]	[31, 72]	Frequency transform [31: 20, 72: 12]	Tree-based models
CCCCCC TATAAA GGGGGG	["CCCCCC", "TATAAA", "GGGGGG"]	["CCCCCC", "TATAAA", "GGGGGG"]	[42]	Word vectors from embedding [[0.1 0.9 0.7], [0.7 0.8 0.9]]	Artificial neural networks

Applications of machine learning in the biological and clinical sciences

Text classification



Structure prediction

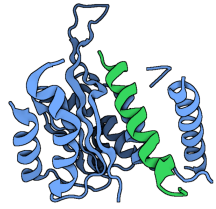
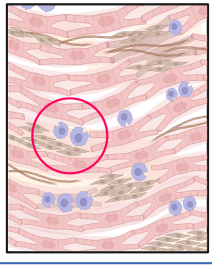
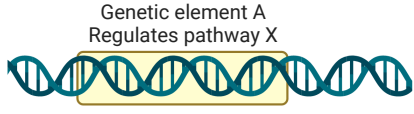


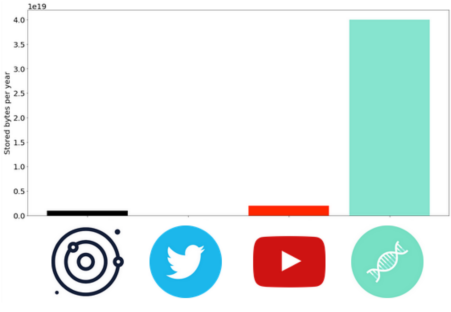
Image recognition



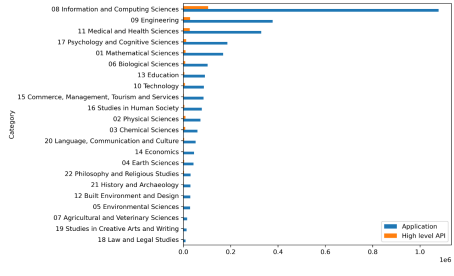
Gene annotation

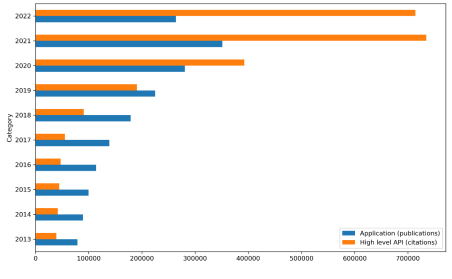


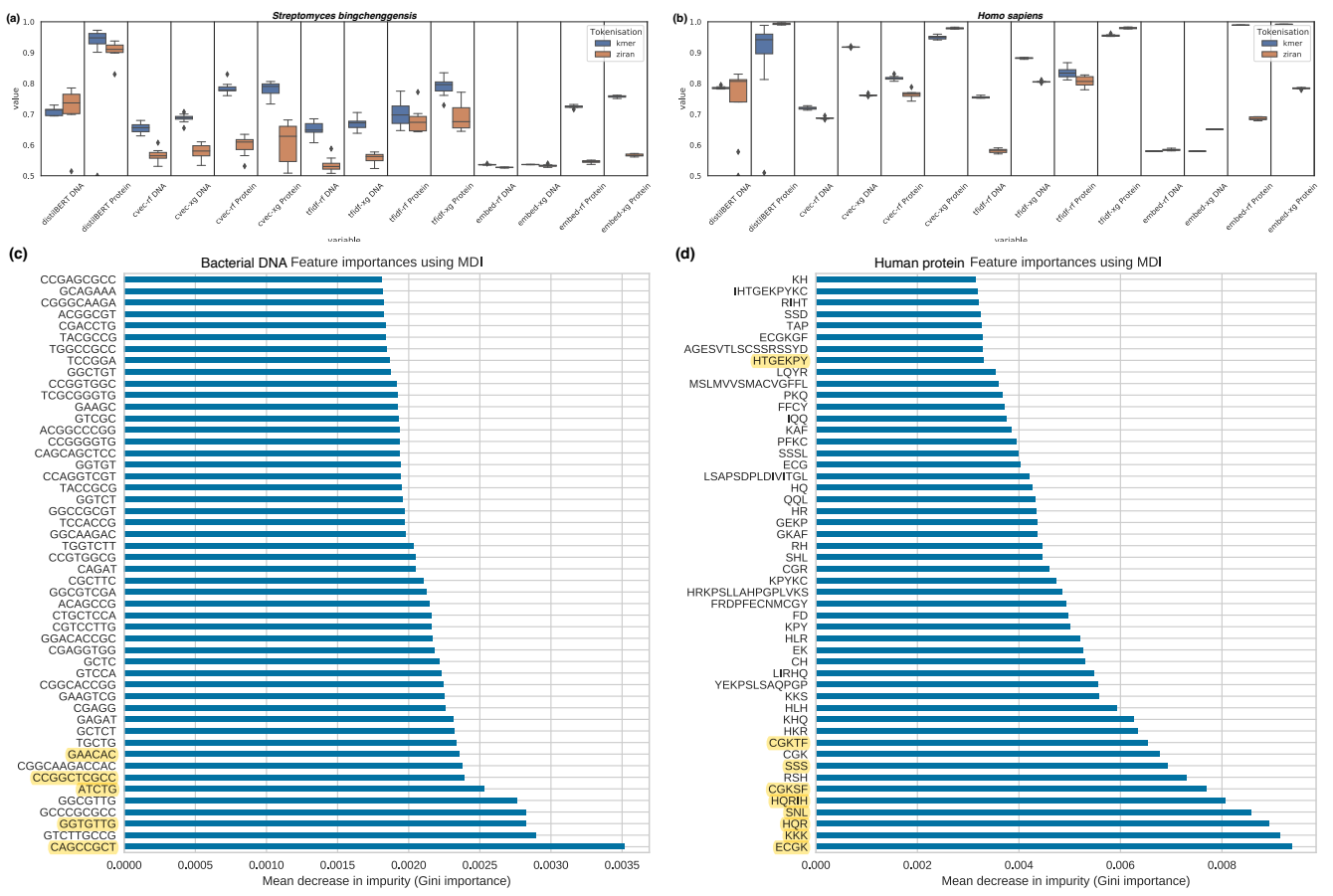
Genomics is the primary contributor to all data growth

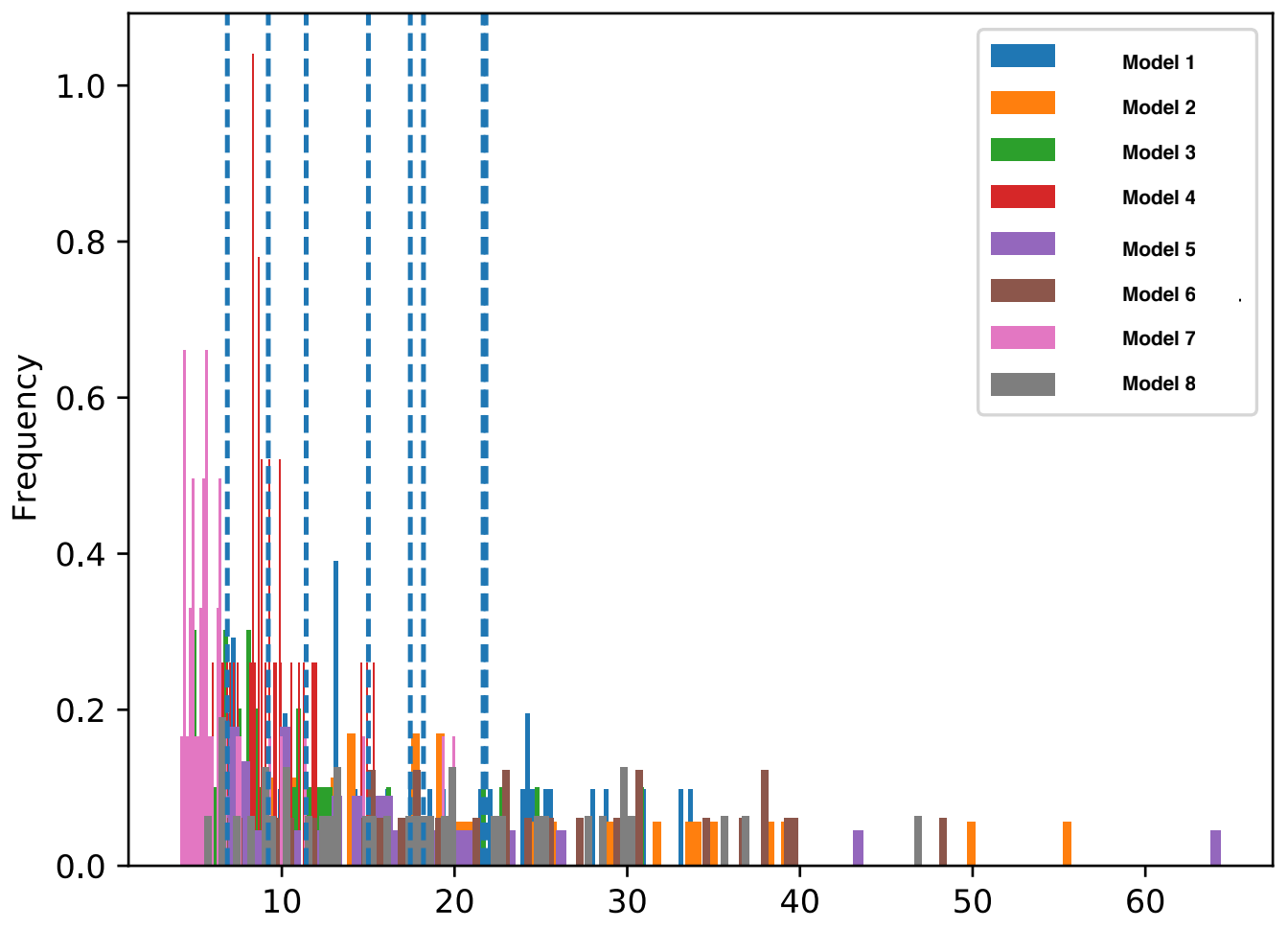


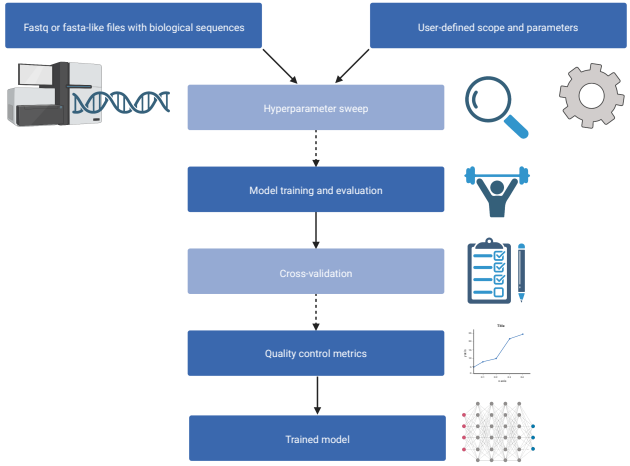
High level APIs accelerate the development of new methods

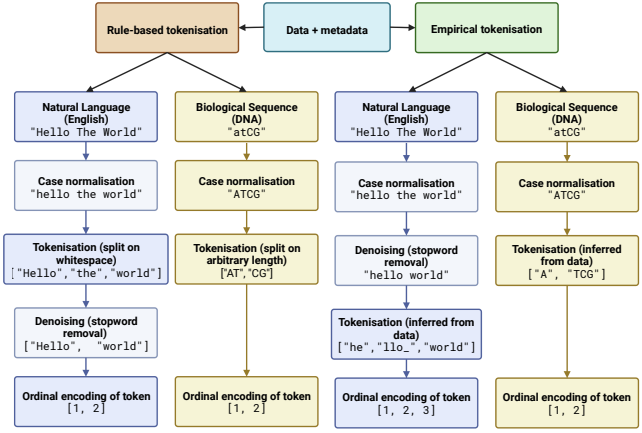


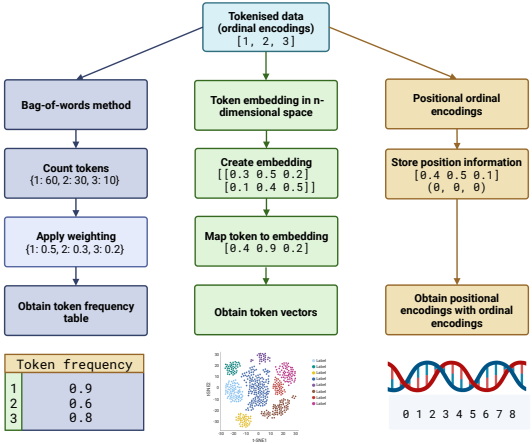


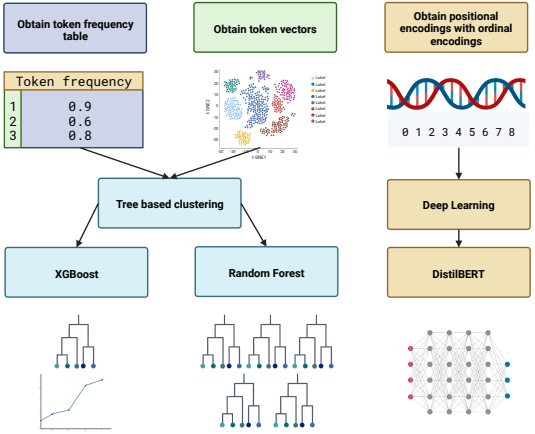

(a)
Streptomyces bingchenggensis
Tokenisation
kmer
ziran
value
variable
distilBERT DNA
distilBERT Protein
cvec-rf DNA
cvec-xg DNA
cvec-rf Protein
cvec-xg Protein
tfidf-rf DNA
tfidf-xg DNA
tfidf-rf Protein
tfidf-xg Protein
embed-rf DNA
embed-xg DNA
embed-rf Protein
embed-xg Protein
(b)
Homo sapiens
(c)
Bacterial DNA Feature importances using MDI
CCGAGCGCC
GCAGAAGA
CGGGCAAGA
ACGGCGT
CGACCTG
TACGCCG
TGGCCGCC
TCCGGA
GGCTGT
CCGGTGGC
TCGCGGGTG
GAAGC
GTCGC
ACGGCCCGG
CCGGGGTG
CAGCAGCTCC
GGTGT
CCAGGTCGT
TACCGCG
GGTCT
GGCCGCGT
TCCACCG
GGCAAGAC
TGGTCTT
CCGTGGCG
CAGAT
CGCTTC
GGCGTCGA
ACAGCCG
CTGCTCCA
CGTCCTTG
GGACACCGC
CGAGGTGG
GCTC
GTCCA
CGGCACCGG
GAAGTCG
CGAGG
GAGAT
GCTCT
TGCTG
GAACAC
CGGCAAGACCAC
CCGGCTCGGC
ATCTG
GGCGTTG
GCCCGCGCC
GGTGTTG
GTCTTGCCG
CAGCCGCT
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
0.0035
Mean decrease in impurity (Gini importance)
(d)
Human protein Feature importances using MDI
KH
IHTGEKPYKC
RIHT
SSD
TAP
ECGKGF
AGESVTLSCSSRSSYD
HTGEKPY
LQYR
MSLMVVSMACVGFFL
PKQ
FFCY
IQQ
KAF
PFKC
SSSL
ECG
LSAPSDPLDIVITGL
HQ
QQL
HR
GEKP
GKAF
RH
SHL
CGR
KPYKC
HRKPSLLAHPGPLVKS
FRDPFECNMCGY
FD
KPY
HLR
EK
CH
LIRHQ
YEKPSLSAQPGP
KKS
HLH
KHQ
HKR
CGKTF
CGK
SSS
RSH
CGKSF
HQRIH
SNL
HQR
KKK
ECGK
0.000
0.002
0.004
0.006
0.008
Mean decrease in impurity (Gini importance)


Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7
Model 8
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40
50
60


Fastq or fasta-like files with biological sequences
User-defined scope and parameters
Hyperparameter sweep
Model training and evaluation
Cross-validation
Quality control metrics
Trained model


Rule-based tokenisation
Data + metadata
Empirical tokenisation
Natural Language (English) "Hello The World"
Biological Sequence (DNA) "atCG"
Natural Language (English) "Hello The World"
Biological Sequence (DNA) "atCG"
Case normalisation "hello the world"
Case normalisation "ATCG"
Case normalisation "hello the world"
Case normalisation "ATCG"
Tokenisation (split on whitespace) ["Hello","the","world"]
Tokenisation (split on arbitrary length) ["AT","CG"]
Denoising (stopword removal) "hello world"
Tokenisation (inferred from data) ["A", "TCG"]
Denoising (stopword removal) ["Hello", "world"]
Tokenisation (inferred from data) ["he","llo_","world"]
Ordinal encoding of token [1, 2]
Ordinal encoding of token [1, 2]
Ordinal encoding of token [1, 2, 3]
Ordinal encoding of token [1, 2]


Tokenised data (ordinal encodings) [1, 2, 3]
Bag-of-words method
Token embedding in n-dimensional space
Positional ordinal encodings
Count tokens {1: 60, 2: 30, 3: 10}
Create embedding [[0.3 0.5 0.2] [0.1 0.4 0.5]]
Store position information [0.4 0.5 0.1] (0, 0, 0)
Apply weighting {1: 0.5, 2: 0.3, 3: 0.2}
Map token to embedding [0.4 0.9 0.2]
Obtain token frequency table
Obtain token vectors
Obtain positional encodings with ordinal encodings
Token frequency
1 0.9
2 0.6
3 0.8
0 1 2 3 4 5 6 7 8


Obtain token frequency table
Obtain token vectors
Obtain positional encodings with ordinal encodings
Token frequency
1 0.9
2 0.6
3 0.8
0 1 2 3 4 5 6 7 8
Tree based clustering
Deep Learning
XGBoost
Random Forest
DistilBERT