scientific reports

Check for updates

OPEN

Generative language models on nucleotide sequences of human genes

Musa Nuri İhtiyar✉ & Arzucan Özgür✉

Language models, especially transformer-based ones, have achieved colossal success in natural language processing. To be precise, studies like BERT for natural language understanding and works like GPT-3 for natural language generation are very important. If we consider DNA sequences as a text written with an alphabet of four letters representing the nucleotides, they are similar in structure to natural languages. This similarity has led to the development of discriminative language models such as DNABERT in the field of DNA-related bioinformatics. To our knowledge, however, the generative side of the coin is still largely unexplored. Therefore, we have focused on the development of an autoregressive generative language model such as GPT-3 for DNA sequences. Since working with whole DNA sequences is challenging without extensive computational resources, we decided to conduct our study on a smaller scale and focus on nucleotide sequences of human genes, i.e. unique parts of DNA with specific functions, rather than the whole DNA. This decision has not significantly changed the structure of the problem, as both DNA and genes can be considered as 1D sequences consisting of four different nucleotides without losing much information and without oversimplification. First of all, we systematically studied an almost entirely unexplored problem and observed that recurrent neural networks (RNNs) perform best, while simple techniques such as N-grams are also promising. Another beneficial point was learning how to work with generative models on languages we do not understand, unlike natural languages. The importance of using real-world tasks beyond classical metrics such as perplexity was noted. In addition, we examined whether the data-hungry nature of these models can be altered by selecting a language with minimal vocabulary size, four due to four different types of nucleotides. The reason for reviewing this was that choosing such a language might make the problem easier. However, in this study, we found that this did not change the amount of data required very much.

DNA is a significant building block of living organisms, responsible for carrying fundamental data about various factors affecting the life of the living beings to which it belongs, such as humans, animals, plants and others. DNA consists of nucleotides, yet some subsequences, called genes, have unique functionalities and special meaning.

Genes can be represented in textual format as sequences of four different types of nucleotides, namely adenine, thymine, cytosine and guanine, as shown in Fig. 1. The sequence length can vary from 100 to 10,000. Consequently, despite looking simple, the number of possible nucleotide sequences for a chosen size like 10,000 is vast (about 10^{6000}). Yet, only a few exist in humans, animals and other living beings. If we focus specifically on the genes that occur in humans, the number drops to about 70,000 (in the NCBI Gene Database as of March 2023).

Artificial intelligence and machine learning are among the most popular and exciting fields of our time. These fields involve helping computers or robots perform operations usually done by human beings, such as recognizing objects in an image, understanding natural language in the form of text, playing games like chess, or driving a car. However, they can also be used for more sophisticated goals, like helping humans make new scientific discoveries.

Because DNA nucleotide sequences resemble the text format of natural languages, methods commonly used for natural language processing could theoretically be used to solve various genomics tasks. Although the use of natural language understanding techniques, which involve understanding a particular sentence, is an option, the use of natural language generation methods, which is about modeling language using given sentences, is also possible in the field of genomics.

In this study, we investigated the problem of modeling nucleotide sequences for genes associated with humans. In other words, we looked at the performance of different machine learning models in the task of looking at

Department of Computer Engineering, Boğaziçi University, 34342 Istanbul, Turkey. ✉email: musa.ihtiyar@bogazici.edu.tr; arzucan.ozgur@bogazici.edu.tr

Gene X: ACCTTGGGGTAA

Gene Y: TTTGTCCAA

Gene Z: ATCGCTAG

Fig. 1. Nucleotide sequences for symbolic genes.

nucleotide sequences for some of the genes to obtain a model so that they can predict how likely it is that unseen sequences belong to a human gene. They could also generate new nucleotide sequences for candidate genes. The expected behavior is illustrated in Fig. 2.

Generative models have proven their success in different domains, most obviously with ChatGPT[1]. Theoretically, they can carry out almost all of the tasks done by a discriminative model. Especially combining different domains such as natural language and programming yielded incredible outcomes like OpenAI Codex[2], which can write code by only getting natural language as input. DALL-E 2[3] is another nice example from a different field. One of its most exciting achievements is replacing an object in an image with another thing without changing the background, called in-painting. Similar work can be done for the combination of any domain and natural language. Although the current version of our study only focuses on genes, the same idea could be applied to whole DNA sequences, and a similar combination with natural language could provide huge benefits. To give concrete examples, we might show a mutated DNA sequence causing disease and prompt it to convert this DNA sequence into the healthy version. Another option could be giving a sequence carrying a specific property like being resistant as input and wanting the program to describe how to integrate that feature into another organism's DNA so that it has the same property as well. In fact, theoretical limits for what can be done may be very close to the limits of what genomics could do at all. When all these possibilities are taken into account, we had better think carefully about ethical considerations too.

In Section "Related work", we discuss some related studies. That is, we give some concrete and specific examples on related areas to better understand the current research direction. Then, in Section "Dataset", we describe how the dataset we worked on was obtained and give various statistics about the dataset. In Section "Experiments and results", we explain in detail how the experiments were conducted and the results obtained. Finally, conclusions and future work are discussed in Sections "Conclusion" and "Future work", respectively.

Related work

The invention of computers hasn't affected only well-known areas such as digital sectors. It was very crucial for other fields like science in general as well. Due to the power of fast and reliable calculations they provided, computers were beneficial for lots of different scientific and technological processes.

Biology benefits from these machines with great potential. Various biological building blocks could be studied much more robustly and quickly with these solid computing machines[4,5]. Such efforts are sometimes referred to as bioinformatics or computational biology.

Proteins are one of the most fundamental building blocks when it comes to biology. Both discriminative and generative models are used for these critical elements. While McDowall and Hunter[6] provide a tool for automatic classification of proteins using models such as Hidden Markov Models, Madani et al.[7] attempt to develop a generative model for proteins based on big data that can be used for various application domains like medicine.

Since proteins can have different 3D structures, working with them requires special care. In addition, there are studies that focus on the interactions between proteins and various chemicals, as these are of great importance for topics such as drug development. For example, Nagamine and Sakakibara[8] use Support Vector Machines, a conventional machine learning algorithm, to predict the probability of interaction for a given protein and chemical pair. On the other hand, Cobanoglu et al.[9] use probabilistic matrix factorization, which is an alternative method for a closely related task.

DNA and RNA sequences have a massive importance in our lives; consequently, applications related to them are prevalent. The area is not new, even though various facilities in the past were limited. For example, Wang et al.[10] devised two algorithms, one unsupervised, for classifying DNA sequences. Nonetheless, Nguyen et al.[11]

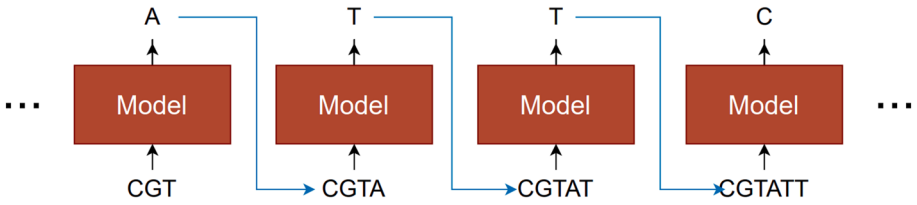

Fig. 2. An autoregressive generative language model for genes.

prefer using convolutional neural networks for classifying DNA sequences even though convolutional neural networks are not commonly used for classifying sequence-like inputs. Furthermore, Qi et al.[12] use an unsupervised clustering method on RNA sequences in addition to supervised applications.

Some studies used classical artificial intelligence methods to solve bioinformatics problems without using machine learning. Altschul et al.[13] use such an approach to measure the similarity of two sequences, as this is a crucial problem for tasks like search. Burge and Karlin[14] use a probabilistic approach to solve tasks such as gene identification. Lowe and Eddy[15] use the same technique for RNA-related work with a very low error rate. Also, Trapnell et al.[16] develop a robust method for gene expression by taking advantage of RNA sequences. Despite the existence of such studies, the integration of machine learning techniques in one form or another is extremely popular nowadays.

Traditional machine learning methods are a serious option for tasks with limited data, as deep learning approaches, their main competitors, need more data to perform better. Rannala and Yang[17] pursue a statistical method for estimating different species using DNA. Choi et al.[18] investigate the success of the classical logistic regression algorithm in RNA sequence studies.

As in other fields, deep learning is inevitably very common in bioinformatics because it is very successful when enough data is available for a given task. Lan et al.[19] mention several use cases for this, which is expected since deep learning is extremely popular in this field. As an interesting example, Zeng et al.[20] use deep learning to make predictions about RNA and protein pairs to better understand these structures.

There are some other works using machine learning on proteins. AIPs-SnTCN[21] uses a combination of embedding techniques to make classification on peptides. Moreover, cACP-DeepGram[22] uses combinations of deep learning methods for a cancer-related study.

Lastly, using language models is also a popular topic. For instance, DNABERT[23] uses an approach similar to the classical BERT, where data comes from human DNA for training and various DNA sources for testing instead of natural language in order to solve various tasks related to genomics. While DNABERT worked on datasets where samples were DNA sequences, our study was on a smaller scale since our samples were nucleotide sequences of genes.

In addition, Luo et al.[24] try to do a similar study for the generative part of the coin, yet for biomedical text instead of directly using biological units. However, a successful study which trained a gigantic generative transformer model on structures like DNA or proteins is not available, to the best of our knowledge.

Like natural language processing, biomedical domains try to use language models as much as possible due to their vast potential and proven success in several applications. These models are also called biological language models. It is possible to find very different use cases, ranging from discriminative to generative when the subject is examined in depth[5,25–29].

As a result, bioinformatics is a very hot field, and our study focuses directly on DNA instead of other molecules such as RNA or proteins or interactions between pairs. In addition, currently, popular machine learning-based approaches, especially deep learning, are employed in our work. Also, in spite of the fact that DNABERT is one of the closely related studies to our interests, our task is selected as a generative problem, not a discriminative one, since the latter was studied more extensively and the former has, arguably, more importance. Lastly, our work is relatively small sized in terms of datasets and models owing to factors like limited resources. Nevertheless, it is work on an exciting and unexplored problem.

To provide a summary, Transformer model[30] provided huge improvements in natural language processing. This is valid for both discriminative models like BERT[31], RoBERTa[32] and generative models like GPT[33], GPT-2[34], GPT-3[35]. Owing to similarity between natural languages and biological languages, they were tried on biological data like DNA and protein sequences as well, yet it was done mainly for discriminative cases. DNABERT[23], ProteinBERT[36] and DNABERT-2[37] gave successful results in these domains. These crucial milestones from natural language processing or bioinformatics are listed chronologically in Table 1.

Dataset

The aim of preparing the dataset was to use the nucleotide sequences for genes existing in human beings. The National Center for Biotechnology Information (NCBI)[38], a government institution, provides different databases such as Genome, Gene, and Nucleotide. From the Gene Database, we filtered genes such that only the ones

Model	Year
Transformer[30]	2017
GPT[33]	2018
BERT[31]	2018
GPT-2[34]	2019
RoBERTa[32]	2019
GPT-3[35]	2020
DNABERT[23]	2021
ProteinBERT[36]	2022
DNABERT-2[37]	2023

Table 1. Some of the critical works related to our paper.

existing in human beings remained. Then we got information like the commonly used name, id used in the database, and nucleotide sequence for all these genes. Nucleotide sequences for a specific gene can be accessed using the Nucleotide database of NCBI, and the dataset we obtained is online[39]. Some general statistics about the dataset are given in Table 2 to provide a general overview of what kind of data we worked on.

Next, we had to filter out genes whose sequence length exceeded a certain threshold, since transformer-based models assume a certain maximum length. This threshold was set at 1000, since larger values led to problems in fitting the models to the memory of the computers. Then we used 80% of it for training and 20% of the dataset for testing purposes. The split was done randomly. Some general statistics on the datasets can be found in Table 3.

In addition to these operations, the use of a validation set for models requiring hyperparameter optimization was necessary. For this reason, the training set was further split into 75% and 25%, where the former corresponds to the training set without validation and the latter is the validation set.

The nucleotide sequence was the essential information for each gene, yet other details such as NCBI gene id, symbol, description, and gene type information were also given. Lastly, < and > symbols were used to represent the beginning and end of the genes on the nucleotide sequence column of the corresponding files, and processing was done accordingly in the experiments.

Experiments and results

The dataset and the code used for implementing the ideas described in this paper is reproducible and available online at https://github.com/boun-tabi/GenerativeLM-Genes for all different techniques used.

N-gram language models

For N-gram language models[40], we experimented with six different values of N, starting from one and going up to six. The perplexity values obtained are given in Table 4. The Natural Language Toolkit (NLTK) library[41] was helpful for both using N-grams and calculating perplexity.

N-gram language models with laplace smoothing

Getting the perplexity of infinity when N is selected as six was an interesting result since the training set contained thousands of genes having thousands of nucleotides, so encountering an unseen sequence with length six was not

	Whole dataset
Number of samples	70,475
Min length	2
Max length	2,473,603
Mean length	26,417.25
Median length	3566.0

Table 2. General statistics about the dataset.

	Actual dataset	Training/validation set	Test set
Number of samples	22,886	18,308	4578
Min length	2	2	4
Max length	1000	1000	1000
Mean length	365.51	364.35	370.13
Median length	295.0	295.0	295.0

Table 3. General statistics about the train and test datasets.

N value	Perplexity
1	3.9959
2	3.9214
3	3.8985
4	3.8834
5	3.8723
6	∞

Table 4. Results for different N values.

expected. Nevertheless, when the cause of the situation is inspected, it is noticed that the problem was related to understanding the beginning or end of the gene, which is a crucial component for modeling.

Then Laplace smoothing[42] is used to avoid this problem. Different N values are tried starting from six and going up to eight. The corresponding perplexity values obtained are given in Table 5.

The best model we obtained had about 66,000 parameters.

Long short-term memory based language model

After experimenting with N-gram language models, we continued with deep learning-based approaches. The first option was using Reccurrent Neural Network (RNN) based models, specifically Long Short-Term Memory (LSTM)[43] based models. For getting input, a layer for learning embedding and position information was used at the beginning. Then a single or multiple LSTM layers are used depending on hyperparameter configuration. Lastly, the aim was predicting the next word; because of this, a linear layer with the size of vocabulary many units are used with Softmax to get predictions. The model architecture is visualized in Fig. 3. Adam was chosen as the optimizer to use.

Hyperparameter optimization was a crucial component. The basic procedure followed is like the following. We started with a hyperparameter configuration which looked reasonable. After that, if the model was underfitting, relevant hyperparameters are changed so that the model's capacity is increased. When overfitting was seen, the opposite is done. The details can be seen in Table S1.

There were three main hyperparameters to optimize: embedding dimension, the number of LSTM layers to use and LSTM dimension. For any of these hyperparameter values, the model was run for 40 epochs. Then the result where the absolute difference between training and validation perplexities was less than 0.02 was chosen as the result of that hyperparameter trial. The corresponding perplexity values obtained for the best model are given in Table 6.

N value	Perplexity
6	3.8616
7	3.8390
8	3.8263

Table 5. Laplace smoothing results for different N values.

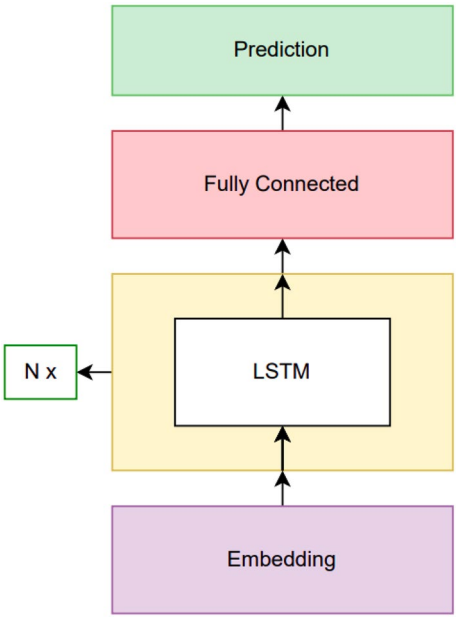

Fig. 3. Long short-term memory based architecture.

N value	Validation set perplexity	Test set perplexity
1024	3.2726	3.2878

Table 6. Long short-term memory result.

The best model we obtained had 8,140,552 parameters.

Transformer based language model

After using Long Short-Term Memory based models, transformer-based models were the next candidate. Actually, most of the procedures were very similar to the previous case. For getting input, a layer for learning embedding and position information was used at the beginning. Then a single or multiple transformer blocks are used depending on the hyperparameter configuration. Lastly, the aim was predicting the next word; because of this, a linear layer with the vocabulary size many units are used with Softmax to get predictions. The model architecture is visualized in Fig. 4. Adam was chosen as the optimizer to use.

Hyperparameter optimization was a crucial component. The basic procedure followed is like the following. We started with a hyperparameter configuration which looked reasonable. After that, if the model was underfitting, relevant hyperparameters are changed so that the model's capacity is increased. When overfitting was seen, the opposite is done. The details can be seen in Table S2.

There were four main hyperparameters to optimize: embedding dimension, the number of transformer blocks to use transformer feed forward dimension and number of heads to use for a certain transformer block. The model was run for 40 epochs for any of these hyperparameter values. Then the result where the absolute difference between training and validation perplexities was less than 0.02 was chosen as the result of that hyperparameter trial. The corresponding perplexity values obtained for the best model are given in Table 7.

Lastly, Tensorflow library[44], its high-level API Keras[45] and its NLP tool KerasNLP[46] were very beneficial for implementing our deep learning based models. The best model we obtained had 1,844,988 parameters.

The results obtained for different models based on test set perplexity is visualized in Fig. 5 using Matplotlib library[47].

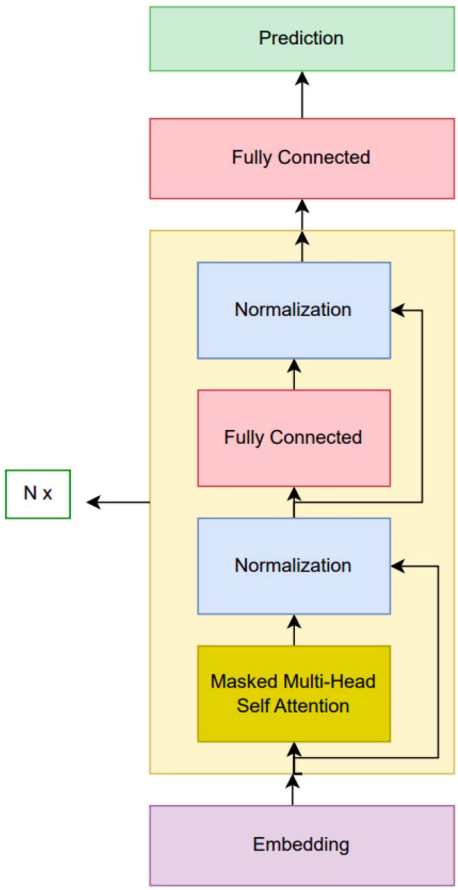

Fig. 4. Transformer architecture.

N value	Validation set perplexity	Test set perplexity
1024	3.6880	3.6905

Table 7. Transformer result.

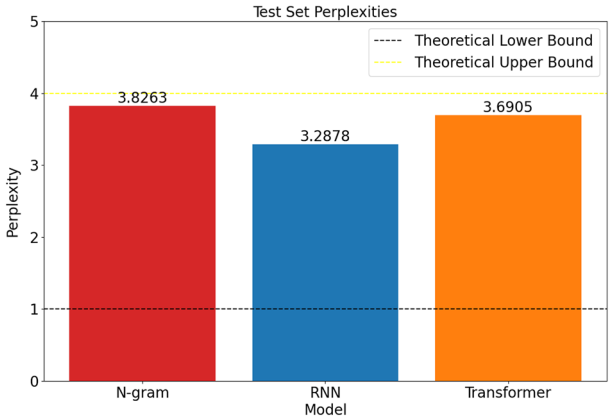

Fig. 5. Comparison of different methods based on test set perplexity.

Performances on real life tasks

Evaluating the success of the methods is not trivial, so a real-life task could be very useful. For this reason, we tried to find a real-life problem where modeling nucleotide sequences of human genes could be helpful. Distinguishing between real nucleotide sequences of a real gene and a mutated version seemed like an exciting task, since the ability to do such a task well requires some level of understanding of the data we were working on. In fact, what we did is similar to the evaluation of GPT-2[34] on tasks like Winograd Schema Challenge[48], in the sense that our model is trained as a generative model, but it could be adopted to do discriminative tasks when the correct setting is prepared.

Our strategy was to obtain pairs such that each pair consists of the nucleotide sequence of an actual human gene and its mutated version. Then obtained models would assign a probability or perplexity to both of them and predict which one is real and which one is mutated based on these values. If it uses probability, the one with a higher predicted probability would be selected as the real one. If the model uses perplexity, the gene with a lower value will be chosen as the real one since the lower, the better for the perplexity metric. Then the accuracy for predictions made will be checked to see the success of a certain model. After deciding on this strategy, the only remaining part was to obtain a dataset of real and mutated gene pairs.

Synthetic mutation dataset

One way to obtain a dataset consisting of real and mutated pairs is basically generating the mutated samples from the normal ones with the help of a computer by using a random process.

The method followed was like the following. Firstly, some hyperparameters such as the number of samples to obtain for the dataset and the maximum number of changes to be applied to a selected real gene, are determined. We selected the first as 1000. For the second, we experimented with three different values, which are 1, 5, and 10 to see how the performance changes according to this parameter. Lastly, since the process contains random number generation, a random seed was used to obtain reproducible results. For that part, we, again, used three different seeds to be able to use the general behavior.

After determining the hyperparameters, for each selected sample from the test set, the number of changes to apply was selected, which is between 1 and the maximum number of changes (both inclusive) with equal probability. Then for each change, an option among three different candidates was chosen with equal probability. These options were adding a new randomly chosen nucleotide to a random position, deleting an arbitrary position in the sequence, and changing the nucleotide on a randomly selected position to another one. The described operations are visualized in Fig. 6 and the mentioned algorithm is also described in Fig. 7.

As stated, this procedure was carried out for all three different values of the maximum number of mutations and three different seeds, leading to nine different synthetic mutation datasets. For synthetically obtained nine different datasets, containing real and mutated gene pairs, the best models from N-gram with Laplace smoothing, recurrent neural networks (i.e., LSTM in our case) and transformer-based language models were chosen. Their predictions were taken for each couple and the accuracy was calculated for each of them. The results obtained based on perplexity for three different random seeds, which were 419432, 623598 and 638453 in the code are given in Table 8.

The results obtained for different models based on synthetic mutation dataset perplexity is visualized in Fig. 8 using the Matplotlib library.

Real mutation dataset

Although synthetic datasets are beneficial for measuring the success of different alternatives, checking the performance on the real data is also very important. Consequently, finding mutations for genes in our test set was needed.

Human Gene Mutation Database[49–51] was a nice resource with almost no alternative for our goal. Even though the number of mutations we could find for genes in our test set was only 10, which is very limited, it still helped

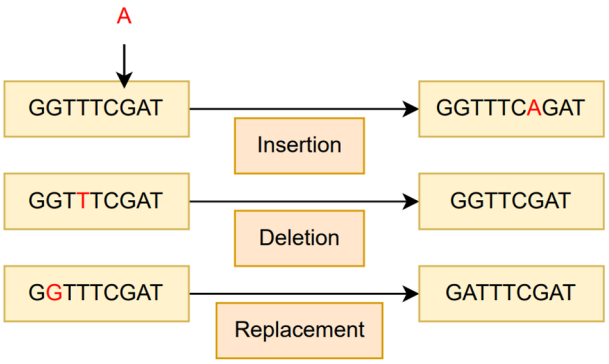

Fig. 6. Synthetic mutation dataset operation types.

```
Procedure: generateSyntheticMutationDataset
Input: realGeneSet: A set of nucleotide sequences for real genes
Output: realMutatedPairsSet: A set of nucleotide sequences pair for real and mutated genes
Begin
    Set hyperparameters N, M, SEED
    Set random seed
    realMutatedPairsSet ← []
    for i = 1...N do
        geneIndex ← randInt(1, len(realGeneSet))
        selectedSequence, changedSequence ← realGeneSet[geneIndex], realGeneSet[geneIndex]
        howManyChanges ← randInt(1, M)
        for j = 1...howManyChanges do
            changedSequence ← makeSingleChange(changedSequence)
        end
        realMutatedPairsSet.add( ( selectedSequence, changedSequence) )
    end
    return realMutatedPairsSet
End
```

Fig. 7. Procedure for generating a synthetic mutation dataset.

Model	Max change: 1	Max change: 5	Max change: 10
N-gram	(0.649, 0.647, 0.653)	(0.749, 0.741, 0.729)	(0.787, 0.793, 0.782)
RNN	(0.728, 0.745, 0.721)	(0.816, 0.785, 0.814)	(0.840, 0.846, 0.853)
Transformer	(0.638, 0.652, 0.665)	(0.711, 0.700, 0.690)	(0.766, 0.745, 0.753)

Table 8. Synthetic mutation dataset accuracy results based on perplexity predictions. The accuracy scores for each of the three seeds are shown in parentheses.

to show that our models could make nice predictions for real examples as well. Since the database is not publicly available, we will, unfortunately, not be able to share the obtained mutation dataset.

Details about the mutations and which model made a mistake in which example can be found in Table 9.

Conclusion

In this study, we used different natural language processing-based language modeling techniques to model nucleotide sequences of human genes. For big datasets, expecting better performance from deep learning-based methods, primarily transformer-based language models were standard; however, the problem we focused on was inherently the opposite. In other words, the actual genes existing in nature are limited. However, the language in nucleotide sequences looked relatively simple because there were only four words excluding the beginning and end of genes. Consequently, it was interesting to examine whether or not this phenomenon will reduce the gap for deep learning-based models.

When the results obtained are examined, it is clear that simple models such as N-gram language models performed better than their transformer-based competitors and performed close to recurrent neural network competitors. Also, it was seen that classical metrics used to measure success were not perfect in the sense that

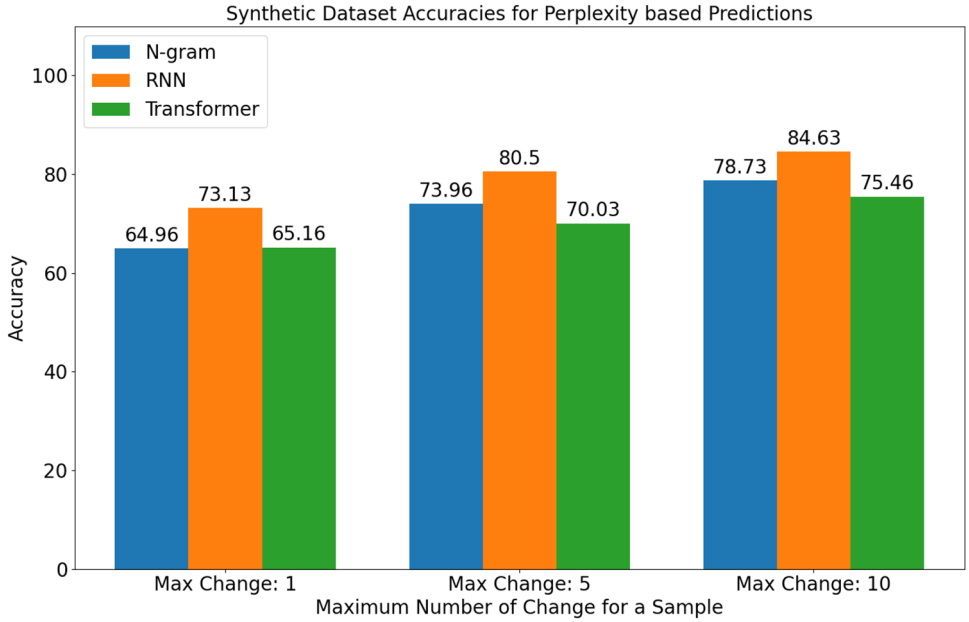

Fig. 8. Averaged accuracy results using perplexity for predictions.

Gene number	Mutation information	N-gram	RNN	Transformer
1	1 Substitution	False	True	False
2	1 Substitution	True	True	True
3	1 Insertion	True	True	False
4	1 Insertion	False	True	False
5	1 Substitution	True	True	True
6	1 Substitution	True	False	True
7	1 Substitution	False	True	False
8	1 Substitution	False	False	True
9	1 Substitution	True	True	False
10	10 Deletions	True	True	True

Table 9. Correctness of perplexity based predictions for different models.

N-gram models performed better than transformer-based models in real-world tasks, although they looked worse when the perplexity metric was used for the comparison. Moreover, because deep learning-based models had access to the same or more information about previous elements in the sequence at each prediction, neither recurrent neural networks nor transformer-based language models could achieve the desired level of improvement over classical N-gram models.

In addition, the obtained N-gram language models are superior to deep learning approaches regarding space complexity due to the fact that their model sizes are much smaller. When it comes to time complexity, the situation is again similar since deep learning methods require hyperparameter tuning, whereas N-gram language models require only a single training. When the results obtained are comprehensively taken into account, deep learning-based language models do not look far more appropriate for the task than simple N-grams although the recurrent neural network model was still the best. Furthermore, it seems like the data-hungry nature of deep learning models is still valid even for a case where we work on a language where the vocabulary size is minimal.

Future work

There is room for improvement in most scientific work. To further improve our work, some factors such as expecting developments from different areas of science, being able to access more powerful computational resources over time, and encouraging other people to work on similar problems by emphasizing possible outcomes of successful works on the studied problem ought to be remembered.

Firstly, genomics is a relatively new discipline compared to other branches of science like mathematics. Because of this, it is very normal to expect that our understanding will get better over time, which means that conducting a similar study in the future could give far more significant insights. Also, a greater theoretical lower

bound could be determined for the problem at hand over time. Coming up with such a reliable bound with our current knowledge was not easy for us.

Second of all, more detailed hyperparameter optimization strategies could give more intuition about why deep learning methods are not that successful. Of course, using powerful GPUs is essential for this, yet it could still be helpful for further studies.

Furthermore, expanding the dataset by using nucleotide sequences of genes existing in animals, plants and others could be an interesting work as well. Being able to find them might not be as simple as it was in this study and it requires huge computation resources such as GPUs, but the outcomes of the study could be far more general and conclusive.

Lastly, finding out more real-life problems strongly related to modeling nucleotide sequences of human genes could increase the scientific community's attention dramatically, which increases the amount of effort and investment put in to develop better strategies to model these incredible structures existing in nature, including our bodies.

Data availability

The dataset analysed during the current study is available in the following repositories. Code repository, https://github.com/boun-tabi/GenerativeLM-Genes and HGMD repository, https://www.hgmd.cf.ac.uk/ac/index.php.

Received: 17 October 2023; Accepted: 9 September 2024
Published online: 27 September 2024

References

1. OpenAI. Introducing ChatGPT. Accessed 8 March 2023. https://openai.com/blog/chatgpt (2022).
2. Chen, M. *et al.* Evaluating large language models trained on code (2021). arXiv:2107.03374 [cs]
3. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. & Chen, M. Hierarchical text-conditional image generation with clip latentsg (2022). arXiv:2204.06125 [cs]
4. Yang, A. *et al.* Review on the application of machine learning algorithms in the sequence data mining of DNA. *Front. Bioeng. Biotechnol.* **8**, 1032 (2020).
5. Li, H.-L., Pang, Y.-H. & Liu, B. BioSeq-BLM: A platform for analyzing DNA, RNA and protein sequences based on biological language models. *Nucleic Acids Res.* **49**, e129–e129 (2021).
6. McDowall, J. & Hunter, S. Interpro protein classification. *Bioinform. Comp. Proteomics* 37–47 (2011).
7. Madani, A. *et al.* Progen: Language modeling for protein generation (2020). arXiv:2004.03497 [q-bio.BM]
8. Nagamine, N. & Sakakibara, Y. Statistical prediction of protein-chemical interactions based on chemical structure and mass spectrometry data. *Bioinformatics* **23**, 2004–2012 (2007).
9. Cobanoglu, M. C., Liu, C., Hu, F., Oltvai, Z. N. & Bahar, I. Predicting drug-target interactions using probabilistic matrix factorization. *J. Chem. Inf. Model.* **53**, 3399–3409 (2013).
10. Wang, J. T. *et al.* New techniques for DNA sequence classification. *J. Comput. Biol.* **6**, 209–218 (1999).
11. Nguyen, N. G. *et al.* DNA sequence classification by convolutional neural network. *J. Biomed. Sci. Eng.* **9**, 280–286 (2016).
12. Qi, R., Ma, A., Ma, Q. & Zou, Q. Clustering and classification methods for single-cell RNA-sequencing data. *Brief. Bioinform.* **21**, 1196–1208 (2020).
13. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).
14. Burge, C. & Karlin, S. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* **268**, 78–94 (1997).
15. Lowe, T. M. & Eddy, S. R. tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Res.* **25**, 955–964 (1997).
16. Trapnell, C. *et al.* Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nat. Biotechnol.* **31**, 46–53 (2013).
17. Rannala, B. & Yang, Z. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics* **164**, 1645–1656 (2003).
18. Choi, S. H. *et al.* Evaluation of logistic regression models and effect of covariates for case-control study in RNA-seq analysis. *BMC Bioinform.* **18**, 1–13 (2017).
19. Lan, K. *et al.* A survey of data mining and deep learning in bioinformatics. *J. Med. Syst.* **42**, 1–20 (2018).
20. Zeng, C., Jian, Y., Vosoughi, S., Zeng, C. & Zhao, Y. Evaluating native-like structures of RNA-protein complexes through the deep learning method. *Nat. Commun.* **14**, 1060 (2023).
21. Raza, A. *et al.* AIPs-SnTCN: Predicting anti-inflammatory peptides using fastText and transformer encoder-based hybrid word embedding with self-normalized temporal convolutional networks. *J. Chem. Inf. Model.* **63**, 6537–6554 (2023).
22. Akbar, S., Hayat, M., Tahir, M., Khan, S. & Alarfaj, F. K. cACP-DeepGram: classification of anticancer peptides via deep neural network and skip-gram-based word embedding model. *Artif. Intell. Med.* **131**, 102349 (2022).
23. Ji, Y., Zhou, Z., Liu, H. & Davuluri, R. V. DNABERT: Pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. *Bioinformatics* **37**, 2112–2120 (2021).
24. Luo, R. *et al.* BioGPT: Generative pre-trained transformer for biomedical text generation and mining. *Brief. Bioinform.* **23** (2022).
25. Osmanbeyoglu, H. U. & Ganapathiraju, M. K. N-gram analysis of 970 microbial organisms reveals presence of biological language models. *BMC Bioinform.* **12**, 1–12 (2011).
26. Wang, Y. *et al.* A high efficient biological language model for predicting protein–protein interactions. *Cells* **8**, 122 (2019).
27. Benegas, G., Batra, S. S. & Song, Y. S. DNA language models are powerful predictors of genome-wide variant effects. *Proc. Natl. Acad. Sci.* **120** (2023).
28. Karollus, A. *et al.* Species-aware dna language models capture regulatory elements and their evolution. *Genome Biology* **25**(1), 83 (2024).
29. Liang, W. Segmenting DNA sequence into words based on statistical language model. *Nat. Proc.* 1–1 (2012).
30. Vaswani, A. *et al.* Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017).
31. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding (2018). arXiv:1810.04805 [cs]
32. Liu, Y. *et al.* Roberta: A robustly optimized Bert pretraining approach. arXiv:1907.11692 (2019).
33. Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. Improving language understanding by generative pre-training (2018). Accessed on February 9, 2024.
34. Radford, A. *et al.* Language models are unsupervised multitask learners. *OpenAI blog* **1**, 9 (2019).
35. Brown, T. *et al.* Language models are few-shot learners. *Adv. Neural. Inf. Process. Syst.* **33**, 1877–1901 (2020).

36. Brandes, N., Ofer, D., Peleg, Y., Rappoport, N. & Linial, M. ProteinBERT: A universal deep-learning model of protein sequence and function. *Bioinformatics* **38**, 2102–2110 (2022).
37. Zhou, Z. *et al.* DNABERT-2: Efficient foundation model and benchmark for multi-species genomes. In *The TwelfthInternational Conference on Learning Representations* (2024).
38. National center for biotechnology information. Accessed 8 March 2023. https://www.ncbi.nlm.nih.gov/ (1996).
39. İhtiyar, M.N. Datasets used including the whole dataset obtained and actual short gene dataset used including training, test and validation sets. 2023 (2023). Accessed on April 14, 2023.
40. Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423 (1948).
41. NLTK: Natural language toolkit. Accessed 8 March 2023. https://www.nltk.org/ (2023).
42. Lidstone, G. J. Note on the general case of the Bayes–Laplace formula for inductive or a posteriori probabilities. *Trans. Fac. Actuar.* **8**, 13 (1920).
43. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
44. Google. Tensorflow. Accessed 3 April 2023. https://www.tensorflow.org/ (2023).
45. Google. Keras: Deep learning for humans. Accessed 17 April 2023. https://keras.io/ (2023).
46. Google. Kerasnlp. Accessed 17 April 2023. https://keras.io/keras_nlp/ (2022).
47. Hunter, J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95. https://doi.org/10.1109/MCSE.2007.55 (2007).
48. Levesque, H., Davis, E. & Morgenstern, L. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning* (2012).
49. Stenson, P. D. *et al.* Human gene mutation database (hgmd®): 2003 update. *Hum. Mutat.* **21**, 577–581 (2003).
50. Stenson, P. D. *et al.* The human gene mutation database: Building a comprehensive mutation repository for clinical and molecular genetics, diagnostic testing and personalized genomic medicine. *Hum. Genet.* **133**, 1–9 (2014).
51. Stenson, P. D. *et al.* The human gene mutation database (hgmd®): Optimizing its use in a clinical diagnostic or research setting. *Hum. Genet.* **139**, 1197–1207 (2020).

Acknowledgements

This work is partially supported by Boğaziçi University Research Fund under the Grant Number 16903 and by ERC grant (LifeLU, 101089287). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Author contributions

M.N.İ. carried out literature review, obtained the dataset and conducted a number of experiments to compare different models. A.Ö. provided guidance and feedback for all of the stages of the study. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-72512-x.

Correspondence and requests for materials should be addressed to M.N.İ. or A.Ö.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2024