Enhancing Base Large Language Model Using Knowledge Graphs For Genomic Annotation

Pranav Desai*
Computer Science and Engineering
PES University
Bengaluru, India
pranav.ajaydesai@gmail.com

Sanket Padhi*
Computer Science and Engineering
PES University
Bengaluru, India
sanketff9088@gmail.com

Kavya B Panicker
Computer Science and Engineering
PES University
Bengaluru, India
kavyabpanicker18@gmail.com

Karishma Ravi Kumar
Computer Science and Engineering
PES University
Bengaluru, India
karishmaravi22@gmail.com

Divyaprabha KN
Computer Science and Engineering
Assistant Professor, PES University
Bengaluru, India
divyaprabhamadhu@gmail.com

***Abstract*—Annotating sequences of genes is a complex and constantly evolving process; existing annotations are refined constantly and new data is always being added. Attempts at automating this process have led to the use of Large Language Models and while these LLMs have great language processing capabilities, the standing notion is that they suffer from hallucination and have the tendency to "discover" new knowledge akin to fact fabrication. Now, contrary to this is a knowledge graph - unlike an LLM it cannot interpret knowledge it holds. However, it possesses a structured representation of domain-specific knowledge allowing it to accommodate new information with minimal computational overhead. The evolving landscape of genomic annotation means that LLMs will need to be finetuned to new data constantly and perform poorly when working with genomic sequences. To address these issues, we introduce a searchable knowledge graph constructed on various sources of gene and protein data demonstrating the efficacy of a system involving knowledge graphs and LLMs. We stipulate a framework to construct and search a knowledge graph, and index the hay of genomes into a more structured representation allowing for robust searching. When applied to a downstream task such as annotation generation, we show that such a symbiotic system results in improved accuracy for gene attribute prediction implying improved interpretability (providing insights into the decision-making of an LLM), reasoning and factual accuracy as a result of decreased hallucination from increased context.**

***Index Terms*—Large Language Models, Knowledge Graph, Genomic Annotation, Proteins, Fuzzy Search**

I. Introduction

Over the recent decades, Natural Language Processing has matured greatly and has proven its ability to capture the inherent patterns in the language we use. Transformer-based [1] models such as BERT [2], *Bidirectional Encoder Representations from Transformer* and GPT [3] *Generative Pre-trained Transformer* achieve great natural language understanding in this aspect. When this idea is extended to biological research, there seems to be a natural use case in NLP models for DNA sequences. There have been quite a few models that achieve great downstream results predicting various motifs of DNA sequences such as DNABERT [4] [5], Nucleotide Transformer [6], HyenaDNA [7], etc.

The common denominator amongst these models are that they are Large Language Models and are more specifically based on the Transformer architecture. These models consist of several layers of encoders/decoders stacked on top of each other where each layer adds additional understanding of the input through several attention heads and feed-forward mechanisms. Due to the stochastic nature of these models where every output is a probability that is based on its understanding from the pre-training, LLMs are prone to hallucination. There is a need to reduce hallucination for accurate results. A popular method discussed in [8], [9] and [10] is to create a system where an external knowledge base complements the LLM by grounding every output of the LLM in reality and giving it the necessary context that it may not have learned during its pretraining/finetuning. Thus allowing it to skip this step altogether.

Applying this system to biological research, a use-case emerges - genomic annotation. This is fundamental in advancing biomedical research as it involves attaching

*Both authors contributed equally to this work.

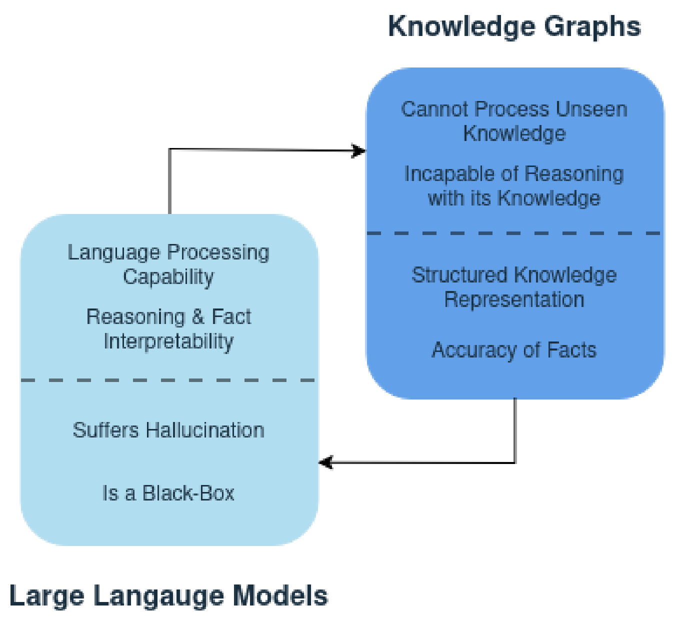
Knowledge Graphs
Cannot Process Unseen Knowledge
Incapable of Reasoning with its Knowledge
Structured Knowledge Representation
Accuracy of Facts
Language Processing Capability
Reasoning & Fact Interpretability
Suffers Hallucination
Is a Black-Box
Large Langauge Models

Fig. 1. LLMs and KGs Remedying Each Other

information to a sequence of genes. This information pertains to various important attributes such as gene boundaries, coding sequences, regulatory regions, and functional motifs. It provides an understanding of gene functionality, regulation, and expression patterns. Information from genomic annotation helps other researchers quickly identify sequences of interest for biomedical research. While there exist tools such as BLAST [11] or Ensembl Browsers [12] that already do a tremendous job at nucleotide and protein sequence matching and presenting relevant information across numerous species, there is no provision for an LLM to accomplish the same.

Taking ideas from these parallels, we stipulate an architecture that functions as follows. Given an input sequence queried by the user, it searches a knowledge base of indexed exon sequences to efficiently find one that is similar to the query. Then, it retrieves the facts related to the found similar gene and passes them to a general-purpose natural language model to perform further reasoning and inference.

In this work, we will also discuss how various datasets of gene and protein data were sourced to create a knowledge graph, describe the search tree that indexes the sequences in the constructed graph and prove how a base model pre-trained on DNA sequences can be used effectively with the search tree. The final system lets a user search for information like genomic annotations for an exon sequence and retrieves annotated gene and protein information sourced from GENCODE [13], UniProtKB [12], and Gene Ontology [14]. We select a few downstream tasks and show the improvement in the accuracies by using the pre-trained DNABERT-2 model as a base in our system. Ultimately, we present a unique application of VP trees to search for similar genes with a knowledge graph on base LLMs to show improvements in gene-protein information retrieval.

The organisation of the paper is as follows, Section II is all about related work, Section III is the methodology of the research work and implementation details, Section IV presents the results and discusses how they may be interpreted and finally the ending is the conclusion of the presented work in Section-V (with some further information in the appendix).

II. Related Work

Discussed to a great extent, the authors of [9] talk about how Knowledge Graphs (KGs) can significantly enhance the accuracy of Large Langauge Models (LLMs) by addressing their shortcomings - hallucination, lack of domain-specific knowledge and challenges they have with interpretability. These shortcomings stem from the fact that LLMs are stochastic models, have inherent biases or noise in their training corpora and semantic gaps in their understanding, often attributed to “common-sense”. Similarly, the authors of this paper [10] back the claim and propose enhancements solely to increase the factuality of an LLM. They suggest mitigating model-level deficiencies such as knowledge gaps and retrieval errors by integrating a knowledge base extrinsic to the base model itself.

These limitations seem to be in direct contrast to KGs, offering a structured representation of knowledge and facts. Different KGs serve different purposes such as Encyclopedic, Common-Sense, Domain-Specific and Multi-Modal KGs. They not only provide symbolic representations of knowledge allowing for context-driven reasoning, they also allow LLMs to be more interpretible and reason on domain-specific knowledge. Another use case for a KG is that it qualifies the knowledge cut-off, instead of spending compute in finetuning on the more recent events, potentially leading to catastrophic forgetting [15]. There are provisions to this too, methods such as Parameter-Efficient Fine-Tuning (PEFT) [16] and Low-Rank Adaptation (LoRA) [17] freeze most model parameters and fine-tune on a small subset - effectively reducing the compute needed to fine-tune and preserving the knowledge state of the model. These methods are recorded to work tremendously well especially with the addition of quantisation [18], but from the perspective KGs, merely adding more KG triples would be more than enough. The general idea behind these papers is to couple KGs with LLMs while also maintaining a degree of cohesion slotting well-defined responsibilities to each. The paper also goes in-depth on other paradigms of LLMs and KGs, but we focus our studies on developing the idea and constructing a domain-specific KG and allowing it to augment base LLMs.

Discussed before, several models perform well on genomic downstream tasks, of which the two most commonly cited and open-sourced models are - I) Nucleotide Transformer, a transformer model ranging 500 million to nearly 2.5 billion parameters pre-trained on a diverse set of sequences of genes through attention mechanisms, it can then be finetuned through supervised modelling for various downstream tasks of which it excels at many, some of them being promoter detection, transcript and splice site prediction etc. II) Another genomic pre-trained model is DNABERT2, with 117M model parameters, it is able to learn context-specific representations of DNA sequence and has performed better than Nucleotide Transformer on most benchmarks because of its novel Byte Pair Encoding (BPE) tokenisation and by modifying the BERT architecture to better suit its needs. Being a far lighter model than NT and providing marginally better results, we largely employ DNABERT2 to create embeddings for our search tree.

[19] highlights the challenges faced by LLMs in comprehending protein sequences, and by extension gene sequences to which genomic pre-trained models were briefly discussed. InstructProtein tries to employ a two-step process - pre-training on protein corpora such as UniprotKB and supervised fine-tuning with instructions derived from their Knowledge Graph (acyclic) which encode protein relationships such as their structure and function. To enhance this instruction dataset, they make use of Knowledge Causal Modelling, which is like Chain-Of-Thought [20], used to represent the causal chain of protein knowledge. This, when paired with a debiased sampling of instruction triples, leads to the model achieving state-of-the-art performance on protein understanding and design tasks. InstructProtein's results indicate that the integration of the KG, in a more abstract sense enhances the accuracy in protein-related downstream tasks such as function prediction and protein localisation. We pick up ideas from this paper on constructing a knowledge base of gene corpora that houses quality data with structured and causal relationships within.

III. Methodology

The architecture that we propose in our research introduces an approach to query a knowledge graph with any sequence of gene even if it does not exist in the graph.

The datasets are used to create relations that represent the attributes of the sequences. A tree is created that connects the sequence nodes in the graph for efficient retrieval. The algorithm requires a distance function. In our examples, we use one powered by the LLM DNABERT-2 that we will discuss shortly.

A. *Creating the Knowledge Graph (KG)*

In our research, we construct a KG that integrates diverse biological data sources focusing specifically on protein-coding data. Protein-coding genes are central to many biological processes, and understanding their structure and function is critical for various downstream genomic tasks, such as protein prediction, gene regulation, and disease research. The primary data sources we utilize for constructing the KG include Ensembl [21], GENCODE [13], RefSeq [22] , UniProtKB [12] , and Gene Ontology (GO) [14]. These resources provide

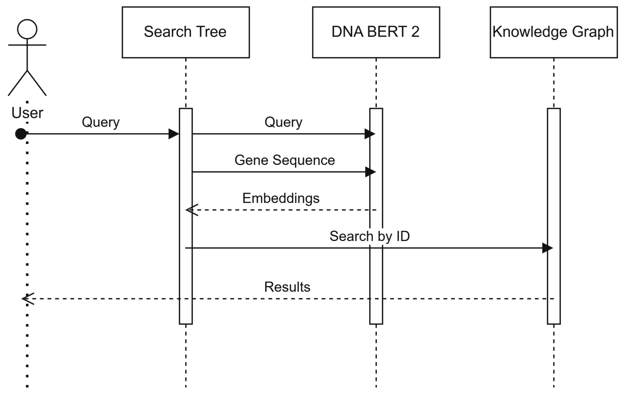

Fig. 2. Sequence Diagram

reliable, curated, and comprehensive data on genes, proteins, and their functions, making them ideal candidates for building a KG for effective genomic analysis.

*1) **Preprocessing and Cleaning the Data:*** Before constructing the KG, we first preprocess and clean the data obtained from the sources as listed in Appendix A. The data is carefully curated, removing redundancies and errors that could introduce noise into the model. The cleaned data is then stored as a CSV file making it easily accessible for further processing. One of the key steps in this process is performing multiple joins on selected fields such as gene IDs, protein IDs, and transcript IDs ensuring that the relationships between genes, proteins, and transcripts are well-represented in the graph.

*2) **Extraction of Exon Sequences:*** The next step in constructing the KG is the extraction of exon sequences from the Ensembl REST server. Exons are the coding regions of genes that are transcribed into mRNA and ultimately translated into proteins. Since exon sequences are directly related to gene and transcript structures, they serve as a convenient and efficient way to link genes and proteins in the KG. Exon sequences are relatively short compared to full gene sequences, which are often tens of thousands of nucleotides long, making them easier to deal with computationally.

*3) **Contextual Search and Integration with Natural Language Models:*** Once the knowledge graph is constructed, it becomes a valuable resource for contextual searches. When a user queries the system with an exon sequence, the search begins by traversing a tree that connects the sequences in the knowledge graph to find a set of sequence nodes that are similar to the query. We call this tree the "Search Tree".

The search then uses this set of nodes for retrieving relations and extracting the information, such as associated biological processes, molecular functions, and protein-protein interactions.

B. *The Search Tree*

Our architecture retrieves attributes of genes. Since the sequence queried by the user may not be present in the graph, a similarity search is required to find a sequences most similar to the query.

In our research, we have used a similarity search algorithm [23] that identifies similar sequences and is agnostic to the specific distance measure used. A distance measure is any function $d(a, b)$ that compares two sequences a and b and returns a scalar representing the distance between them in some abstract space. Such a set of objects associated with a distance function is known as a "Metric Space" [24]. Being agnostic to the distance metric makes the search algorithm suitable for any data that represents a metric space and thus can be applied to applications beyond gene research.

A distance measure we have tested involves using the genomic LLM DNABERT-2 [5]. We discuss using DNABERT-2 and the distance measure in more detail in the next subsection.

*1) **Constructing the Search Tree**:* We first decide on the maximum number of branches n that each sequence of the search tree will contain. Then, the method of constructing the tree from a set of sequences is as follows:

1) Let the number of sequences in the current set be l
2) The root of the tree r is selected randomly from the current set.
3) The distance from the root to every other sequence in the search space is calculated using the chosen distance measure.
4) These sequences are sorted based on the distance to the root.
5) The sorted set is divided into n equally sized parts such that each subset contains $\frac{l}{n}$ sequences.
6) This process is repeated for every subset recursively until there are no subsets formed.
7) The roots of the subsets are taken as the children of r. The minimum and maximum distances of each subset are also recorded.

The resulting structure of subsets forms the search tree, which organizes the search space for efficient querying.

*2) **Querying the Search Tree for a Single Sequence**:* When querying the tree for a specific sequence, the algorithm follows these steps:

1) The distance to the root sequence is computed using the same distance measure used during tree construction.
2) Based on the precomputed minimum and maximum values of the children, the query is directed to the appropriate subset.
3) The search algorithm is then applied recursively within the selected subset.
4) The search continues recursively until a single sequence is identified in the subset, at which point the algorithm terminates and returns the identified sequence.

For a balanced tree, the time complexity of this search process is $O(\log n)$, where n is the number of sequences in the search space.

Algorithm 1 Construct Search Tree

```
Require: Set of sequences $S$, maximum number of branches
    $n$, distance metric $d$
    BuildSearchTree $S$, $n$, $d$
 1: $l \leftarrow |S|$ {Get number of sequences in current set}
 2: if $l = 0$ then
 3:     return NULL
 4: end if
 5: $r \leftarrow$ RandomSequence($S$) {Select random sequence as root}
 6: $S' \leftarrow S \setminus \{r\}$ {Remove root from set}
 7: if $|S'| = 0$ then
 8:     return Sequence($r$)
 9: end if
10: $distances \leftarrow$ CalculateDistances($r$, $S'$)
11: $sortedNodes \leftarrow$ SortByDistance($S'$, $distances$)
12: $subsetSize \leftarrow \lceil \frac{l-1}{n} \rceil$
13: $subsets \leftarrow$ SplitIntoSubsets($sortedNodes$, $subsetSize$)
14: for each $subset\ s_i$ in subsets do
15:     $minDist$[i] $\leftarrow$ MinDistance($s_i$)
16:     $maxDist$[i] $\leftarrow$ MaxDistance($s_i$)
17:     $children$[i] $\leftarrow$ BuildSearchTree($s_i$, $n$)
18: end for
19: return Sequence($r$, $children$, $minDist$, $maxDist$)
```

Algorithm 2 Recursive Search for Single Sequence

```
Require: Root sequence $R$, query $q$, distance measure $d$
    RecursiveSearch $R$, $q$, $d$
 1: $dist \leftarrow$ ComputeDistance($R$, $q$, $d$)
 2: $subset \leftarrow$ FindSubset($R$, $q$)
 3: if $subset$ is empty then
 4:     return $R$
 5: end if
 6: return RecursiveSearch($subset$, $q$, $d$)
```

*3) **Querying the Search Tree for Sequences Within a Radius**:* In some cases, we may need all sequences within a specific radius q from the query sequence. The search process for sequences within a radius follows these steps:

1) The distance from the query to the root sequence is computed d using the same distance measure used during tree construction.
2) If the distance to the root is less than the specified radius, the root is added to the output list.
3) The distance range is calculated as $[d - q, d + q]$
4) We identify the subsets whose stored distances intersect with this range.
5) We recursively perform the search for all of the identified subsets.
6) All the sequences that were found are then returned at the end.

This method is useful to retrieve multiple sequences which provides more information for further downstream tasks.

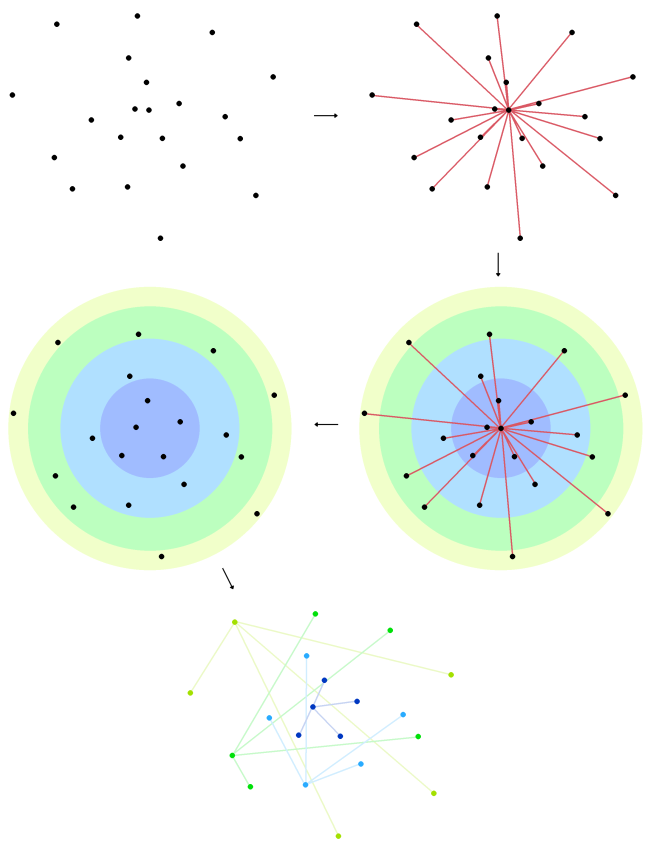

Fig. 3. Steps involved in creating the search tree: First, the distances (in red) are calculated. Then, the sequences are divided into equally sized sets (concentric circles). Finally, this process continues recursively to create a tree.

Algorithm 3 Range Query Search in Tree

```
Require: Root sequence $R$, query $q$, radius $r$, distance measure $d$ QuerySearch $R$, $q$, $r$, $d$
 1: $dist \leftarrow$ ComputeDistance($R$, $q$, $d$)
 2: if $dist \leq r$ then
 3:     Add $R$ to outputList
 4: end if
 5: $range \leftarrow [dist - r, dist + r]$
 6: $subsets \leftarrow$ FindIntersectingSubsets($R$, $range$)
 7: for each subset $s_i$ in $subsets$ do
 8:     outputList $\leftarrow$ outputList $\cup$ QuerySearch($s_i$, $q$, $r$, $d$)
 9: end for
10: return outputList
```

C. Using DNABERT-2 for a distance measure

DNABERT-2 is a pretrained model highly effective for genomic tasks that has shown impressive performance on various tasks such as coiled coil prediction, catalytic activity identification, binding site recognition, and activity site detection. Given its strong foundation from pretraining on extensive genomic data, DNABERT-2 already excels in capturing the complex patterns and relationships inherent in DNA sequences.

We take advantage of these capabilities of DNABERT-2 by using it to compute a distance measure between gene sequences. Using the weights in the output layer of DNABERT-2, we can generate the embedding of a given gene sequence g. This embedding is a matrix of dimensions $|g| \times 768$, where $|g|$ is the length of the gene sequence. We compute the mean along one of its axes such that it yields a 768-dimensional vector that we shall call $v(g)$. The distance is then defined as $d(q, s) = ||v(s) - v(q)||$, which is the normal distance between the two vectors of the sequences.

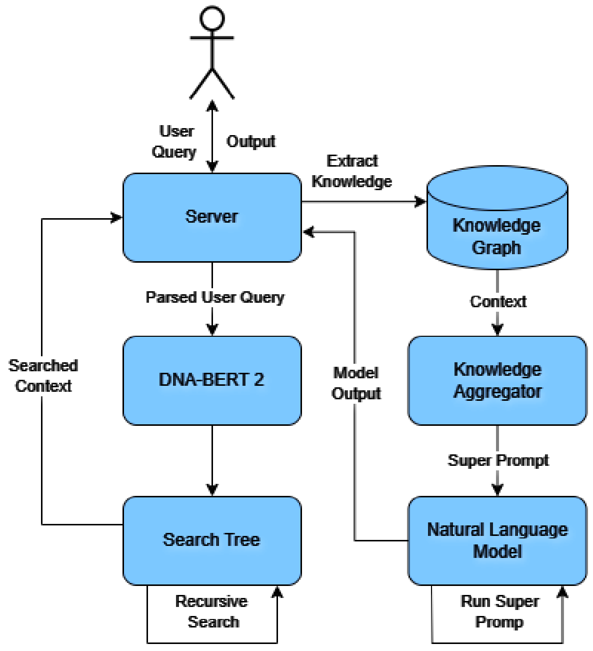

Fig. 4. Complete Data Flow

We have shown that we have used high-quality genomic datasets to construct a knowledge graph. We indexed the exon sequences of our graph to create a tree that can be used to search for similar sequences efficiently. As the final output of our system, we provide the users with the requested attributes by transitively traversing the graph from the matched sequence (the most similar sequence) to its related gene properties A. Finally, we showed how to use DNABERT-2 to compute a distance measure better capturing the similarity among gene sequences.

Through this approach, we leverage the strengths of both DNABERT-2 and the knowledge graph, resulting in a system that provides accurate and contextually relevant insights for downstream tasks. Ultimately, the model, enhanced with knowledge graph data, will offer new capabilities in genomic research and help address some challenges associated with interpreting complex biological data.

IV. Results and Discussion

In this chapter, we discuss the way we evaluate the performance of our architecture. We then present how varying different parameters affects the system performance.

A. *Evaluation Process*

We assume a use case where we query our system with a newly discovered gene sequence and estimate its attributes. Since the query does not exist in our knowledge graph, our system retrieves the attributes of a sequence that most closely matches the query. This means that the similarity measure is a crucial factor affecting the accuracy of our system.

The distance measure that we use for our search tree is the euclidean distance between two vectors that are a function of the two sequences the distance between which needs to be found. We use DNABERT-2 to compute these vectors as discussed in the previous section.

To conduct an experiment which closely resembles this use case, we must simulate gene sequences that are not present in the graph. This is because querying the system with sequences that are already in our datasets would result in an accuracy of 100%. This is not realistic. Instead, we edit the existing genes and query the system with the edited genes. This helps us two fold:

1) The dataset for evaluation need not be gathered from another source.
2) It gives us a measure of how our performance is affected by the dissimilarity of a queried sequence to the existing sequences.

We also define an edit function $e_k(s)$ that makes k edits to an input gene sequence s.

We have seen that when we query our system with a gene sequence q, we can specify a radius r for the search. The search returns $S_{r,q}$ which is the set of every sequence in our dataset D whose distance to the query $d(q,s)$ is less than r. Specifically:

$$S_{r,q} = \{s \in D : d(q,s) \leq r\}$$

We consider the search to be a hit if the original sequence exists among the outputs and a miss if it does not. More precisely, a sequence s results in a hit if and only if $s \in S_{r,e_k(s)}$. We compute this for all sequences within our dataset. The final accuracy is defined as the proportion of sequences from our dataset that produced a hit.

$$\text{accuracy}_{k,r} = \frac{|\{s \in D : s \in S_{r,e_k(s)}\}|}{|D|}$$

We can observe that this result does not depend on the choice of roots of the search tree or the number of children of its nodes. It solely depends on the search result set $S_{r,q}$, which is directly affected by the edits, the radius of the search and the distance metric.

B. *System Performance on different Parameters*

In the following results, we vary two parameters: the number of edits we make on the sequences e and the radius of the search r. The accuracy of our system is calculated for these parameters in the last column.

TABLE I
Accuracies of the KG-Integrated System

Search Radius (r)	**No. of Edits (k**	**Accuracy (%)**
1	1	81
1	2	79
1	5	86
1	15	79
1	25	71
1.6	1	91
1.6	2	90
1.6	5	92
1.6	15	88
1.6	25	83

The general trend observed is that the accuracy increases when e decreases or when r increases. In a realistic use case, an input query sequence has an unknown number of edits. For sequences within our knowledge graph, it means that it is able to retrieve the same information of the original gene with the stated accuracy. For sequences outside of our knowledge graph, the table above provides a confidence score for each given sequence which can be mapped based on the number of ”edits” the query is from the nearest sequence in our knowledge base. It is also to be noted that, the larger the knowledge base (i.e. holding more sequences and their related properties), the better the matches will be.

V. Conclusion and Future Work

The field of LLMs has seen a great rise in popularity in the recent years including their augmentation by the use of knowledge graphs. In this paper, we have described a system that leverages this concept in the field of genomic annotation. It makes use of an existing state-of-the-art model, DNABERT-2, to create an initial estimate and feed the result into a knowledge graph to obtain a more accurate prediction of the annotation. We have described a way to make use of the data from the knowledge graph without completely pretraining an LLM, but only fine-tuning it, and also described a way the knowledge graph could be used to retrieve information related to data not explicitly present in it.

Future advancements on this idea may benefit from pretraining an LLM on protein coding and JSON data. This would enable the model to understand the sequences better and be capable of directly accepting additional data from the knowledge graph. The architecture we described will automate and improve the processes that help us understand and decode the structure of genes.

References

[1] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al.. Attention Is All You Need; 2023. Available from: https://arxiv.org/abs/1706.03762.

[2] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding; 2019. Available from: https://arxiv.org/abs/1810.04805.

[3] Radford A. Improving language understanding by generative pre-training. Hi. 2018. Available from: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[4] Ji Y, Zhou Z, Liu H, Davuluri RV. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. bioRxiv. 2020. Available from: https://www.biorxiv.org/content/early/2020/09/19/2020.09.17.301879.

[5] Zhou Z, Ji Y, Li W, Dutta P, Davuluri R, Liu H. DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome; 2023.

[6] Dalla-Torre H, Gonzalez L, Mendoza-Revilla J, Carranza NL, Grzywaczewski AH, Oteri F, et al. The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics. bioRxiv. 2023. Available from: https://www.biorxiv.org/content/early/2023/03/09/2023.01.11.523679.

[7] Nguyen E, Poli M, Faizi M, Thomas A, Birch-Sykes C, Wornow M, et al.. HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution; 2023. Available from: https://arxiv.org/abs/2306.15794.

[8] Agrawal G, Kumarage T, Alghamdi Z, Liu H. Can Knowledge Graphs Reduce Hallucinations in LLMs? : A Survey; 2024. Available from: https://arxiv.org/abs/2311.07914.

[9] Pan S, Luo L, Wang Y, Chen C, Wang J, Wu X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. IEEE Transactions on Knowledge and Data Engineering. 2024 Jul;36(7):3580–3599. Available from: http://dx.doi.org/10.1109/TKDE.2024.3352100.

[10] Yang L, Chen H, Li Z, Ding X, Wu X. Give Us the Facts: Enhancing Large Language Models with Knowledge Graphs for Fact-aware Language Modeling; 2024. Available from: https://arxiv.org/abs/2306.11489.

[11] Ye J, McGinnis S, Madden TL. BLAST: improvements for better sequence analysis. Nucleic acids research. 2006;34(suppl_2):W6-9.

[12] Consortium TU. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Research. 2018 11;47(D1):D506-15. Available from: https://doi.org/10.1093/nar/gky1049.

[13] Frankish A, Carbonell-Sala S, Diekhans M, Jungreis I, Loveland J, Mudge J, et al. GENCODE: reference annotation for the human and mouse genomes in 2023. Nucleic Acids Research. 2022 11;51(D1):D942-9. Available from: https://doi.org/10.1093/nar/gkac1071.

[14] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, et al. Gene ontology: tool for the unification of biology. Nature genetics. 2000;25(1):25-9.

[15] Luo Y, Yang Z, Meng F, Li Y, Zhou J, Zhang Y. An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning; 2025. Available from: https://arxiv.org/abs/2308.08747.

[16] Xu L, Xie H, Qin SZJ, Tao X, Wang FL. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment; 2023. Available from: https://arxiv.org/abs/2312.12148.

[17] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al.. LoRA: Low-Rank Adaptation of Large Language Models; 2021. Available from: https://arxiv.org/abs/2106.09685.

[18] Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. QLoRA: Efficient Finetuning of Quantized LLMs; 2023. Available from: https://arxiv.org/abs/2305.14314.

[19] Wang Z, Zhang Q, Ding K, Qin M, Zhuang X, Li X, et al.. InstructProtein: Aligning Human and Protein Language via Knowledge Instruction; 2023. Available from: https://arxiv.org/abs/2310.03269.

[20] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al.. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models; 2023. Available from: https://arxiv.org/abs/2201.11903.

[21] Martin FJ, Amode MR, Aneja A, Austine-Orimoloye O, Azov A, Barnes I, et al. Ensembl 2023. Nucleic Acids Research. 2022 11;51(D1):D933-41. Available from: https://doi.org/10.1093/nar/gkac958.

[22] Pruitt KD, Tatusova T, Brown GR, Maglott DR. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. Nucleic Acids Research. 2011 11;40(D1):D130-5. Available from: https://doi.org/10.1093/nar/gkr1079.

[23] Yianilos PN. Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '93. USA: Society for Industrial and Applied Mathematics; 1993. p. 311–321.

[24] Fréchet MM. Sur quelques points du calcul fonctionnel. Rendiconti del Circolo Matematico di Palermo (1884-1940). 1906 Dec;22(1):1-72. Available from: https://doi.org/10.1007/BF03018603.

Appendix

Algorithms and Code Snippets

Here we show a high-level description of the inference routine which takes in an exon sequence and retrieves the related genomic annotation and protein information.

Algorithm 4 Perform Inference

```
Require: Prompt $Instr$
1: $ExonEmbeddings \leftarrow$ GenerateEmbeddings($ExonSeq$)
2: $ExonNode \leftarrow$ Search($ExonEmbeddings$, $Radius$, $QThresh$)
3: if $ExonNode \neq$ NULL then
4:     $Context \leftarrow$ InferKnowledgeFromGraph($ExonNode$)
5:     $Output \leftarrow$ NaturalLanguageModel($Instr$, $Context$)
6:     return Output
7: end if
8: return NULL
```

There were some optimisations to be on the inference routine such as -

- **Importing Search Graph In-Memory**: By emulating the search graph in-memory, we can mitigate the time taken by the API to run a query on a remote graph cluster. By cutting down on thousands of such calls, the search speed hastened up to more acceptable levels.
- **Pre-computing Sequence Embeddings**: The search function works with embeddings and needs them to be computed for every child it traverses. By precomputing all of the embeddings, we cuts down on the runtime of constantly calling DNABERT2.
- **Use of Multithreading**: Whenever massive amounts of data needed to be pulled or multiple search queries needed to be executed, the use of multithreading has expedited the process and cut down on the overall inference time.

Graph Schema

Working With DNABERT2

DNABERT2 is an advanced genomic foundation model designed to enhance the analysis of multi-species genomic sequences. It employs Byte Pair Encoding (BPE) for tokenization, which allows for variable-length tokens by merging frequently occurring nucleotide pairs. The biggest takeaway from tokenising the input sequence this way is that despite multiple edits to the gene sequence, it is able to generate similar embeddings allowing for holistic comparisions between sequences of genes. DNABERT2

```
Uniprot -[HAS_UNIPROT_PROPERTY]-> UniprotProperty [23969
relations]
Transcript -[PART_OF]-> Gene [2577 relations]
Sequence -[MAPS_TO]-> Transcript [20692 relations]
Sequence -[HAS_UNIPROT]-> Uniprot [20448 relations]
Sequence -[MULTI_SEARCH_BRANCH]-> Sequence [19994 relations]
Gene -[HAS_GO]-> Go [8173 relations]
Go -[has_namespace]-> GoProperty [9080 relations]
Go -[is_a]-> GoProperty [5702 relations]
Go -[has_id]-> GoProperty [2232 relations]
Go -[has_definition]-> GoProperty [2232 relations]
Go -[has_name]-> GoProperty [2232 relations]
Go -[part_of]-> GoProperty [492 relations]
Go -[positively_regulates]-> GoProperty [170 relations]
Go -[regulates]-> GoProperty [139 relations]
Go -[negatively_regulates]-> GoProperty [137 relations]
Go -[has_part]-> GoProperty [85 relations]
Go -[occurs_in]-> GoProperty [15 relations]
Go -[happens_during]-> GoProperty [1 relations]
```

Fig. 5. Knowledge Graph schema, the format of this is in Cypher, used to query entities, relations and properties in graphs. We use a Neo4j instance to host and query the graph.

modifies the original BERT architecture by implementing Attention with Linear Biases (ALiBi) instead of fixed positional embeddings, enabling it to handle longer input sequences without the typical constraints. This makes DNABERT2 a great tool for genomic sequence classification and analysis, facilitating deeper insights into genetic data across diverse species.

There are several reasons we employ DNABERT2 in our inference routine. Apart from being a state of the art model outperforming its competitors on most genomic downstream tasks, it is also a fairly lightweight model having only 117M parameters. This allowed for faster and more efficient finetuning. It also greatly cut down the inference time since it is computationally cheaper than its counter parts such as Nucleotide Transformers punching in at nearly 500m parameters with similar performance.

The finetuning process is pretty straightforward, we use a batch size of 4096 and set the learning rate to 3e-5. A total of 100,000 steps are used to train the model. To stabilize training at the beginning, we use a 50-step warmup. We then evaluate the model's performance every 200 steps during the training process.

TABLE II
FINETUNE RESULTS OF DNABERT2

Downstream Task	**Score (%)**
Coiled Coil	74
Catalytic Activity	81
Binding Site	70
Activity Site	78
Biological Process (GO)	76
Molecular Function (GO)	72
Cellular Component	83

We also finetuned DNABERT2 to act as a control for an identical set of downstream tasks. While we observed our system has definite improvements over DNABERT2 on all downstream tasks, we also want to highlight that the mere addition of facts to the KG allowed for better results as opposed fine-tuning the model all over again.

DATASETS

A. Gene Ontology

An essential tool in functional genomics, Gene Ontology offers a structured vocabulary for describing gene functions in a variety of biological situations. Gene Ontology (GO) was primarily created to standardize gene functions across different species. Before GO, different databases used inconsistent terms, making it hard to compare gene functions. The three ontologies it includes—Biological Process, Molecular Function, and Cellular Component—collectively provide a hierarchical description of the functions of genes and their byproducts. Scientists can formulate theories and make significant inferences regarding the roles and interactions of genes in intricate biological systems by quickly associating particular genes with GO keywords. An indispensable tool for genomic research, GO annotations are constantly curated and updated to guarantee that researchers have access to the most thorough and accurate functional classifications possible.

B. UniProtKB

Bioinformatics makes extensive use of UniProtKB, a vast database of protein sequences and functional information. It offers comprehensive protein annotations that cover their relationships, structures, functions, and post-translational modifications. UniProtKB is separated into two primary sections: UniProtKB/TrEMBL, which comprises automatically annotated entries from translated nucleotide sequences and UniProtKB/Swiss-Prot. The database's cross-referencing means that each sequence can be associated with other biological databases and resources, like the Gene Ontology. This integration facilitates the study of extensive datasets and advances our knowledge of the functions that proteins play.

C. GENCODE

A thorough annotation of the human genome, with an emphasis on gene characteristics and their functional components, can be found on the NCBI website that houses Gencode (GRCh38). Gencode is an essential tool for genomic research because it provides high-quality gene annotations, including protein-coding genes, non-coding RNAs, and pseudogenes. Researchers can reliably align sequencing data and discover genetic variants by using the GRCh38 version of the human reference genome as a basis for a variety of genomic investigations. In order to give researchers access to the most recent data on gene structures and functions, gencode annotations are updated often to reflect new genomic discoveries.