AI-based functional annotation of viral genomes

Sisard Fatess

July 1, 2025

Abstract

This project tests an AI method for functional assigment of bacteriophage genomes, trying to overcome the usual problems that arise from the extreme diversity of viral DNA. Borrowing ideas from studies of synteny-conserved gene order and from Natural Language Processing, we treat individual genes as "words" and the entire genome as one long "sentence". Genes are tagged with PHROG ortholog group IDs so that a Skip-gram Word2Vec model can read the genomes, capturing local neighborhoods in a sliding window of three genes and storing impressions in thirty-two-dimensional embeddings. Two simple follow-up strategies then guess gene function: one uses transfer-learning neural network to predict PHROG labels, the other groups similar embeddings by clustering at a threshold of 0.6 cosine distance. Classifier accuracy so far is modest, nearly matching the size of the training set and pausing on synteny alone, yet the clustering cleanly pulled together known families like HNH endonucleases and virion shell proteins.

Keywords: Viral genome annotation, Bacteriophage, Synteny, Word2Vec, Gene embeddings, Functional prediction, PHROG, Clustering, Neural network, Bioinformatics.

1 INTRODUCTION

Genome annotation – the process of deriving the structural and functional information of a protein or gene– [1] remains a fundamental challenge in bioinformatics. This initial step is essential for deciphering biological functions, understanding disease mechanisms, and guiding therapeutic development. While traditional methodologies have proven very effective for many organisms, the extreme genomic diversity characteristic of viruses, and for our case of study, bacteriophage virus, remains a consistent obstacle to accurate annotation. A substantial proportion of viral genes lack detectable homology to proteins with experimentally validated functions, often being annotated as "hypothetical protein" (up to 70

Conventional approaches for functional annotation primarily rely on homology searches, which infer function based on sequence or structural similarity to genes of known function. Homology in a gene context refers to "the share of biological features including genes and their products that are descended from a feature present in a common ancestor" [2]). Mentioned annotation methods commonly employ Hidden Markov Models (HMMs), derived from multiple sequence alignments within established databases such as PFAM and PHROG. Although those methods have proven very effective on predicting function based on homology they are unable to infer function for more diverse genomes, specially for viral genomes that present extreme variance and low homology.

An alternative principle for functional inference in highly heterogeneous genomes is synteny, which is defined as the "conservation of blocks of order within two sets of chromosomes that are being compared with each other"[3]. That is the conservation of genomic organization. Despite significant sequence divergence, viral genomes regularly exhibit conserved modularity, where genes contributing to a common biological viral life cycle stage are co-localized. This conservation of gene order and proximity can be observed even when individual gene sequences are evolutionarily unrelated(i.e. analogous). A representative example of this fenomena can be found with bacteriophage Q proteins, where distinct protein families for example Q21, Q, Q82 families[4] where no sequence similarity is observed among them but perform equivalent antitermination and antipausing activities and occupy analogous genomic locations. This is a proof that genes can share functional equivalence and genomic context without a shared evolutionary origin detectable by homology.Therefore, synteny seems as an interesting alternative to overcome the limitations of homologue-based annotation techniques. The work presented in this tfg is based on that principle, and we will focus on Q proteins, which will be the main reference when

• Contact E-mail: xxx@yyy.zzz
• Supervised by: Name and Surname of the Tutor (Department)
• Academic Year 2024/25

evaluating and judging embedding performance.

Taking inspiration from natural language processing (NLP), specially in capturing semantic relationship between words based on their contextual usage, this project proposes an AI based method for bacteriophage gene functional annotation, based on synteny. This approach follows a very common idea in gene bioinformatics that consists of framing genes as words i.e. tokens, within "sentences" in this case genomes. Specifically we will use Prokaryotic Virus Remote Homologous Groups database (aka. PHROGS)[5] in order to take advantage of the advancements in NLP, we intend to derive function through their genomic neighbourhood. The starting point will be to generate gene embedding via Word2Vec[6]. Protein-coding genes from bacteriophage genome datasets will be tokenized using one-hot encoding of their PHROG identifiers. A skip-gram word2vec model will then be trained on these tokenized genomes with a specified window size (w=3). This model will generate 32-dimensional gene embeddings that capture the local neighborhood associations, such that genes frequently co-occurring within the genomic window will possess similar vector representations. Then two methods are proposed to infer functionality. First one is the use of a neural network to learn patterns within the embedding space that correlate with gene functions. By processing these numerical representations of genomic context, the neural network aims to generalize and predict the functions of previously unannotated genes based on their learned associations with characterized genes. Also a clustering Analysis for embedding interpretation is performed to get the underlying relationships captured by the gene embeddings and to gain insights into functional groups, clustering algorithms will be applied to the generated embedding space. This will allow for the unsupervised identification of groups of genes that are assumed to be functionally similar based on their genomic context, providing a valuable tool for inspecting and understanding the semantic representations learned by the word2vec model. .

2 Previous Work

The idea of applying natural language processing (NLP) techniques to infer gene function based on genomic context has been previously demonstrated for bacterial genomes[7]. Miller et al proposed a technique that utilized deep learning approaches, specifically adapting NLP algorithms to model "gene semantics" within a large biological corpus. Similar to our proposed solution, their work considered genes as "words" and genomes as "sentences," employing a word2vec (skip-gram) model to generate numerical "gene embeddings" that capture local co-occurrence relationships. Miller et al also used a neural network for predicting functional categories and interpreted their embeddings using clustering, genes of similar function tended to cluster together. A significant distinction between their approach and ours lies in the genomic dataset used: Miller et al.'s study focused on bacterial (microbial) genes, training their models on an extensive corpus of over 360 million microbial genes, using KEGG ortholog groups (KOs) and clustered hypothetical proteins as their tokens. Our project introduces a novel application of this paradigm by specifically targeting bacteriophages genomes. This focuses on a significantly smaller, and distinct genomic corpus requiring adaptation. Furthermore, we leverage PHROG identifiers as tokens, which constitute a database of orthologous groups widely used for bacteriophage gene families. This specific application to bacteriophage genomes, with a high proportion of uncharacterized genes, constitutes a novel contribution to the field.

3 Methodology

A fundamental first step in the development of this project involves the precise definition of gene families. Because we are taking advantage of algorithms developed for NLP it is necessary to have a clear definition of what a token is (i.e. word) genes could be used as tokens but they are to diverse and it would be hard to infer meaningful meaning directly by using genes especially due to the high diversity characteristic of bacteriophages and the lack of huge dataset to learn a complex semantic representation. Therefore a decision was made to use gene families in concrete PHROGs orthologous groups this way we reduce the vocabulary size (different genes would be mapped to the same PHROG).

Once the tokens are defined we can proceed to define our corpus data, the Millard lab team has made an amazing work grouping and creating a dataset of bacteriophage virus therefore we will use this dataset as corpus but Millard lab data only provides protein sequences that means we have to manually "Translate those proteins sequences in to phrogs"

3.1 Corpus Creation

In order to annotate all protein sequences, PHROG provides multiple sequence alignments (MSAs) to construct hidden Markov models (HMMs), which can be used to classify protein sequences into PHROGs.

Using the `pyhmmer` library, we convert these MSAs into HMMs. Each HMM is run against a protein sequence, producing an e-value. This e-value serves as a threshold to determine whether a sequence is classified under a specific PHROG.

`pyhmmer` offers two modes: "scan" and "search." In "scan" mode, the inner loop iterates over HMMs, returning the best HMM for each sequence. In "search" mode, the loop iterates over sequences, returning the best sequence for each HMM. While "scan" is more intuitive for our use case, it is computationally expensive. For optimization, we chose "search" mode despite requiring additional post-processing.

The results from `pyhmmer search` are saved into a TSV file with columns: `query_HMM_name`, `target_sequence_name`, and `e_value`. A single sequence can match multiple PHROGs. To retain only the best hit, we load the file into a pandas DataFrame, sort by e-value, and remove duplicates, keeping the entry with the lowest e-value.

We set a strict e-value cutoff of 10^{-5} to accept a match as valid. From a corpus of 2,484,471 sequences, 403,804 remain unclassified using this threshold—approximately 16%. To mitigate this and provide more context to the model, we also retain hits with higher e-values, but assign them new identifiers (e.g., `dummy_phrog_X`) to prevent confusing weak matches with true PHROG assignments.

This strategy reduces the number of unclassified sequences to 169,627 and introduces 234,177 dummy PHROGs. The final vocabulary consists of 65,232 unique terms (true and dummy PHROGs), up from 38,880 without dummy PHROGs.

3.2 Corpus Creation

Explain how the dataset or corpus was assembled and preprocessed.

3.3 Skipgram Word2Vec

In order to convert protein sequences into meaningful vector representations, we employ the Skip-gram Word2Vec algorithm. The core idea is to train a model which, given a target word w_t, predicts its surrounding context words. To achieve optimal predictions, the model must learn vector embeddings that capture semantic information. Once training is complete, these learned vectors serve as embeddings for downstream tasks.

More formally, given a sequence of training words $w_1, w_2, \ldots, w_T$, the objective is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq cj\neq 0}\log P(w_{t+j} \mid w_t), \tag{1}$$

where:

- T is the total number of words in the training sequence,
- c is the context window size (e.g., 2 or 5 words on each side),
- w_t is the center (target) word,
- w_{t+j} are the context words,
- $P(w_{t+j} \mid w_t)$ is the probability of observing w_{t+j} given w_t.

The Skip-gram model is a shallow neural network with one hidden layer. Its architecture comprises:

- **Input Layer**: A one-hot vector of size V (vocabulary size) representing w_t,
- **Hidden Layer**: Weight matrix $W \in R^{V\times N}$, where N is the embedding dimension. This layer learns the word vectors,
- **Output Layer**: Weight matrix $W' \in R^{N\times V}$ projecting from embedding space back to the vocabulary space.

No activation functions are applied between layers; instead, a softmax is used at the output to compute probabilities.

The key hyperparameters of the Skip-gram model are the window size c and the embedding dimension N. To select optimal values, we trained multiple models with different (c, N) combinations. We then evaluated each embedding set by training a simple neural network classifier for gene category prediction: better embeddings yield superior classification performance.

Empirically, the best embeddings were obtained using a context window size $c = 3$ and embedding dimension $N = 32$.

3.4 Neural Network Classifier

The first application of our embeddings is to predict PHROG annotations. Given the complexity of this task, we employ transfer learning to boost performance. Specifically:

1. **Pre-training**: The model is initially trained on a simpler task—predicting one of approximately 10 gene categories—using the same architecture.
2. **Fine-tuning**: We then fine-tune the pre-trained model on the primary task of PHROG annotation prediction.
3. **Baseline**: For comparison, a separate model is trained from scratch on the PHROG annotation task without transfer learning.

Our network architecture is as follows:

- **Input Layer**: Dimensionality equal to the embedding size ($N = 32$),
- **Hidden Layer**: Single layer with 64 units,
- **Output Layer**: Size equal to the number of classes (dynamically set for category or annotation task).

Other components:

- **Non-linearity**: ReLU,
- **Regularization**: Dropout and entropy regularization during training,
- **Normalization**: BatchNorm1d on hidden activations,
- **Output**: Raw class logits,
- **Loss Function**: Cross-Entropy Loss defined as

$$\mathcal{L}(y, \hat{y}) = -\sum_{k=1}^{C} y_k \log(\hat{y}_k), \tag{2}$$

where C is the number of classes, y_k is the true one-hot label, and $\hat{y}_k$ is the softmax probability for class k.

3.5 Clustering

The initial cluster algorithm used was dbscan which had to be discarded because of terrible results it provided, so agglomerative clustering was the algorithm used to cluster the embeddings, parameters chosen where, the distance metric was cosine distance, with a threshold of 0.6.

4 Results

4.1 Neural Network Classifier Results

Results from both category and annotation classifiers show relatively low accuracy and precision. This was expected given the complexity of the task, limited data, and the lack of homology-based information — the only available cues stem from gene synteny rather than protein structure or function.

One advantage of using neural networks is the availability of logits (raw prediction scores), which can be converted to probabilities using a normalization function like softmax. This allows filtering for high-confidence predictions. While full annotation remains difficult, the model may still offer reliable predictions for some protein groups.

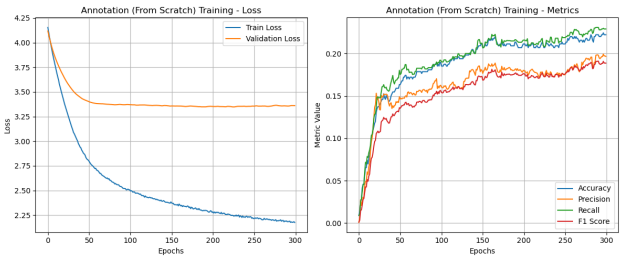

Fig. 1: Training history for annotation model (from scratch)

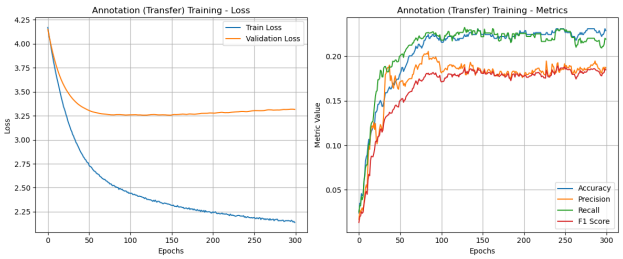

Fig. 2: Training history for annotation model (transfer learning)

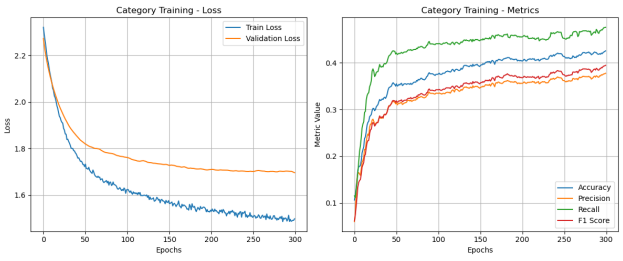

Fig. 3: Training history for category classifier

To convert the logits (raw model outputs) into probabilities, we use the softmax function:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z_i is the logit for class i, and K is the total number of classes.

To evaluate the reliability of these predicted probabilities, we use a reliability diagram:

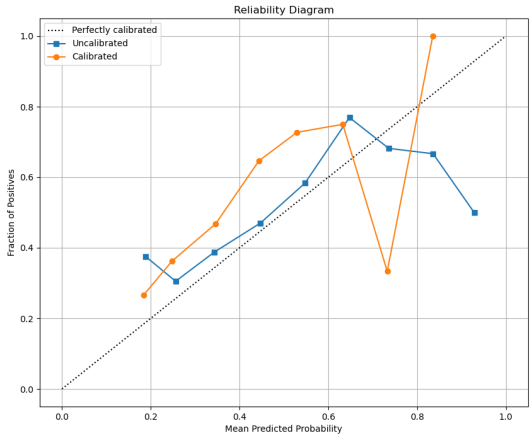

Fig. 4: Reliability diagram for predicted annotation probabilities

4.2 Clustering Results

After performing agglomerative clustering, we can begin evaluating whether our hypothesis holds. Specifically, we expect that proteins with similar functionality — such as antitermination protein Q — cluster together. In this analysis, phrogs `127`, `304`, and `79` were of particular interest.

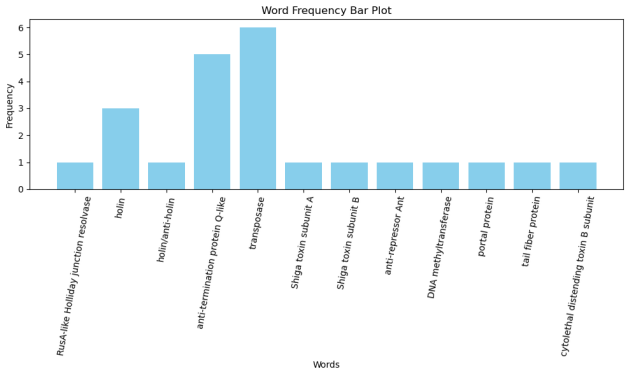

Fig. 5: Distribution of protein types within the cluster containing phrog 127 and 79

This suggests that the cluster is heterogeneous. Despite this, two protein types are notably more prevalent — antitermination protein Q-like and transposase — which implies that the learned embeddings capture at least some degree of functional similarity.

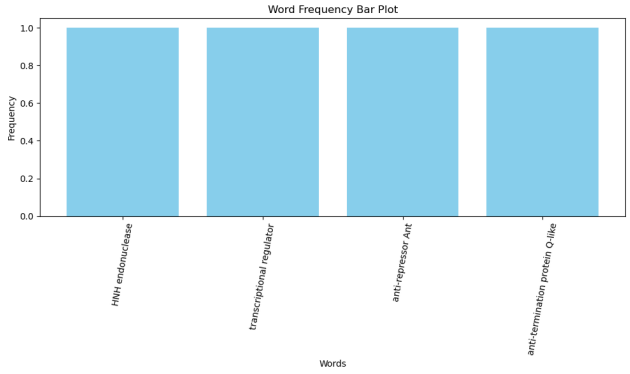

Fig. 6: Functional annotation of the small cluster containing phrog 304

Phrog 304 falls into a much smaller cluster of only four annotated phrogs. Three out of four proteins in this cluster are transcription regulation-related, suggesting some degree of internal consistency. However, the small size makes it difficult to draw stronger conclusions.

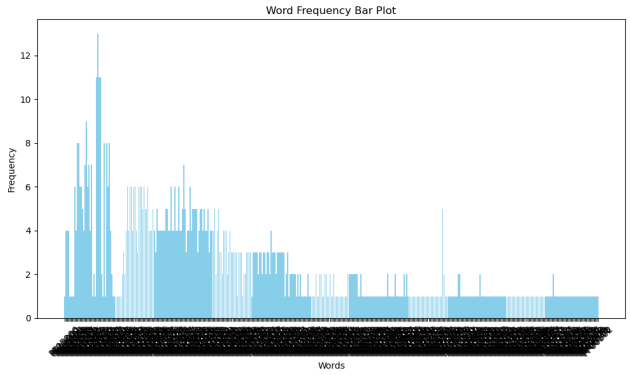

Fig. 7: Pfam domain distribution within the cluster of phrog 127 and 79

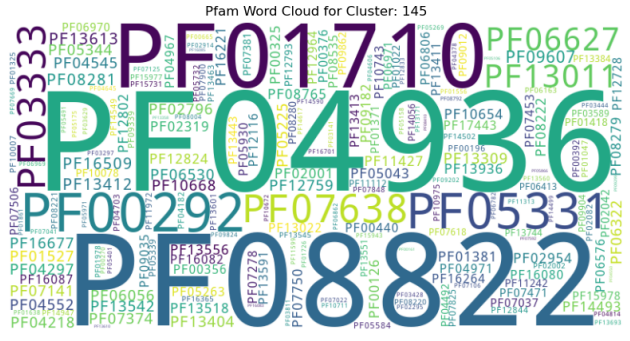

Fig. 8: Pfam domain word cloud for the same cluster

While many Pfam domains appear only once, several are repeated — indicating some recurring structural or functional motifs. This supports the notion that embeddings reflect partial biological relevance.

Finally, we used clustering to identify entirely homogeneous groups. These may suggest new annotations for currently unannotated phrogs. Two notable examples include:

Cluster 853 (HNH endonuclease):

- 824 — HNH endonuclease
- 5017 — HNH endonuclease
- 13789 — HNH endonuclease
- 17787 — HNH endonuclease

Cluster 1194 (Virion structural protein):

- 5222 — virion structural protein
- 5223 — virion structural protein
- 5269 — virion structural protein
- 5273 — virion structural protein

These homogeneous clusters are strong candidates for functional inference and could guide future experimental annotation.

5 Conclusions

This project explored the feasibility of applying natural language processing techniques—particularly Word2Vec embeddings based on gene synteny—to the functional annotation of bacteriophage genomes. While the approach demonstrated conceptual promise and yielded some encouraging clustering results (e.g., grouping of HNH endonucleases and virion proteins), the overall classifier performance remained modest. The challenges stem largely from limited training data, the absence of strong homology signals, and the inherent complexity of the task.

Despite these limitations, the results suggest that synteny-informed embeddings do capture partial biological relevance. However, current models are insufficiently accurate for robust genome-wide annotation. Future work should aim to expand the dataset and refine the embedding methodology to improve predictive power and functional resolution.

Acknowledgments

I would like to thank my tfg suprevisor Ivan Erill Sagales, who has been extremely helpful and dedicated all the time and whithout whom this project could not be possible.

References

[1] A. Harbola, D. Negi, M. Manchanda, and R. K. Kesharwani, “Chapter 27 - Bioinformatics and biological data mining,” in *Bioinformatics*, D. B. Singh and R. K. Pathak, Eds., Academic Press, 2022, pp. 457–471. doi: https://doi.org/10.1016/B978-0-323-89775-4.00019-510.1016/B978-0-323-89775-4.00019-5.

[2] National Library of Medicine, “Homology: Orthologs and Paralogs.” [Online]. Available: https://www.nlm.nih.gov/ncbi/workshops/2023-08$_BLAST_evol/ortho_para.html. Accessed$: $Jun.28, 2025$.

[3] P. Myers, “Synteny: Inferring Ancestral genomes,” *Nat. Educ.*, vol. 1, no. 1, p. 47, 2008.

[4] Z. Yin, J. G. Bird, J. T. Kaelber, B. E. Nickels, and R. H. Ebright, "In transcription antitermination by Q, NusA induces refolding of Q to form a nozzle that extends the RNA polymerase RNA-exit channel," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 119, no. 33, p. e2205278119, Aug. 2022. doi: https://doi.org/10.1073/pnas.220527811910.1073/pnas.2205278119.

[5] P. Terzian *et al.*, "PHROG: families of prokaryotic virus proteins clustered using remote homology," *NAR Genomics Bioinformatics*, vol. 3, no. 3, p. lqab067, Jun. 2021. doi: https://doi.org/10.1093/nargab/lqab06710.1093/nargab/lqab067.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv:1301.3781, Sep. 2013. doi: https://doi.org/10.48550/arXiv.1301.378110.48550/arXiv.1301.3781.

[7] D. Miller, A. Stern, and D. Burstein, "Deciphering microbial gene function using natural language processing," *Nat. Commun.*, vol. 13, no. 1, p. 5731, Sep. 2022. doi: https://doi.org/10.1038/s41467-022-33397-410.1038/s41467-022-33397-4.

[8] M. Larralde and G. Zeller, "PyHMMER: a Python library binding to HMMER for efficient sequence analysis," May 04, 2023. [Online]. Available: https://pyhmmer.readthedocs.io/en/stable/examples/performance$_t ips.html. Accessed: Jun. 28, 2025.$