Neural Sequence Distance Embeddings

Anonymous Author(s)
Affiliation
Address
email

Abstract

The development of data-dependent heuristics and representations for biological sequences that reflect their evolutionary distance is critical for large-scale biological research. However, popular machine learning approaches, based on continuous Euclidean spaces, have struggled with the discrete combinatorial formulation of the edit distance that models evolution and the hierarchical relationship that characterises real-world datasets. We present Neural Sequence Distance Embeddings (`NeuroSEED`), a general framework to embed sequences in geometric vector spaces, and illustrate the effectiveness of the hyperbolic space that captures the hierarchical structure and provides an average 38% reduction in embedding RMSE against the best competing geometry. The capacity of the framework and the significance of these improvements are then demonstrated devising supervised and unsupervised `NeuroSEED` approaches to multiple core tasks in bioinformatics. Benchmarked with common baselines, the proposed approaches display significant accuracy and/or runtime improvements on real-world datasets. As an example for hierarchical clustering, the proposed pretrained and from-scratch methods match the quality of competing baselines with 30x and 15x runtime reduction, respectively.

1 Introduction

Over the course of evolution, biological sequences constantly mutate and a large part of biological research is based on the analysis of these mutations. Biologists have developed accurate statistical models to estimate the evolutionary distance between pairs of sequences based on their edit distance $D(s_1, s_2)$: the minimum number of (weighted) insertions, deletions or substitutions required to transform a string s_1 into another string s_2.

However, the computation of this edit distance kernel D with traditional methods is bound to a quadratic complexity and hardly parallelizable, making its computation a bottleneck in large scale analyses, such as microbiome studies [1, 2, 3]. Furthermore, the accurate computation of similarities among multiple sequences, at the foundation of critical tasks such as hierarchical clustering and multiple sequence alignment, is computationally intractable even for relatively small numbers of sequences. Problems that in other spaces are relatively simple become combinatorially hard in the space of sequences defined by the edit distance. For example, finding the Steiner string, a classical problem in bioinformatics that can be thought of as computing the geometric median in the space of sequences, is NP-complete.

Classical algorithms and heuristics [4, 5, 6, 7] widely used in bioinformatics for these tasks are data-independent and, therefore, cannot exploit the low-dimensional manifold assumption that characterises real-world data [8, 9, 10]. Leveraging the available data to produce efficient and data-dependent heuristics and representations would greatly accelerate large-scale analyses that are critical to biological research.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

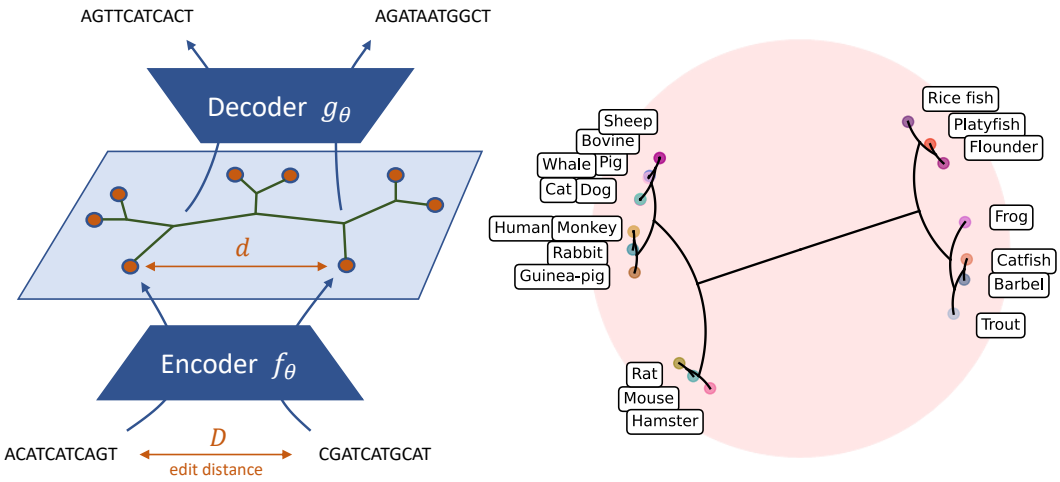

Figure 1: On the left, the key idea of NeuroSEED: learn an encoder function f_θ that preserves distances between the sequence and vector space (D and d). The vector space can then be used to study the relationship between sequences and, potentially, decode new ones. On the right, an example of the *hierarchical clustering* produced on the Poincaré disk. The data was downloaded from UniProt [14] and consists of the P53 tumour protein from 20 different organisms.

While the number of available biological sequences has grown exponentially over the past decades, machine learning approaches to problems related to string matching [11, 12] have not been adopted widely in bioinformatics due to their limitation in accuracy and speed. In contrast to most tasks in computer vision and NLP, string matching problems are typically formalised as combinatorial optimisation problems. These discrete formulations do not fit well with the current deep learning approaches. Moreover, representation learning methods based on Euclidean spaces struggle to capture the hierarchical structure that characterises real-world biological datasets due to evolution. Finally, common self-supervised learning approaches, very successful in NLP, are less effective in the biological context where relations tend to be between sequences rather than between bases [13].

In this work, we present Neural Sequence Distance Embeddings (NeuroSEED), a general framework to produce representations for biological sequences where the distance in the embedding space is correlated with the evolutionary distance D between sequences. NeuroSEED provides fast approximations of the distance kernel D, low-dimensional representations for biological sequences, tractable analysis of the relationship between multiple sequences in the embedding geometry and a way to decode novel sequences.

Firstly, we reformulate several existing approaches into NeuroSEED highlighting their contributions and limitations. Then, we examine the task of embedding sequences to preserve the edit distance that is the basis of the framework. This analysis reveals the importance of data-dependent approaches and of using a geometry that matches the underlying data distribution well. The hyperbolic space is able to capture the implicit hierarchical structure given by biological evolution and provides an average 38% reduction in embedding RMSE against the best competing geometry.

We show the potential of the framework and its wide applicability by analysing two fundamental tasks in bioinformatics involving the relations between multiple sequences: *hierarchical clustering* and *multiple sequence alignment*. For both tasks, unsupervised approaches using NeuroSEED encoders are able to match the accuracy of common heuristics while being orders of magnitude faster. For *hierarchical clustering*, we also explore a method based on the continuous relaxation of Dasgupta's cost in the hyperbolic space which provides a 15x runtime reduction at similar quality levels. Finally, for *multiple sequence alignment*, we devise an original approach based on variational autoencoders that matches the performance of competitive baselines while significantly reducing the runtime complexity.

As a summary our contributions are: (*i*) We introduce NeuroSEED, a general framework to map sequences in geometric vector spaces, and reformulate existing approaches into it. (*ii*) We show how the hyperbolic space can bring significant improvements to the data-dependent analysis of biological

sequences. (*iii*) We propose several heuristic approaches to classical bioinformatics problems that can be constructed on top of `NeuroSEED` embeddings and provide significant running time reduction against classical baselines.

2 Bioinformatics tasks

The field of bioinformatics has developed a wide range of algorithms to tackle the classical problems that we explore. Here we present the tasks and briefly mention their motivation and some of the baselines we test. More details are provided in Appendix B.

Edit distance approximation In this work, we always deal with the classical edit distance where the same weight is given to every string operation, but all the approaches developed can be applied to any distance function of choice (which is given as an oracle). As baseline heuristic, we take k-mer, which is the most commonly used alignment-free method and represents sequences by the frequency vector of subsequences of a certain length.

Hierarchical clustering (HC) Discovering the intrinsic hierarchical structure given by evolutionary history is a critical step of many biological analyses. Hierarchical clustering (HC) consists of, given a pairwise distance function, defining a tree with internal points corresponding to clusters and leaves to datapoints. Dasgupta's cost [15] measures how well the tree generated respects the similarities between datapoints. As baselines we consider classical agglomerative clustering algorithms (Single [16], Complete [17] and Average Linkage [6]) and the recent technique [18] that uses a continuous relaxation of Dasgupta's cost in the hyperbolic space.

Multiple sequence alignment (MSA) Aligning three or more sequences is used for the identification of active and binding sites as well as conserved protein structures, but finding its optimal solution is NP-complete. A related task to MSA is the approximation of the Steiner string which minimises the sum of the distances (consensus error) to the sequences in a set.

Datasets To evaluate the heuristics we chose two datasets containing different portions of the 16S rRNA gene, crucial in microbiome analysis [19], one of the most promising applications of our approach. The first, Qiita [19], contains more than 6M sequences of up to 152 bp that cover the V4 hyper-variable region. The second, RT988 [11], has only 6.7k publicly available sequences of length up to 465 bp covering the V3-V4 regions. Both datasets were generated by Illumina MiSeq [20] and contain sequences of approximately the same length. Qiita was collected from skin, saliva and faeces samples, while RT988 from oral plaques. Moreover, we used a dataset of synthetically generated sequences to test the importance of data-dependent approaches. A full description of the data splits for each of the tasks is provided in Appendix B.4.

3 Neural Sequence Distance Embeddings

The underlying idea behind the `NeuroSEED` framework, represented in Figure 1, is to map sequences in a continuous space so that the distance between embedded points is correlated to the one between sequences. Given a distribution of sequences and a distance function D between them, any `NeuroSEED` approach is formed by four main components: an embedding geometry, an encoder model, a decoder model, and a loss function.

Embedding geometry The distance function d between the embedded points defines the geometry of the embedding space. While this factor has been mostly ignored by previous work [11, 21, 22, 23, 24], we show that it is critical for this geometry to reflect the relations between the sequences in the domain. In our experiments, we provide a comparison between Euclidean, Manhattan, cosine, squared Euclidean (referred to as Square) and hyperbolic distances (details in Appendix D).

Encoder model The encoder model f_θ maps sequences to points in the embedding space. In this work we test a variety of models as encoder functions: linear layer, MLP, CNN, GRU [25] and transformer [26] with local and global attention. The details on how the models are adapted to the sequences are provided in Appendix C. Chen *et al.* [21] proposed CSM, an encoder architecture based on the convolution of subsequences. However, as also noted by Koide *et al.* [12], this model does not perform well when various layers are stacked and, due to the interdependence of cells in the dynamic programming routine, it cannot be efficiently parallelised on GPU.

Decoder model For some tasks it is also useful to decode sequences from the embedding space. This idea, employed in Section 7.2 and novel among the works related to NeuroSEED, enables to apply the framework to a wider set of problems.

Loss function The simplest way to train a NeuroSEED encoder is to directly minimise the MSE between the sequences' distance and its approximation as the distance between the embeddings:

$$L(\theta, S) = \sum_{s_1, s_2 \in S} \left(D(s_1, s_2) - \alpha \, d(f_\theta(s_1), f_\theta(s_2))\right)^2 \quad (1)$$

where α is a constant or learnable scalar. Depending on the application that the learned embeddings are used for, the MSE loss may be combined or substituted with other loss functions such as the triplet loss for *closest string retrieval* (Appendix F), the relaxation of Dasgupta's cost for *hierarchical clustering* (Section 7.1) or the sequence reconstruction loss for *multiple sequence alignment* (Section 7.2).

There are at least five previous works [11, 21, 22, 23, 24] that have used approaches that can be described using the NeuroSEED framework. These methods, summarised in Table 1, show the potential of approaches based on the idea of NeuroSEED, but share two significant limitations. The first is the lack of analysis of the geometry of the embedding space, which we show to be critical. The second is that the range of tasks is limited to *edit distance approximation* and *closest string retrieval*. We highlight how this framework has the flexibility to be adapted to significantly more complex tasks involving relations between multiple sequences such as *hierarchical clustering* and *multiple sequence alignment*.

Table 1: Summary of the previous and the proposed NeuroSEED approaches. EDA stands for edit distance approximation and CSR for closest string retrievals. For our experiments, in the columns geometry and encoder we report those that performed best among the ones tested.

Method	Geometry	Encoder	Decoder	Loss	Tasks
Zheng *et al.* [11]	Jaccard	CNN	✗	MSE	EDA
Chen *et al.* [21]	Cosine	CSM	✗	MSE	EDA
Zhang *et al.* [22]	Euclidean	GRU	✗	MAE + triplet	EDA & CSR
Dai *et al.* [23]	Euclidean	CNN	✗	MAE + triplet	EDA & CSR
Gomez *et al.* [24]	Square	CNN	✗	MSE	EDA & CSR
Section 5	Hyperbolic	CNN & transformer	✗	MSE	EDA
Section 6	Hyperbolic	CNN & transformer	✗	MSE	HC & MSA
Section 7.1	Hyperbolic	Linear	✗	Relaxed Dasgupta	HC
Section 7.2	Cosine	Linear	✓	MSE + reconstr.	MSA
Appendix F	Hyperbolic	CNN & transformer	✗	MSE & triplet	CSR

4 Related work

Hyperbolic embeddings Hyperbolic geometry is a non-Euclidean geometry with constant negative sectional curvature and is often referred to as a continuous version of a tree for its ability to embed trees with arbitrarily low distortion. The advantages of mapping objects with implicit or explicit hierarchical structure in the hyperbolic space have also been shown in other domains [27, 28, 29, 10]. In comparison, this work deals with a very different space defined by the edit distance in biological sequences and, unlike most of the previous works, we do not only derive embeddings for a particular set of datapoints, but train an encoder to map arbitrary sequences from the domain in the space.

Sequence Distance Embeddings The clear advantage of working in more computationally tractable spaces has motivated significant research in *Sequence Distance Embeddings* [30] (also known as *low-distortion embeddings*) approaches to variants of the edit distance [31, 32, 33, 34, 35, 36, 37]. However, they are all *data-independent* and have shown weak performance on the 'unconstrained' edit distance.

Hashing and metric learning NeuroSEED also fits well into the wider research on *learning to hash* [38], where the goal is typically to map a high dimensional vector space into a smaller one preserving distances. Finally, another field related to NeuroSEED is *metric learning* [39, 40], where models are trained to learn embeddings from examples of similar and dissimilar pairs.

5 Edit distance approximation

In this section we test[1] the performance of different encoder models and distance functions to preserve an approximation of the edit distance in the `NeuroSEED` framework trained with the MSE loss. To make the results more interpretable and comparable across datasets, we report results using % RMSE:

$$\%\ \text{RMSE}(\theta, S) = \frac{100}{n}\sqrt{L(\theta, S)} = \frac{100}{n}\sqrt{\sum_{s_1, s_2 \in S}(ED(s_1, s_2) - n\ d(f_\theta(s_1), f_\theta(s_2)))^2} \quad (2)$$

where n is the maximum sequence length. This can be interpreted as an approximate average error in the distance prediction as a percentage of the size of the sequences.

	RT988					Qiita				
Model	**Cosine**	**Euclidean**	**Square**	**Manhattan**	**Hyperbolic**	**Cosine**	**Euclidean**	**Square**	**Manhattan**	**Hyperbolic**
2-mer	7.782	4.927	8.000	5.036	4.859	21.222	11.752	30.453	11.639	10.481
3-mer	3.392	3.351	3.520	2.987	3.308	12.352	7.962	32.219	7.439	6.657
4-mer	1.790	3.314	1.899	2.318	3.294	6.006	7.015	34.098	5.636	6.728
5-mer	1.409	3.449	1.422	1.801	3.470	5.027	7.638	34.559	5.391	7.600
6-mer	1.471	3.710	1.450	1.686	3.730	5.723	8.383	34.616	5.844	8.275
Linear	0.62±0.03	21.36±7.07	27.28±10.89	-	0.51±0.01	3.38±0.06	4.39±0.09	5.83±0.21	3.82±0.09	2.50±0.01
MLP	1.57±0.16	1.10±0.05	6.78±2.50	1.01±0.04	0.59±0.20	4.98±0.11	4.36±0.19	8.52±0.78	4.92±0.10	1.85±0.02
CNN	0.69±0.03	0.58±0.05	2.95±1.09	0.98±0.06	0.59±0.01	2.54±0.04	2.68±0.05	5.03±0.85	4.06±0.21	**1.56±0.01**
GRU	14.90±4.56	1.10±0.11	1.96±0.47	1.13±0.15	2.56±3.33	-	3.30±0.06	5.52±0.15	3.74±0.01	2.60±0.16
Global T.	0.49±0.01	0.52±0.01	0.88±0.02	**0.44±0.01**	**0.46±0.01**	2.61±0.01	2.10±0.05	3.71±0.04	2.57±0.11	1.83±0.03
Local T.	0.51±0.03	0.57±0.00	0.58±0.02	0.48±0.01	**0.45±0.01**	2.67±0.04	2.42±0.02	3.72±0.06	2.46±0.02	1.86±0.02

Best

Worst

Figure 2: % RMSE test set results (4 runs). The first five models are the k-mer baselines, each k-mer has an embedding dimension of 4^k. The remaining models all have an embedding space dimension of 128. In all the tables: T. stands for transformer, - indicates that the model did not converge, **bold** the best results and the green-to-white colour scale the range of results best-to-worst.

Data-dependent vs data-independent methods Figures 2 and 3 show that, across the datasets and the distance functions, the data-dependent models learn significantly better representations than data-independent baselines. The main reason for this is their ability to focus on and dedicate the embedding space to a manifold of much lower dimensionality than the complete string space. This observation is further supported by the results in Appendix E, where the same models are trained on synthetic random sequences and the data-independent baselines are able to better generalise to the test set. The results in the RT988 dataset are lower because its sequences contain not only the hyper-variable regions but also conserved regions for which distances are low.

Our analysis also confirms the results from Zheng *et al.* [11] and Dai *et al.* [23] which showed that convolutional models outperform feedforward and recurrent models. We also show that transformers, even when with local attention, produce, in many cases, better representations. Attention could provide significant advantages when considering more complex definitions of distance that include, for example, inversions [41], duplications and transpositions [42].

Hyperbolic space The most interesting and novel results come from the analysis of the geometry of the embedding space. In these biological datasets, there is an implicit hierarchical structure that is well reflected by the hyperbolic space. Thanks to this close correspondence even relatively simple models perform very well with the hyperbolic distance. In convolutional and attention models, the hyperbolic space provides a 38% average RMSE reduction against the best competing geometry for each model.

The benefit of using the hyperbolic space is clear when analysing the dimension required (Figure 4). The hyperbolic space provides significantly more efficient embeddings, with the model reaching the 'elbow' at dimension 32 and matching the performance of the other spaces with dimension 128 with only 4 to 16. Given that the space to store the embeddings and the time to compute distances between them scale linearly with the dimension, this provides a significant improvement in downstream tasks over other `NeuroSEED` approaches.

[1] Code, datasets and tuned hyperparameters can be found at https://anonymous.4open.science/r/NeuroSEED.

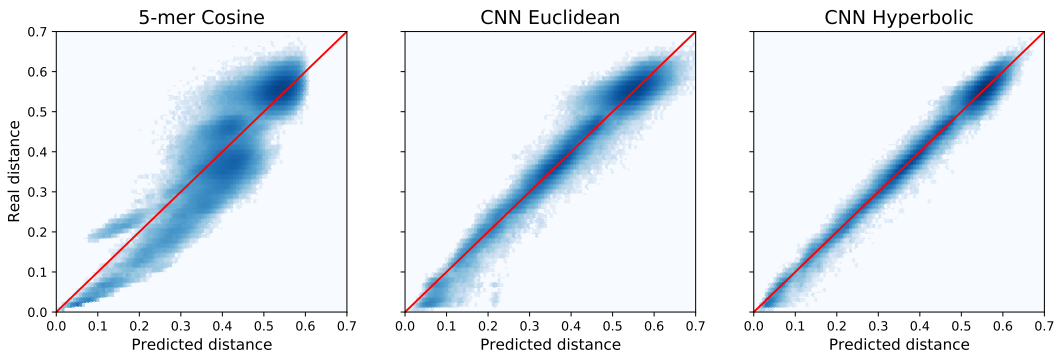

Figure 3: Qualitative comparison in the Qiita dataset between the best performing baseline (5-mer with cosine distance) and the CNN in the Euclidean and hyperbolic space. For every test set sequence pair, predicted vs real distances are plotted, the darkness represents the density of points. The CNN model follows much more tightly the red line of the oracle across the whole range of distances in the hyperbolic space.

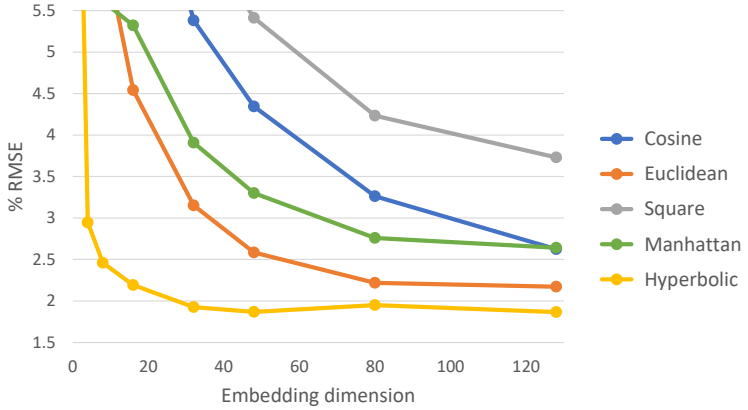

Figure 4: *Edit distance approximation* % RMSE on Qiita dataset for a global transformer with different distance functions.

Running time Computing the pairwise distance matrix of a set of sequences is a critical step of many algorithms including the ones analysed in the next section. Taking as an example 6700 sequences from the RT988 dataset, optimised C code computes on a CPU approximately 2700 pairwise distances per second and takes 2.5 hours for the whole matrix. In comparison, using a trained `NeuroSEED` model, the same matrix can be approximated in 0.3-3s on the same CPU (similar value for the k-mer baseline). The computational complexity for N sequences of length M is reduced from $O(N^2\ M^2/\log M)$ to $O(N(M+N))$ assuming constant embedding size and a model linear with respect to the sequence length. The training process takes typically 0.5-3 hours on GPU. However, in applications such as microbiome analysis, biologists typically analyse data coming from the same distribution (e.g. the 16S rRNA gene) for multiple individuals, therefore the initial cost would be significantly amortised.

6 Unsupervised heuristics

In this section, we show how competitive heuristics for *hierarchical clustering* and *multiple sequence alignment* can be built on the low-distortion embeddings produced by the models trained in the previous section.

Hierarchical clustering Agglomerative clustering, the most commonly used class of HC algorithms, can be accelerated when run directly on `NeuroSEED` embeddings produced by the pretrained model. This reduces the complexity to generate the pairwise distance matrix from $O(N^2M^2/\log M)$

to $O(N(M+N))$ and allows to accelerate the clustering itself using geometric optimisations like locality-sensitive hashing.

We test models with no further tuning from Section 5 on a dataset of 10k unseen sequences from the Qiita dataset. The results (Figure 5) show that there is no statistical difference in the quality of the hierarchical clustering produced with ground truth distances compared to that with convolutional or attention hyperbolic `NeuroSEED` embeddings. Instead, the difference in Dasgupta's cost between different architectures and geometries is statistically significant and it results in a large performance gap when these trees are used for tasks such as MSA shown below. The total CPU time taken to construct the tree is reduced from more than 30 minutes to less than one in this dataset and the difference gets significantly larger when scaling to datasets of more and longer sequences.

Baselines	
Single L.	0.628
Complete L.	0.479
Average L.	**0.000**

Model	Cosine	Euclidean	Square	Manhattan	Hyperbolic
4-mer	0.261	0.260	0.242	0.191	0.299
Linear	0.062±0.007	0.172±0.036	0.153±0.037	0.177±0.026	0.028±0.005
MLP	0.169±0.054	0.095±0.021	0.289±0.094	0.178±0.029	0.035±0.004
CNN	0.028±0.003	0.030±0.004	0.067±0.022	0.081±0.047	**-0.004±0.015**
GRU	-	0.042±0.006	0.068±0.010	0.069±0.015	0.066±0.043
Global T.	0.032±0.014	**0.003±0.008**	0.038±0.005	**0.002±0.003**	**0.000±0.006**
Local T.	0.035±0.003	0.022±0.008	0.034±0.005	0.022±0.003	**0.000±0.007**

Figure 5: Average Linkage % increase in Dasgupta's cost of `NeuroSEED` models compared to the performance of clustering on the ground truth distances, ubiquitously used in bioinformatics. Average Linkage was the best performing clustering heuristic across all models.

Multiple sequence alignment Clustal, the most popular MSA heuristic, is formed by a phylogenetic tree estimation phase that produces a guide tree then used by a progressive alignment phase to compute the complete alignment. However, the first of the two phases, based on hierarchical clustering, is typically the bottleneck of the algorithm. On 1200 RT988 sequences (used below), the construction of the guide tree takes 35 minutes compared to 24s for the progressive alignment. Therefore, it can be significantly accelerated using `NeuroSEED` heuristics to generate matrix and guide tree. In these experiments, we construct the tree running the Neighbour Joining algorithm (NJ) [43] on the `NeuroSEED` embeddings and then pass it on the Clustal command-line routine that performs the alignment and returns an alignment score.

Again, the results reported in Figure 6 show that the alignment scores obtained when using the `NeuroSEED` heuristics with attention models are not statistically different from those obtained with the ground truth distances. Most of the models also show a relatively large variance in performance across different runs. This has positive and negative consequences: the alignment obtained using a single run may not be very accurate, but, by training an ensemble of models and applying each of them, we are likely to obtain a significantly better alignment than the baseline while still only taking a fraction of the time.

Model	Cosine	Euclidean	Hyperbolic
Linear	60.6±35.1	111.3±3.6	57.5±22.0
MLP	72.3±11.8	53.6±3.1	-11.7±18.9
CNN	31.0±16.2	4.7±9.7	**-16.3±16.1**
Global T.	39.4±74.3	1.9±3.8	31.1±21.8
Local T.	31.9±30.5	8.6±14.1	**-20.1±7.3**

Figure 6: Percentage improvement (average of 3 runs) in the alignment cost (the lower the better) returned by Clustal when using the heuristics to generate the tree as opposed to its default setting using NJ on real distances.

7 Supervised heuristics

In this section we propose and evaluate two methods to adapt `NeuroSEED` to the tasks of *hierarchical clustering* and *multiple sequence alignment* with tailored loss functions.

7.1 Relaxed approach to hierarchical clustering

An alternative approach to *hierarchical clustering* uses the continuous relaxation of Dasgupta's cost [18] as loss function to embed sequences in the hyperbolic space. In comparison to Chami *et al.* [18], we show that it is possible to significantly decrease the number of pairwise distances required by directly mapping the sequences into the space. This allows to considerably accelerate the construction, especially when dealing with a large number of sequences without requiring any pretrained model. Figure 1 shows an example of the relaxed approach when applied to a small dataset of proteins.

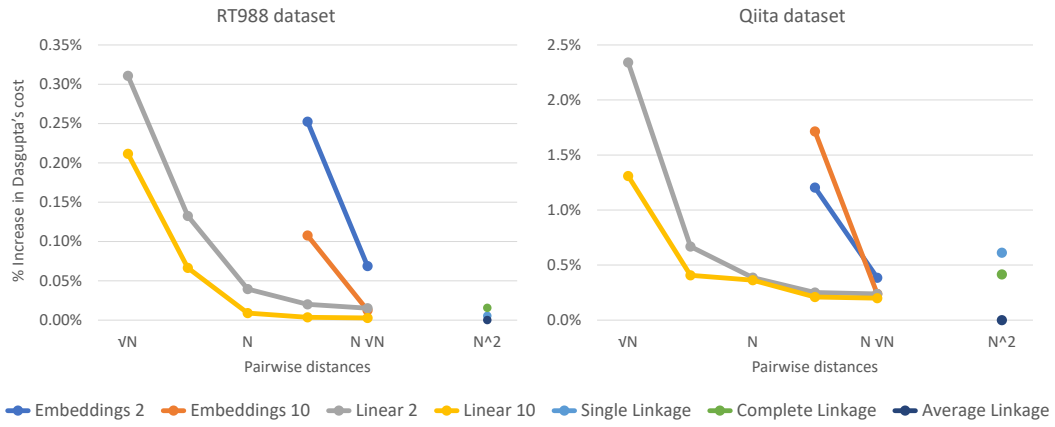

Figure 7: Average Dasgupta's cost of the various approaches with respect to the number of pairwise distances used in the RT988 and Qiita datasets. The performances are reported as the percentage increase in cost compared to the one of the Average Linkage (best performing). Embedding refers to the baseline [18] while Linear to the relaxed `NeuroSEED` approach. The attached number represents the dimension of the hyperbolic space used.

The results, plotted in Figure 7, show that a simple linear layer mapping sequences to the hyperbolic space is capable of obtaining with only N pairwise distances very similar results to those from agglomerative clustering (N^2 distances) and hyperbolic embedding baselines ($N\sqrt{N}$ distances). In the RT988 dataset this corresponds to, respectively, 6700x and 82x fewer labels and leads to a reduction of the total running time from several hours (>2.5h on CPU for agglomerative clustering, 1-4h on GPU for hyperbolic embeddings) to less than 10 minutes on CPU for the relaxed `NeuroSEED` approach (including label generation, training and tree inference) with no pretraining required. Finally, using more complex encoding architectures such as MLPs or CNNs does not provide significant improvements.

7.2 Steiner string approach to multiple sequence alignment

An alternative approach to *multiple sequence alignment* uses a decoder from the vector space to convert the Steiner string approximation problem (Appendix B.3) in a continuous optimisation task.

This method, summarised in Figure 8 and detailed in Appendix G, consists of training an autoencoder to map sequences to and from a continuous space preserving distances using only pairs of sequence at a time (where calculating the distance is feasible). This is achieved by combining in the loss function a component for the latent space edit distance approximation and one for the reconstruction accuracy of the decoder. The first is expressed as the MSE between the edit distance and the vector distance in the latent space. The second consists of the mean element-wise cross-entropy loss of the decoder's outputs with the real sequences. At test time the encoder embeds all the sequences in the set, the geometric median point (minimising the sum of distances in the embedding space) is found with a relatively simple optimisation procedure and then the decoder is used to find an approximation of the Steiner string. During training, Gaussian noise is added to the embedded point in the latent space forcing the decoder to be robust to points not directly produced by the encoder.

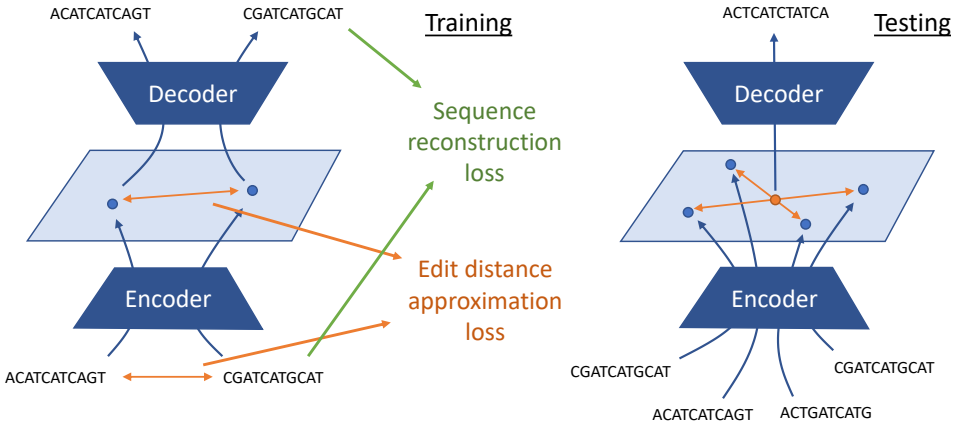

Figure 8: Diagram for the Steiner string approach to *multiple sequence alignment*. On the left, the training procedure using pairs of sequences and a loss combining edit distance approximation and sequence reconstruction. On the right the extrapolation for the generation of the Steiner string by decoding the geometric median in the embedding space.

Baselines	
Random	75.98
Centre	62.52
Greedy-1	**59.43**
Greedy-2	**59.41**

Model	Cosine	Euclidean	Square	Hyperbolic
Linear	**59.41**±0.11	59.96±0.27	60.53±0.49	60.89±0.82
MLP	60.80±0.35	60.00±0.18	59.81±0.22	59.86±0.12
CNN	60.96±0.48	60.20±0.26	60.76±1.09	60.48±0.52

Figure 9: Average consensus error for the baselines (left) and `NeuroSEED` models (right).

As baselines, we report the average consensus error (average distance to the strings in the set) obtained using: a random sequence in the set (random), the centre string of the set (centre) and two competitive greedy heuristics (greedy-1 and greedy-2) proposed respectively by [44] and [45].

The results show that the models consistently outperform the centre string baseline and are close to the performance of the greedy heuristics suggesting that they are effectively decoding useful information from the embedding space. The computational complexity for N strings of size M is reduced from $O(N^2M^2/\log M)$ for the centre string and $O(N^2M)$ for the greedy baselines to $O(NM)$ for the proposed method. Future work could employ models that directly operate in the hyperbolic space [46] to further improve the performance.

8 Conclusion

In this work, we proposed and explored Neural Sequence Distance Embeddings, a framework that exploits the recent advances in representation learning to embed biological sequences in geometric vector spaces. By studying the capacity to approximate the evolutionary edit distance between sequences, we showed the strong advantage provided by the *hyperbolic space* which reflects the biological hierarchical structure.

We then demonstrated the effectiveness and wide applicability of `NeuroSEED` on the problems of *hierarchical clustering* and *multiple sequence alignment*. For each task, we experimented with two different approaches: one unsupervised tying `NeuroSEED` embeddings into existing heuristics and a second based on direct exploitation of the geometry of the embedding space via a tailored loss function. In all cases, the proposed approach performed on par with or better than existing baselines while being significantly faster.

Finally, `NeuroSEED` provides representations that are well suited for human interaction as the embeddings produced can be visualised and easily interpreted. Towards this goal, the very compact representation of hyperbolic spaces is of critical importance [10]. This work also opens many opportunities for future research direction with different types of sequences, labels, architectures and tasks. We present and motivate these directions in Appendix A.

References

[1] Hila Sberro, Brayon J Fremin, Soumaya Zlitni, Fredrik Edfors, Nicholas Greenfield, Michael P Snyder, Georgios A Pavlopoulos, Nikos C Kyrpides, and Ami S Bhatt. Large-scale analyses of human microbiomes reveal thousands of small, novel genes. *Cell*, 2019.

[2] Edoardo Pasolli, Francesca De Filippis, Italia E Mauriello, Fabio Cumbo, Aaron M Walsh, John Leech, Paul D Cotter, Nicola Segata, and Danilo Ercolini. Large-scale genome-wide analysis links lactic acid bacteria from food with the gut microbiome. *Nature communications*, 2020.

[3] Alexander Kurilshikov, Carolina Medina-Gomez, Rodrigo Bacigalupe, Djawad Radjabzadeh, Jun Wang, Ayse Demirkan, Caroline I Le Roy, Juan Antonio Raygoza Garay, Casey T Finnicum, Xingrong Liu, et al. Large-scale association analyses identify host factors influencing human gut microbiome composition. *Nature Genetics*, 2021.

[4] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 1970.

[5] Samuel Kariin and Chris Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends in genetics*, 1995.

[6] Robert R Sokal. A statistical method for evaluating systematic relationships. *Univ. Kansas, Sci. Bull.*, 1958.

[7] Ramu Chenna, Hideaki Sugawara, Tadashi Koike, Rodrigo Lopez, Toby J Gibson, Desmond G Higgins, and Julie D Thompson. Multiple sequence alignment with the Clustal series of programs. *Nucleic acids research*, 2003.

[8] Archit Verma and Barbara E Engelhardt. A robust nonlinear low-dimensional manifold for single cell rna-seq data. *BMC bioinformatics*, 2020.

[9] Richard C Tillquist. Low-dimensional representation of biological sequence data. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2019.

[10] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré maps for analyzing complex hierarchies in single-cell data. *Nature communications*, 2020.

[11] Wei Zheng, Le Yang, Robert J Genco, Jean Wactawski-Wende, Michael Buck, and Yijun Sun. SENSE: Siamese neural network for sequence embedding and alignment-free comparison. *Bioinformatics*, 2019.

[12] Satoshi Koide, Keisuke Kawano, and Takuro Kutsuna. Neural edit operations for biological sequences. In *NeurIPS*, 2018.

[13] Matthew McDermott, Brendan Yap, Peter Szolovits, and Marinka Zitnik. Rethinking relational encoding in language model: Pre-training for general sequences. *arXiv preprint arXiv:2103.10334*, 2021.

[14] UniProt Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 2015.

[15] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proc. Annual ACM SIGACT Symposium on Theory of Computing*, 2016.

[16] Kazimierz Florek, Jan Łukaszewicz, Julian Perkal, Hugo Steinhaus, and Stefan Zubrzycki. Sur la liaison et la division des points d'un ensemble fini. In *Colloquium mathematicum*, 1951.

[17] Thorvald Julius Sørensen. *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons*. 1948.

[18] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In *NeurIPS*, 2020.

[19] Jose C Clemente, Erica C Pehrsson, Martin J Blaser, Kuldip Sandhu, Zhan Gao, Bin Wang, Magda Magris, Glida Hidalgo, Monica Contreras, Óscar Noya-Alarcón, et al. The microbiome of uncontacted amerindians. *Science advances*, 2015.

[20] Rupesh Kanchi Ravi, Kendra Walton, and Mahdieh Khosroheidari. Miseq: a next generation sequencing platform for genomic analysis. *Disease gene identification*, 2018.

[21] Jian Chen, Le Yang, Lu Li, and Yijun Sun. Predicting alignment distances via continuous sequence matching. *bioRxiv*, 2020.

[22] Xiyuan Zhang, Yang Yuan, and Piotr Indyk. Neural embeddings for nearest neighbor search under edit distance. 2019.

[23] Xinyan Dai, Xiao Yan, Kaiwen Zhou, Yuxuan Wang, Han Yang, and James Cheng. Convolutional embedding for edit distance. In *Proc. of SIGIR*, 2020.

[24] Lluís Gómez, Marçal Rusinol, and Dimosthenis Karatzas. Lsde: Levenshtein space deep embedding for query-by-string word spotting. In *ICDAR*, 2017.

[25] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, 2014.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[27] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*, 2017.

[28] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*, 2017.

[29] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In *Proc. of ACL*, 2020.

[30] Graham Cormode. *Sequence distance embeddings*. PhD thesis, Department of Computer Science, University of Warwick, 2003.

[31] S Muthukrishnan and Süleyman Cenk Sahinalp. Approximate nearest neighbors and sequence comparison with block operations. In *Proc. annual ACM symposium on Theory of computing*, 2000.

[32] Graham Cormode, Shan Muthukrishnan, and Süleyman Cenk Sahinalp. Permutation editing and matching via embeddings. In *International Colloquium on Automata, Languages, and Programming*, 2001.

[33] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)*, 2007.

[34] Tugkan Batu, Funda Ergun, and Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *SODA*, 2006.

[35] Hossein Jowhari. Efficient communication protocols for deciding edit distance. In *European Symposium on Algorithms*, 2012.

[36] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM Journal on Computing*, 2012.

[37] Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proc. of STOC*, 2016.

[38] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[39] Brian Kulis et al. Metric learning: A survey. *Foundations and trends in machine learning*, 2012.

[40] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[41] Th Dobzhansky and Alfred H Sturtevant. Inversions in the chromosomes of Drosophila pseudoobscura. *Genetics*, 1938.

[42] Pavel Pevzner. *Computational molecular biology: an algorithmic approach*. 2000.

[43] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 1987.

[44] Ferenc Kruzslicz. Improved greedy algorithm for computing approximate median strings. *Acta Cybernetica*, 1999.

[45] Francisco Casacuberta and M De Antonio. A greedy algorithm for computing approximate median strings. In *Proc. of National Symposium on Pattern Recognition and Image Analysis*, 1997.

[46] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *arXiv preprint arXiv:2101.04562*, 2021.

[47] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *Proc. of ICLR*, 2019.

[48] Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. In *Proc. of ICLR*, 2020.

[49] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *NeurIPS*, 2018.

[50] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *NeurIPS*, 2019.

[51] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 2017.

[52] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

[53] Dan Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. *Acm Sigact News*, 1997.

[54] Phillip Compeau and PA Pevzner. *Bioinformatics algorithms: an active learning approach*. 2018.

[55] William J Masek and Michael S Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 1980.

[56] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. of STOC*, 2015.

[57] Gregory E Sims, Se-Ran Jun, Guohong A Wu, and Sung-Hou Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 2009.

[58] Igor Ulitsky, David Burstein, Tamir Tuller, and Benny Chor. The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 2006.

[59] Chris-Andre Leimeister and Burkhard Morgenstern. Kmacs: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, 2014.

[60] Lusheng Wang and Tao Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1994.

[61] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 2004.

[62] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 2014.

[63] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[64] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[65] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

[66] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, 2015.

[67] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proc. of BMVC*, 2016.

[68] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[69] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014.

[70] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with Poincaré variational auto-encoders. In *NeurIPS*, 2019.

[71] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*. 1994.

[72] Charles George Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 1970.

[73] Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 1970.

[74] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 1970.

[75] David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 1970.

[76] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix A
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix A
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] As mentioned in Section 5 they can all be found in the public repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The tuned hyperparameters can be found in the public repository and the data splits in Appendix B.4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] For every type of experiment we report the approximate running time, the total can be computed adding up all the experiments and lies around 1000h GPU hours (NVIDIA Tesla K80).
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Real-world datasets were taken from [19] and [11] as specified in Section B. Parts of code were adapted from existing repositories, these are clearly specified in the repository.
   (b) Did you mention the license of the assets? [Yes] The copyright notice was copied when present.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Links available from the repository.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]