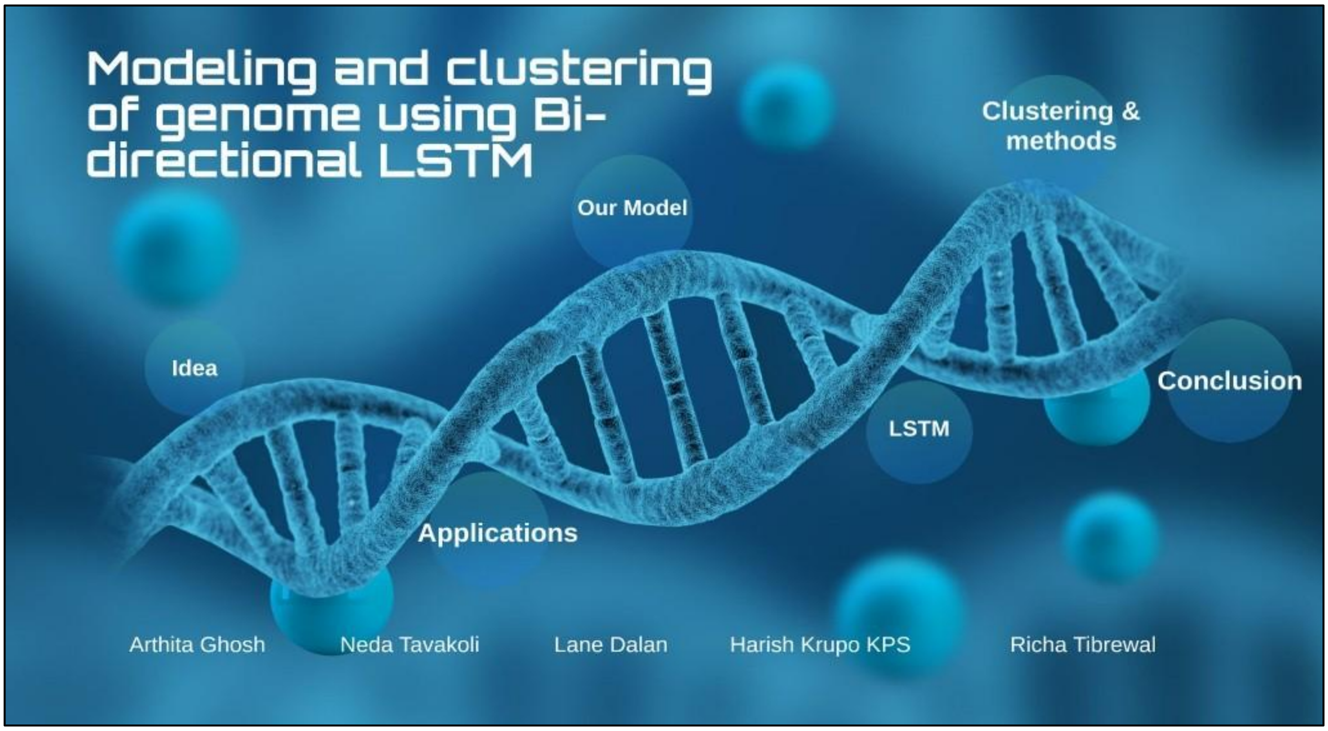
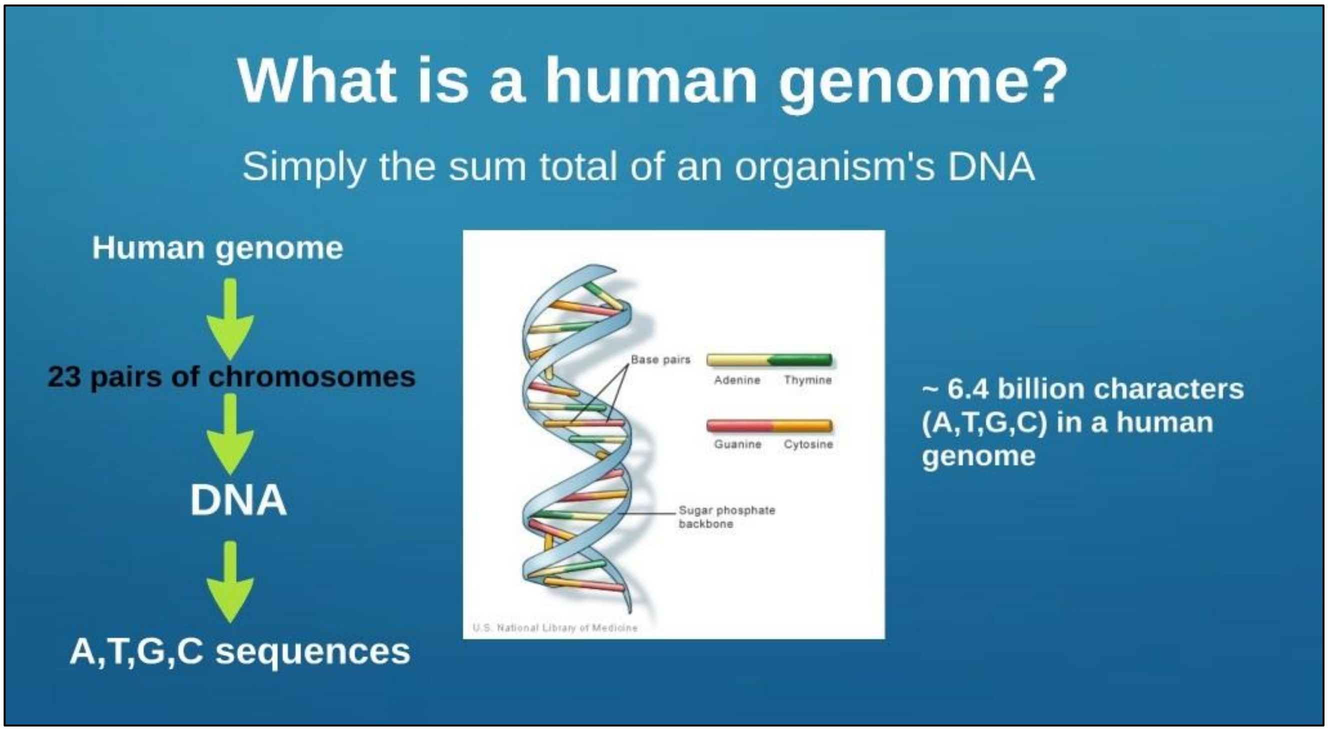


Modeling and clustering of genome using Bi-directional LSTM

Our Model
Clustering & methods
Idea
Conclusion
LSTM
Applications

Arthita Ghosh Neda Tavakoli Lane Dalan Harish Krupo KPS Richa Tibrewal



What is a human genome?

Simply the sum total of an organism's DNA

Human genome

↓

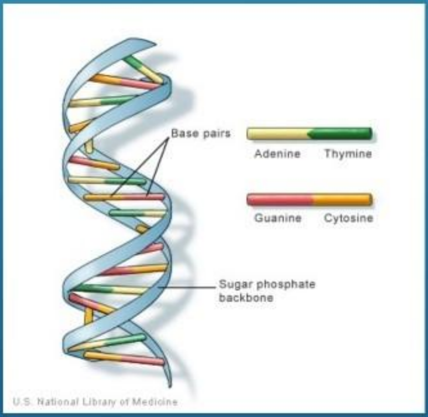
23 pairs of chromosomes

↓

DNA

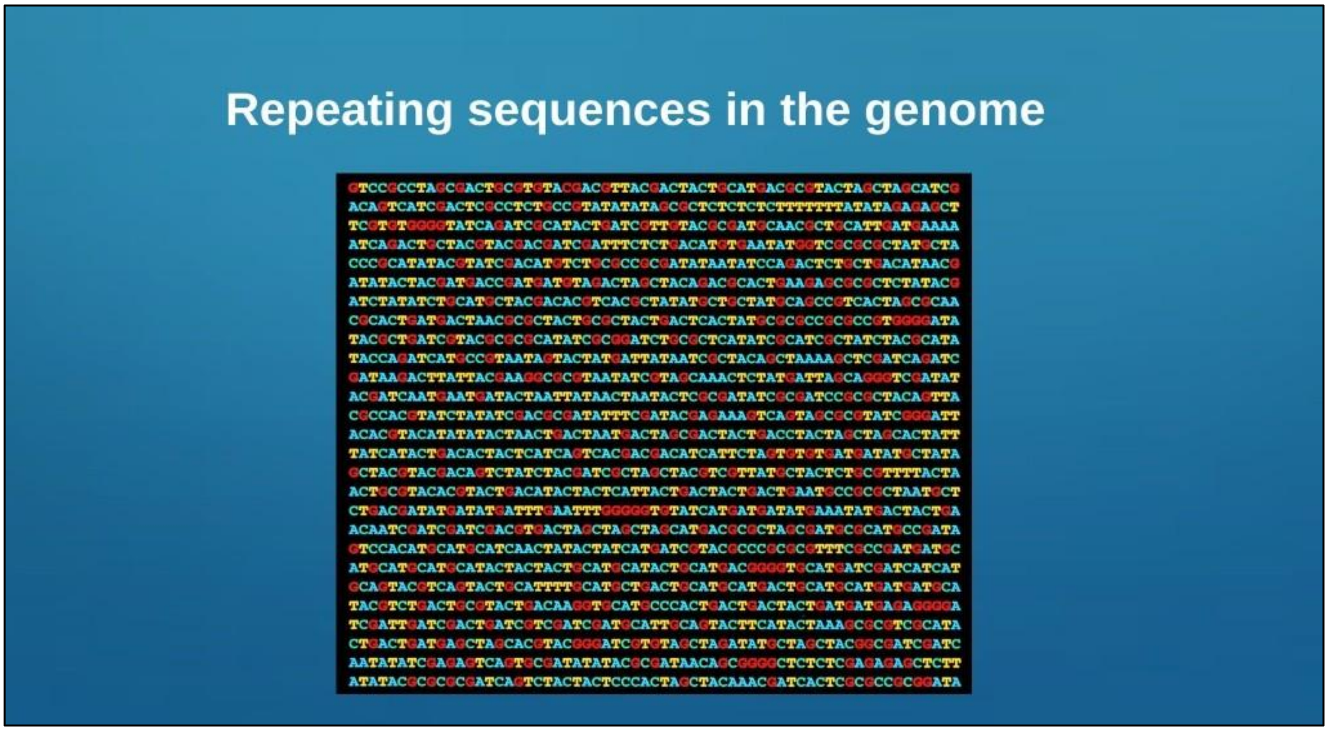
↓

A,T,G,C sequences



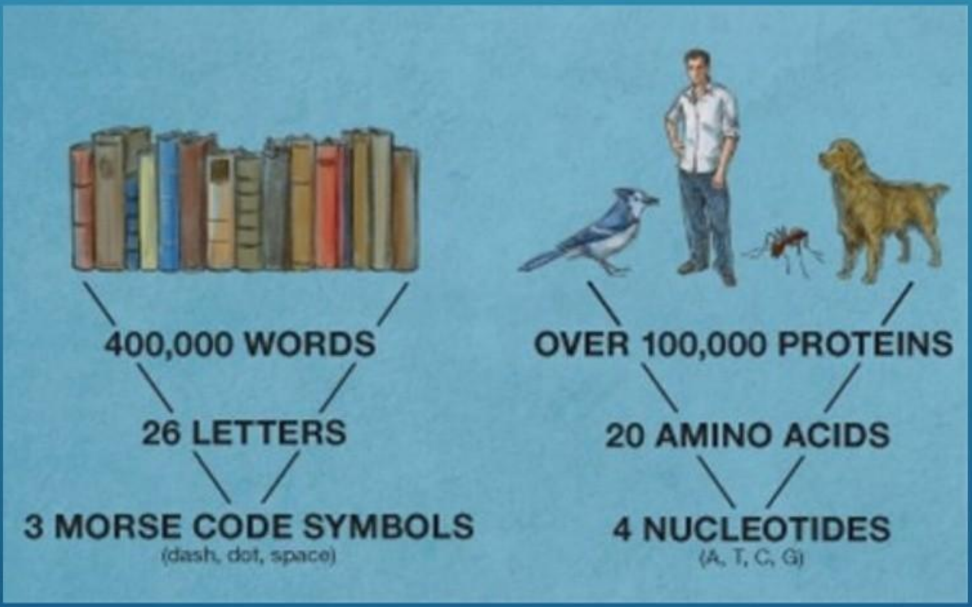
~ 6.4 billion characters (A,T,G,C) in a human genome

Before delving into the objective of our project, it is important to have a high-level idea of what a genome is. Every cell in our body comprises 23 chromosomes and these chromosomes are nothing but long chains of dna comprising 4 nucleotides - Adenine (A), Cytosine (C), Thymine (T) and Guanin (G). The DNA forms a double-stranded helix in which cytosine (C) pairs with guanine (G), and adenine (A) pairs with thymine (T). The genome is made up of the DNA sequences of all these 23 pairs of chromosomes and therefore, contains an organism's complete set of genetic code.



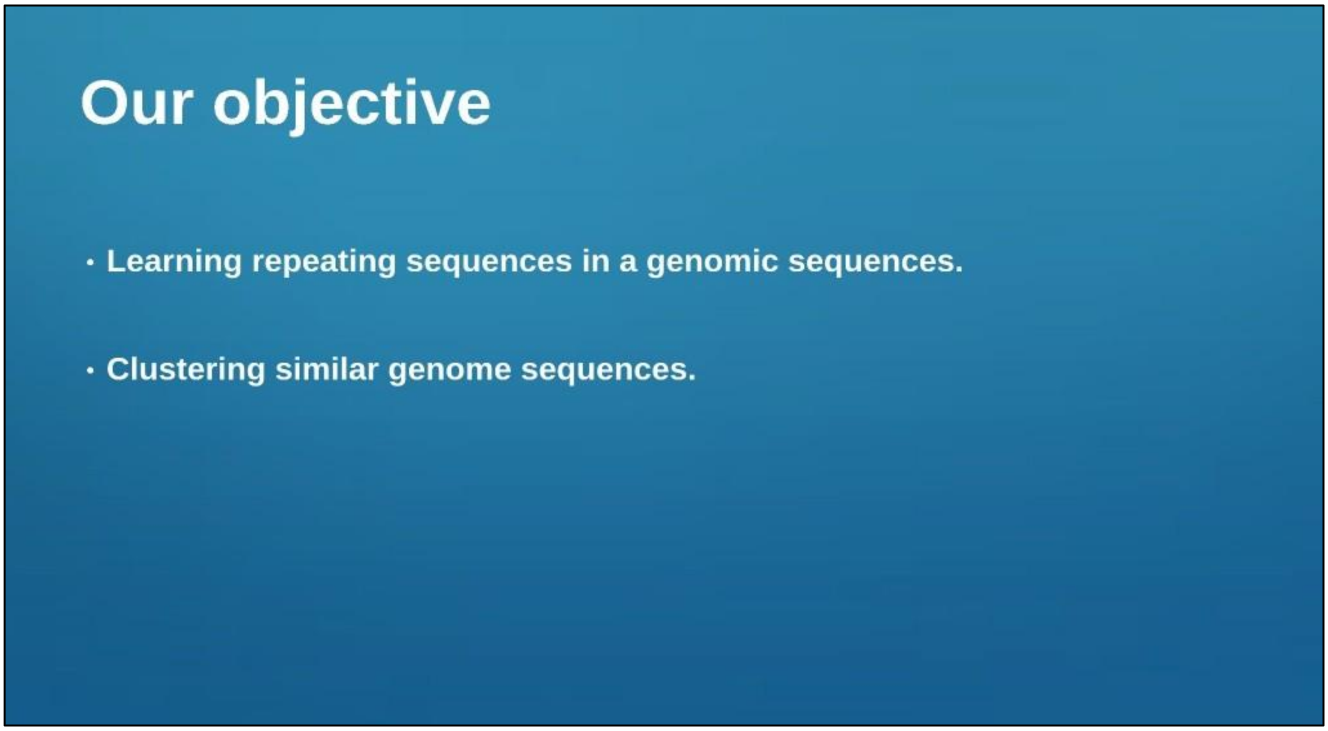
Repeating sequences in the genome

However, the DNA sequence in a genome is not random. In fact, repeating sequences, as shown in figure, convey meaning, i.e., they may be responsible for the generation of some protein which in turn can influence some trait. The sequences shown in the figure are over-simplified and meaningful sequences can comprise over 50 nucleotides. So, it is important to capture these repeating sequences.



Language of Life: Francis Collins

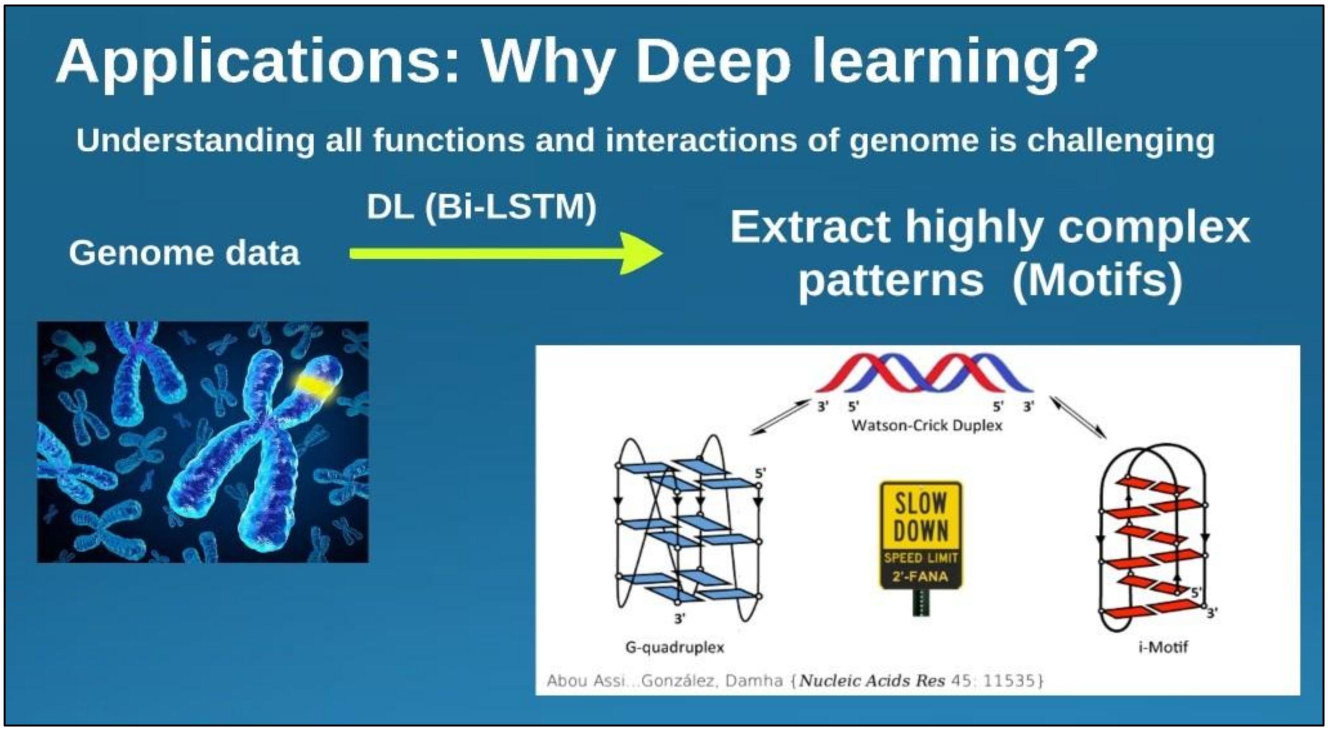
In the famous book - “Language of Life”, geneticist and physicist Francis Collins draws an analogy between the natural language we speak and the genetic code present in our genomes. He thus compares the morse code symbols which can be used to produce the alphabet to the four nucleotides - A,C,T, G present in our genomes. Just like letters of the alphabet combine to form meaningful words, amino acids, too, combine to form proteins. Hence, dna sequences in our genome too can be interpreted to convey meaning just like sentences in English are and they are highly correlated to what sequences follow then or occur before them.



Our objective

- Learning repeating sequences in a genomic sequences.
- Clustering similar genome sequences.

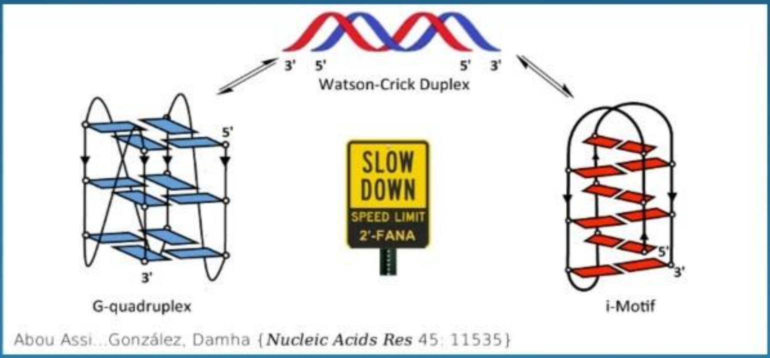
The objective of this project therefore is to train a bidirectional LSTM model to understand and learn the underlying patterns in a genomic sequence and then to use this trained model to detect and cluster group of sequences in the genome which are similar and close to each other.



Applications: Why Deep learning?

Understanding all functions and interactions of genome is challenging

Genome data $\xrightarrow{\text{DL (Bi-LSTM)}}$ Extract highly complex patterns (Motifs)



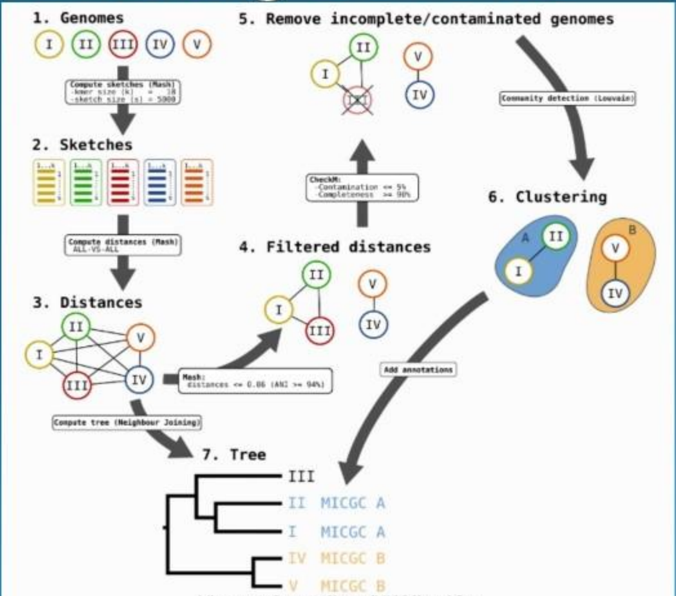
Genome dataset is large and complicated, so understanding all functions and interactions of genome is very challenging. In this project, we use deep learning approaches (specifically a bi-directional LSTM) to model genome data and extract highly complex patterns inside that. To do so, we introduce a novel learning algorithm using the bidirectional LSTM model to generate fixed-length vector representations of the reference sequence; then we conduct a case study on real DNA sequencing dataset, and compute the perplexity, accuracy and sensitivity of the LSTM-based sequence modeling. Then we perform different clustering algorithms on the obtained models to cluster genome dataset. We use the human reference genome as our training dataset, in addition to a set of short reads generated using Illumina sequencing technology. We first decompose the reference genome into multiple sequences. These sequences are then fed into the bidirectional LSTM model and then mapped into fixed-length vectors. The obtained vectors are then used to cluster and group similar sequences into same clusters. We then evaluate the accuracy of our clustering algorithms.

Applications: Clustering Genome

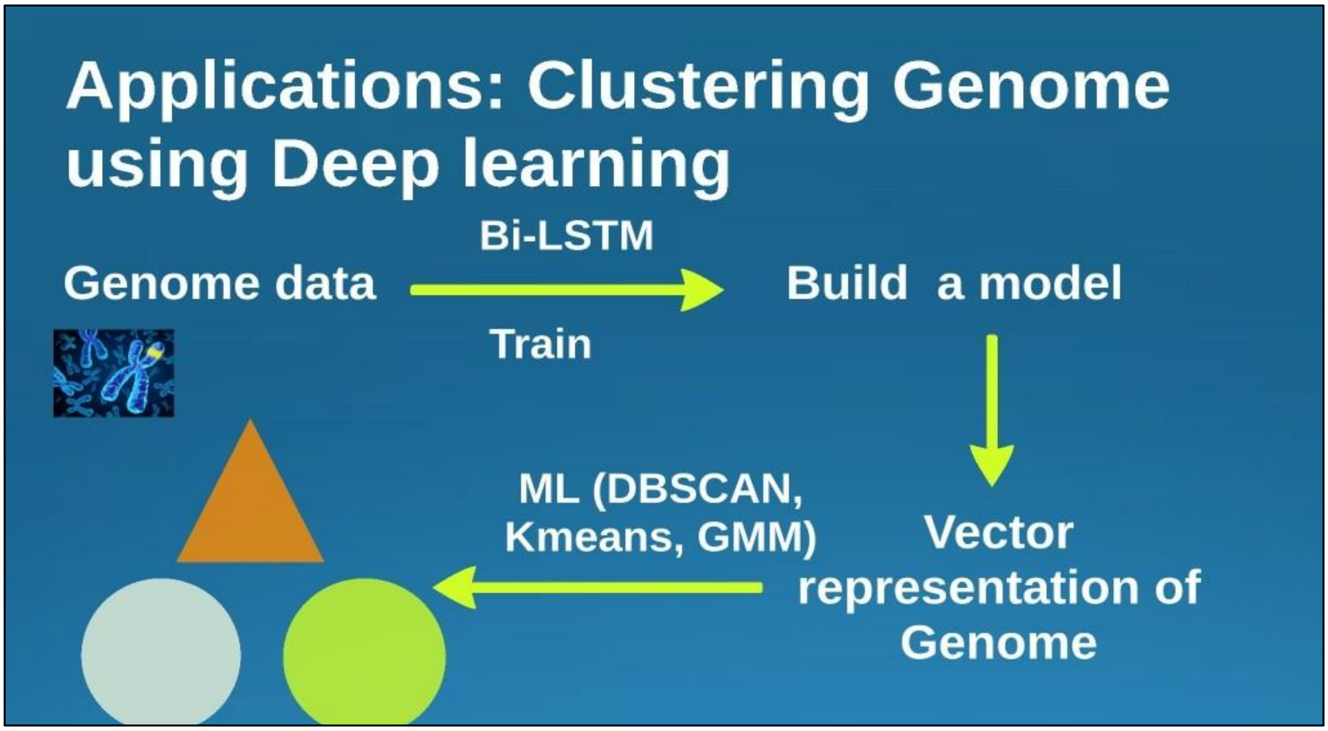
Traditional Genome clustering:

- ~~Computationally intensive~~
- ~~Impractical for large genome~~

Deep learning: Train once , use multiple times


1. Genomes
I II III IV V
Compute sketches (Mash) -kmer size (k) = 18 -sketch size (s) = 5000
2. Sketches
Compute distances (Mash) ALL-VS-ALL
3. Distances
Mash: distances <= 0.06 (ANI >= 94%)
Compute tree (Neighbour Joining)
4. Filtered distances
CheckM: -Contamination <= 5% -Completeness >= 90%
5. Remove incomplete/contaminated genomes
Community detection (Louvain)
6. Clustering
A B
Add annotations
7. Tree
III
II MICGC A
I MICGC A
IV MICGC B
V MICGC B
Microscope Genome Cluster (MICGC) workflow.

Modeling genome dataset using deep learning has different applications, one of them is genome clustering, which is used for this project. The main goal of genome clustering is to detect group of sequences in the genome that are close together (in the sense of pairwise distance and far from other sequences. The main reason of clustering genome using deep learning model is that traditional genome clustering methods are extremely computationally- intensive which make them impractical for real applications. The picture shows one of the common traditional genome clustering methods which is called MicroScope. As the picture shows, at the first step the genome is divided to sketches which are set of sequences of the same length, then pairwise comparisons are done on each pair of sequences (which is a bottleneck step and very time-consuming). Once the pairwise distances are computed, some filtering methods are used to filter any redundant distances. This process has to be done for each set of query sequences, however in this project we are modeling the genome dataset using BI-LSTM once, and use the model multiple times for any list of query sequences.


Applications: Clustering Genome using Deep learning
Genome data
Bi-LSTM
Train
Build a model
Vector representation of Genome
ML (DBSCAN, Kmeans, GMM)

More specifically, in this project we model the genome dataset using Bi-LSTM. We use the following approach to train the model:

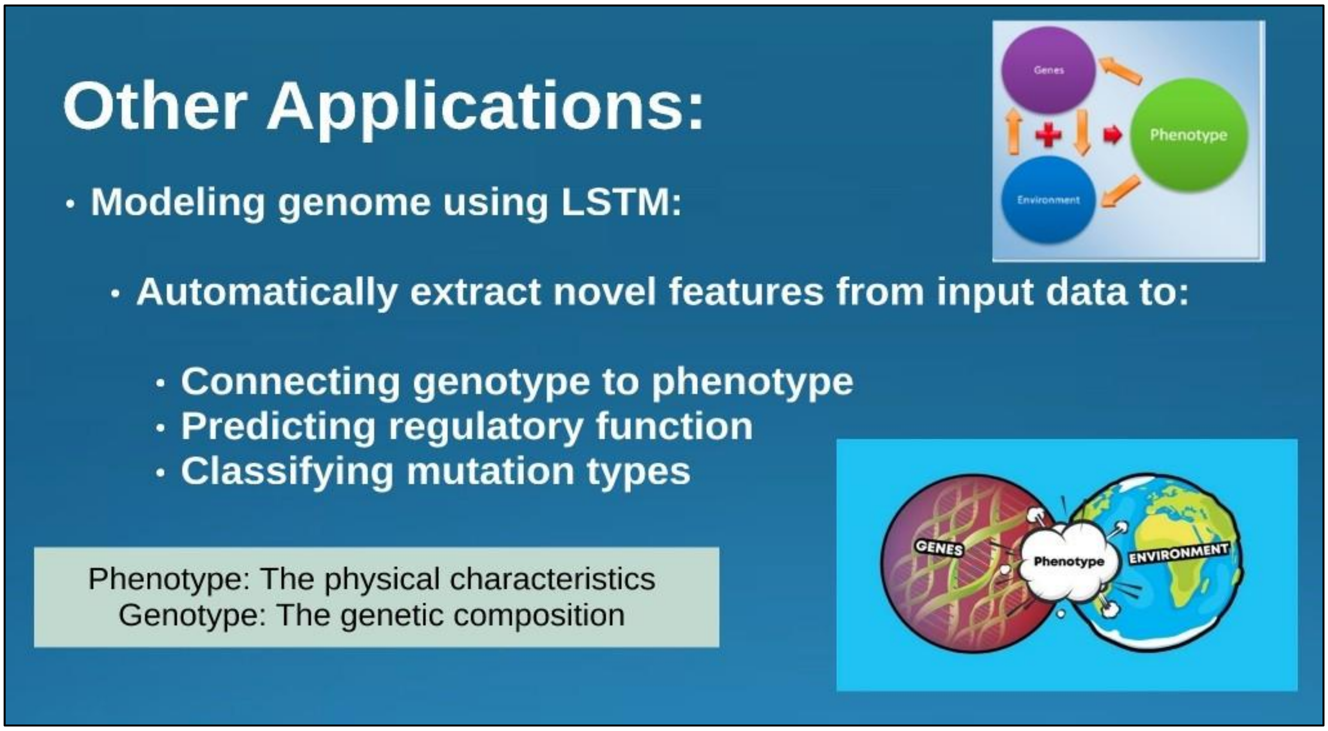
Step 1 (Constructing Words): Each word is made of w characters, so the reference genome that includes a set of characters is split into unique words of size w characters (where w represents the word size). Hence, for the reference genome of size |G|, the number of unique words is at most |G| w , and for the alphabet size of Σ, the max number of unique words is bounded by |Σ| w.

Step 2 (Constructing Dictionary): Dictionary is defined as a collection of unique words. To construct a dictionary, each unique word, generated at the previous step, is added to the dictionary. The dictionary size is bounded by |Σ| w.

Step 3 (Constructing Batch): Each batch is a collection of fixed-length words (words of length w). Indeed, the reference genome is split into multiple batches where each batch consists of multiple words.

Step 4 (Constructing Epoch (Iteration)): Epoch is a collection of all batches. Each epoch runs over the whole reference genome. Hence, epoch size equals to |G|. Indeed, epoch consists of |G| B batches. LSTM network runs batch by batch sequentially. The final updated weights are what is called the training model.

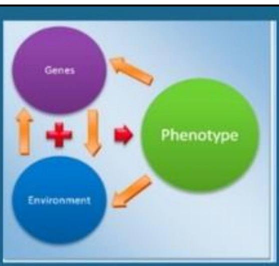
The model (vector representations) is then used for different clustering algorithms (i.e., DBSCAN, Kmeans, GMM).

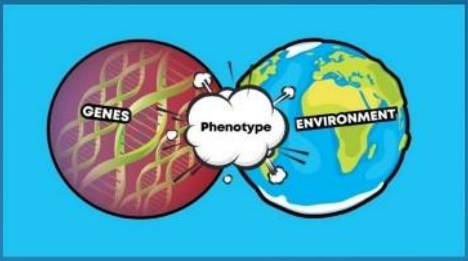


Other Applications:

- Modeling genome using LSTM:
  - Automatically extract novel features from input data to:
    - Connecting genotype to phenotype
    - Predicting regulatory function
    - Classifying mutation types

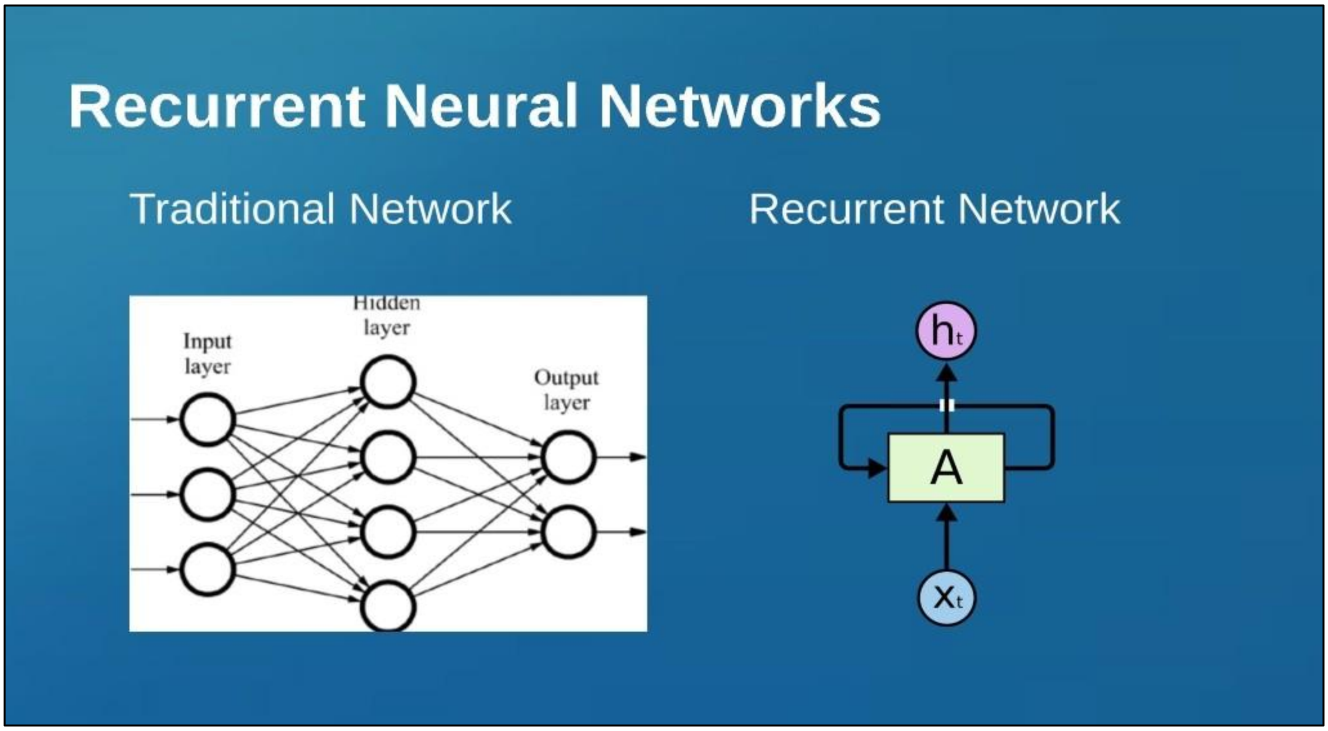
Phenotype: The physical characteristics
Genotype: The genetic composition





Other applications of genome modeling are as follows (which are not used for this project):

1. Connecting genotype to phenotype
2. Predicting regulatory function
3. Classifying mutation types



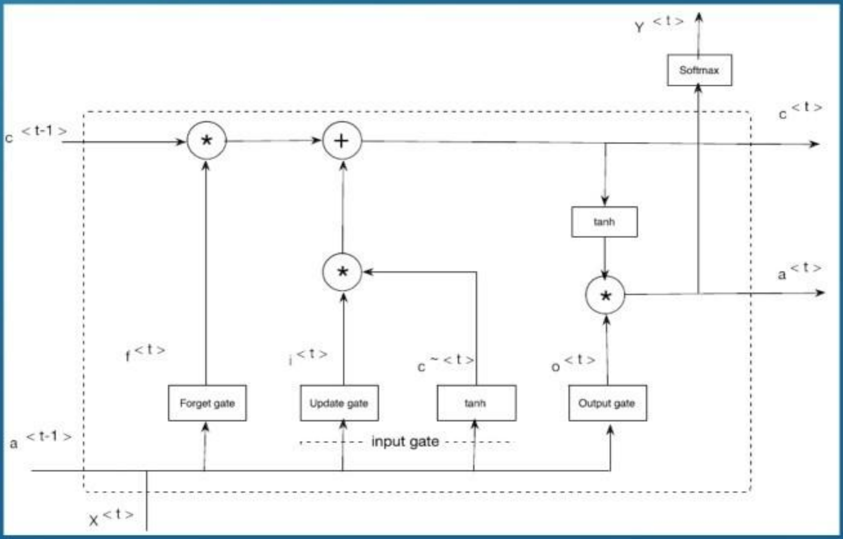
Recurrent Neural Networks

Traditional Network

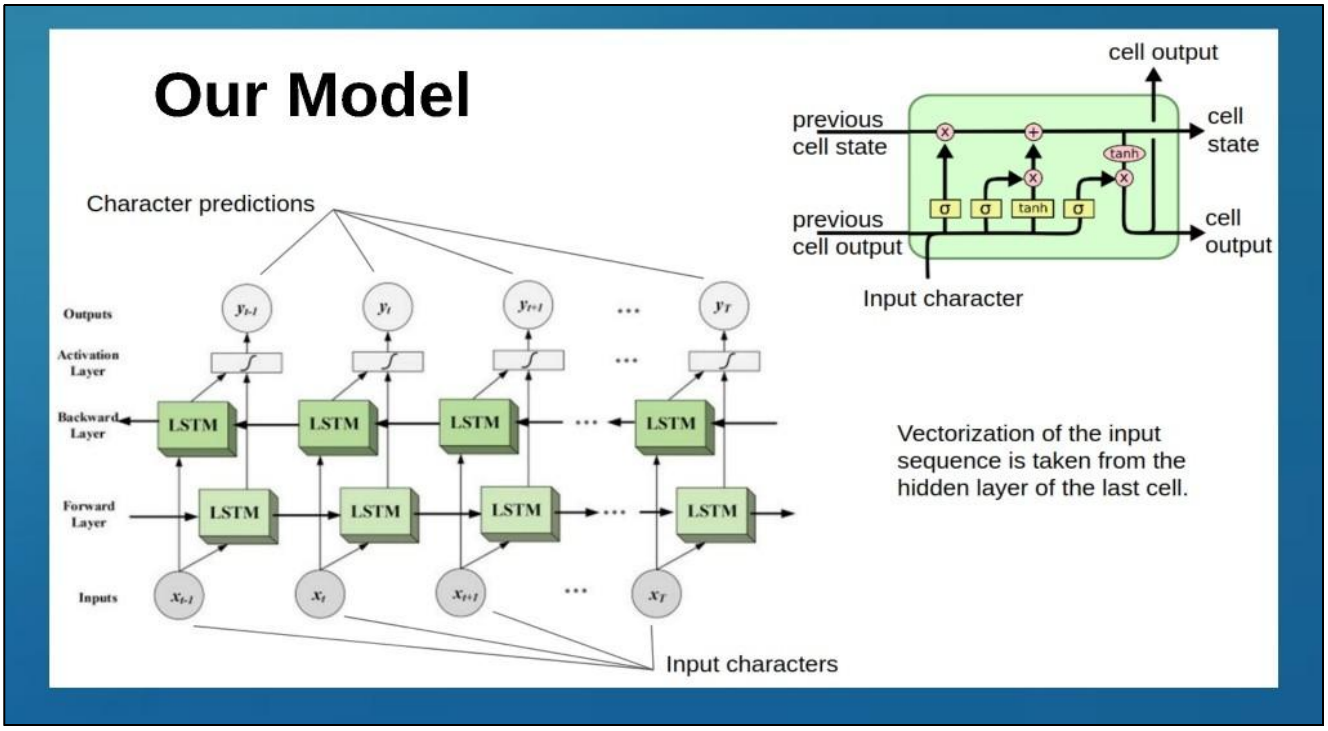
Recurrent Network

By utilizing a recurrent network we were able to train for one sequence length and then vectorize a different sequence length. This made the model more robust to varying input data, which can be especially useful when processing the natural language validation.

LSTM, Long Short Term Memory



The cell output was used for the vectorization of sequences by the bi-directional model. The cell state was used for the vectorization of the forward backward model (discussed on the following page). By utilizing a human interpretable sequence, like natural language, we can determine which vectorization is more applicable.

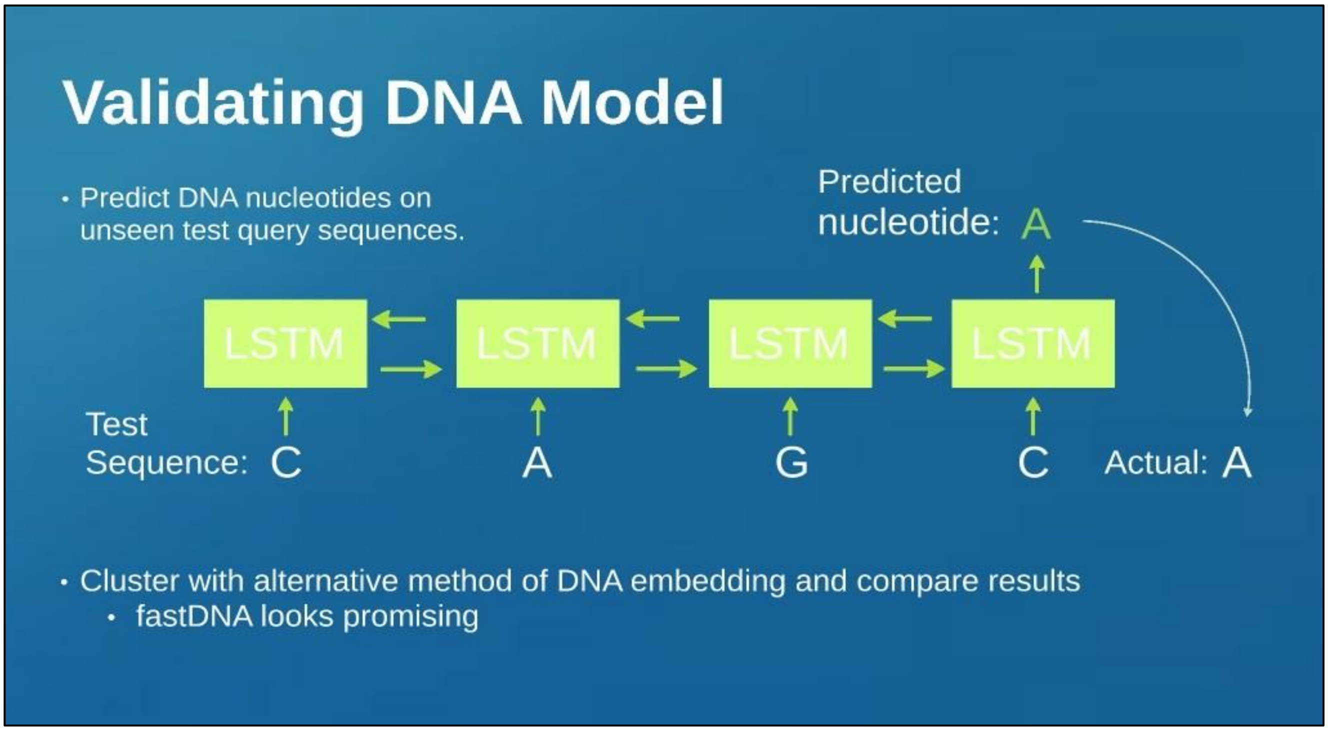


Our Model

Vectorization of the input sequence is taken from the hidden layer of the last cell.

A possible issue was found with training on bi-directional LSTM with an output sequence similar to the input sequence. As an alternative to the bi-directional structure, a forward backward structure was also implemented. The foward LSTM predicts the following entry to the sequence and the backward LSTM predicts the previous entry. This allows for the sequence to be examined both ways while eliminating a possible sequencing issue.

Not only can the sequence be used to create a vectorization (from cell output / cell state) but the sequence embedding can also be utilized. Each unique component (a “word” is a good analogy for natural language) of the sequence has a unique embedding. This embedding itself could help vetorize and thus cluster like DNA strands (“words”).



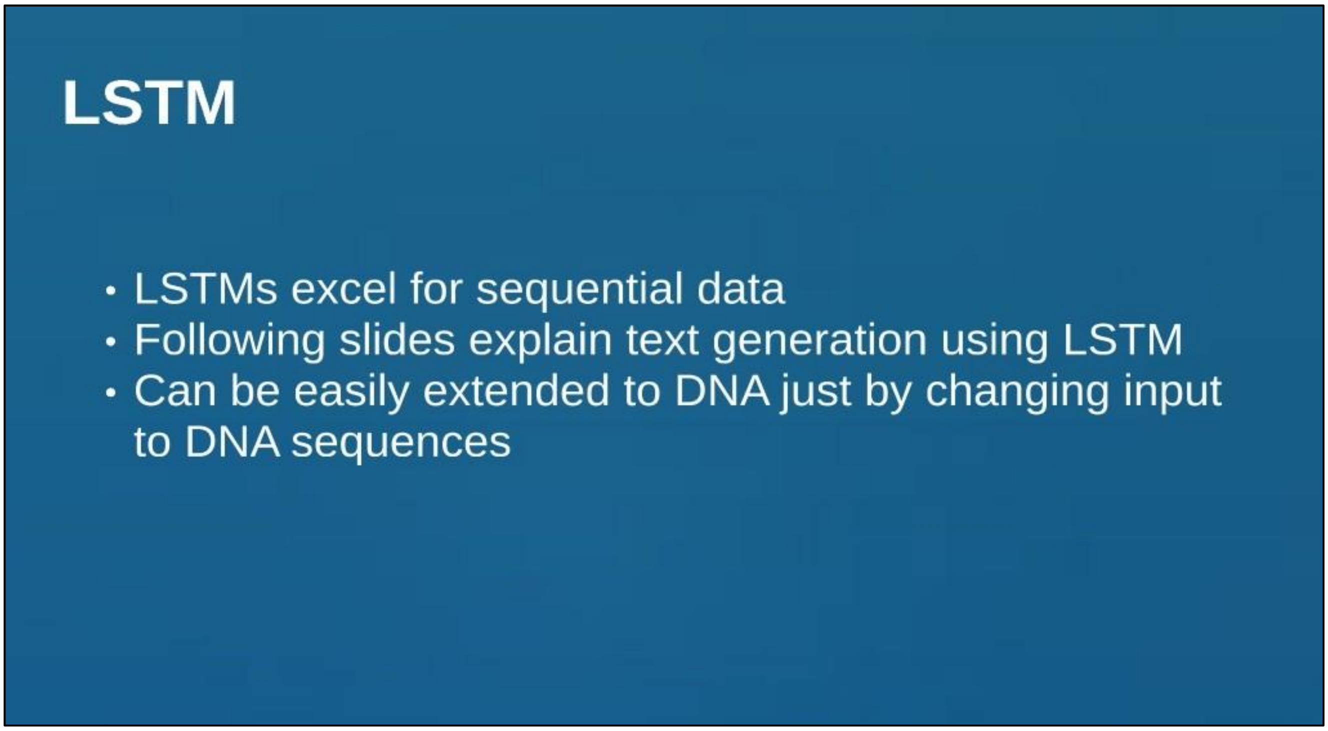
Validating DNA Model

- Predict DNA nucleotides on unseen test query sequences.

Predicted nucleotide: A
LSTM
LSTM
LSTM
LSTM
Test Sequence: C
A
G
C
Actual: A

- Cluster with alternative method of DNA embedding and compare results
  - fastDNA looks promising

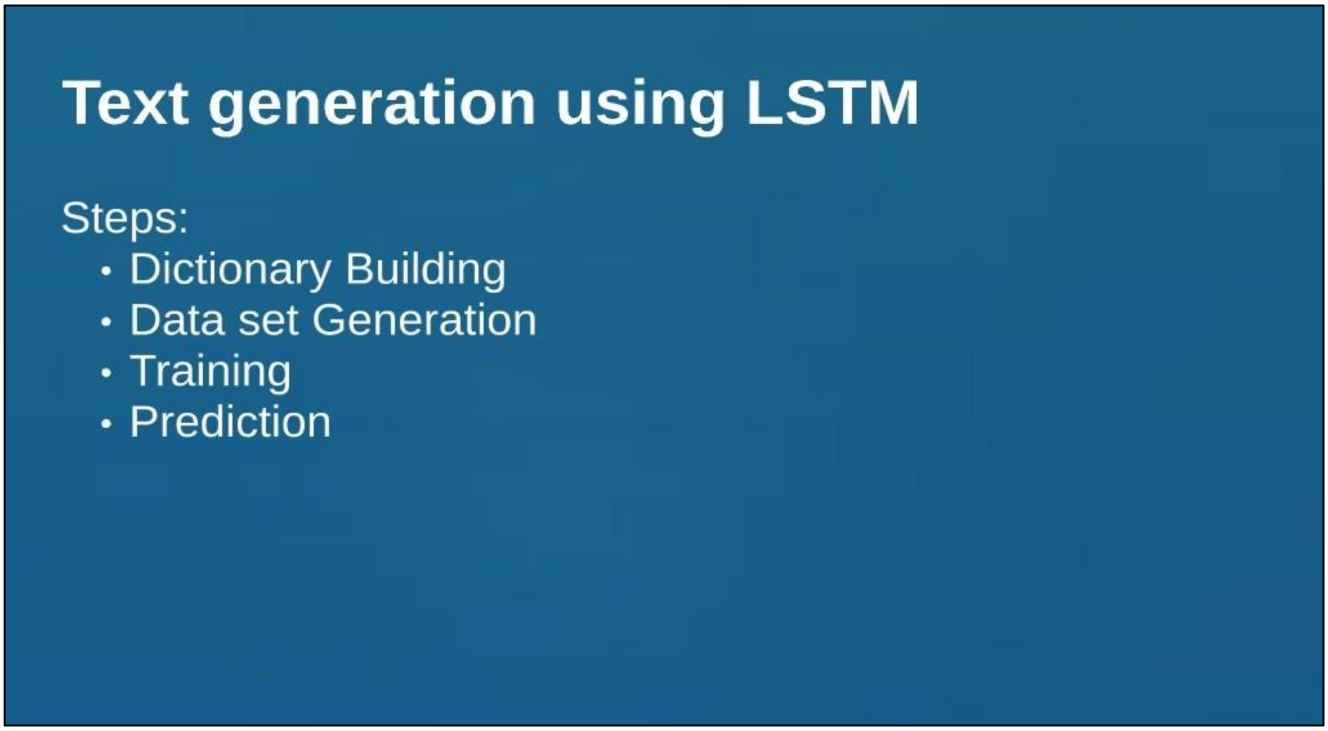
Validating the model would give a significant indication that it is working correctly.



LSTM

- LSTMs excel for sequential data
- Following slides explain text generation using LSTM
- Can be easily extended to DNA just by changing input to DNA sequences

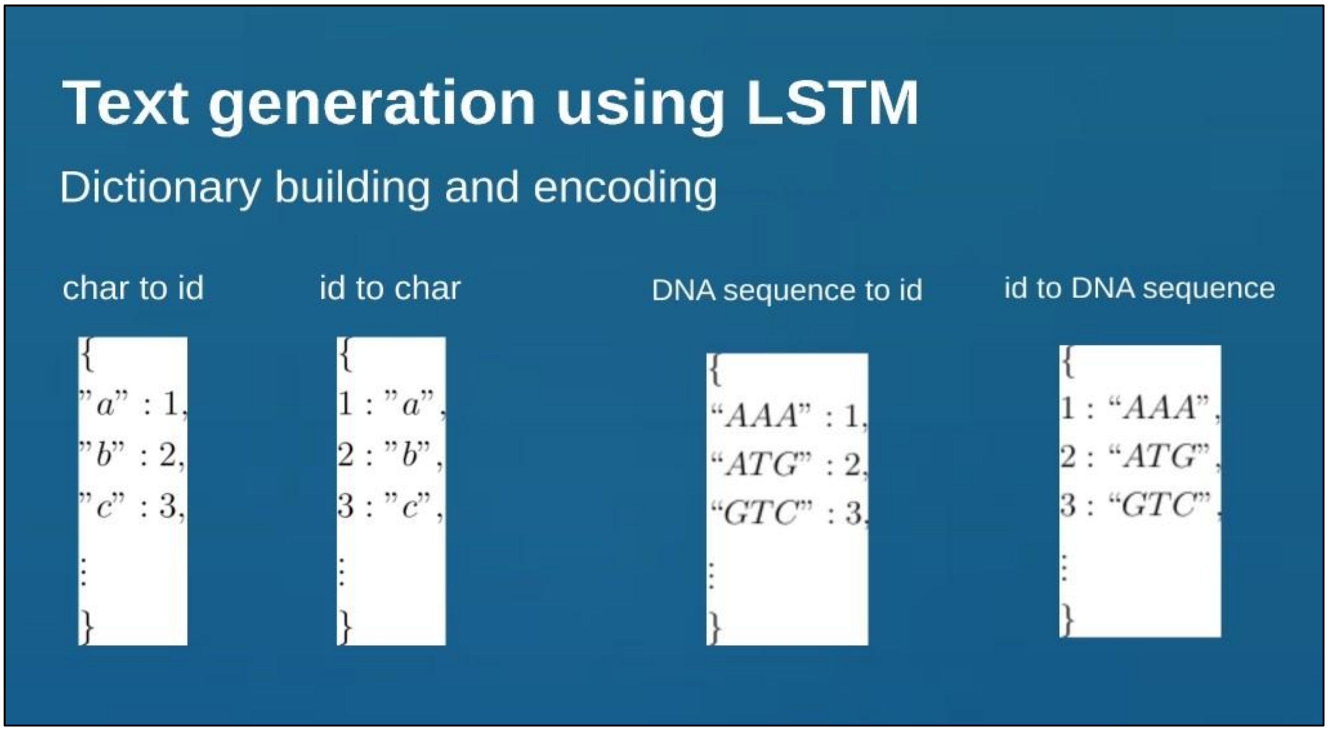
Lstms have been proven to be extremely effective in learning time series data. To be specific, sequential data. They have wide ranging application due to their ability to selectively remember previously seen data.
They learn sequential dependencies in the data and can be used for DNA which is sequential in nature. DNA of an organism is also sequential in nature and thus we use an LSTM to train it. Before looking at LSTM on DNA, lets look at text data.



Text generation using LSTM

Steps:
- Dictionary Building
- Data set Generation
- Training
- Prediction

Steps involved in learning using an LSTM. Dictionary building, Data set generation, Training and Prediction.



Text generation using LSTM

Dictionary building and encoding

char to id

```
{
"a" : 1,
"b" : 2,
"c" : 3,
⋮
}
```

id to char

```
{
1 : "a",
2 : "b",
3 : "c",
⋮
}
```

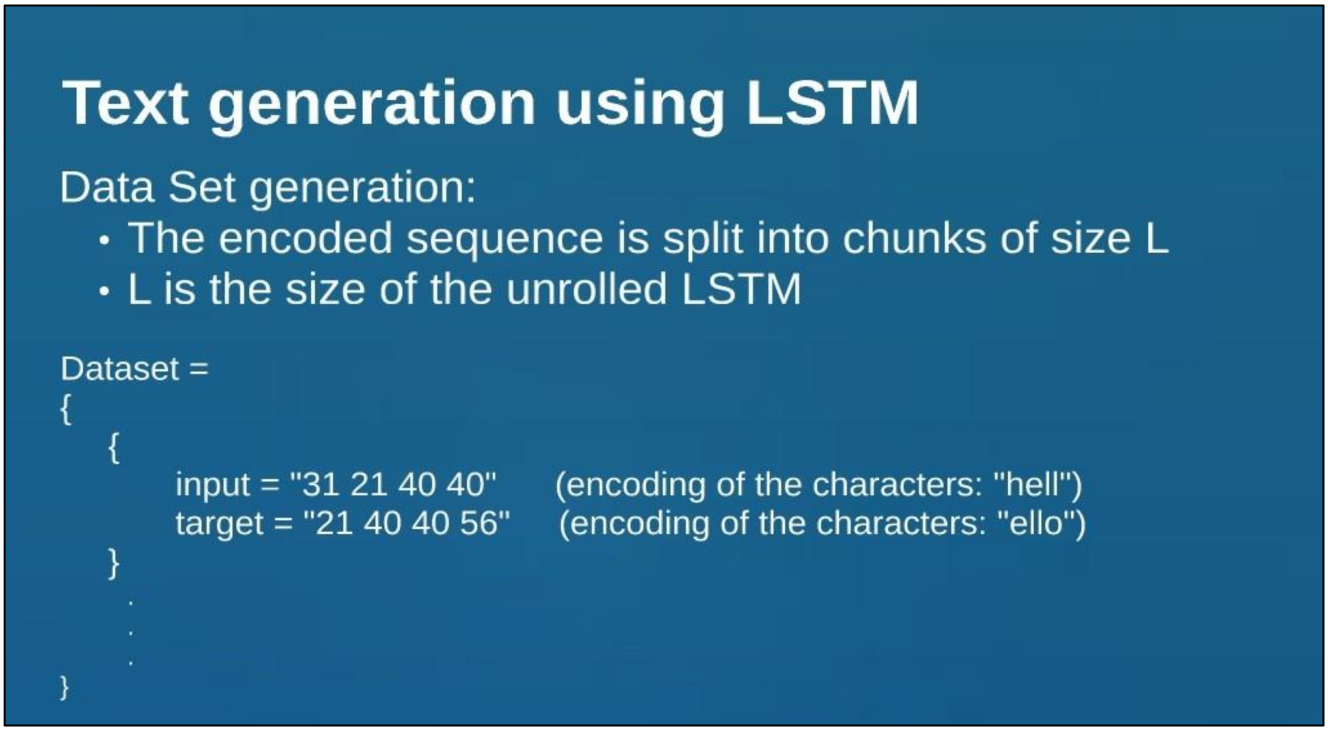
DNA sequence to id

```
{
"AAA" : 1,
"ATG" : 2,
"GTC" : 3,
⋮
}
```

id to DNA sequence

```
{
1 : "AAA",
2 : "ATG",
3 : "GTC",
⋮
}
```

LSTMs understand numbers, more specifically one hot representations or vector embeddings as input. This requires that the application specific data, what ever it may be needs to be encoded using numbers. This is achieved using dictionary generation. Dictionaries map from application specific “vocabulary” to their encoding. A reverse dictionary is also required to convert from the output generated from the LSTM (prediction) to the characters)
For a character generator: list of all the unique characters
For a Word predictor: list of all unique words
And for DNA sequences: ATCG sequences



Text generation using LSTM

Data Set generation:

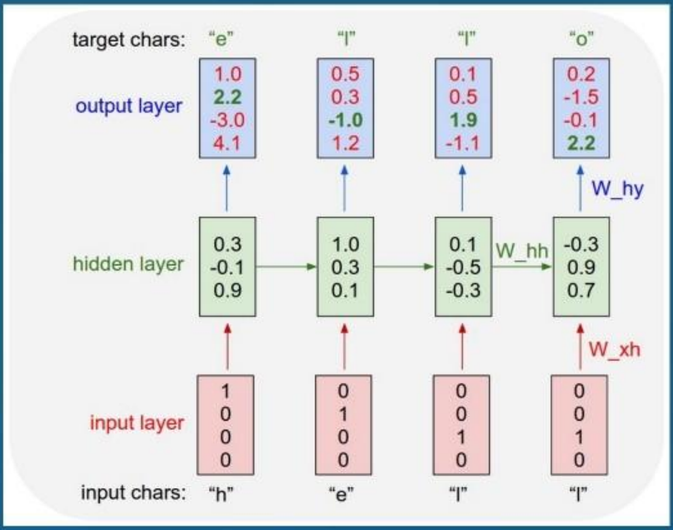
- The encoded sequence is split into chunks of size L
- L is the size of the unrolled LSTM

```
Dataset =
{
    {
        input = "31 21 40 40"      (encoding of the characters: "hell")
        target = "21 40 40 56"     (encoding of the characters: "ello")
    }
      .
      .
      .
}
```

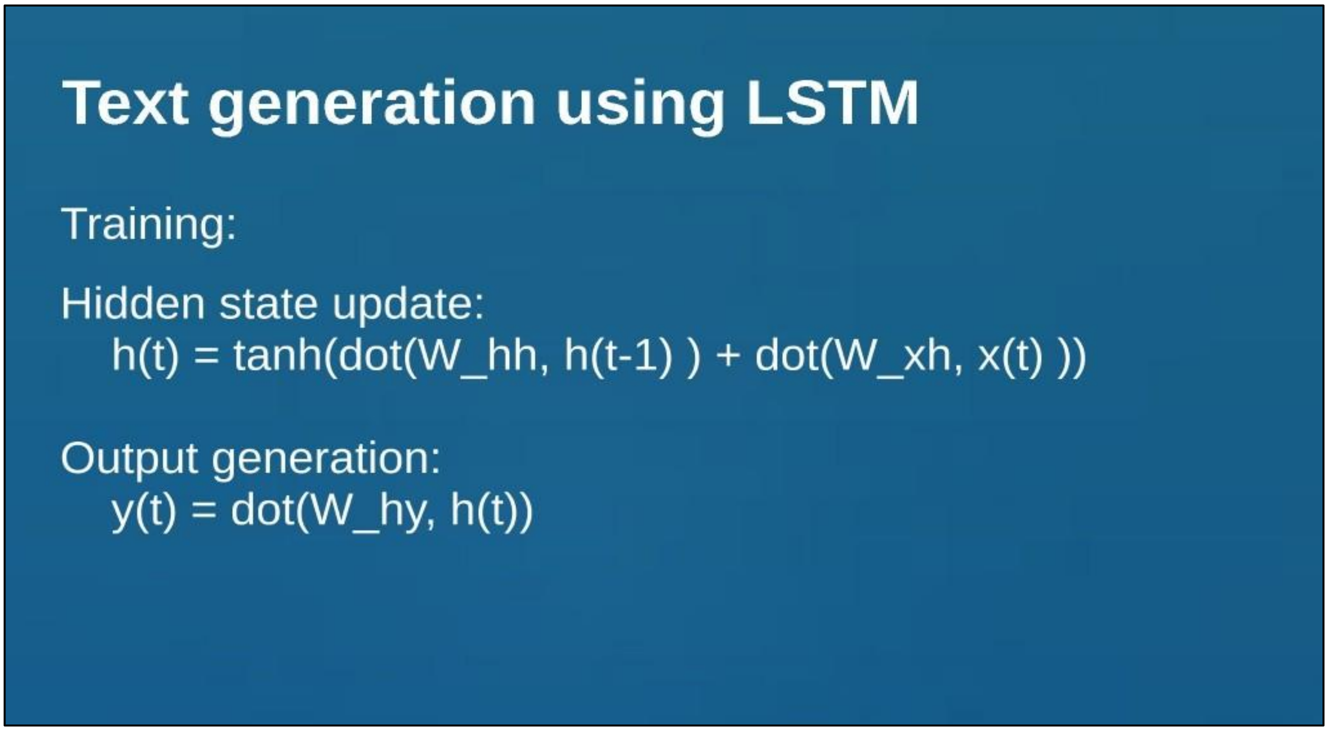
Data set is split into sequences of length l. This is the same length of the LSTM when unrolled. For a character prediction model, the input is the encoded characters. Target is the next set of characters that follow. It is a left shifted version of the input. For example when we see "h", the data contains "e" as next character

Text generation using LSTM

Training and prediction



The image shows a very simplistic view of the LSTM. It is an RNN to be specific and is picked for ease of explanation. The LSTM is unrolled/unfolded to its size.



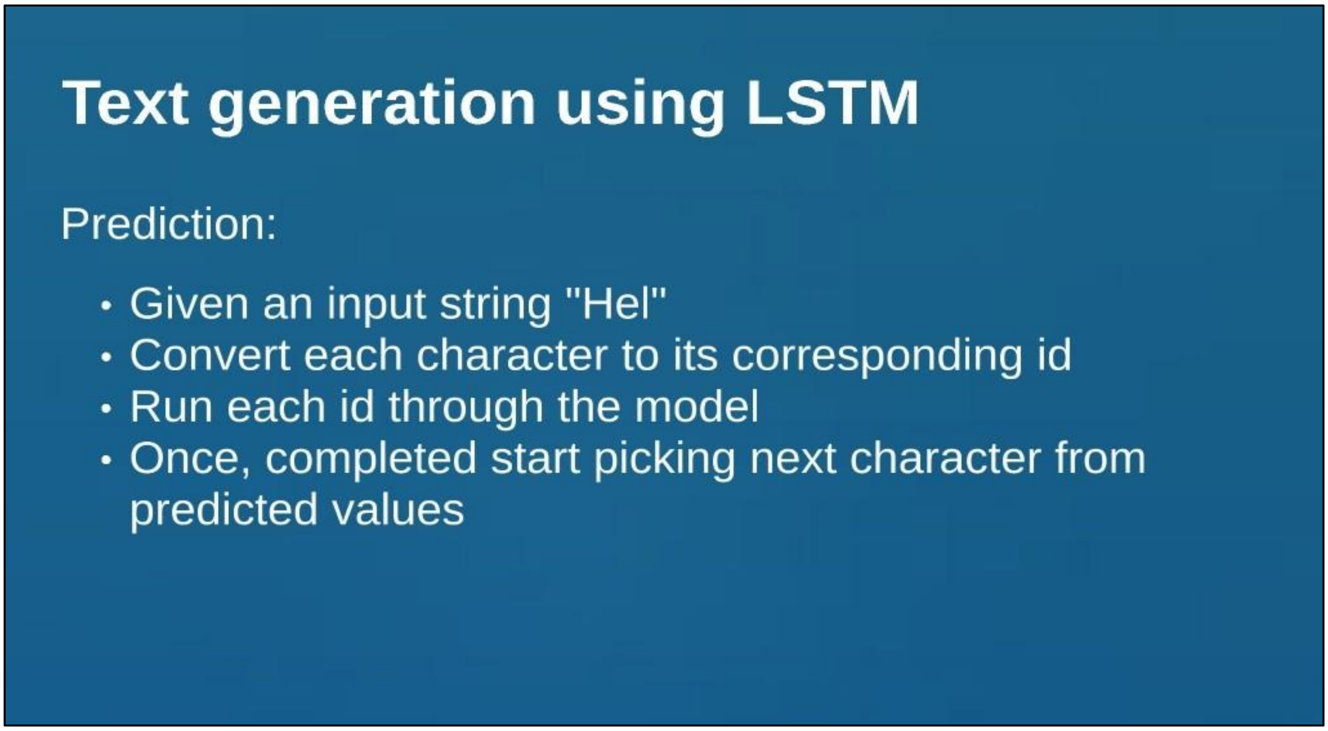
Text generation using LSTM

Training:

Hidden state update:
h(t) = tanh(dot(W_hh, h(t-1)) + dot(W_xh, x(t)))

Output generation:
y(t) = dot(W_hy, h(t))

For training, the unfolded LSTM (show previous slide) is fed characters from left to right. After unfolding, it is trained very similar to a normal neural network. In the forward pass the hidden state is calculated from the equation shown above and then output is generated from the hidden layer. The weight matrices whh, wxh and wyh are the parameters that are learnt during training. Once the number of characters fed is equal to the length of the LSTM, a slightly modified backward propagation is applied which starts from the last unrolled layer and goes back in time and also passed down the gradient backwards in time.



Text generation using LSTM

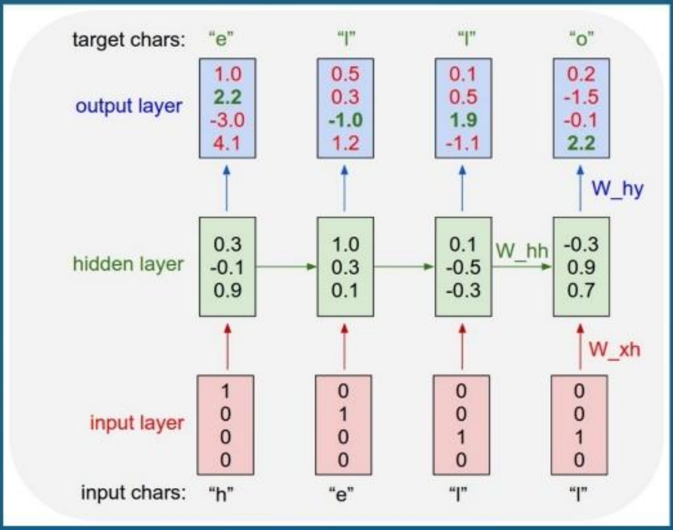
Prediction:

- Given an input string "Hel"
- Convert each character to its corresponding id
- Run each id through the model
- Once, completed start picking next character from predicted values

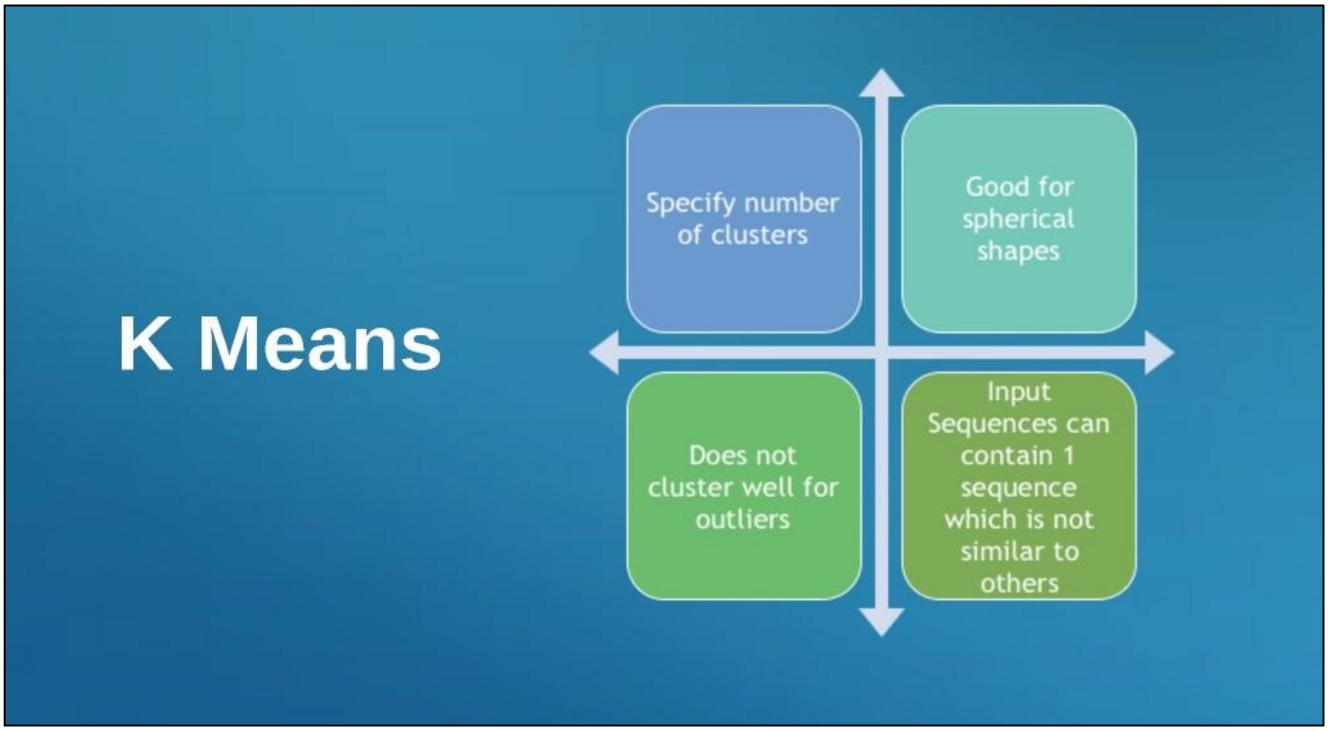
Prediction is straight forward. Given an input string “Hel”, we convert it to its numerical encoding and pass it to the model. While passing the input, we ignore the prediction made by the model. Once the input is completely fed into the model, we start picking new values from the prediction. (Show next slide)

Text generation using LSTM

Training and prediction



Clustering

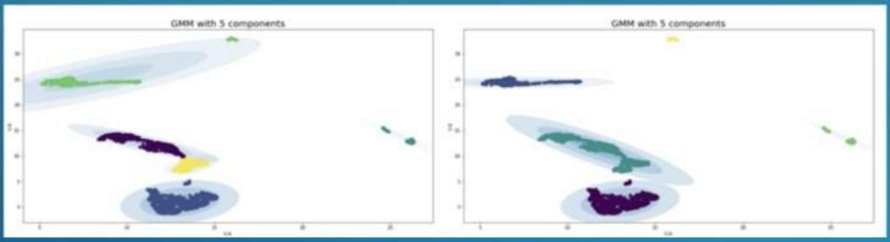


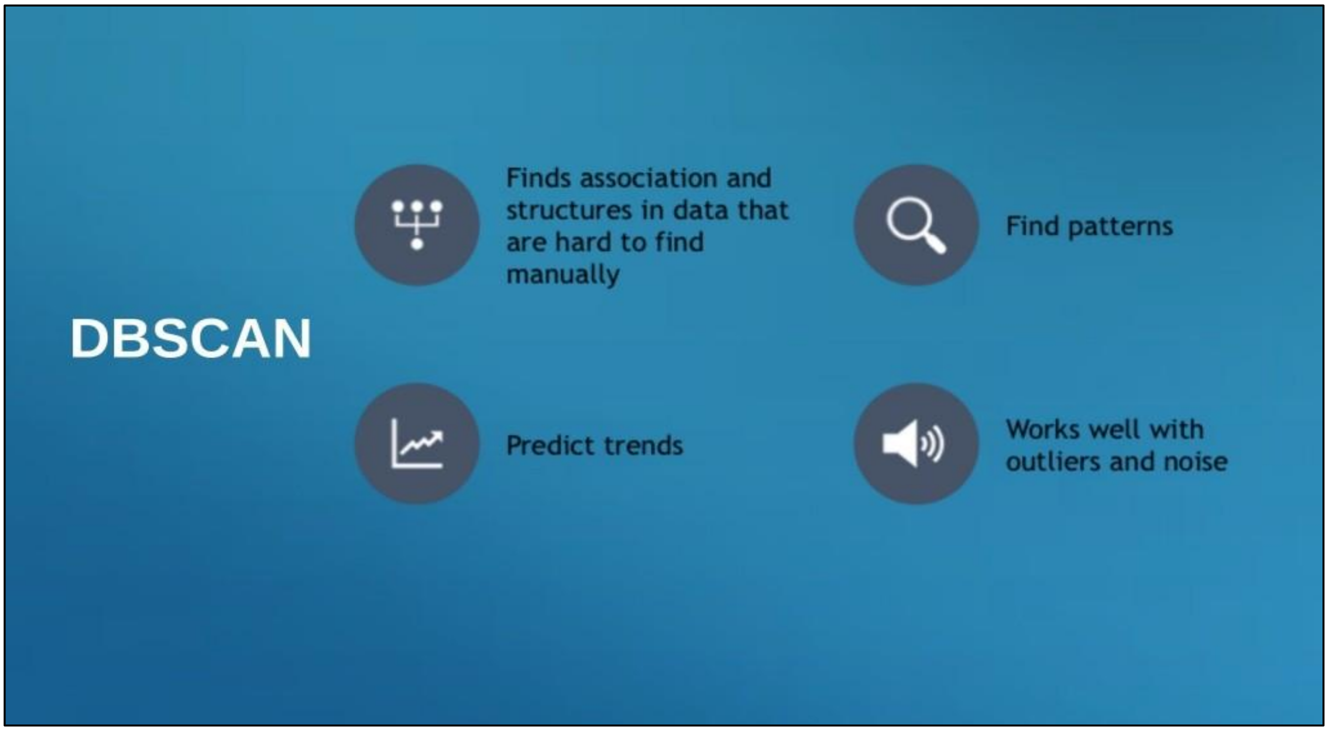
K Means

Specify number of clusters
Good for spherical shapes
Does not cluster well for outliers
Input Sequences can contain 1 sequence which is not similar to others

GMM

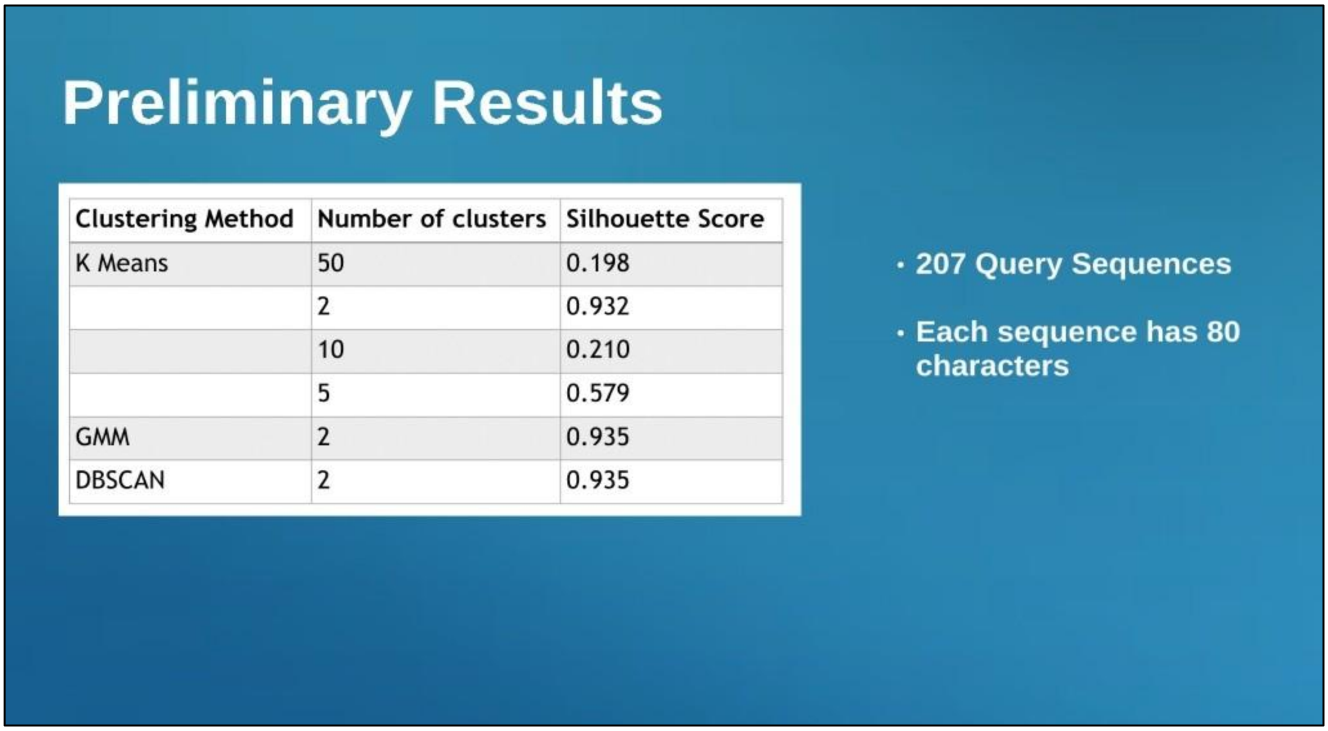
- Takes into account elliptical shapes
- Central Limit Theorem
- Different initialization points may lead to different configurations





DBSCAN

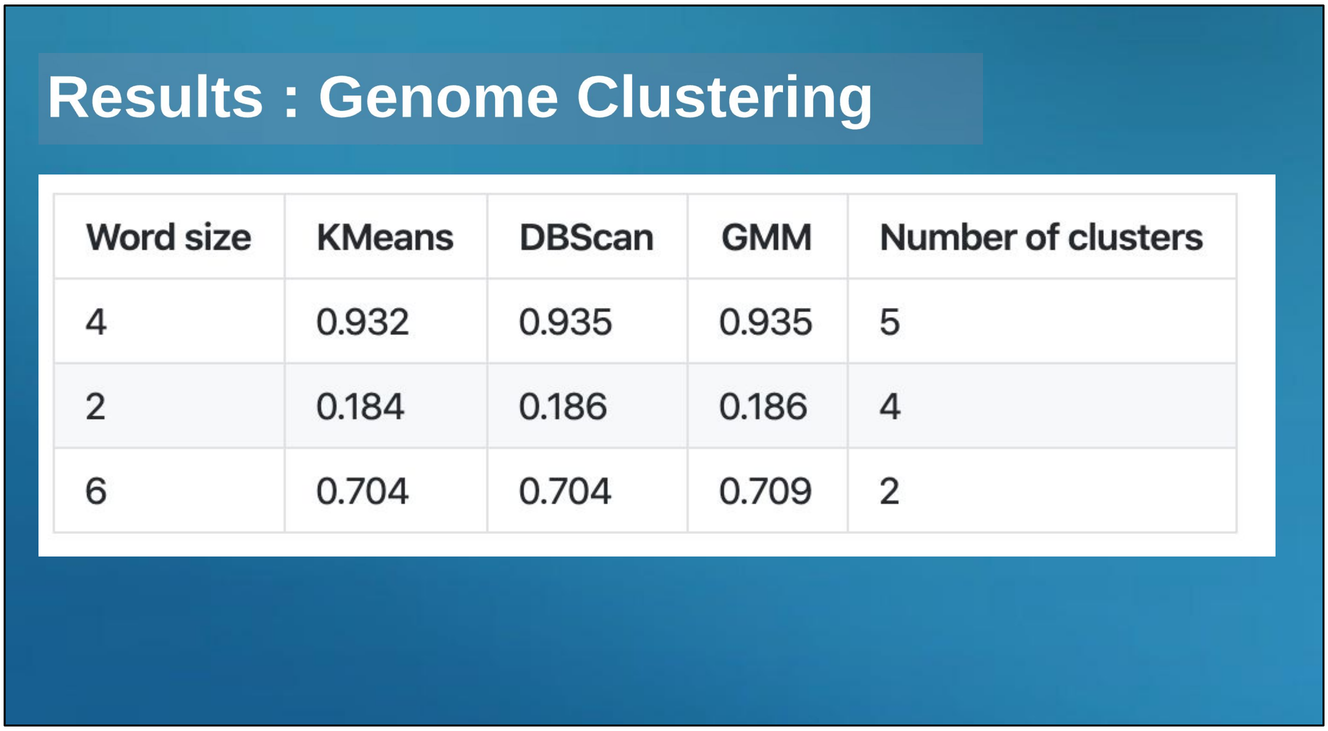
- Finds association and structures in data that are hard to find manually
- Find patterns
- Predict trends
- Works well with outliers and noise



Preliminary Results

Clustering Method	Number of clusters	Silhouette Score
K Means	50	0.198
	2	0.932
	10	0.210
	5	0.579
GMM	2	0.935
DBSCAN	2	0.935

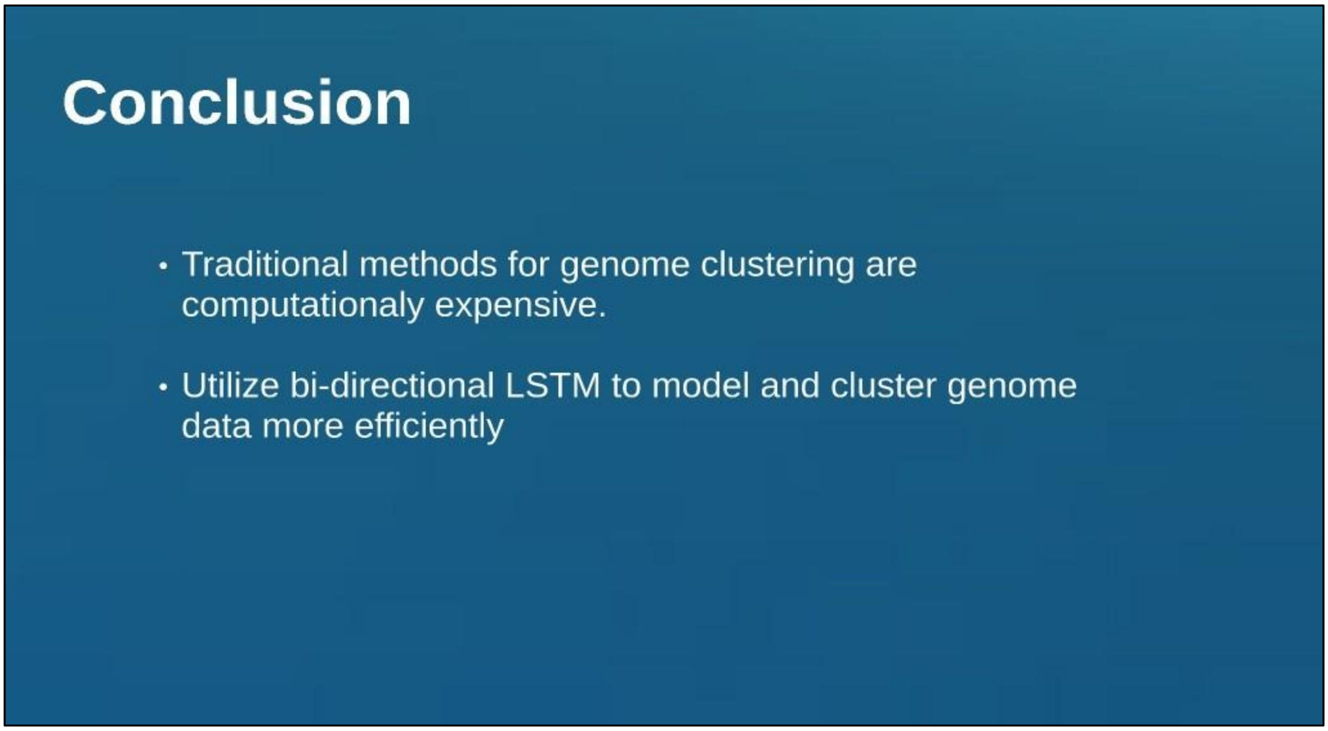
- 207 Query Sequences
- Each sequence has 80 characters



Results : Genome Clustering

Word size	KMeans	DBScan	GMM	Number of clusters
4	0.932	0.935	0.935	5
2	0.184	0.186	0.186	4
6	0.704	0.704	0.709	2

Word size is the size of the word taken for each query sequence. We got the best results for word size 4 and worst result for word size 2. On analyzing this result, we can conclude that since these query sequences are a combination of ‘A’, ‘C’, ‘T’ and ‘G’, word size of 2 will not carry any meaningful data and will not convey any information about protein sets.



Conclusion

- Traditional methods for genome clustering are computationaly expensive.
- Utilize bi-directional LSTM to model and cluster genome data more efficiently

Thank You !!