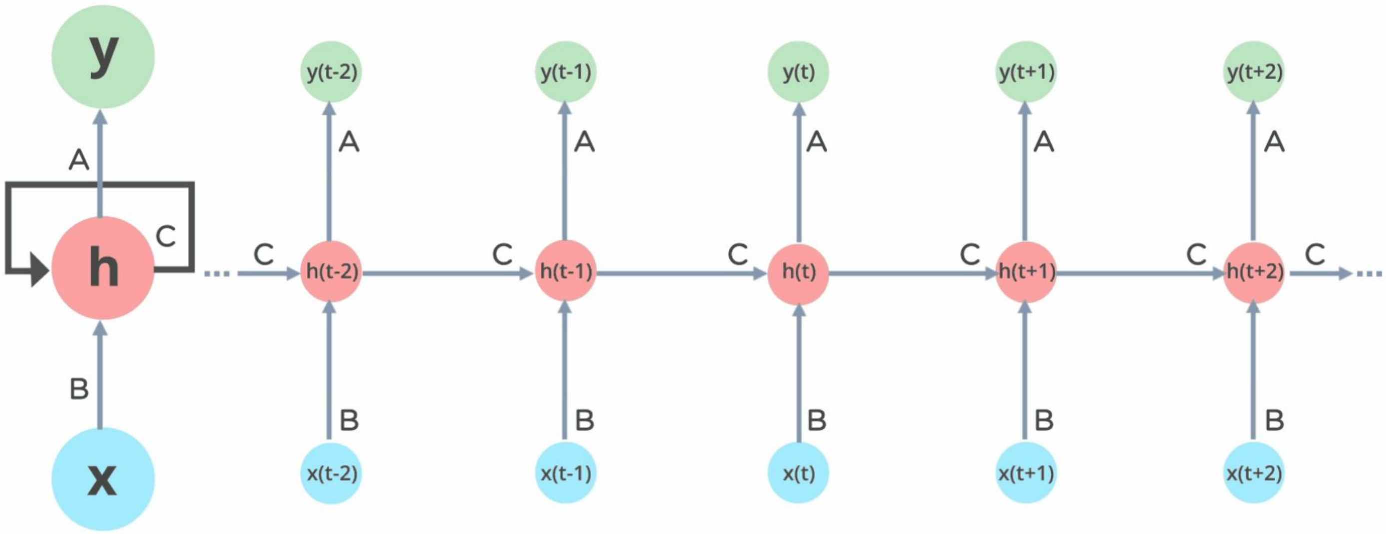
LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush

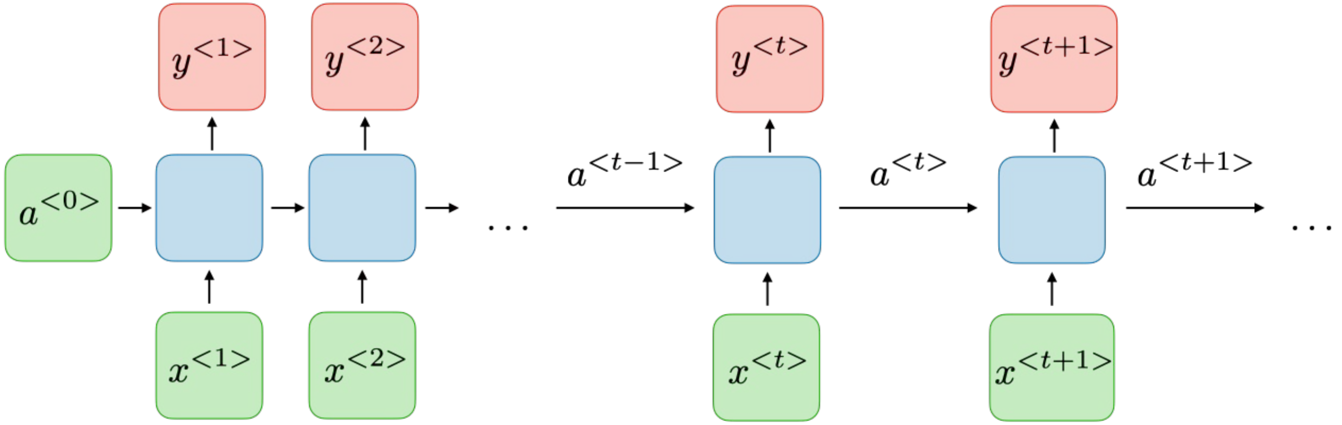
Main Concepts

What is Recurrent Neural Networks (RNN)?



- RNN use the same weights for each element of the sequence.
- Decreasing the number of parameters.
- Allows the model to generalize to sequences of varying lengths.
- A RNN can anticipate sequential data in a way that other algorithms can't.

The Architecture of a Traditional RNN



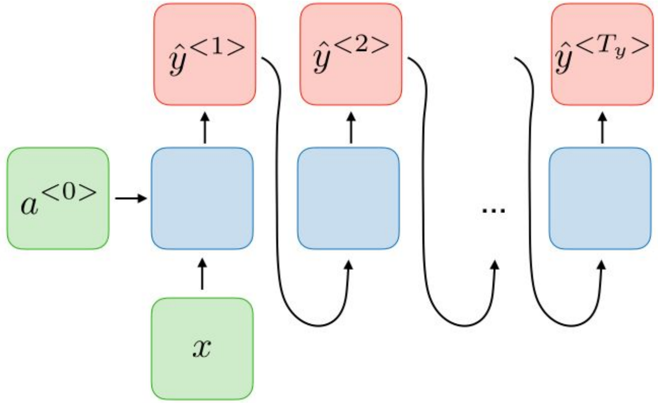
For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

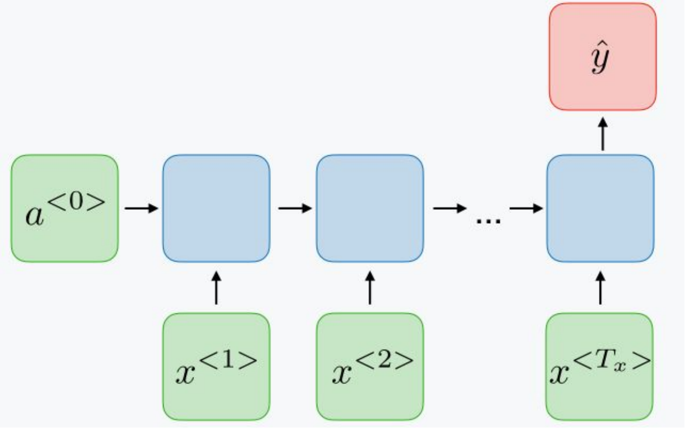
where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.

Types of RNN

- **One to Many**: There is only one pair here. A one-to-one architecture is used in traditional neural networks. E.g, Music generation.

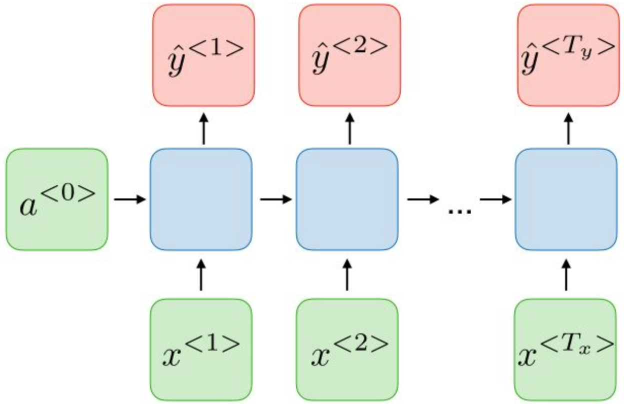


- **Many To One:** A single output is produced by combining many inputs from distinct time steps. E.g., Sentiment analysis and emotion identification

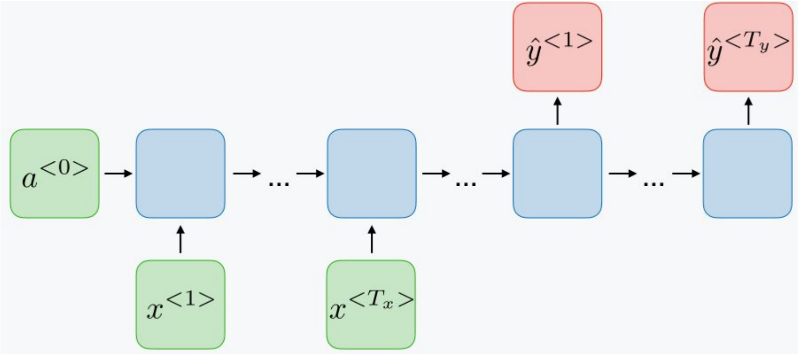


Types of RNN

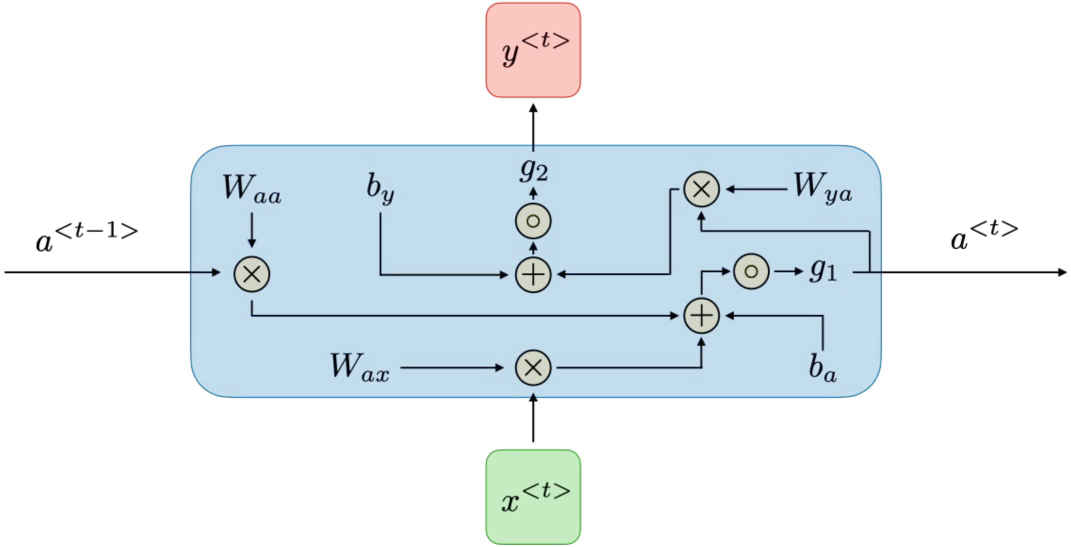
- **Many to Many**: Each single input has an output. e.g., Machine Translation.



- **Many To Many:** Multiple sequence of outputs from multiple sequence of inputs.



Forward propagation

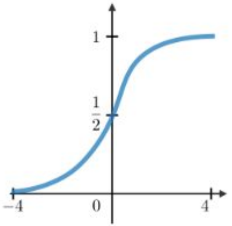
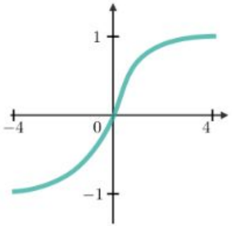
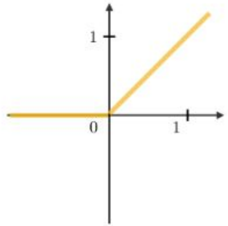


For each time step t, the activation a<t> and the output y<t> is expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \qquad \hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Forward propagation and Loss Functions

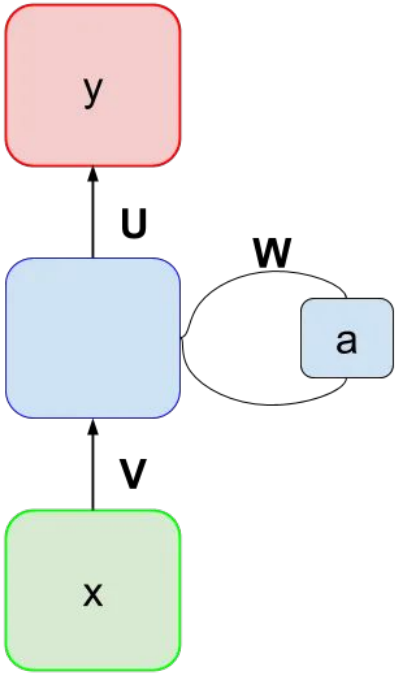
In our model, g1 usually is **Tanh** or **ReLU** and g2 is **sigmoid** or **Softmax** (depends on how variables you do like to identify)

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

In the case of a recurrent neural network, the loss function L of all time steps is defined based on the loss at every time step as follows:

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} E^{(t)} \qquad E^{(t)} = L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Backward propagation



We know:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Let's define:

$$q^{<t>} = Va^{<t>} + b_y$$

$$z^{<t>} = Wa^{<t-1>} + Ux^{<t>} + b_a$$

We have:

$$a^{<t>} = g_1(z^{<t>})$$

$$\hat{y}^{<t>} = g_2(q^{<t>})$$

Backward propagation

At timestep T, the derivative of the loss L with respect to some weight matrix M is expressed as follows:

$$\frac{\partial L^{(T)}}{\partial M} = \sum_{t=1}^{T} \frac{\partial E^{(T)}}{\partial M}\Big|_{(t)}$$

We can rewrite as (using U, W, V):

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T_y} \frac{\partial E^{(t)}}{\partial U}\Big|_{(t)} \qquad \frac{\partial L}{\partial W} = \sum_{t=1}^{T_y} \frac{\partial E^{(t)}}{\partial W}\Big|_{(t)} \qquad \frac{\partial L}{\partial V} = \sum_{t=1}^{T_y} \frac{\partial E^{(t)}}{\partial V}\Big|_{(t)}$$

Where:

$$\frac{\partial E^{(t)}}{\partial U} = (\hat{y}^{<t>} - y^{<t>}).V.\sum_{k=0}^{t}[\frac{\partial a^{<t>}}{\partial a^{<k>}}\frac{\partial a^{<k>}}{\partial z^{<k>}}.(x^{<k>})^T]$$

$$\frac{\partial E^{(t)}}{\partial W} = (\hat{y}^{<t>} - y^{<t>}).V.\sum_{k=0}^{t}[\frac{\partial a^{<t>}}{\partial a^{<k>}}\frac{\partial a^{<k>}}{\partial z^{<k>}}.(a^{<k-1>})^T]$$

$$\frac{\partial E^{(t)}}{\partial V} = (\hat{y}^{<t>} - y^{<t>}).(a^{<t>})^T$$

Vanishing gradient problem

The reason why they happen is that it is difficult to capture long term dependencies

$$\frac{\partial a^{<t>}}{\partial a^{<k>}} = \frac{\partial a^{<t>}}{\partial a^{<t-1>}} \frac{\partial a^{<t-1>}}{\partial a^{<t-2>}} \cdots \frac{\partial a^{<k+2>}}{\partial a^{<k+1>}} \frac{\partial a^{<k+1>}}{\partial a^{<k>}}$$

$$\frac{\partial a^{<t>}}{\partial a^{<k>}} = \prod_{i=k+1}^{t} \frac{\partial a^{<i>}}{\partial a^{<i-1>}} \qquad \frac{\partial a^{<t>}}{\partial a^{<k>}} = \prod_{i=k+1}^{t} W^T diag[\frac{\partial g_1(a^{<i-1>})}{\partial a^{<i-1>}}]$$

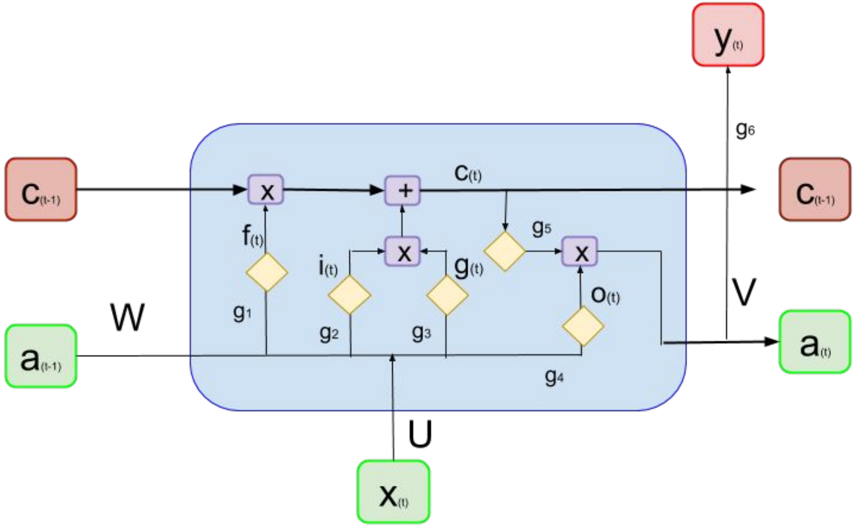
Taking non-linear functions to analyze, we obtain:

$$\|diag[\frac{\partial g_1(a^{<i-1>})}{\partial a^{<i-1>}}]\| \leq \gamma \qquad \|\frac{\partial a^{<i>}}{\partial a^{<i-1>}}\| \leq \|W^T\| \|diag[\frac{\partial g_1(a^{<i-1>})}{\partial a^{<i-1>}}]\| \leq \gamma_w \cdot \gamma$$

$$\|\frac{\partial a^{<t>}}{\partial a^{<k>}}\| \leq (\gamma_w \cdot \gamma)^{(t-k)} = (\lambda)^{(t-k)}$$

If lambda << 1, Then Vanishing Gradient. Otherwise, lambda >1, Then Exploding Gradient.

Models of RNN: Long Short Term Memory (LSTM)



$$a^{<t>} = o^{<t>} \circ g_5(c^{<t>})$$

$$\hat{y}^{<t>} = g_6(Va^{<t>} + b_y)$$

Forget gate:

$$f^{<t>} = g_1(W_f a^{<t-1>} + U_f x^{<t>} + b_f)$$

Input gate:

$$i^{<t>} = g_2(W_i a^{<t-1>} + U_i x^{<t>} + b_i)$$

Update gate: Candidate

$$g^{<t>} = g_3(W_c a^{<t-1>} + U_c x^{<t>} + b_c)$$

Update gate: Memory

$$c^{<t>} = f^{<t>} \circ c^{<t-1>} + i^{<t>} \circ g^{<t>}$$

Output gate:

$$o^{<t>} = g_4(W_o a^{<t-1>} + U_o x^{<t>} + b_o)$$

Models of RNN: LSTM backpropagation

$$p^{<t>} = g_5(c^{<t>}) \qquad s^{<t>} = W_o a^{<t-1>} + U_o x^{<t>} + b_o$$

$$\frac{\partial E^{(t)}}{\partial V} = (\hat{y}^{<t>} - y^{<t>}).(a^{<t>})^T \quad \frac{\partial L}{\partial V} = \sum_{t=1}^{T_y}[(\hat{y}^{<t>} - y^{<t>}).(a^{<t>})^T]$$

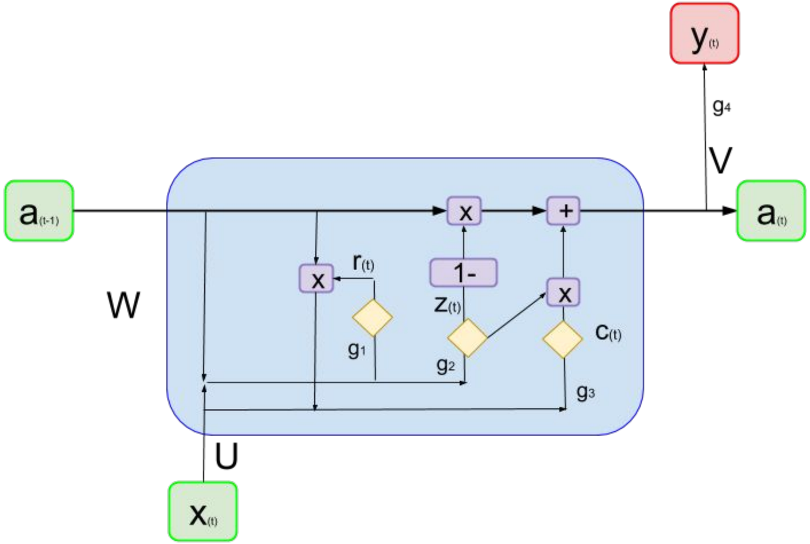
$$\frac{\partial E^{(t)}}{\partial W_o} = (\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).p^{<t>}.\frac{\partial o^{<t>}}{\partial s^{<t>}}.(a^{<t-1>})^T$$

$$\frac{\partial L}{\partial W_o} = \sum_{t=1}^{T_y}[(\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).p^{<t>}.\frac{\partial o^{<t>}}{\partial s^{<t>}}.(a^{<t-1>})^T]$$

$$\frac{\partial E^{(t)}}{\partial U_o} = (\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).p^{<t>}.\frac{\partial o^{<t>}}{\partial s^{<t>}}.(x^{<t>})^T$$

$$\frac{\partial L}{\partial U_o} = \sum_{t=1}^{T_y}[(\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).p^{<t>}.\frac{\partial o^{<t>}}{\partial s^{<t>}}.(x^{<t>})^T]$$

Models of RNN: Gated Recurrent Unit (GRU)



Update gate:

$$z^{<t>} = g_1(W_z a^{<t-1>} + U_z x^{<t>} + b_z)$$

Reset gate:

$$r^{<t>} = g_2(W_r a^{<t-1>} + U_r x^{<t>} + b_r)$$

Candidate gate:

$$c^{<t>} = g_3(W_c(r^{<t>} \circ a^{<t-1>}) + U_c x^{<t>} + b_c)$$

$$a^{<t>} = (1 - z^{<t>}) \circ a^{<t-1>} + z^{<t>} \circ c^{<t>}$$

$$\hat{y}^{<t>} = g_4(V a^{<t>} + b_y)$$

Models of RNN: GRU backpropagation

$$s^{<t>} = W_c(r^{<t>} \circ a^{<t-1>}) + U_c x^{<t>} + b_c$$

$$\frac{\partial E^{(t)}}{\partial V} = (\hat{y}^{<t>} - y^{<t>}).(a^{<t>})^T \qquad \frac{\partial L}{\partial V} = \sum_{t=1}^{T_y}[(\hat{y}^{<t>} - y^{<t>}).(a^{<t>})^T]$$

$$\frac{\partial E^{(t)}}{\partial W_c} = (\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).z^{<t>}\frac{\partial c^{<t>}}{\partial s^{<t>}}.(r^{<t>} \circ a^{<t-1>})^T$$

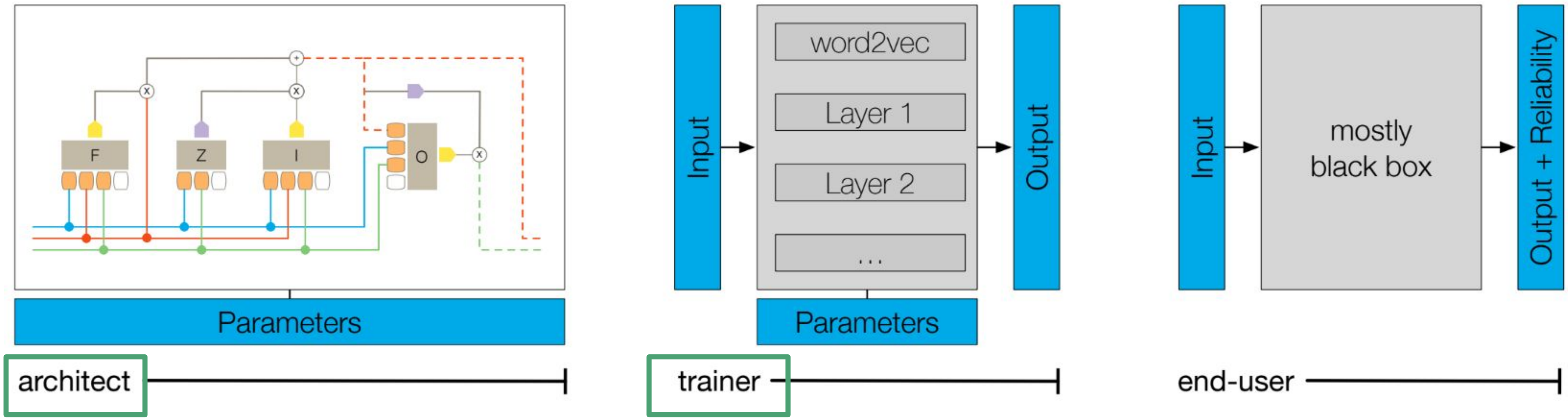
$$\frac{\partial L}{\partial W_c} = \sum_{t=1}^{T_y}[(\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).z^{<t>}\frac{\partial c^{<t>}}{\partial s^{<t>}}.(r^{<t>} \circ a^{<t-1>})^T]$$

$$\frac{\partial E^{(t)}}{\partial U_c} = (\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).z^{<t>}\frac{\partial c^{<t>}}{\partial s^{<t>}}.(x^{<t>})^T$$

$$\frac{\partial L}{\partial U_c} = \sum_{t=1}^{T_y}[(\frac{\partial E^{(t)}}{\partial a^{<t>}} + \frac{\partial E^{(t+1)}}{\partial a^{<t>}}).z^{<t>}\frac{\partial c^{<t>}}{\partial s^{<t>}}.(x^{<t>})^T$$

LSTMVis

LSTMVis: Point of view/interest



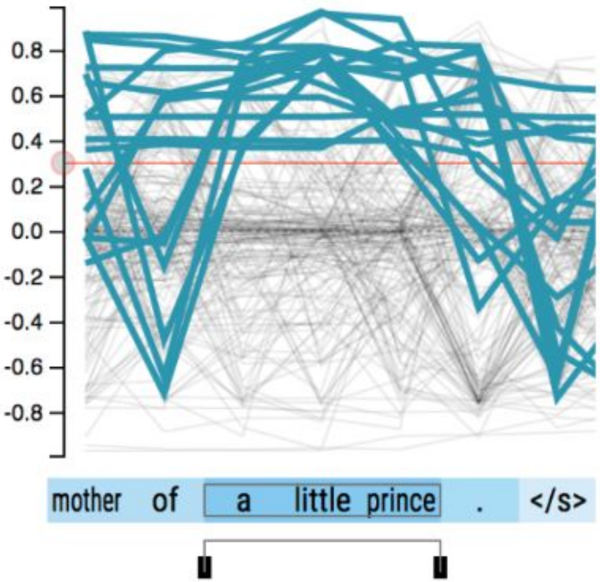
- The **architect** analyzes and modifies all components of the system.
- The **trainer** abstracts the model to the main components/parameters focusing on training on different data sets.
- The **end user** has the most abstract view on the model and considers whether the output is coherent for a given input.

LSTMVis: General View

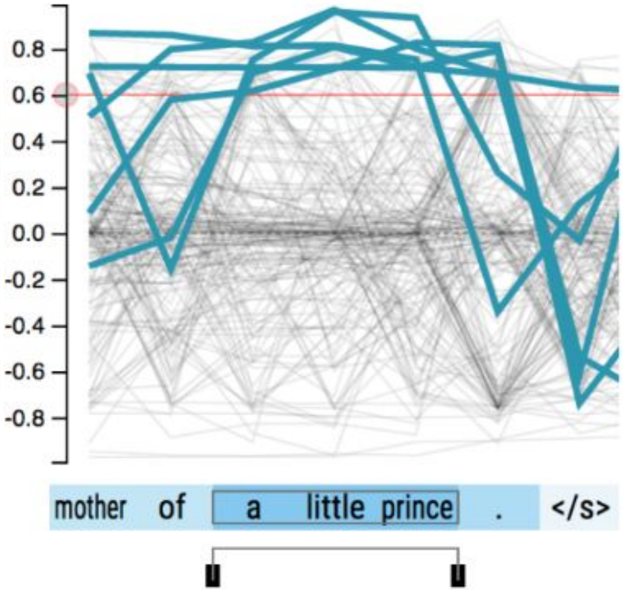
LSTMVis: Select View

LSTMVis: Select View

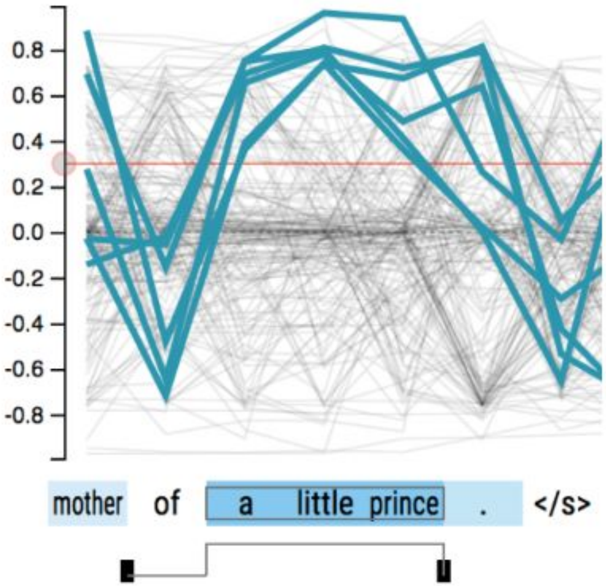
Threshold 0.3



Threshold 0.6



eliminate hidden states with values above L after reading “of”



LSTMVis: Match View

LSTMVis: Activation View

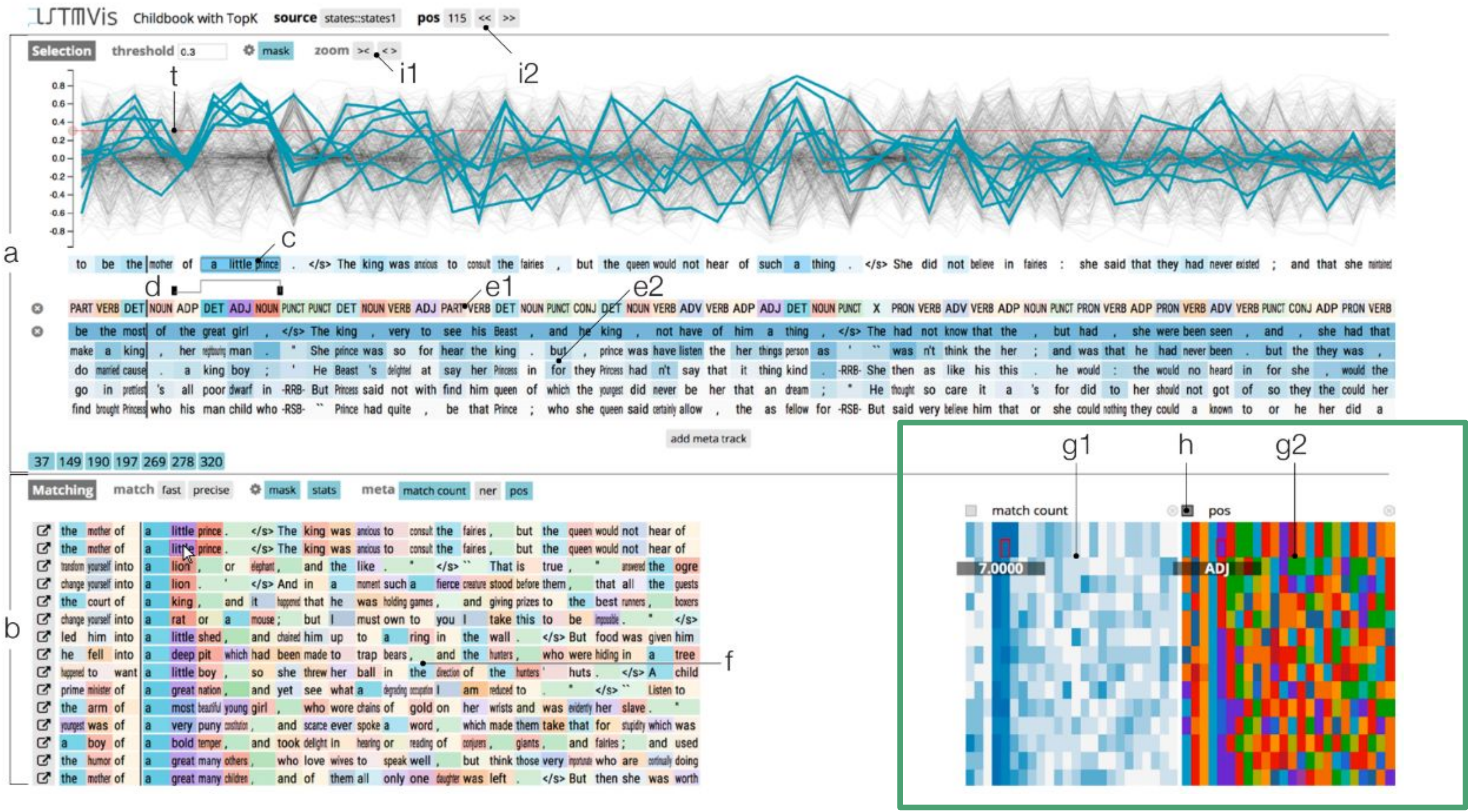
LSTMVis: Pattern plot

LSTMVis: Part-Of-Speech (POS)

LSTMVis: Top K predictions

LSTMVis: Word Matrix

LSTMVis: Encoded meta-data



LSTMVis: Encode color by POS

LSTMVis: Word-with

LSTMVis: Move timeline forward or backward

Case Use

LSTMVis: Parenthesis language

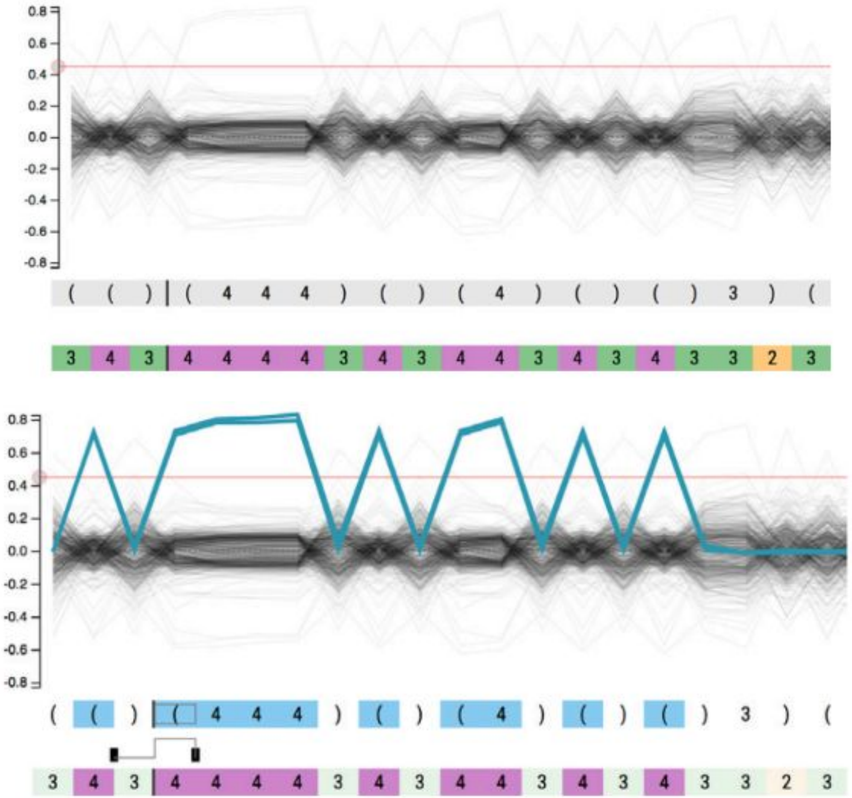
Synthetic data and alphabet: match parenthesis and nesting limited to 4 levels.

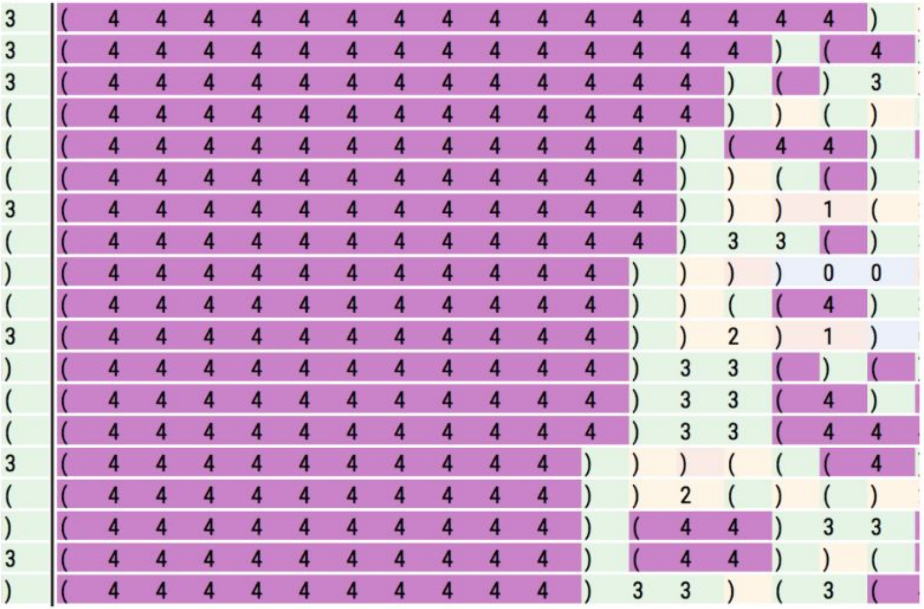
$$\Sigma = \{ (\ )\ 0\ 1\ 2\ 3\ 4 \}$$

Numbers are generated randomly, but are constrained to indicate the nesting level at their position.

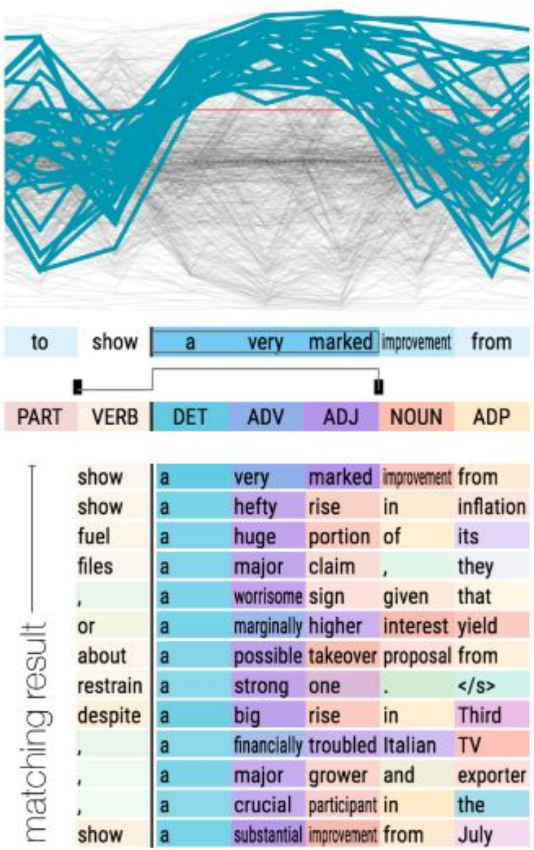
```
( 1 ( 2 ) ( ) ) 0 ( ( ( 3 ) ) 1 )
```

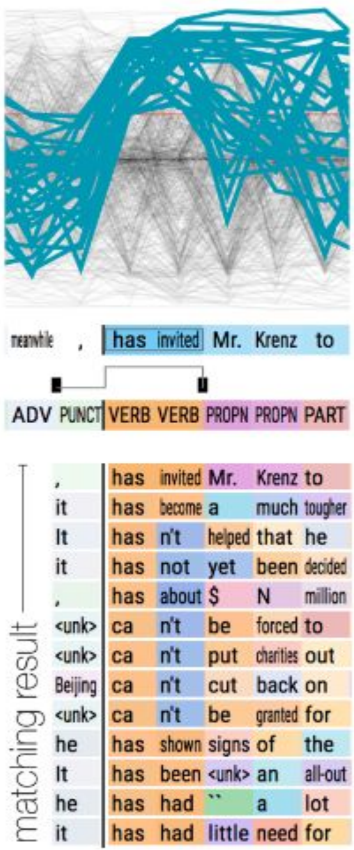
LSTMVis: Parenthesis language



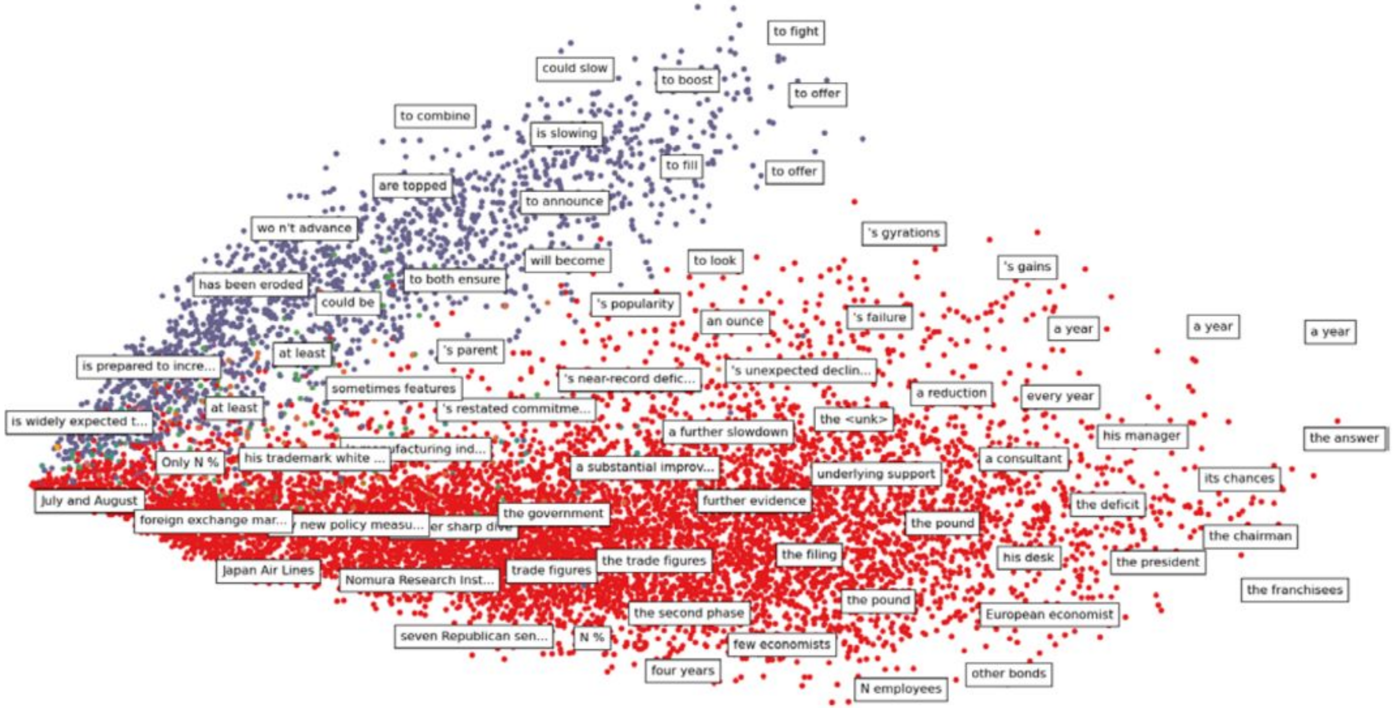


LSTMVis: Phrase Separation in Language Modeling



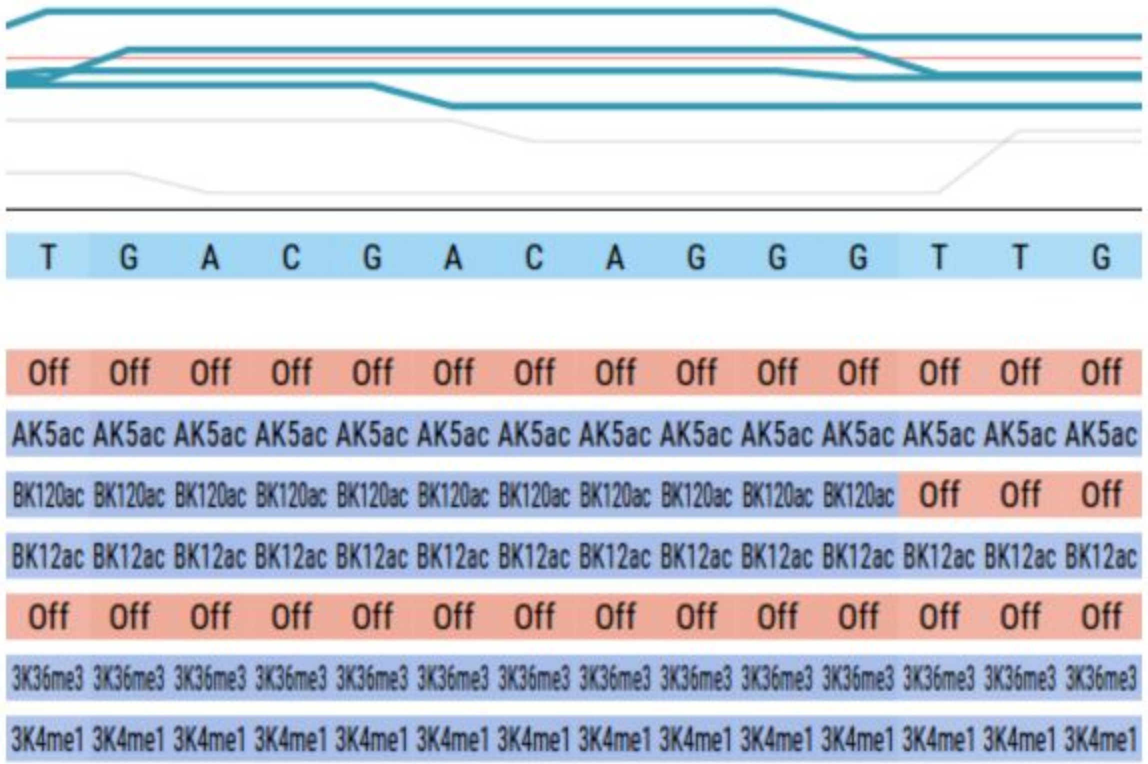


LSTMVis: Phrase Separation in Language Modeling

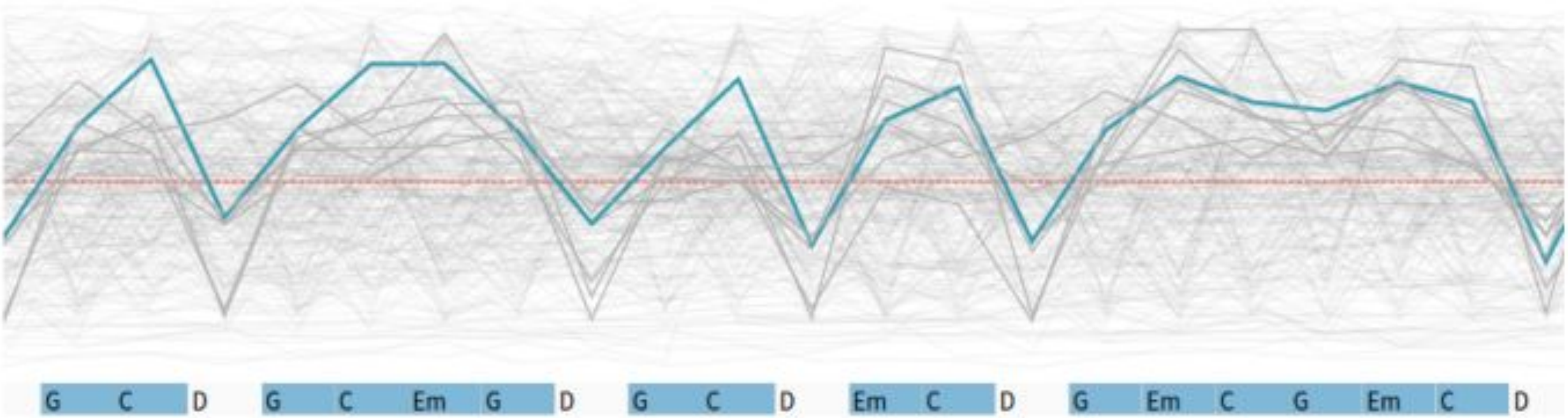


Red points indicate noun phrases, blue points indicate verb phrases, other colors indicate remaining phrase types.

LSTMVis: Biological sequence analysis



LSTMVis: Musical chord progressions



“Don’t Stop Believing” song

References

- LSTMVis
- RNN review