Recurrent Neural Network

MUKESH KUMAR

AGENDA

- Understanding Sequential Data
- Working with Sequential Data
- Understanding Recurrent Neural Network
- Long Short Term Memory (LSTM)
- Working with Time Series Data
- Sequence to Sequence Models
- Attention Mechanism
- Gated Recurrent Unit (GRU)

Understanding Sequential Data

Guess the next word

The Sky is ___?

Exact same words but different meaning

I had fixed my laptop

I had my laptop fixed

Why???

Sequential Data

- Order of data points is important
- Number of data points in a sequence can vary

Example of Sequential Data

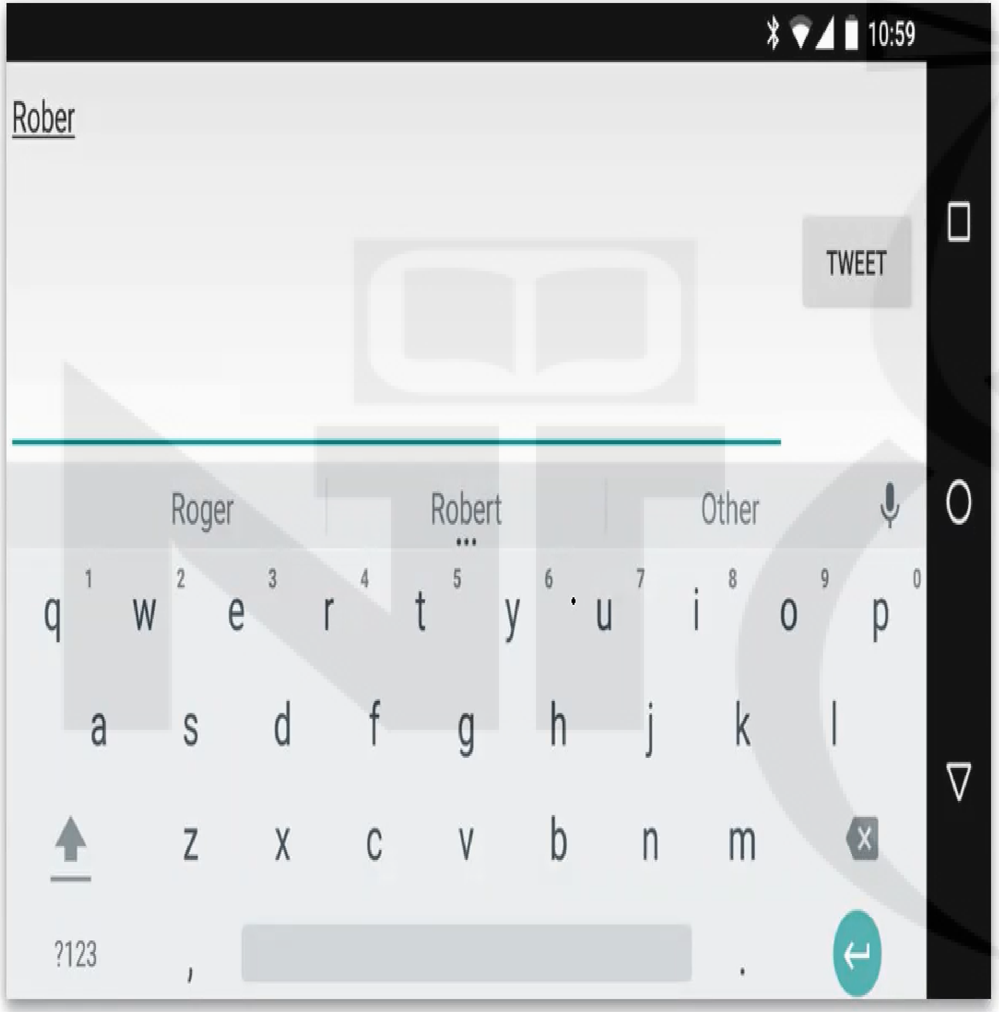
Music

Text

DNA sequence

Voice

Time related data

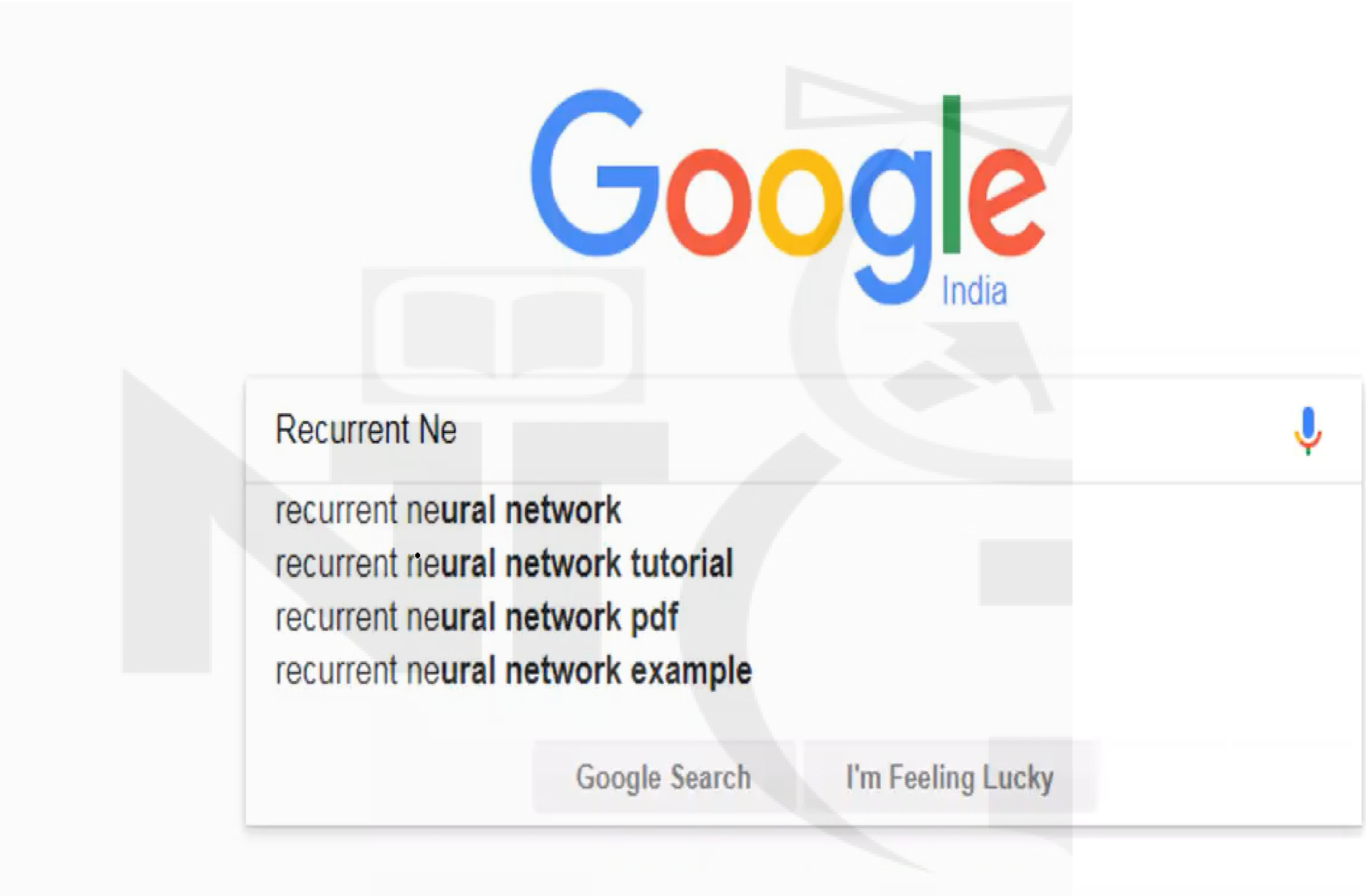


Mobile phone keyboard

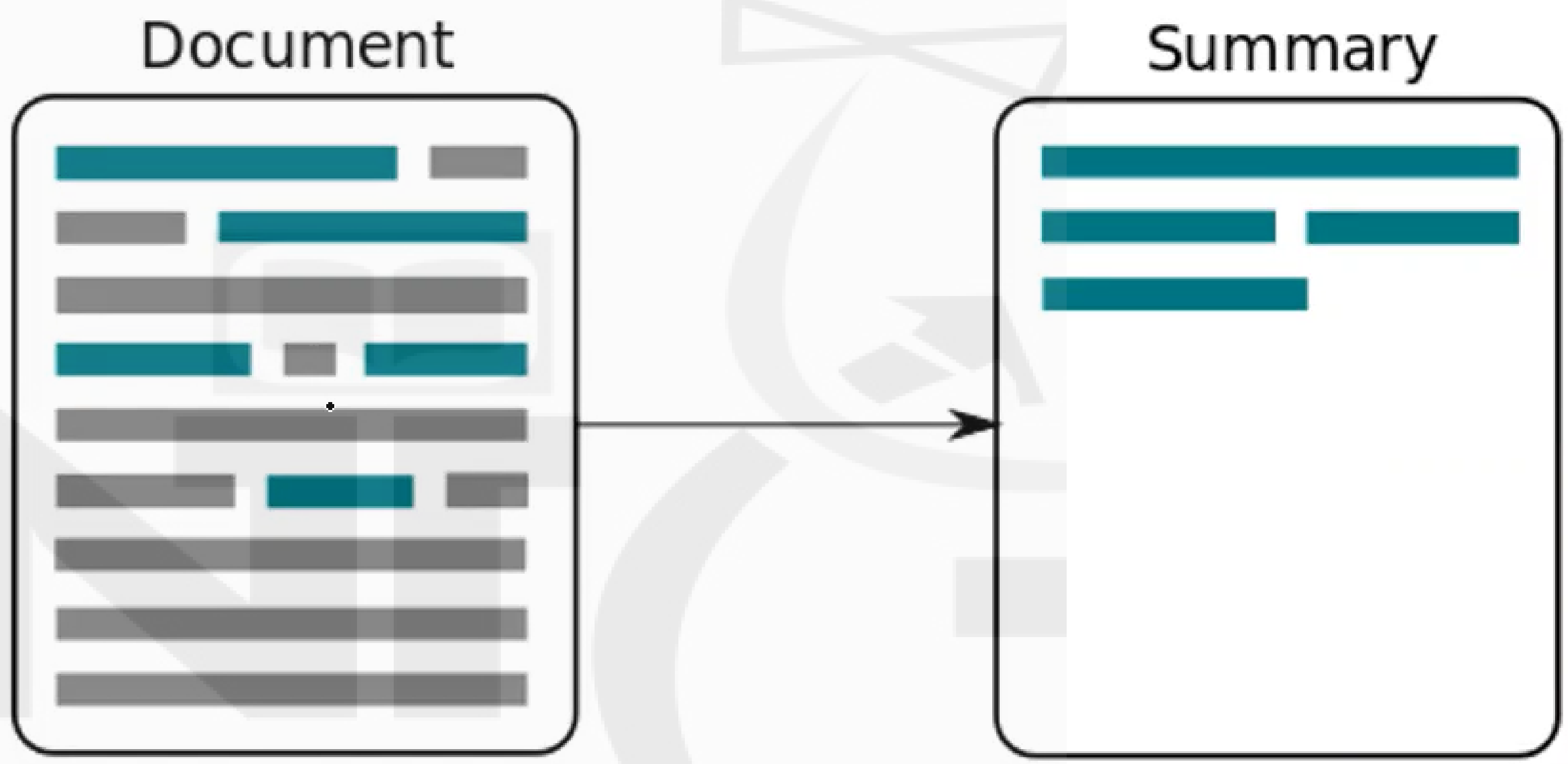
Predict next word(s) as we type in...

Convert Speech to Text

e.g Alexa, Siri, Google Home



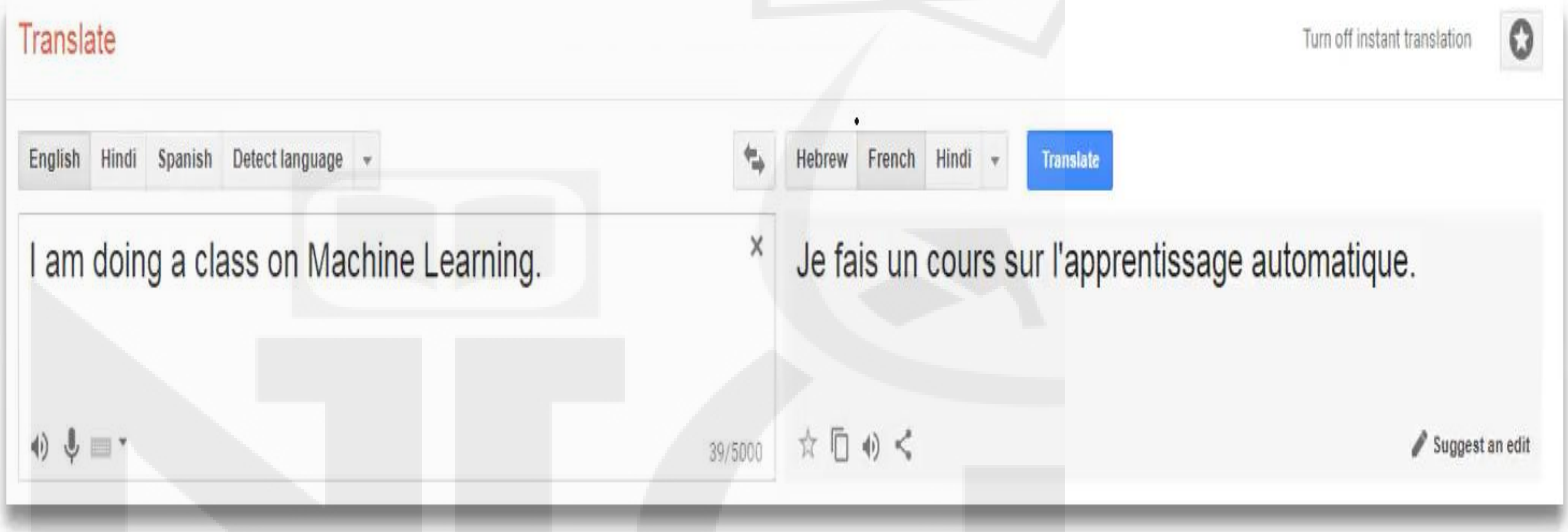
Search suggestions



Text Summarization

Describe a picture

Tendulkar playing cricket



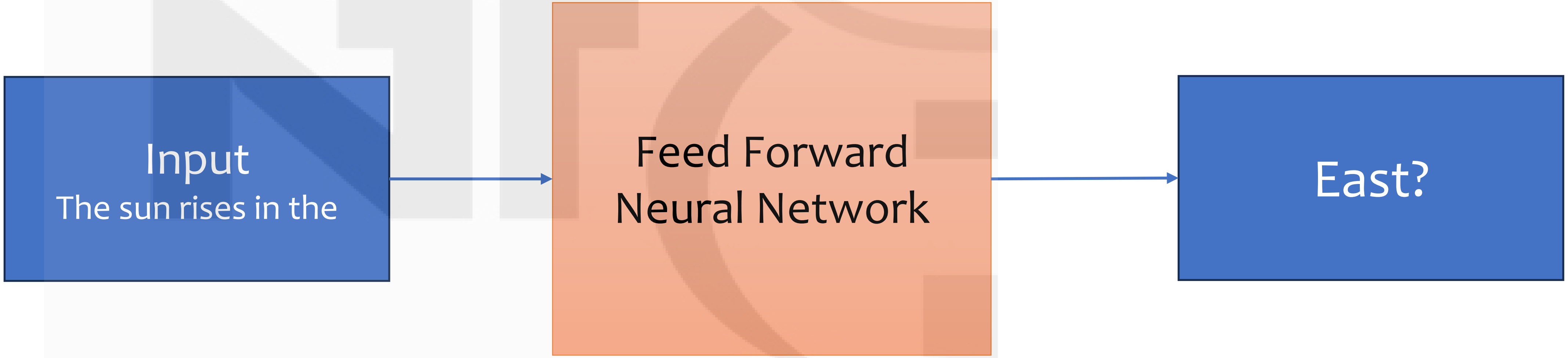
Language Translation

Demand forecasting

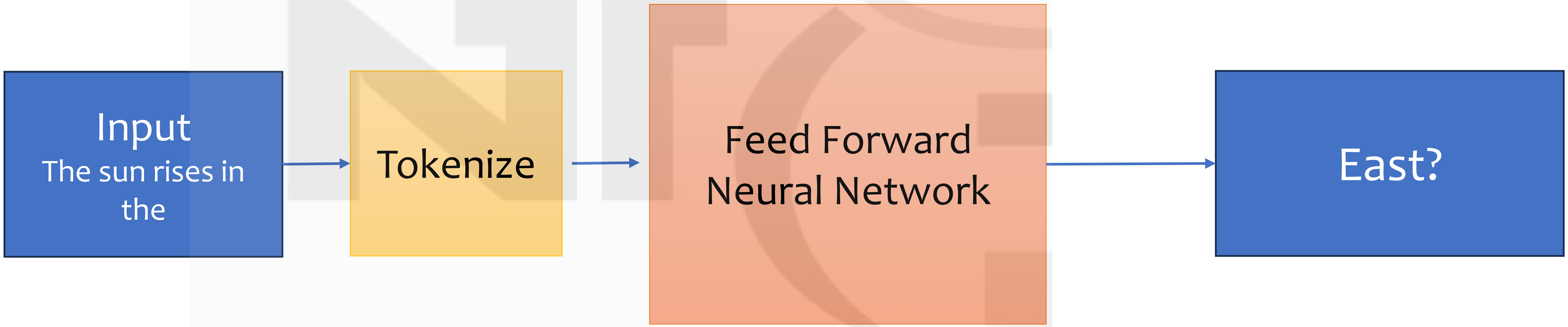
How much to build

How do we make machine understand….. Sequential data???

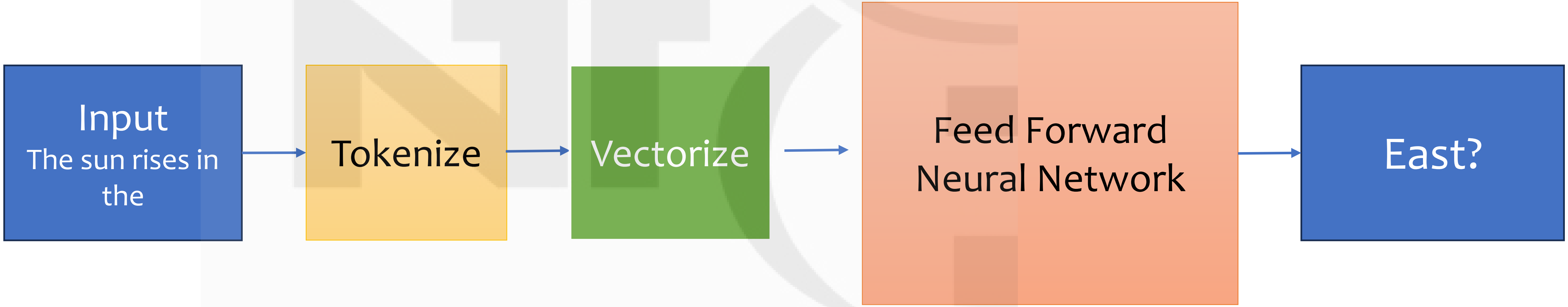
Let’s try Word2Vec model



Feed Forward Dense Neural Network



Feed Forward Dense Neural Network



Feed Forward Dense Neural Network

What approaches available for **vectorization?**

What approaches available for **vectorization?**

1. **Count Vectorizer**
2. **TF-IDF**
3. **One hot encoding**
4. **Word2Vec**

What approaches will work with Sequential Data?

1. **Count Vectorizer**
2. **TF-IDF**
3. **One hot encoding**
4. **Word2Vec**

What approaches will work with Sequential Data?

1. **Count Vectorizer** ✕
2. **TF-IDF** ✕
3. **One hot encoding**
4. **Word2Vec**

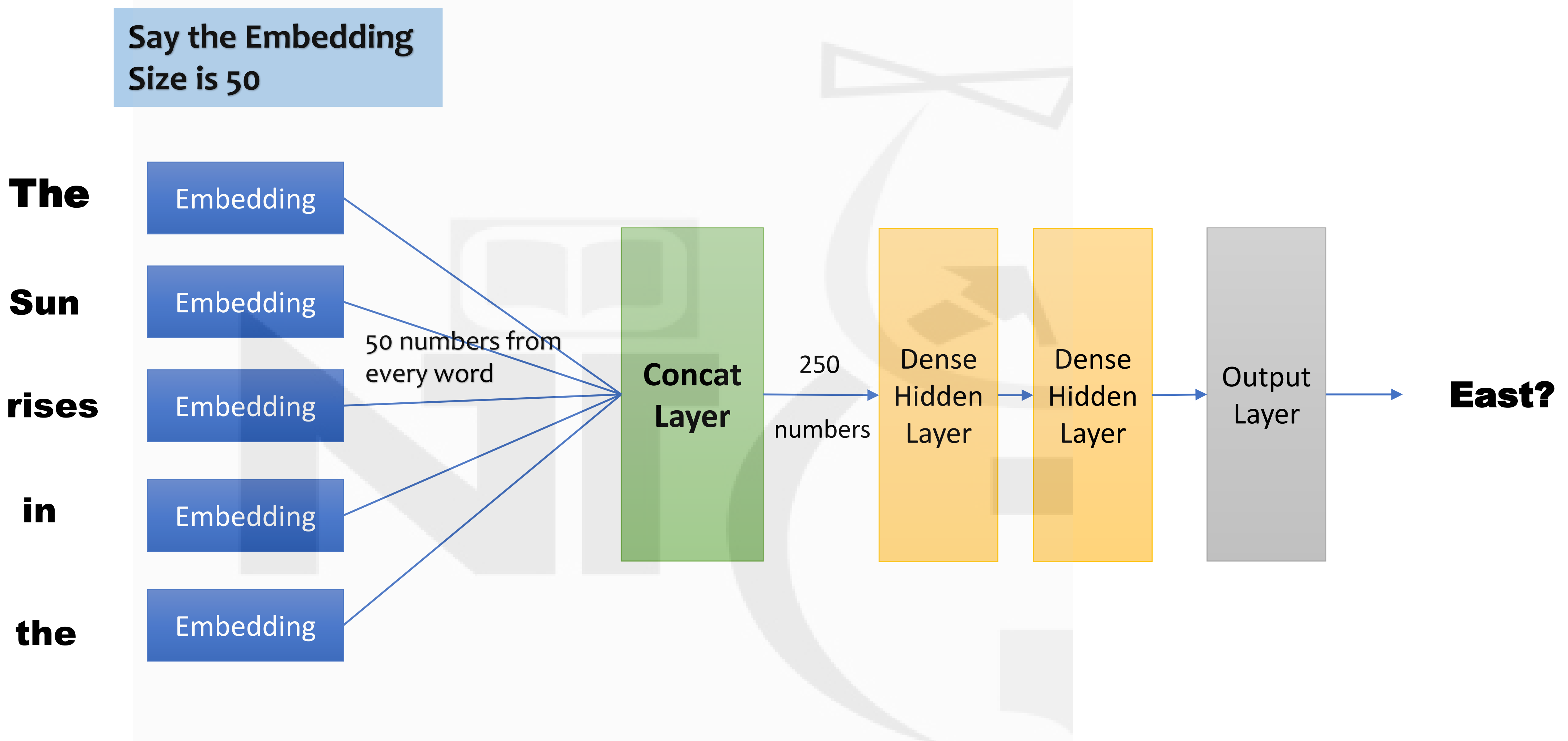
What approaches will work with Sequential Data?

1. **Count Vectorizer** ✗
2. **TF-IDF** ✗
3. **One hot encoding**
4. **Word2Vec**

Human Logic

ML Logic

Word2Vec is most popular approach for vectorization of sequential data

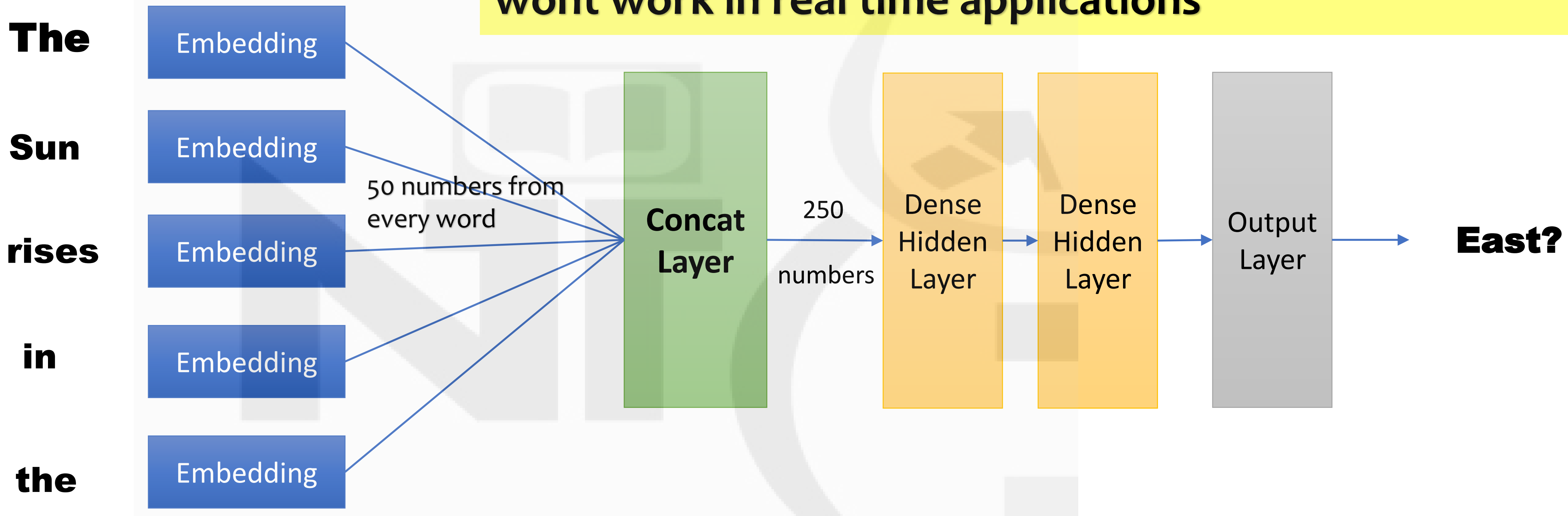


Feed Forward Network

with WordtoVec Embeddings

Say the Embedding Size is 50

When the number of words in the input change, we need to build a new model, this is not a generic model, wont work in real time applications



Feed Forward Network

with Word2Vec Embeddings

We need a new layer …

- Which can work with sequence of different lengths
- Number of weights should not change with number of inputs (words)

How do we build such a layer?

Why Dense layer uses more weights when sequence length increases?

Because a neuron looks at all the input words (features) in Dense layer at once

How do we solve this?

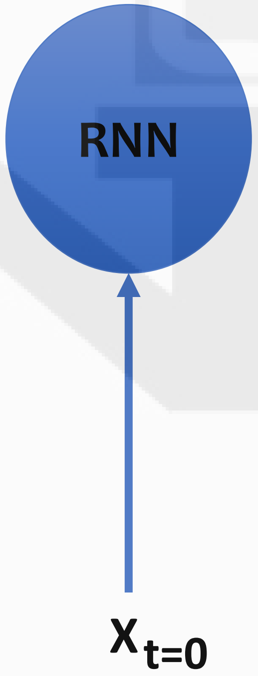
What if neuron looks at only one word at a time?

And remembers what it has looked so far?

This is the core idea behind RNN

RNN: A new kind of Neural Network

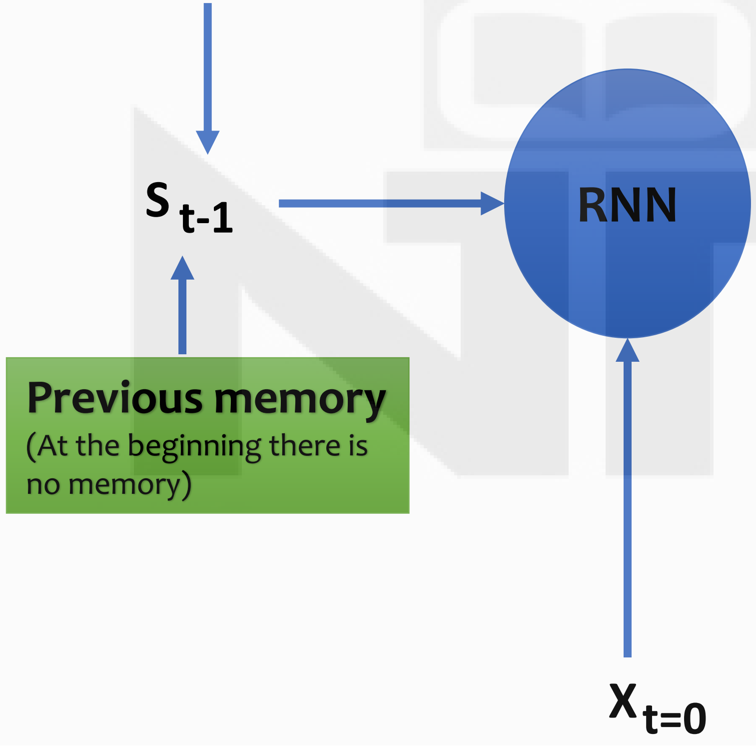
Can Remember Sequences and has memory



The input at step t=0

- RNN only looks at one input (one word) at a time

RNN uses its memory to
a sequence



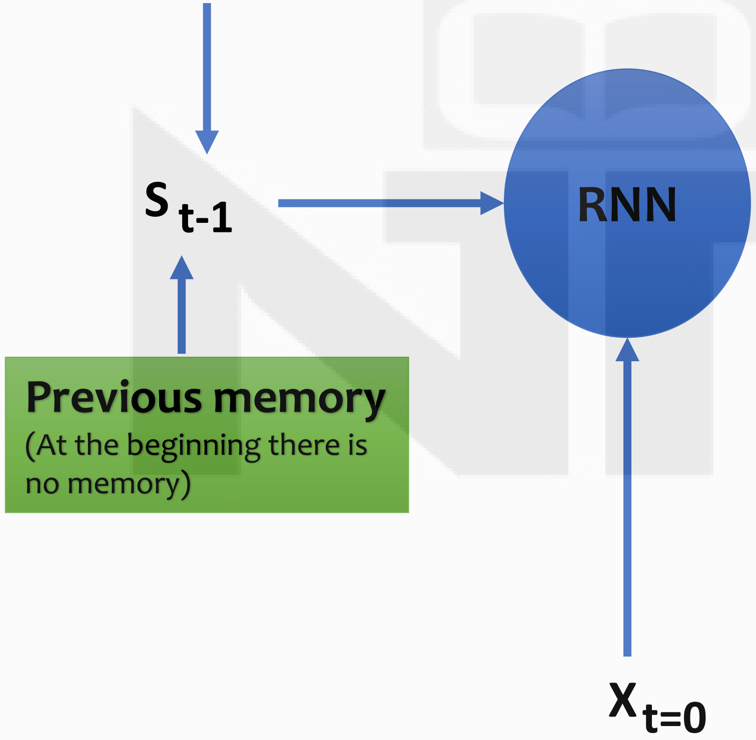
RNN only looks at one input (one word) at a time

The input at step t=0

Now , lets feed the following sequence :

“The sun rises in the “

RNN uses its memory to a sequence

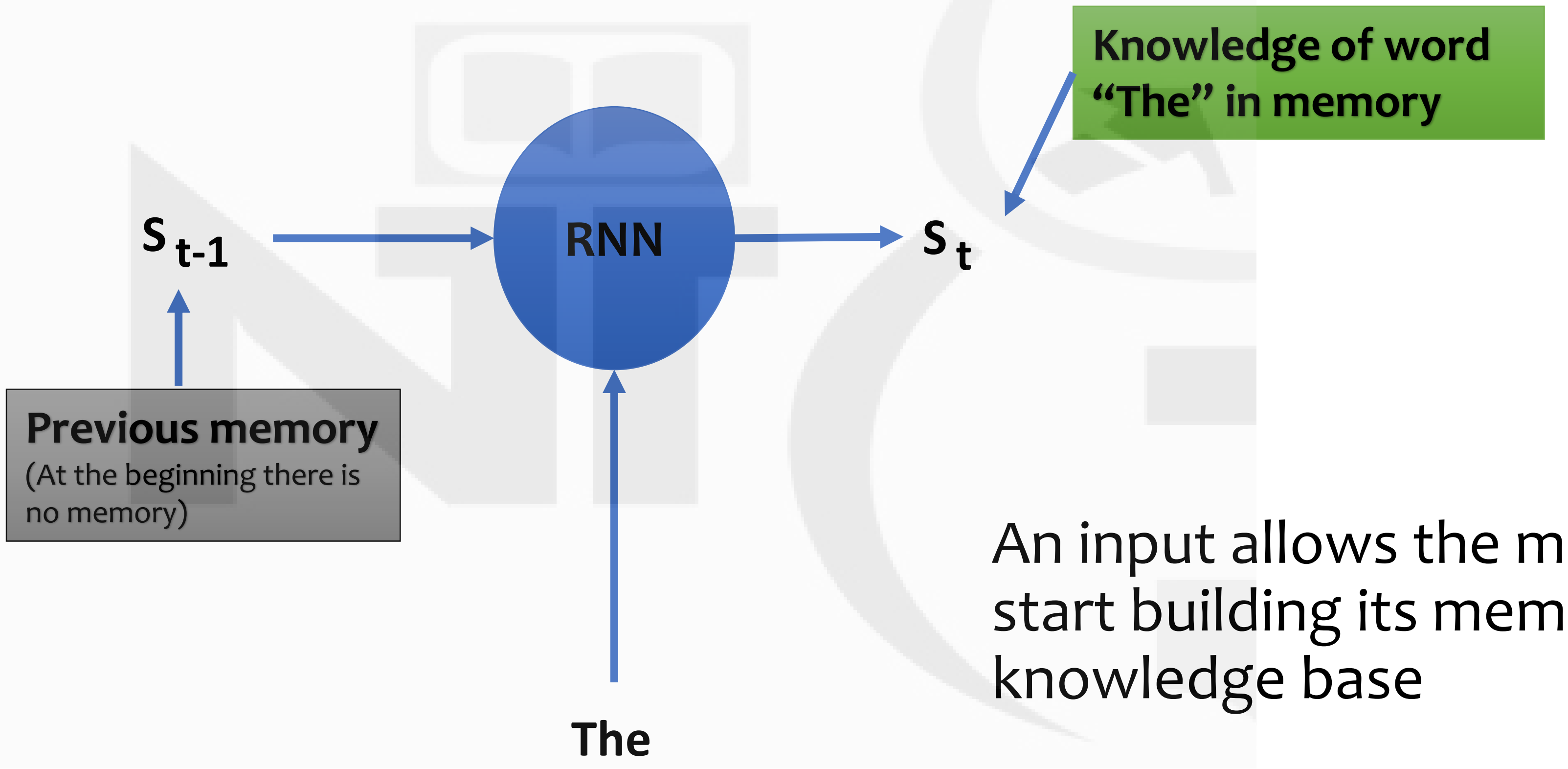


Inputs for RNN:

- A Word
- Previous Memory

The input at step t=0

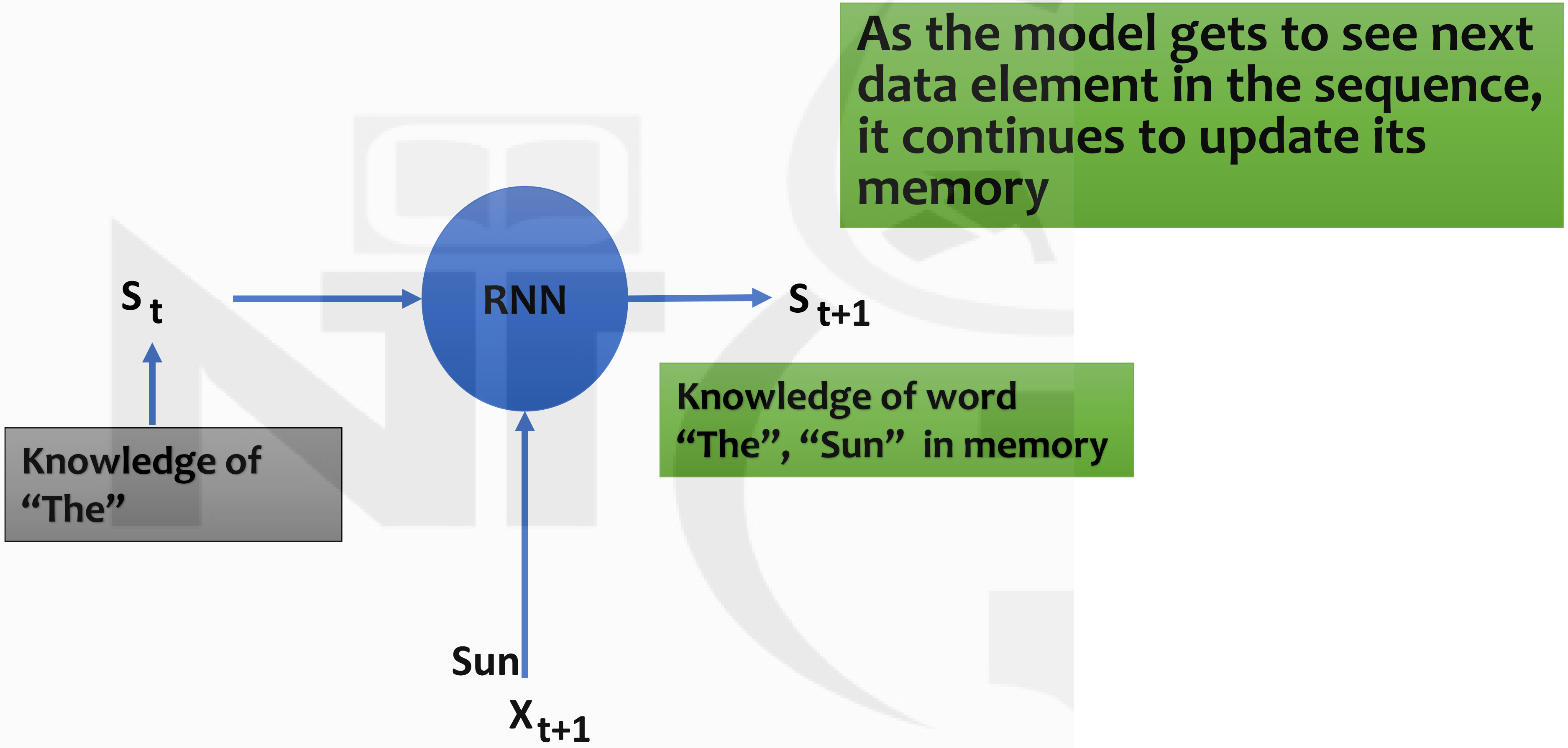
Feeding the first word "The"



The input at step t=0

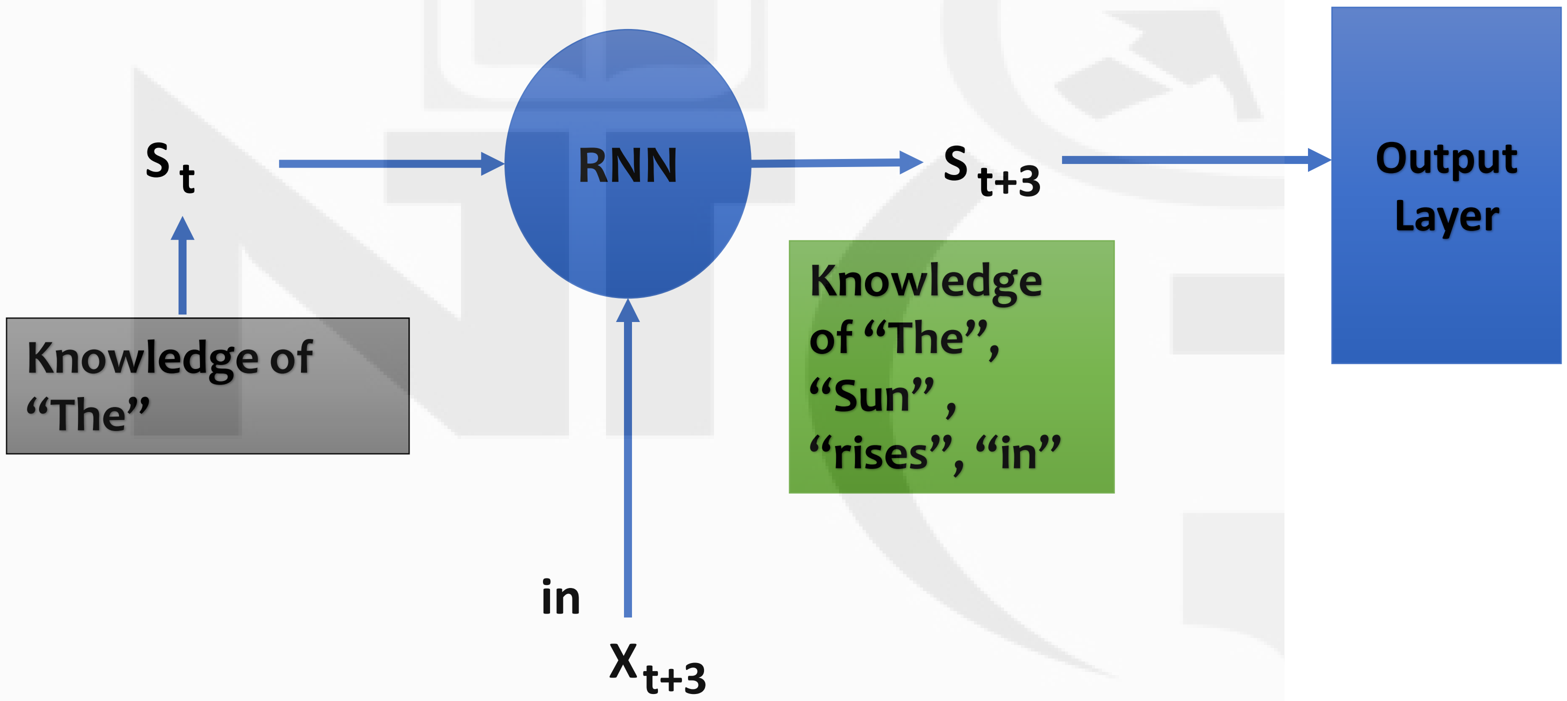
An input allows the model to start building its memory or knowledge base

Next word "Sun"



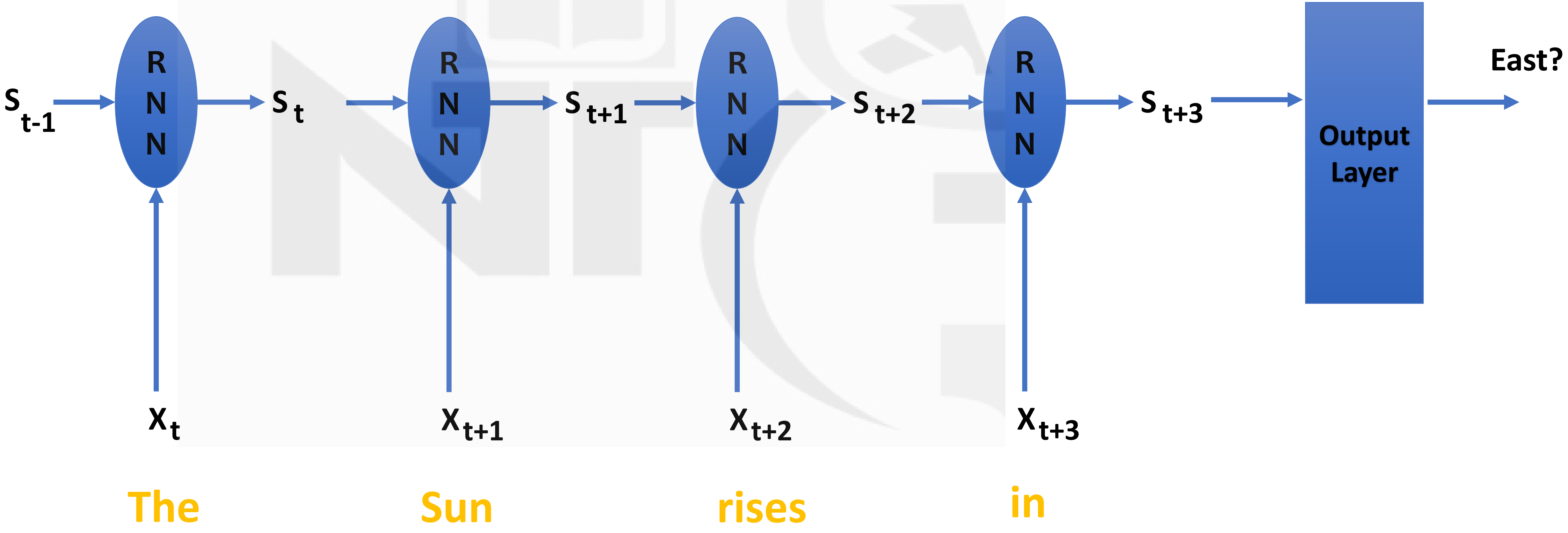
The input at step t+1

Once all the words have been fed, we can use the final memory state to generate the output through output layer

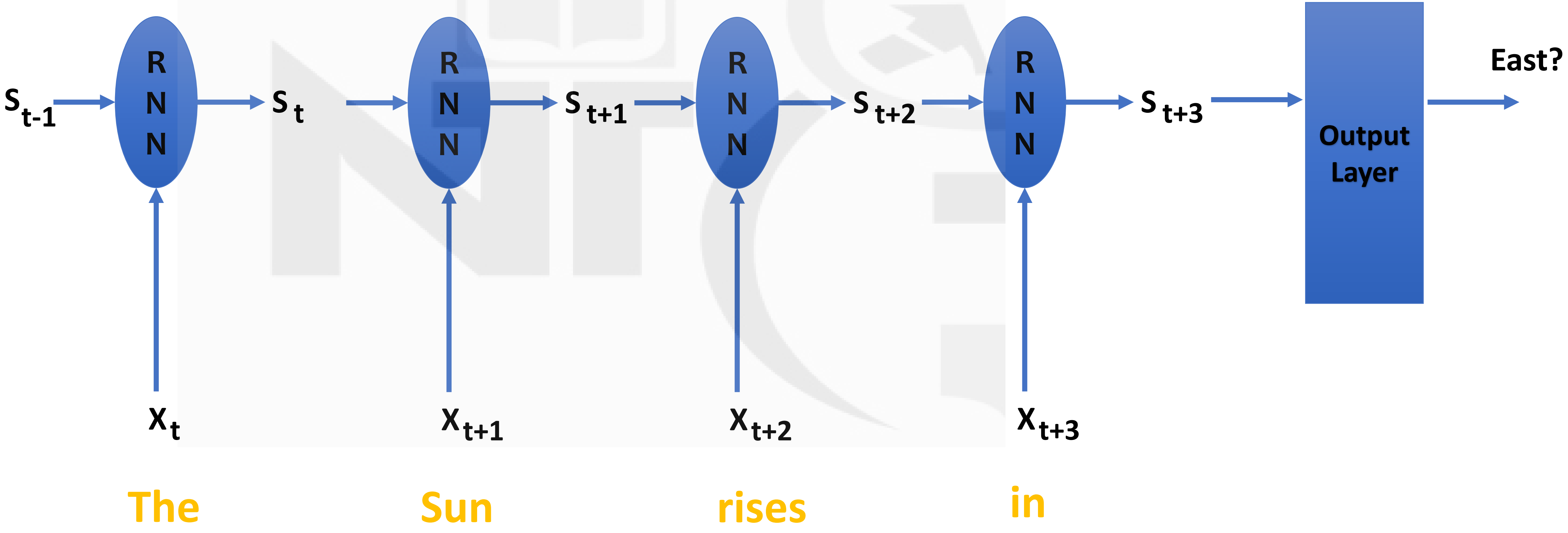


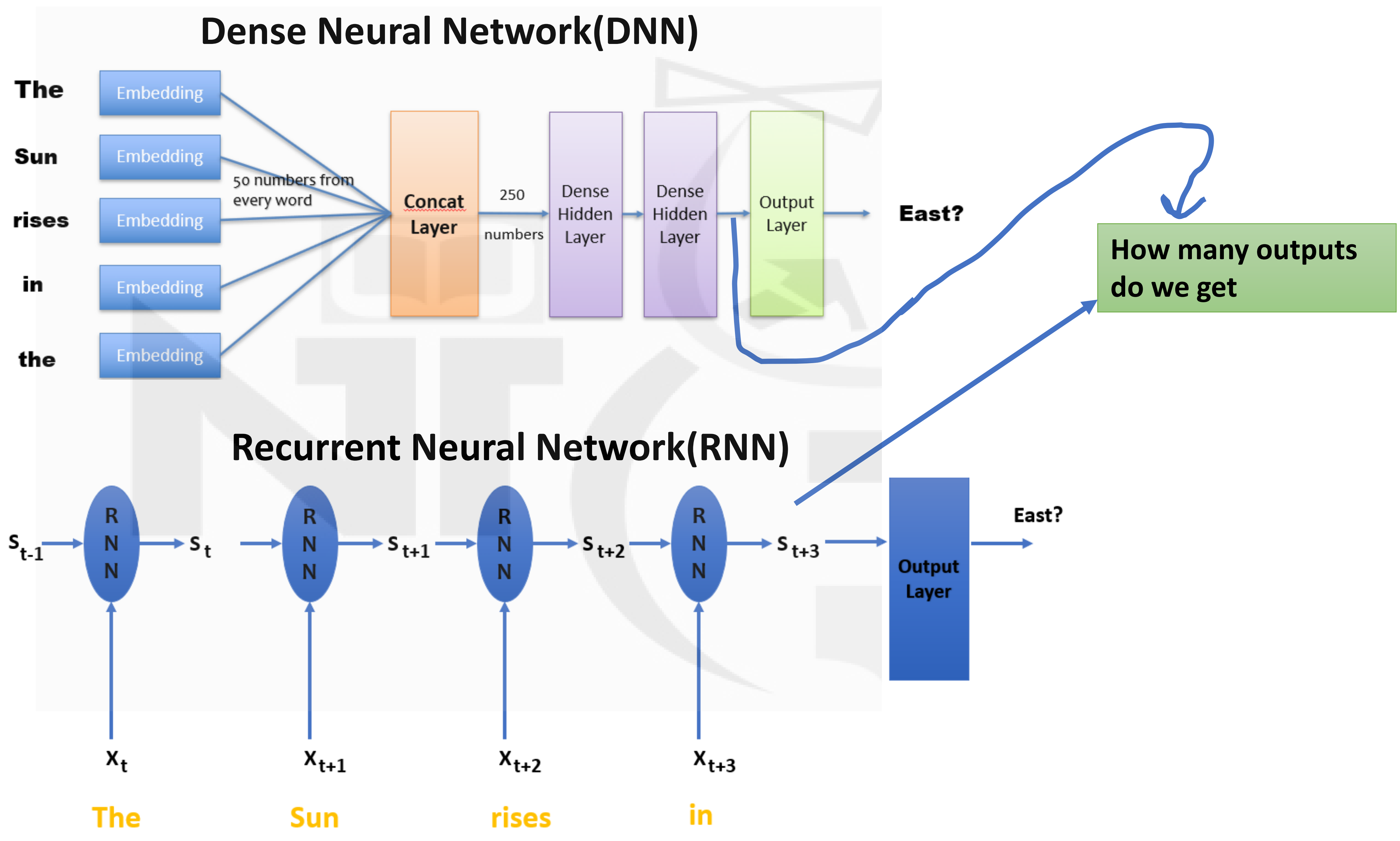
The input at step t+1

RNN over multiple time steps



In this picture how many rnns are there?





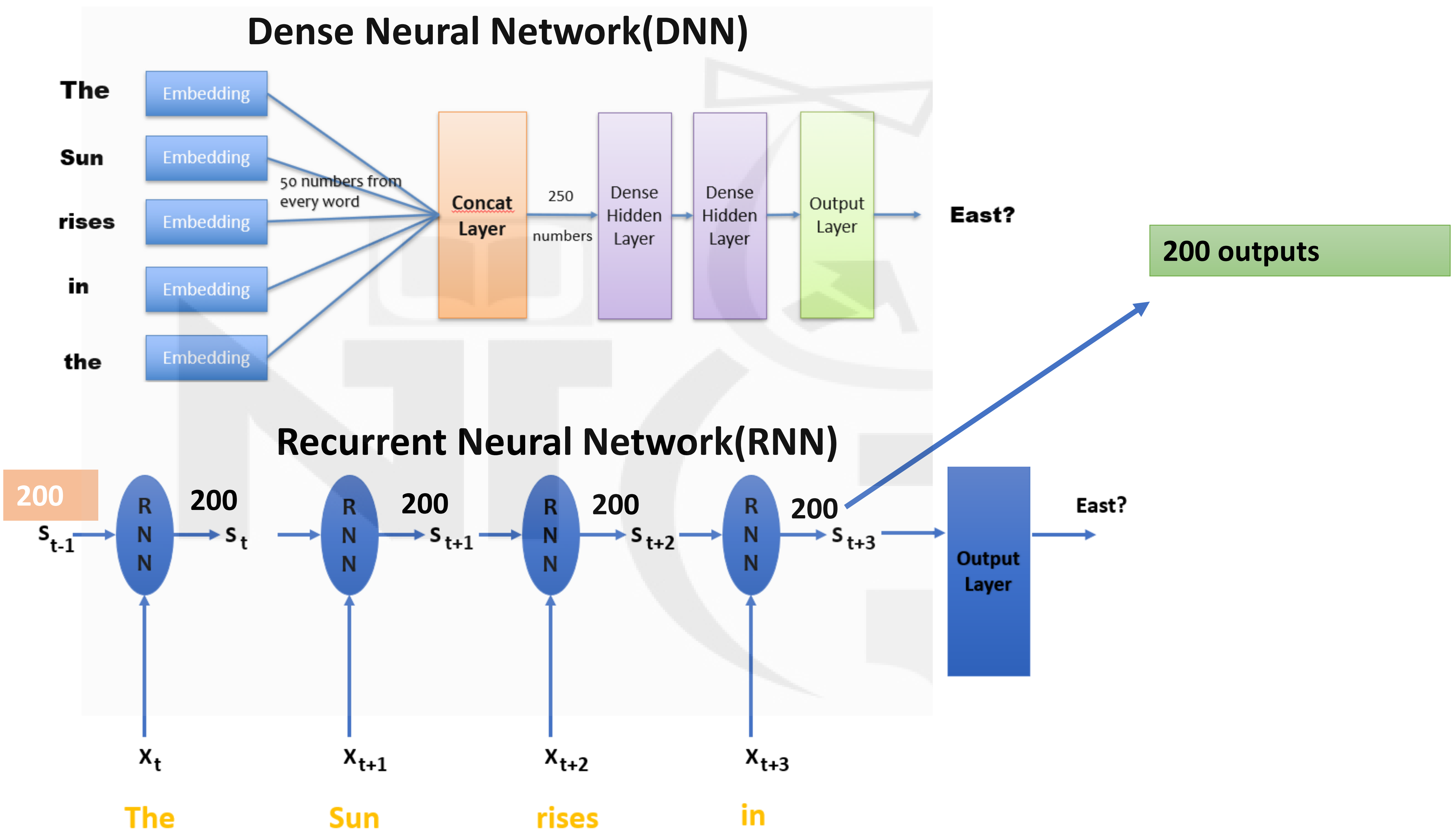
Dense Neural Network(DNN)
The
Sun
rises
in
the
Embedding
Embedding
Embedding
Embedding
Embedding
50 numbers from every word
Concat Layer
250 numbers
Dense Hidden Layer
Dense Hidden Layer
Output Layer
East?
How many outputs do we get
Recurrent Neural Network(RNN)
S_{t-1}
RNN
S_t
RNN
S_{t+1}
RNN
S_{t+2}
RNN
S_{t+3}
Output Layer
East?
X_t
X_{t+1}
X_{t+2}
X_{t+3}
The
Sun
rises
in

How many outputs do we get from RNN

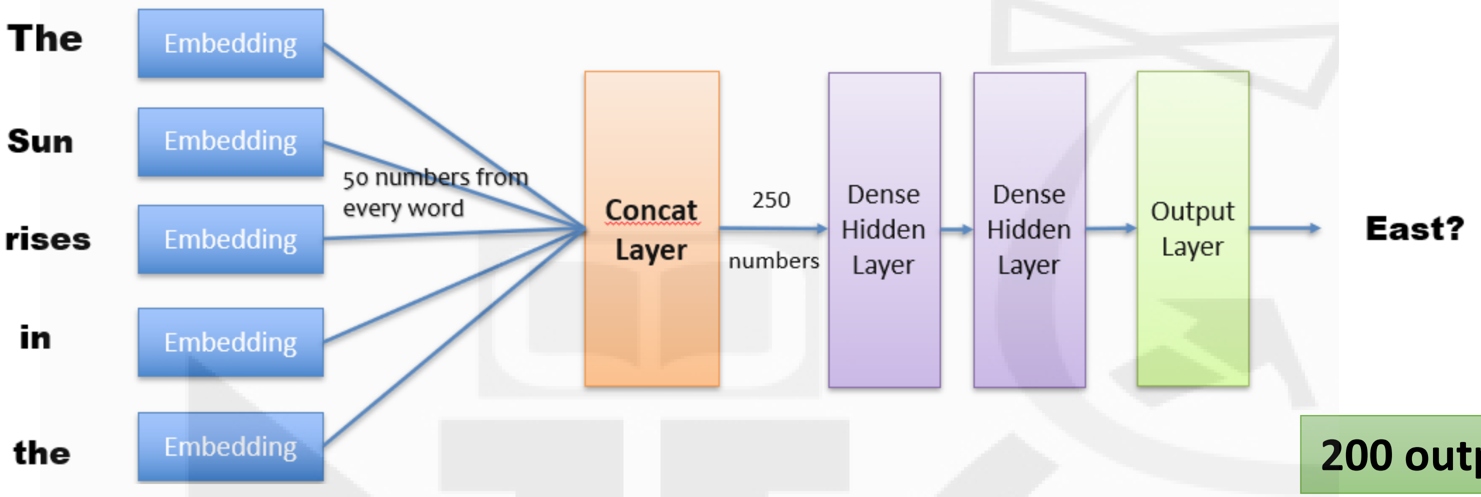
Output of RNN is the final updated memory

It's a hyperparameter (Size of RNN state or memory)

Let's say memory size is **200**


Dense Neural Network(DNN)
The
Sun
rises
in
the
Embedding
Embedding
Embedding
Embedding
Embedding
50 numbers from every word
Concat Layer
250 numbers
Dense Hidden Layer
Dense Hidden Layer
Output Layer
East?
200 outputs
Recurrent Neural Network(RNN)
200
RNN
RNN
RNN
RNN
200
200
200
200
s t-1
s t
s t+1
s t+2
s t+3
Output Layer
East?
x t
x t+1
x t+2
x t+3
The
Sun
rises
in

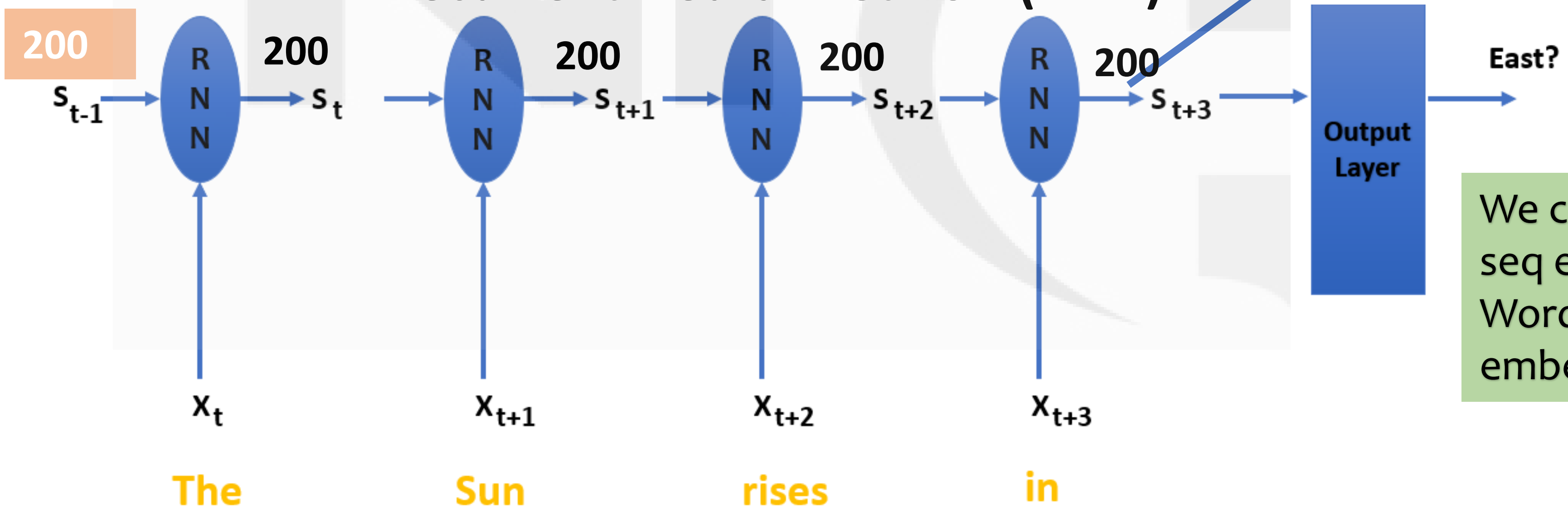
Dense Neural Network(DNN)



At every time step the memory size remains same (e.g. 200) in this case. The value in memory keeps Changning with new input

200 outputs

Recurrent Neural Network(RNN)



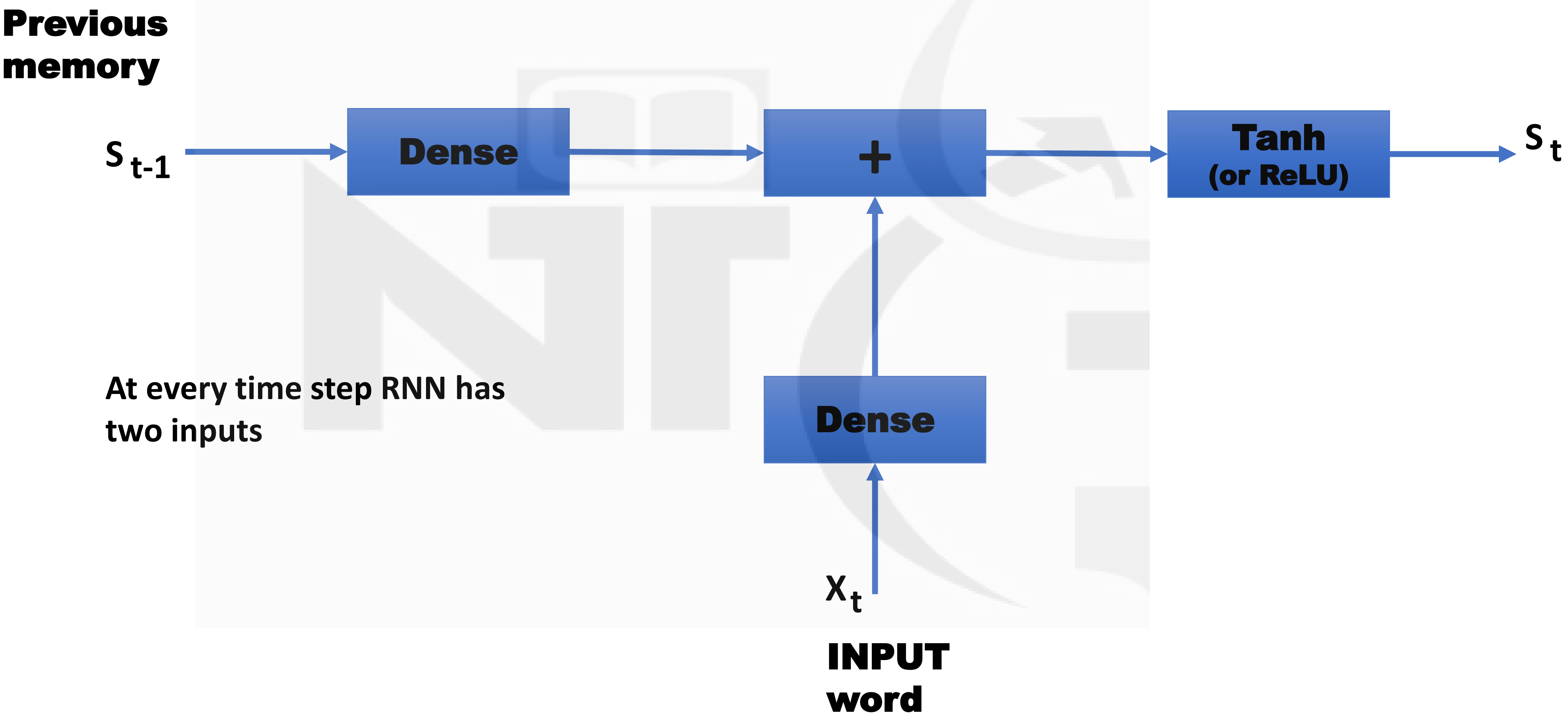
We can think of this output as seq embedding just like Word2Vec gives us word embeddings

Quick comparison of Dense vs Rnn approach

- As the seq size changes the ouput size change in dense
- RNN size remains same = memory size

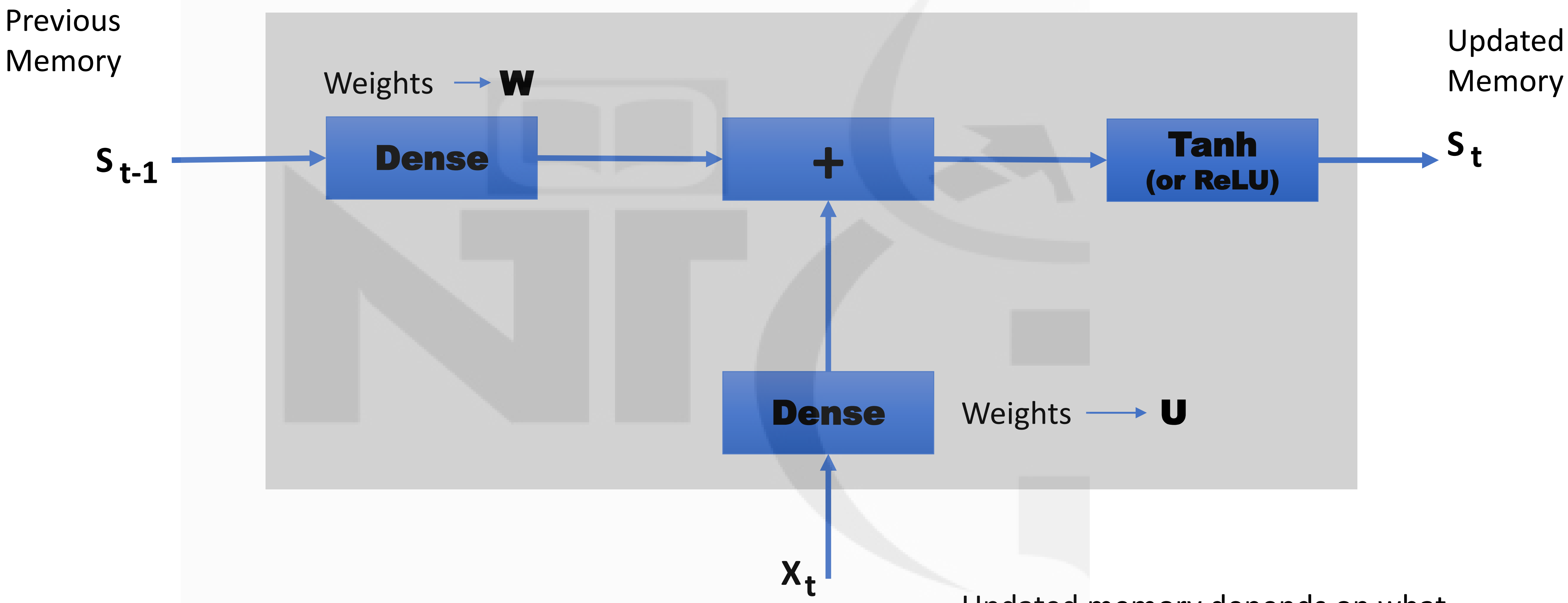
How RNN builds its memory?

RNN CELL



At every time step RNN has two inputs

RNN CELL

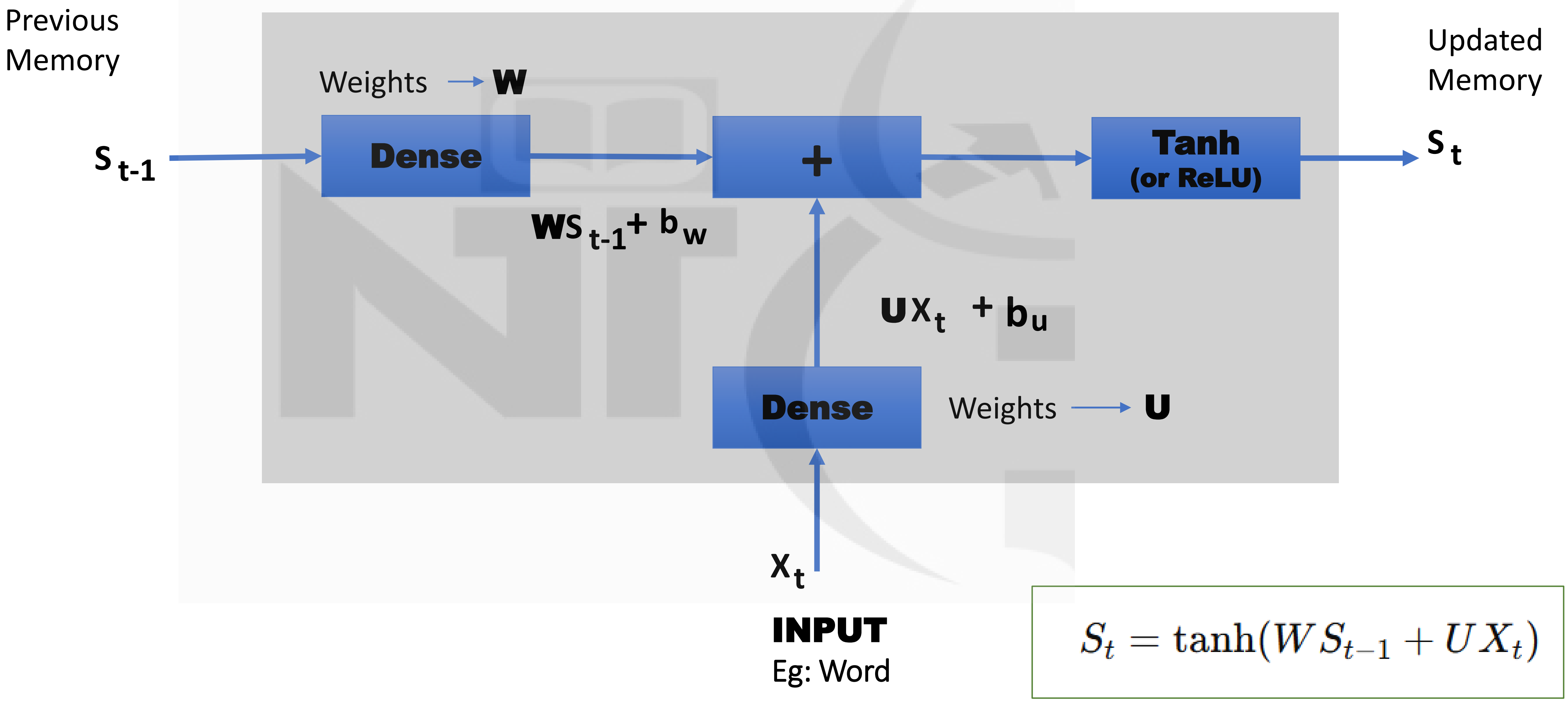


INPUT
Eg: Word

Updated memory depends on what was there in memory earlier and what we learnt just now

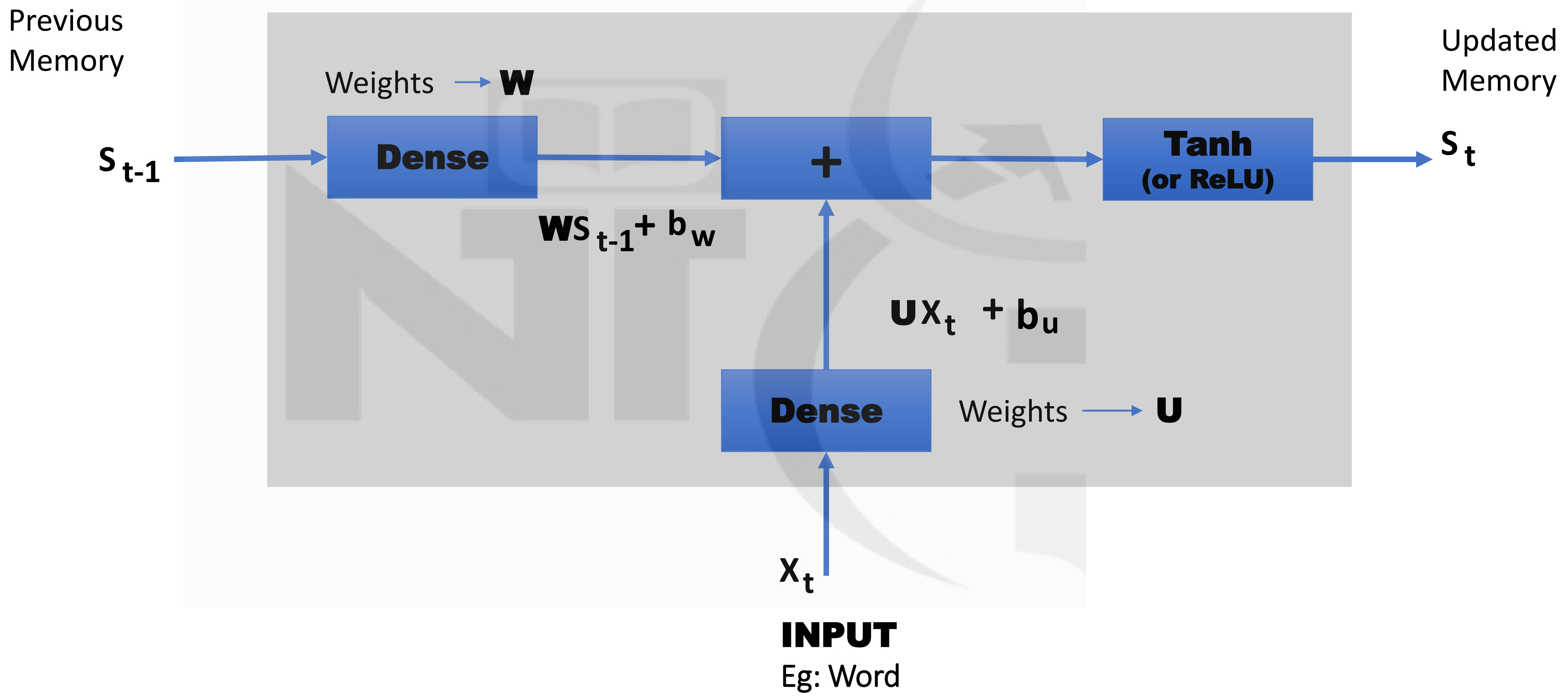
RNN CELL

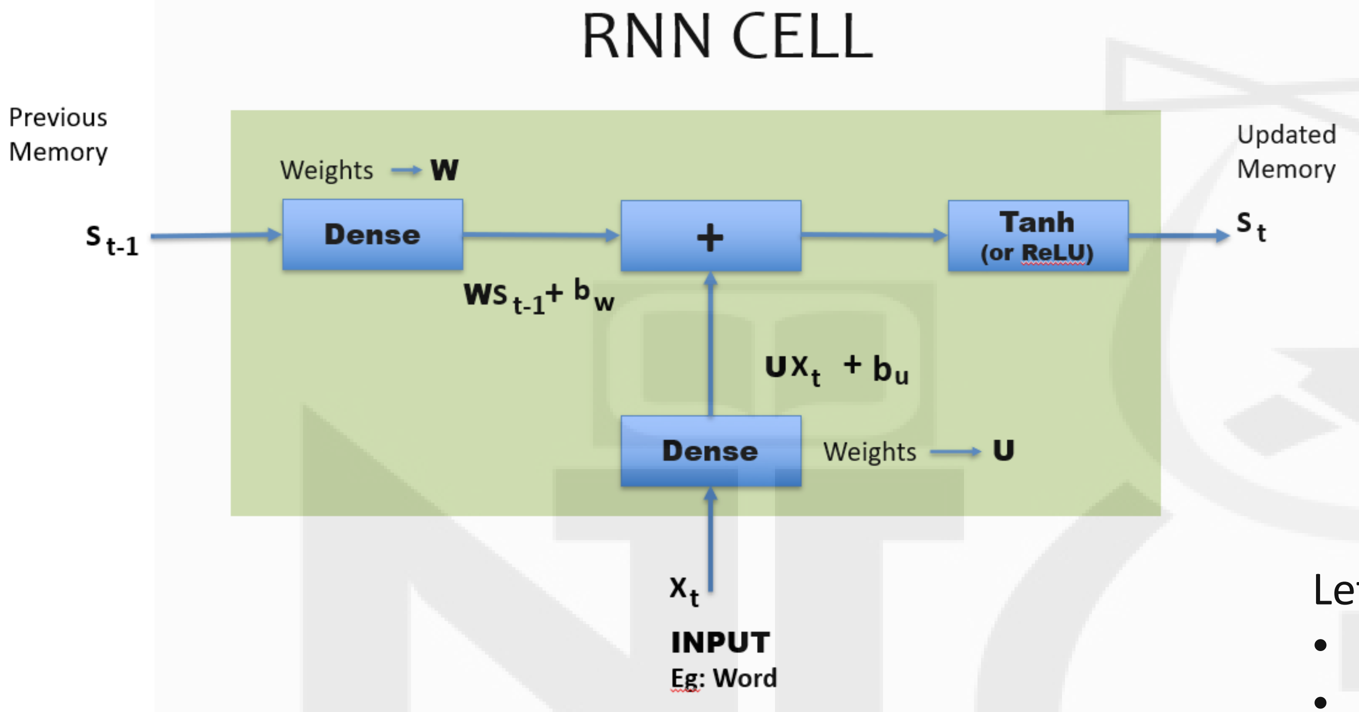
Size of U & W ?



$$S_t = \tanh(W S_{t-1} + U X_t)$$

RNN CELL


Previous Memory
S_{t-1}
Weights → W
Dense
$WS_{t-1} + b_w$
+
Tanh (or ReLU)
Updated Memory
S_t
$UX_t + b_u$
Dense
Weights → U
X_t
INPUT
Eg: Word



Size of U & W ?

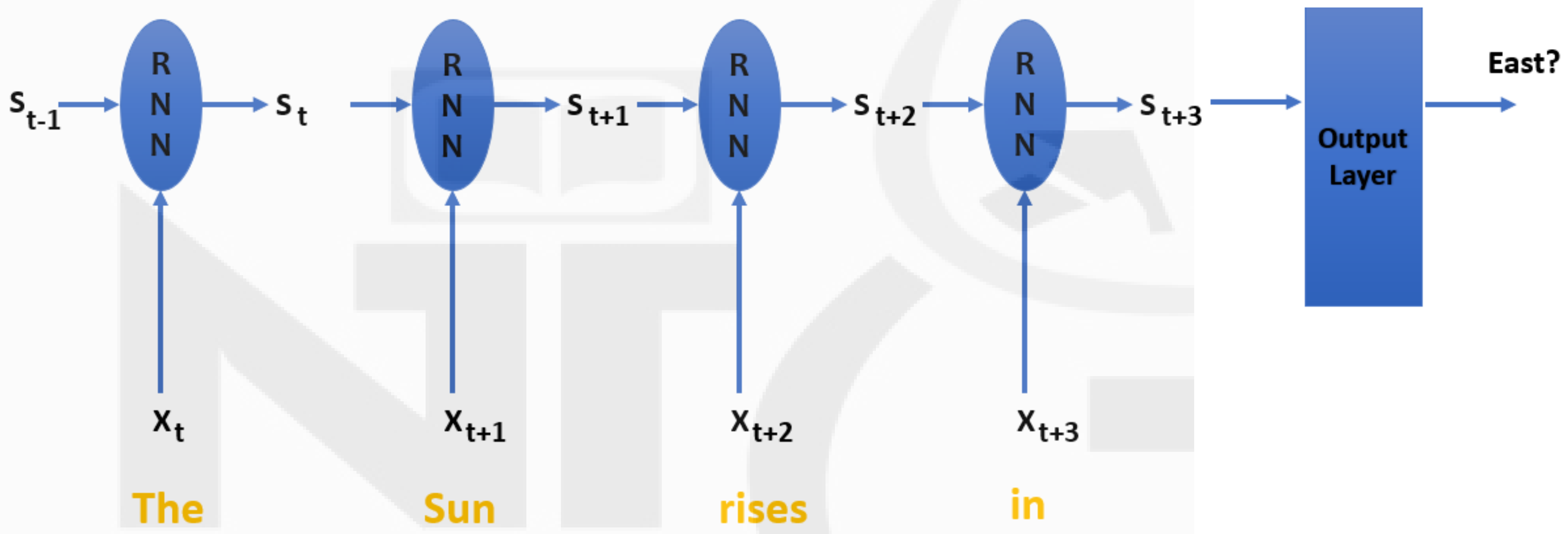
Let's assume size of …

- RNN State (S) = [200,1]
- Input word embedding (X) = [50,1]

Use matrix multiplication rules to calculate the size of U and W

U = [200, 50]
W= [200, 200]

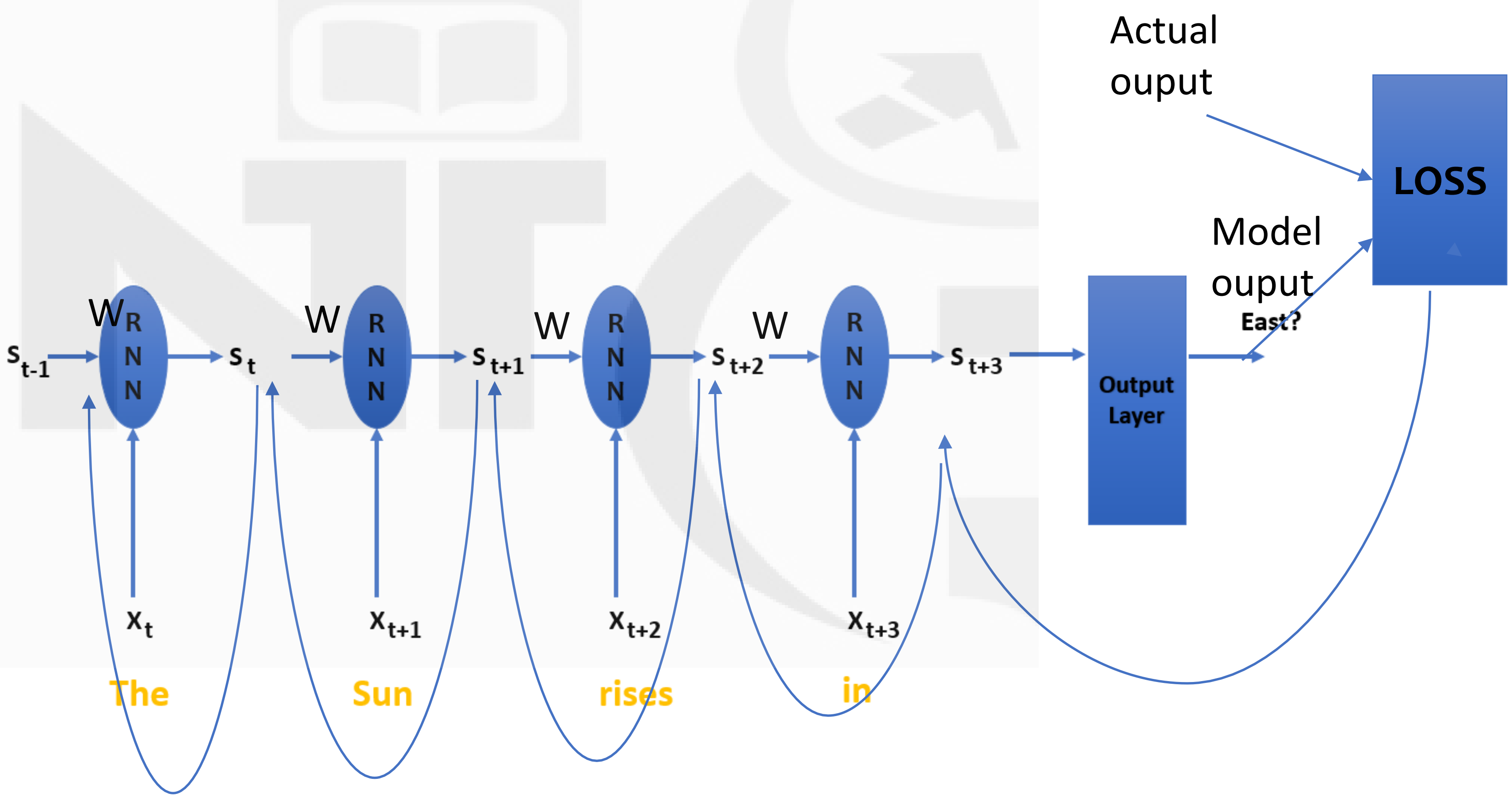
Recurrent Neural Network(RNN)



Weights of RNN (U, W) remain same at all time steps

"How is Loss related to W?

Loss → O_{t+3} → S_{t+3} → S_{t+2} → S_{t+1} → S_t → W"



Gradient of Loss w.r.t W using Chain rule

Loss → O_{t+3} → S_{t+3} → S_{t+2} → S_{t+1} → S_t → W

$$\frac{d\mathrm{Loss}}{dW} = \frac{d\mathrm{Loss}}{dO_{t+3}} \cdot \frac{dO_{t+3}}{dS_{t+3}} \cdot \frac{dS_{t+3}}{dS_{t+2}} \cdot \frac{dS_{t+2}}{dS_{t+1}} \cdot \frac{dS_{t+1}}{dS_t} \cdot \frac{dS_t}{dW}$$

BackPropagation through Time (BPTT)

Gradient of S_{t+n} with respect to S_{t+n-1}

$$S_{t+n} = \tanh(W S_{t+n-1} + U X_{t+n})$$

n → relative time step

$$\frac{dS_{t+n}}{dS_{t+n-1}} = W$$

Gradient of Loss w.r.t W using Chain rule

$$Loss \rightarrow O_{t+3} \rightarrow S_{t+3} \rightarrow S_{t+2} \rightarrow S_{t+1} \rightarrow S_t \rightarrow W$$

$$\frac{dLoss}{dW} = \frac{dLoss}{dO_{t+3}} * \frac{dO_{t+3}}{dS_{t+3}} * \frac{dS_{t+3}}{dS_{t+2}} * \frac{dS_{t+2}}{dS_{t+1}} * \frac{dS_{t+1}}{dS_t} * \frac{dS_t}{dW}$$

$$\frac{dLoss}{dW} = \frac{dLoss}{dO_{t+3}} * \frac{dO_{t+3}}{dS_{t+3}} * W * W * W * \frac{dS_t}{dW}$$

Gradient calculation requires repeated multiplication by 'W'

Vanishing or Exploding Gradient

(If W is small or large)

Cannot remember for long

(very short memory)

Hence, we do not use RNN in real-time applications