NLP-101-Introduction

Deep Learning for NLP

Ali Abdelaal, Language Engineer @Apple

Session Agenda

- Word embeddings (word2vec, GloVe, Fasttext)
- Sequential Modeling (RNN, LSTM)
- Text Classification using LSTM
- Sequence labeling using LSTM
- Neural Machine Translation
- Text Generation

Word Embeddings

Word Embeddings

The need ?

- BOW representation gives no information about the word itself.
- Contextual information about the word enables better understanding.
- Word embeddings enables describing the text in a more reach way.
- In case of low resources problems we can make use of the pre-trained word embeddings to kickstart the model.

BOW

Word Embedding

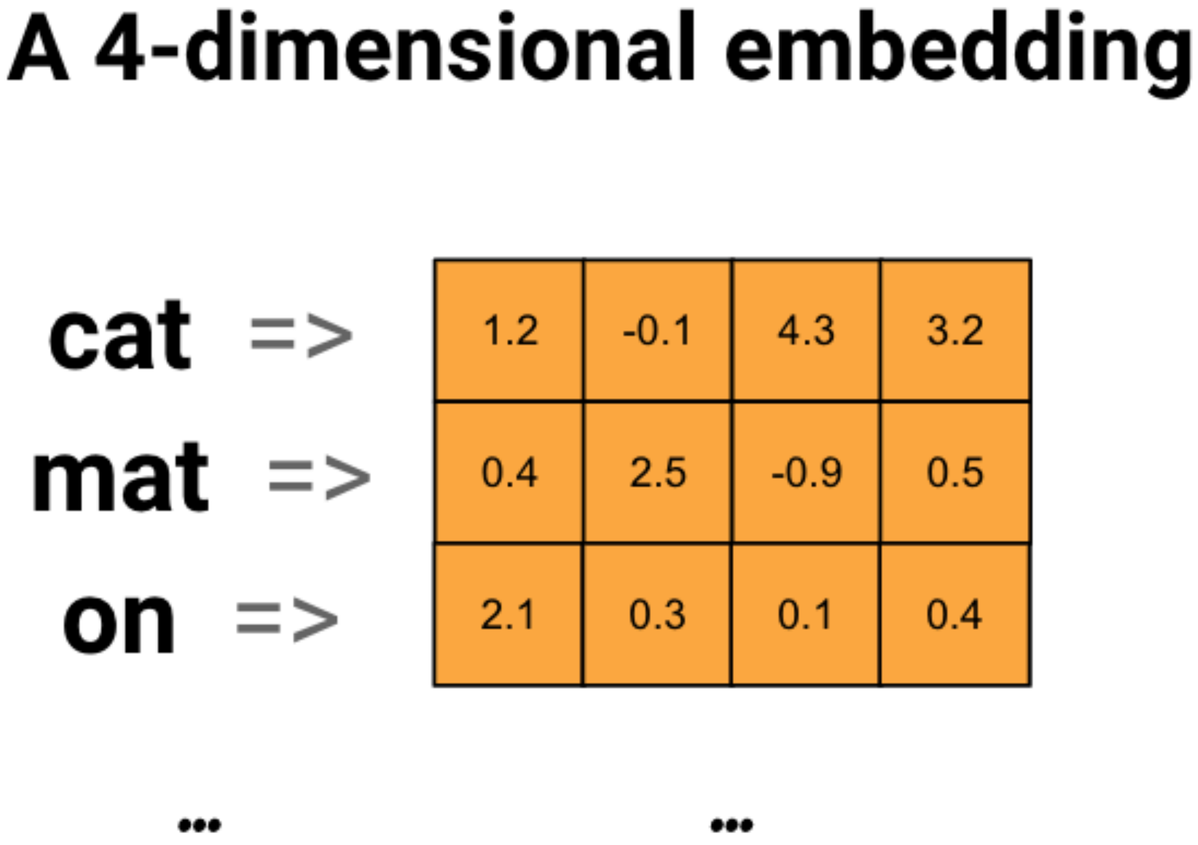
- Simple and quick to build
- Gives no information about the word
- doesn't take context into account

	s1	s2	s3	s4	s5
cat	0	1	0	1	0
dog	1	1	1	0	1
eats	0	1	1	1	0
food	0	0	1	0	0
hot	0	0	0	0	1
red	1	0	0	1	0
the	1	0	0	0	1

word embeddings

Word Embedding

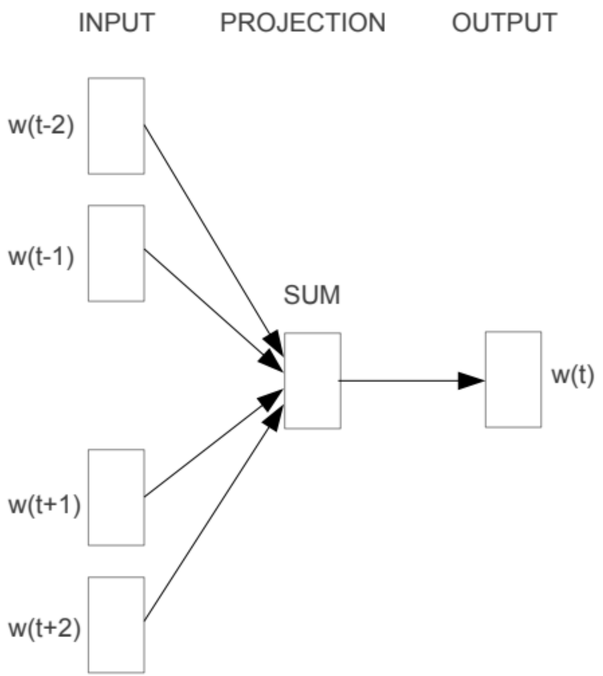
- Can be initialized to any size.
- Starts with random weights and gets updated.
- Represents context information about words when trained with enough data.
- Can be trained and reused in different tasks.



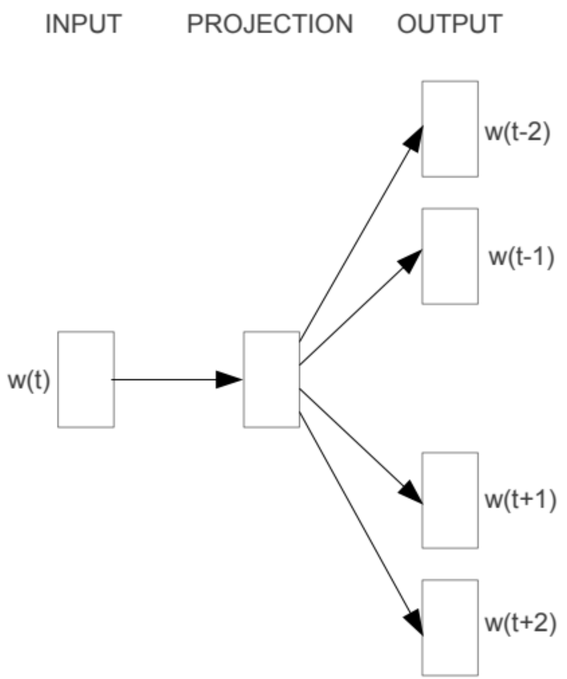
source

Word2Vec
Word Embeddings

- Train a neural network to predict the context given a word and vise versa.
- The model learns to understand the language by predicting context/word



CBOW



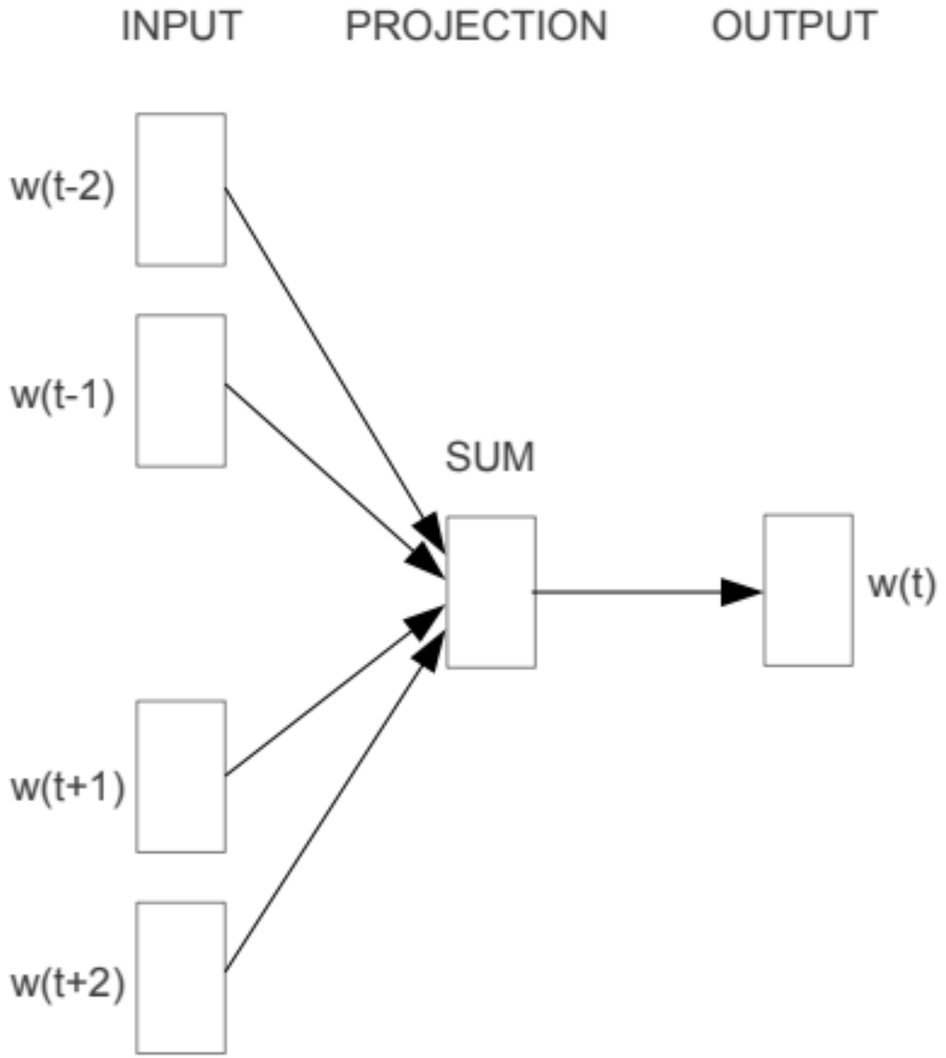
Skip-gram

source

CBOW

word2vec

- CBOW -> Continuous bag of words
- The input to the model is the context words and the objective is to predict center word

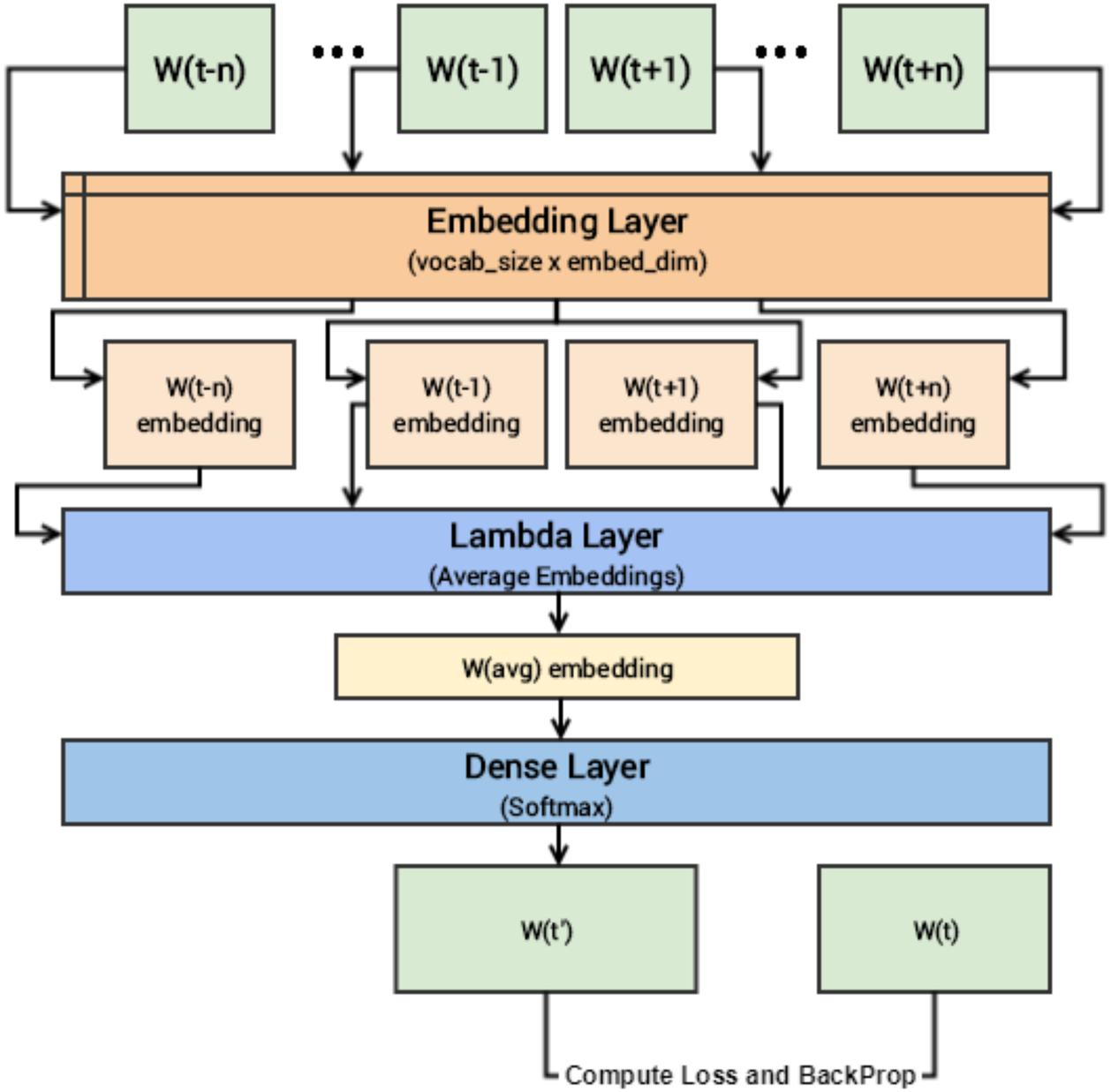


CBOW

source

CBOW
word2vec

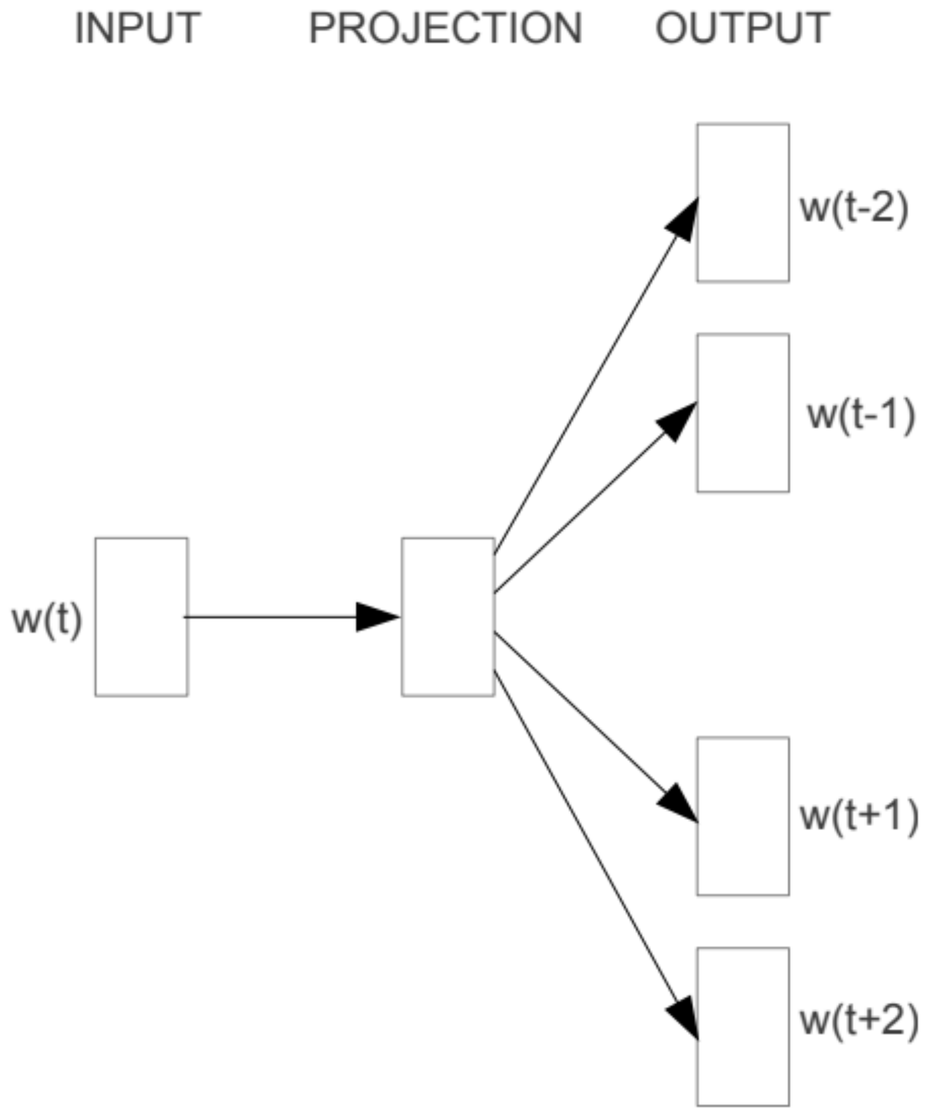
1. feed context words to same embedding layer
2. average embeddings
3. Feed the average to dense layer that tries to predict the word
4. compare predicted and true word and back propagate.



source

Skip-gram
word2vec

- skip-gram model predicts the context (or neighbors) of a word, given the word itself.



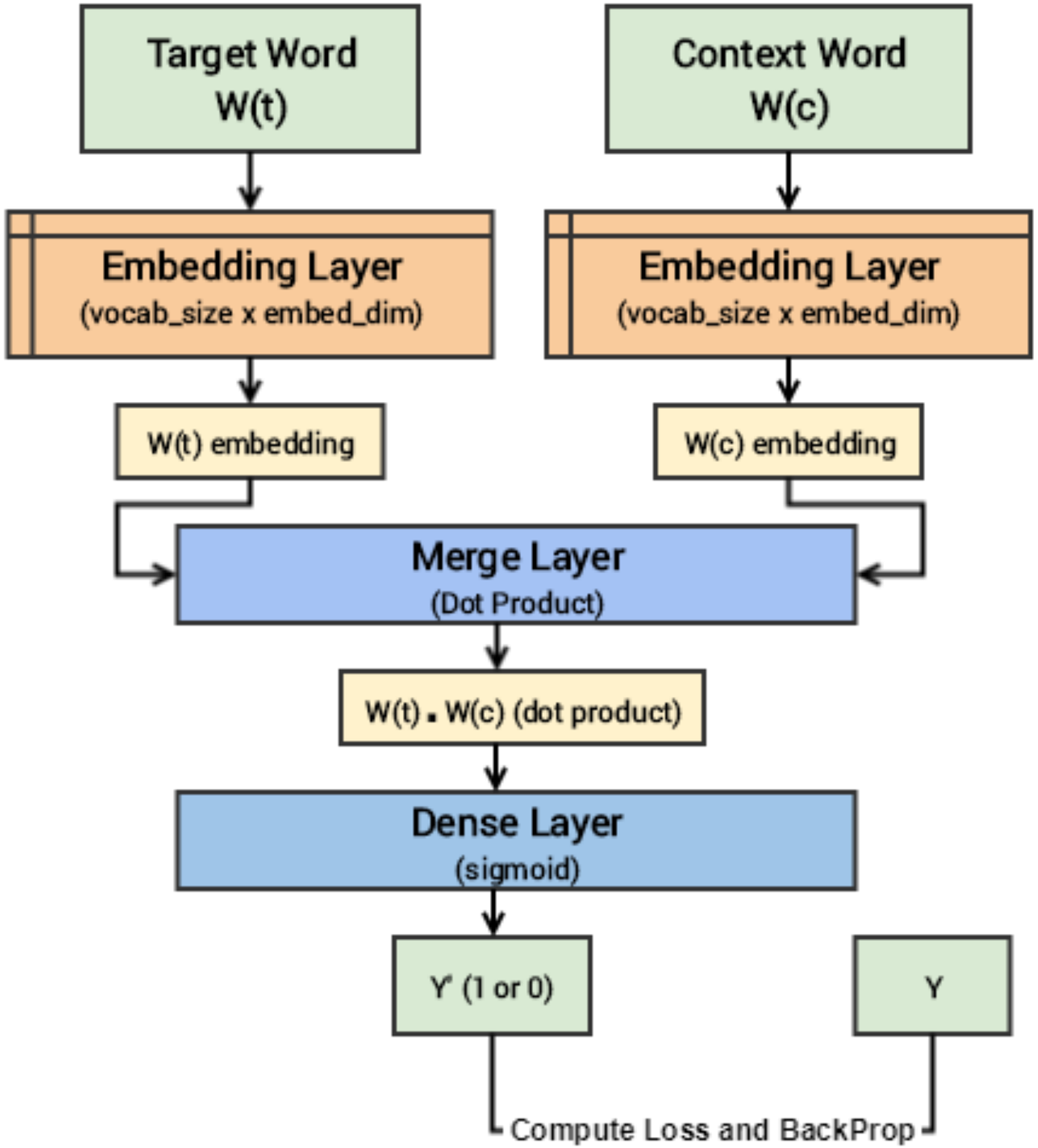
Skip-gram

source

Skip-gram
word2vec

- The model is given a pair of (word, context)
- Generate embeddings for both
- Merge the embeddings and feed to dense layer.
- A sigmoid layer decide whether they are similar or not.
- Compare to ground truth.

source

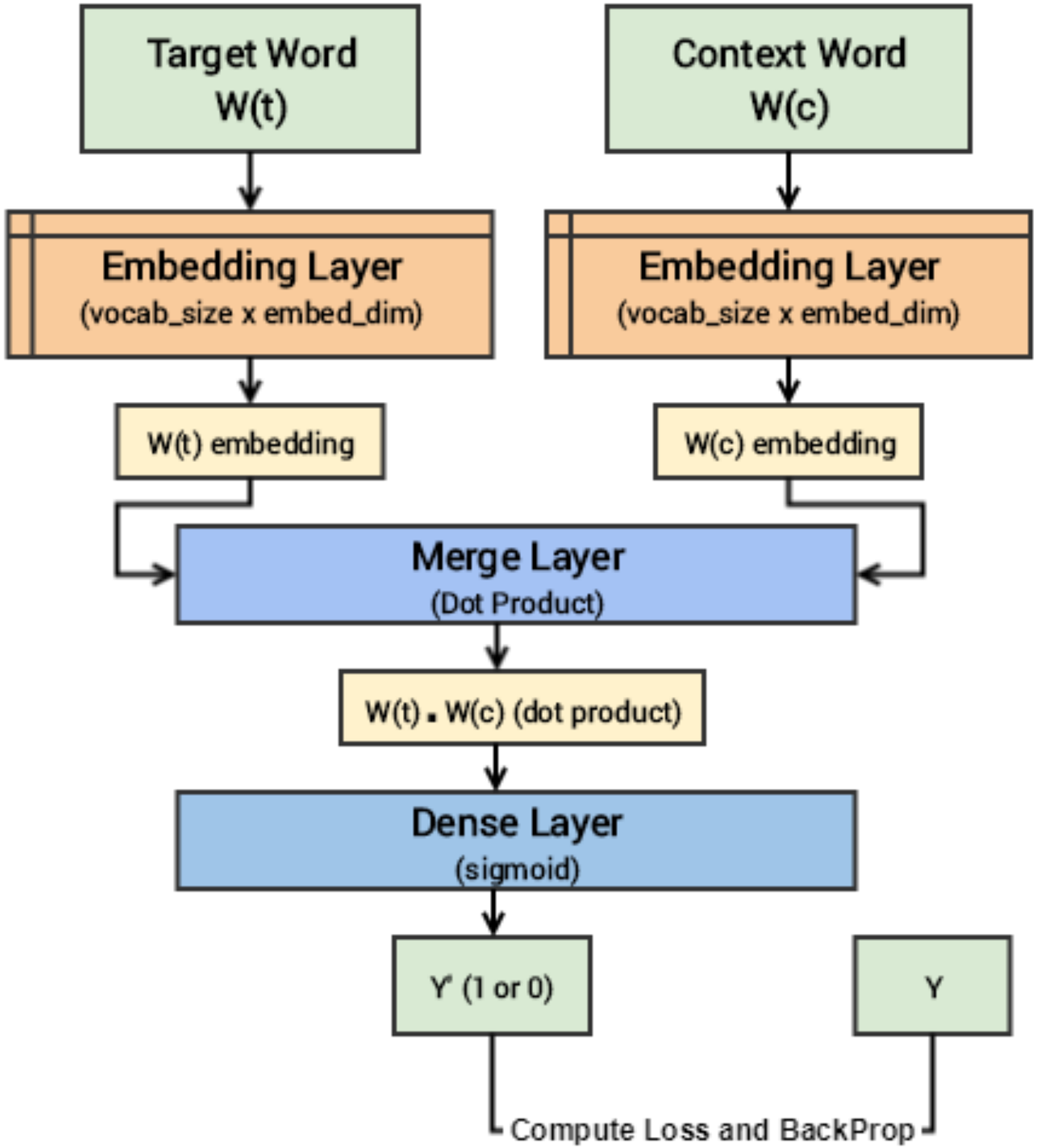


Negative Sampling

word2vec

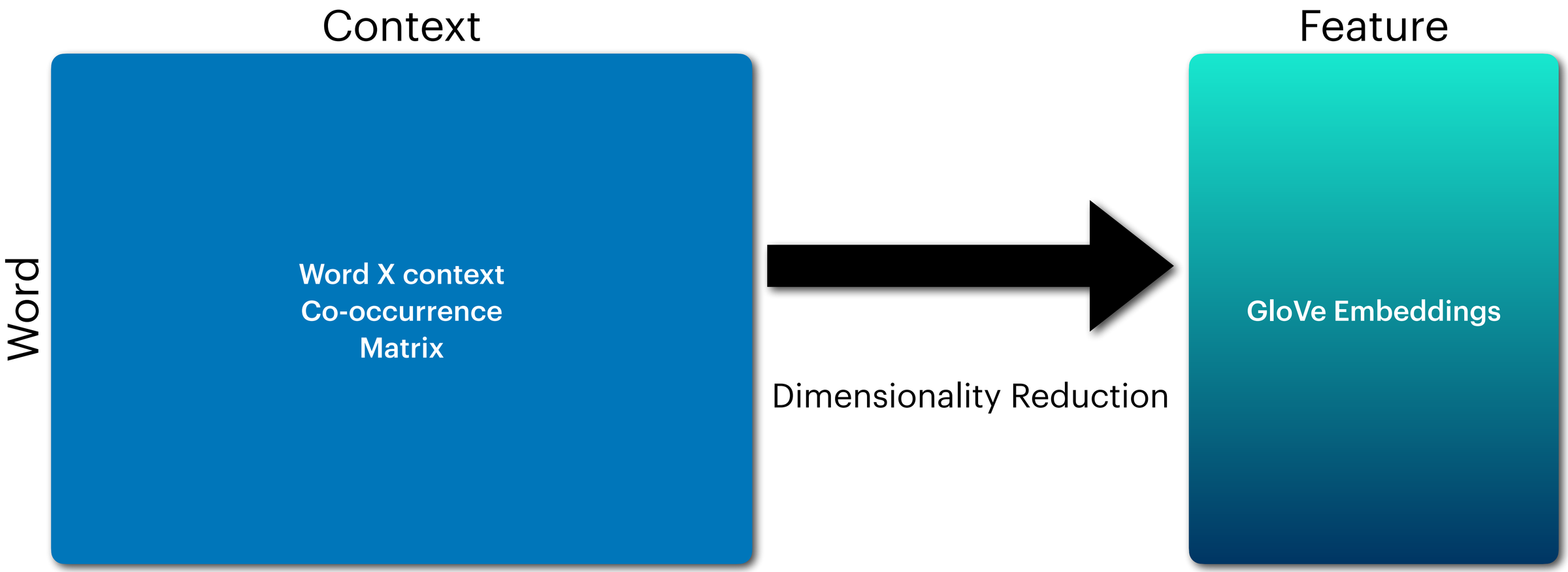
- We use Negative sampling to train skip-gram version of the model.
- Negative sampling is very useful in different tasks.
- example data would be
  - (word, context, 1)
  - (word, context, 0)

source



GloVe vs word2vec

GloVe



source

GloVe vs word2vec

word2vec

- Both models learn geometrical encodings (vectors) of words from their co-occurrence information (how frequently they appear together in large text corpora).
- Unlike word2vec, GloVe is a count-based model, it learns the vectors by essentially doing dimensionality reduction on the co-occurrence counts matrix.
- GloVe is easier to parallelize and has global information.
- In practice they give similar performance on NLP tasks.

source

Limiations ?
Word2Vec and GloVe

- Word based models don't go well with Out Of Vocab (OOV) cases.
- They don't make use of the morphology of the words, for example (eat, eats, eaten, ...) are treated as completely separated words.
- A word will always have the same embeddings regardless of it's context, for example the word كوكب will be treated the same in these two examples:
  - اريد سماع انت عمري لكوكب الشرق
  - كم كوكب في المجموعة الشمسية

FastText
Word Embeddings

- Builds on top of word2vec
- Instead of using only words, the model generates n-grams for the words
- This allows the model to work around OOVs and also make use of the morphology of the text.
- A detailed explanation can be found here

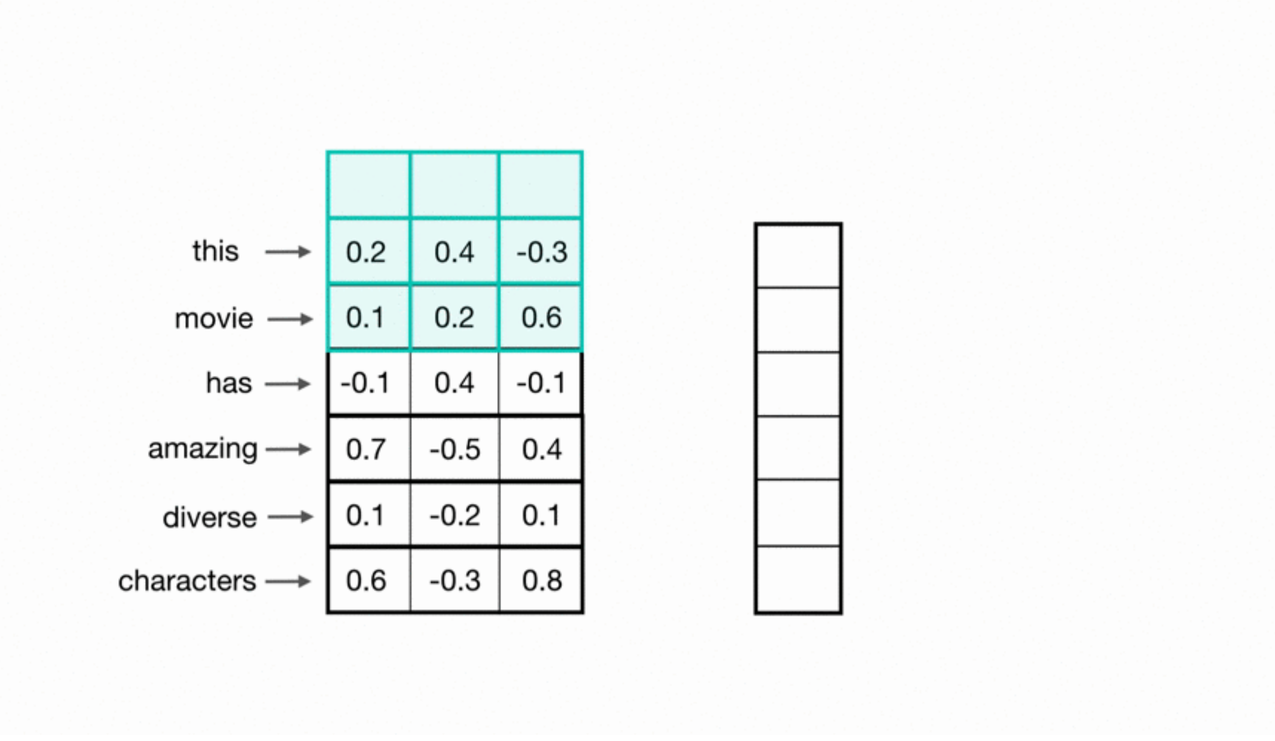
paper

More on embeddings

N-Gram using words embeddings

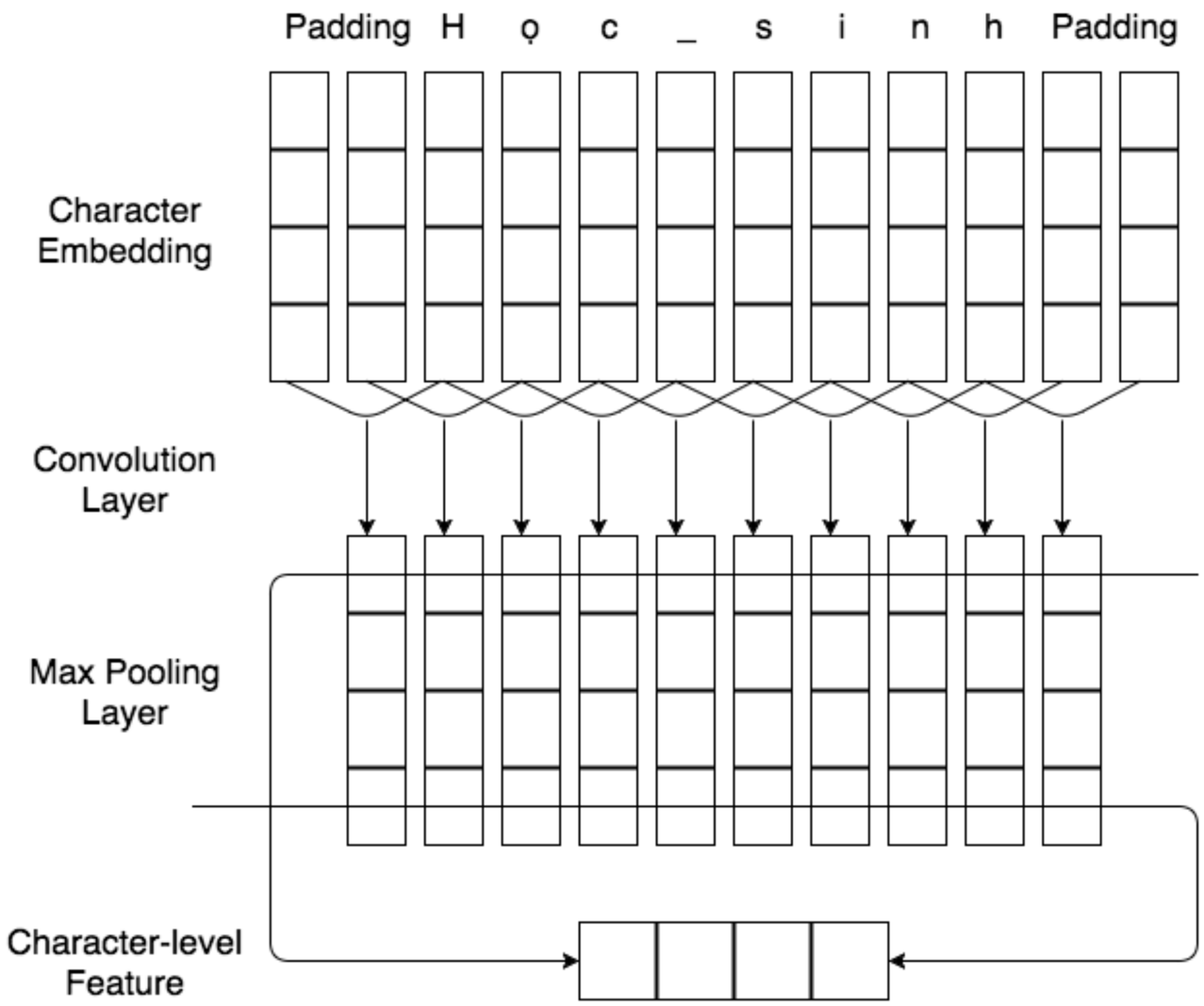
Word Embeddings

- We can use CNN to simulate the n-gram behavior using word embeddings.
- We use 1D convolutions to capture the n-gram



CNN with Char embedding
Word Embedding

- character level embedding can be very helpful as it tends to solve the typos issues and out of vocabulary words.
- Multi layer CNN can learn features from features and thus build some complex knowledge out of the characters



Notebook - 1

Word Embeddings

Questions ?

Word Embeddings

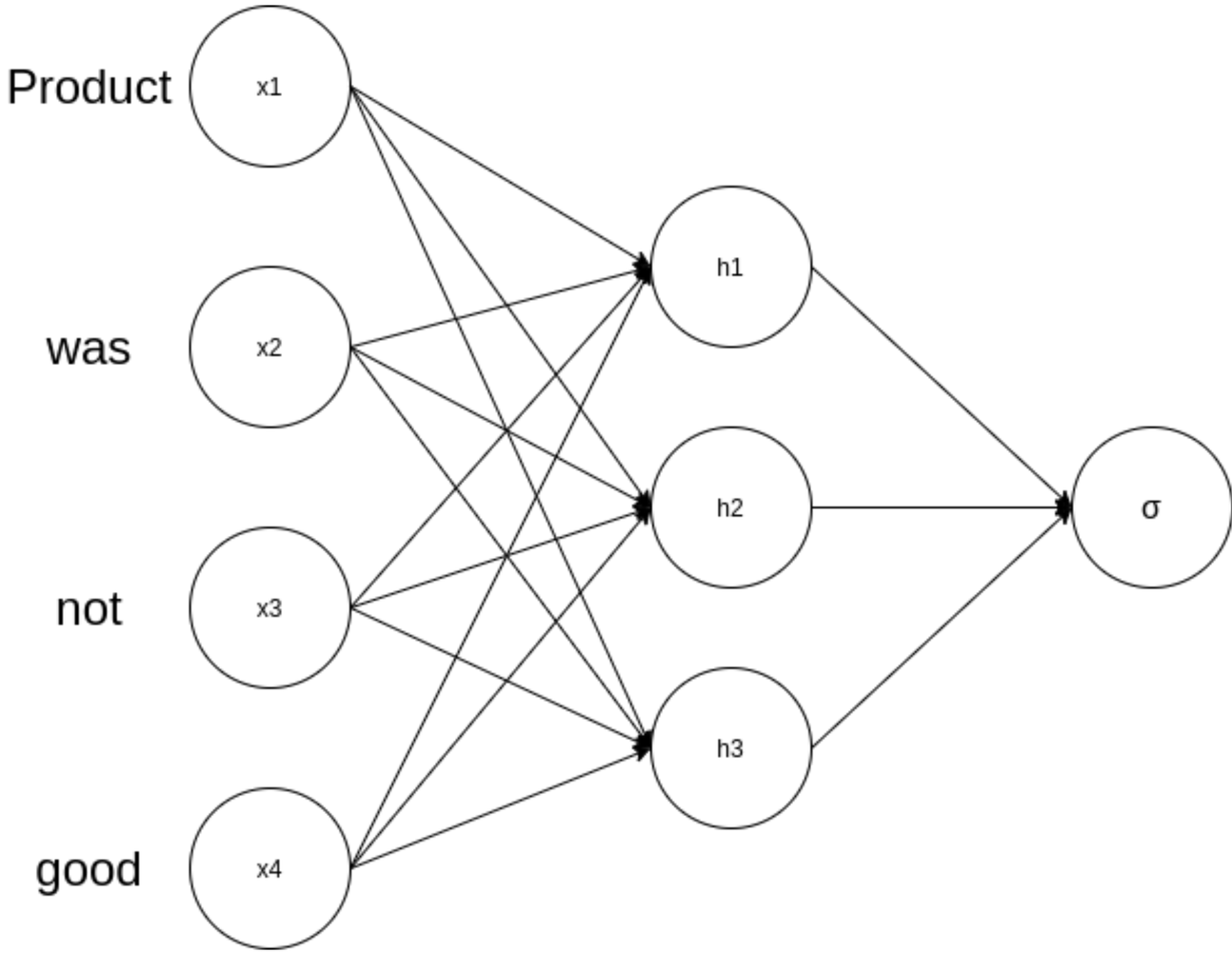
Sequential Modeling

Recurrent NN

Sequential modeling

- Feed Forward network is not able to capture sequential features.
- The order doesn't matter here
- The position is not addressed

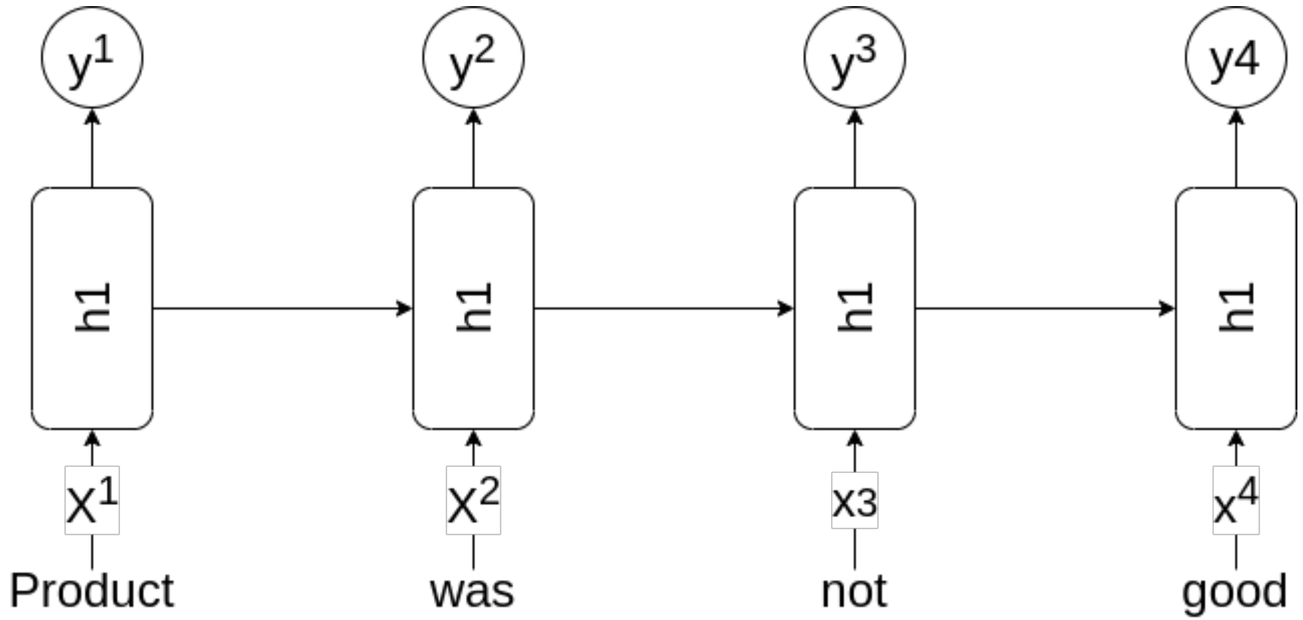
$$h1(x) = h1('product',' was',' not',' good') = W_{1,1} *' product' + W_{1,2} *' was' + W_{1,3} *' not' + W_{1,4} *' good'$$



Recurrent NN

Sequential modeling

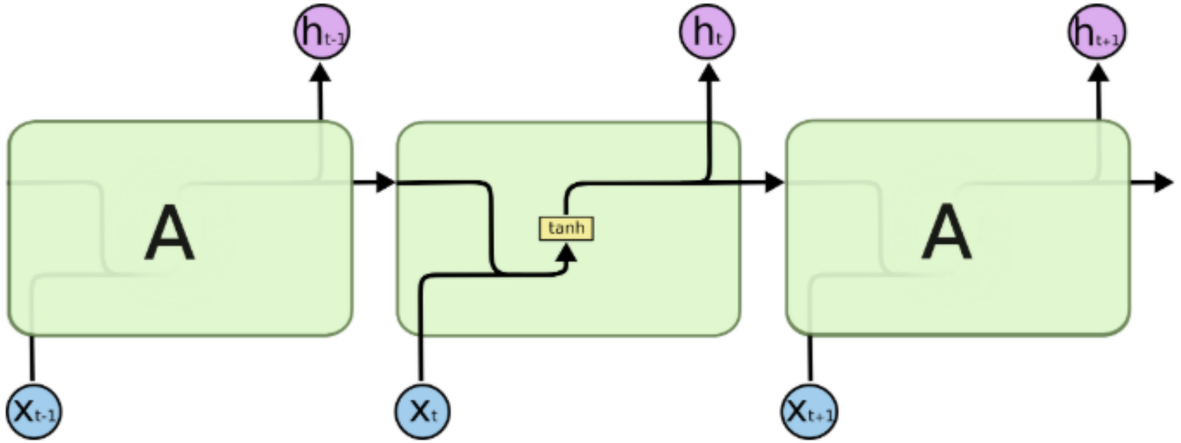
- RNN allows the position encoding by design by feeding the input in sequence to the same hidden layer.



$$h1(x) = h1('good', h1('not', h1('was', h1('product', '< sos >'))))$$

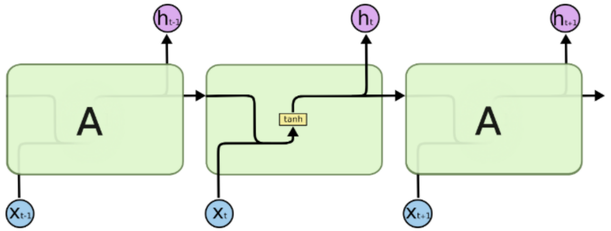
RNN Explained
Sequential Modeling

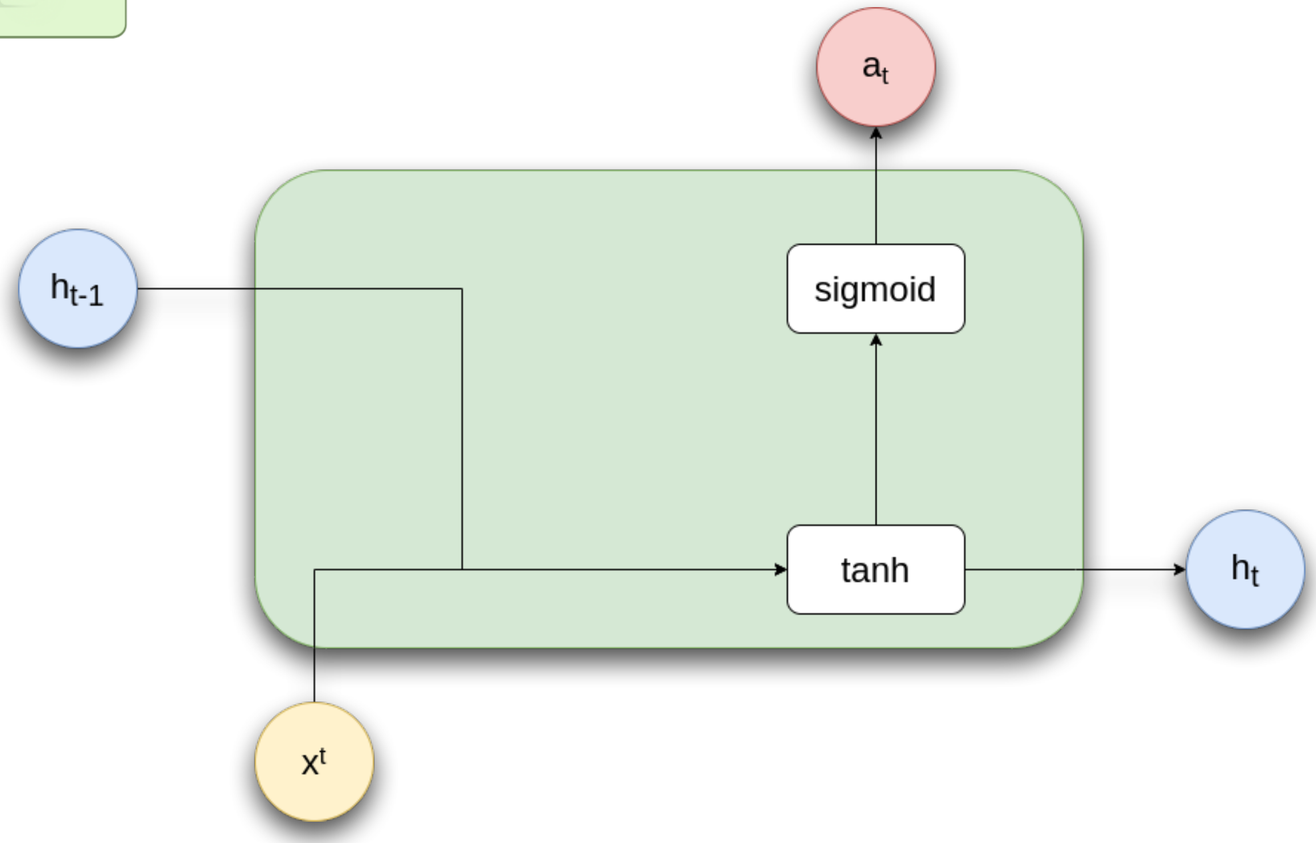
- The cell does one calculation to the given input vector.
- Each cell feed the output vector h(t) to itself in the future with the new input vector h(t-1)
- The cell learns to model the time dependency between it's inputs



RNN Explained

Sequential Modeling





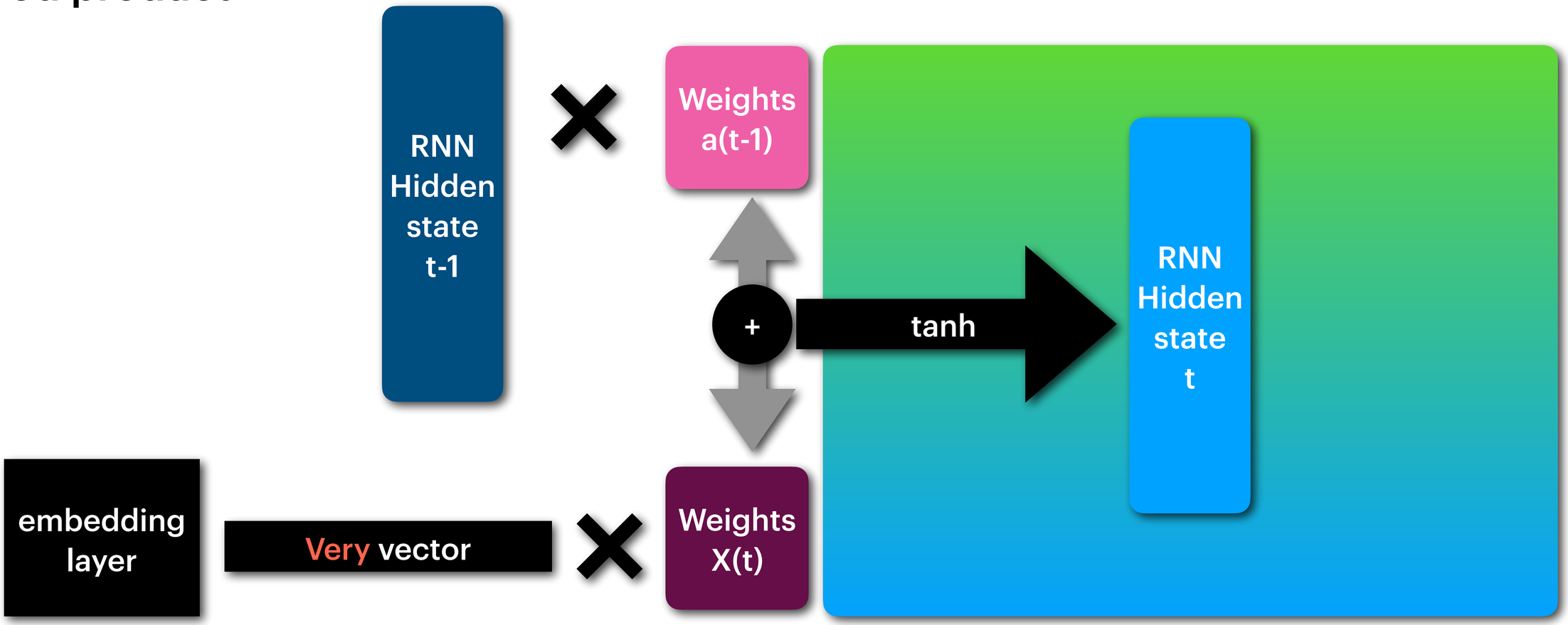
Let's dive deeper

How Sequential models work

How RNN works

Sequential Modeling

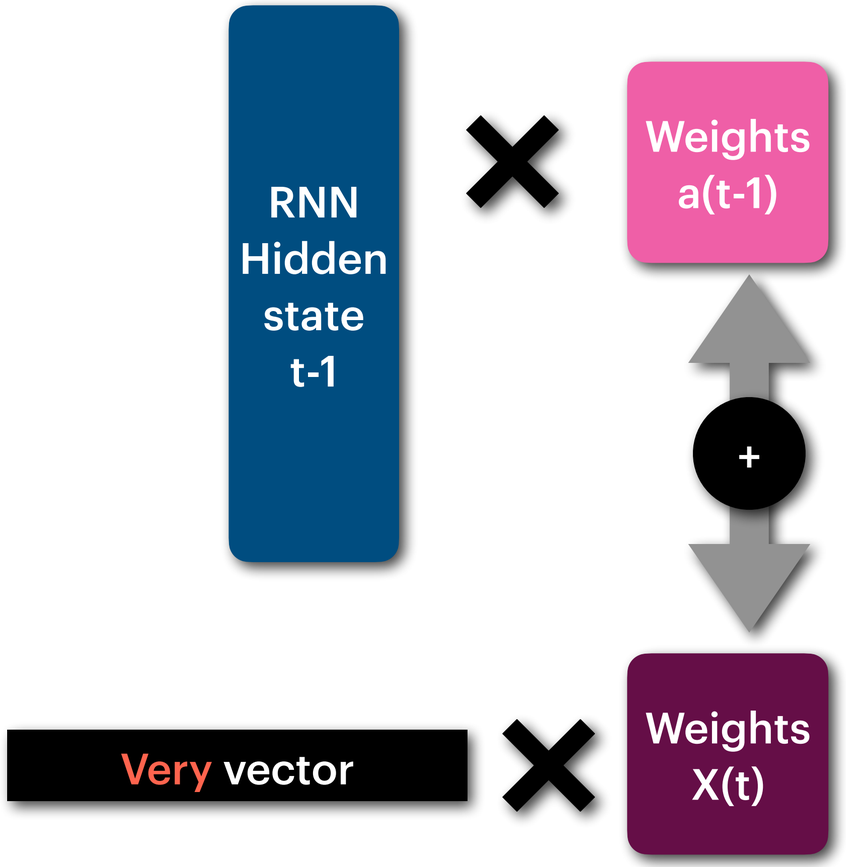
Very good product



How RNN works

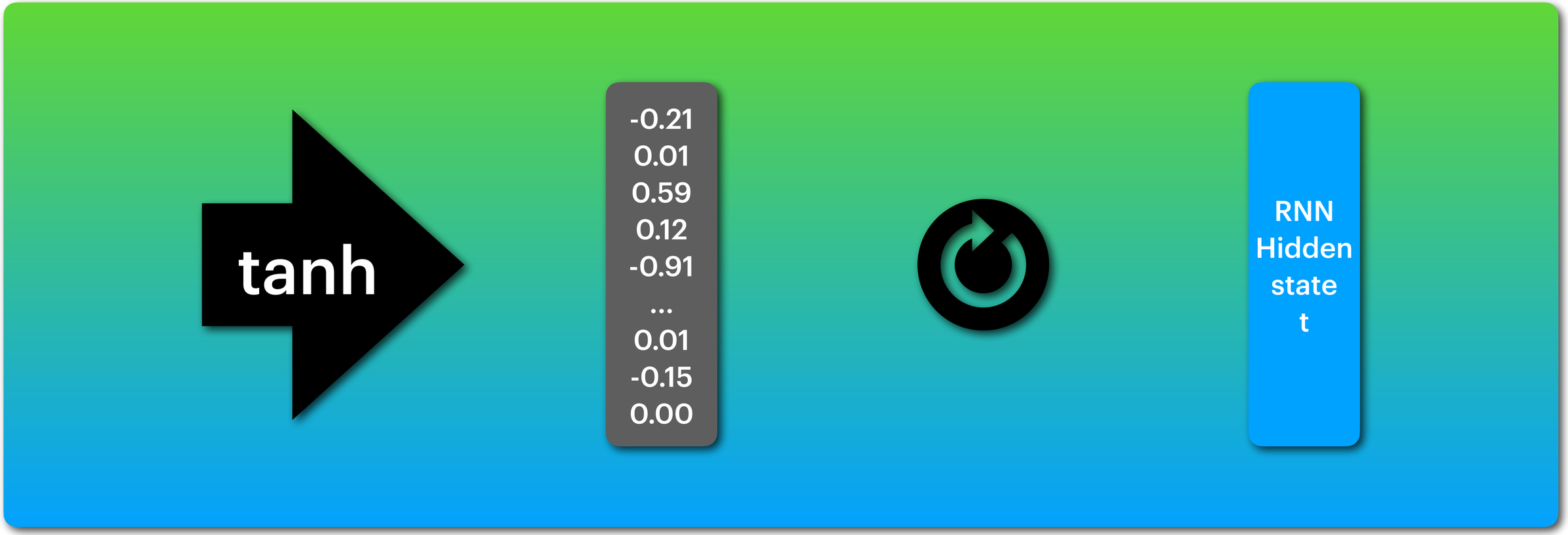
Sequential Modeling

- This allows us to generate a new state based on the previous state and the current input, meaning the cell will have information about the past and the present.
- The cell will then pass this information to the next one.



How RNN works

Sequential Modeling

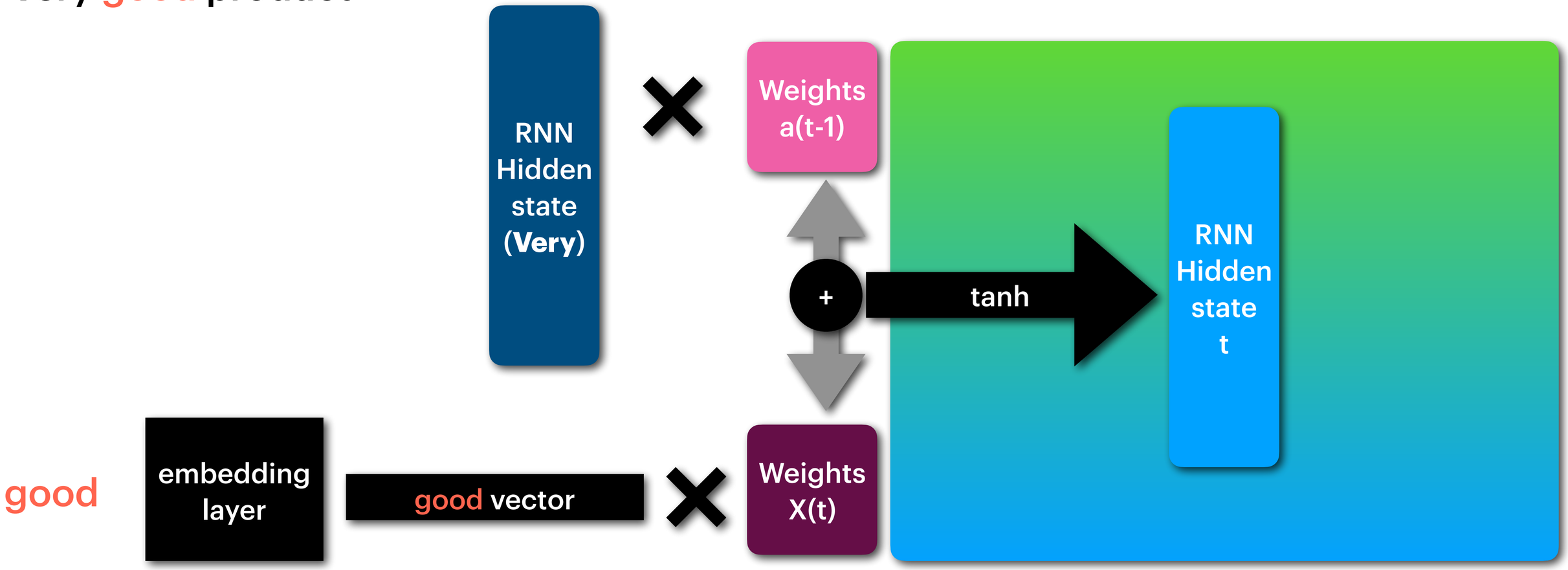


This allows the cell to update it's current state based on this input and the previous state also

How RNN works

Sequential Modeling

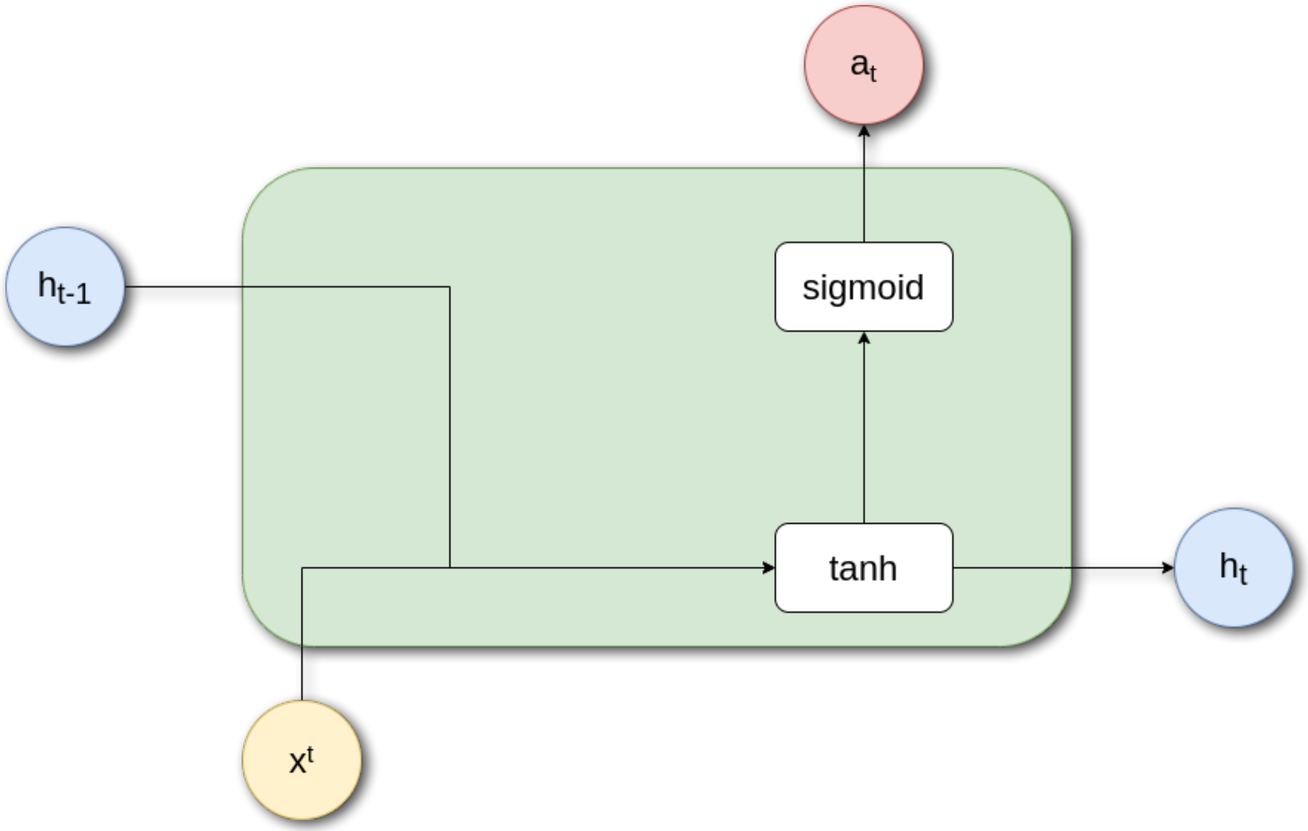
Very good product



Long Sequences

Sequential modeling

- RNN as you can guess doesn't remember long sequences.
- RNN suffers from vanishing and exploding gradients



Gated recurrent network GRU

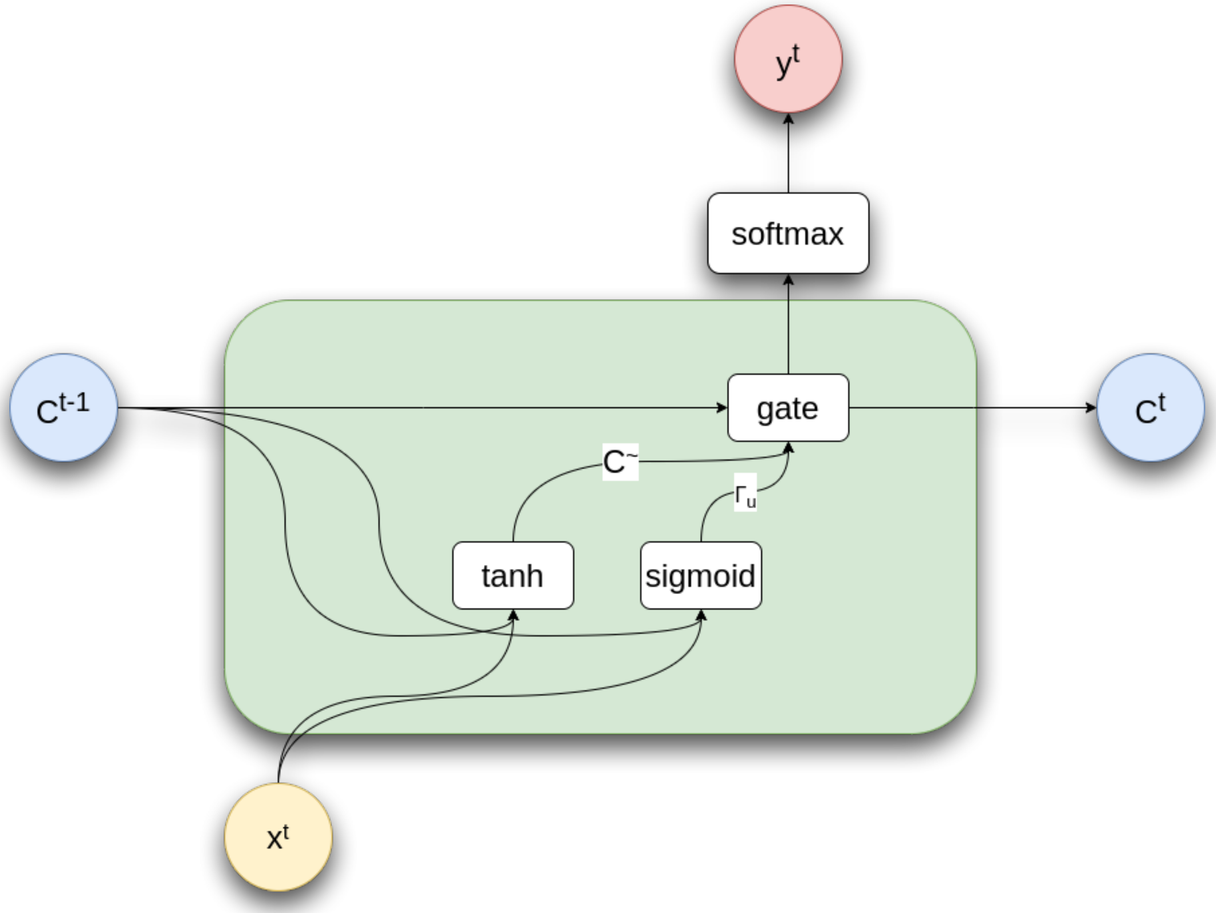
GRU Explained

Sequential Modeling

- GRU is a modification of RNN with additional gate to help keep some memory from being vanished.
- The gate is responsible for updated the new state by keeping and removing from old and new cell state respectively.

GRU Explained

Sequential Modeling



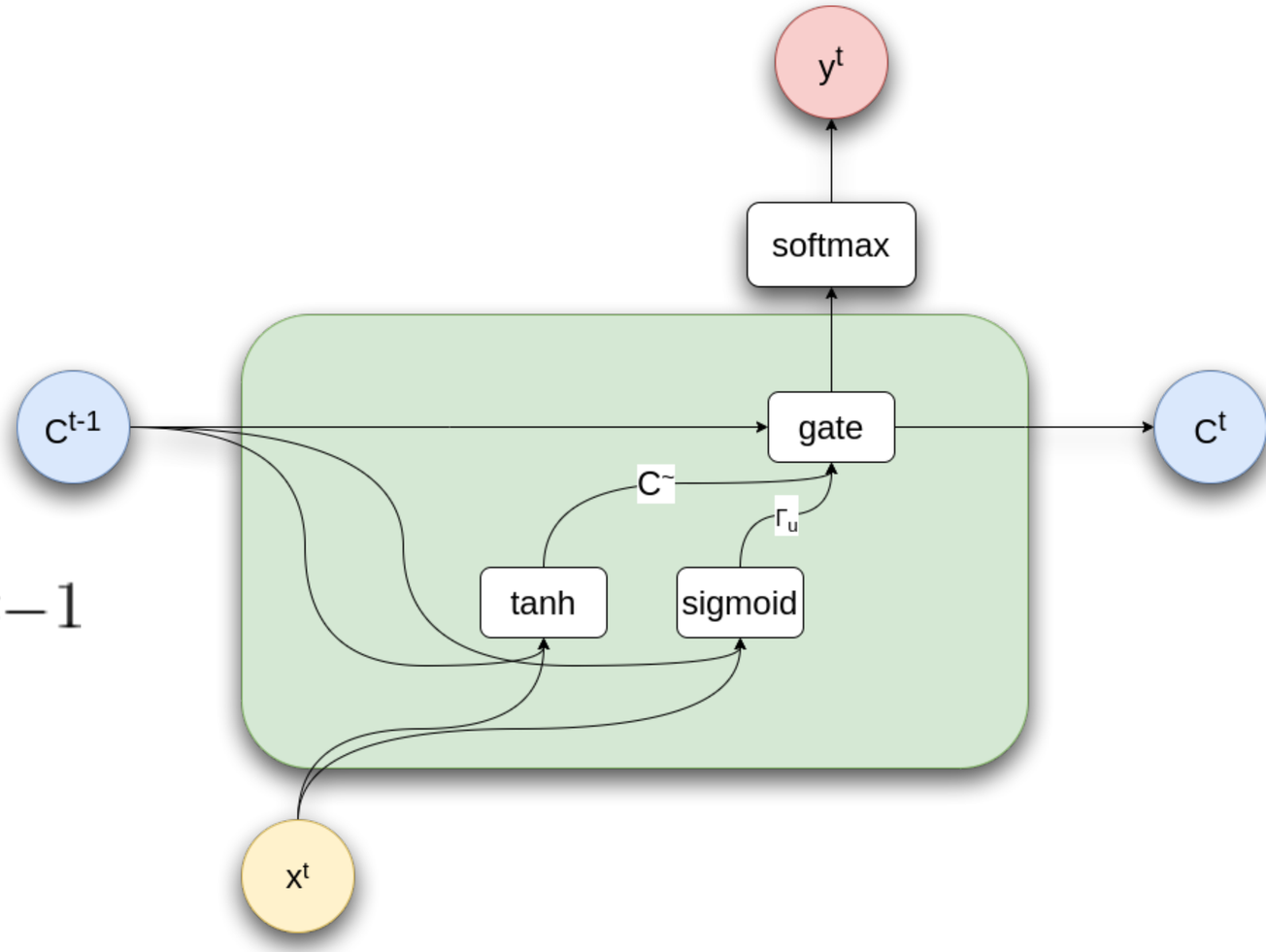
GRU

Sequential Modeling

$$\tilde{C} = tanh(W_c[C^{t-1}, X^t] + b_c)$$

$$\Gamma_u = \sigma(W_u[C^{t-1}, X^t] + b_u)$$

$$C^t = \Gamma_u * \tilde{C} + (1 - \Gamma_u) * C^{t-1}$$



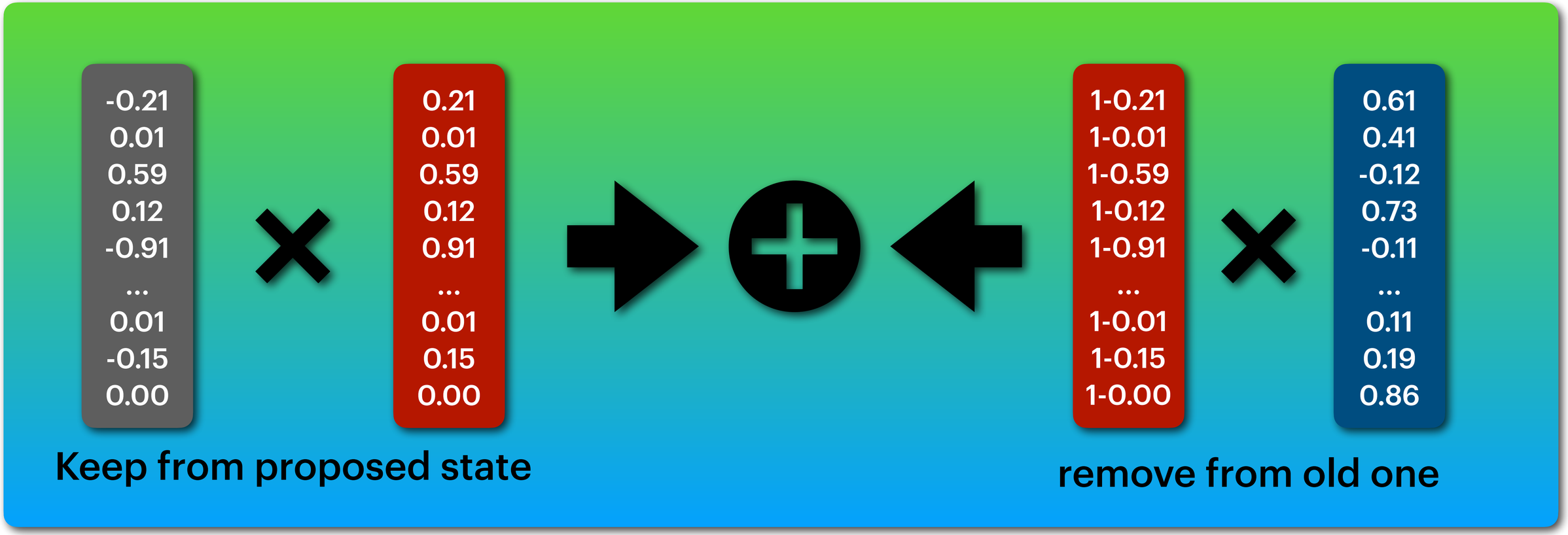
GRU explained

Sequential Modeling

The learnt gate objective is to choose what to stay and what to get updated

GRU explained

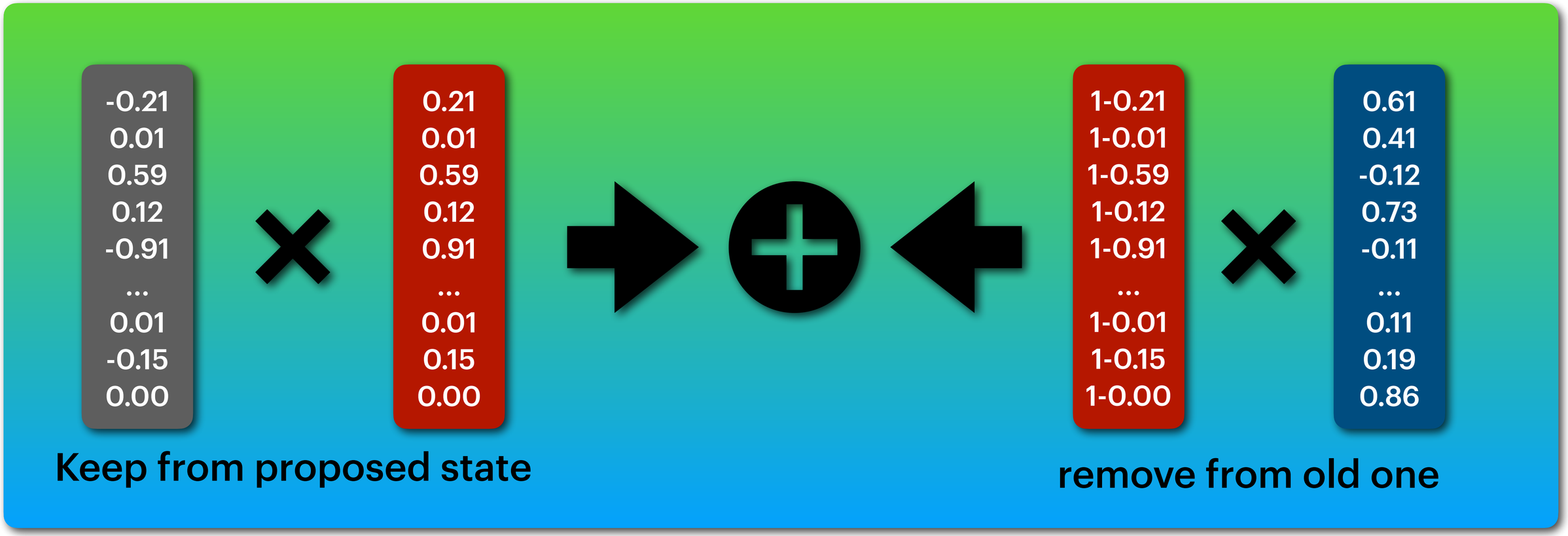
Sequential Modeling



The cell is able to keep and remove from memory and thus can have better memory

GRU explained

Sequential Modeling



Note: sigmoid will not cause vanishing gradient as it keeps from one state and remove the other

let's add more gates !

LSTM

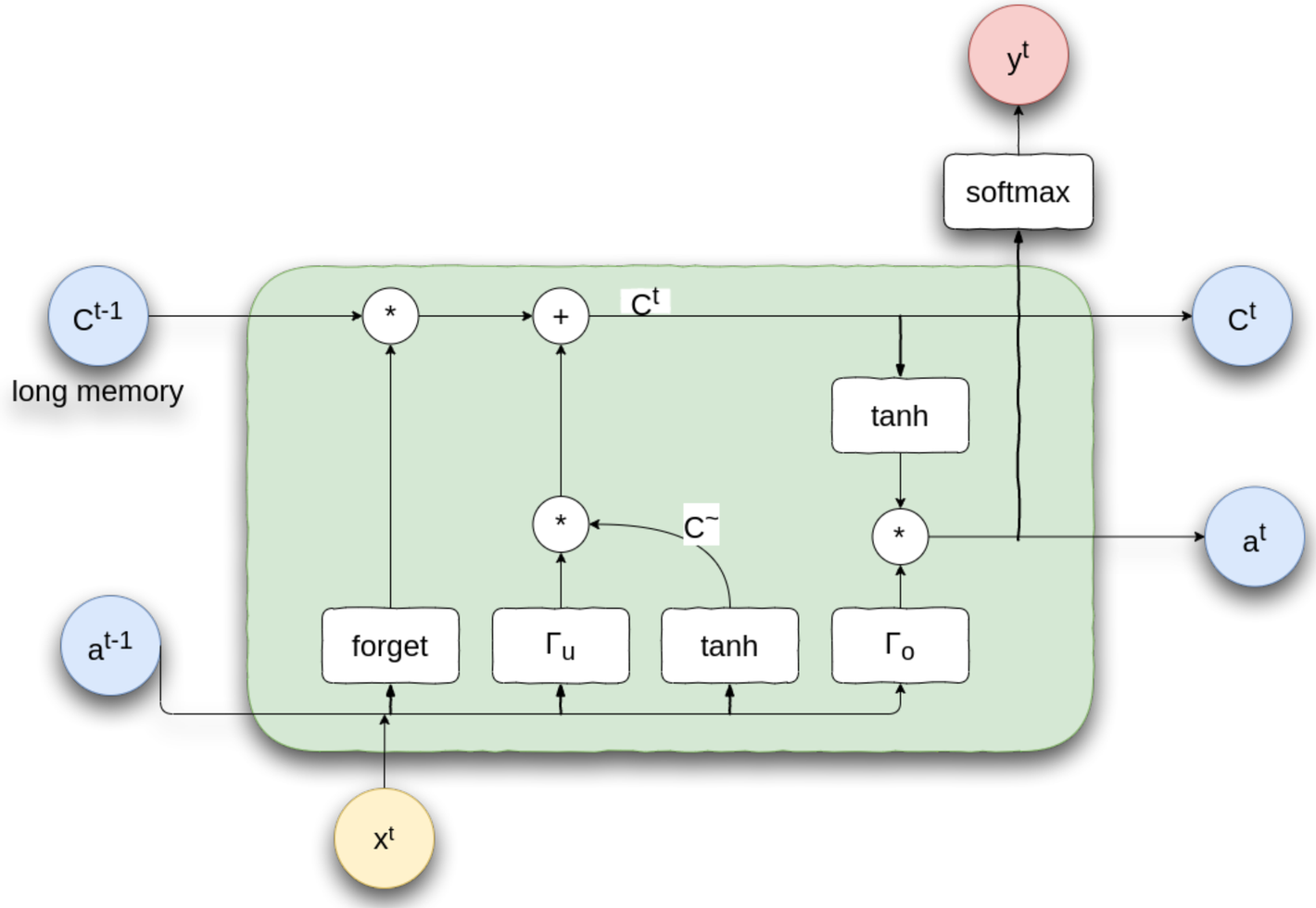
Long Short Term Memory

LSTM
Sequential Modeling

- LSTM adds more gates to the system and have two kinds of memory, a long and a short term memory.
- The new gates are **Forget**, **Update** and **Output** gate.
- **Forget** gate keep/delete from the old long term memory
- **Update** gate to keep/delete from the new long term memory
- **Output** gate to control the new short term memory.

LSTM Explained

Sequential Modeling



LSTM

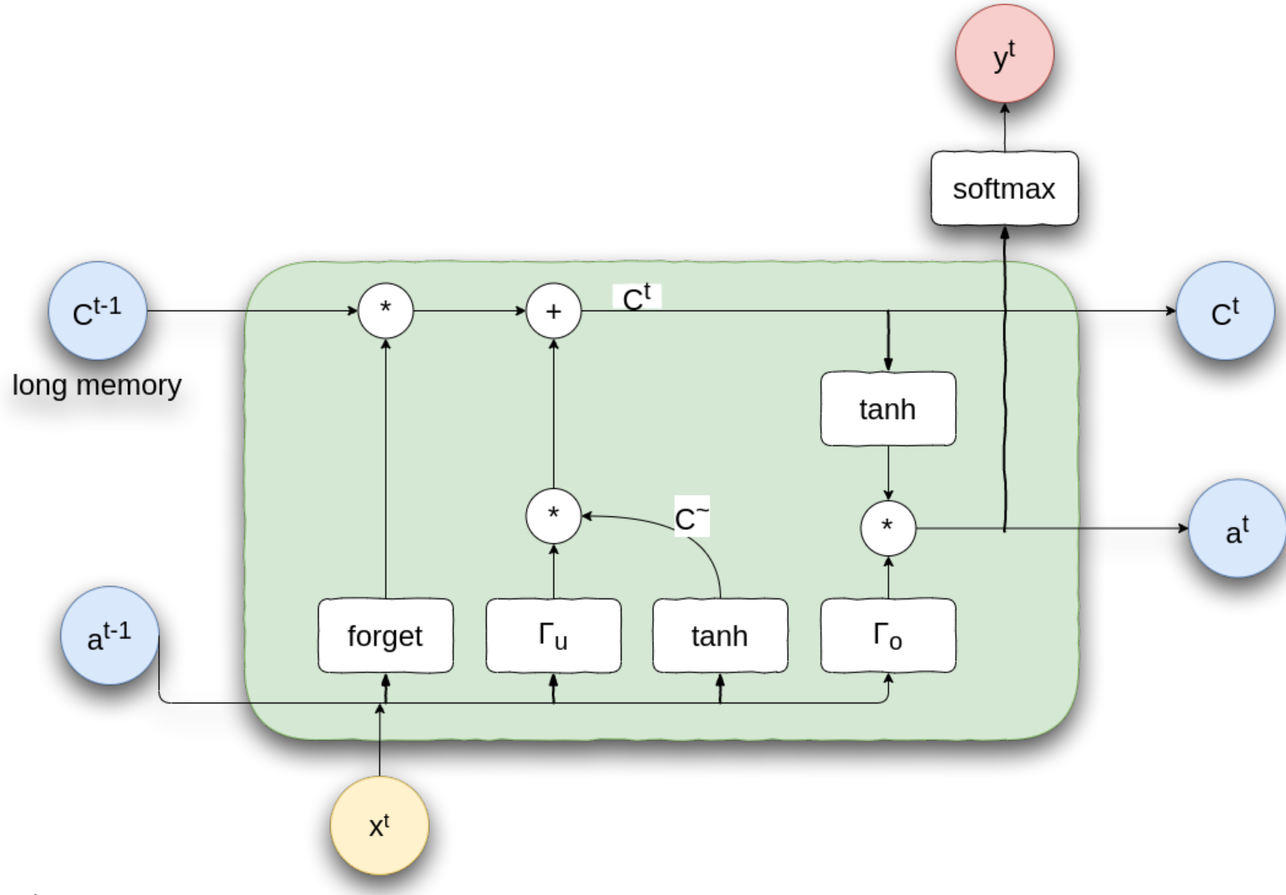
Sequential Modeling

$$\Gamma_f = \sigma(W_f[a^{t-1}, x^t] + b_f)$$

$$\Gamma_u = \sigma(W_u[a^{t-1}, x^t] + b_u)$$

$$\Gamma_o = \sigma(W_o[a^{t-1}, x^t] + b_o)$$

$$\tilde{C} = tanh(W_c[a^{t-1}, x^t] + b_c)$$

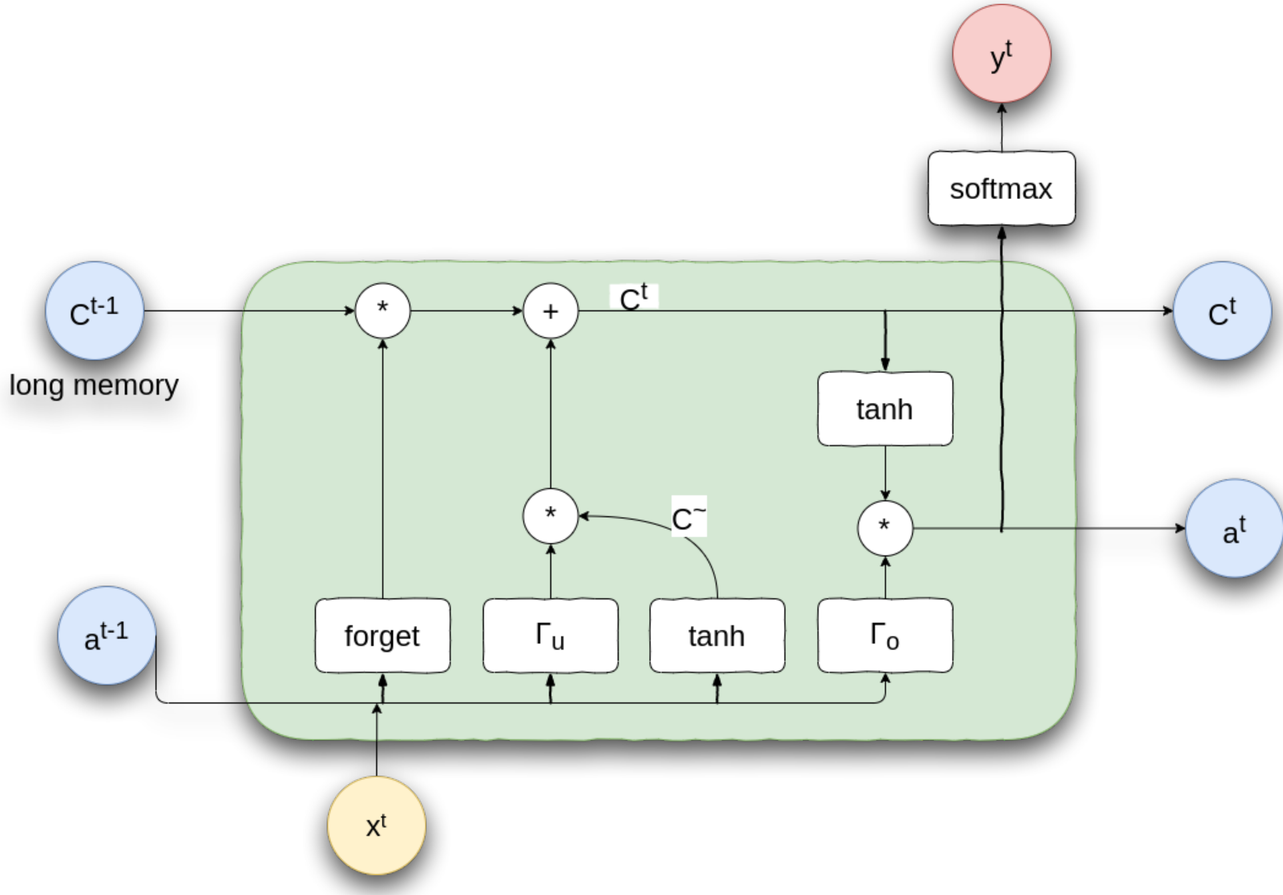


LSTM

Sequential Modeling

$$C^t = \Gamma_u * \tilde{C} + \Gamma_f * C^{t-1}$$

$$a^t = \Gamma_o * tanh(C^t)$$

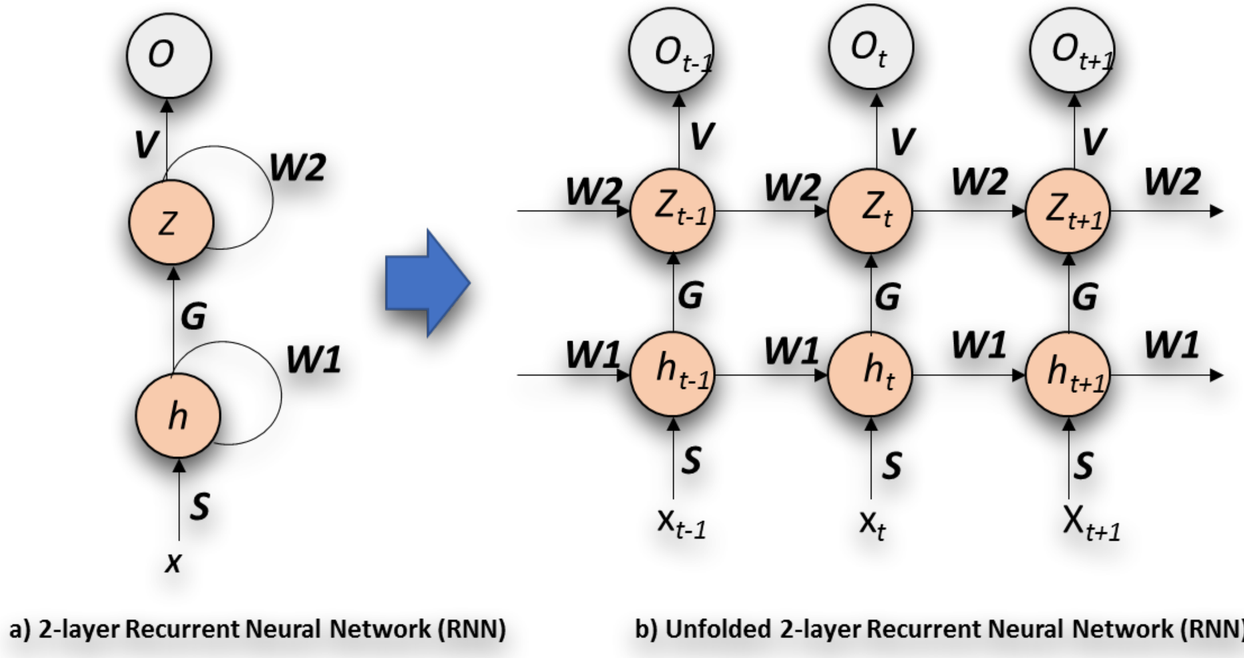


Multi Layer RNN/GRU/LSTM

Multi layer RNN

Sequential Modeling

- Each cell will have to wait for all previous cells in all layers, so it will get complicated pretty fast.
- 3 layers RNN is considered very complicated.



a) 2-layer Recurrent Neural Network (RNN)

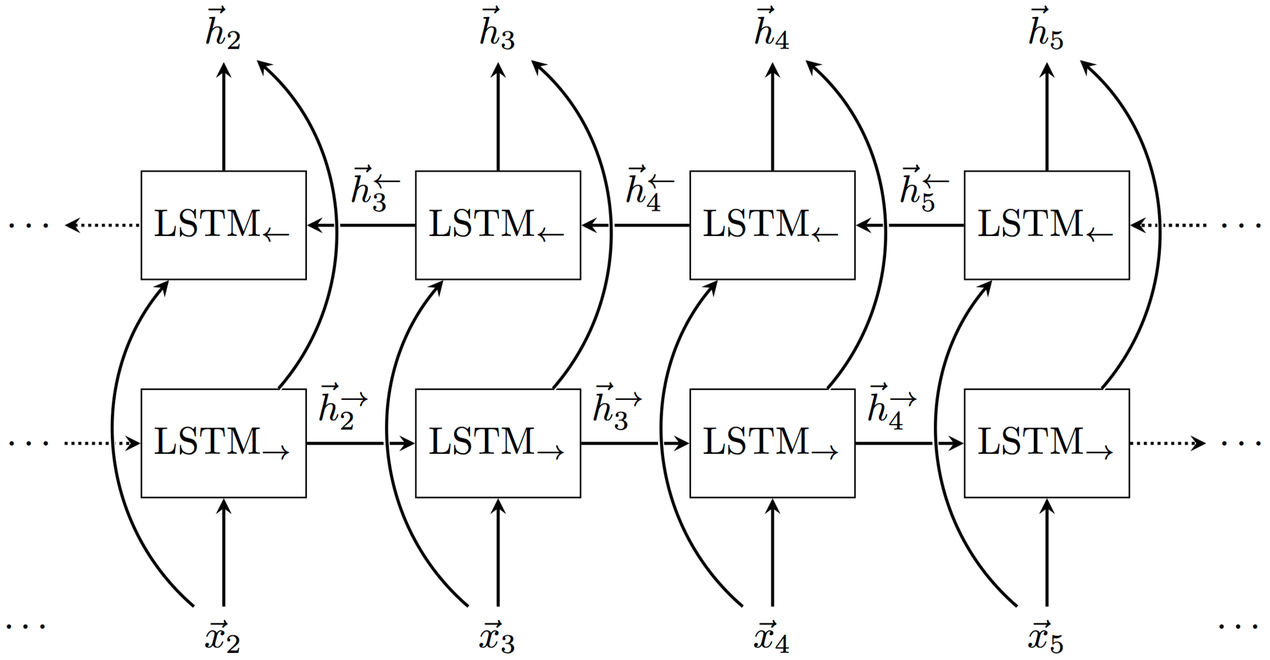
b) Unfolded 2-layer Recurrent Neural Network (RNN)

Bi-directional RNN

Bi-directional RNN

Sequential Modeling

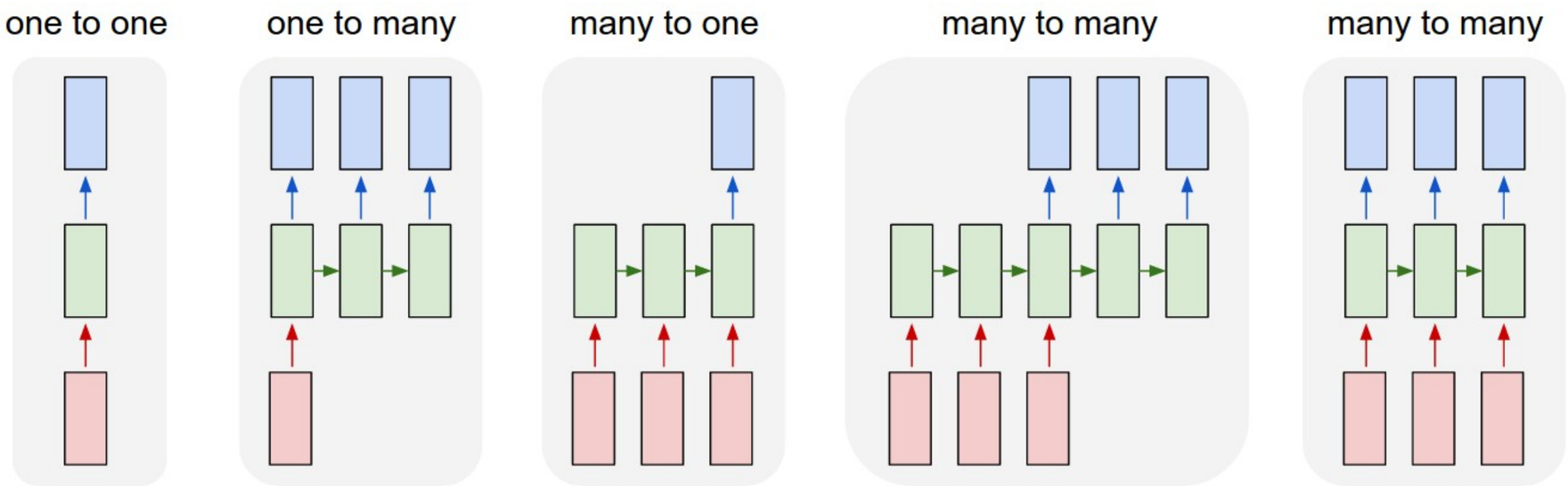
- Bi-Directional RNNs sees in both directions past and forward.
- It's useful when the future token can affect the current.
- More complicated.



Using RNNs

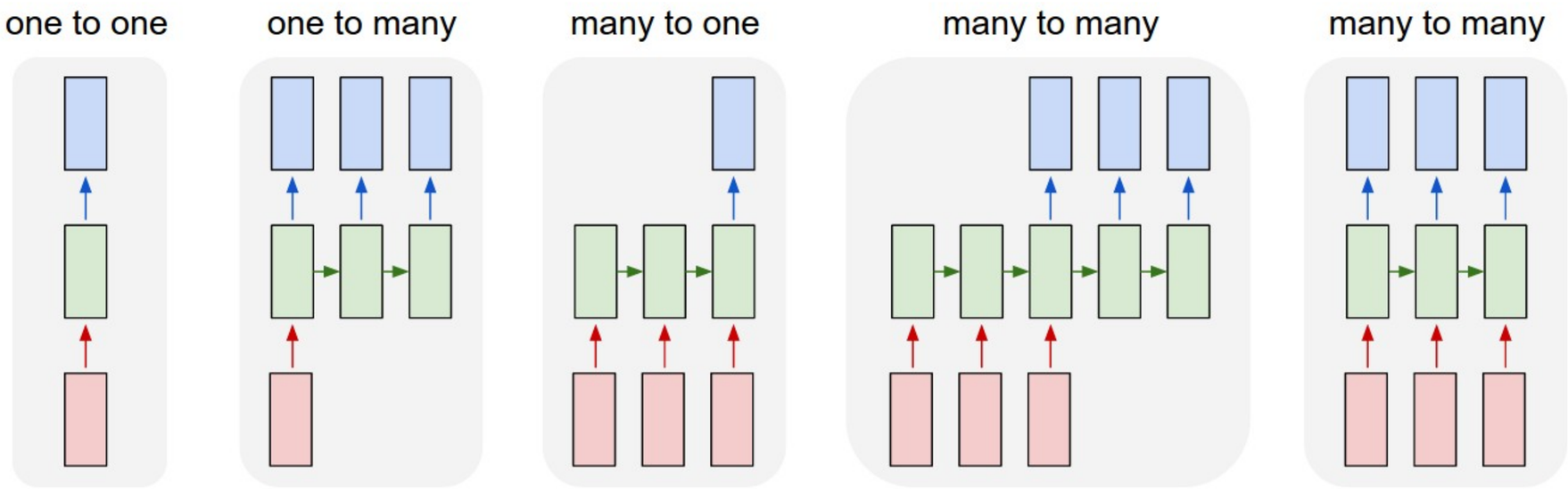
RNN Usage
Sequential Modeling

- **One to Many** tasks include text generation, translation decoder, image to text.



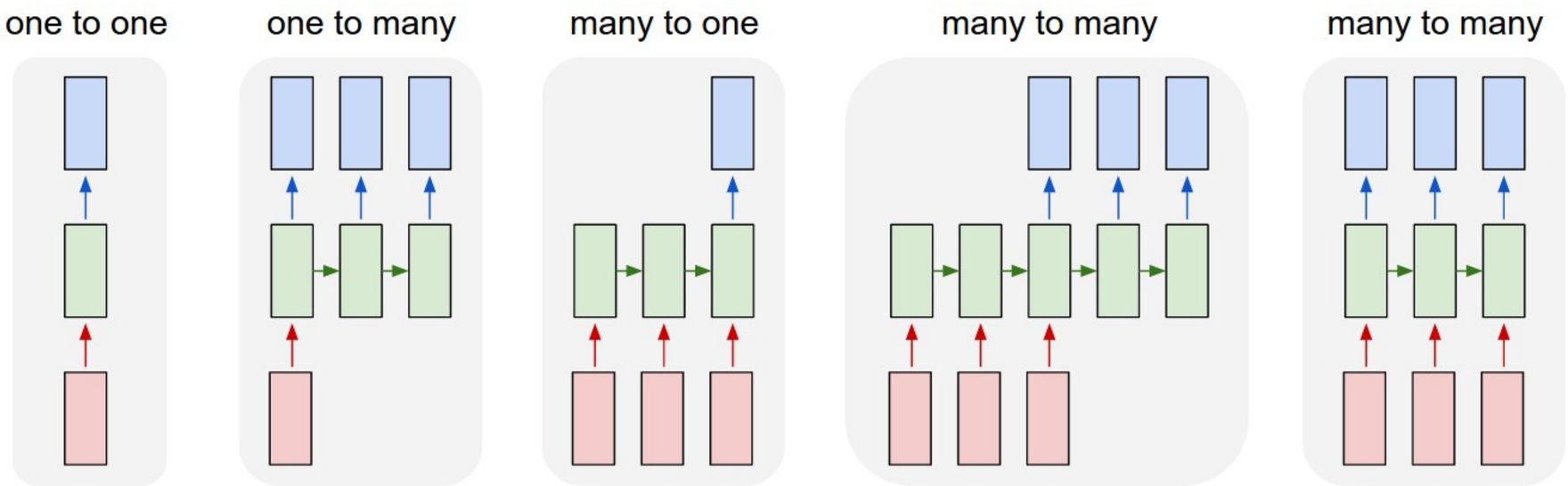
RNN Usage
Sequential Modeling

- **Many to one** tasks include text classification, sentence embedding.



RNN Usage
Sequential Modeling

- **Many to many** tasks include machine translation, text summarization, question answering, sequence labeling.



Notebook - 2

text classification using LSTM

Notebook - 3

Sequence labeling using LSTM

Notebook - 4

text generation using RNN

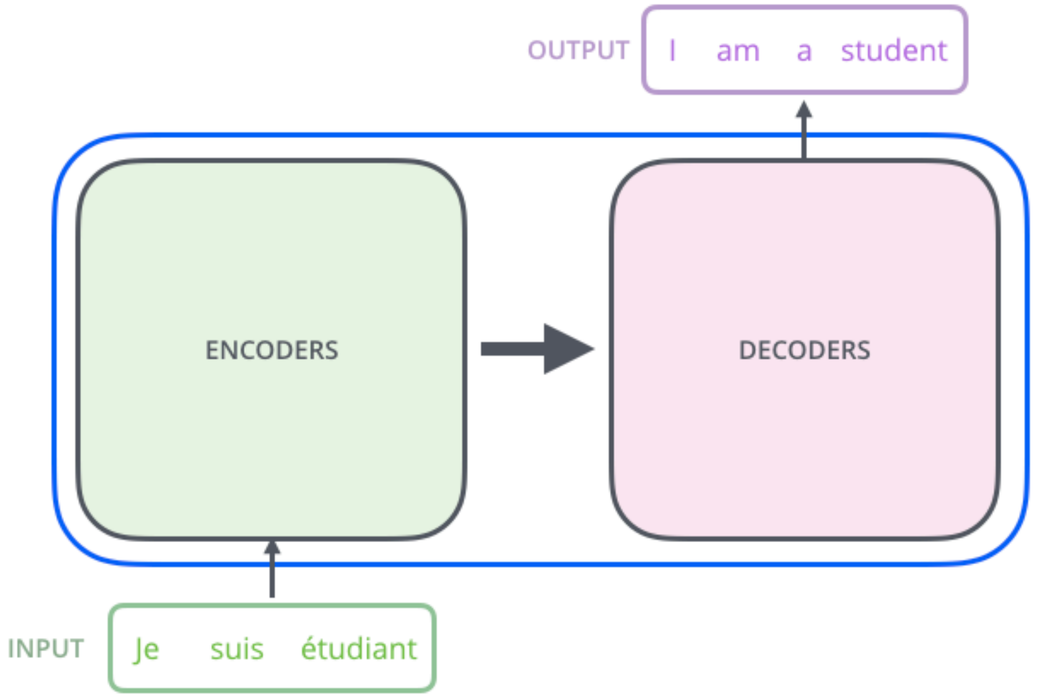
Questions ?

Word Embeddings

Neural Machine Translation

Seq2seq
NMT

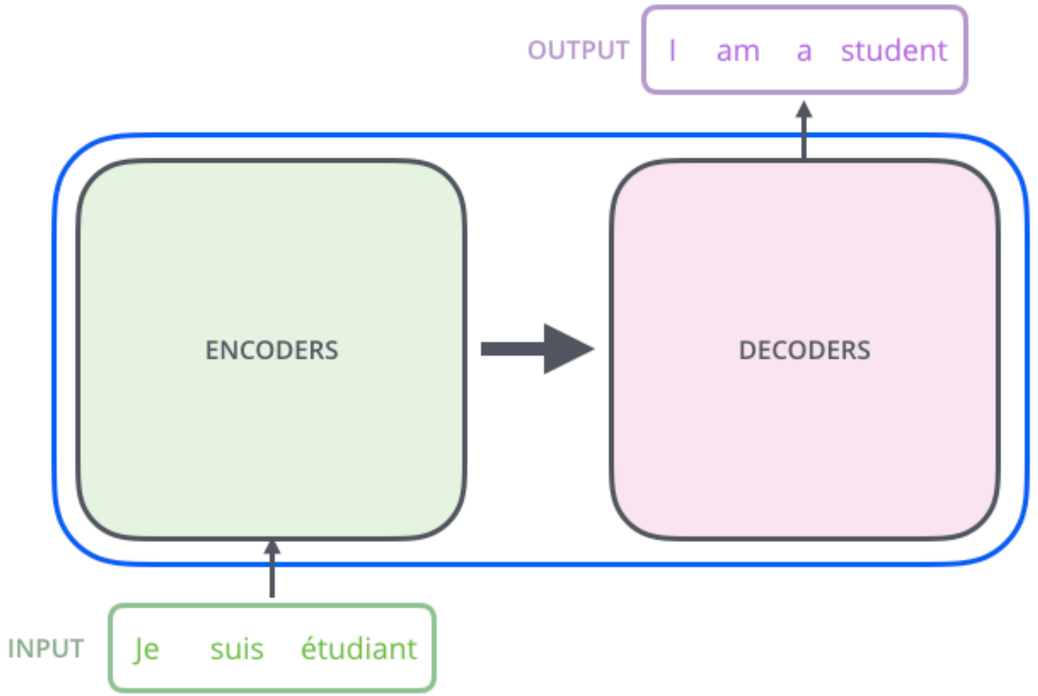
- seq2seq is a very powerful set of models in deep learning that can do machine translation, text summarization and more.
- It was the model in google production for translation around 2016.



source

Seq2seq
NMT

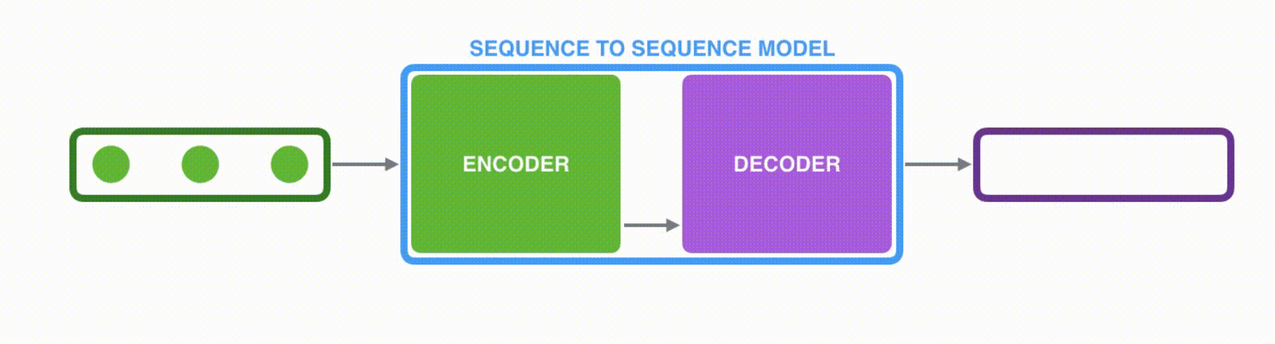
- The encoder and decoder components can be different kind of architecture for example you can use RNN/ LSTM or even a transformer based model like Bert !



source

Seq2seq
NMT

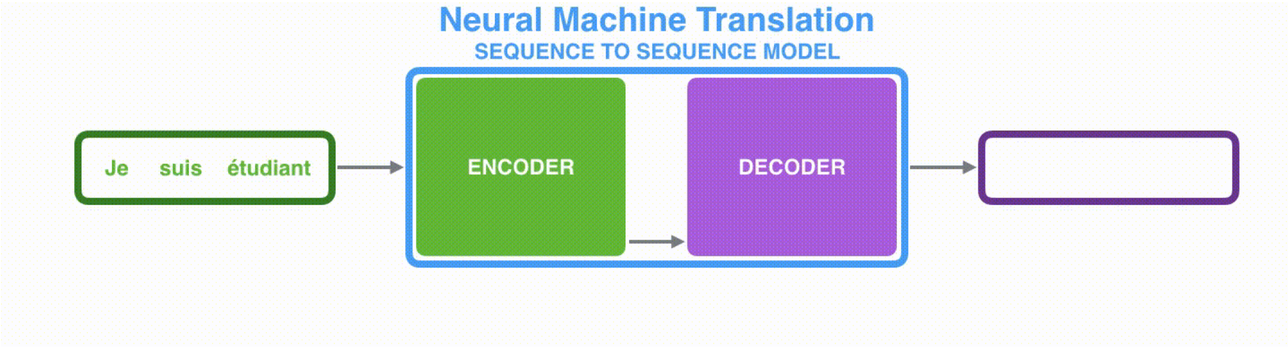
- The encoder objective is to encode all the information necessary for the process (here translation).
- The decoder objective is to reconstruct the input in different language.



source

Seq2seq
NMT

- In NMT the input can be very large sometimes and the encoder might not be able to encode all the information in one context vector.
- Attention was produced to solve this problem.

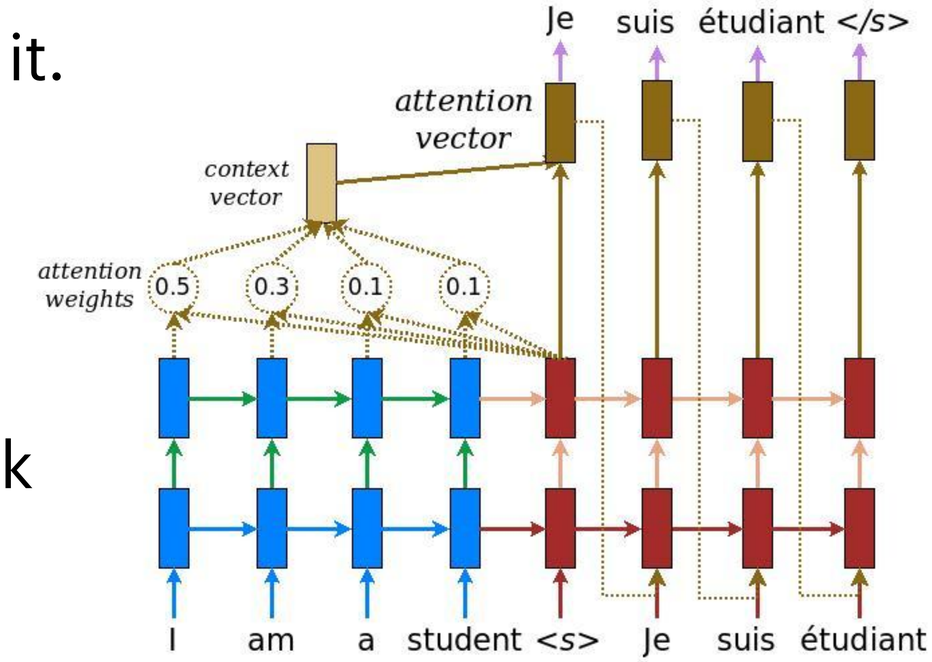


source

Attention in NMT

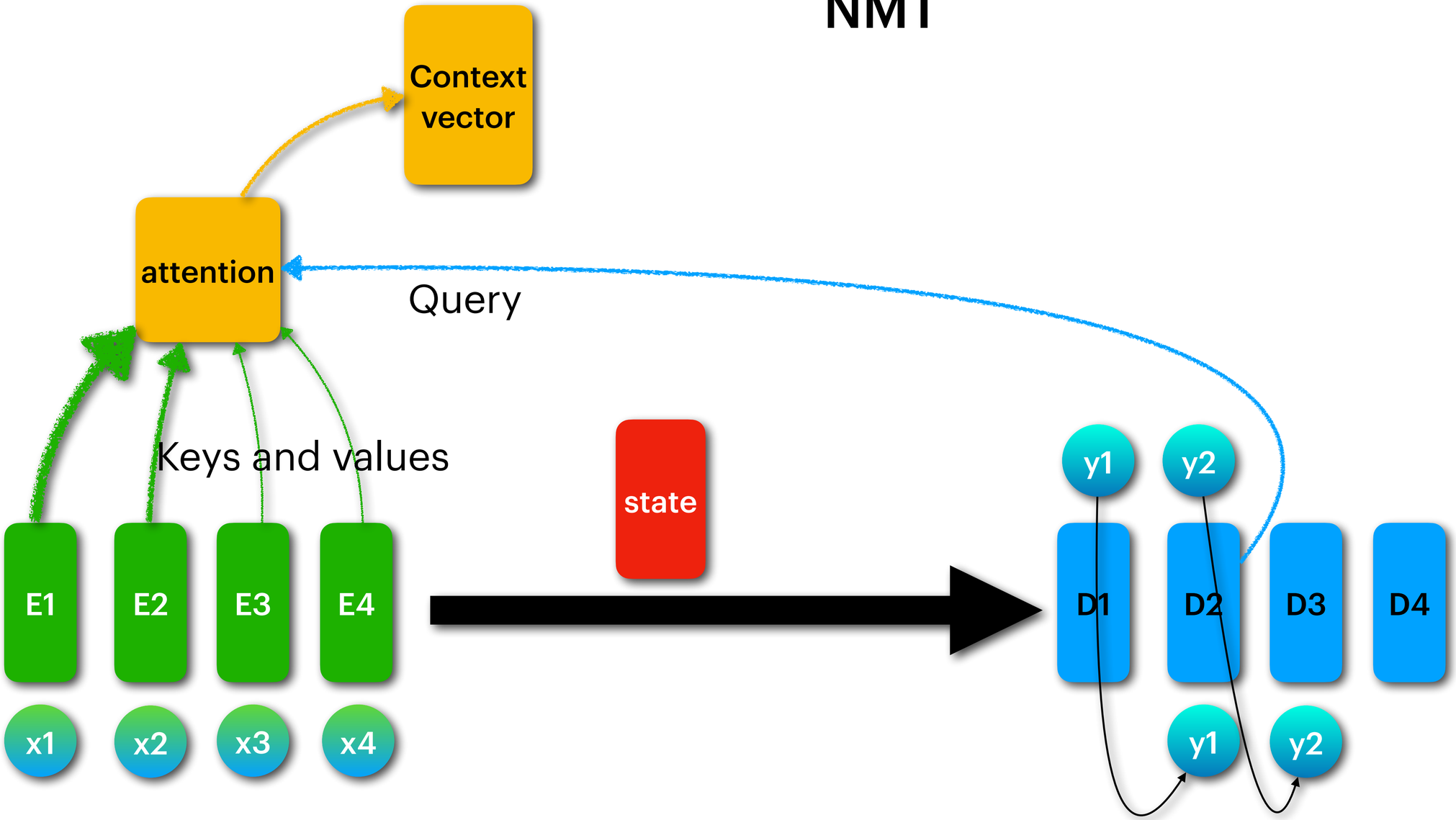
NMT

- Attention supports the decoder output at every prediction.
- It helps every cell focus on what matters to it.
- The decoder uses the attention to build a context vector that is then used to generate the output logit.
- Every output step of the decoder is fed back to the next step as input.



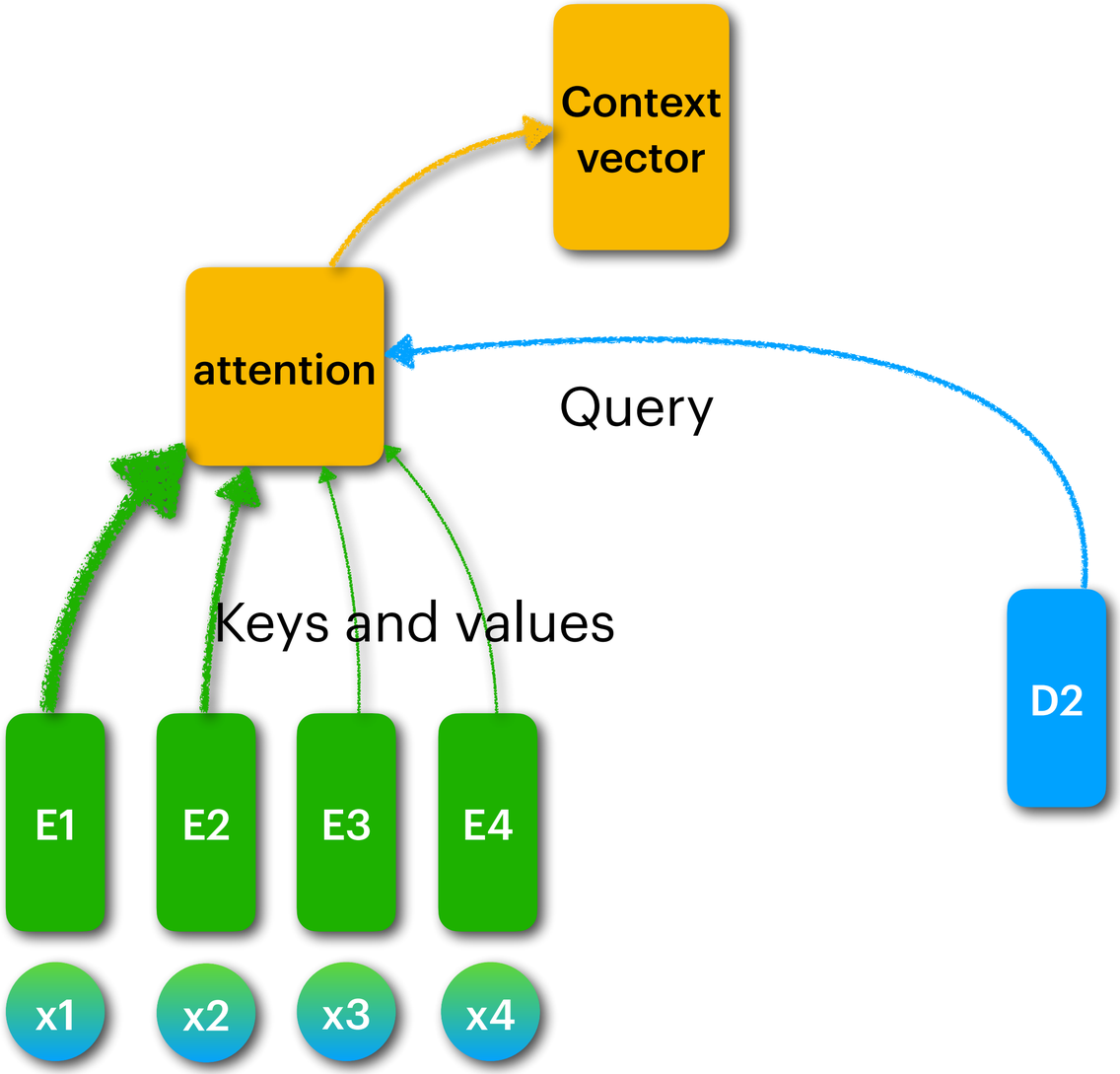
Attention in NMT

NMT



Attention in NMT

NMT



$$\alpha_{de} = \frac{exp(score(h_e, h_d)))}{a \sum_{e'=1}^{e} exp(score(h_{e'}, h_d)))}$$

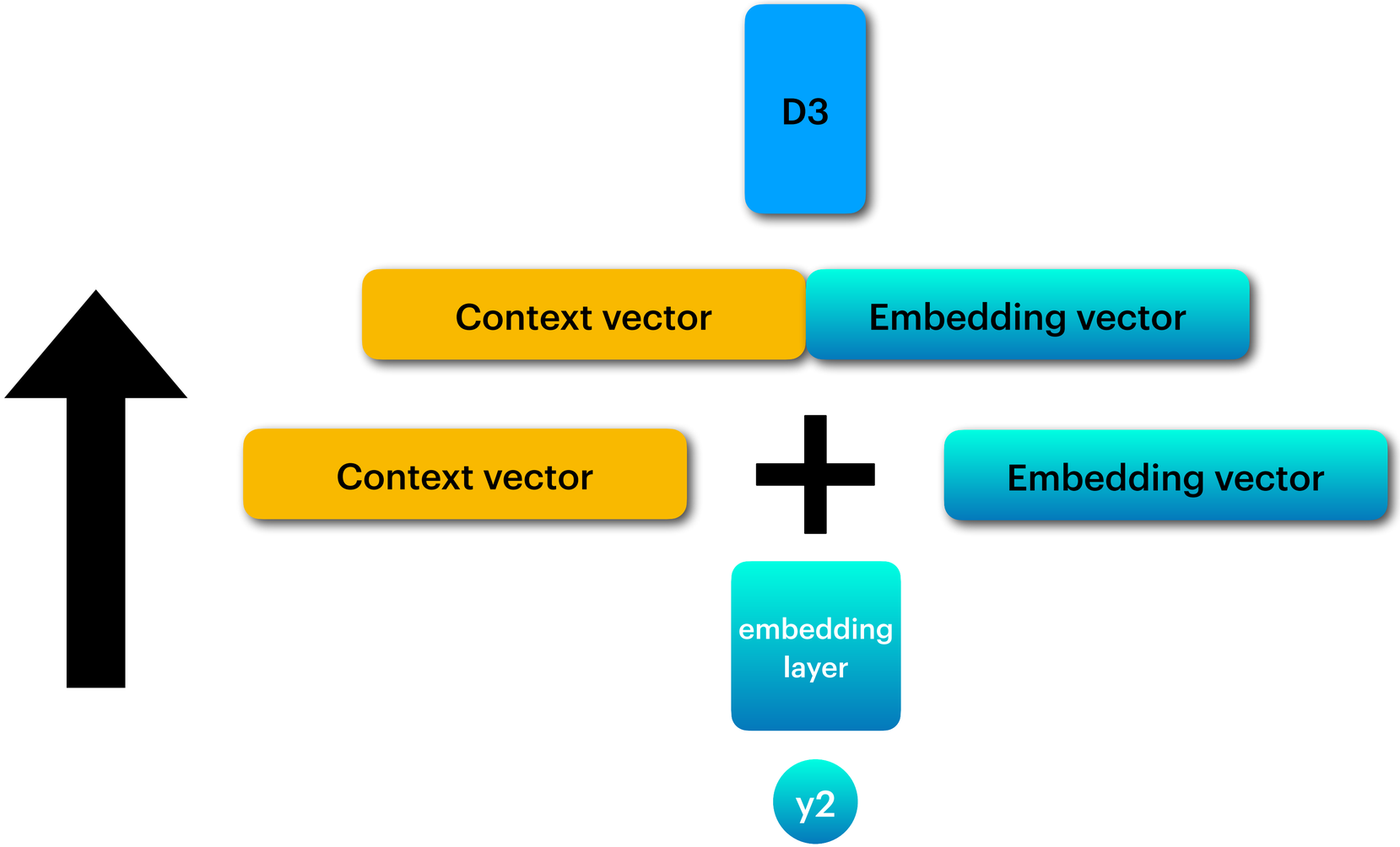
Attention weights

$$c_t = \sum_e \alpha_{de} h_e$$

Context vector

Attention in NMT

NMT



Notebook - 5

NMT

Questions ?

NMT


Thank you !