Recurrent Neural Network for NLP

Sequential NLP Data

Sequential Data

- Characters in words
- Words in sentences
- Sentences in paragraphs
- Paragraphs in documents

NLP is full of **sequential** data

Dependencies in Language

Dependencies: the relationship between two words (it can be semantic or syntax)

- It is equal to the sequential information contained in the sequence data.

Long-distance Dependencies in Languages

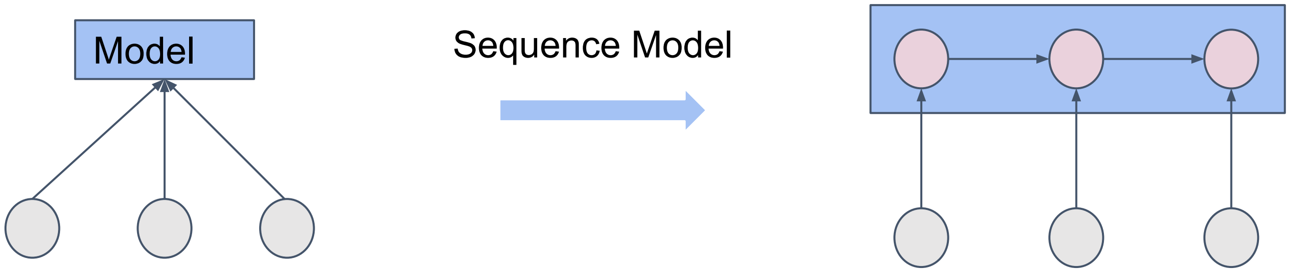
Examples

- He does not have very much confidence in himself

- The rain has lasted as long as the life of clouds

- The *trophy* would not fit in the brown *suitcase* because it was too small

Sequential NLP Data



Machine learning models should capture this kind of sequential information in NLP data.

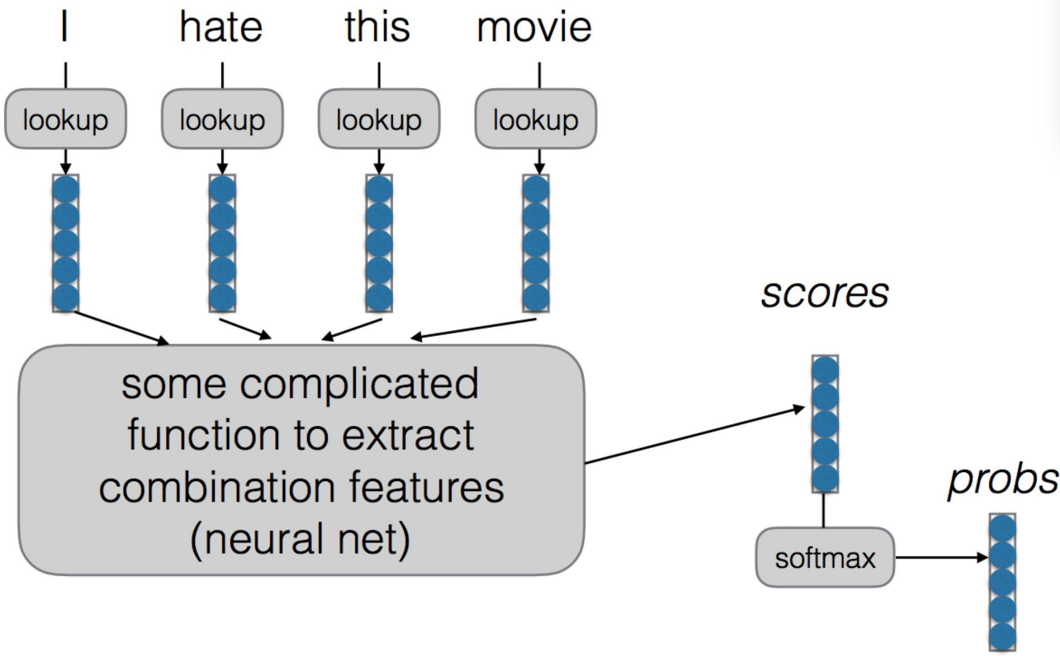
Complex Semantic

1. **Input Text:** a sequence of words;

2. **Through Word Embedding Look-up:** a sequence of word vectors;

3. Neural networks is applied upon the vector sequences to learn semantic **composition** for final prediction;

➔ Human understand the word meaning firstly, then get the whole sentence meaning by composing these words' meaning together.



RNN

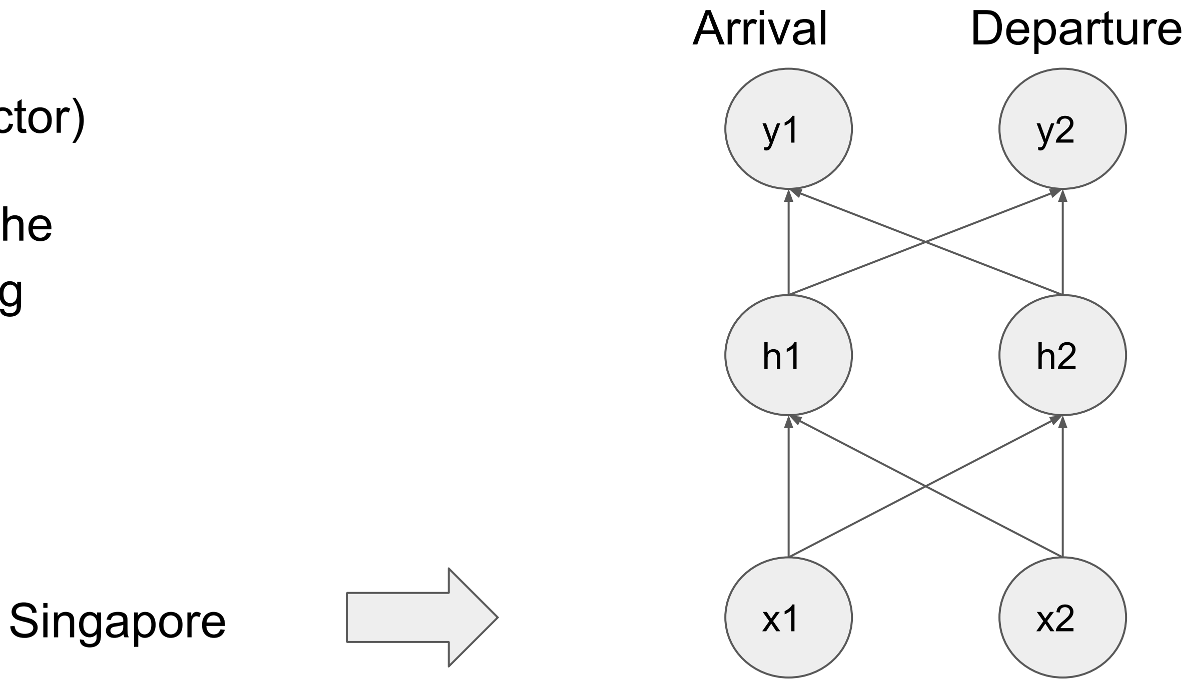
Examples

- Tagging Task: Assuming we have a predefined set of tags, we assign a tag to each word in input sentences.
- Given an input sentence: I would like to arrive at Singapore on Mar. 29.

Singapore -> Dest. Place
Mar. 29 -> Time of Arrival

Feed-forward Network

- Input: Each word (word vector)
- Output: Prob. Scores that the input word belong to the tag



Confusing Case

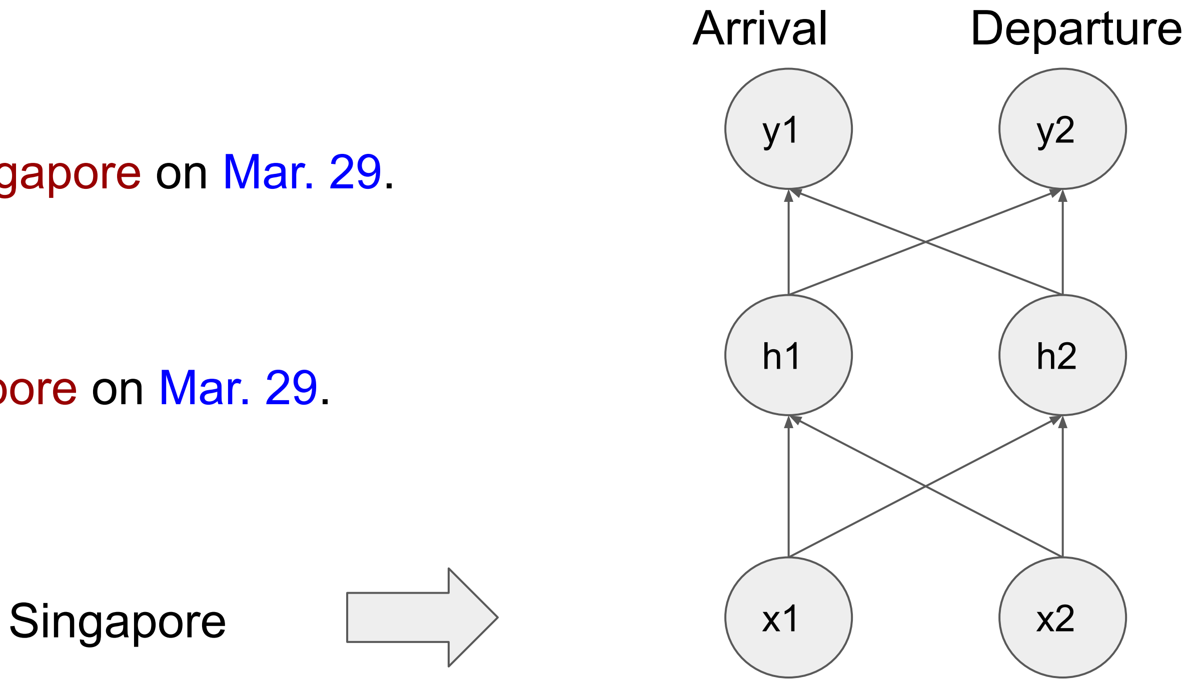
- Input Sentence 1:

  I would like to arrive at Singapore on Mar. 29.

- Input Sentence 2:

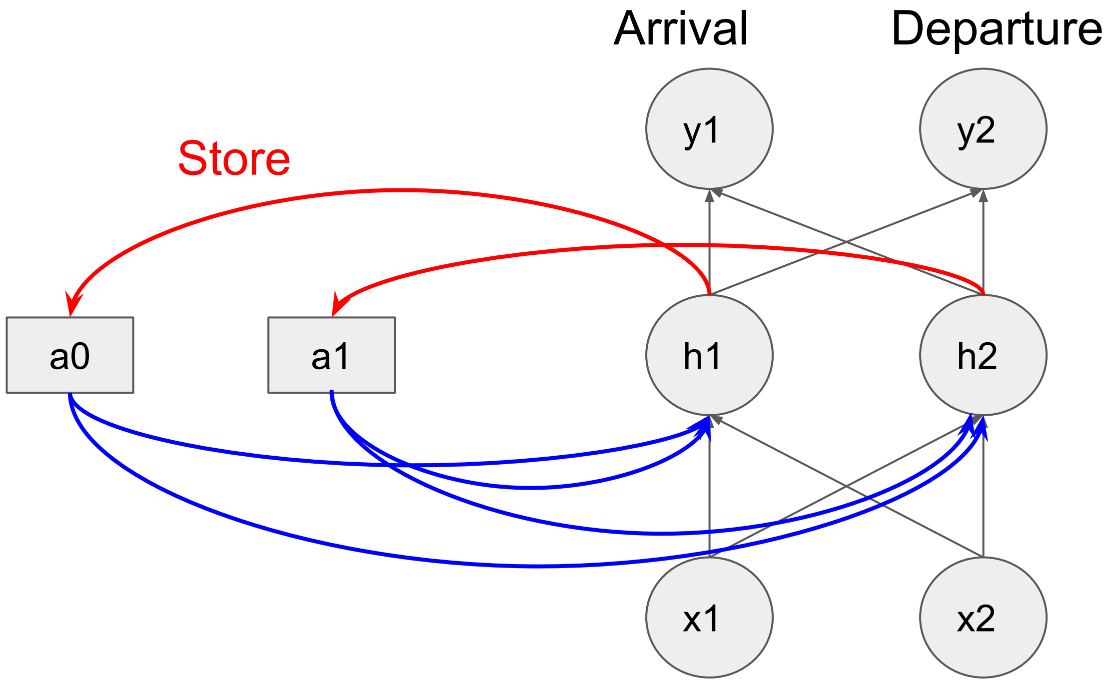
  I would like to leave Singapore on Mar. 29.

Neural Network needs Memory



Neural Network with Memory

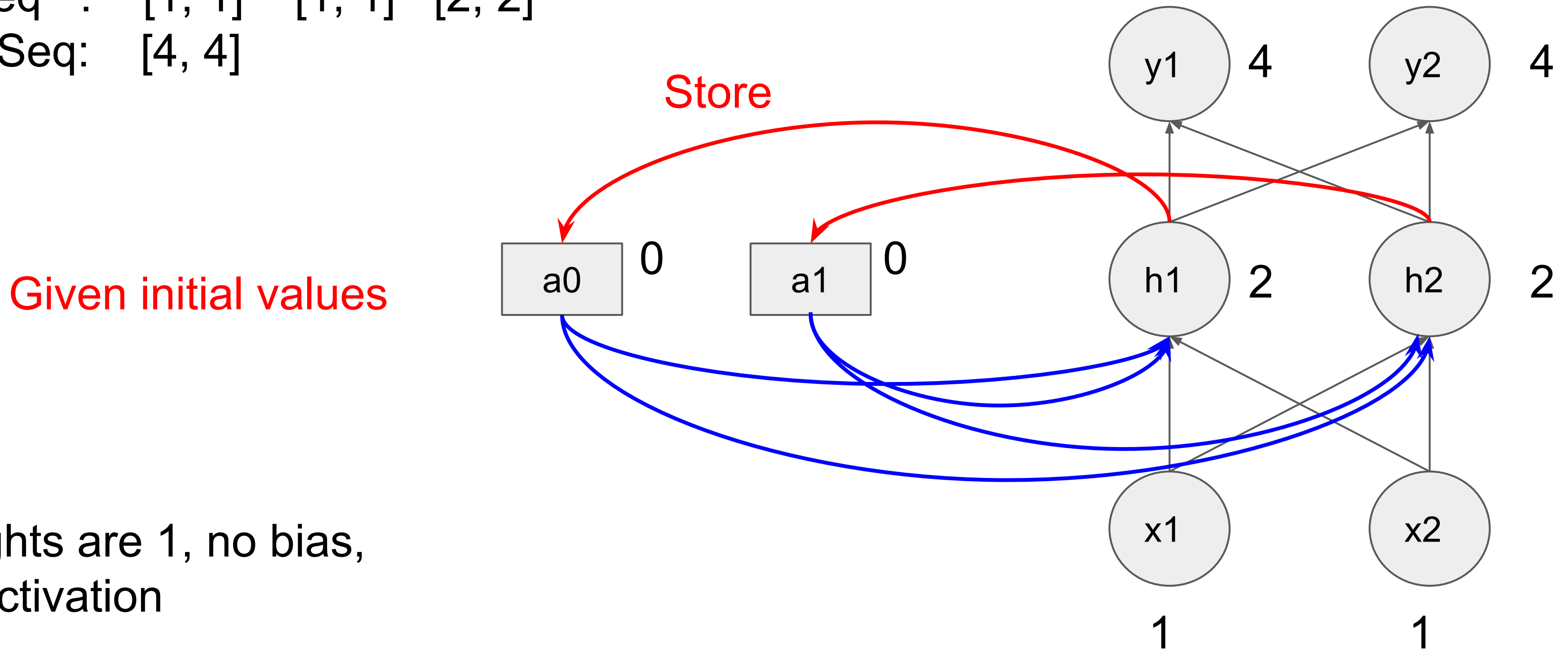
The outputs of hidden layer are stored in the memory.



Memory can be considered as another input

Neural Network with Memory

Input Seq : [1, 1] [1, 1] [2, 2]
Output Seq: [4, 4]

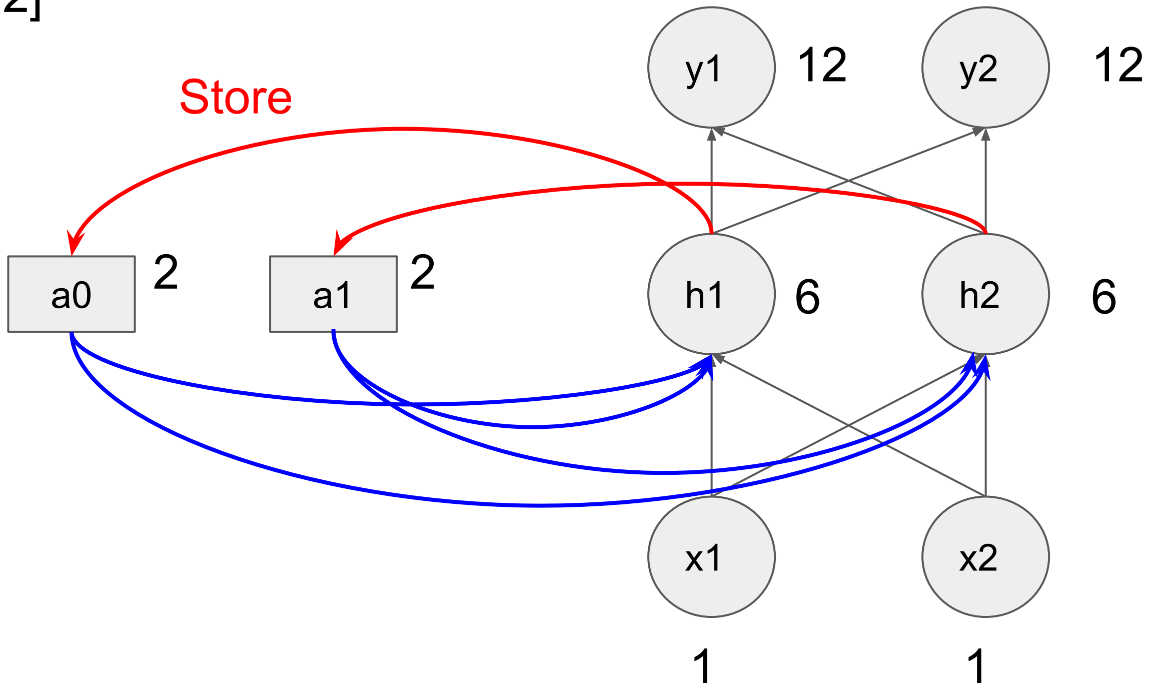


All weights are 1, no bias,
linear activation

Neural Network with Memory

Input Seq : [1, 1] [1, 1] [2, 2]
Output Seq: [4, 4] [12, 12]

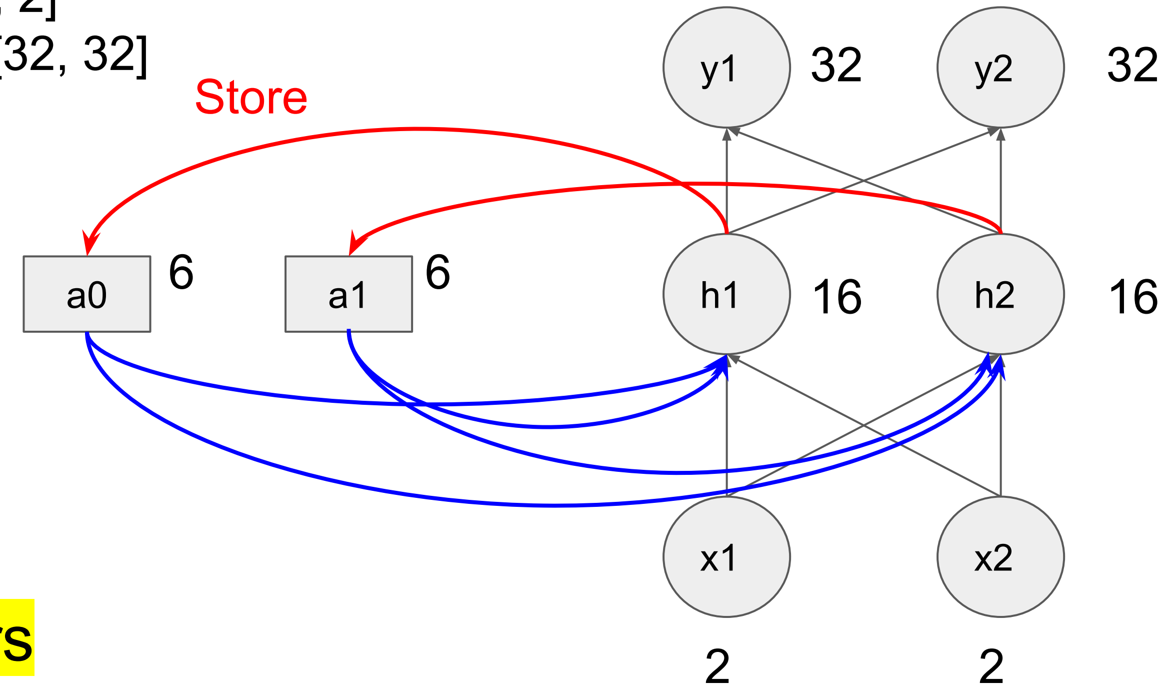
Using previous
hidden output



All weights are 1, no bias,
linear activation

Neural Network with Memory

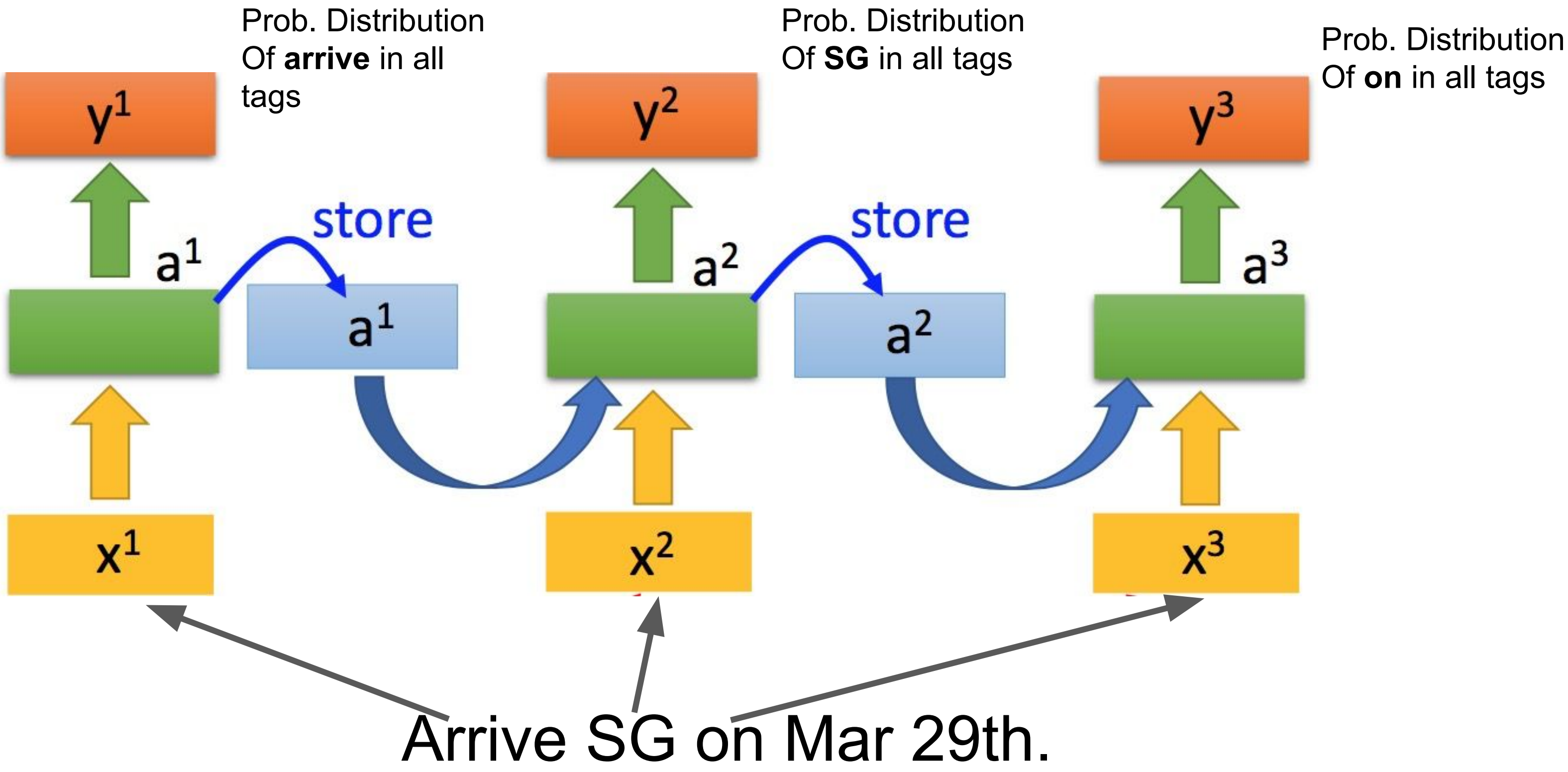
Input Seq : [1, 1] [1, 1] [2, 2]
Output Seq: [4, 4] [12, 12] [32, 32]



Try [1, 1] [2, 2], [1,1]

Sequence Order Matters

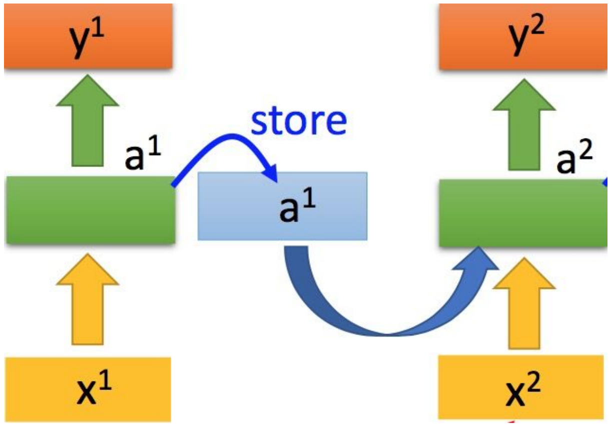
Back to Our Case



Back to Our Case

Prob. Distribution
Of **arrive** in all
tags

Prob. Distribution
Of **SG** in all tags

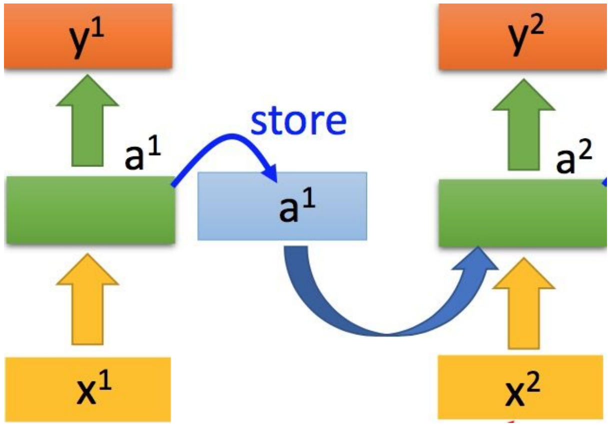


Arrive

SG

Prob. Distribution
Of **leave** in all
tags

Prob. Distribution
Of **SG** in all tags



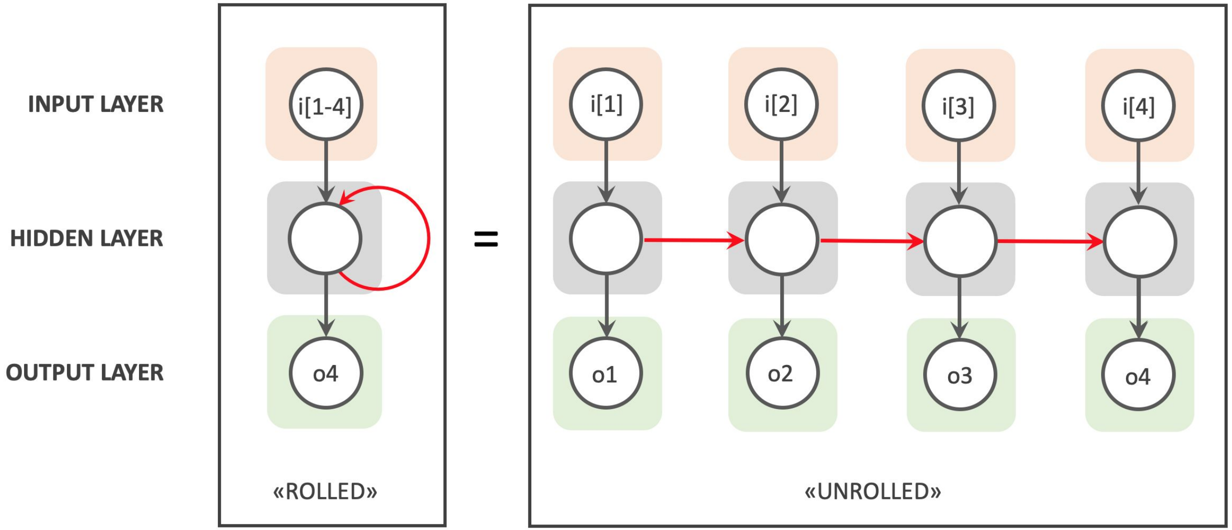
Leave

SG

Are these two probs. distribution of SG same ? And Why?

Recurrent Neural Network (Elman 1990)

- Recurrent neural network is proposed to utilize information from **previous** time steps and **current** information to make reasoning at the current step



https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

Neuron computation of RNN

- Model parameters are tied across all time steps (run the **same** RNN cell)

Hidden layers vector **at time step t**

Input vector **at time step t**

$$\mathbf{h}_t = \sigma(\mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{W}^{hx}\mathbf{x}_t)$$

Time-Invariant Model Parameters

Predictions at time step t

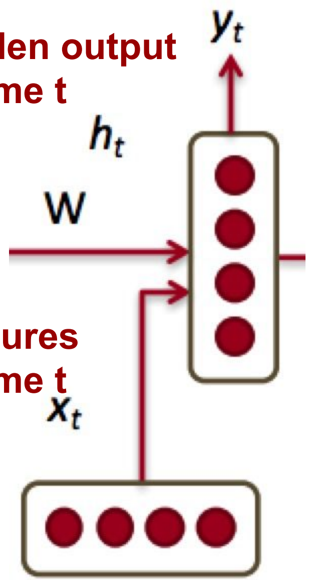
$$\tilde{y}_t = softmax(\mathbf{W}^s\mathbf{h}_t)$$

- We need $h_0 \in \mathbb{R}^{D_h}$ as the initialization vector for the hidden layer at time step **0**.

- Inputs enter and move forward at each time step

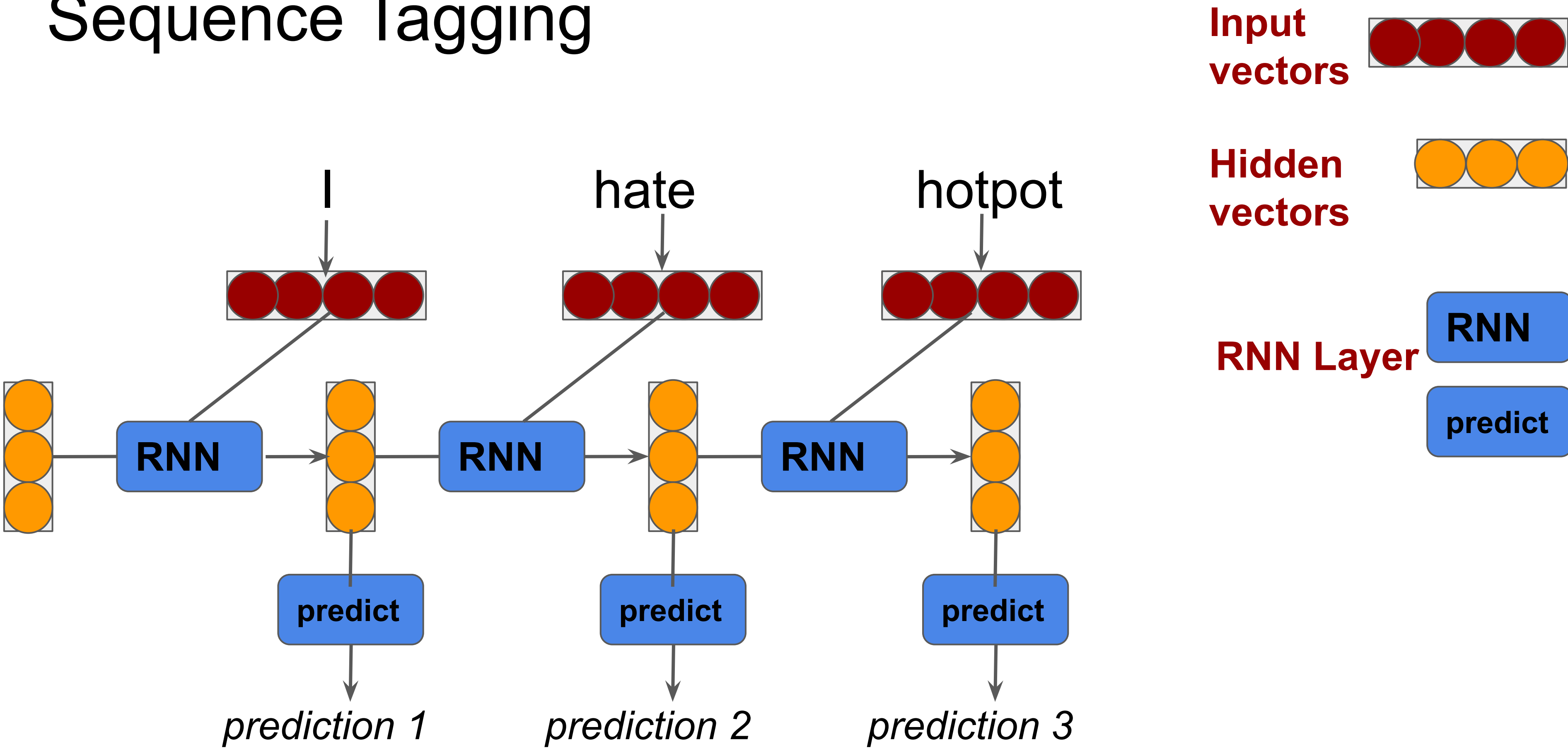
Hidden output at time t

Features at time t



Focus on certain time step

Sequence Tagging



Recurrent Neural Network

- Recurrent neural network works in a chain way
- The method is in naturally suitable for processing sequences data
- A broad applications:
    - Speech Recognition
    - Time series Prediction
    - Language Modeling
    - Machine Translation

Language Models

- A language model computes **a probability for a sequence of words**:
  - P(w1,..., wT)
- Useful for machine translation/Question Answering
  - Word Ordering
    - p(the cat is small) > p(small is the cat)
  - Word Choice
    - p(walking home after school) > p(walking house after school)

Traditional Methods

- Probability is usually conditioned on **window of *n*** previous words
- An incorrect but practical **Markov** Assumption
- To estimate probabilities, compute for unigram and bigrams

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

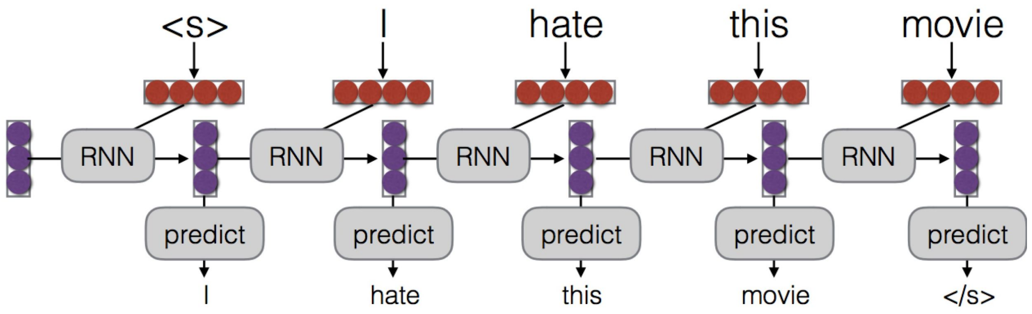
$$p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

Traditional Methods

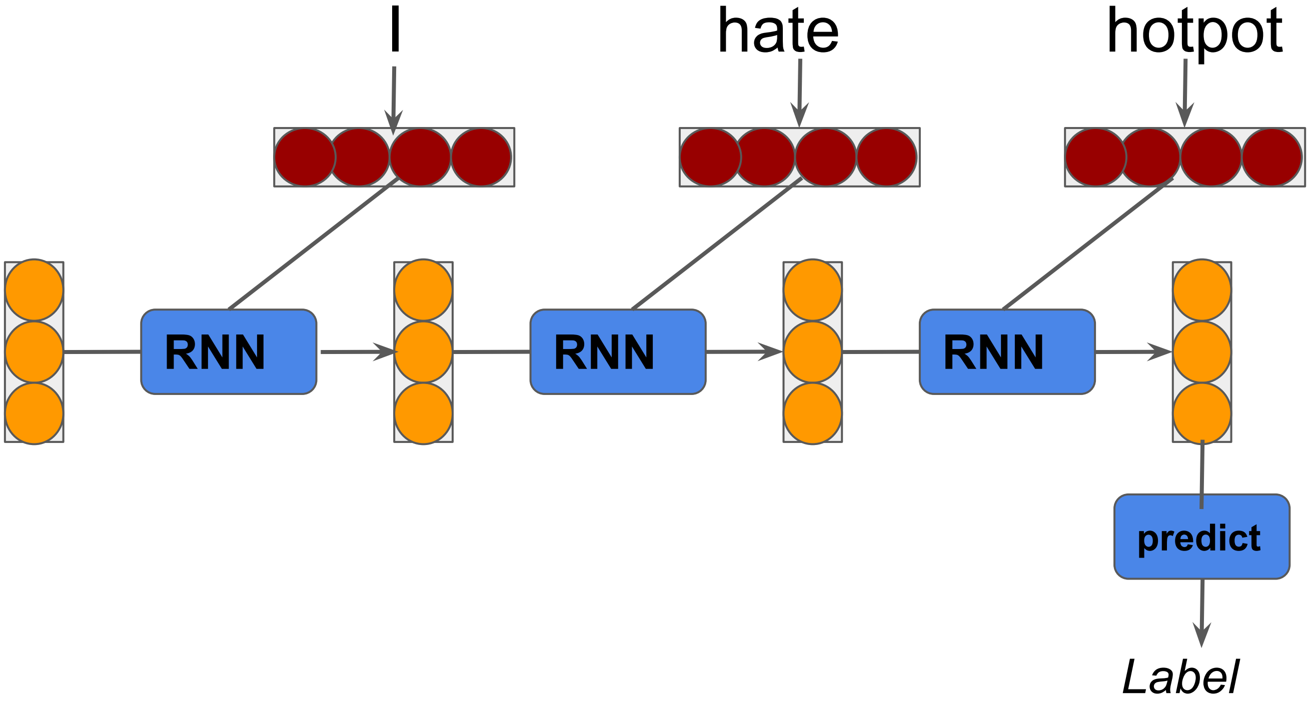
- A lot of n-grams and extremely large combinations
    - Requires large RAM requirements
- Use one machine with 140GB RAM for 2.8days to built a model on 126 billion tokens.

RNN-based Language Model

- A language model computes **a probability for a sequence of words**:
    - P(w1,..., wT)
    - Useful for machine translation, Chatbot and Question Answer Systems.
- Language Modelling can be formulated as a tagging problem
- Each label/tag is the next word!

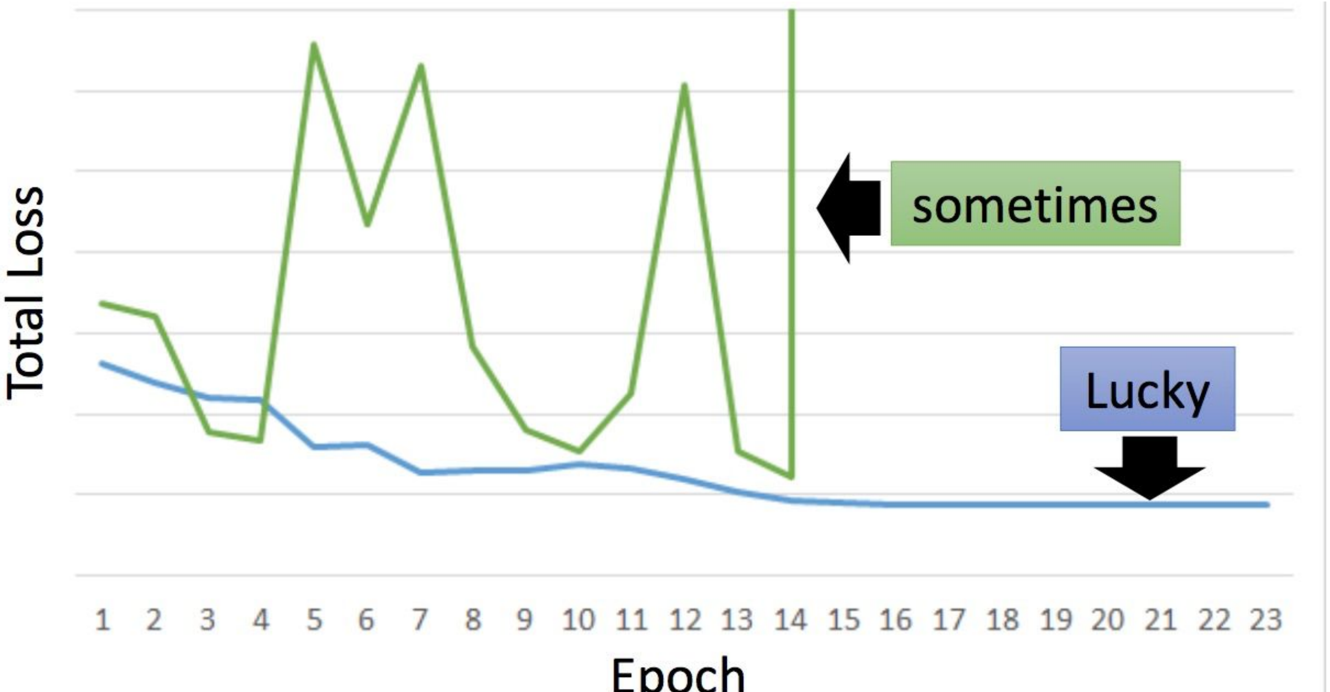


Sequence Classification

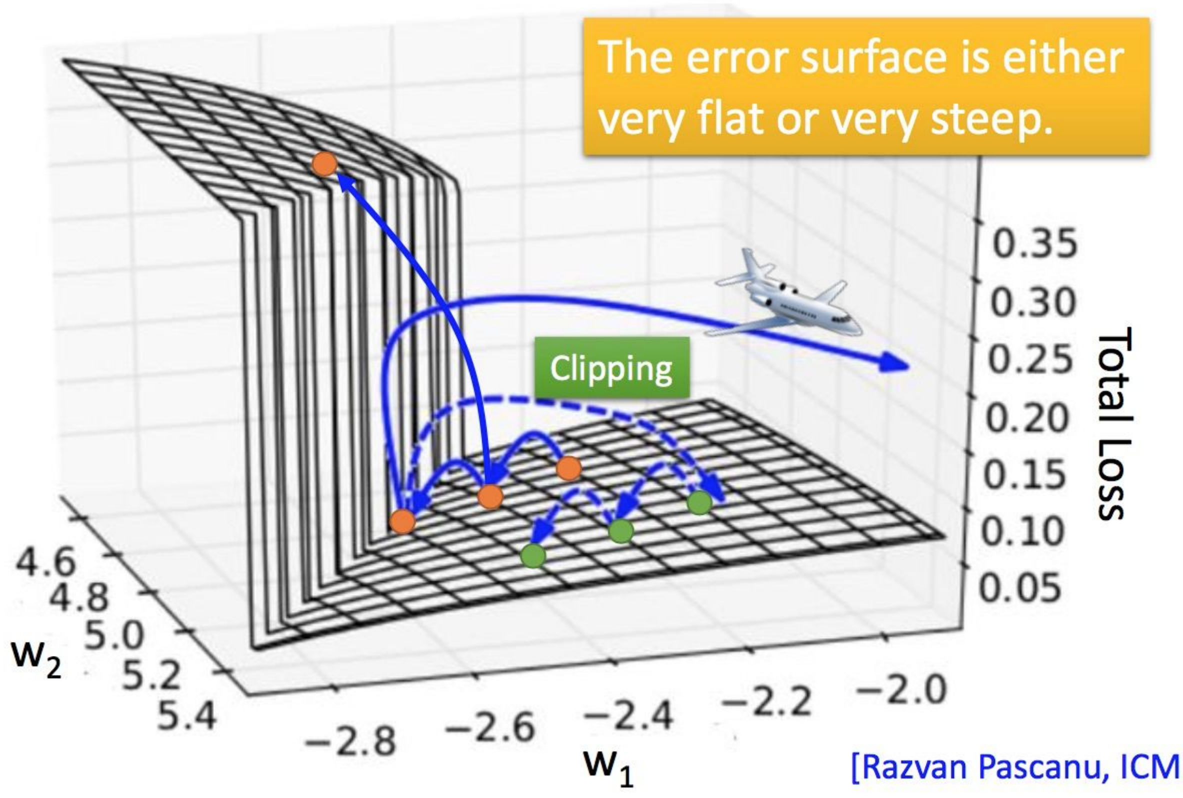


RNN Training is Hard

- Real experiments on Language Models



Rough Error Surface of RNN



Exploding/Vanishing Gradient

[Razvan Pascanu, ICML'13]

Toy Example

$w = 1 \Rightarrow y_{1000} = 1$

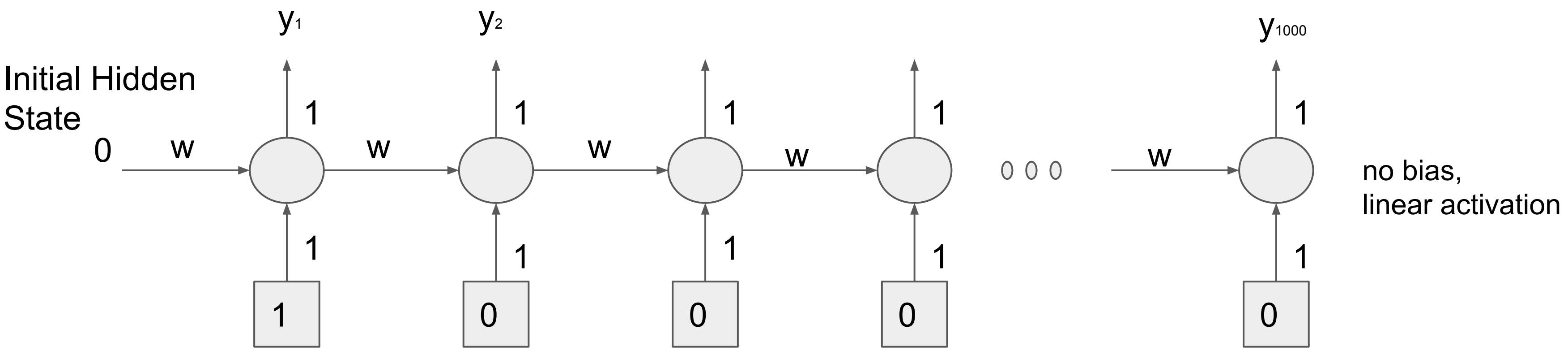
$w = 1.01 \Rightarrow y_{1000} = 20000$

$\Rightarrow$ Large Gradient: dL/dw $\Rightarrow$ **Irregular Error Surface**

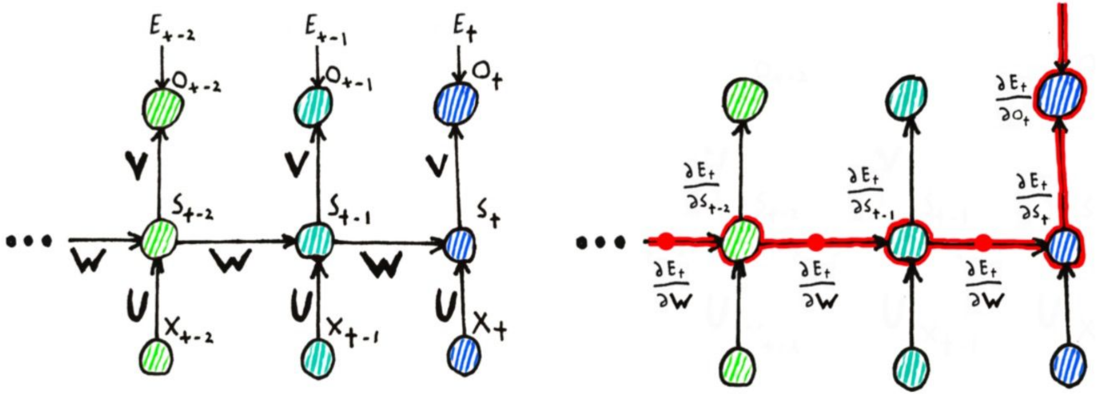
$w = 0.99 \Rightarrow y_{1000} \sim= 0$

$w = 0.01 \Rightarrow y_{1000} \sim= 0$

$\Rightarrow$ Small Gradient: dL/dw $\Rightarrow$ **Irregular Error Surface**



Backpropagation Through Time



$$\frac{\partial E_t}{\partial \mathbf{W}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \mathbf{o_t}} \frac{\partial \mathbf{o_t}}{\partial \mathbf{s_t}} \boxed{\frac{\partial \mathbf{s_t}}{\partial \mathbf{s_k}}} \frac{\partial \mathbf{s_k}}{\partial \mathbf{W}}$$

Chain rule => Multiplications

Can explode or vanish

$$\prod_{j=k+1}^{t} \frac{\partial \mathbf{s_j}}{\partial \mathbf{s_{j-1}}}$$

Exploding Gradient Solutions

- Truncated BPTT
  - Do not take the derivative all the way back to the beginning of the input sequence

$$\frac{\partial E_t}{\partial \mathbf{W}} = \sum_{\boxed{k=t-T}}^{t} \frac{\partial E_t}{\partial \mathbf{o_t}} \frac{\partial \mathbf{o_t}}{\partial \mathbf{s_t}} \frac{\partial \mathbf{s_t}}{\partial \mathbf{s_k}} \frac{\partial \mathbf{s_k}}{\partial \mathbf{W}}$$

Only through T time steps if t>= T

- **Clip gradients at threshold**

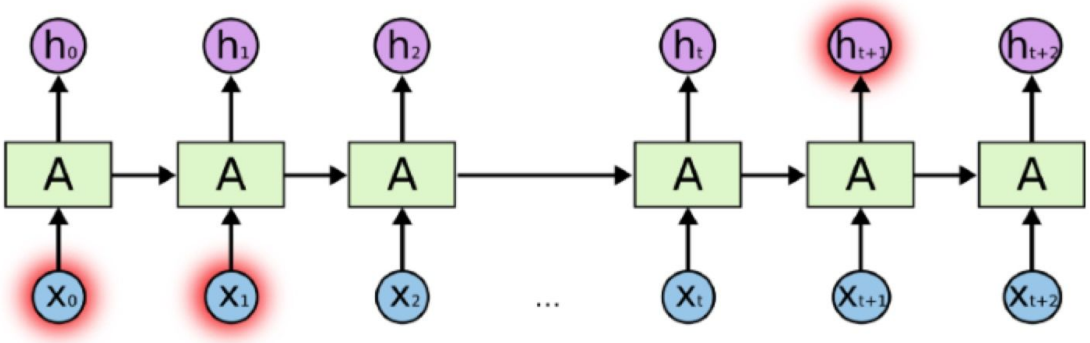
- RMSprop to adjust learning rate
  - Adapt learning rate by dividing by the root of squared gradient

Vanishing Gradient Problem

- The error at a time step **ideally** can tell a previous time step from many steps away to change during backprop
- **Can not capture long-term dependency**

- The representation from time steps 0 and t can not travel to influence the time step t+1

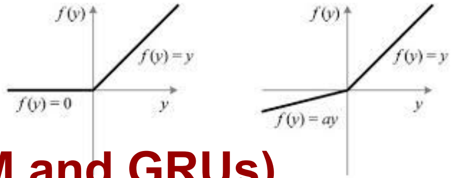
- **Harder to detect**



Vanishing Gradient Solutions

- RMSprop
  - Adapt learning rate by dividing by the root of squared gradient
- Advanced activation functions such as leakyRelu function

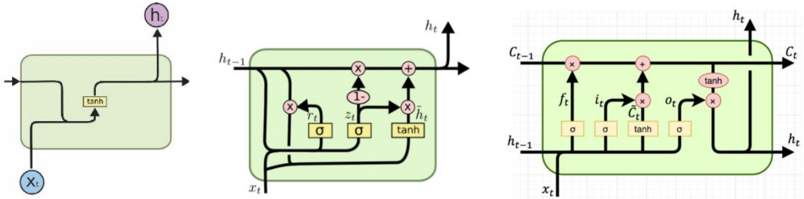
ReLu



Leaky ReLu

- **Gated RNN (LSTM and GRUs)**
  - Using gates in cell computation to control information flow

Partially Solved



RNN GRU LSTM

RNN’s Bottleneck

- RNN is not suitable for parallel computation.

- RNN’s training is not easy
    - Gradient Vanishing
    - Gradient Exploding