

Recurrent Neural Networks

Session outline

RNN history

- Introducing recurrences

Sequence Data

- One is Many

RNN structure

- General Principles

Modern RNN

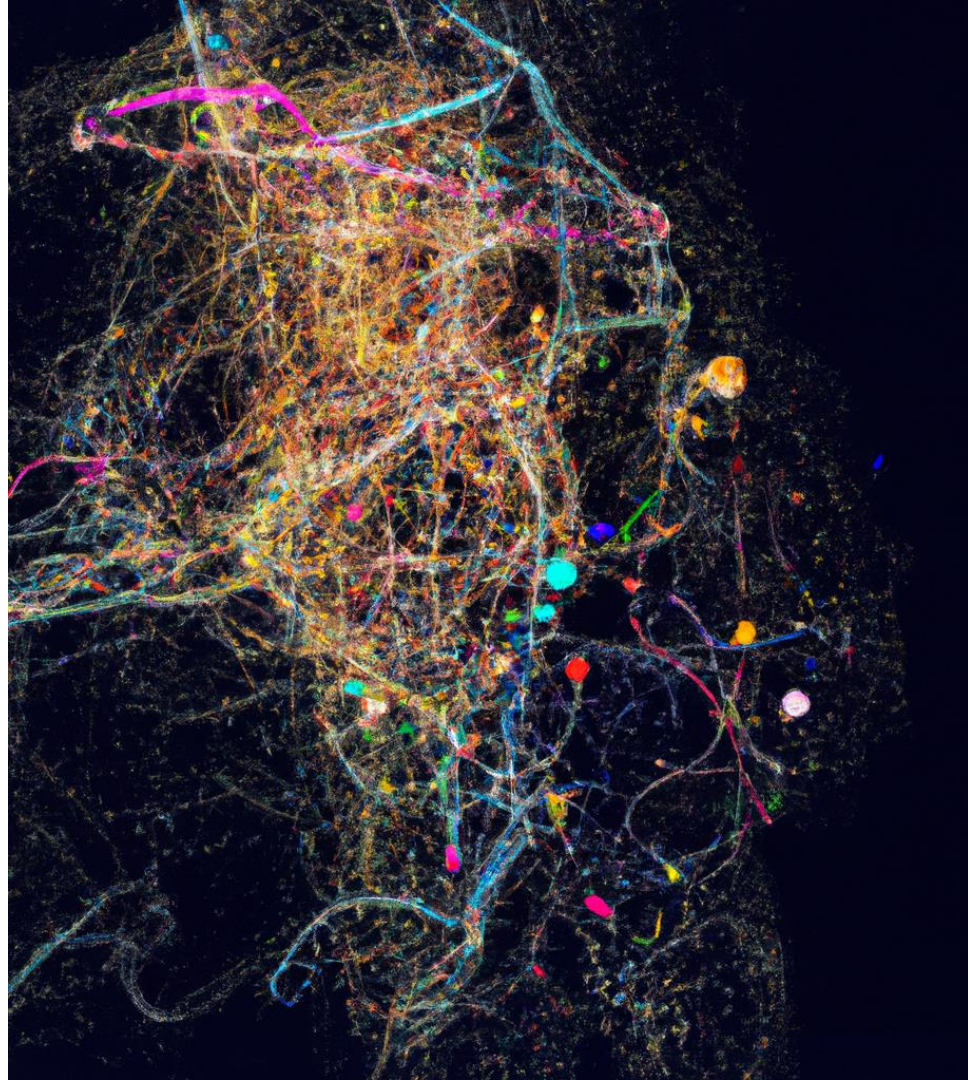
- LSTM & GRU

Embeddings

- Language models

RNN History

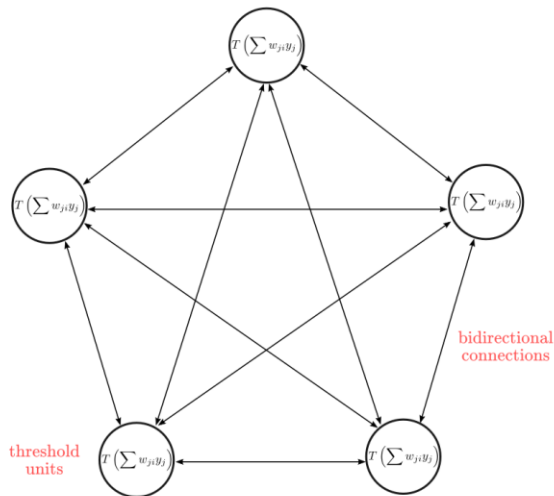
Introducing recurrences



Hopfield Network -1982

A practical introduction to recurrences

- Hopfield Networks were introduced in 1982 by John Hopfield, a physicist at Caltech and helped to reignite the interest in neural networks in the early '80s
- Hopfield aimed to address the question of emergence in cognitive systems through the collective action of simple neurons
- The properties of Hopfield networks derive from a global energy-function with neurons being binary threshold units with recurrent connections
- Each unit is bi-directionally connected to every other unit
- As a result, the weights values are symmetric, such that weights coming into a unit are the same as the ones coming out of a unit

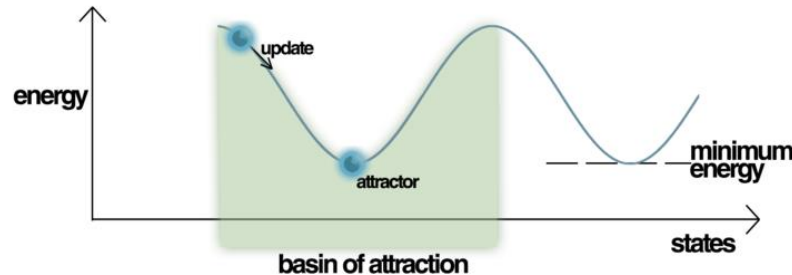


$$E = - \sum_{i < j} w_{ji} y_j y_i - \sum_i b_i y_i \quad \text{weight update} \quad \Delta w_{ji} = y_j y_i$$

Hopfield Network -1982

An interpretation of Hopfield Networks

- Hopfield Networks are based on neuroscience research on learning and memory, particularly Hebbian learning (Hebb, 1949)
- Hopfield proposed this model to capture memory formation and retrieval
- Hopfield networks are systems that "evolve" until they find a stable low-energy state, rather than being driven by error correction
- The system can "return" to a previous stable-state after a perturbation and this ability to "return" to a previous stable-state after a perturbation is why Hopfield networks serve as models of memory.



Elman Network - 1990

Elman's approach to sequence representation

- In 1990, cognitive scientist Jeffrey Elman introduced the Elman Network, a successful example of a RNN trained with backpropagation
- Elman's work, "Finding Structure in Time," particularly addressed the representation of "time" or "sequences" in neural networks
- Elman proposed two approaches: an **explicit** approach and an **implicit** approach

Explicit approach

Input signal
of fixed size



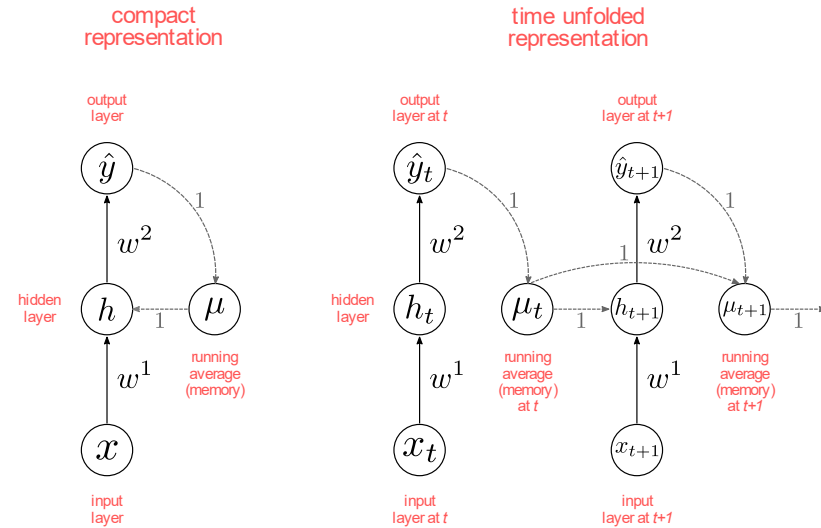
Main Drawbacks

- Fixed number of input units for each sequence
- Hard to distinguish relative temporal position from absolute temporal position

Elman Network - 1990

Jordan's memory unit

- Elman added a context/memory unit to his network to save past computations and incorporate them into future computations
- Elman's approach was based on the work of Michael I. Jordan on serial processing, who proposed the use of recurrent connections from the network output to hidden units through a **memory unit** to keep a running average of past outputs
- The memory unit in Jordan's network keeps a fixed weight equal to 1
- This allows the network to implicitly account for past history in each new computation.

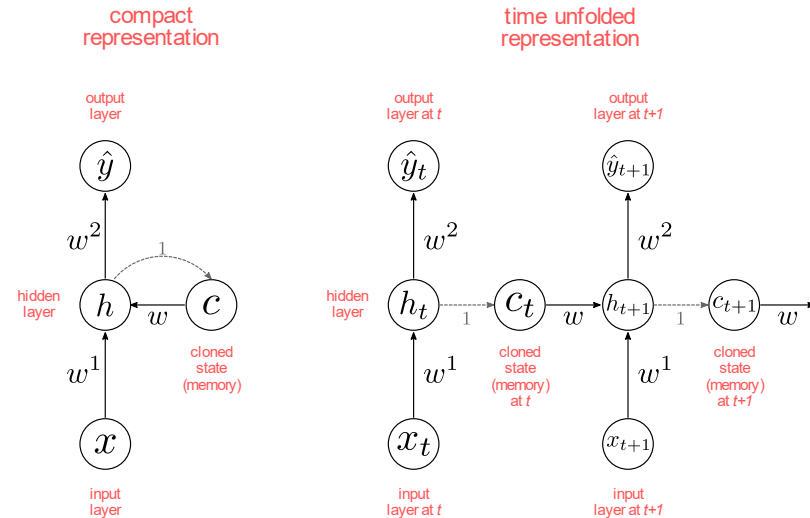


Jordan's Network

Elman Network - 1990

The implicit approach using backpropagation

- Elman introduced trainable parameters from memory units to hidden units
- The weights are trained using backpropagation
- Memory units **remember** the past state of hidden units
- Memory units **clone** the value at the previous time-step $t-1$
- Memory units learn useful representations (weights) for encoding temporal properties of sequential input
- Elman demonstrated the capabilities of his architecture through multiple experiments, showing it was able to solve problems with sequential structure, such as a temporal version of the **XOR** problem

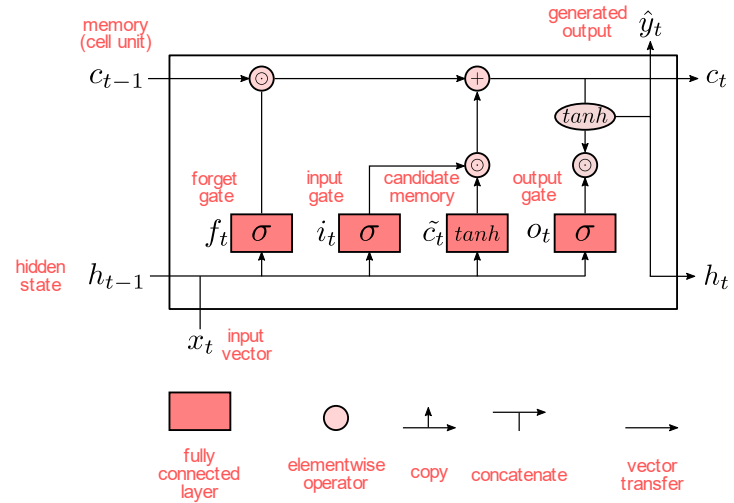


Elman's Network

LSTM Architecture - 1997

A modern way to handle sequential data

- Early RNNs faced challenges such as vanishing and exploding gradients and slow forward computation due to the sequential nature of the layers
- The architecture that moved the field forward was the **Long Short-Term Memory (LSTM)** Network, introduced by Sepp Hochreiter and Jurgen Schmidhuber in 1997
- LSTMs added units with both short-memory and long-memory capabilities, replacing the simple memory unit of Elman networks with a cell unit (long-term memory storage) and a hidden-state (memory controller)
- The cell unit and hidden-state are integrated as a circuit of logic gates controlling the flow of information at each time-step.



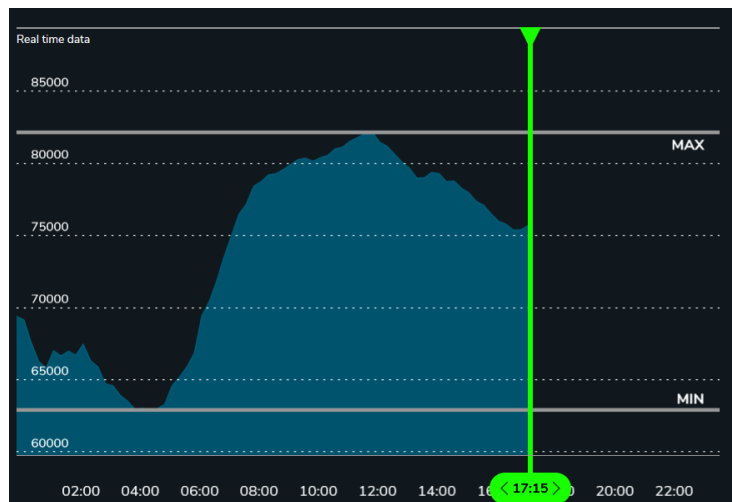
Sequential data

One is Many



A few data examples

Sequential data



Time series (electricity demand)

JACQUES LE FATALISTE ET SON MAITRE.

COMMENT s'étaient-ils rencontrés? Par hasard, comme tout le monde. Comment s'appelaient-ils? Que vous importe? D'où venaient-ils? Du lieu le plus prochain. Où allaient-ils? Est-ce que l'on sait où l'on va? Que disaient-ils? Le maître ne disait rien, et Jacques disait que son capitaine disait que tout ce qui nous arrive de bien et de mal ici bas était écrit là-haut.

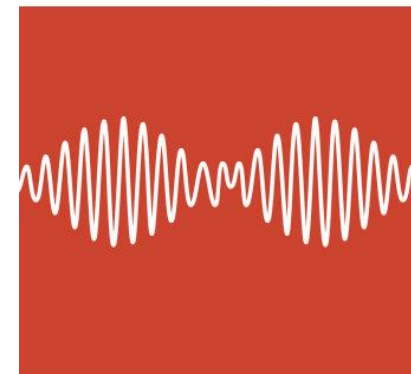
LE MAITRE.

C'est un grand mot que cela.

JACQUES.

Mon capitaine ajoutait que chaque
...

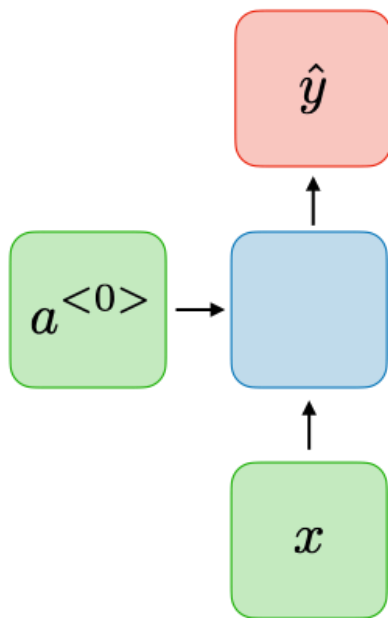
Text



Audio

Applications – One to One

Sequential data

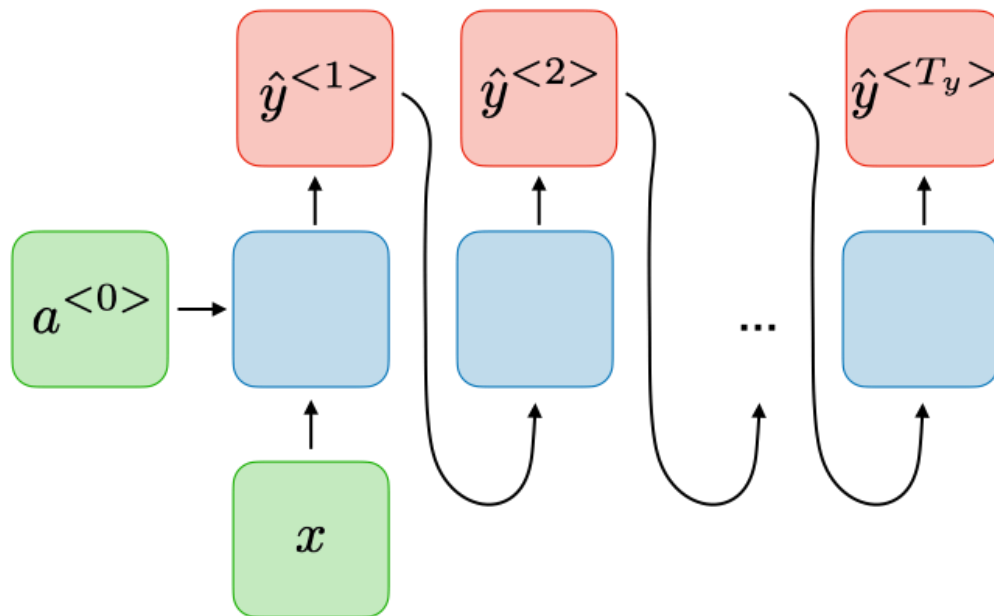


$$T_x = T_y = 1$$

e.g., Image Classification

Applications – One to Many

Sequential data

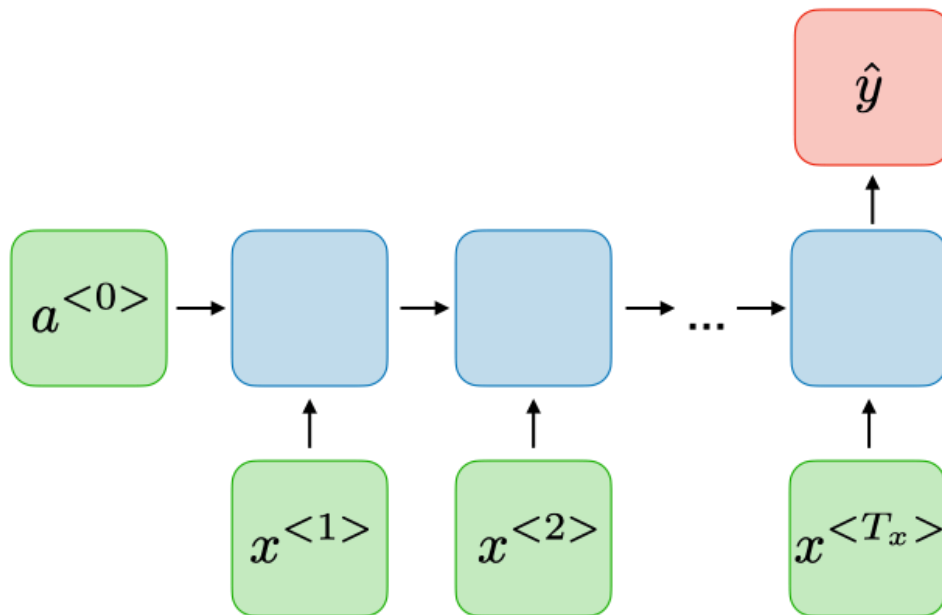


$$T_x = 1, T_y > 1$$

e.g., Image Captioning, Music generation

Applications – Many to One

Sequential data

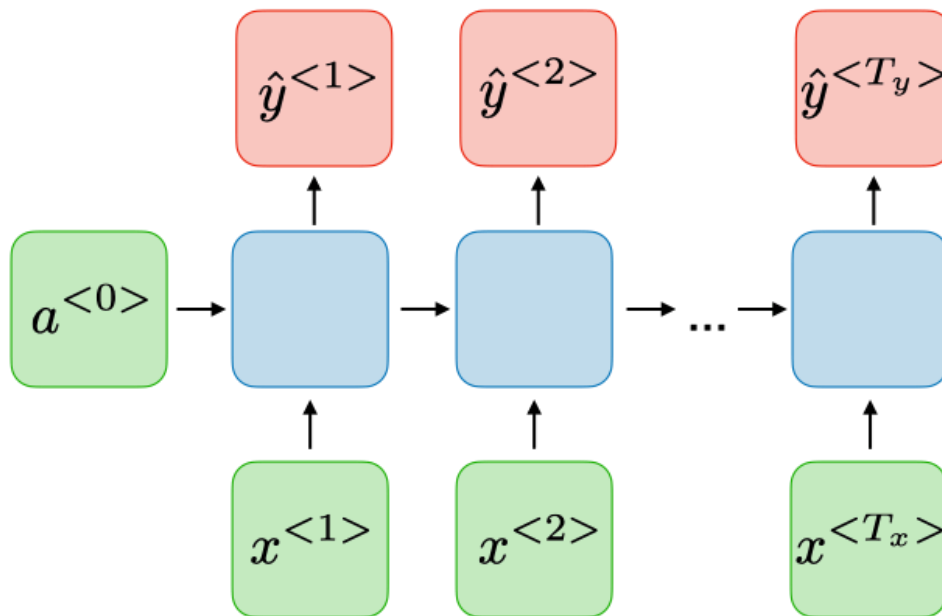


$$T_x > 1, T_y = 1$$

e.g., Sentiment Analysis

Applications – Many to Many (1/2)

Sequential data

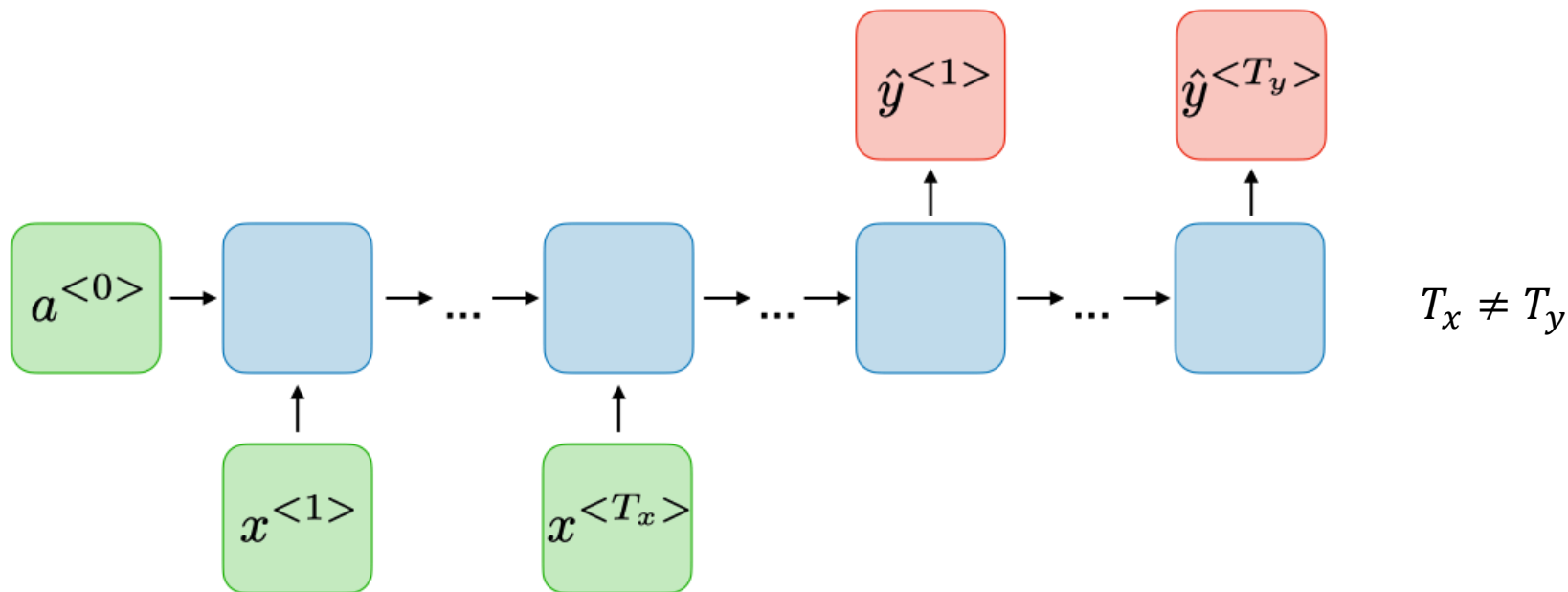


$$T_x = T_y$$

e.g., Named Entity Recognition

Applications – Many to Many (2/2)

Sequential data



e.g., Machine Translation

RNN structure

General principles

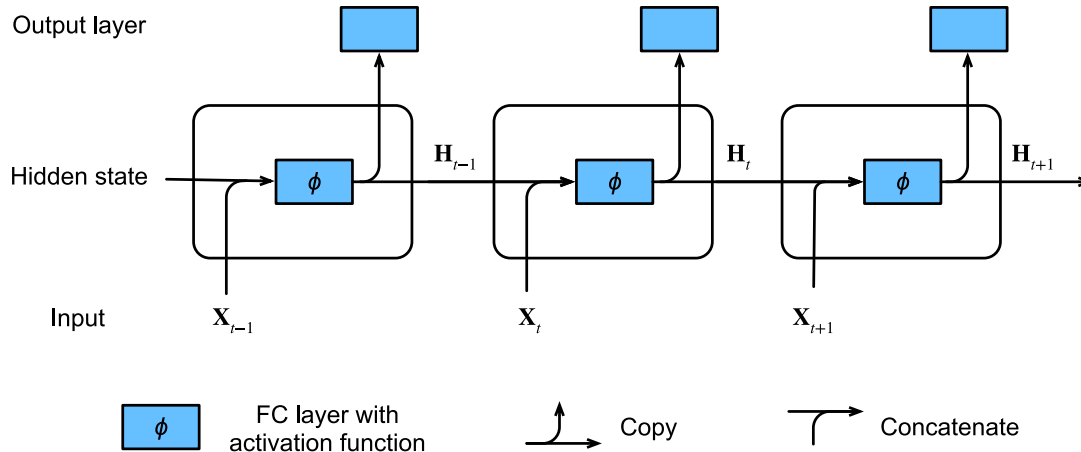


A Recurrent Neuron

The structure of recurrent cells

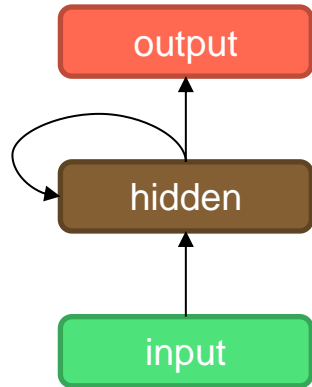
- Instead of modelling $X_t | X_{t-1} \dots X_1$, we approximate the conditional probability with a latent variable H_{t-1}

$$P(X_t | X_{t-1} \dots X_1) \approx P(X_t | H_{t-1})$$



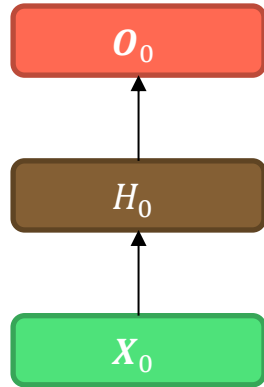
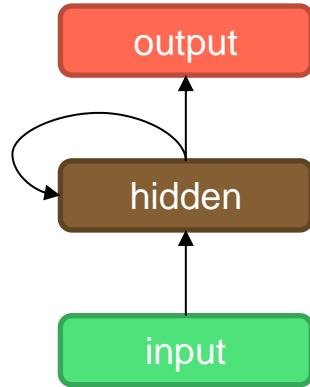
Unrolling the Recurrent Neuron

General Principles



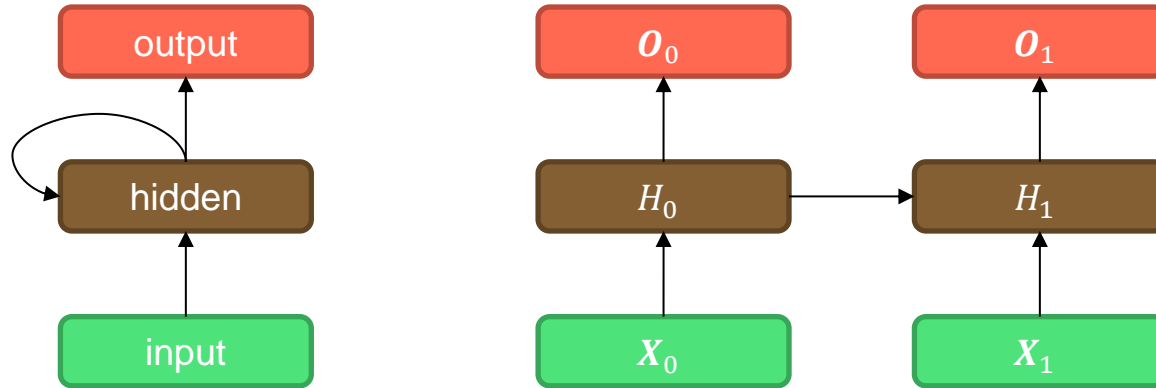
Unrolling the Recurrent Neuron

General Principles



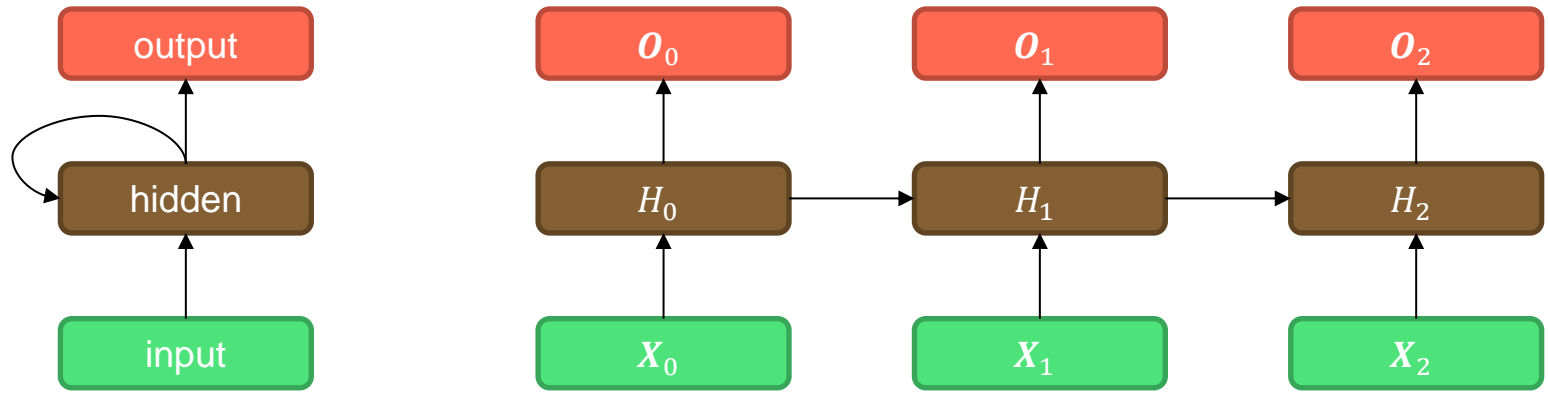
Unrolling the Recurrent Neuron

General Principles

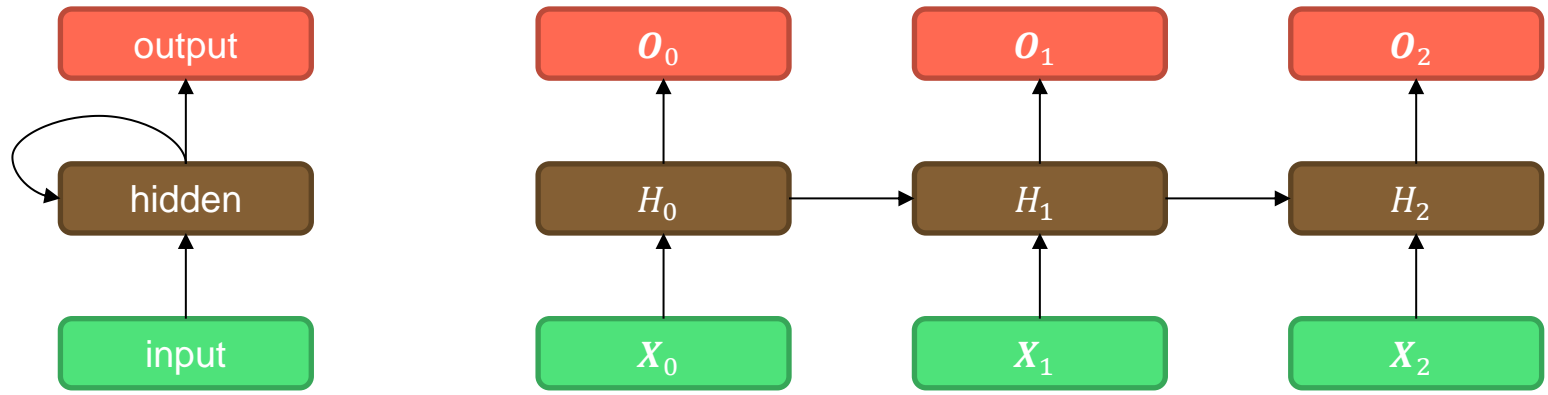


Unrolling the Recurrent Neuron

General Principles



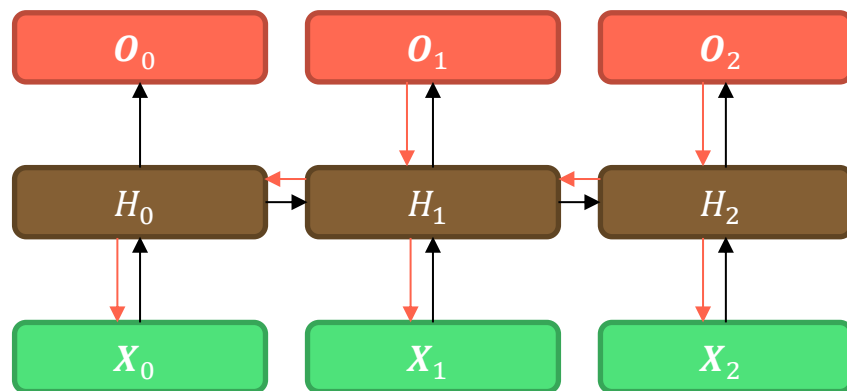
Unrolling the Recurrent Neuron



Unrolled on three timesteps

Backpropagation through time

- Backpropagation in RNNs is known as backpropagation through time (BPTT)
- BPTT requires expanding the computational graph of an RNN one time step at a time
- The unrolled RNN is equivalent to a feedforward neural network with the same parameters repeated at each time step
- Long sequences can cause computational and optimization problems
- Input from the first step passes through many matrix products before arriving at the output which can cause numerical instability



The intuition behind BPTT

- For timestep t we can define

$$\begin{aligned} \mathbf{H}_t &= f(\mathbf{X}_t, \mathbf{H}_{t-1}, \mathbf{w}_h) \\ \mathbf{O}_t &= g(\mathbf{H}_t, \mathbf{w}_o) \end{aligned}$$

- Generally we have the following chain of values

$$\{\dots, (\mathbf{X}_{t-1}, \mathbf{H}_{t-1}, \mathbf{O}_{t-1}), (\mathbf{X}_t, \mathbf{H}_t, \mathbf{O}_t), \dots\}$$

- The objective function can be written as follows

$$L(\mathbf{X}_1, \dots, \mathbf{X}_T, \mathbf{Y}_1, \dots, \mathbf{Y}_T, \mathbf{w}_h, \mathbf{w}_o) = \frac{1}{T} \sum_{t=1}^T l(\mathbf{Y}_t, \mathbf{O}_t)$$

The intuition behind BPTT

- Let's compute the backpropagation for \mathbf{w}_h

$$\frac{\partial L}{\partial \mathbf{w}_h} = \sum_{t=1}^T \frac{\partial l(\mathbf{Y}_t, \mathbf{O}_t)}{\partial \mathbf{w}_h}$$

- And we obtain the following via the chain rule

$$\frac{\partial L}{\partial \mathbf{w}_h} = \sum_{t=1}^T \frac{\partial l(\mathbf{Y}_t, \mathbf{O}_t)}{\partial \mathbf{O}_t} \frac{\partial g(\mathbf{H}_t, \mathbf{w}_o)}{\partial \mathbf{H}_t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{w}_h}$$

- The two first terms inside the sum above can be calculated directly
- However, the term $\frac{\partial \mathbf{H}_t}{\partial \mathbf{w}_h}$ requires more work

The intuition behind BPTT

- \mathbf{H}_t depends both on \mathbf{H}_{t-1} and \mathbf{w}_h where the computation of \mathbf{H}_{t-1} also depends on \mathbf{w}_h

$$\frac{\partial \mathbf{H}_t}{\partial \mathbf{w}_h} = \frac{\partial f(\mathbf{X}_t, \mathbf{H}_{t-1}, \mathbf{w}_h)}{\partial \mathbf{w}_h} + \frac{\partial f(\mathbf{X}_t, \mathbf{H}_{t-1}, \mathbf{w}_h)}{\partial \mathbf{H}_{t-1}} \frac{\partial \mathbf{H}_{t-1}}{\partial \mathbf{w}_h}$$

- By recurrence we can show that the above equation can lead to

$$\frac{\partial \mathbf{H}_t}{\partial \mathbf{w}_h} = \frac{\partial f(\mathbf{X}_t, \mathbf{H}_{t-1}, \mathbf{w}_h)}{\partial \mathbf{w}_h} + \sum_{i=1}^{t-1} \prod_{j=i+1}^t \frac{\partial f(\mathbf{X}_j, \mathbf{H}_{j-1}, \mathbf{w}_h)}{\partial \mathbf{H}_{j-1}} \frac{\partial f(\mathbf{X}_i, \mathbf{H}_{i-1}, \mathbf{w}_h)}{\partial \mathbf{w}_h}$$

- Therefore, the chain rule can be used for recursive computation of $\frac{\partial \mathbf{H}_t}{\partial \mathbf{w}_h}$
- However, if t is too large, $\frac{\partial \mathbf{H}_t}{\partial \mathbf{w}_h}$ can become difficult to compute

Computation of $\frac{\partial H_t}{\partial \mathbf{w}_h}$

Full computation

- The first idea is to compute the full sum defined in the previous slide
- However, this is very slow, and gradients can explode
- This is because subtle changes in the initial conditions can have a large impact on the outcome
- It does not lead to robust estimators that generalise well, so it is almost never used in practice.

Truncating timesteps

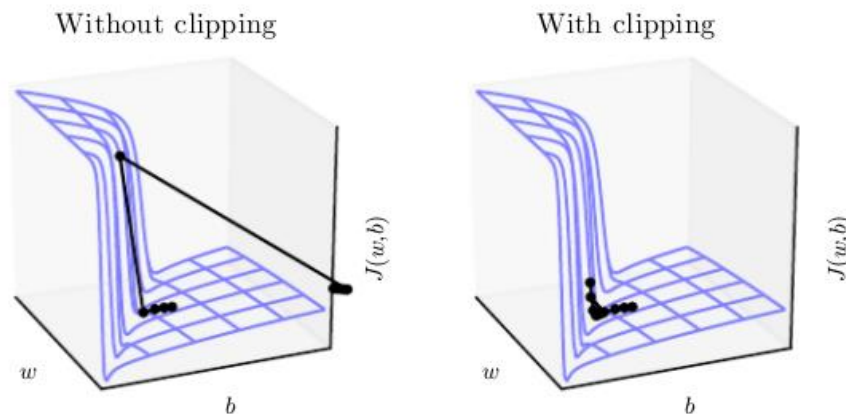
- Instead of computing the full sum, we can truncate it after τ timesteps
- This leads to an approximation of the true gradient by terminating the sum at a certain point
- This is commonly referred to as truncated backpropagation through time (Jaeger, 2002)
- Truncation leads to the model focusing primarily on short-term rather than long-term influence
- It biases the estimate towards simpler and more stable models, and it works well in practice
- Other variants include Randomised Truncation (Tallec and Ollivier, 2017)

Gradient clipping

Avoiding exploding gradients

- The standard strategy to solve exploding gradient issues is gradient norm clipping
- The idea is to rescale the norm of the gradient to a fixed threshold δ when it is above it

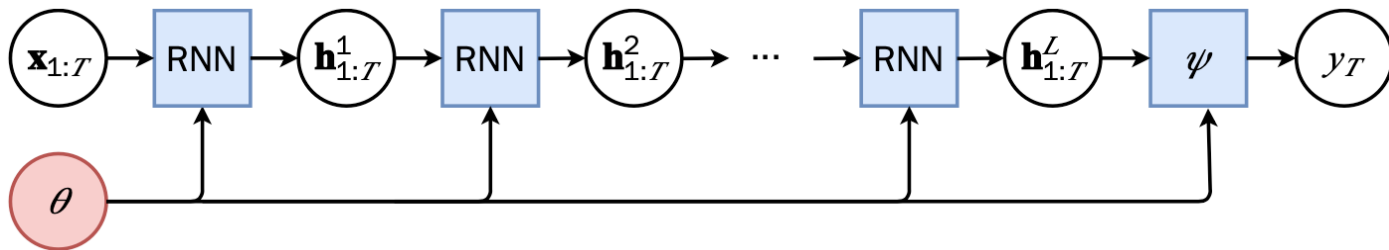
$$\tilde{\nabla} f = \frac{\nabla f}{\|\nabla f\|} \min(\|\nabla f\|, \delta)$$



Stacking recurrent cells

Multiple hidden layers

- RNNs can be viewed as layers producing sequences $\mathbf{h}_{1:T}^l$ acting like hidden layers of activations
- We can stack these layers to form a deep RNN



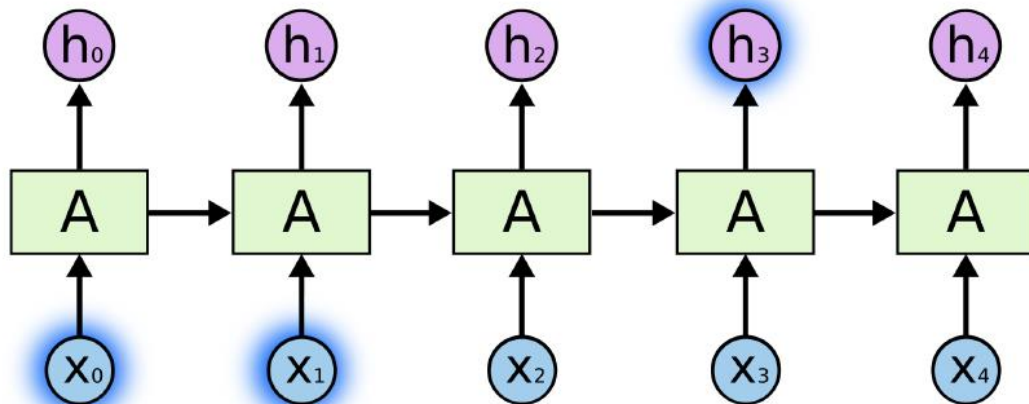
Modern RNN

LSTM, GRU



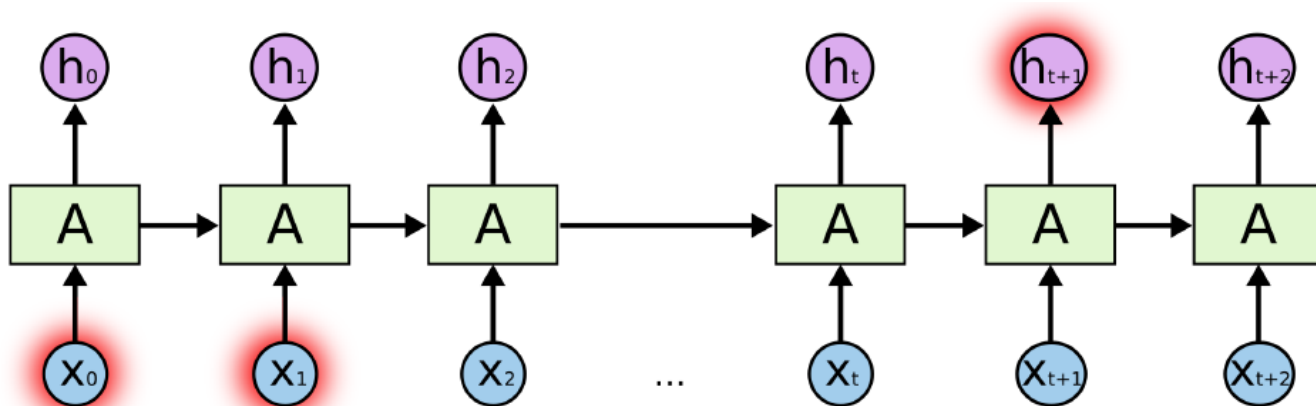
Long-term dependencies

- When sequences are relatively short the matrix products do not vanish or explode for gradient computations
- The network will mainly learn short-term dependencies with short input sequences



Long-term dependencies

- To account for long-term memory, input sequences need to be larger
- However, this leads to vanishing and/or exploding gradients
- We have seen that we can use gradient clipping to avoid exploding gradients, but the vanishing gradient problem remains
- We need a new type of recurrent cells to retain long-term dependencies



Long short-term Memory (LSTM)

Modern RNN

Gated Hidden State

- LSTMs differ from traditional RNNs in that LSTMs support gating of the hidden state
- The gating mechanism is used for updating and resetting the hidden state accordingly
- Precisely, a memory cell has an internal state and is equipped with multiplicative gates
- The gates determine:

Whether a given input should impact the internal state - **input gate**

Whether the internal state should be put to 0 - **forget gate**

Whether the internal state should impact the cell's output - **output gate**

Long short-term Memory (LSTM)

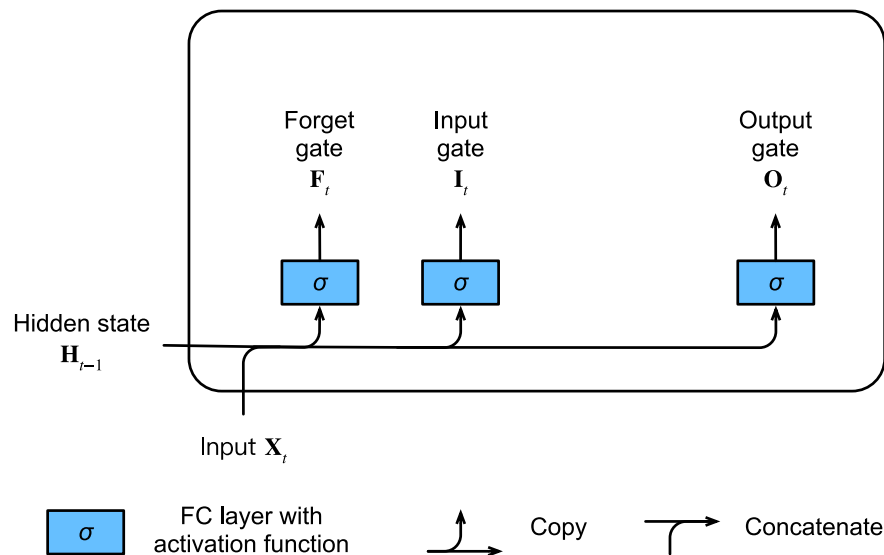
Modern RNN

Data feeding into LSTM gates

- X_t Input at current time step
- H_{t-1} Hidden state of previous time step

Activation functions

- Sigmoid activation \Rightarrow gate values in $[0,1]$



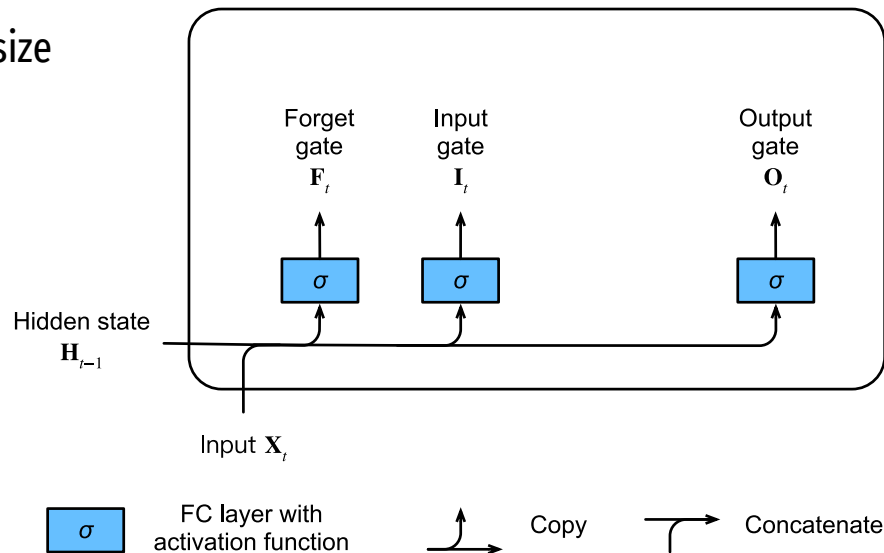
Long short-term Memory (LSTM)

Mathematical formulation

- Suppose that there are h hidden units, a batch size n , and d inputs
- $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ and $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$
- We get the following gate values

$$\begin{aligned} \mathbf{I}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i) \\ \mathbf{F}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f) \\ \mathbf{O}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o) \end{aligned}$$

- $\mathbf{W}_x \in \mathbb{R}^{d \times h}$
- $\mathbf{W}_h \in \mathbb{R}^{h \times h}$
- $\mathbf{I}_t, \mathbf{F}_t, \mathbf{O}_t \in \mathbb{R}^{n \times h}$



Long short-term Memory (LSTM)

Modern RNN

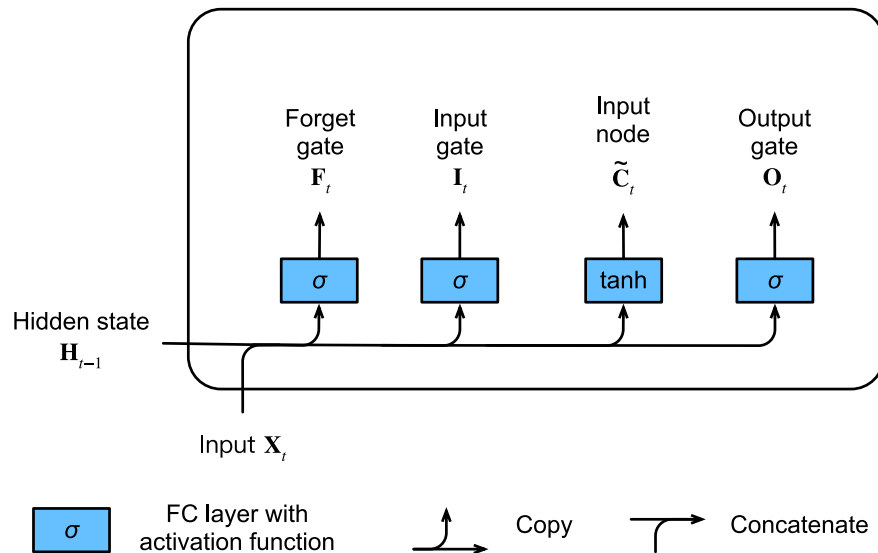
Input node

- $\tilde{\mathbf{C}}_t \in \mathbb{R}^{n \times h}$
- Its computation is equivalent to the one of three gates but with a tanh activation

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c)$$

Link with the input gate

- The value of the input node interacts with the input gate to decide what should be added to the current **internal state**



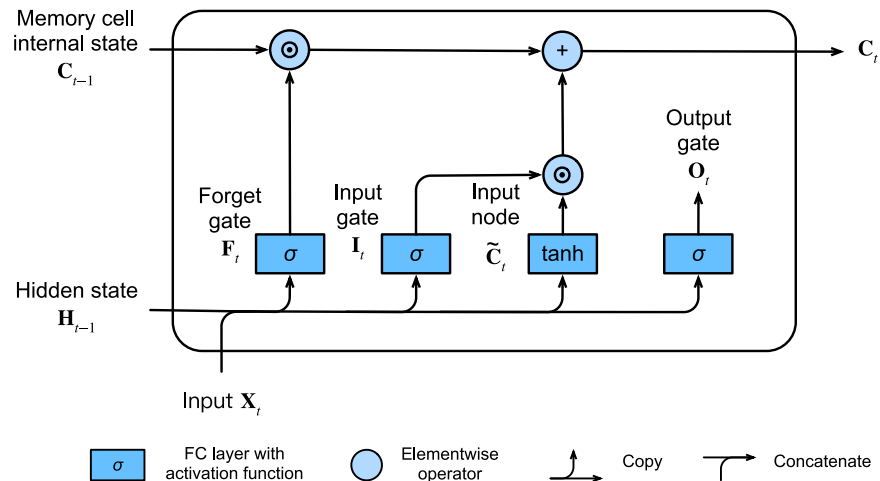
Long short-term Memory (LSTM)

Memory cell state

- Let us define the memory cell state $\mathbf{C}_t \in \mathbb{R}^{n \times h}$ for timestep t

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \oplus \tilde{\mathbf{C}}_t$$

- \mathbf{F}_t addresses how much of the old cell internal state \mathbf{C}_{t-1} we retain
- \mathbf{I}_t governs how much we take new data into account via $\tilde{\mathbf{C}}_t$
- If $\mathbf{F}_t = \mathbf{1}$ and $\mathbf{I}_t = \mathbf{0}$ the memory cell remains constant ($\mathbf{C}_t = \mathbf{C}_{t-1}$)



Long short-term Memory (LSTM)

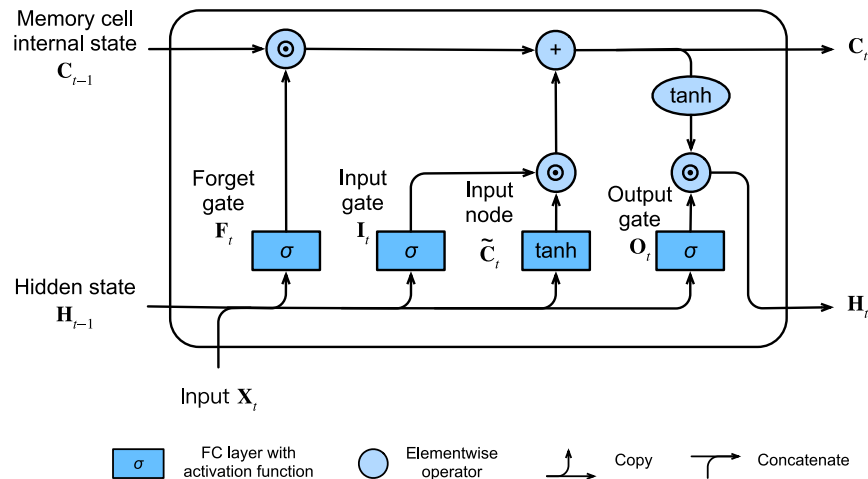
Modern RNN

Output gate and internal state

- Finally, we have to define the output H_t of the memory cell using both O_t and C_t

$$H_t = O_t \odot \tanh(C_t)$$

- When O_t is close to 0, current memory does not impact the subsequent layer of the network
- When O_t is close to 1, current memory adds information to the next layer

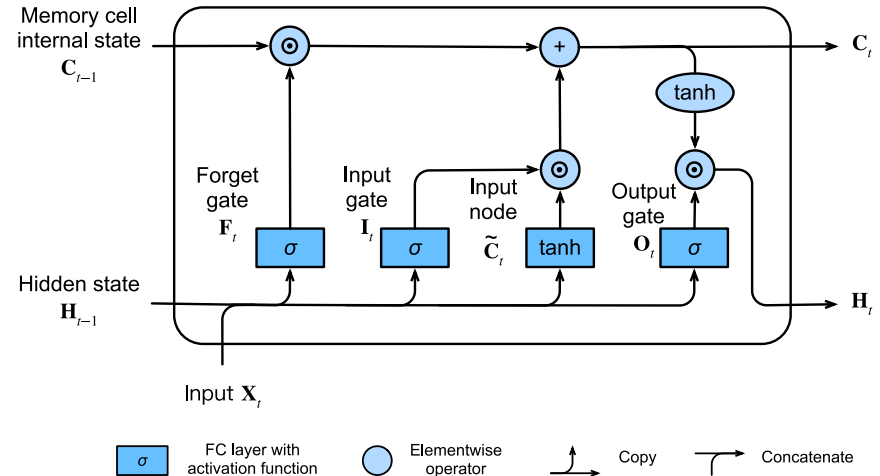


Long short-term Memory (LSTM)

Modern RNN

Overall

- LSTMs long-term memory capabilities make them good at capturing **long-term dependencies**
- The memory cell effectively counteracts the vanishing gradient problem at preserving information as long as the forget gate does not **erase** past information (Graves, 2012)

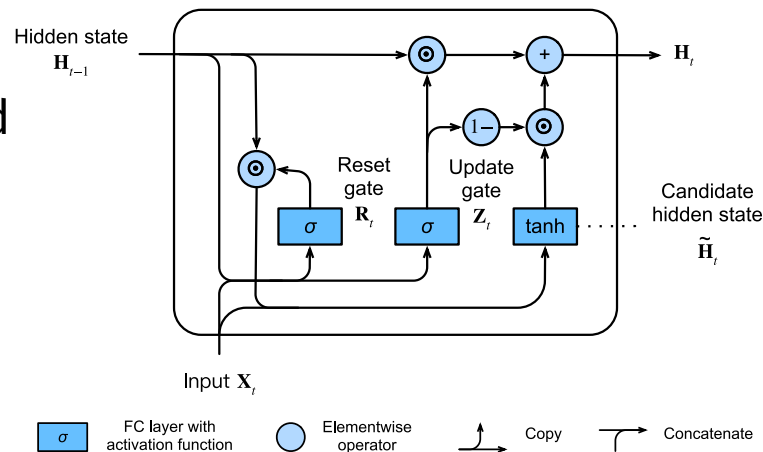


Gated Recurrent Units (GRU)

Modern RNN

Origins

- RNNs and more particularly LSTM architectures became popular in the 2010s
- Researchers began experimenting with simplified architectures to speed up computation while retaining key features
- Gated Recurrent Units (GRU) were introduced as a streamlined version of the LSTM memory cell
- GRU often achieves comparable performance to LSTM but is faster to compute



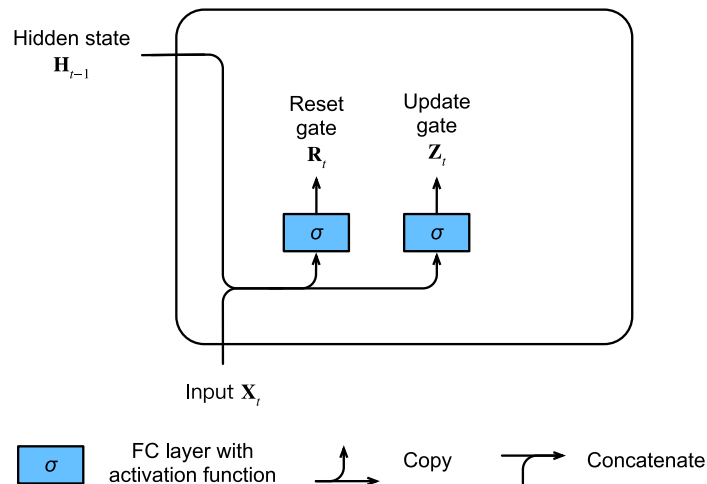
Gated Recurrent Units (GRU)

A more compact memory cell

- LSTM's three gates are replaced by two in GRU
- These gates are also given sigmoid activations, resulting in values in the interval $[0,1]$
- The **reset gate** controls how much of the previous state to remember
- The **update gate** controls how much of the new state is a copy of the old state

$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \mathbf{b}_r)$$

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \mathbf{b}_z)$$



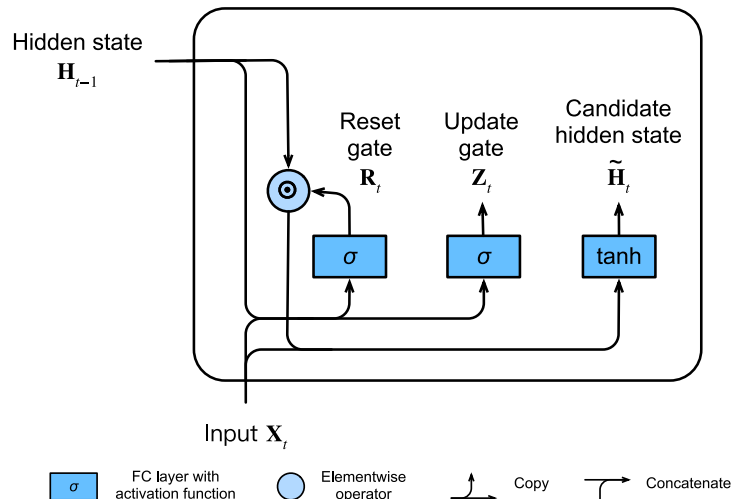
Gated Recurrent Units (GRU)

Modern RNN

Candidate Hidden State

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h)$$

- The influence of previous states can be reduced to the elementwise multiplication of R_t and H_{t-1}
- When the R_t is close to 1, it behaves like a traditional RNN
- When R_t entries are close to 0, the candidate hidden state effectively resets any pre-existing hidden state to defaults



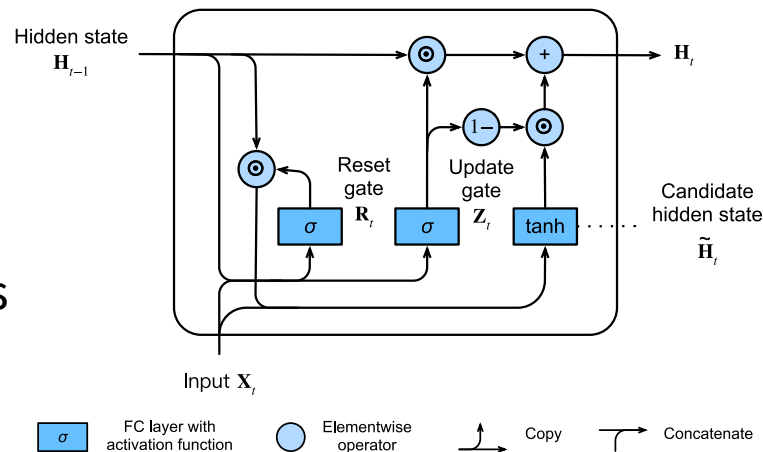
Gated Recurrent Units (GRU)

Hidden State

- Finally, we incorporate the effect of the update gate Z_t on the hidden state H_t

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$$

- Intuitively, Z_t balances H_t between the previous state H_{t-1} and the candidate state \tilde{H}_t
- If Z_t is close to 1 we retain the previous state
- If Z_t is close to 0 the new latent state H_t approaches the candidate latent state \tilde{H}_t

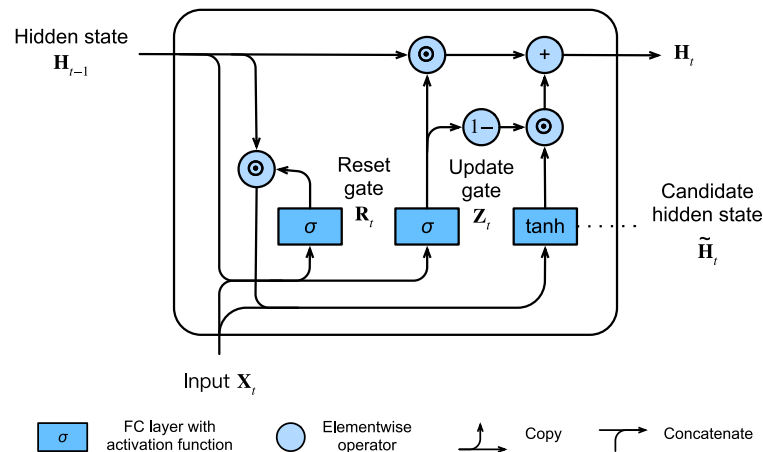


Gated Recurrent Units (GRU)

Modern RNN

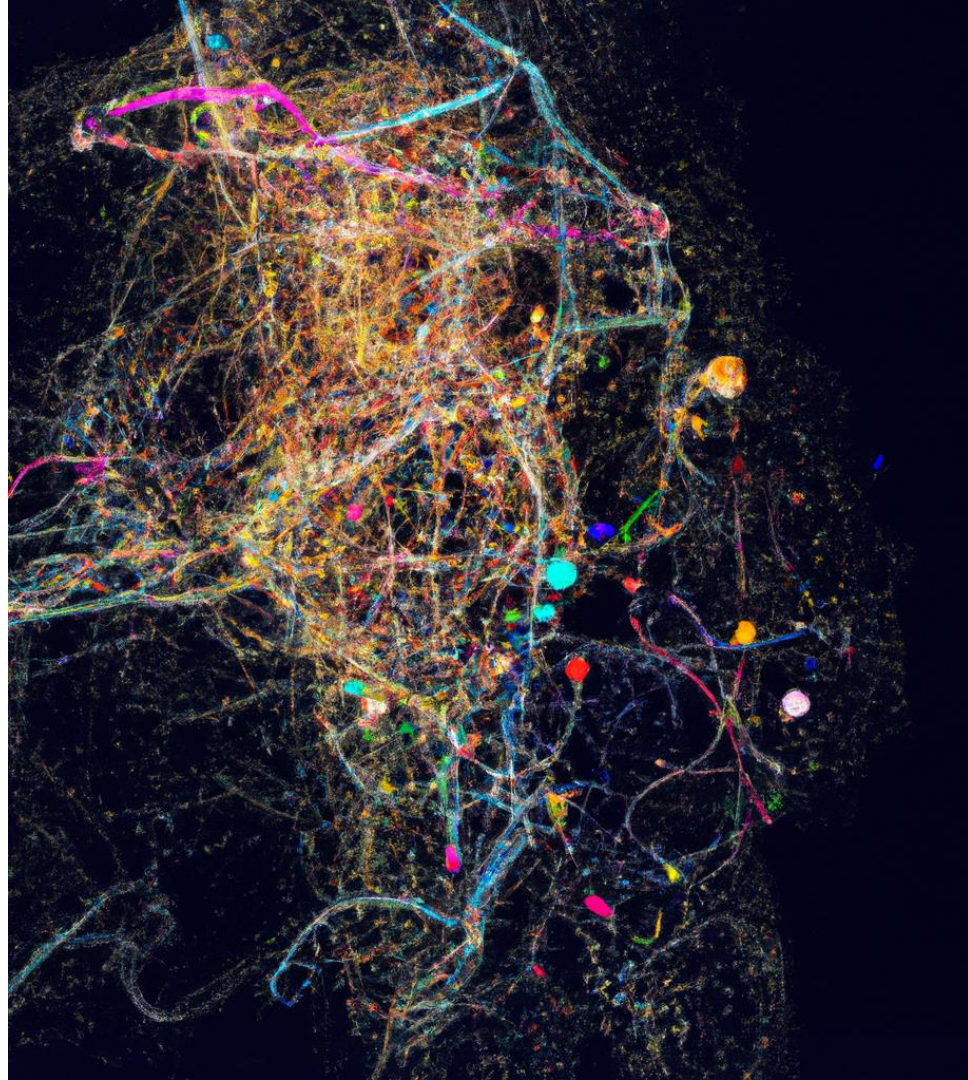
Overall

- **Reset gates** help capture **short-term** dependencies in sequences.
- **Update gates** help capture **long-term** dependencies in sequences.



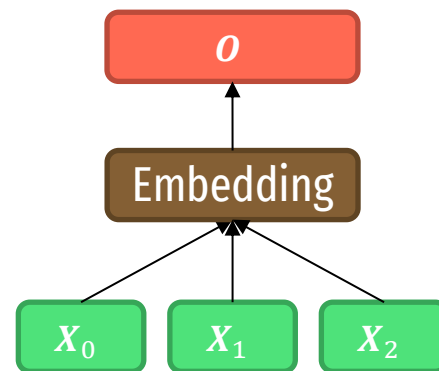
Embeddings

Language models



The principle behind embeddings

- Embeddings are a method for **representing data** (e.g., words, sentences) in a high-dimensional continuous space.
- Embedding vectors capture the semantic and syntactic relationships between the data
- The goal of embeddings is to map data into a **lower-dimensional space** while preserving inner relationships
- This allows for more efficient processing and analysis
- Embeddings are commonly used in **Natural Language Processing (NLP)** and **Computer Vision (CV)** tasks, as well as in recommendation systems, information retrieval, and other applications



Word2Vec

- Word2Vec is a method generating numerical representations of words ⇒ **word embeddings**
- These embeddings can capture **semantic** and **syntactic** relationships between words
- The model is trained on a large corpus using a neural network with a single hidden layer
- The **network input** is a one-hot encoded representation of a word, and the goal of training is to learn weights for the hidden layer which we call **embedding**
- There are two main variants of the model
 - **Continuous Bag of Words (CBOW)** - predicts current word from the context
 - **Skip-Gram** predicts context from the current word
- The model learns the embeddings by trying to predict the surrounding words given a specific word, or predict the word given the context word(s)

Word2Vec

Language models

- Word2Vec is a method generating numerical representations of words ⇒ **word embeddings**
- These embeddings can capture **semantic** and **syntactic** relationships between words
- The model is trained on a large corpus using a neural network with a single hidden layer
- The **network input** is a one-hot encoded representation of a word, and the goal of training is to learn weights for the hidden layer which we call **embedding**
- There are two main variants of the model

Continuous Bag of Words (CBOW) - predicts current word from the context

Skip-Gram predicts context from the current word

- The model learns the embeddings by trying to predict the surrounding words given a specific word, or predict the word given the context word(s)

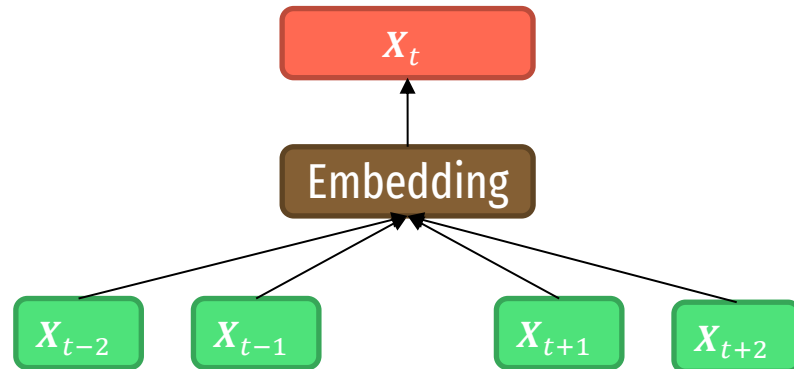
Word2Vec

Language models

Continuous Bag of Words (CBOW)

- Akin to text classification problems \Rightarrow **but** large number of classes $|V|$ (vocabulary considered)
- The softmax output requires to sum over $|V|$ which leads to intractable calculations
- The trick is to use negative sampling to shrink the denominator of the cost function
- We sample just a few 'negative' words from the **unigram** distribution

[...] the best football player of all time is [...]



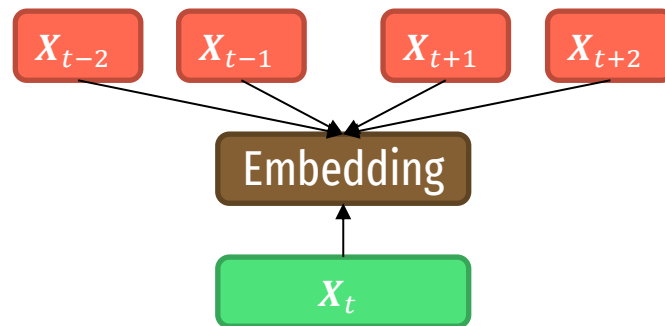
Word2Vec

Language models

Skip-Gram

- Given the central word, predict occurrence of other words in its context.
- Widely used in practice, particularly for large corpuses
- **Negative Sampling** is again used as a cheaper alternative to a full softmax

[...] the best football player of all time is [...]



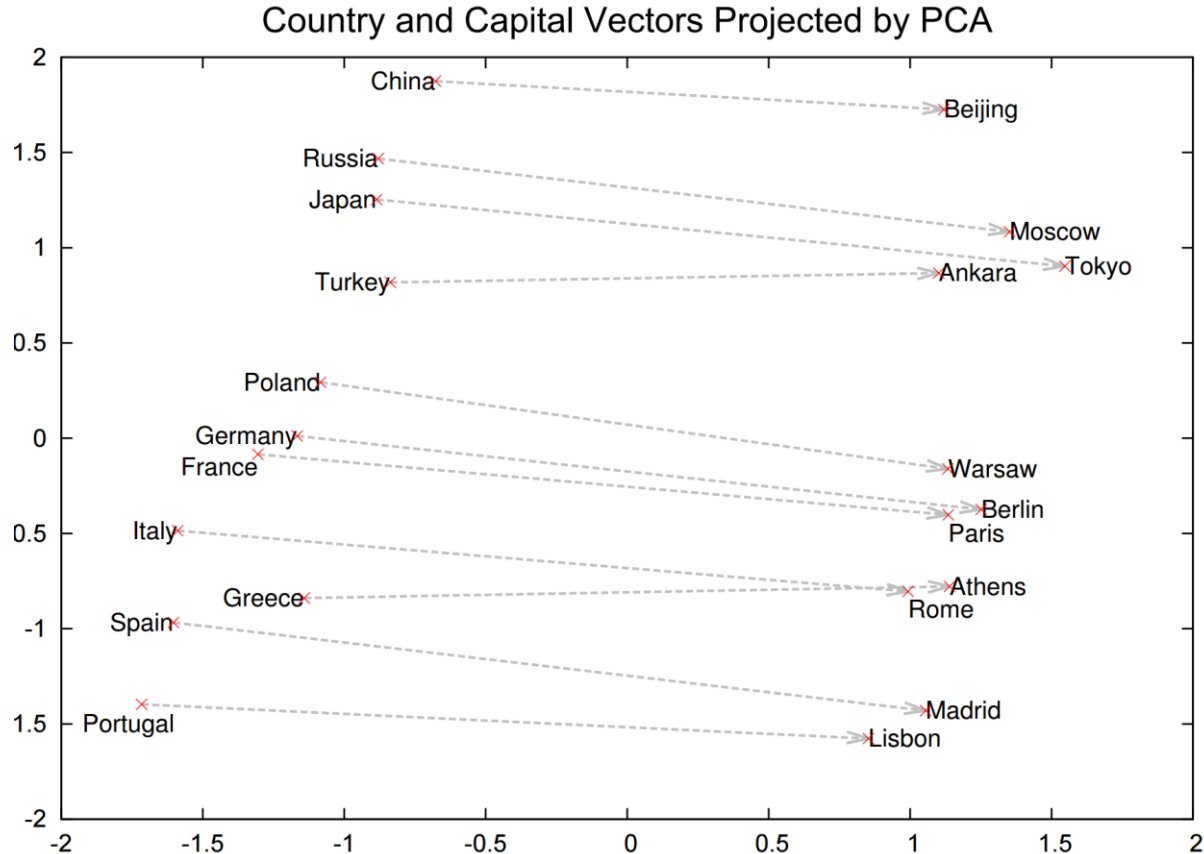
Similarity

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Composition

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna Check crown Polish zolty CTK	Hanoi Ho Chi Minh City Viet Nam Vietnamese	airline Lufthansa carrier Lufthansa flag carrier Lufthansa Lufthansa	Moscow Volga River upriver Russia	Juliette Binoche Vanessa Paradis Charlotte Gainsbourg Cecile De

Word2Vec



Human biases in word embeddings

C is to B as A is to X

Analogy	Reported	Index	1st answer	2nd answer
Mikolov et al. (2013a)				
man king woman	queen	2	king	queen
Paris France Tokyo	Japan	1	Japan	Tokyo
brother sister grandson	granddaughter	1	granddaughter	niece
big bigger cold	colder	2	cold	colder
Einstein scientist Picasso	painter	1	painter	scientist

[Fair Is Better than Sensational: Man Is to Doctor as Woman Is to Doctor](<https://aclanthology.org/2020.cl-2.7>) (Nissim et al., CL 2020)

Human biases in word embeddings

C is to B as A is to X

Analogy	Reported	Index	1st answer	2nd answer
Bolukbasi et al. (2016)				
man computer_programmer woman	homemaker	2	computer_programmer	homemaker
he doctor she	nurse	2	doctor	nurse
she interior_designer he	architect	2	interior_designer	architect
she feminism he	conservatism	4	feminism	liberalism
she lovely he	brilliant	10	lovely	magnificent
she sewing he	carpentry	4	sewing	woodworking

[Fair Is Better than Sensational: Man Is to Doctor as Woman Is to Doctor](<https://aclanthology.org/2020.cl-2.7>) (Nissim et al., CL 2020)

Human biases in word embeddings

Language models

C is to B as A is to X

Analogy	Reported	Index	1st answer	2nd answer
Manzini et al. (2019b)				
black criminal caucasian	lawful	13	legal	statutory
caucasian lawful black	criminal	2	lawful	criminal
caucasian hillbilly asian	yuppie	3	hillbilly	hippy
asian yuppie caucasian	hillbilly	2	yuppie	hillbilly
asian engineer black	killer	39	operator	jockey
black killer asian	engineer	7	killer	impostor
christian conservative jew	liberal	4	centrist	democrat
jew liberal christian	conservative	2	liberal	conservative
muslim terrorist jew	journalist	4	hacker	protestor
jew journalist muslim	terrorist	2	purportedly	terrorist
christian conservative muslim	regressive	53	moderate	conservative
muslim regressive christian	conservative	13	regressive	progressive

[Fair Is Better than Sensational: Man Is to Doctor as Woman Is to Doctor](<https://aclanthology.org/2020.cl-2.7>) (Nissim et al., CL 2020)