Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 8/10: Natural Language Processing

DR VARUN OJHA

Department of Computer Science

University of Reading

Natural Language*

Example of Machine Translation

人工知能が大好き
JAPANESE

I love artificial intelligence
ENGLISH

Я люблю искусственный интеллект
RUSSIAN

我爱人工智能
CHINESE

Jag älskar konstgjord intelligens
SWEDISH

Miluji umělou inteligenci
CZECH

मुझे कृत्रिम बुद्धिमत्ता पसंद है
HINDI

나는 인공 지능을 좋아한다
KOREAN

Szeretem a mesterséges intelligenciát
HUNGARIAN

ฉันรักปัญญาประดิษฐ์
THAI

Ich liebe künstliche Intelligenz
GERMAN

ମୁଁ କୃତ୍ରିମ ବୁଦ୍ଧିମତାକୁ ଭଲ ପାଏ |
ORIYA

Amo la inteligencia artificial
SPANISH

من عاشق هوش مصنوعی هستم
PERSIAN

אני אוהבת בינה מלאכותית
HEBREW

நான் செயற்கை நுண்ணறிவை விரும்புகிறேன்
TAMIL

أحب الذكاء الاصطناعي
ARABIC

എനിക്ക് കൃത്രിമബുദ്ധി ഇഷ്ടമാണ്
MALAYALAM

*Please blame Google for any offensive translation of the sentence “I love artificial intelligence” I can only guarantee the correctness of “Hindi” translation.

Learning objectives

By the end of this week, you will be able to:

- Learn basic tasks of natural language processing
- Learn basic concepts of TEXT DATA classification
- Learn basic text data processing techniques
- Understand basic functioning of recurrent neural
- Workout an example problem of recurrent neural networks

Content of this week

- Part 1: Introduction
  - Tasks of natural language processing (NLP)
  - K-Nearest neighbour classification
- Part 2: Text Data Pre-processing
  - Word to Vector
  - Bag of word
  - Understanding Sequential Data
- Part 3: Recurrent Neural Networks
  - Basic Concepts
  - Long-Short Memory Networks
- Part 4: Practical Exercise (RNN)
- Quiz

Artificial Intelligence

CS3AI18/ CSMAI19

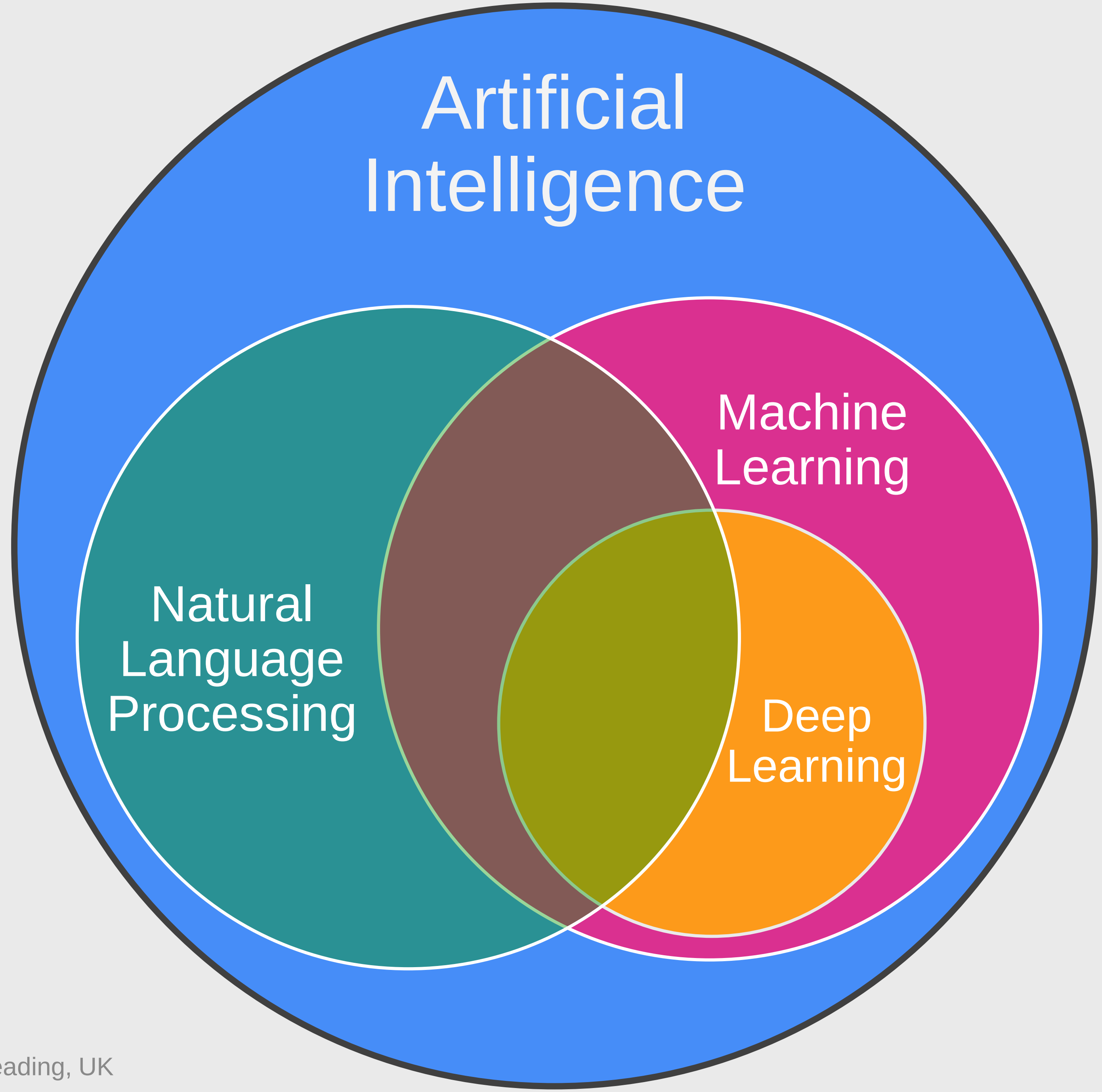
Lecture - 8/10: Natural Language Processing

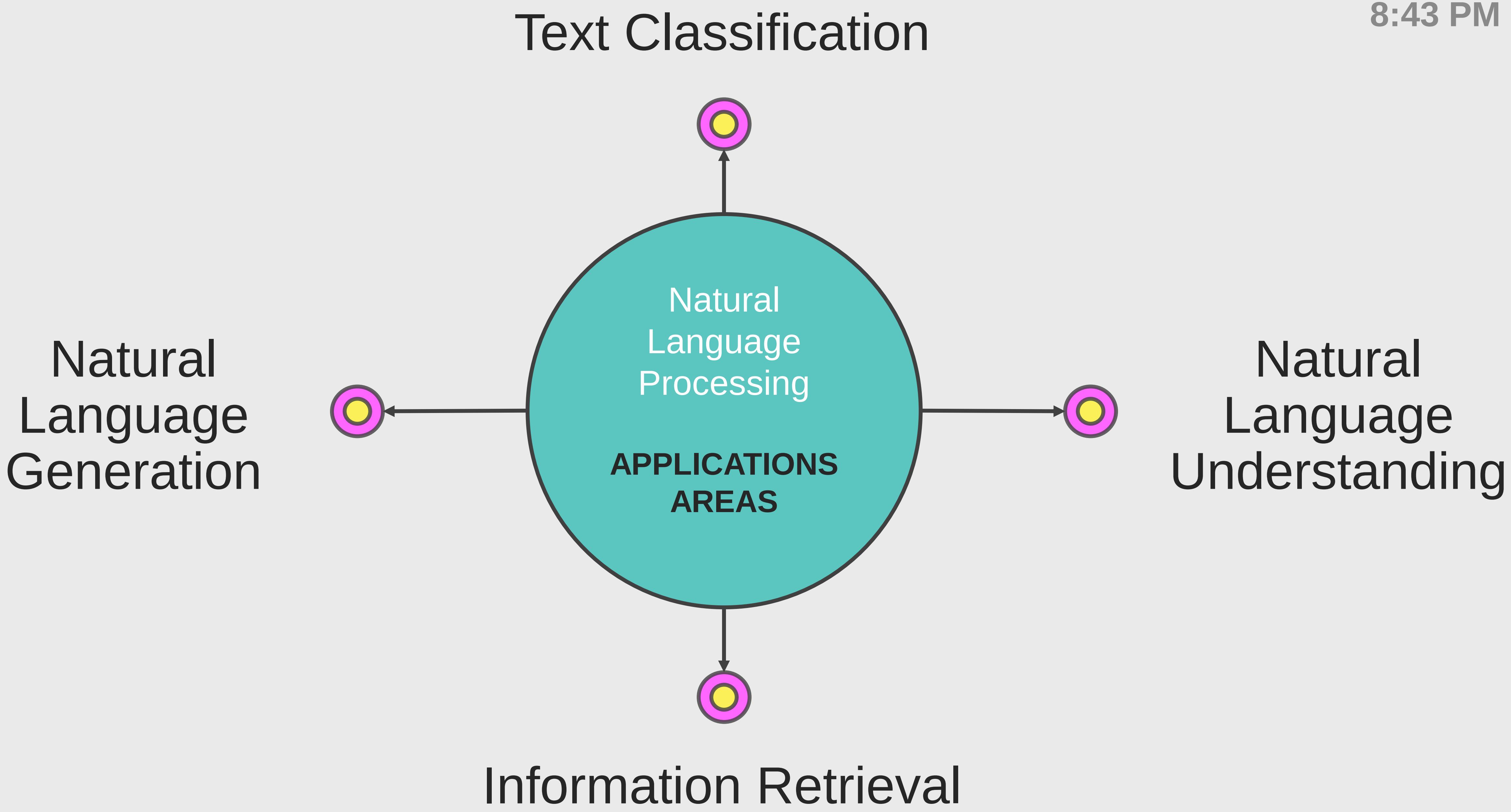
Part 1
Introductions

DR VARUN OJHA

Department of Computer Science

University of Reading


Artificial
Intelligence
Machine
Learning
Natural
Language
Processing
Deep
Learning


Text Classification
Natural Language Processing
APPLICATIONS AREAS
Natural Language Generation
Natural Language Understanding
Information Retrieval

Unstructured Data

Suppose this text is a message in your mailbox

History of natural language processing : The history of natural language processing (NLP) generally started in the 1950s, although work can be found from earlier periods. In 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence[clarification needed].

The Georgetown experiment in 1954 involved fully automatic translation of more than sixty Russian sentences into English. The authors claimed that within three or five years, machine translation would be a solved problem.[2] However, real progress was much slower, and after the ALPAC report in 1966, which found that ten-year-long research had failed to fulfill the expectations, funding for machine translation was dramatically reduced. Little further research in machine translation was conducted until the late 1980s when the first statistical machine translation systems were developed.

Some notably successful natural language processing systems developed in the 1960s were SHRDLU, a natural language system working in restricted "blocks worlds" with restricted vocabularies, and ELIZA, a simulation of a Rogerian psychotherapist, written by Joseph Weizenbaum between 1964 and 1966. Using almost no information about human thought or emotion, ELIZA sometimes provided a startlingly human-like interaction. When the "patient" exceeded the very small knowledge base, ELIZA might provide a generic response, for example, responding to "My head hurts" with "Why do you say your head hurts?".Text Source: https://en.wikipedia.org/wiki/Natural_language_processing

Dataset

No apparent structure in the data

Is it possible to classify a SPAM or Genuine message

Is it possible to identify a positive or negative sentence?

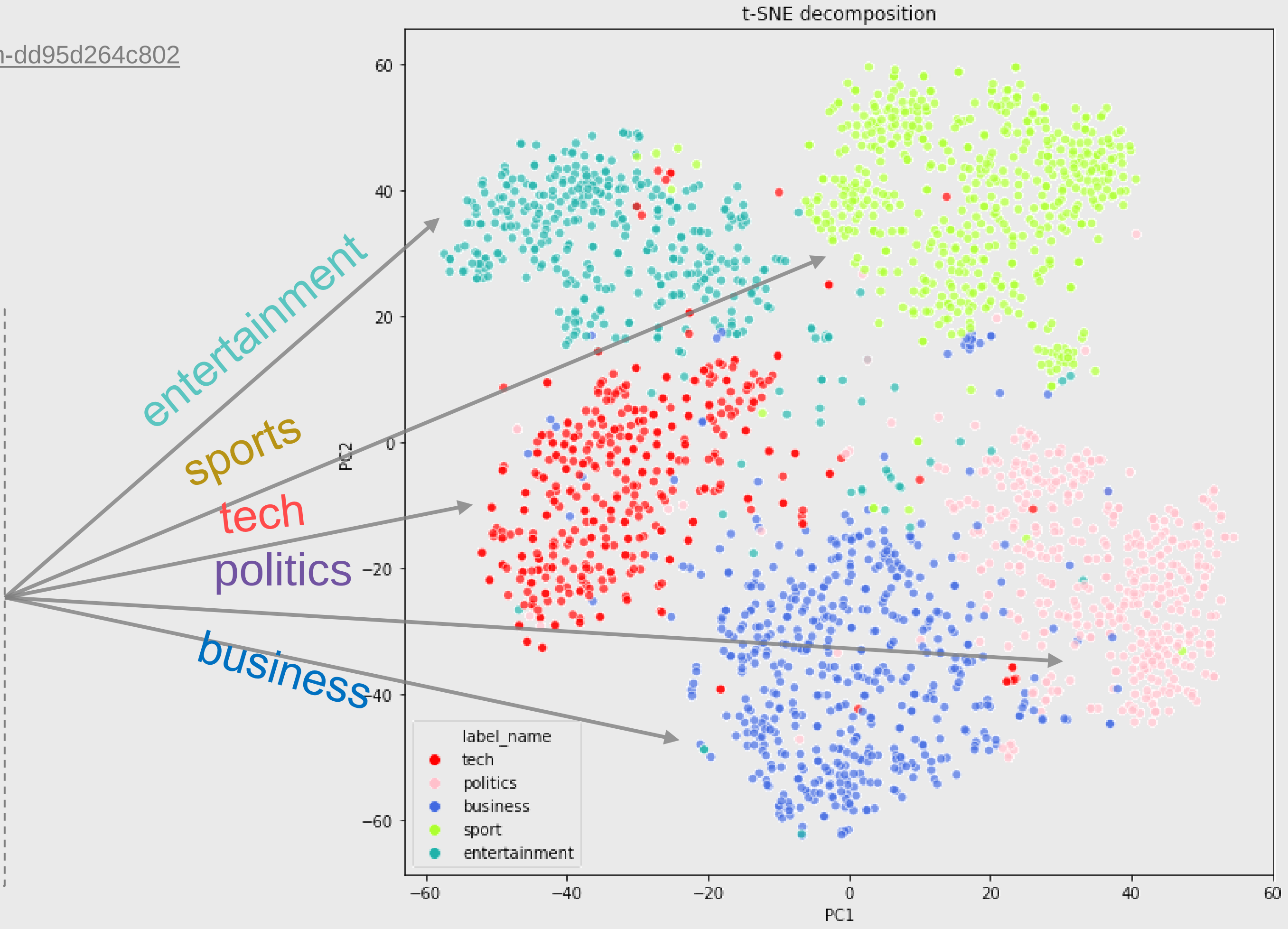
Is it possible to predict the following email text or a reply from the current?

Text Classification

Example Source:
https://towardsdatascience.com/text-classification-in-python-dd95d264c802

Question: The following text belongs to which category?

The use of electronic devices in the Commons chamber has long been frowned on. The sound of a mobile phone or a pager can result in a strong rebuke from either the Speaker or his deputies. The Speaker chairs debates in the Commons and is charged with ensuring order in the chamber and enforcing rules and conventions of the House. He or she is always an MP chosen by colleagues who, once nominated, gives up all party-political allegiances.

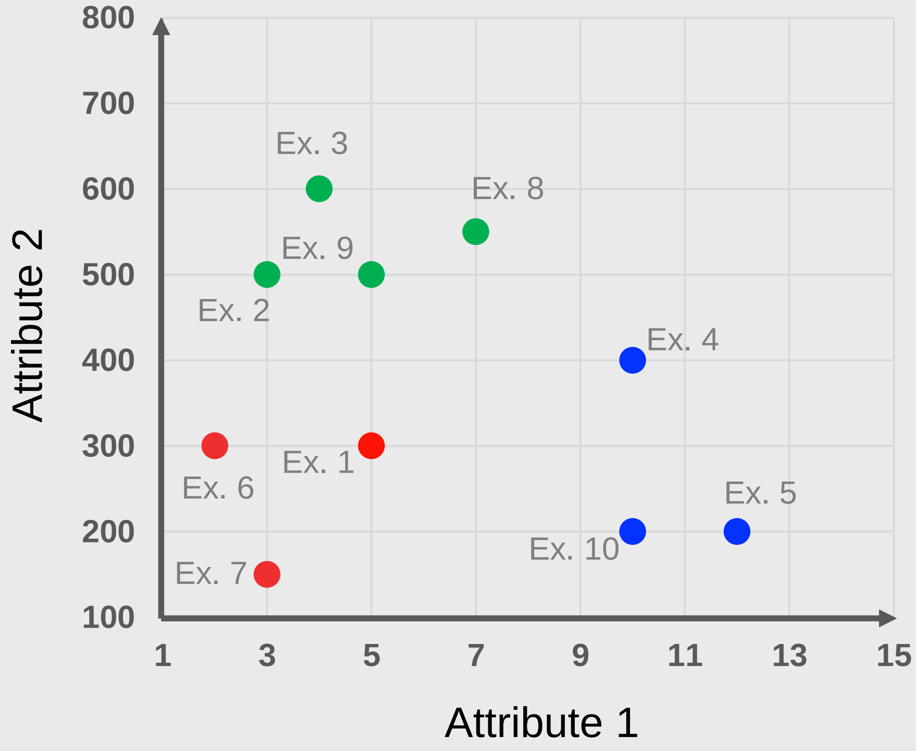


An Example Tools for Text Classification: K-Nearest Neighbor

Instance-based algorithm (Lazy Lerner)

Discrete labeled data

#	Inputs		Target
	Some Attr1	Some Attr2	Document Type
Ex. 1	5	300	Tech
Ex. 2	3	500	Entertainment
Ex. 3	4	600	Entertainment
Ex. 4	10	400	Politics
Ex. 5	12	200	Politics
Ex. 6	2	300	Tech
Ex. 7	3	150	Tech
Ex. 8	7	550	Entertainment
Ex. 9	5	500	Entertainment
Ex. 10	10	200	Politics

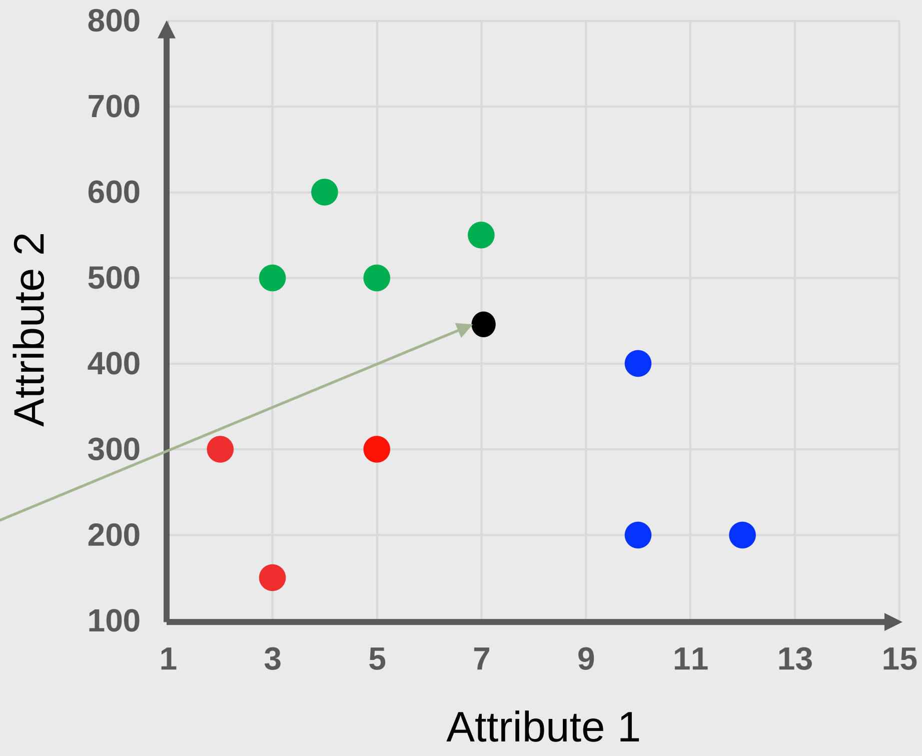


K-Nearest Neighbor

Question: What is the label of a new instance?

Unseen instance:

#	Inputs		Target
	Some Attr1	Some Attr2	Document Type
Ex. 11	7	450	**?**

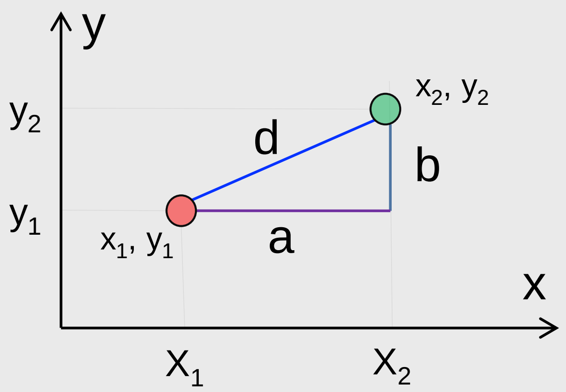


K-Nearest Neighbor

Answer: Find the K-Nearest Neighbors

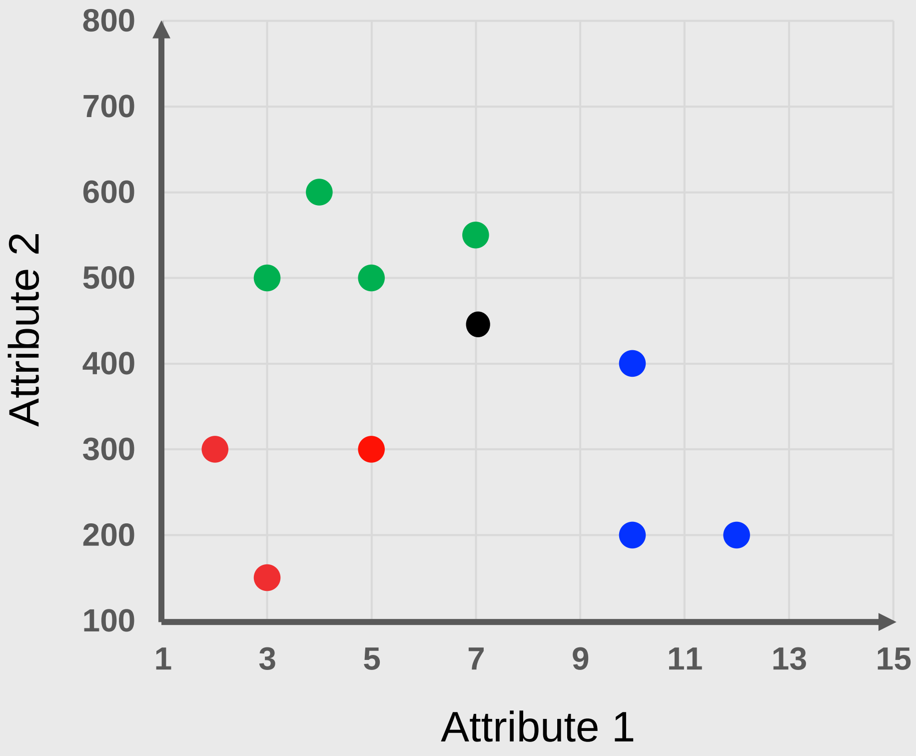
How we do that?

Euclidean distance:



$$d^2 = a^2 + b^2$$

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

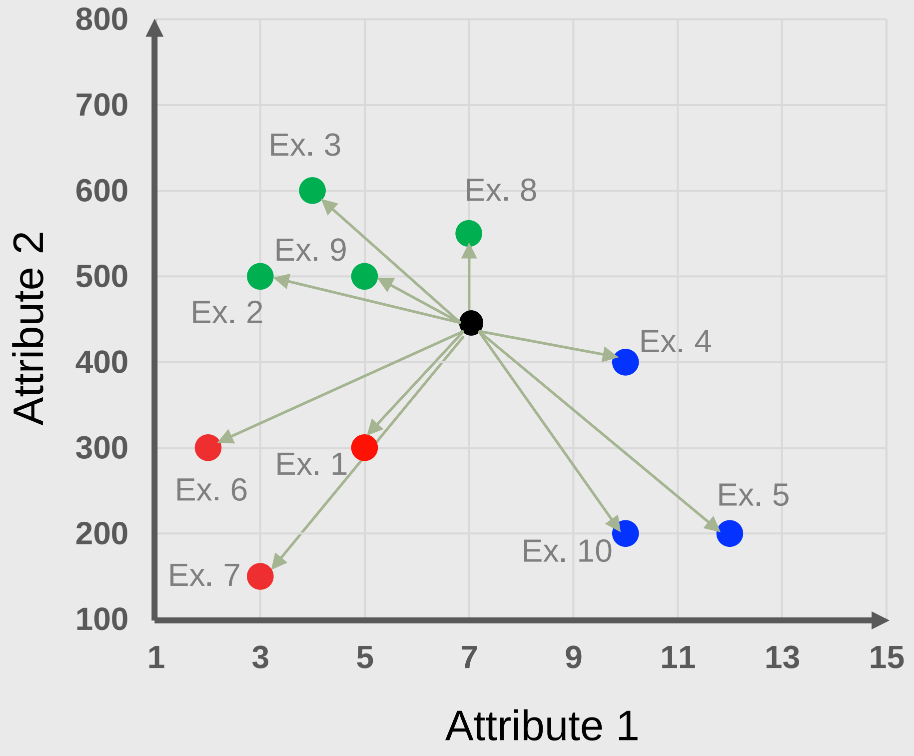


K-Nearest Neighbor

Which neighbor is the nearest?

All examples Euclidean distance:

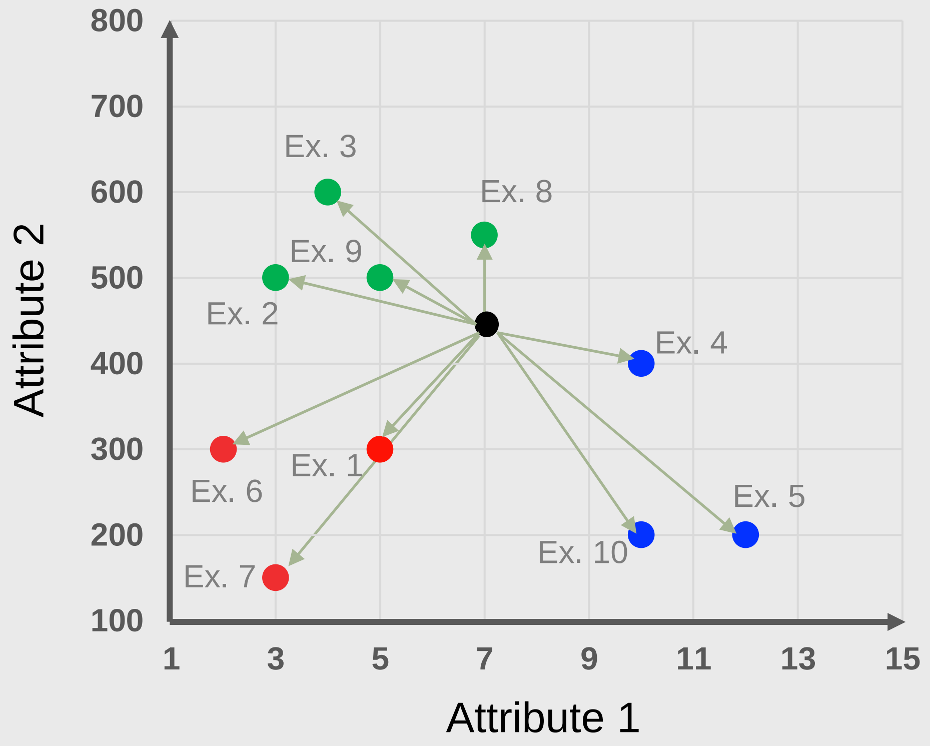
EX. 1	150.02
EX. 2	50.16
EX. 3	150.03
EX. 4	50.09
EX. 5	250.05
EX. 6	150.08
EX. 7	300.03
EX. 8	100.00
EX. 9	50.04
EX. 10	250.02



K-Nearest Neighbor

Shortest distance first?

EX. 9	50.04
EX. 4	50.09
EX. 2	50.16
EX. 8	100.00
EX. 3	150.03
EX. 1	150.02
EX. 6	150.08
EX. 10	250.02
EX. 5	250.05
EX. 7	300.03



K-Nearest Neighbor

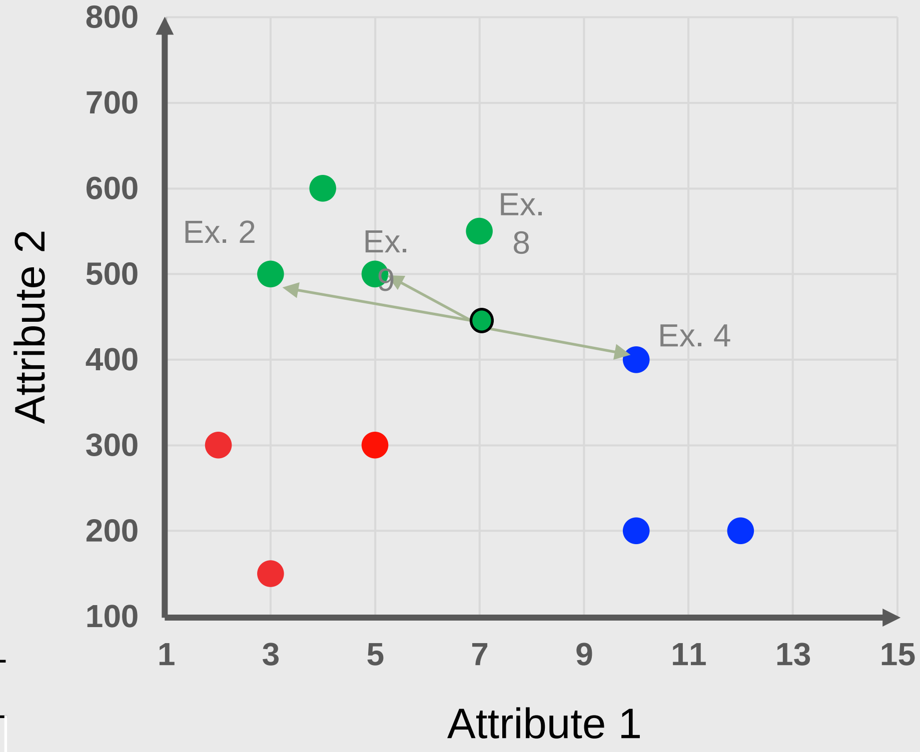
Let’s set K = 3

Ex. 9 → 50.04 Neighbor 1

Ex. 4 → 50.09 Neighbor 2

Ex. 2 → 50.16 Neighbor 3

Majority of neighbors have label
“Green → Entertainment”



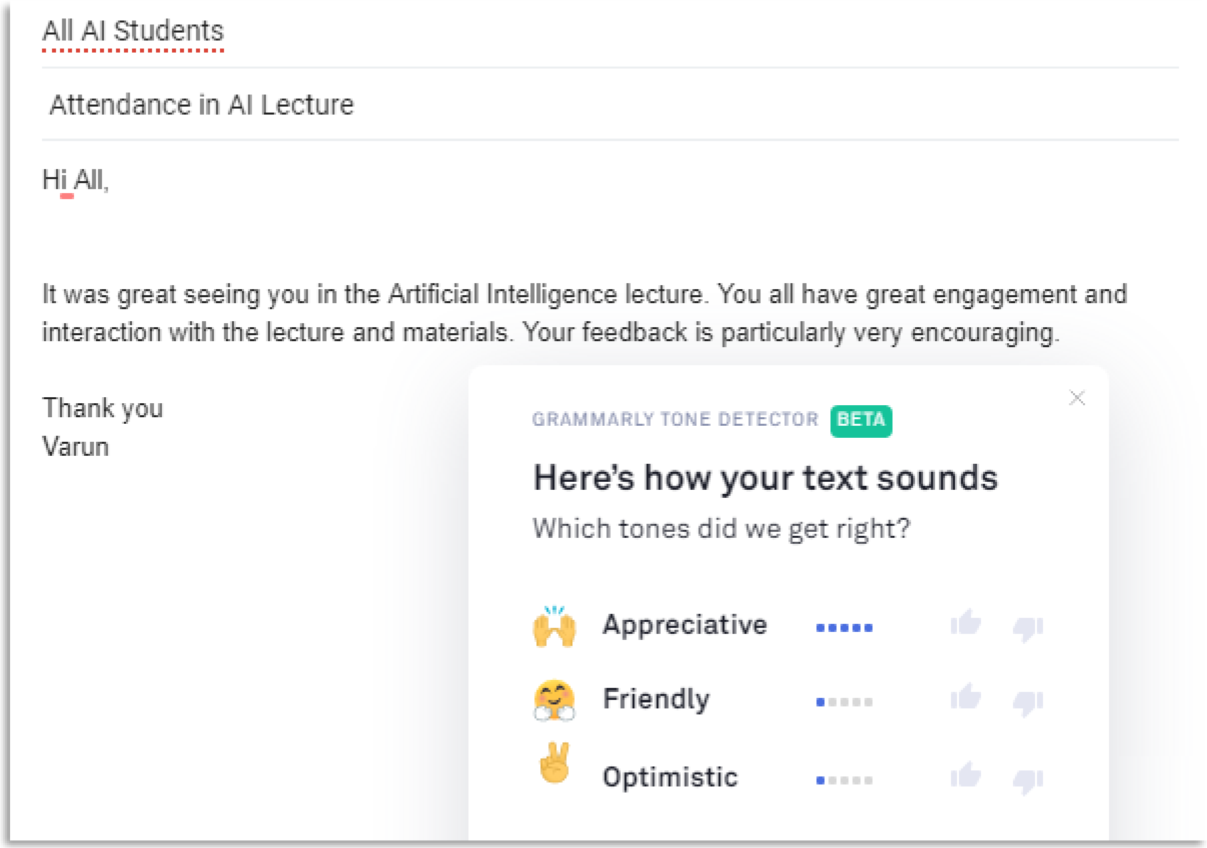
Unseen instance:

#	Inputs		Target
	Some Attr1	Some Attr2	Document Type
Ex. 11	7	450	**Entertainment**

Natural Language Understanding

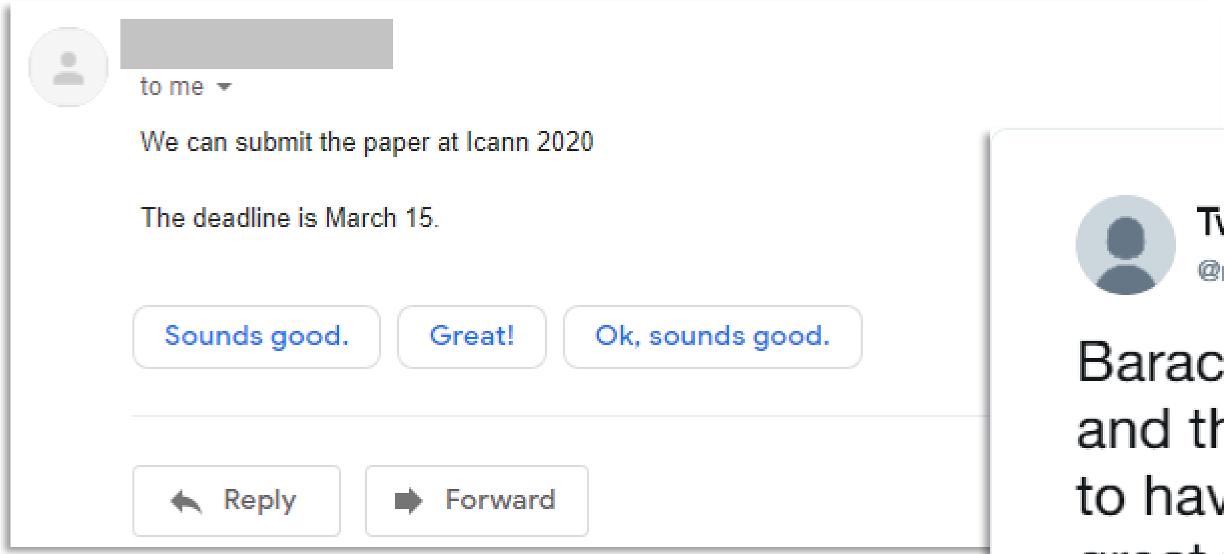
Artificial Intelligence based

Sentiment analysis and Emotion detection



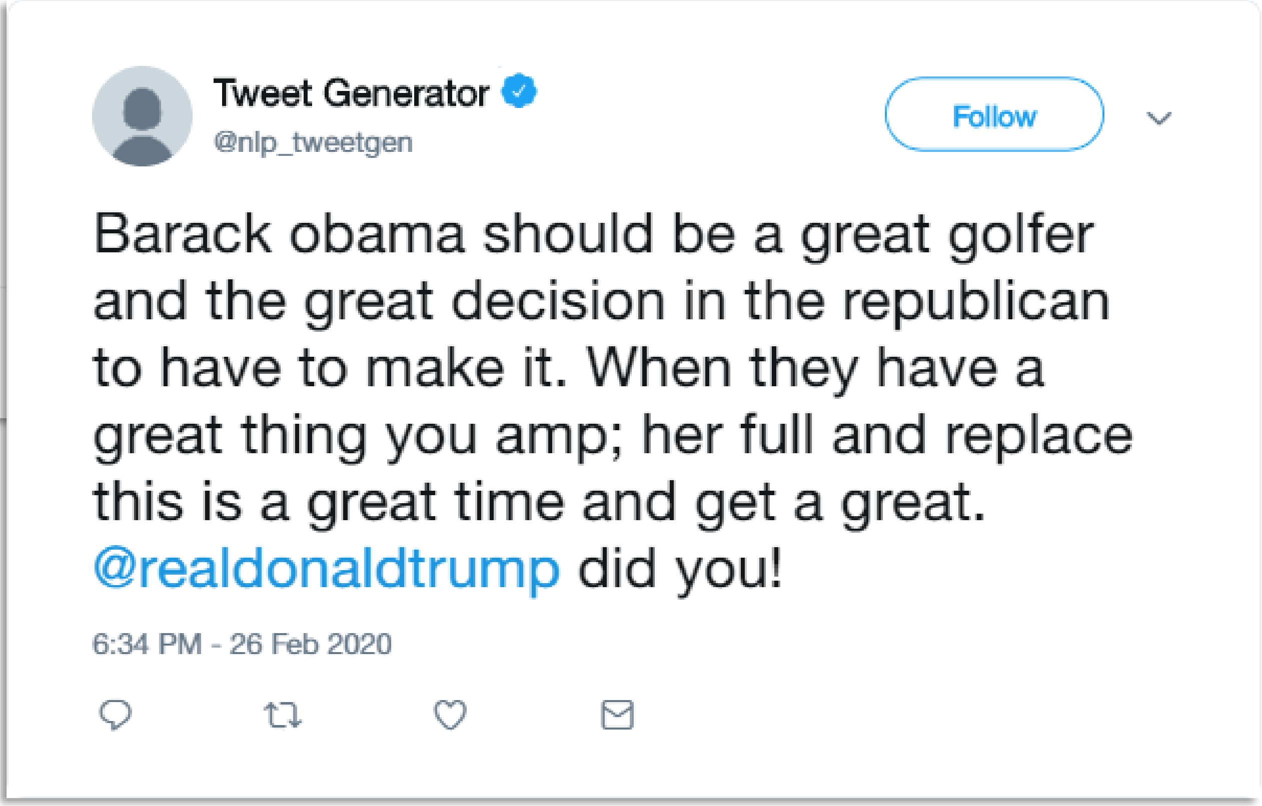
8:43 PM

Natural Language Generation



Gmail response prediction system

Produced by Joe Rickard, CS FY Students, 2020



Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 8/10: Natural Language Processing

Part 2

Text Data Pre-Processing

DR VARUN OJHA

Department of Computer Science

University of Reading

Word Embedding

Given a word W (e.g. “intelligence”) we want W to be a real vector of dimension n. Dimension n is also called word embedding dimension.

$$\boldsymbol{W}\colon \text{words} \rightarrow \mathbb{R}^n$$

“intelligence” $\rightarrow (w_1, w_2, \ldots, w_n) \rightarrow (0.1, -0.8, \ldots, 0.9)$

Word Embedding

$W_1 =$ "I love artificial intelligence"
$W_2 =$ "I like computational intelligence"

We create a vocabulary V collecting all unique words.

$V =$ {"I", "love", "like", "artificial", "computational", "intelligence"}

For this example vocabulary size $|V| = 6$

Word Embedding: Word → Integer

V = {"I", "love", "artificial", "computational", "intelligence", "like"}

I → 0

love → 1

like → 2

artificial → 3

computational → 4

intelligence → 5

Word Embedding: Integer → Word

V = {"I", "love", "artificial", "computational", "intelligence", "like"}

0 → I

1 → love

2 → like

3 → artificial

4 → computational

5 → intelligence

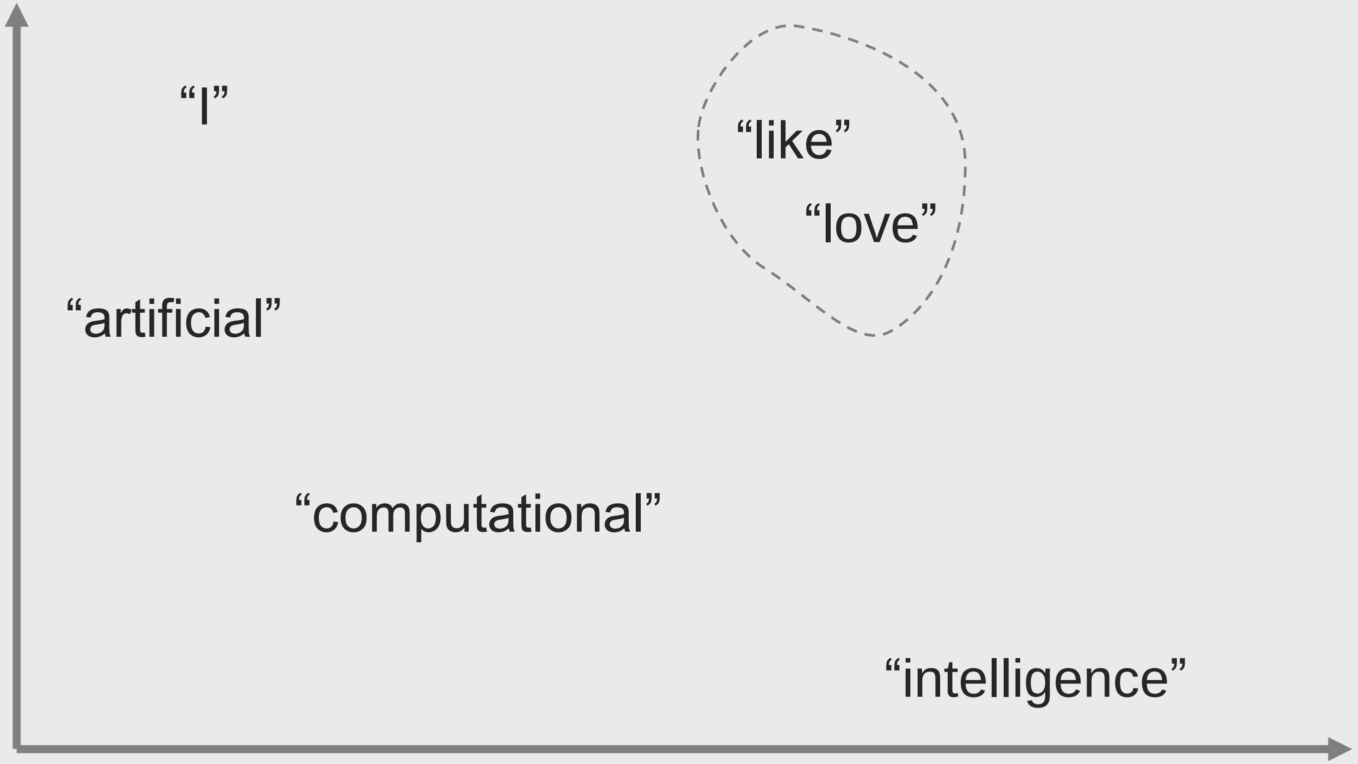
One-Hot Encoding

$V = \{$“I”, “love”, “like”, “artificial”, “computational”, “intelligence”$\}$

“I” = $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ “love” = $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ “like” = $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ “artificial” = $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

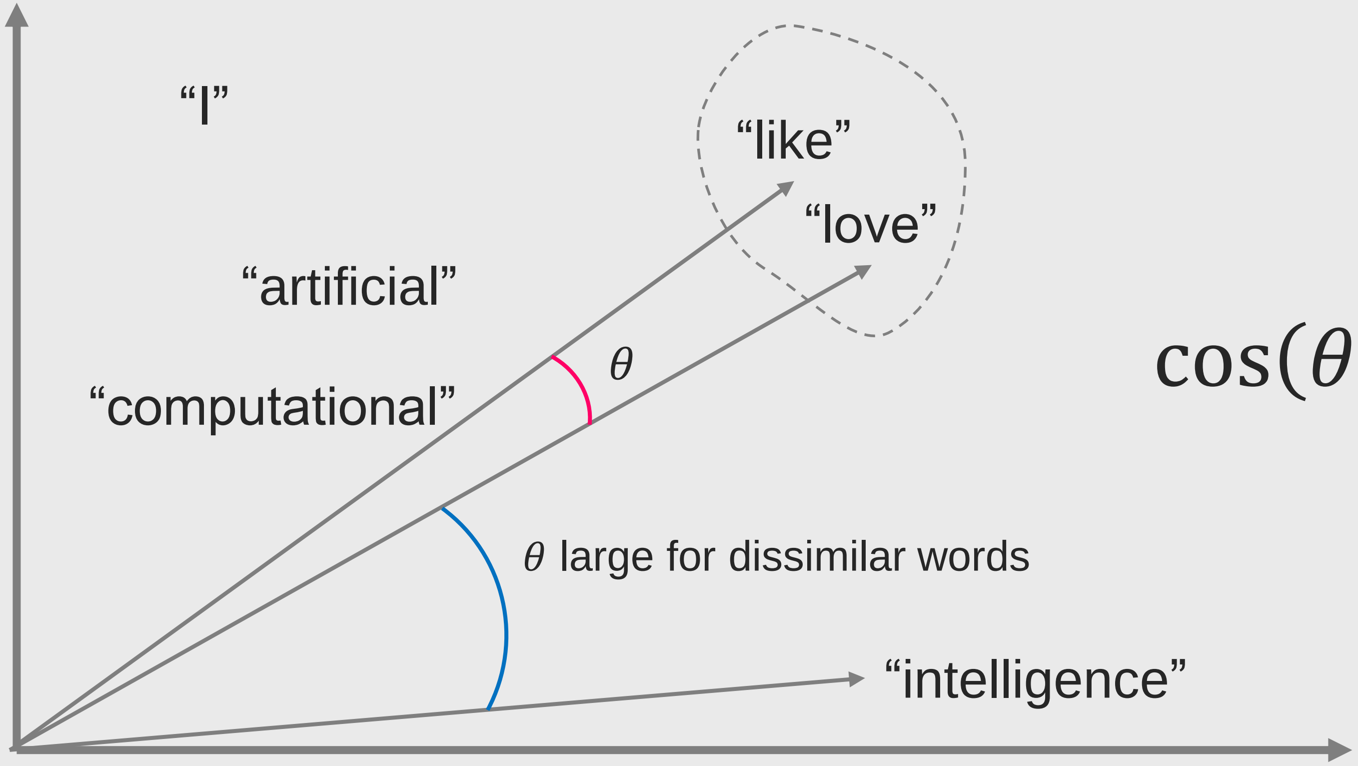
“computational” = $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ “intelligence” = $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Similarity between words?



Objective is to place similar words close to each other

Similarity between words?

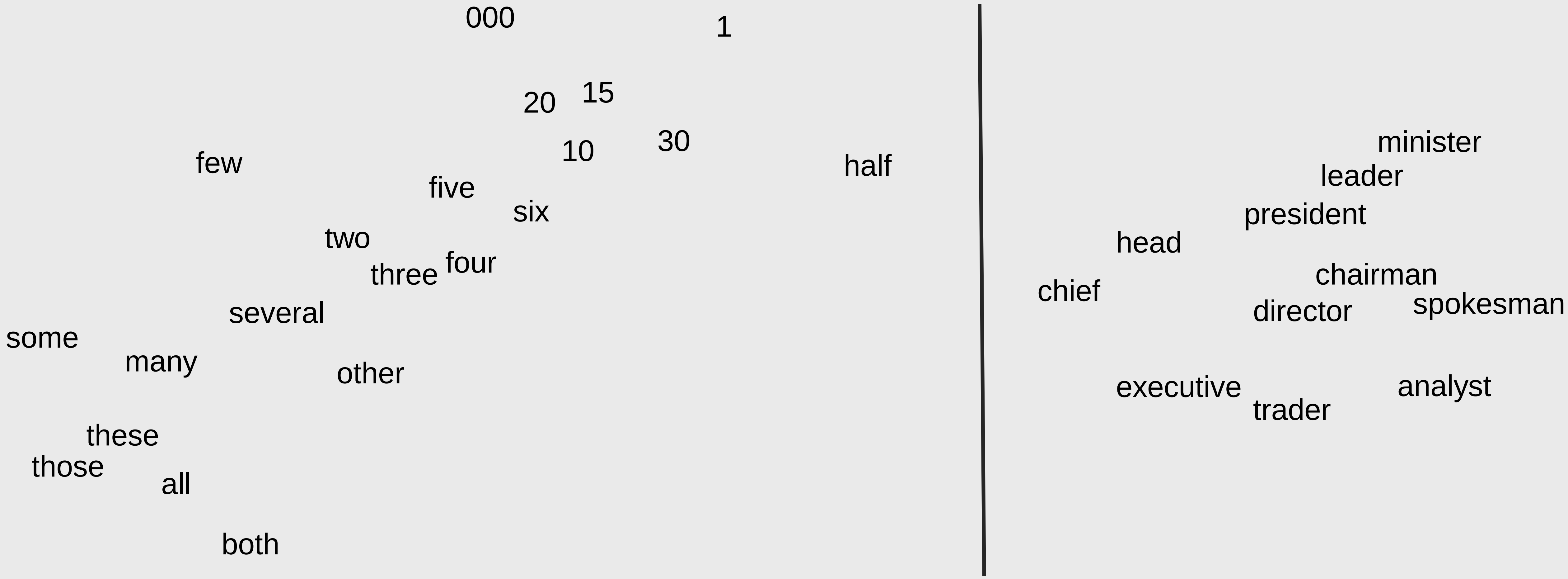


$$\cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

Objective is to place similar words close to each other

t-SNE visualisation of words

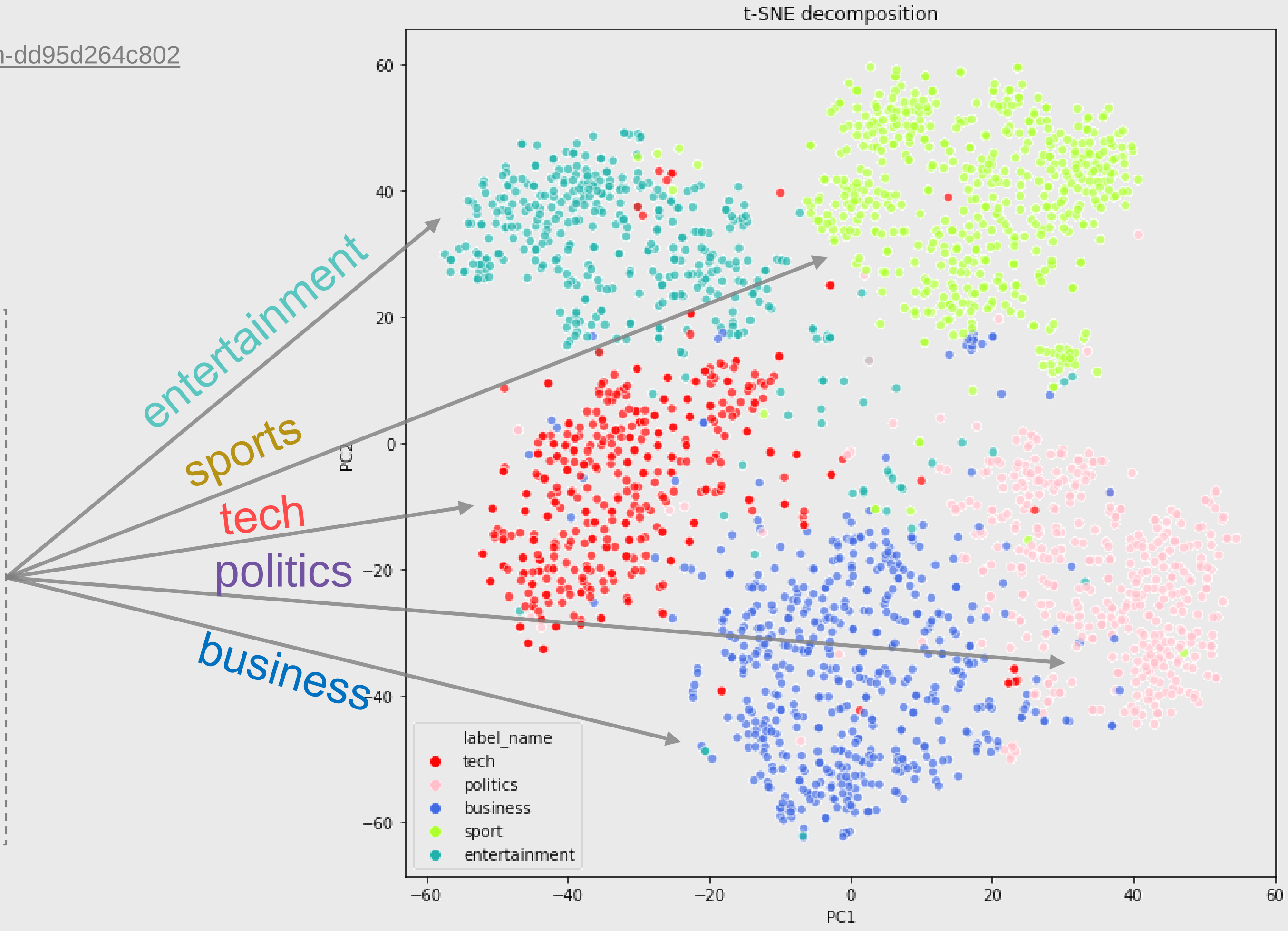
Turian *et al.* (2010)



t-SNE visualisation

Example Source:
https://towardsdatascience.com/text-classification-in-python-dd95d264c802

All similar topics are closer to each other



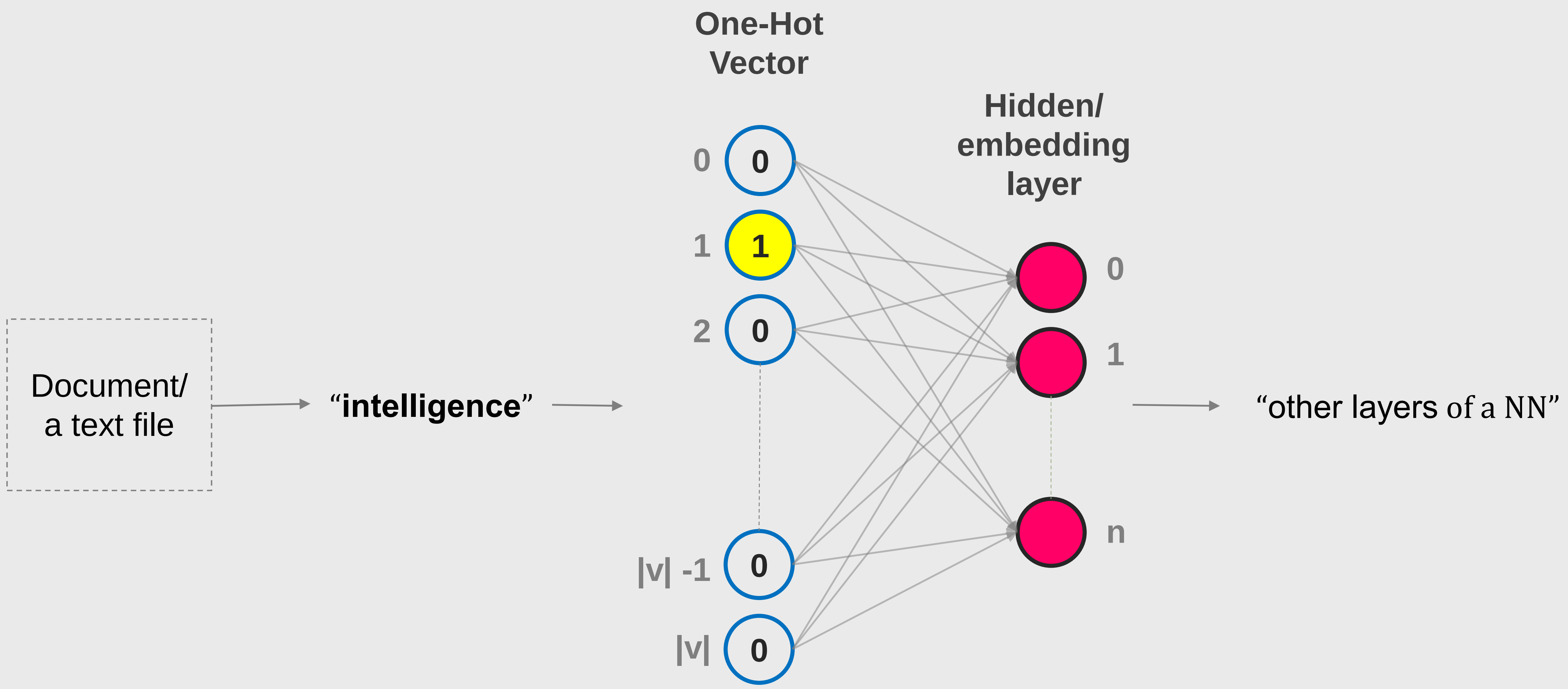
Word Embedding: Objective

Given a word W (e.g. “intelligence”) we want to W a real vector of dimension n

$$\boldsymbol{W}: \text{words} \rightarrow \mathbb{R}^n$$

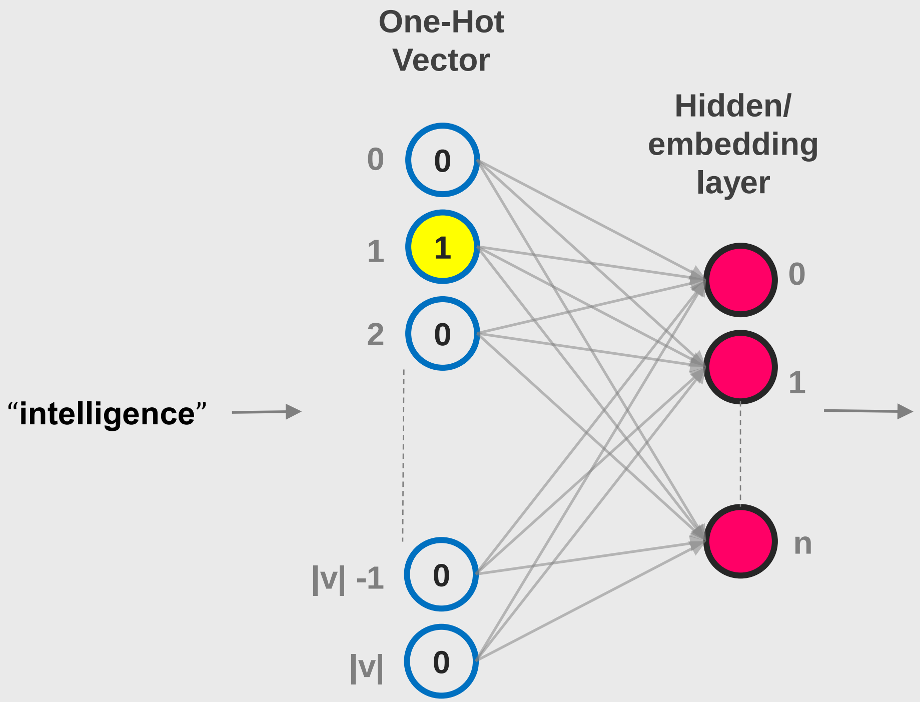
“intelligence” $\rightarrow (w_1, w_2, \ldots, w_n) \rightarrow (0.1, -0.8, \ldots, 0.9)$

Word Embedding: Objective



Dr Varun Ojha, University of Reading, UK

Word Embedding: Objective



$$\text{"intelligence"} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.5 & 4.6 & \dots & 0.9 \\ 0.1 & -0.8 & \dots & 0.7 \\ \vdots & \vdots & & \vdots \\ 0.6 & 0.8 & \dots & 0.3 \\ 0.3 & -0.6 & \dots & -0.8 \\ -0.5 & 0.5 & \dots & 0.1 \end{bmatrix}$$

$$(1 \times |V|) \cdot (|V| \times n) \Rightarrow (1 \times n)$$

Word Embedding: Objective

$$\text{“intelligence”} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.5 & 4.6 & \dots & 0.7 \\ 0.1 & -0.8 & \dots & 0.9 \\ 0.6 & 0.8 & \dots & 0.3 \\ 0.3 & -0.6 & \dots & -0.8 \\ -0.5 & 0.5 & \dots & 0.1 \end{bmatrix}$$

$$= [0.1, -0.8, \dots, 0.9]$$

Word Embedding: Objective

Embedding dimension n

“intelligence”

Input token

$$\begin{bmatrix} 0.5 & 4.6 & \dots & 0.9 \\ 0.1 & -0.8 & \dots & 0.7 \\ 0.6 & 0.8 & \dots & 0.3 \\ \vdots & \vdots & & \vdots \\ 0.3 & -0.6 & \dots & -0.8 \\ -0.5 & 0.5 & \dots & 0.1 \end{bmatrix}$$

Vocabulary size

Lookup Table
Embedding Weight Matrix

Bag of Words (BoW)

$D_1 =$ "John likes to watch movies. Mary likes movies too."

$D_2 =$ "John also likes to watch football games."

BoW_1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1}

BoW_2 = {" John ":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1}

$$BoW_3 = BoW_1 \biguplus BoW_2$$

V = {john, likes, to, watch, movies, mary, too, also, football games}

V = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Bag of Words (BoW): Union of Documents

V = {john, likes, to, watch, movies, mary, too, also, football games}

BoW_1 = {"John":1, "likes":2, "to":1, "watch":1, "movies":2, "Mary":1, "too":1}

$$\boldsymbol{BoW_1} = [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]$$

BoW_2 = {" John ":1, "also":1, "likes":1, "to":1, "watch":1, "football":1, "games":1}

$$\boldsymbol{BoW_2} = [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]$$

n-gram language model

Document = "Varun likes to watch football games"

We can compute probability of a sequence of words

$$P(w_1, w_2, \dots, w_n) = \prod_i P(w_i \mid w_1, w_2, w_{i-1})$$

n-gram language model

$$P(\text{Varun}, \text{likes}, \text{to}, \text{watch}, \text{football}, \text{games})$$

Or **Probability of a word given a sequence of n words**

$$P(w_n \mid w_1, w_2, \dots, w_{n-1})$$

n-gram language model

$$P(\textbf{games} \mid \text{Varun}, \text{likes}, \text{to}, \text{watch}, \text{football})$$

n-gram language model

$$P(w_i \mid w_{i-1}) = P(\textbf{games} \mid \text{football})$$

bi-gram language model because its word words

n-gram language model

Document in *Hindi* = वरुण को फुटबॉल खेल देखना पसंद है

(roughly read as *varun ko phutabol khel dekhana pasand hai)*

Actual English Translation (by human): Varun likes to watch football games

Machine Translation

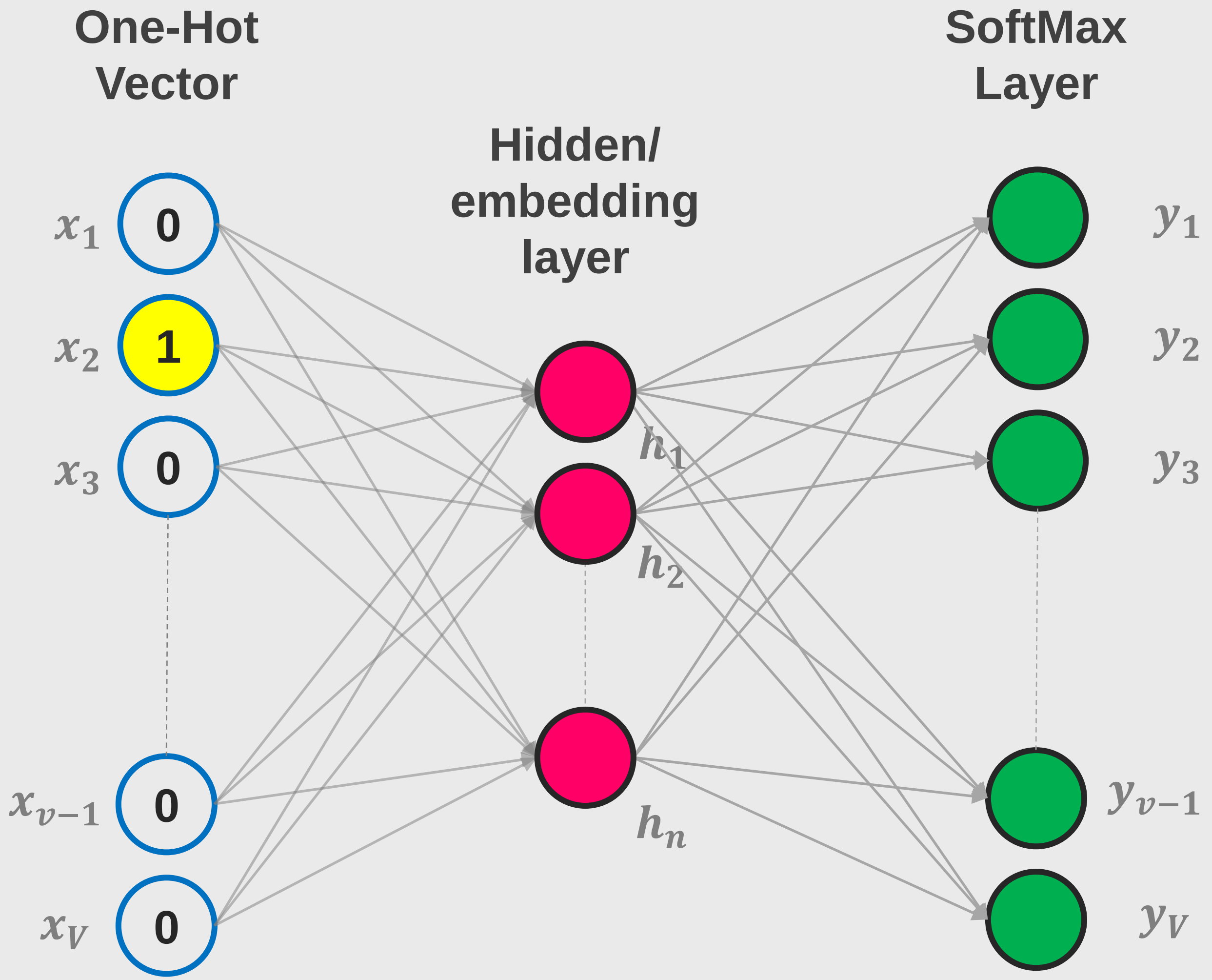
$$P(\ w_1, w_2, \dots, w_n)$$

$$P(\text{Varun}, \text{likes}, \text{to}, \text{watch}, \text{football}, \text{games}) > P(\text{Varun}, \text{love}, \text{to}, \text{watch}, \text{football}, \text{games})$$

Grammar correction

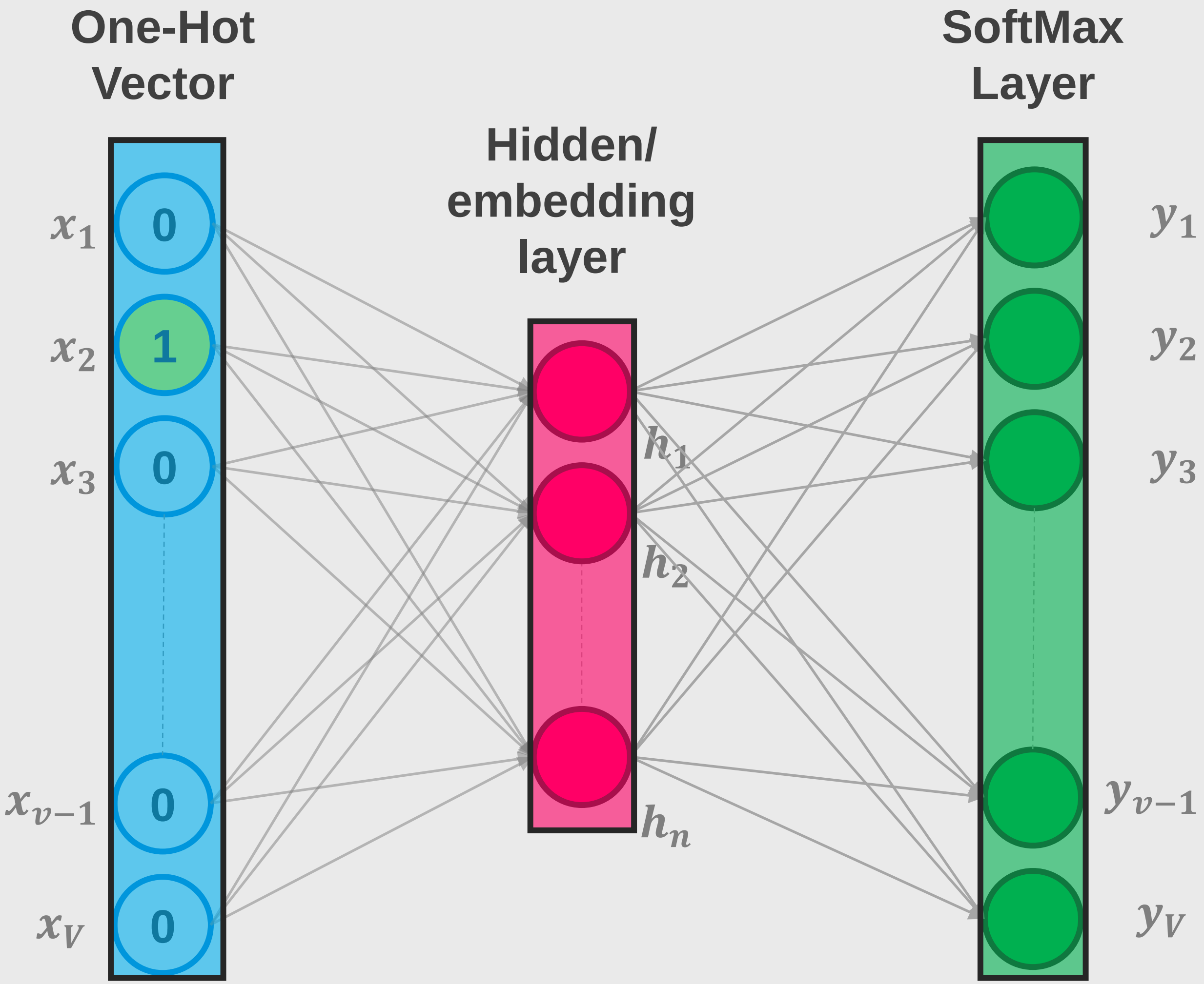
$$P(\text{Varun}, \textbf{likes}, \text{to}, \text{watch}, \text{football}, \text{games}) > P(\text{Varun}, \textbf{like}, \text{to}, \text{watch}, \text{football}, \text{games})$$

Common Bag of Word Encoding: *1*-gram



SoftMax Layer provide probabilistic context $\boldsymbol{y_i}$ to a word (1-gram) $\{\boldsymbol{x_i}\}$ in the Bag of size $\boldsymbol{V}$

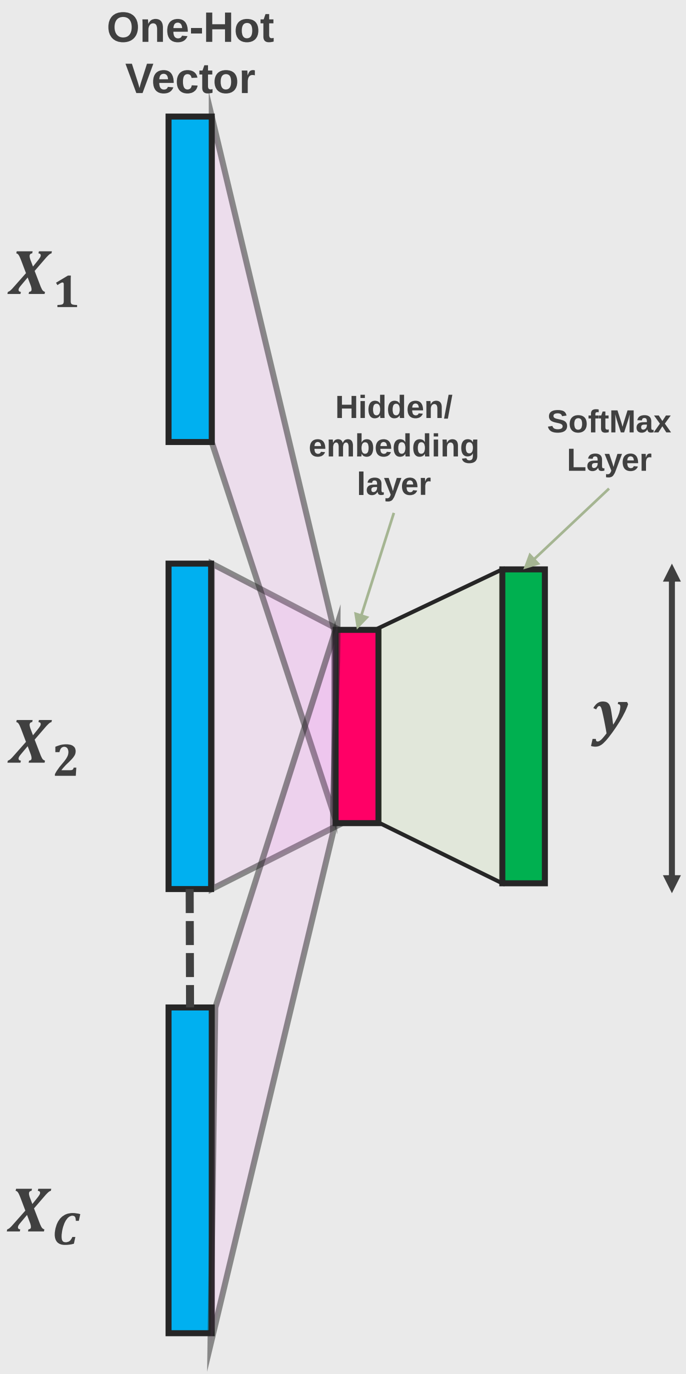
Common Bag of Word Encoding: *1*-gram



Let's represent dense layer like a block

Common Bag of Word (COBW) Encoding:

C-gram



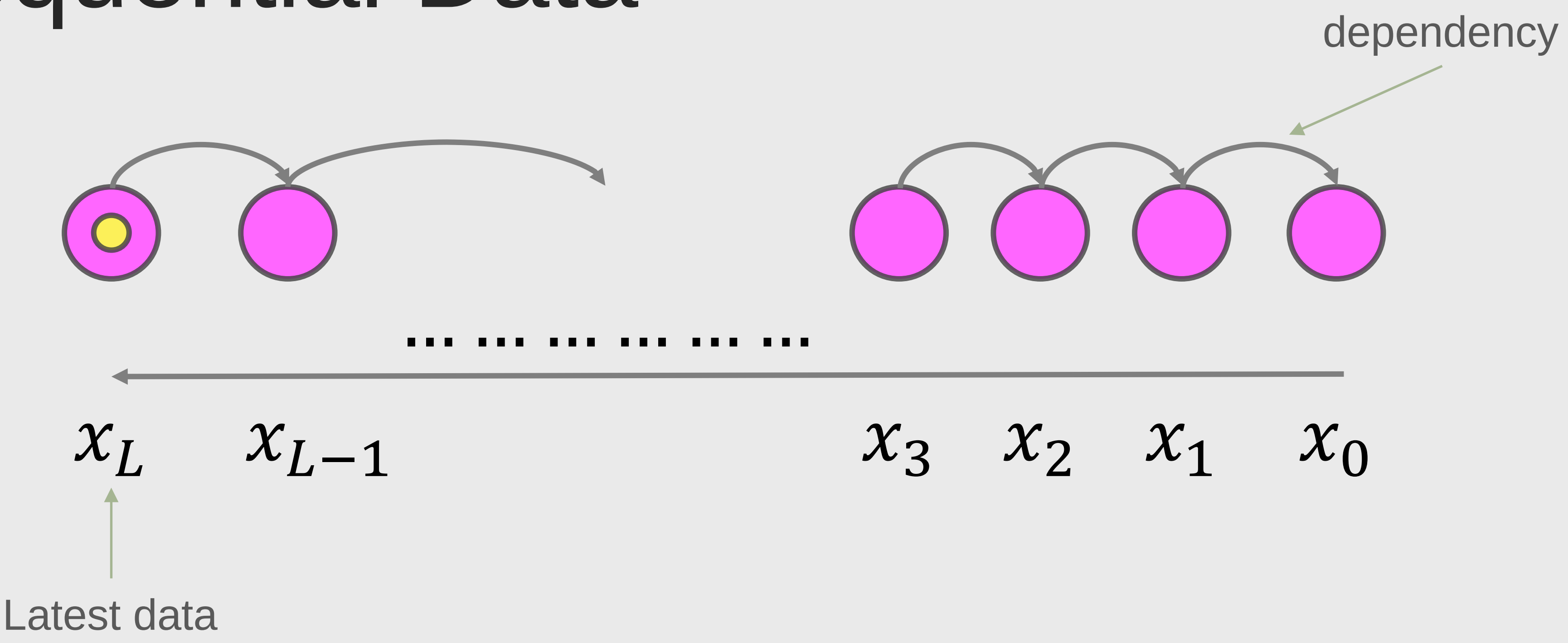
SoftMax Layer provide probabilistic context $\boldsymbol{y_i}$ to a *C-gram* $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_C}\}$ in the Bag of size $\boldsymbol{V}$

Sequential Data

… … … … … …

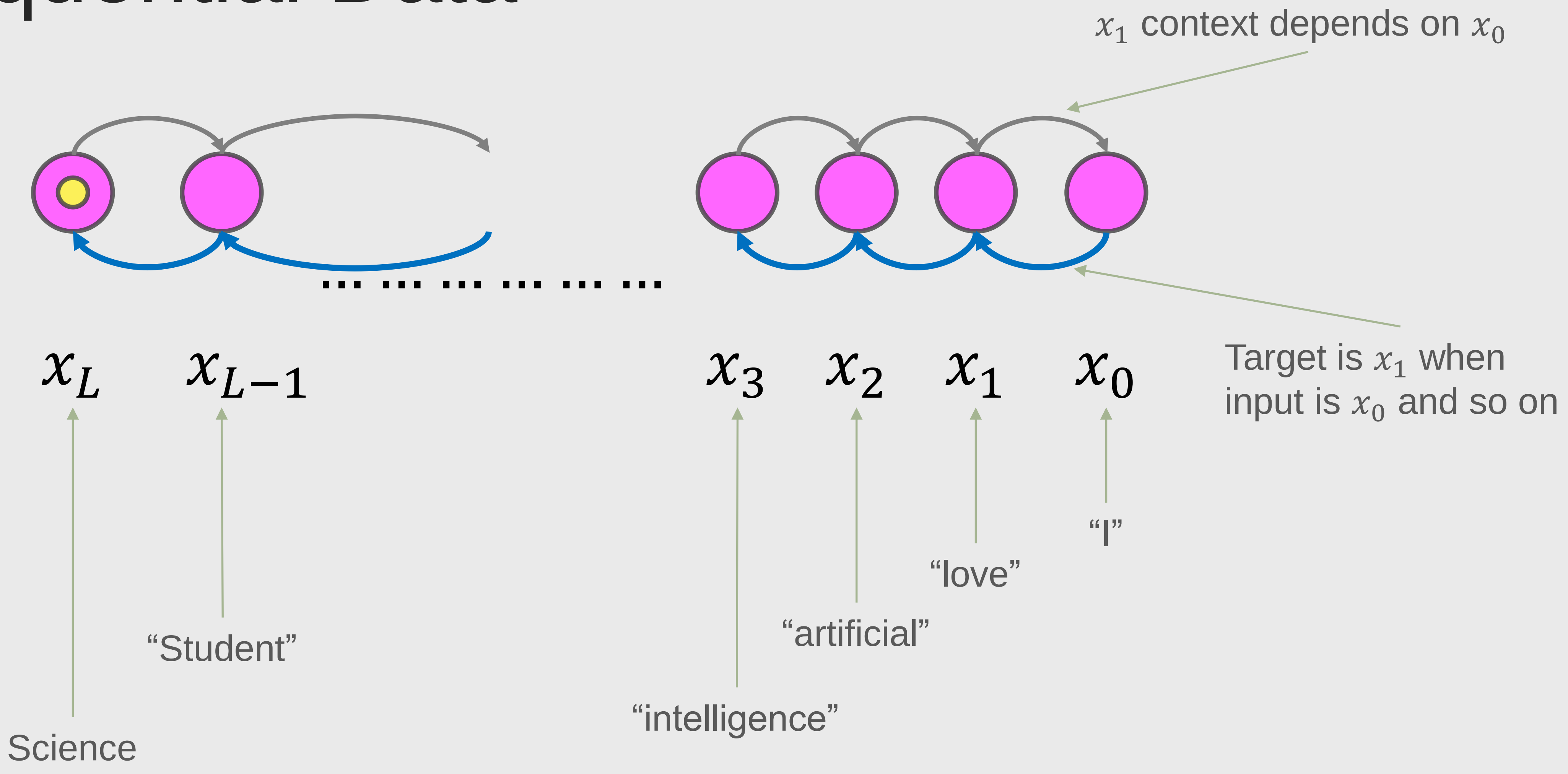
Let’s say, we have data (document) $\mathbf{x} = \{x_0, x_1, x_2, \ldots, x_L\}$ of length L arriving at time $t = 0$ until time $t = L$

Sequential Data



data set $\mathbf{x} = \{x_0, x_1, x_2, \ldots, x_L\}$ of length L arriving at time $t = 0$ until time $t = L$. Hence latest data is x_L and it depends on its previous data

Sequential Data



Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 8/10: Natural Language Processing

Part 3

Recurrent Neural Network

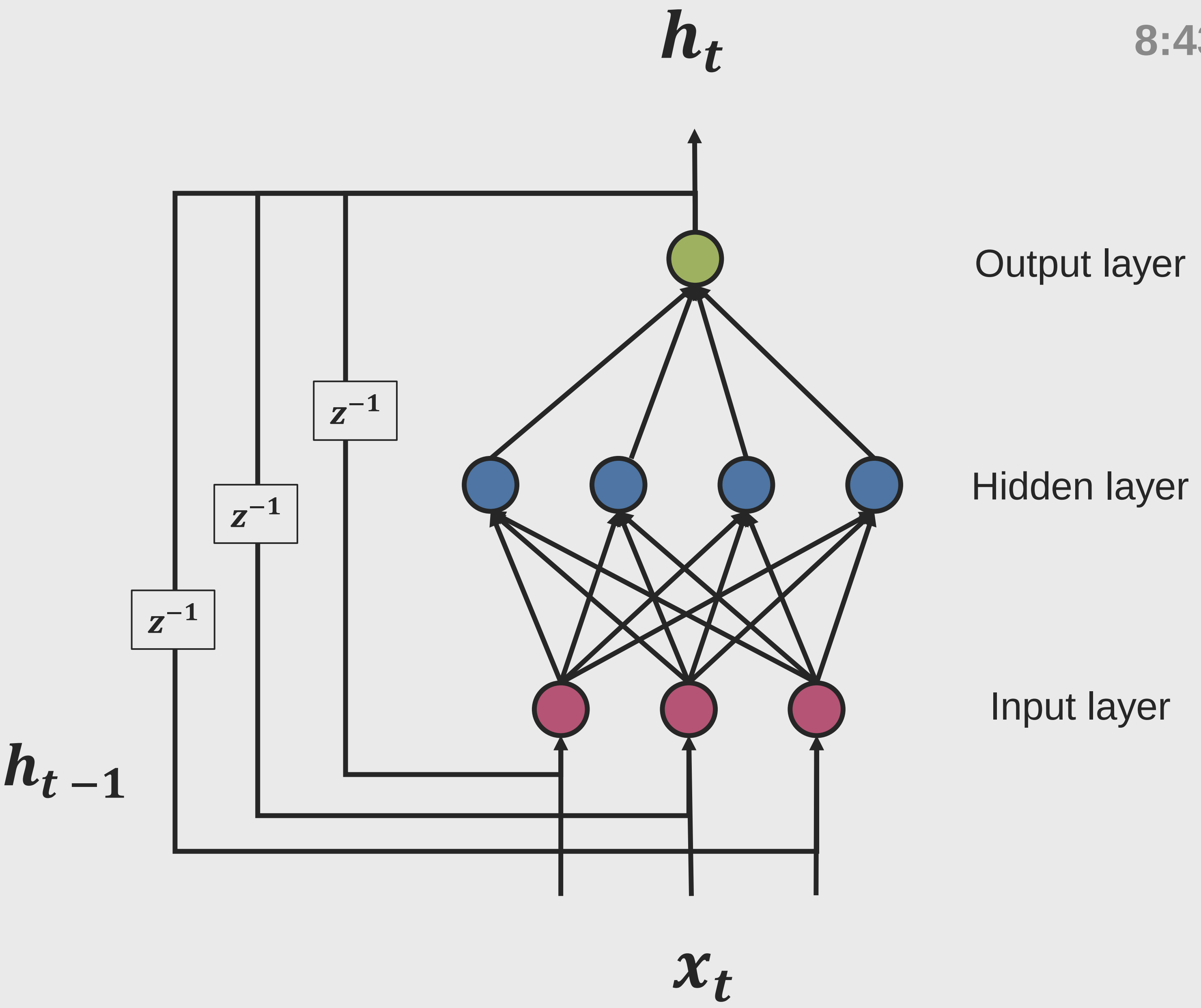
DR VARUN OJHA

Department of Computer Science

University of Reading

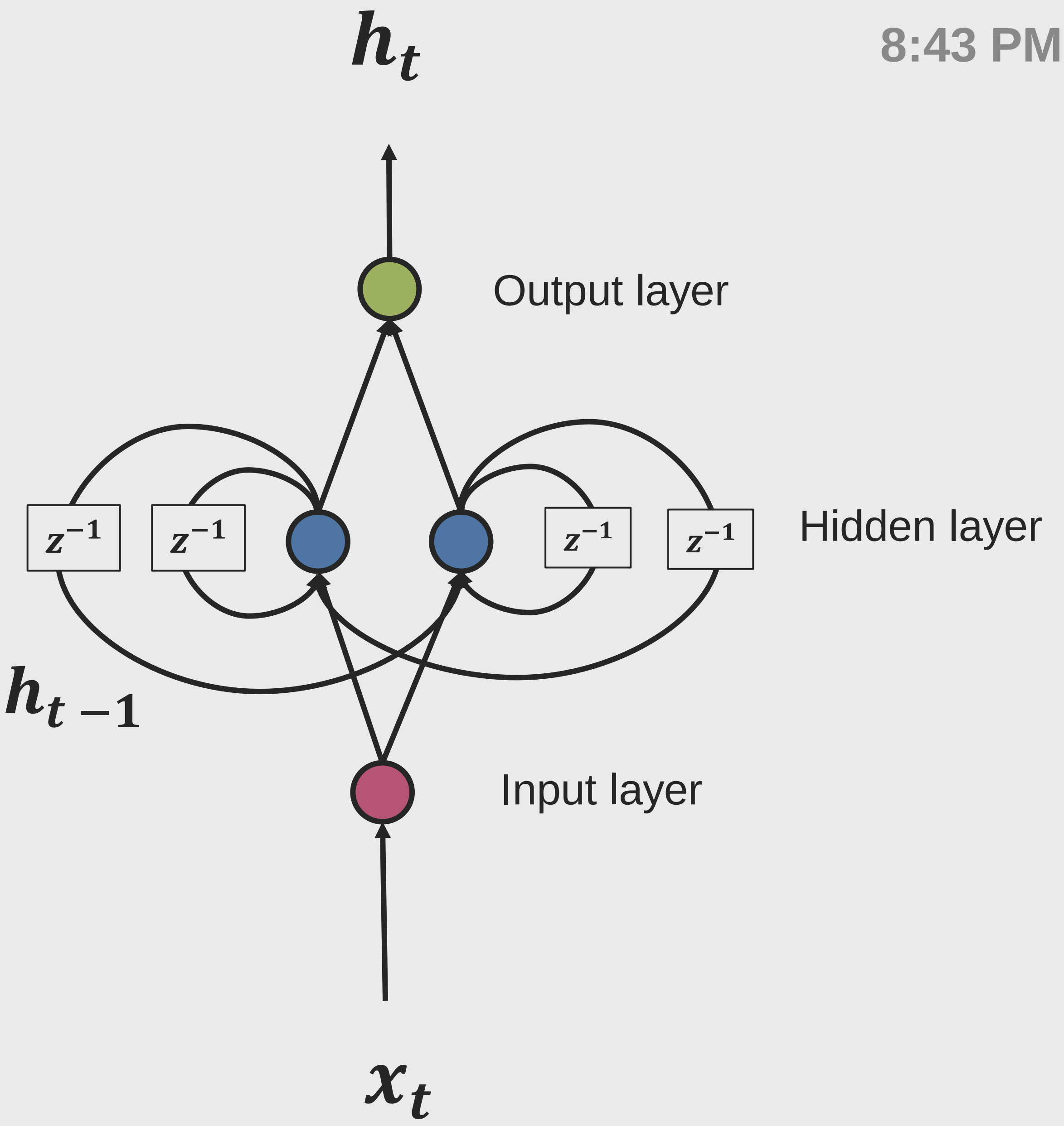
RECURRENT NEURAL NETWORK (RNN)

Architecture



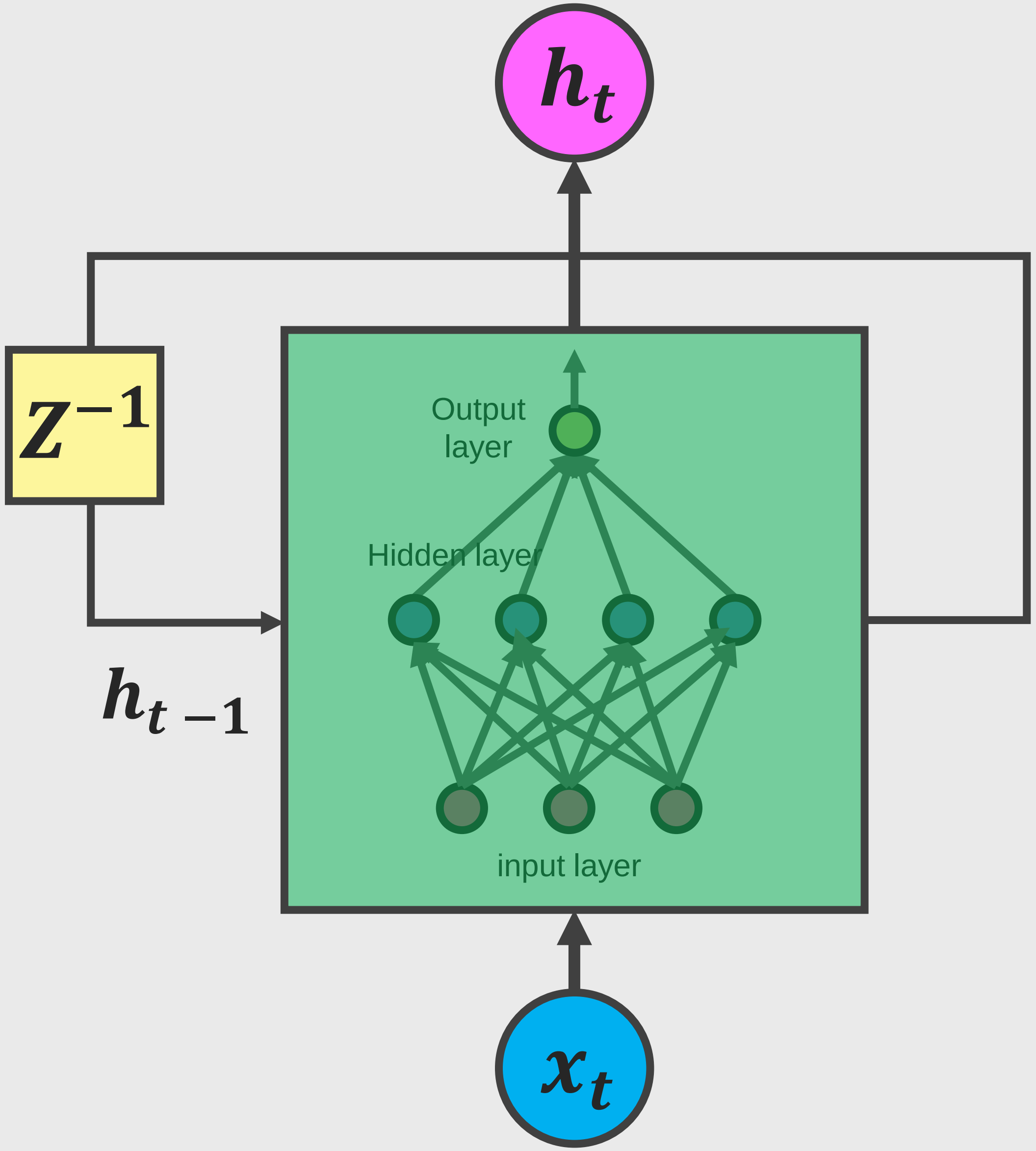
RECURRENT NEURAL NETWORK (RNN)

Architecture



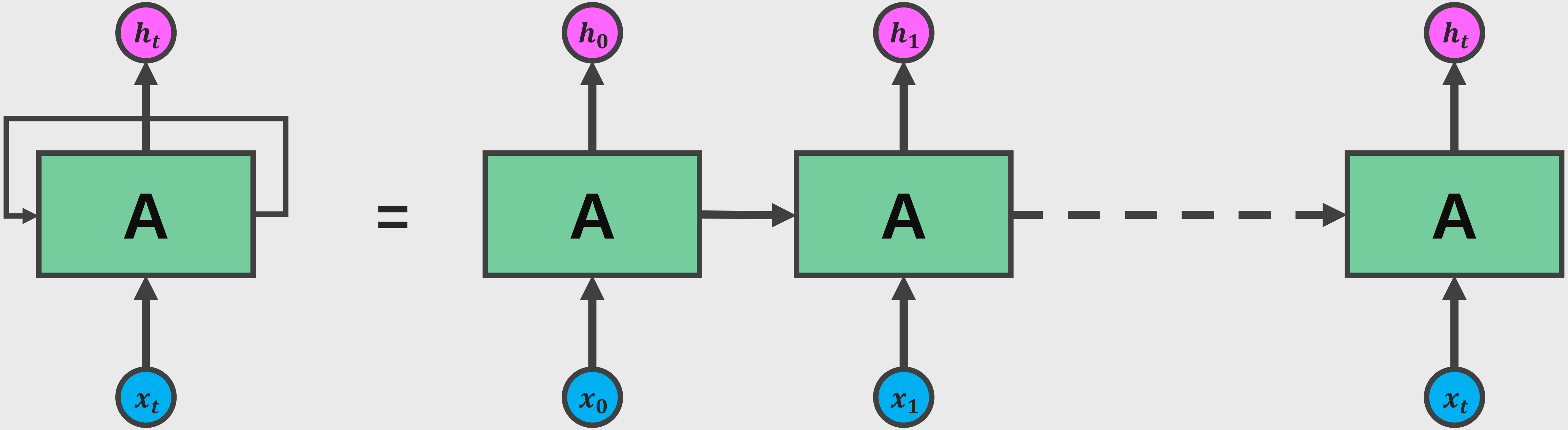
RECURRENT NEURAL NETWORK (RNN)

Architecture



RNN

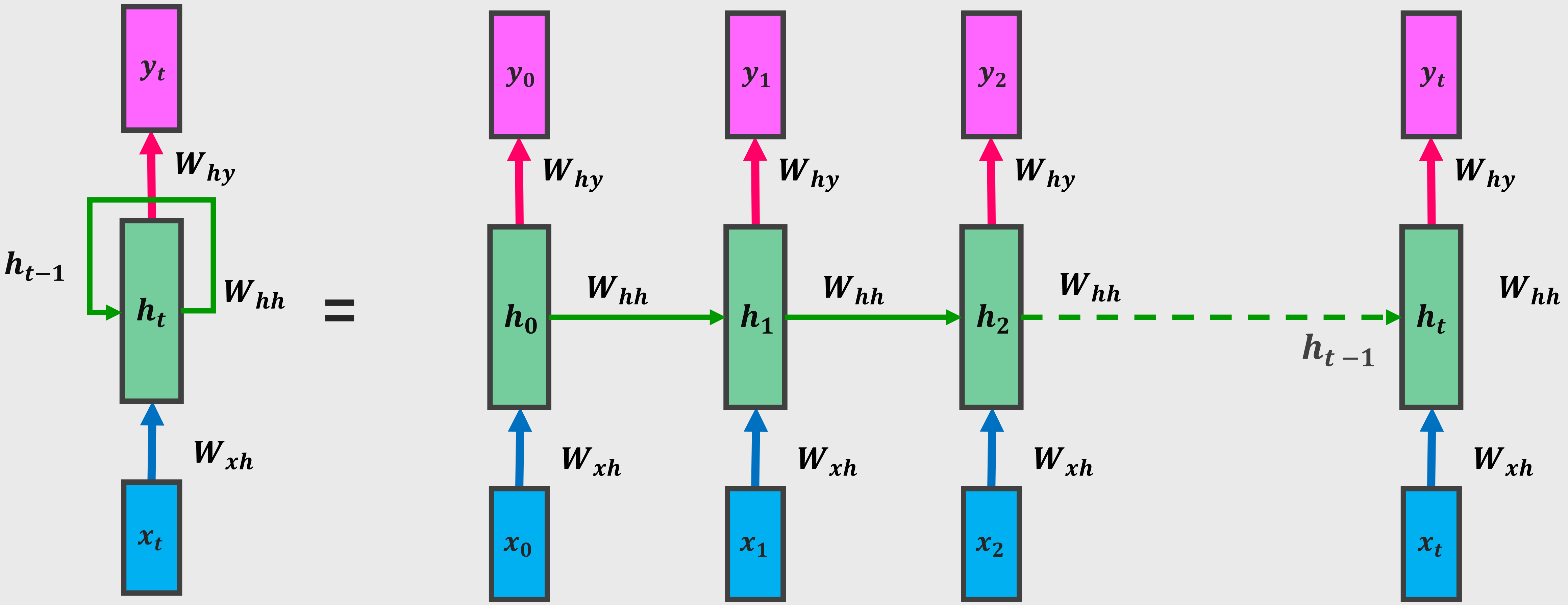
Architecture Unrolled version for sequential data



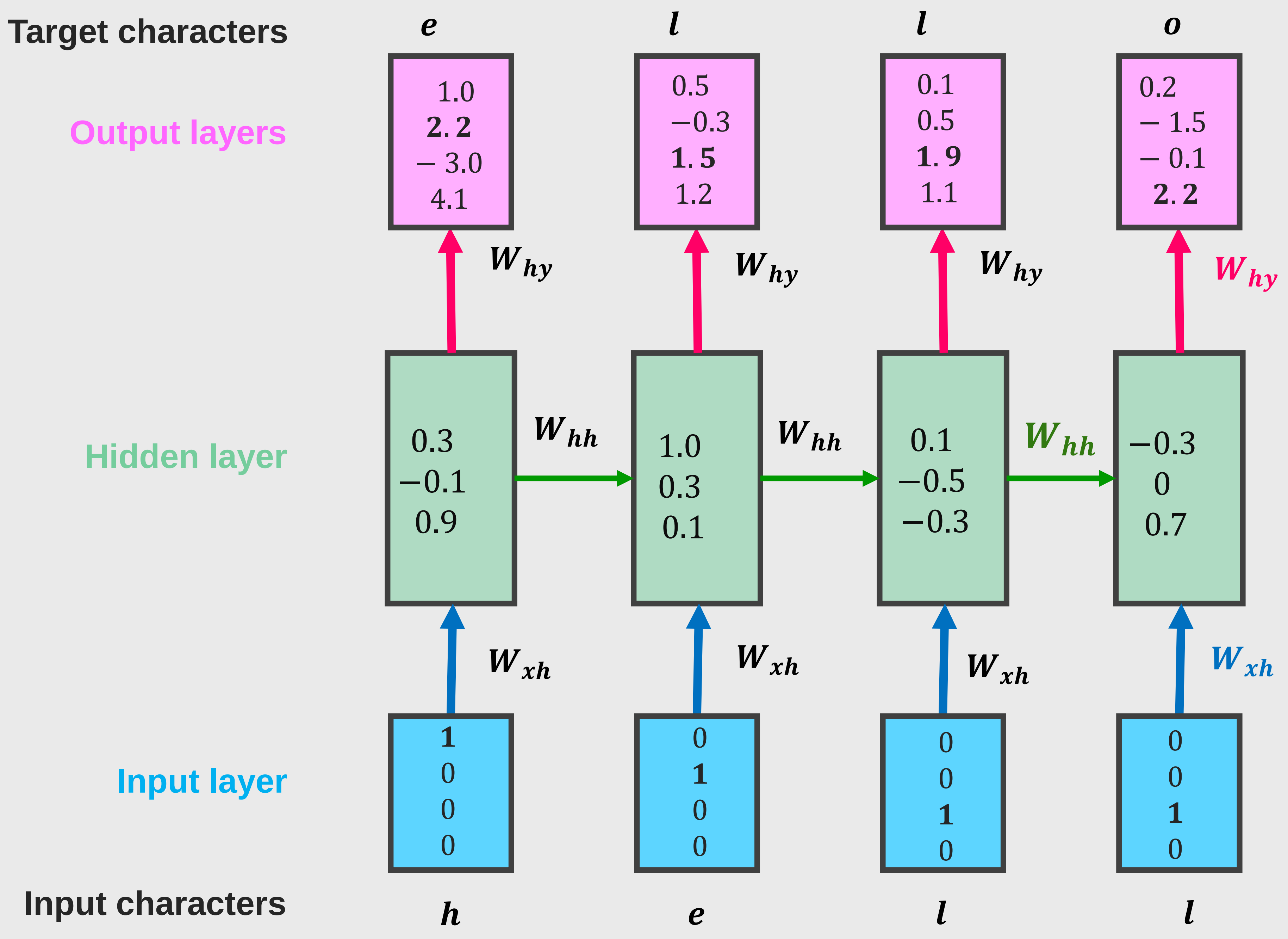
A nice example: http://colah.github.io/posts/2015-08-Understanding-LSTMs

RNN

Architecture Unrolled version for sequential data



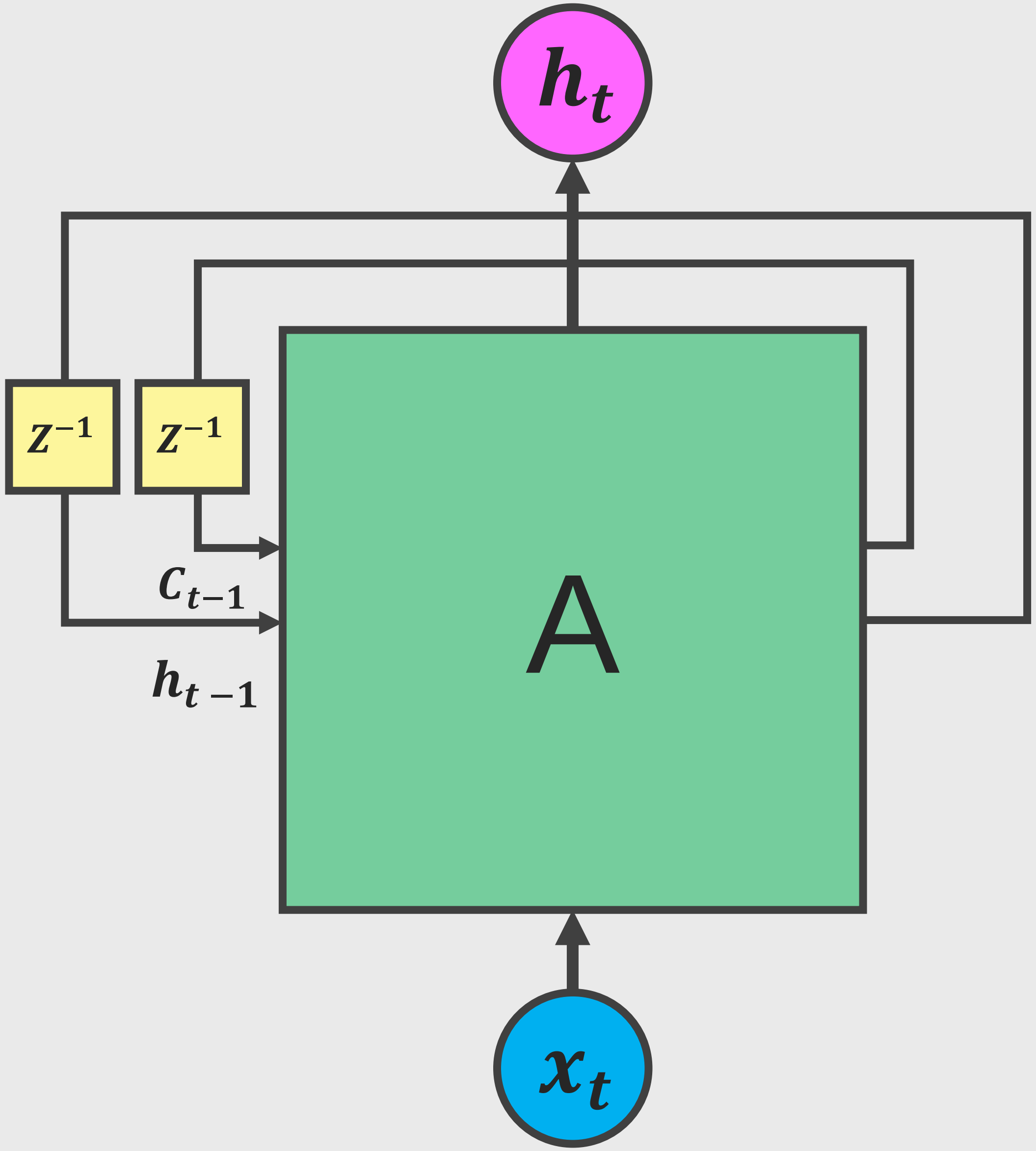
A nice example: http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

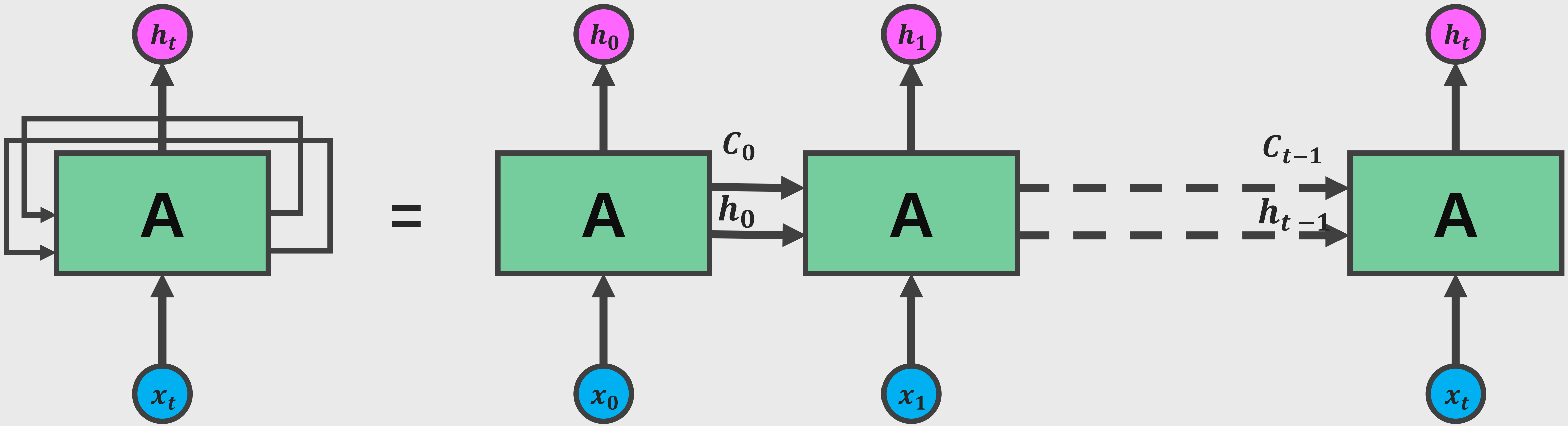
RECURRENT NEURAL NETWORK (RNN)

Architecture



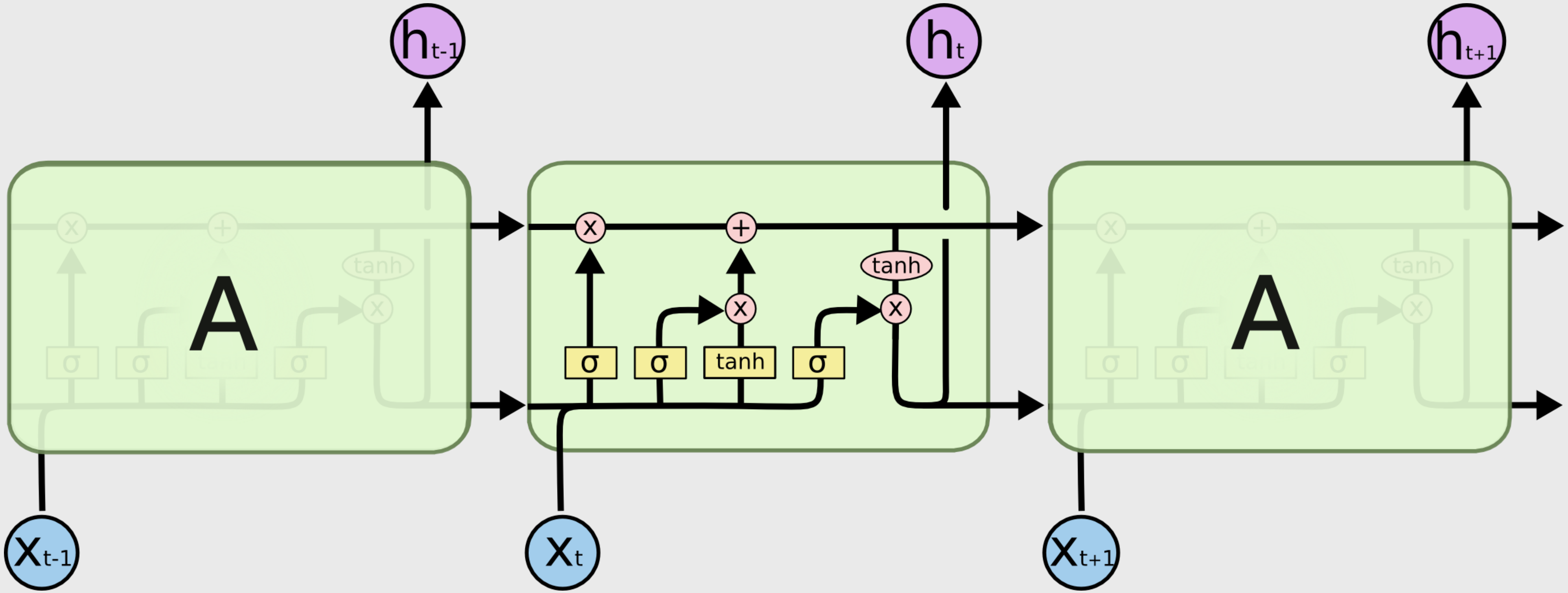
RNN

Architecture Unrolled version for sequential data



A nice example: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Long-Short Term Memory Networks

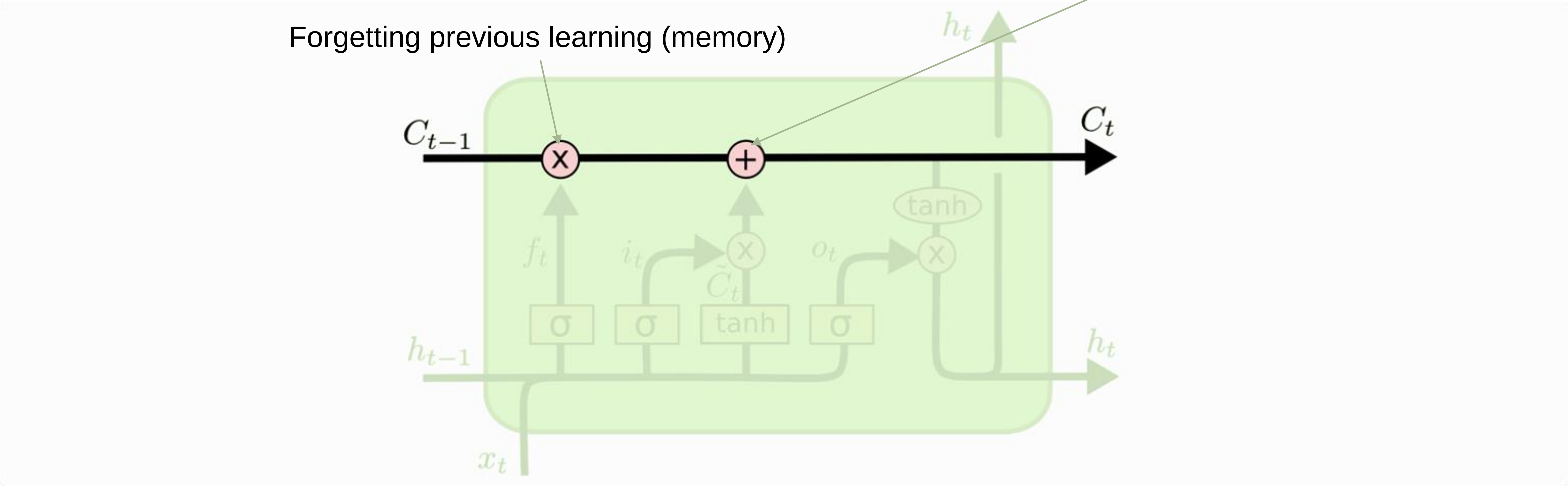


Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

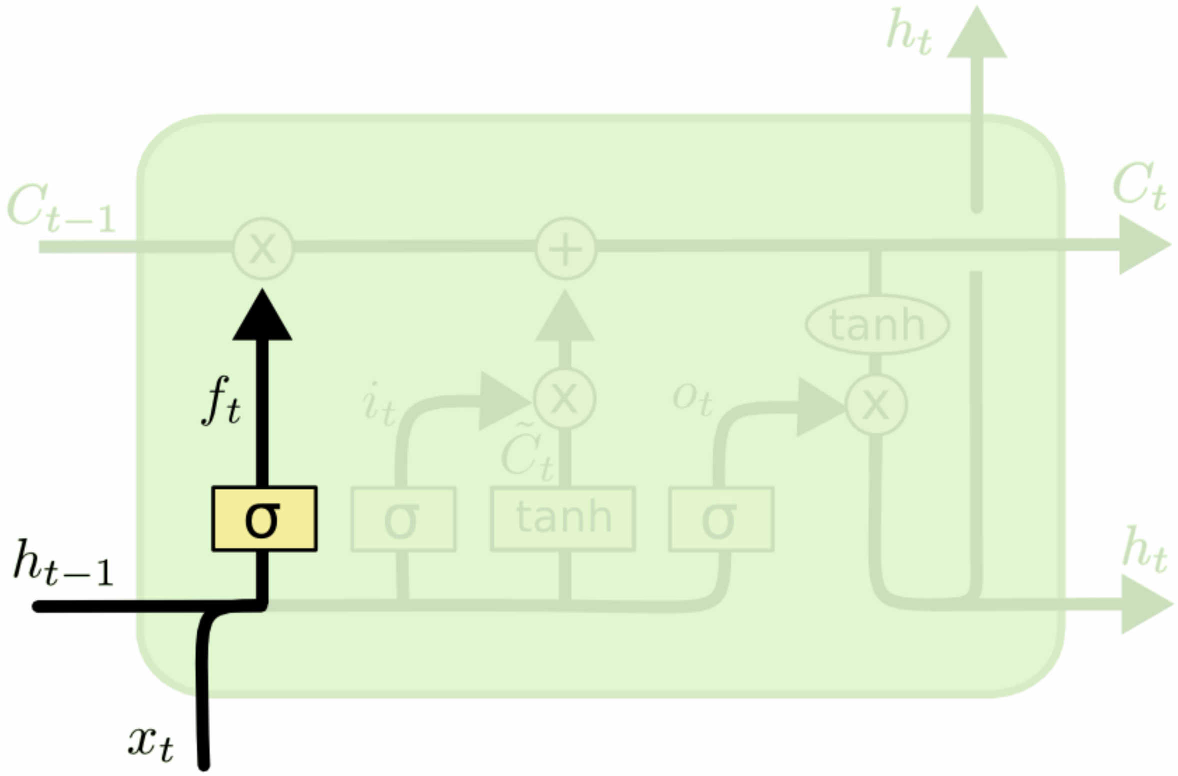
Forgetting and Adding Memory

Adding (updating) previous learning (memory)

Forgetting previous learning (memory)

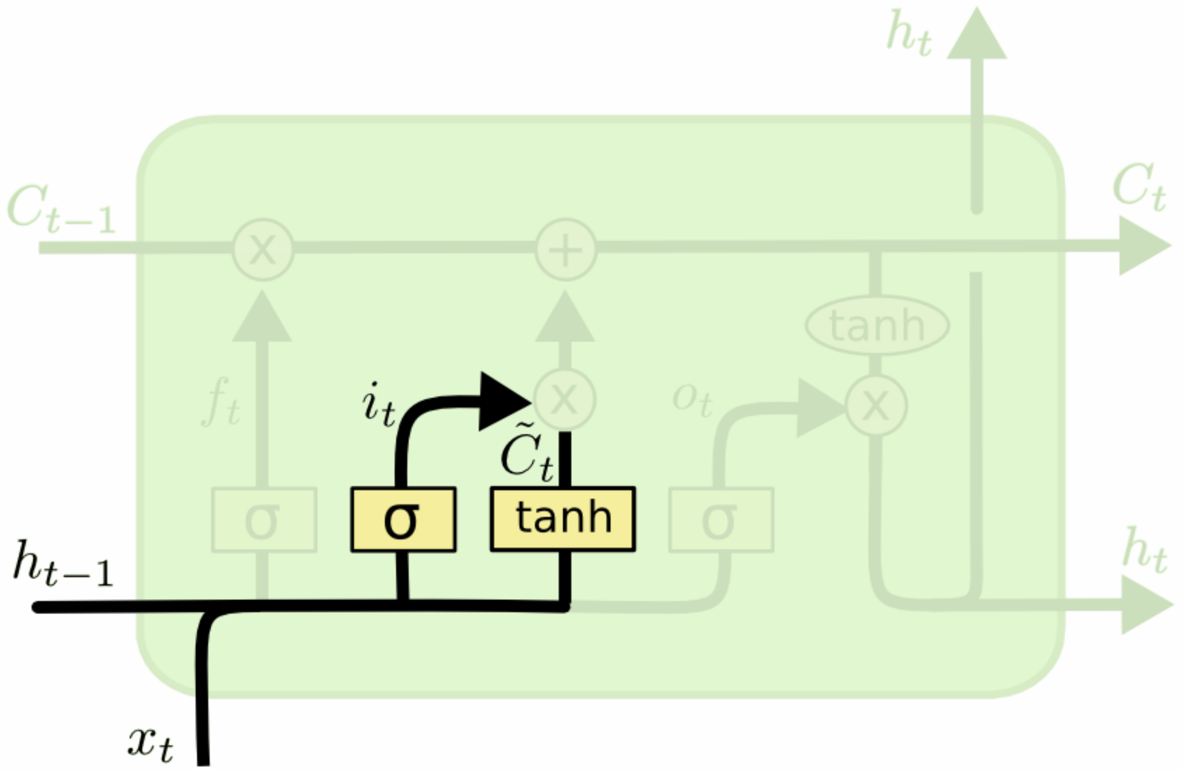


Forgetting Memory



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

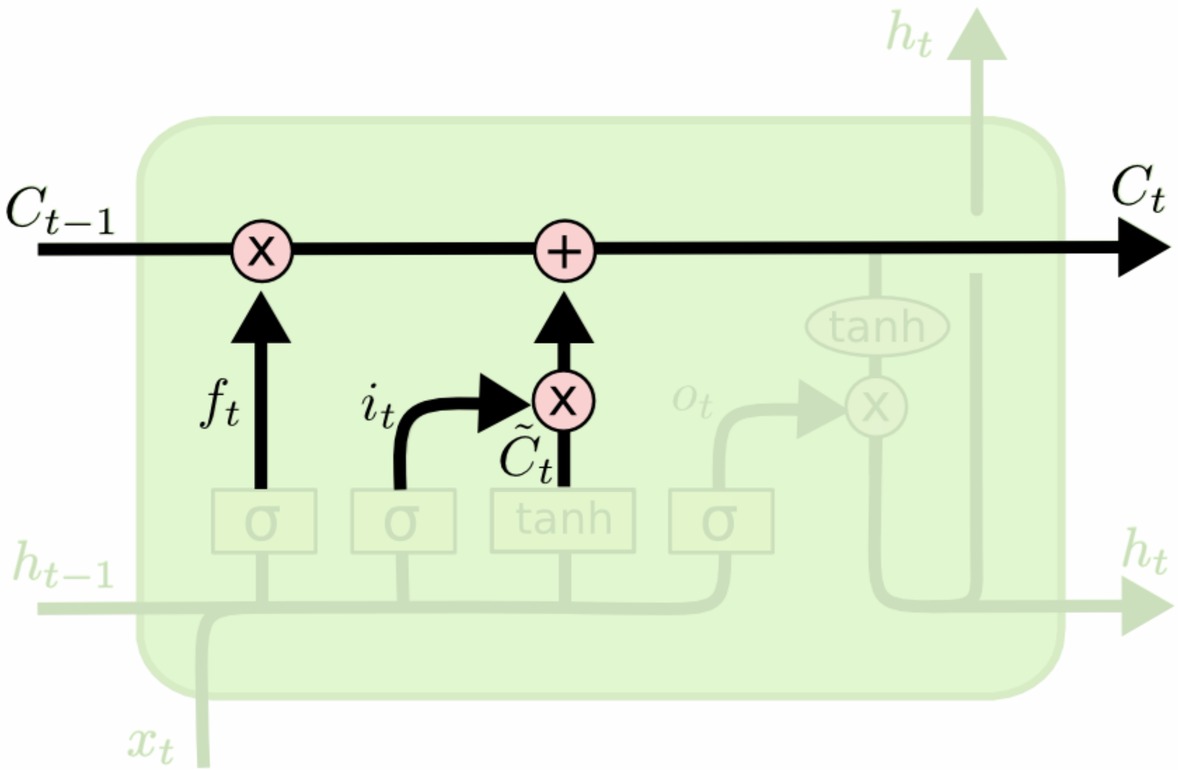
Adding new information to Memory



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

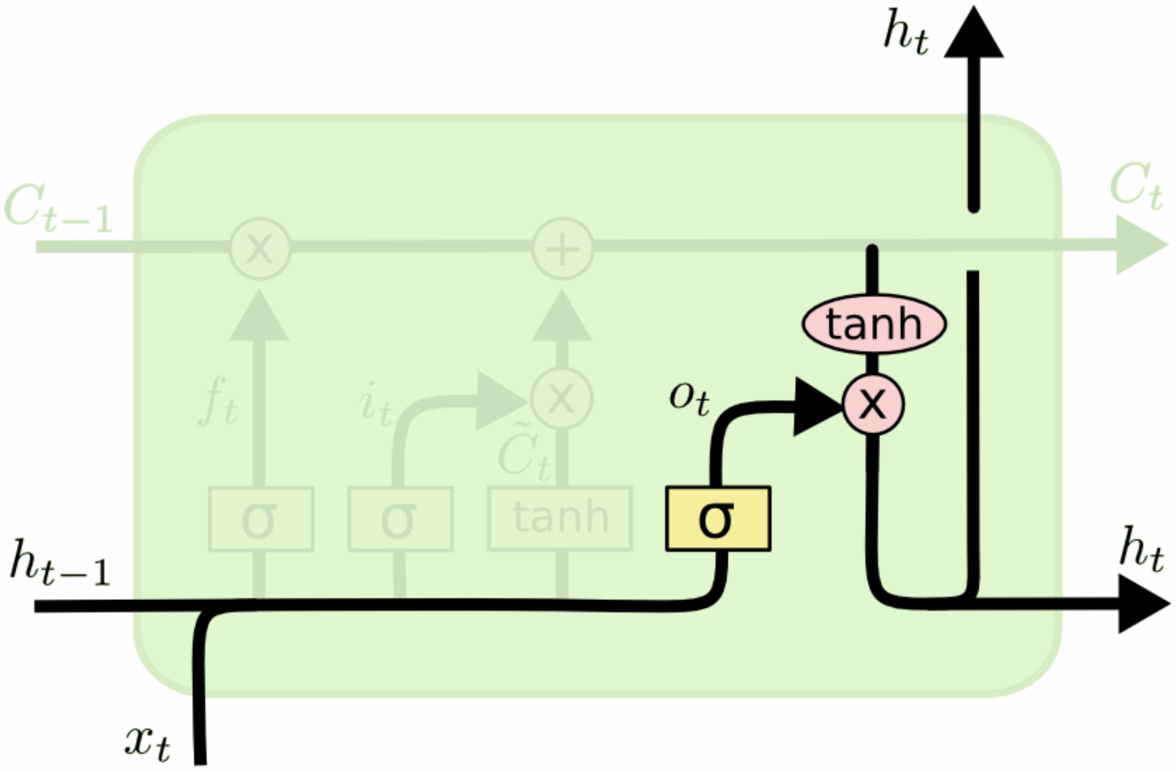
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Updating Memory (for next step) C_t



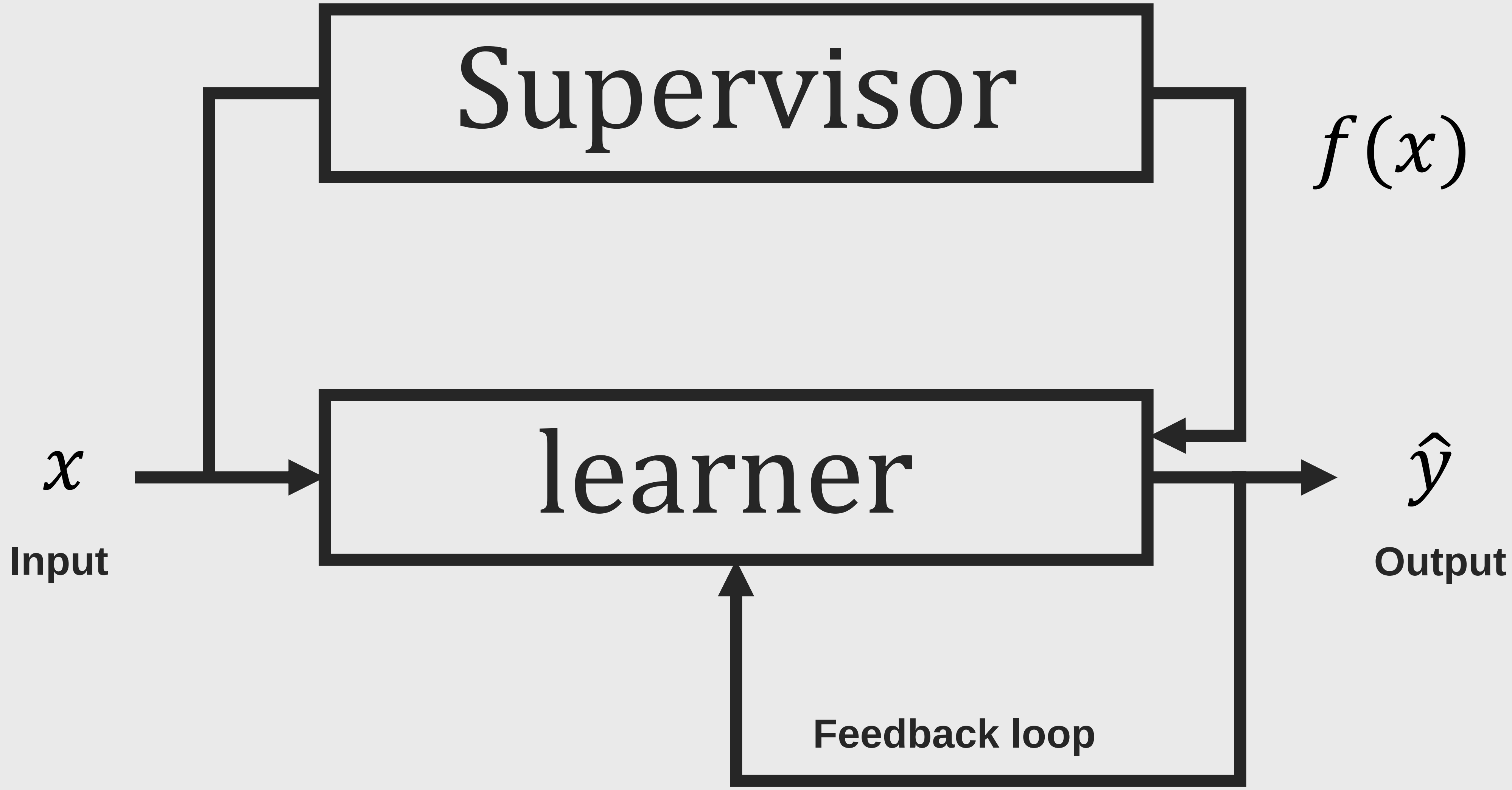
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Producing new outputs (for next step) h_t



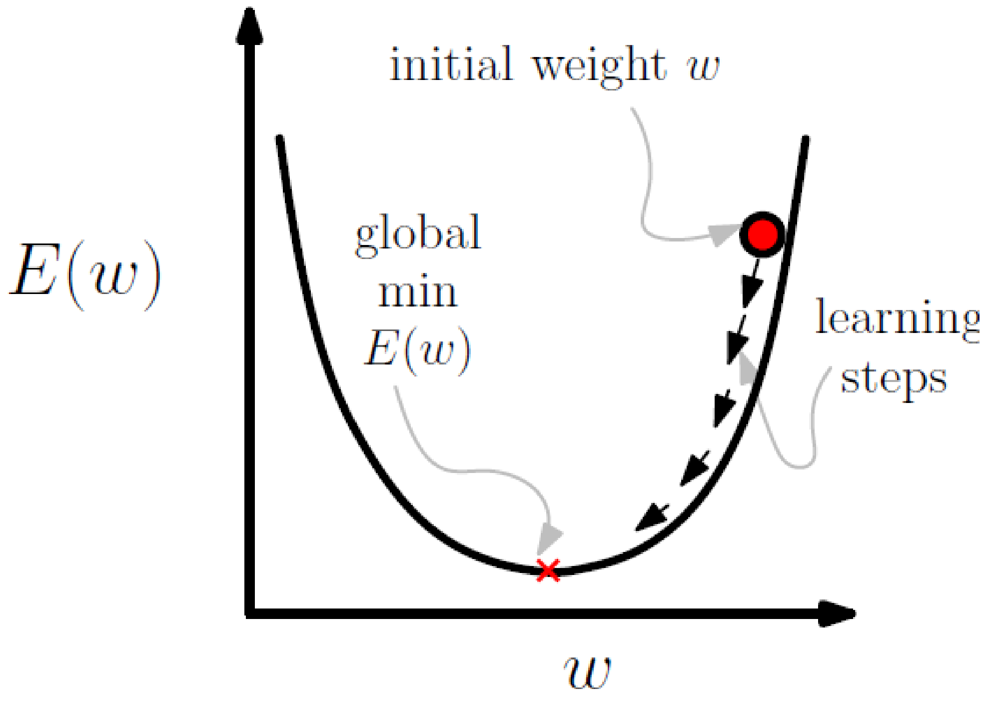
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$


Supervisor
$f(x)$
x
Input
learner
$\hat{y}$
Output
Feedback loop

RNN Optimisation

- Stochastic gradient descent
- Mini-batch gradient descent
- Batch gradient descent
- Backpropagation



Details are in Lecture 5 Slides

Loss function: Cross Entropy loss, $\boldsymbol{E}$

$$E = -\frac{1}{n}\sum_{i=1}^{n} \log(P(x_{i+1} \mid x_i))$$

$P(x_{i+1} \mid x_i)$ predictive probabilities next word x_{i+1} given input x_i

y_i - target output

n - number of examples in training/test set