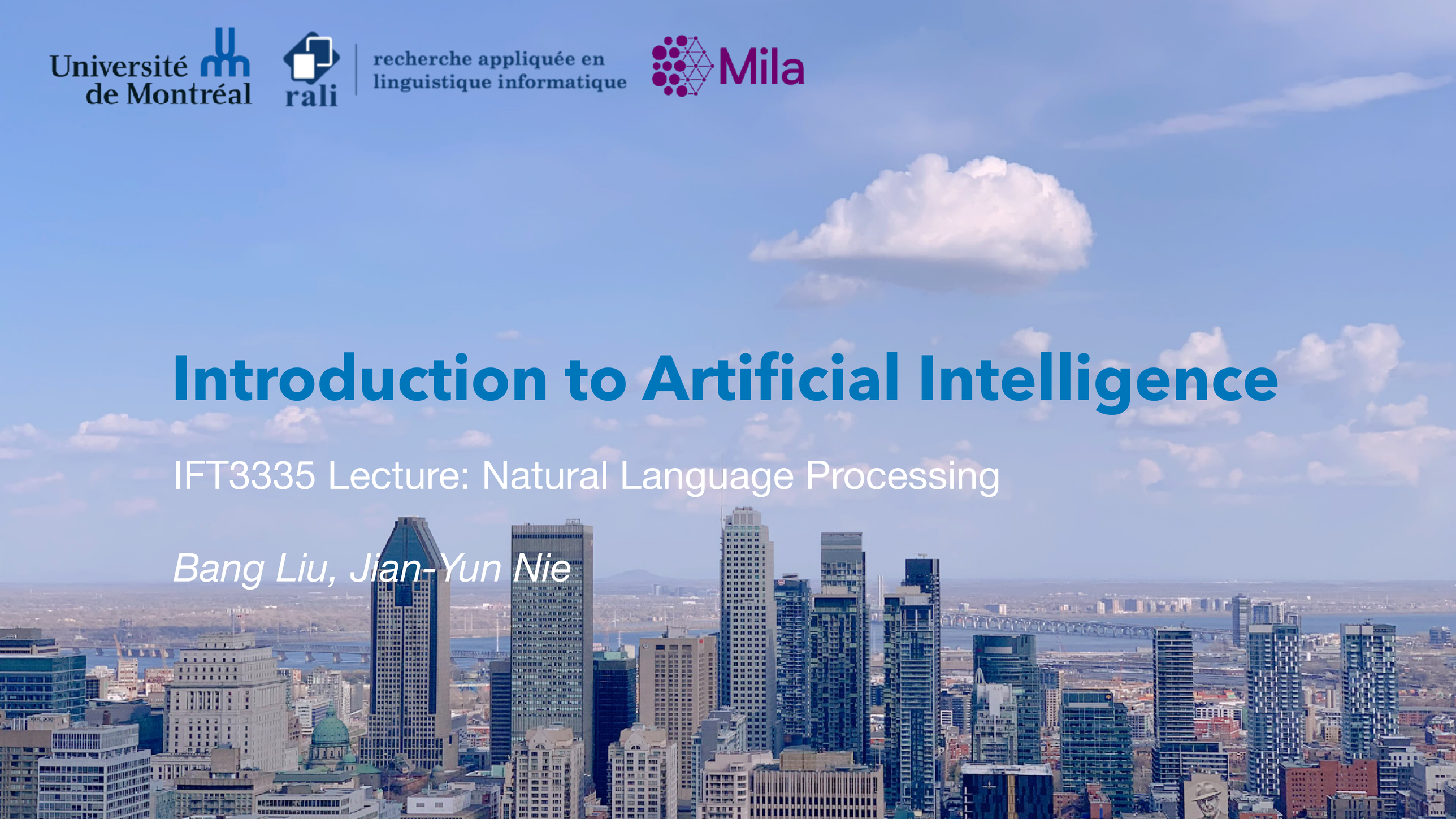


Introduction to Artificial Intelligence

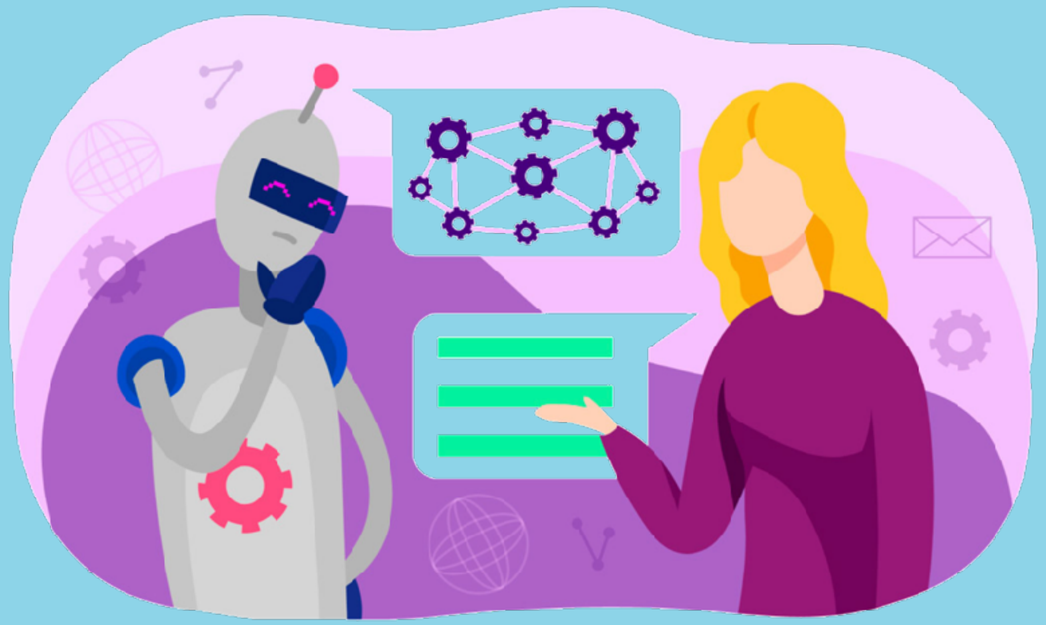
IFT3335 Lecture: Natural Language Processing

Bang Liu, Jian-Yun Nie

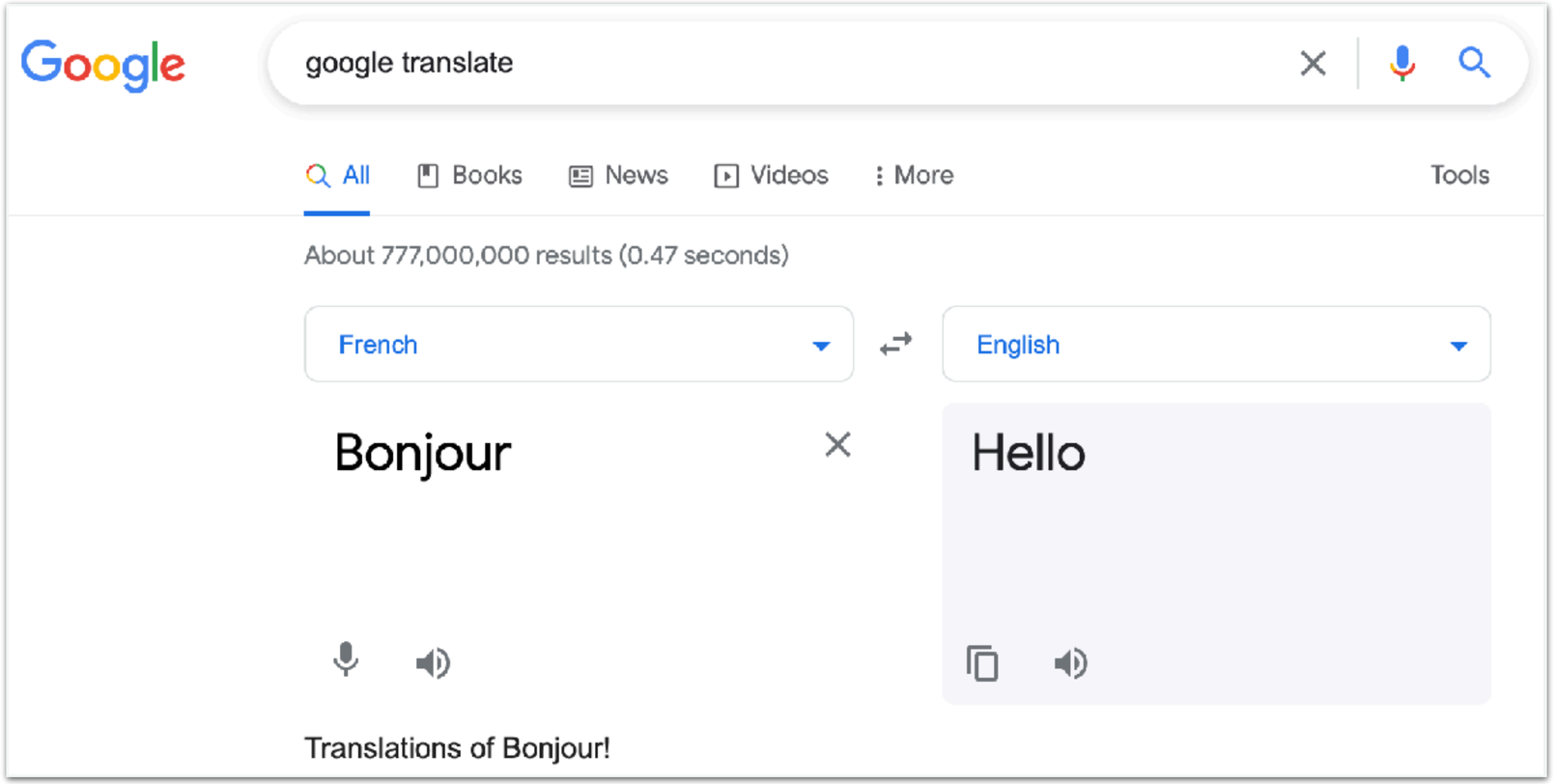
2 Lecture outline

1. What is natural language processing?
2. Why NLP is hard?
3. History of NLP
4. NLP techniques and resources
5. Machine learning and deep learning
6. Training deep neural networks by gradient descent
7. Language modeling
8. Neural language modeling and RNN
9. RNN variants

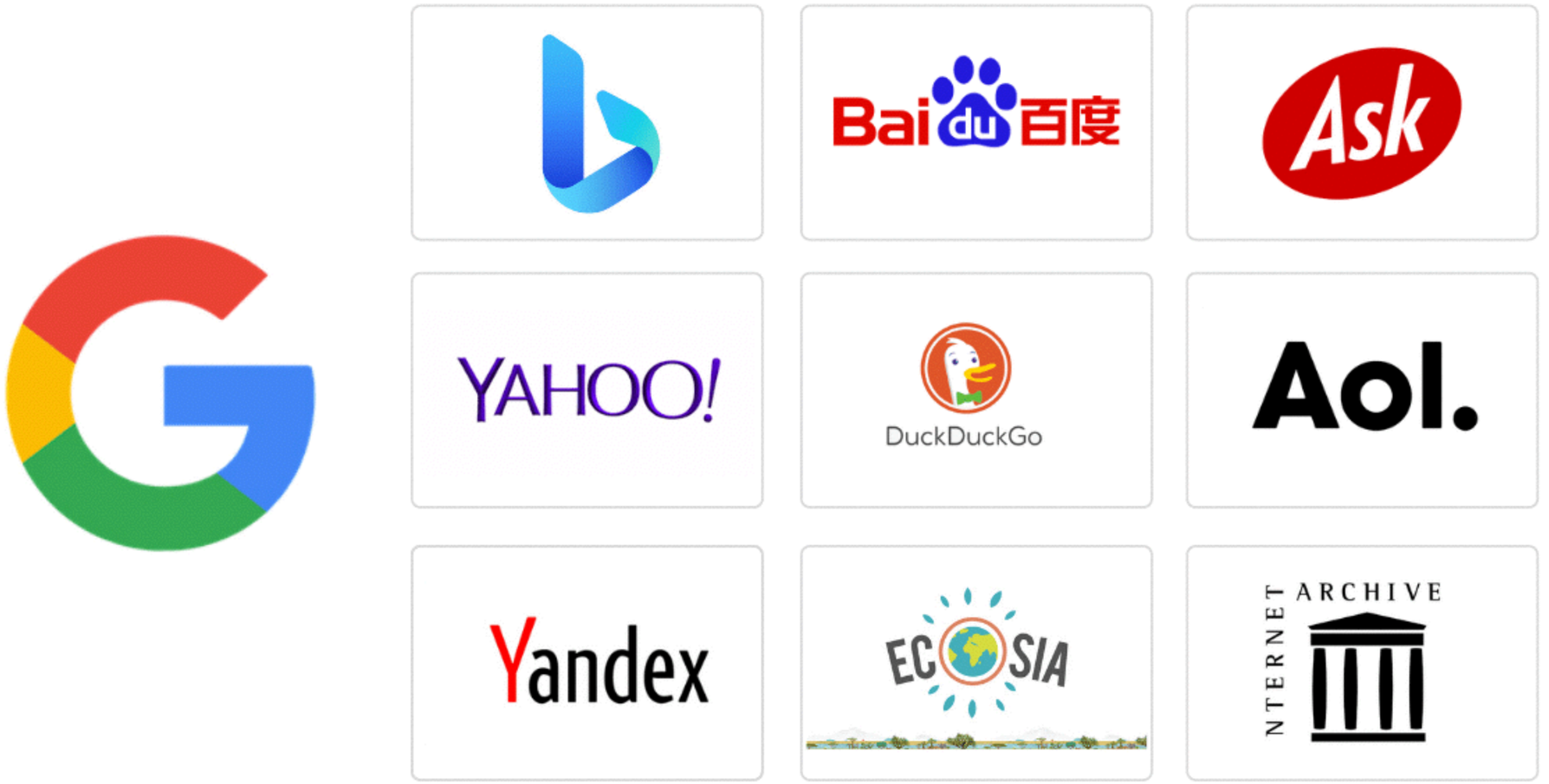
What is natural language processing?



4 NLP is everywhere: machine translation



5 NLP is everywhere: search engines



6 NLP is everywhere: smart speakers

What is natural language processing?

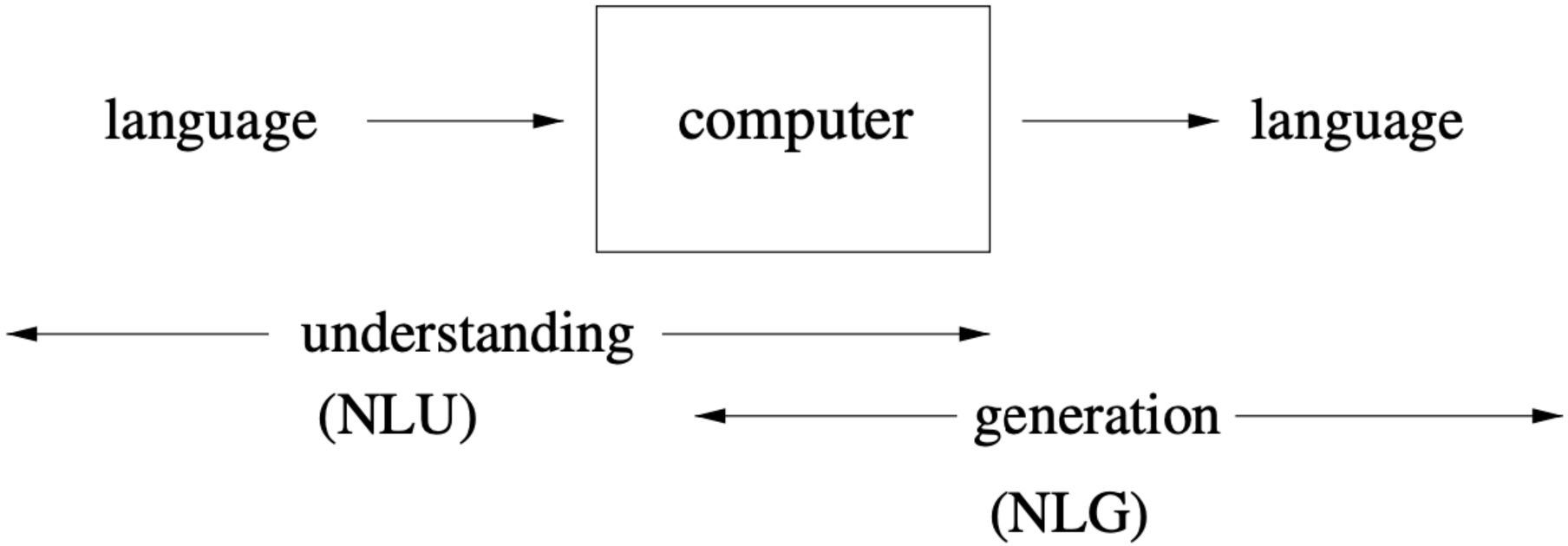
- **Language** is a tool of human communication and a carrier of human thinking.
  - More than 1,900 languages in use all over the world.
  - Structures vary between different languages.

- **Natural language** is a human language (e.g., English, French, Chinese)
  - As opposed to a constructed language (programming languages, language of formal logic, etc.).

Image credit: https://www.vpr.org/post/why-are-there-so-many-different-languages#stream/0

What is natural language processing?

- **Natural language processing (NLP)** is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language.



9 NLP tasks

Lexical Analysis
Word Segmentation
Word Tokenization
New Words Identification
Morphological Analysis
Part-of-speech Tagging
Spelling Correction

Sentence Analysis
Chunking
Super Tagging
Constituency Parsing
Dependency Parsing
Language Modeling
Language Identification

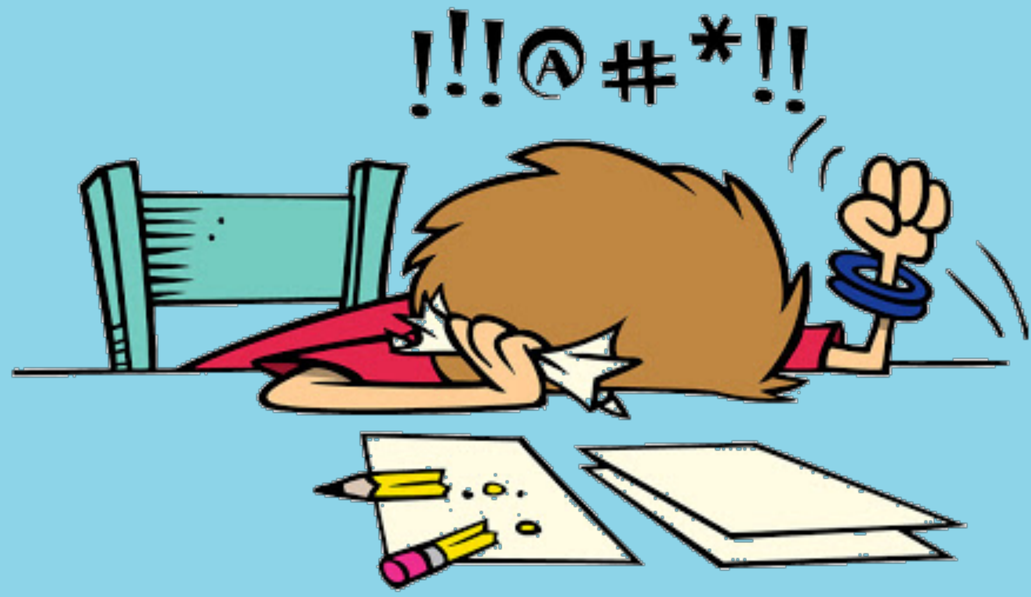
Semantic Analysis
Word Sense Disambiguation
Semantic Role Labeling
Abstract Meaning Representation Parsing
Frame Semantic Parsing
First Order Predicate Calculus
Word/Sentence/Paragraph Vector

Information Extraction
Named Entity Recognition/Disambiguation
Terminology/Glossary Extraction
Coreference Resolution
Relationship Extraction
Event Extraction
Sentiment Analysis

High-level Tasks
Machine Translation
Text Summarization/Simplification
Question Answering
Dialogue System
Reading Comprehension
Automatic Essay Grading

More tasks can be found here: https://nlpprogress.com/

Why NLP is hard?



11 NLP is challenging

- Ambiguity
- Knowledge
- Scale
- Sparsity
- Variation
- Expressivity
- Unknown representations

NLP is challenging: ambiguity

"At last, a computer that
understands you like your mother"

How shall we interpret this sentence?

NLP is challenging: ambiguity

"At last, a computer that
understands you like your mother"

- It understands you as well as your mother understands you
- It understands (that) you like your mother
- It understands you as well as it understands your mother

NLP is challenging: knowledge

We need knowledge:

- Knowledge about language
- Knowledge about the world
- Knowledge about common sense

I dropped the glass on the floor and **it** broke

I dropped the hammer on the glass and **it** broke

15 Case study: GPT-3

Q: How many eyes does a giraffe have?
A: A giraffe has two eyes.

Q: How many eyes does my foot have?
A: Your foot has two eyes.

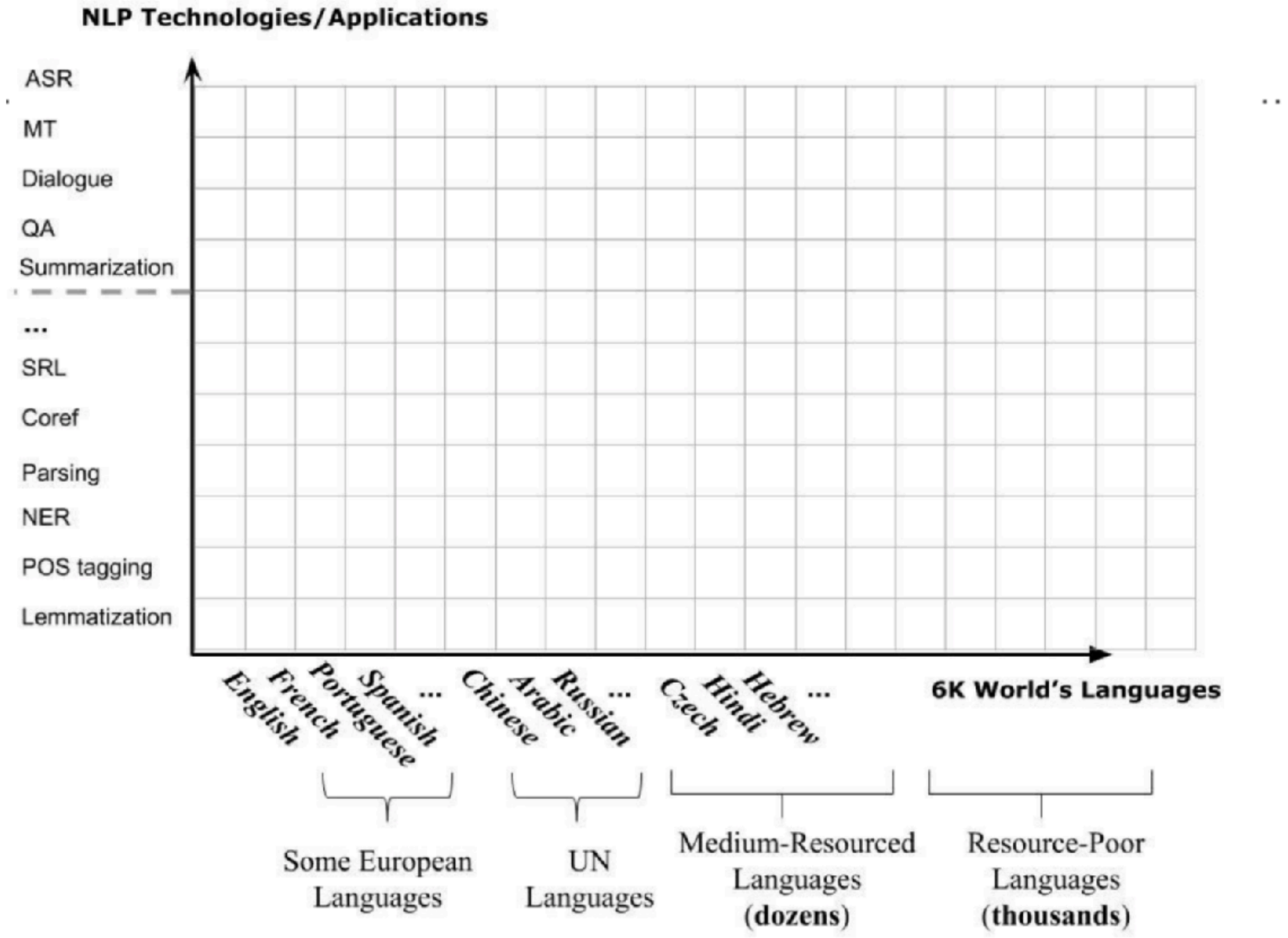
Q: How many eyes does a spider have?
A: A spider has eight eyes.

Q: How many eyes does the sun have?
A: The sun has one eye.

Q: How many eyes does a blade of grass have?
A: A blade of grass has one eye.

Reference: https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html

16 NLP is challenging: scale



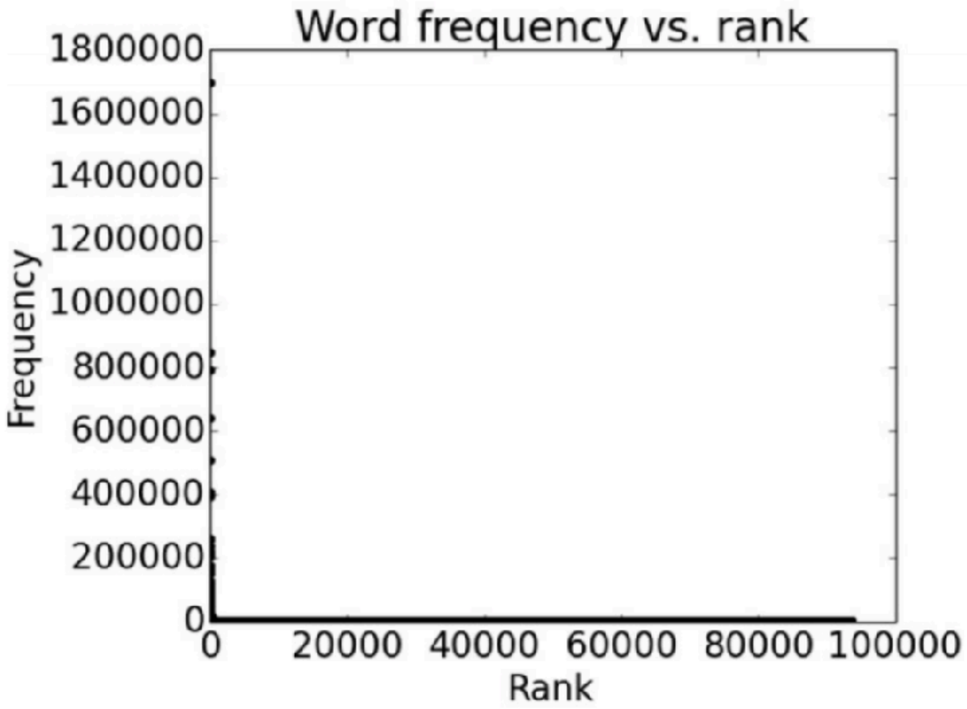
Sparsity

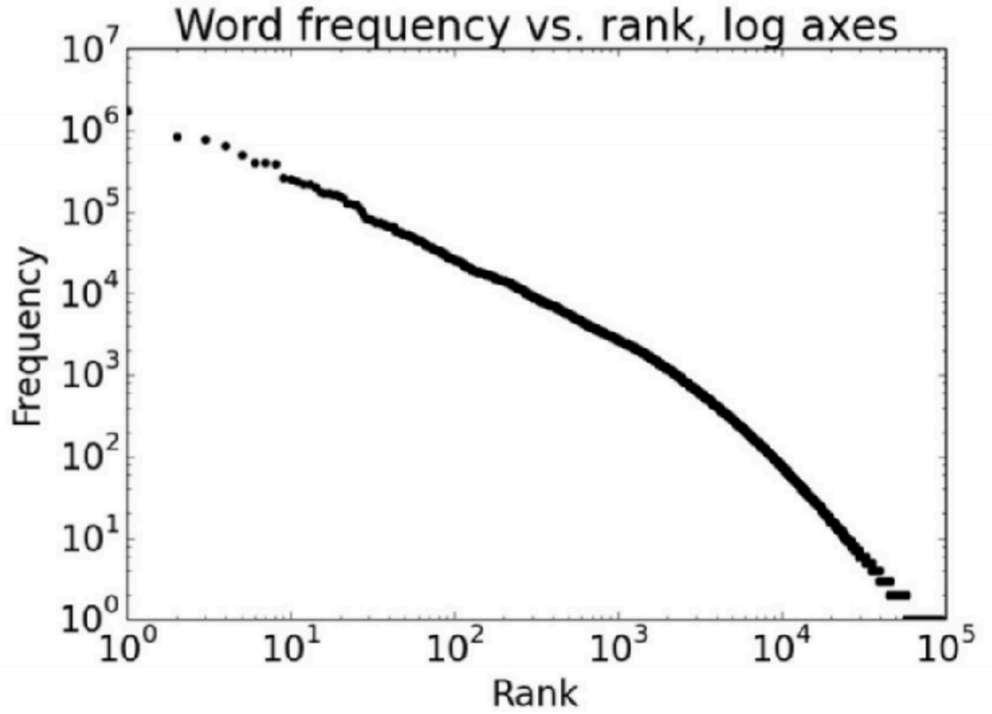
- Sparse data due to **Zipf's law**
- Example: the frequency of different words in a large text corpus

any word		nouns	
Frequency	Token	Frequency	Token
1,698,599	the	124,598	European
849,256	of	104,325	Mr
793,731	to	92,195	Commission
640,257	and	66,781	President
508,560	in	62,867	Parliament
407,638	that	57,804	Union
400,467	is	53,683	report
394,778	a	53,547	Council
263,040	I	45,842	States

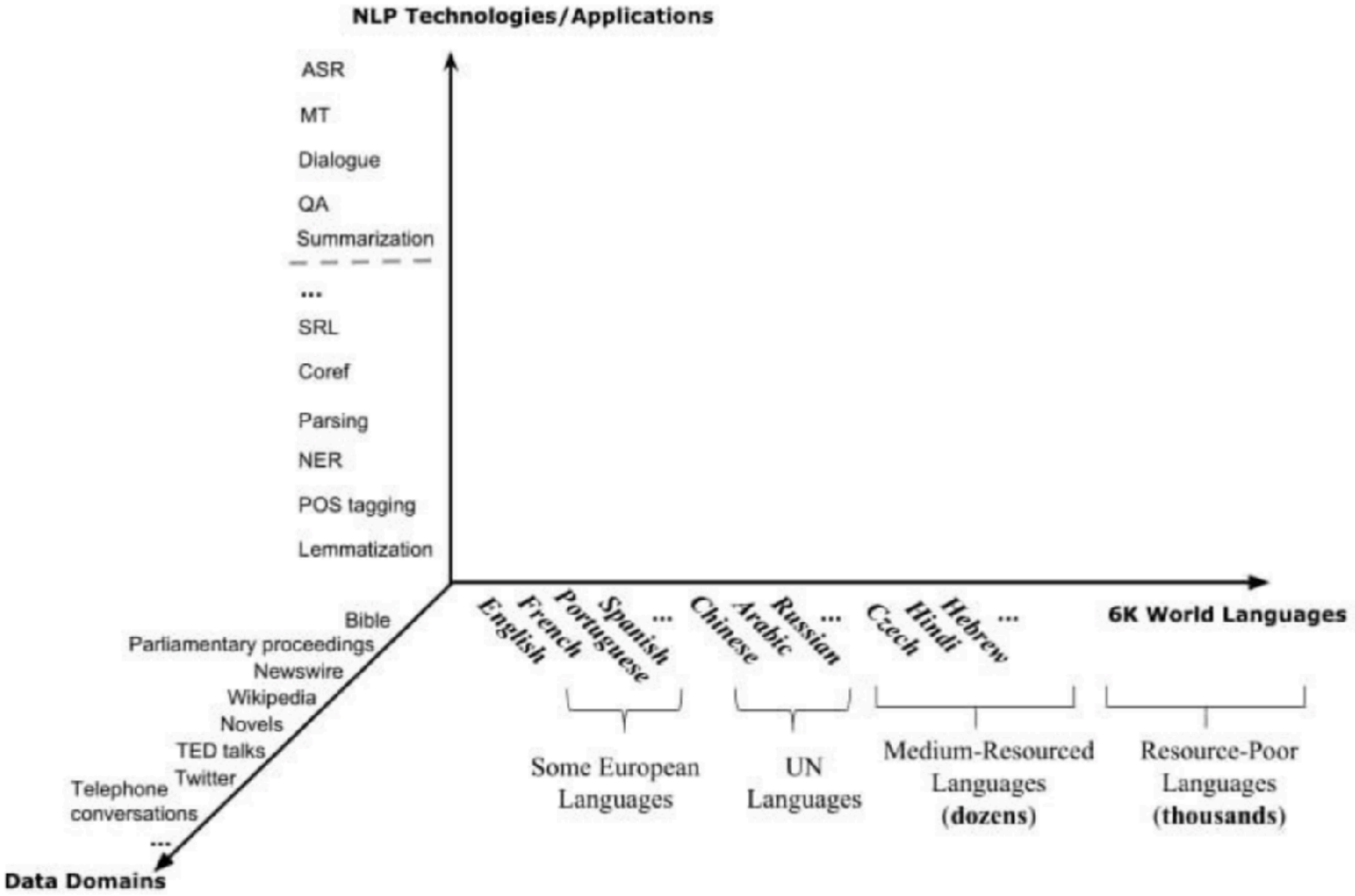
NLP is challenging: sparsity

Order words by frequency. What is the frequency of n-th ranked word?





19 NLP is challenging: variation



NLP is challenging: expressivity

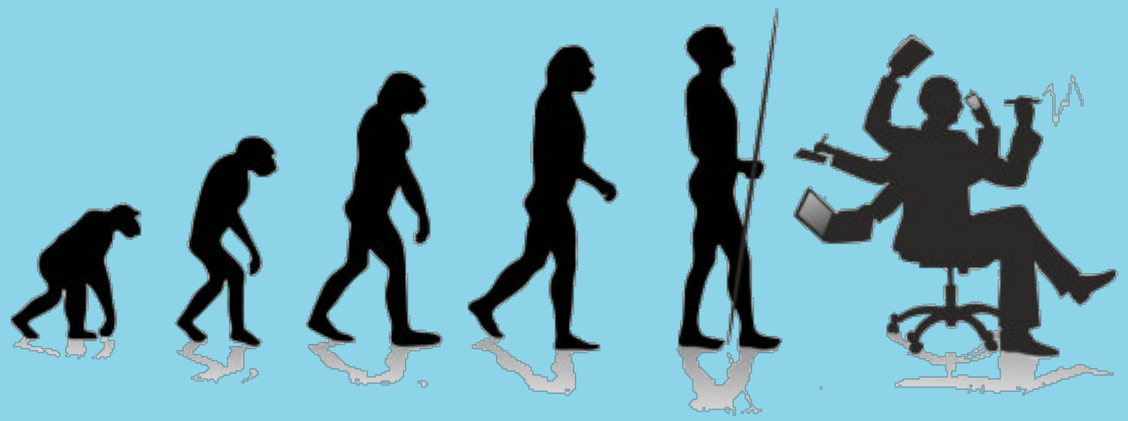
Not only can one form has different meanings (ambiguity), but the same meaning can be expressed with different forms.

NLP is challenging: unmodeled representations

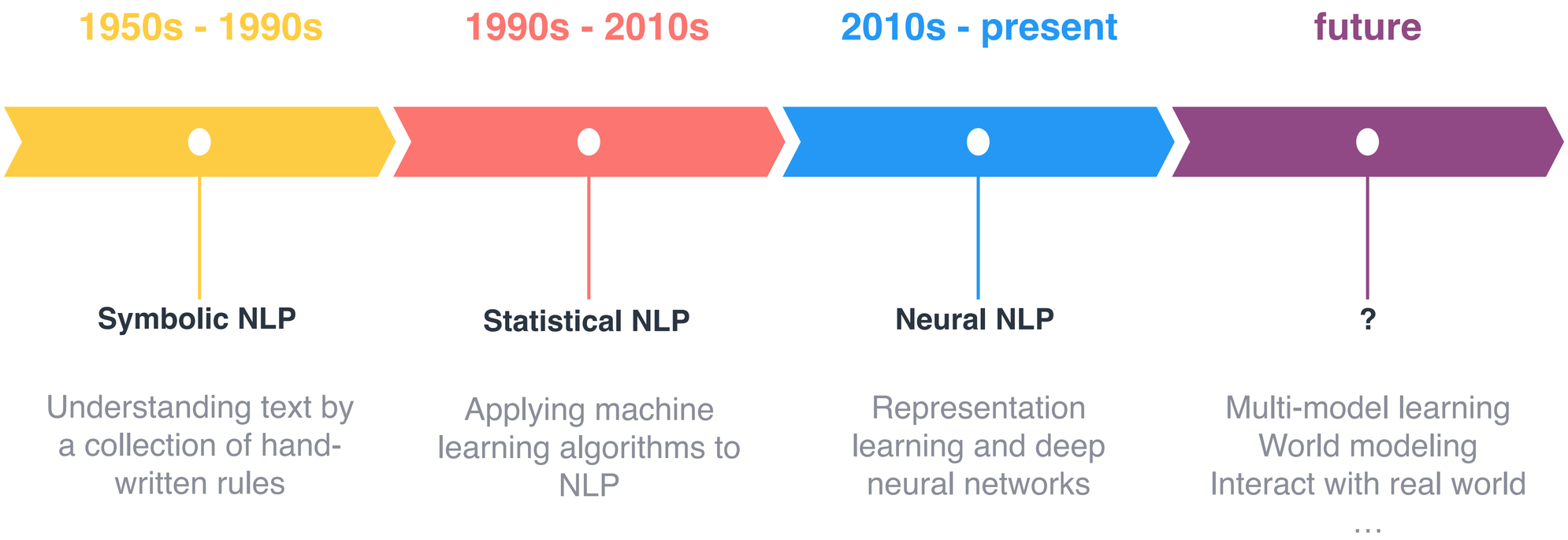
We don’t even know how to represent the knowledge a human has/needs:

- What is the “meaning” of a word or sentence?
- How to model context?
- Other general knowledge?

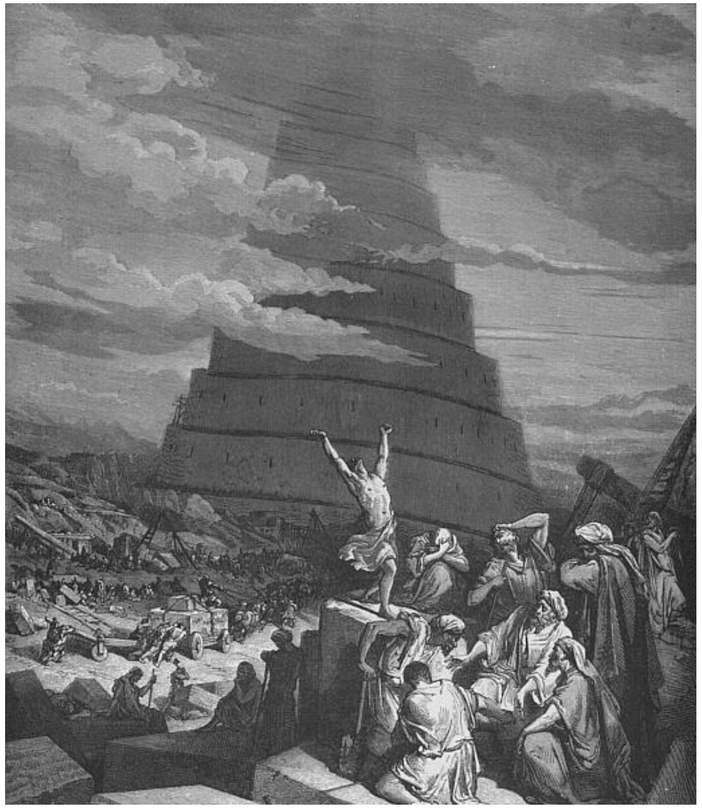
History of NLP



History of NLP



24 The confusion of tongues: Tower of Babel



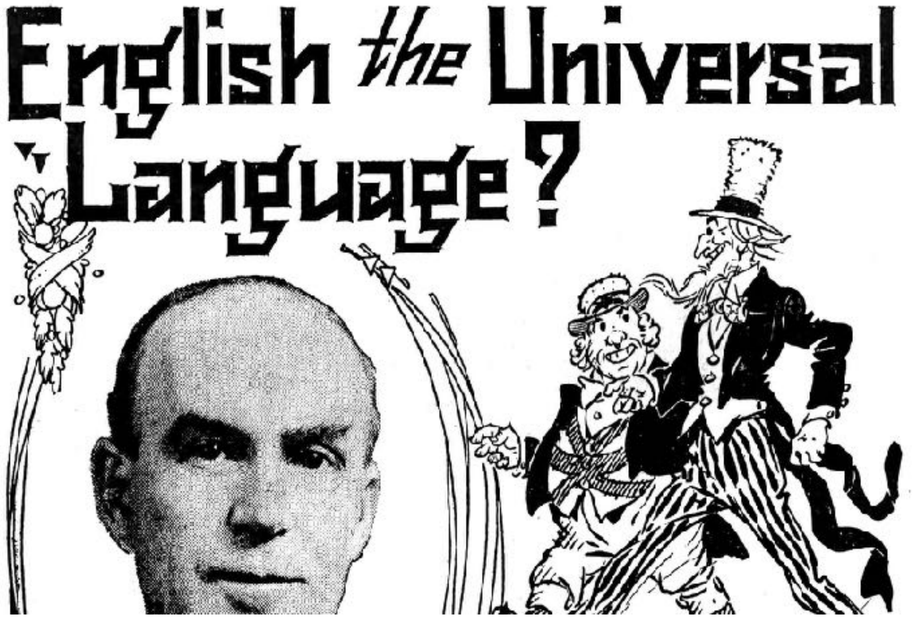
Gustave Doré: The Confusion of Tongues (1865)

The **confusion of tongues** is the origin myth for the fragmentation of human languages described in Genesis.

The Bible "Genesis" says that ancient humans originally spoke a unified language. They once wanted to build a tower called "**Tower of Babel**" as high as the heavens. This shocked God, and God let different people speak different languages so that it becomes impossible for people to coordinate their work. As a result, the Tower of Babel is not built, and the difference in language has become a great obstacle for people to communicate with each other.

25 Universal language

In the 17th century, some insightful people put forward the idea of using **machine dictionaries** to overcome language barriers. Both Descartes and Leibniz tried to write dictionaries based on a unified digital code. In the mid-17th century, such dictionaries were published by Cave Beck, Athanasius Kircher, and Johann Joachim Becher. As a result, a movement about "**universal language**" was launched. Some people tried to create an unambiguous language based on logical principles and graphic symbols, so that people would no longer have to be confused in communication due to misunderstandings.



Language computing have a long history

- In 1847, the Russian mathematician **B. Buljakovski** believed that probabilistic methods could be used to study grammar, etymology, and language history comparison.

- In 1851, the British mathematician **A. De Morgen** used word length as a feature of writing style for statistical research.

- In 1894, Swiss linguist **De Saussure** pointed out that in terms of basic properties, the relationship between quantity and quality in language can be expressed regularly by mathematical formulas. He published "General Linguistics Course" in 1916. It also pointed out that language is like a geometric system, which can be boiled down to some unproven theorems.

Language computing have a long history

- In 1898, German scholar **F.W. Kaeding** calculated the frequency of German vocabulary in the text and compiled the world's first frequency dictionary "German Frequency Dictionary”.

- In 1935, Canadian scholar **E. Varder Beke** proposed the concept of word distribution rate and used it as the main criterion for dictionary selection.

- In 1944, the British mathematician **G.U. Yule** published the book "Statistical Analysis of Literary Words", using probabilistic and statistical methods to study vocabulary.

Four important milestones to NLP

Andrey Markov

Research on Markov model

Alan Turing

Research on model of computation

Claude Shannon

Research on probability and information theory

Noam Chomsky

Research on formal language theory

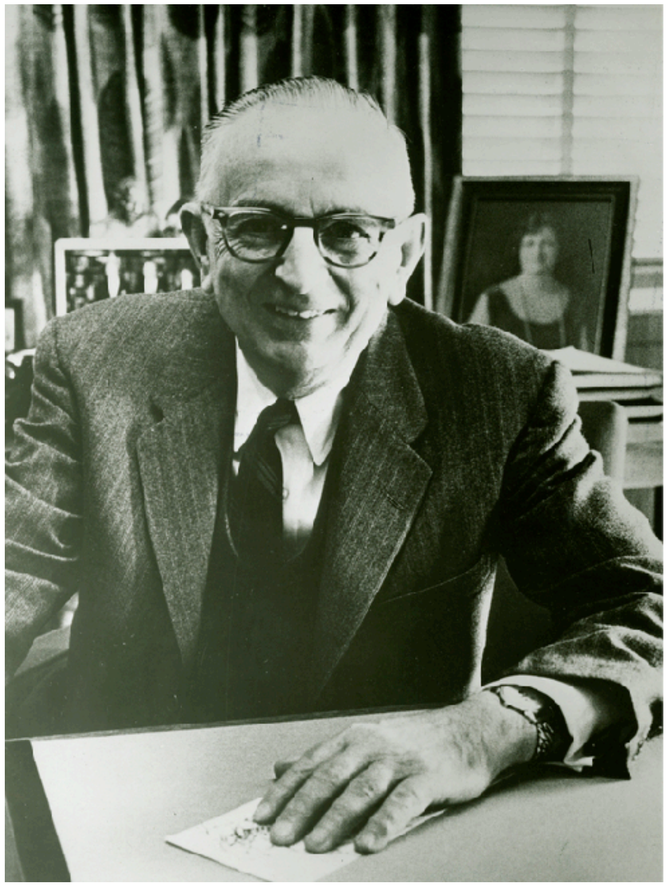
Machine translation: early development

Machine Translation is the key application in the history of computational linguistics.

- In 1933, the Soviet inventor Troyansky **(П.П.ТРОЯНСКИЙ) designed a machine** that mechanically **translates** one language into another, and registered his invention on September 5 of the same year.
- In the early 1930s, the Armenian French engineer **GB Artsouni** proposed the idea of **using machines for language translation**, and on July 22, 1933, he obtained a patent for a "translator" called " **Mechanical brain**" (mechanical brain). The prototype of Artsouni was formally exhibited in 1937, which aroused the interest of the French postal and telecommunications departments. However, due to the outbreak of World War II soon, Artsouni's robotic brain could not be installed and used.

Machine translation: a decoding perspective

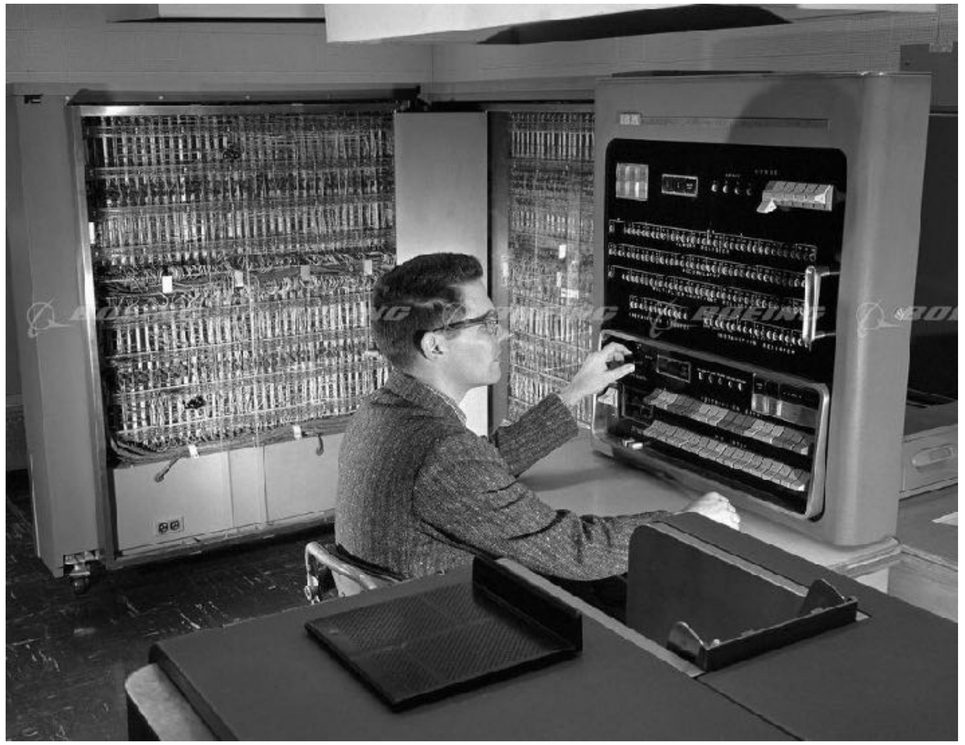
- British engineer Andrew Donald Booth and American Rockefeller Foundation (Rockefeller Foundation) Deputy President W. Weaver put forward the idea of **machine translation.** In 1949, Weaver published a memo titled "Translation", which formally raised the issue of machine translation. He said: "When I read an article written in Chinese, I can say that this article is actually written in English, but it is coded with another strange symbol. When I was reading, I was decoding. Weaver's excellent ideas became the theoretical basis of Statistic Machine Translation (SMT).
- However, translation is more complex than decoding: the complexity of lexical analysis, syntactic analysis, semantic analysis, etc., was largely omitted.



Warren Weaver

Machine translation: the first MT system

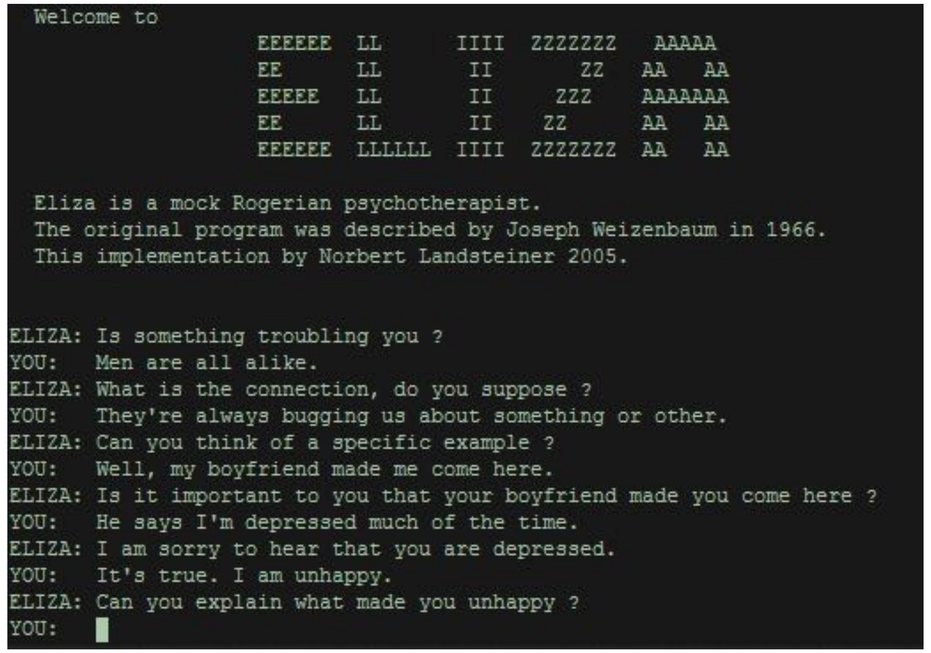
- The academic community in the United States and the United Kingdom have developed a keen interest in machine translation and have received support from the industry.
- In 1954, Georgetown University, with the assistance of IBM, used the IBM-701 computer to implement **the world's first MT system**, realizing Russian-English translation, and the system was publicly demonstrated in New York in January 1954
- In the following 10 years, with the progress of machine translation research, various natural language processing technologies emerged and gradually developed, forming this emerging discipline combining linguistics and computer technology.



IBM-701

AI winter

- In 1966, the **rule-based dialogue robot ELIZA** was born in the MIT Artificial Intelligence Laboratory
- However, in the same year, **ALPAC** (Automatic Language Processing Advisory Committee) proposed in a **report** that the progress of **machine translation research in the past ten years was slow** and did not meet expectations. After the release of the report, research funding for machine translation and natural language has been greatly reduced, and research on natural language processing and artificial intelligence has entered a **winter**.



Rule-based dialogue system ELIZA

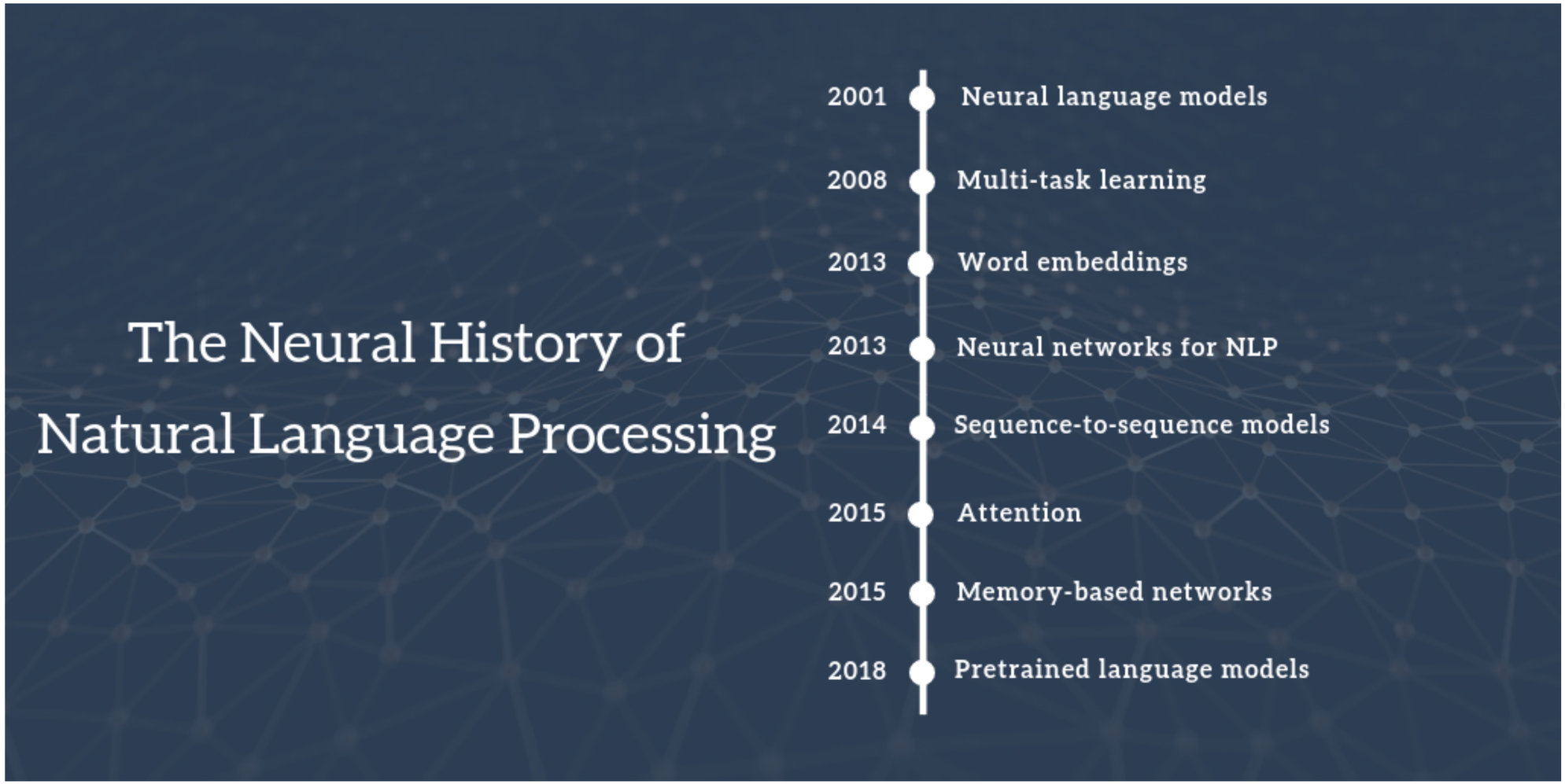
New era of NLP

- Researchers from IBM's Watson Research Center and Baker from Carnegie Mellon University have achieved **success in the development of statistical speech recognition algorithms**: "Hidden Markov Model" and "Noisy Channel Model and Decoding Model".
- At the 4th MT Summit IV held in Kobe, Japan in July 1993, the famous British scholar J. Hutchins pointed out in his special report that **the development of machine translation has entered a new era**. With the beginning of a new era of machine translation, computational linguistics has entered its boom period.


Deep Learning

The neural history of NLP

Deep learning has brought great impacts to NLP



https://ruder.io/a-review-of-the-recent-history-of-nlp/

OpenAI GPT-3

GPT-3, an **autoregressive language model** with **175 billion parameters**, can "generate news articles which human evaluators have difficulty distinguishing from articles written by humans"

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

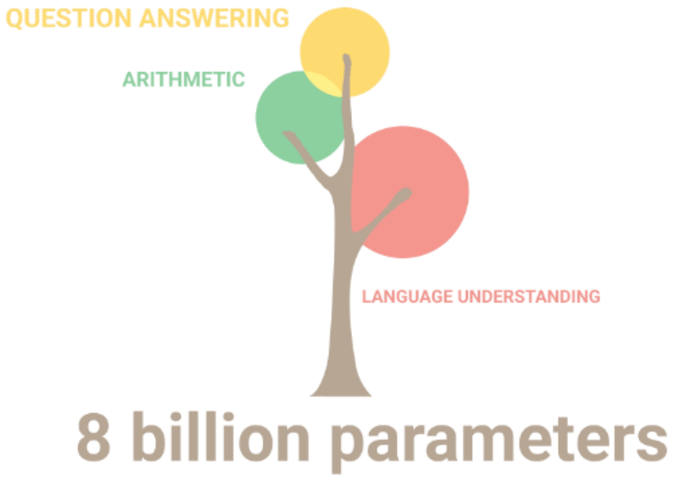
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Pathways Language Model (PaLM)

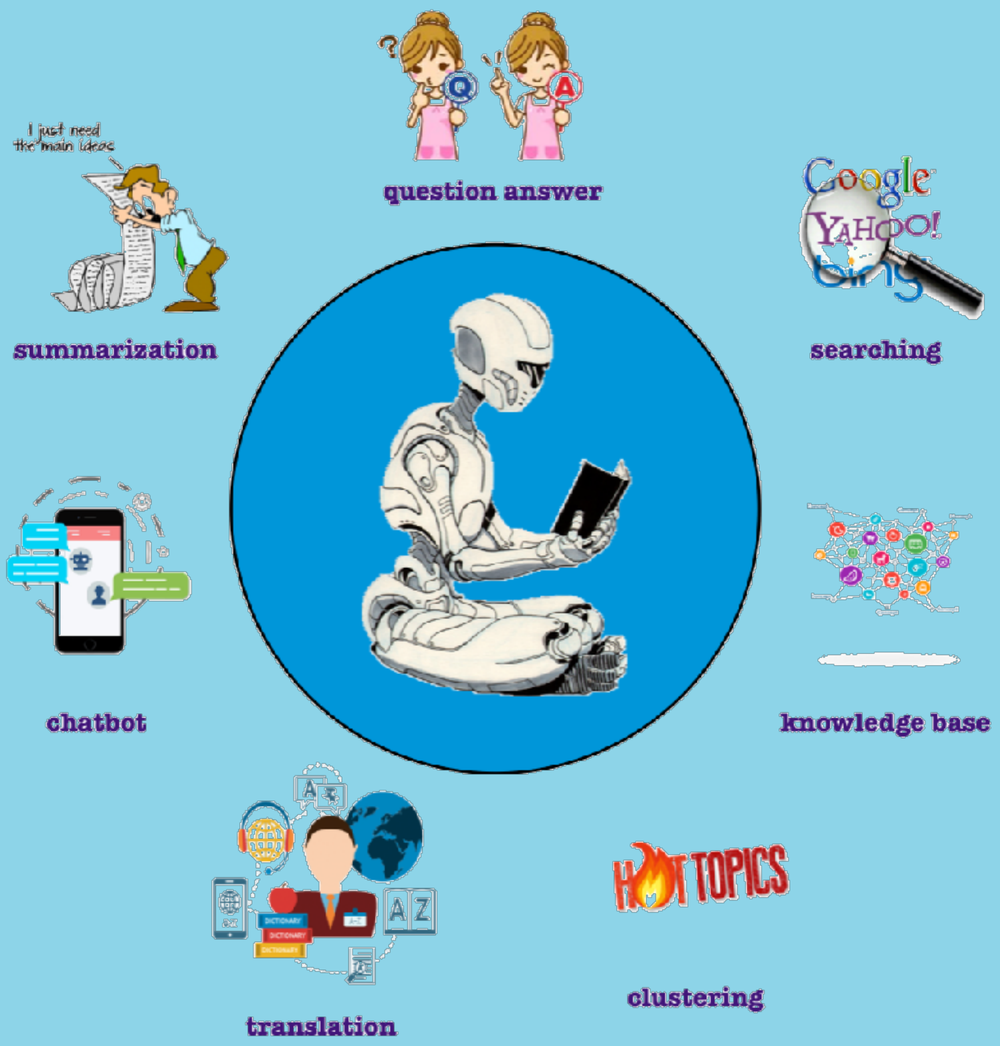
- Last year Google Research announced Pathways, a single model that could generalize across domains and tasks while being highly efficient.

- In “PaLM: Scaling Language Modeling with Pathways”, Google introduces the Pathways Language Model (PaLM), a 540-billion parameter, dense decoder-only Transformer model that achieves state-of-the-art few-shot performance across most tasks, by significant margins in many cases.trained with the Pathways system

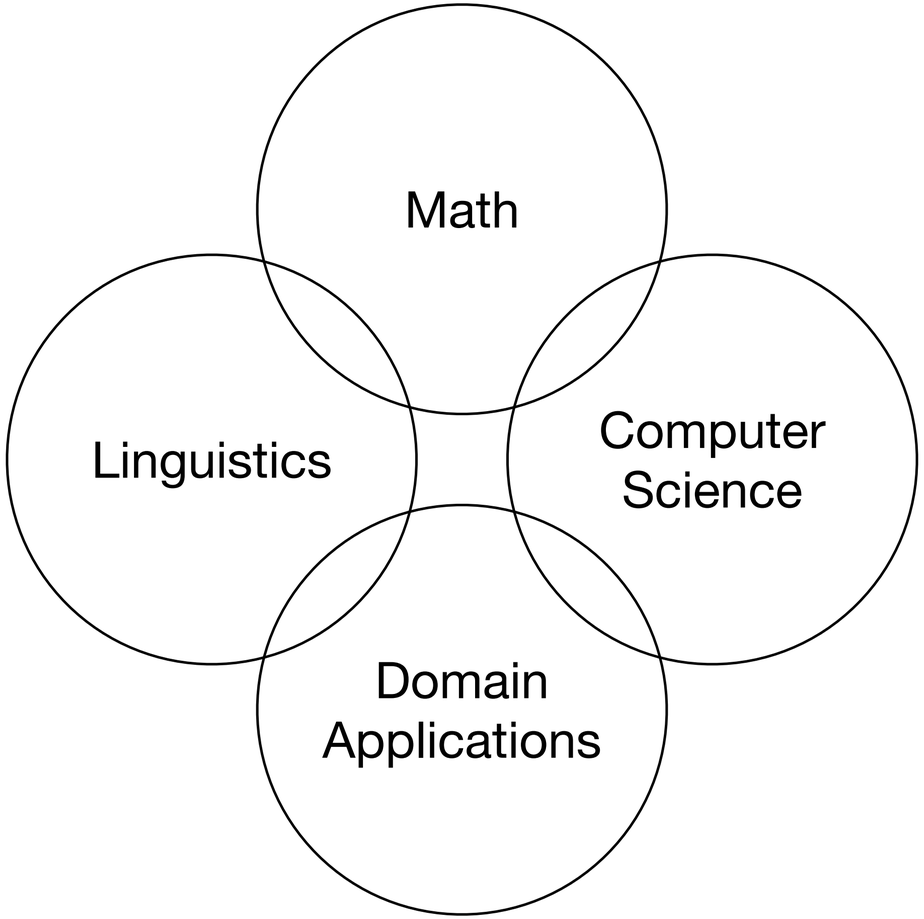


https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

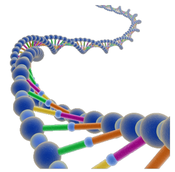
NLP techniques and resources



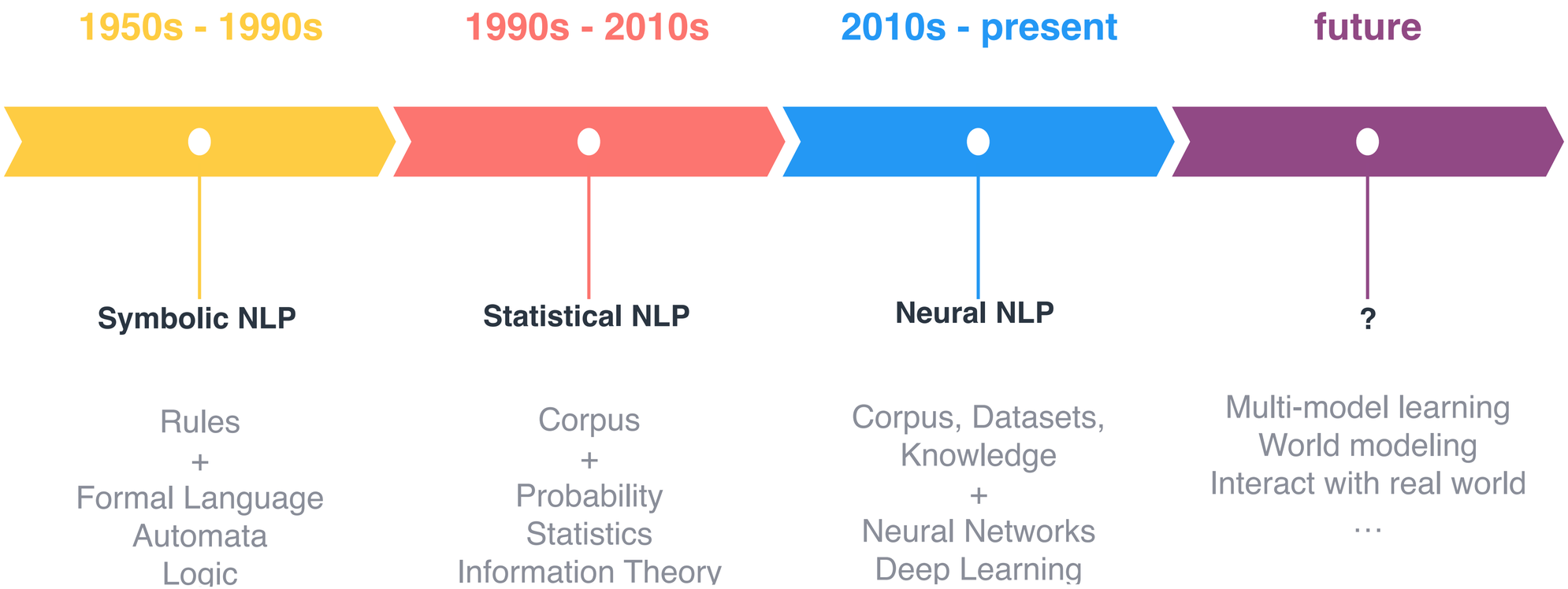
NLP



NLP Building

AI+X Applications	 IT	 Finance	 Biology	 Education	 Law	...
Core Tasks	Text Clustering	 Text Classification	 Text Matching	 Machine Translation	 Information Retrieval	...
	 Summarization	 Information Extraction	 Reading Comprehension	 Question Answer	 Dialog System	...
Basic Techniques	Lexical Analysis	Syntactic Analysis	Semantic Analysis	Discourse Analysis	Pragmatic Analysis	...
Fundamental Theory	Formal Language Automata Logic	Probability Statistics Information Theory	Machine Learning Optimization Matrix Analysis	Neural Networks Deep Learning Graph Analysis	More math ...	...
Data	Rules	Corpus	Datasets	Knowledge	More data, modularity, environment ...	...

Different stages of NLP



Grammatical hierarchy

- **Sentence**: a sentence may consists of one or multiple clauses.
- **Clause**: a clause is a part of the sentence that contains a verb
- **Phrase**: a phrase is a group of words that express a concept and is used as a unit within a sentence, e.g., "Who ate **the last sandwich**?"
- **Word**: A word may consist of a root morpheme only, e.g. science, or a root morpheme plus other morphemes, e.g. "released = release + ed", "motivation = motivate + ion"
- **Morpheme**: morphemes are parts of words and are the smallest grammatical units, e.g., "ed", "ion", and simple words.

Sentences

consist of one or more

Clauses

consist of one or more

Phrases

consist of one or more

Words

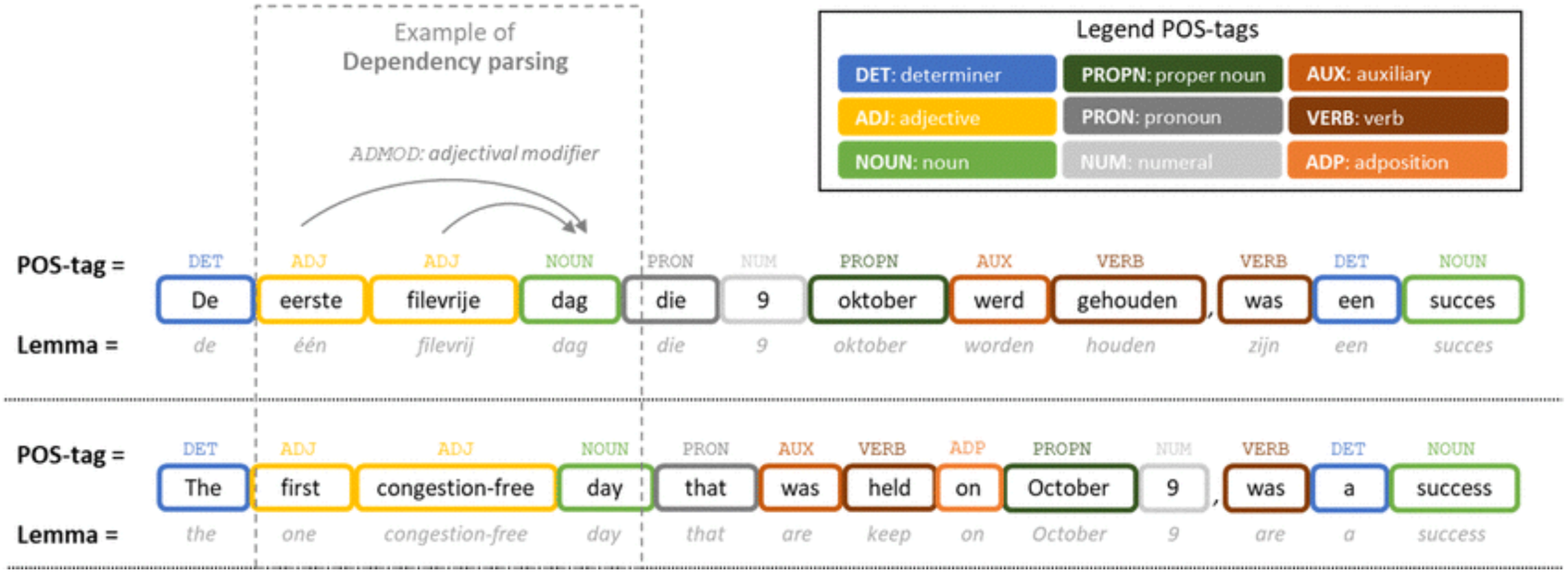
consist of one or more

Morphemes

consist of one or more phonemes .

Lexical analysis

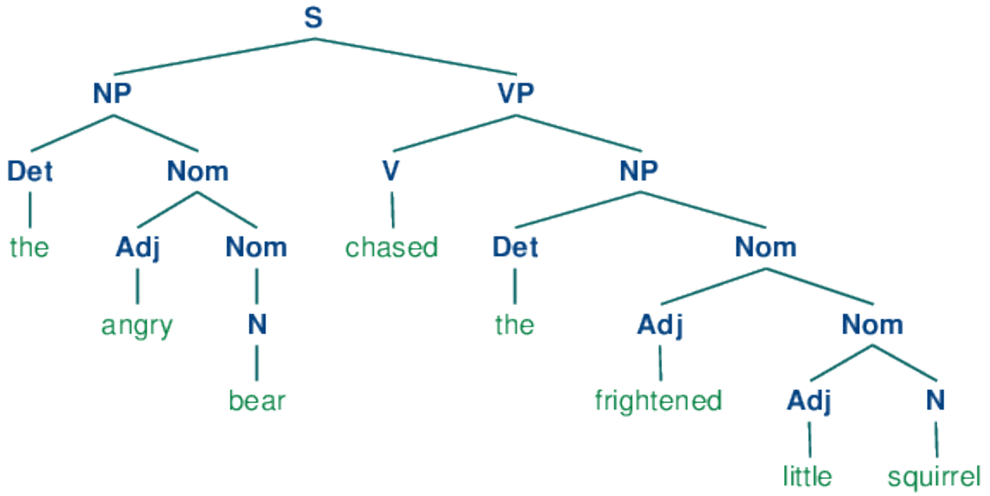
Get tokens or words from sentences, and obtain linguistic information of the words, e.g., tokenization, word segmentation, Part-of-Speech (POS) tagging



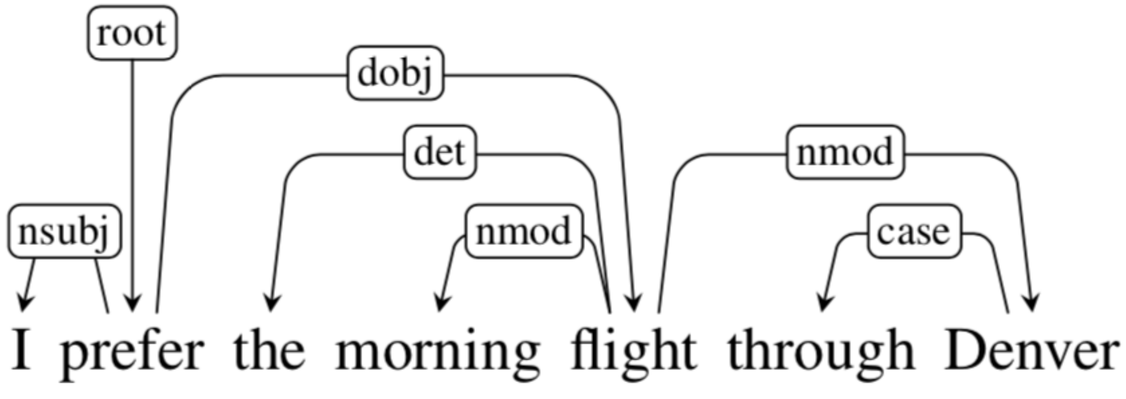
POS tagging

Syntactic analysis

Syntactic analysis, also referred to as **syntax analysis** or **parsing**, is the process of analyzing natural language with the rules of a **formal grammar**. Grammatical rules are applied to categories and groups of words, not individual words. Syntactic analysis basically **assigns a semantic structure to text**.



Constituency Parsing



Dependency Parsing

Semantic analysis

Semantic analysis, simply put, is the process of drawing **meaning** from text. Including Word Sense Disambiguation, Semantic Role Labeling, Semantic Parsing, etc.

LOGIC format:

∃ w, b, g:
instance(w, want-01) ∧ instance(g, go-01) ∧
instance(b, boy) ∧ arg0(w, b) ∧
arg1(w, g) ∧ arg0(g, b)

AMR format (based on PENMAN):

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
           :arg0 b))
```

GRAPH format:

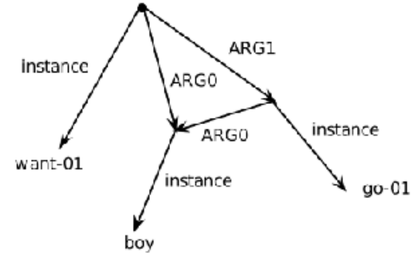
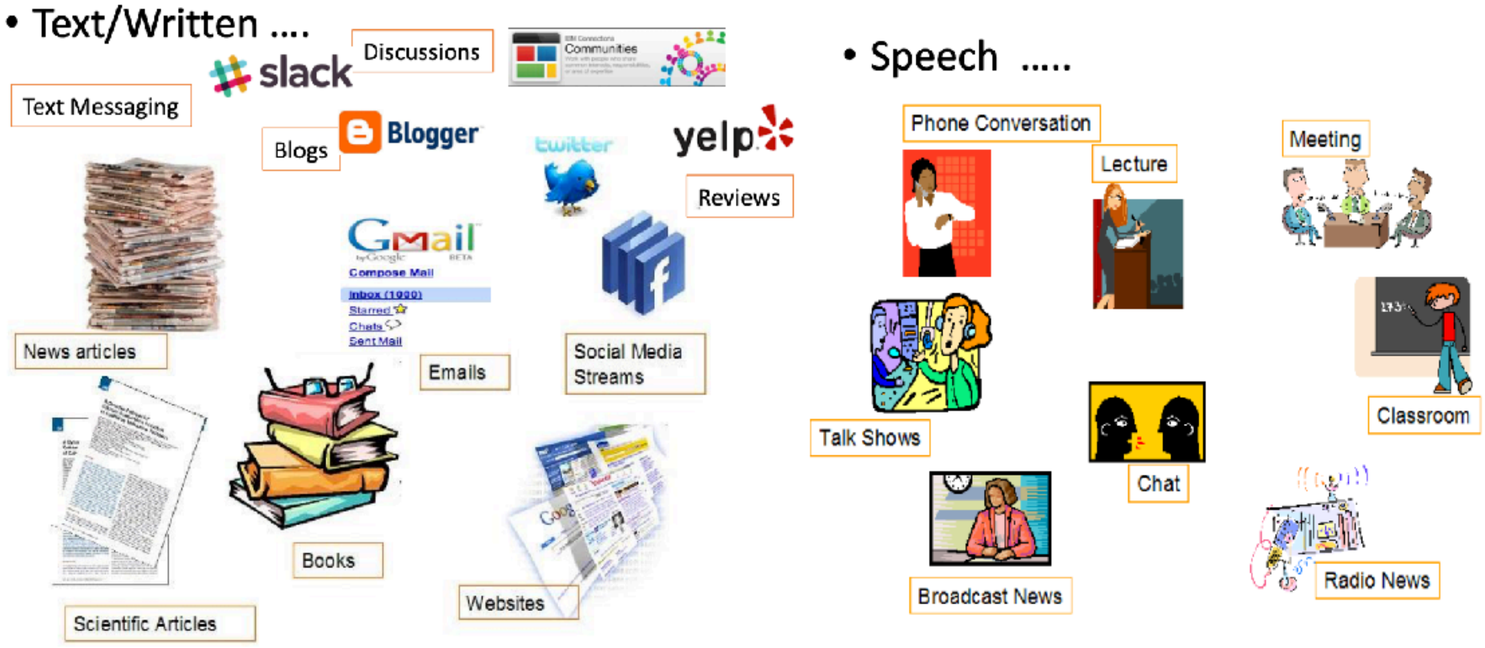


Figure 1: Equivalent formats for representating the meaning of "The boy wants to go".

Abstract Meaning Representation (AMR)

Discourse analysis

Discourse processing is a suite of Natural Language Processing (NLP) tasks to **uncover linguistic structures from texts at several levels**, which can **support many downstream applications**.

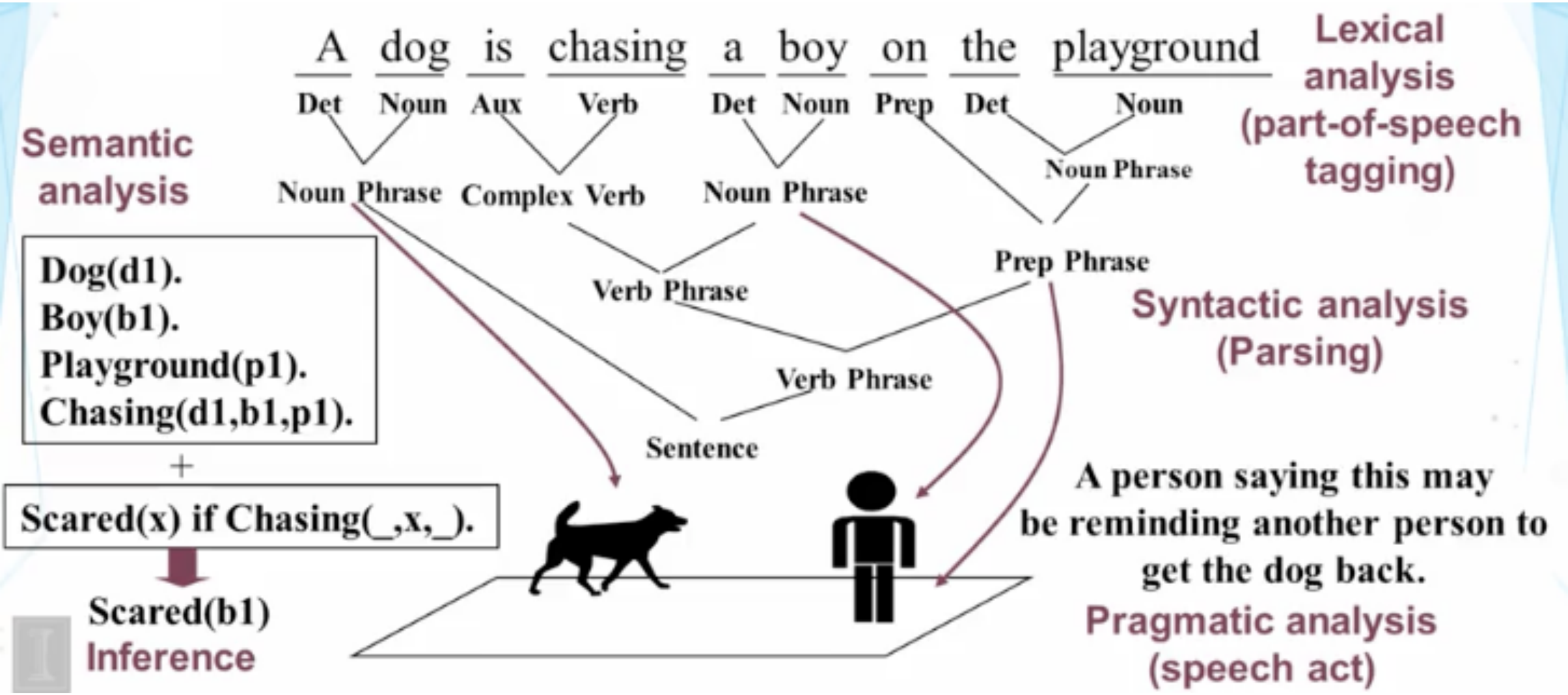


Discourse forms

Pragmatic analysis

Pragmatic analysis deals with outside **word knowledge**, which means knowledge that is external to the documents and/or queries. Pragmatics analysis that focuses on what was described is reinterpreted by what it actually meant, deriving the various aspects of language that require real world knowledge.

Basic concepts in NLP

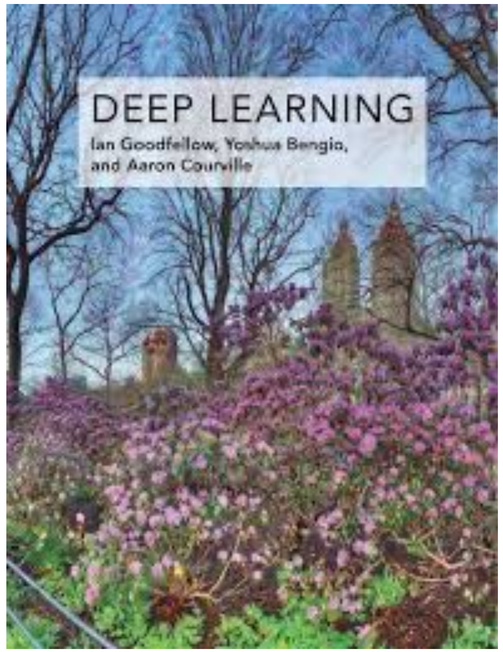


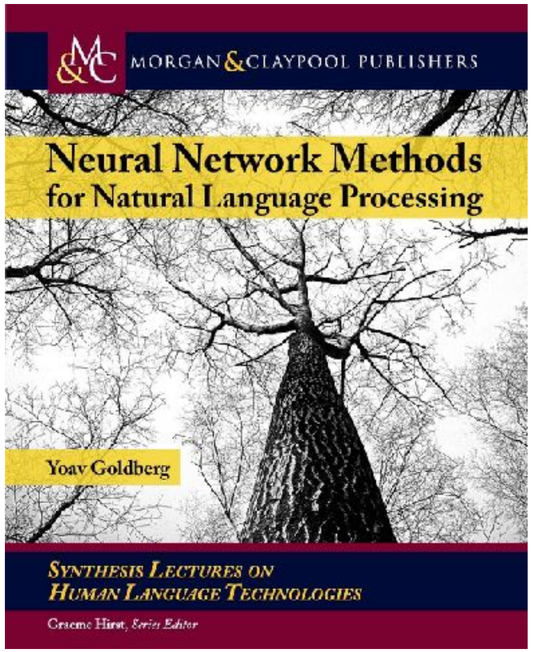

NLP + X
IT
Finance
Service
Health
Biology
Education
Law

Academic Conferences

- ACL (Association of Computational Linguistics)
- EMNLP (Conference on Empirical Methods in Natural language Processing)
- NAACL (The North American Chapter of the Association for Computational Linguistics)
- Coling (International Conference on Computational Linguistics)
- EACL (European Chapter of ACL)
- IJCNLP (International Joint Conference on Natural language Processing)
- SIGIR (SIG Information Retrieval)
- TREC (Text REtrieval Conference)
- SIGKDD (ACM Special Interest Group on Knowledge Discovery in Data)
- WWW (The Web Conference)
- NeurIPS (Conference on Neural Information Processing Systems)
- AAAI (Association for the Advancement of Artificial Intelligence)
- ICML, CIKM, ICDM, etc. (See “**AI Conference Deadlines**”)

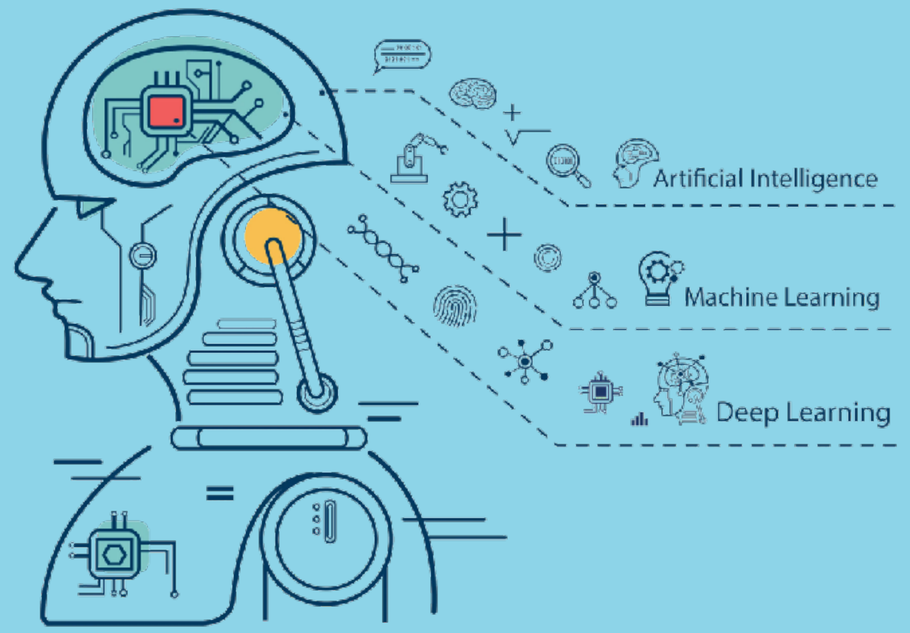
51 Books





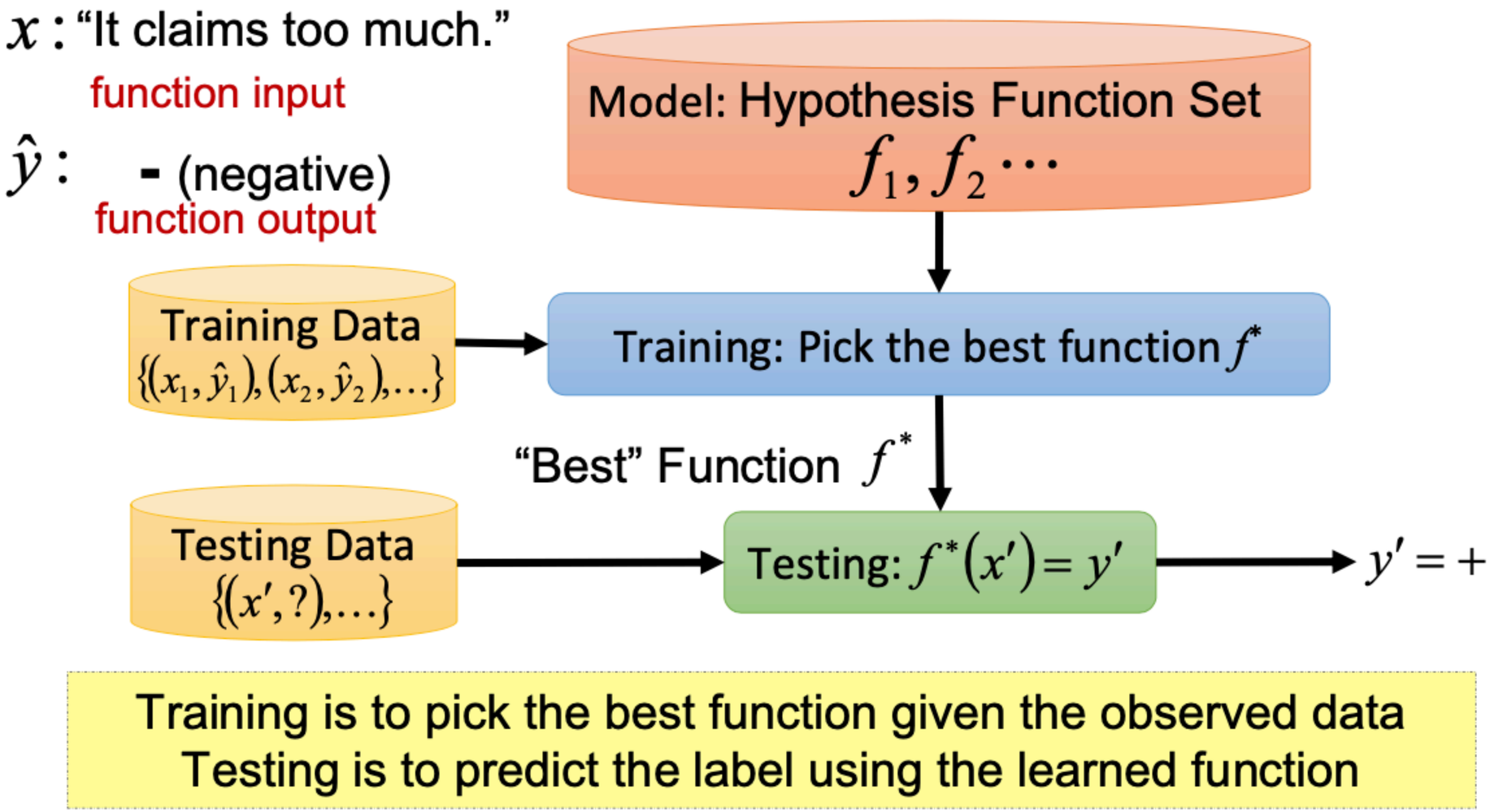
- No textbook is required
- Check the class webpage for more information

Machine Learning and Deep Learning



What is
Machine Learning

Machine Learning Framework

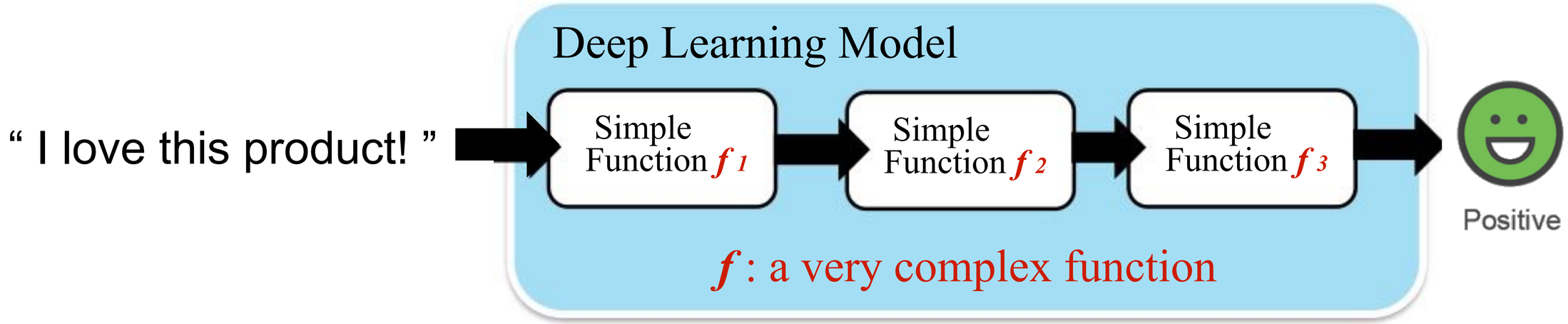


Training is to pick the best function given the observed data
Testing is to predict the label using the learned function

What is
Deep Learning

Stacked Functions Learned by Machine

- Production line

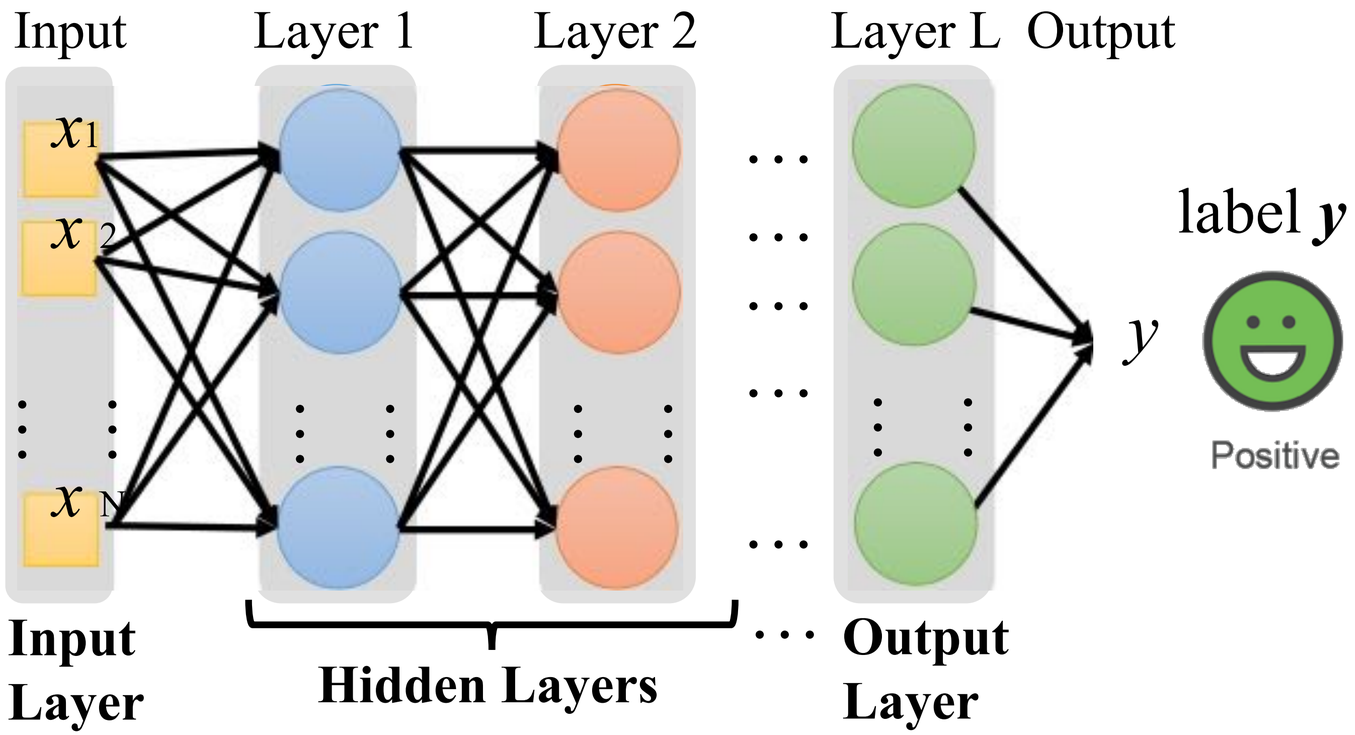


End-to-end training: what each function should do is learned automatically
Deep learning usually refers to neural network based model

Stacked Functions Learned by Machine

vector $\boldsymbol{x}$

"I love this product"

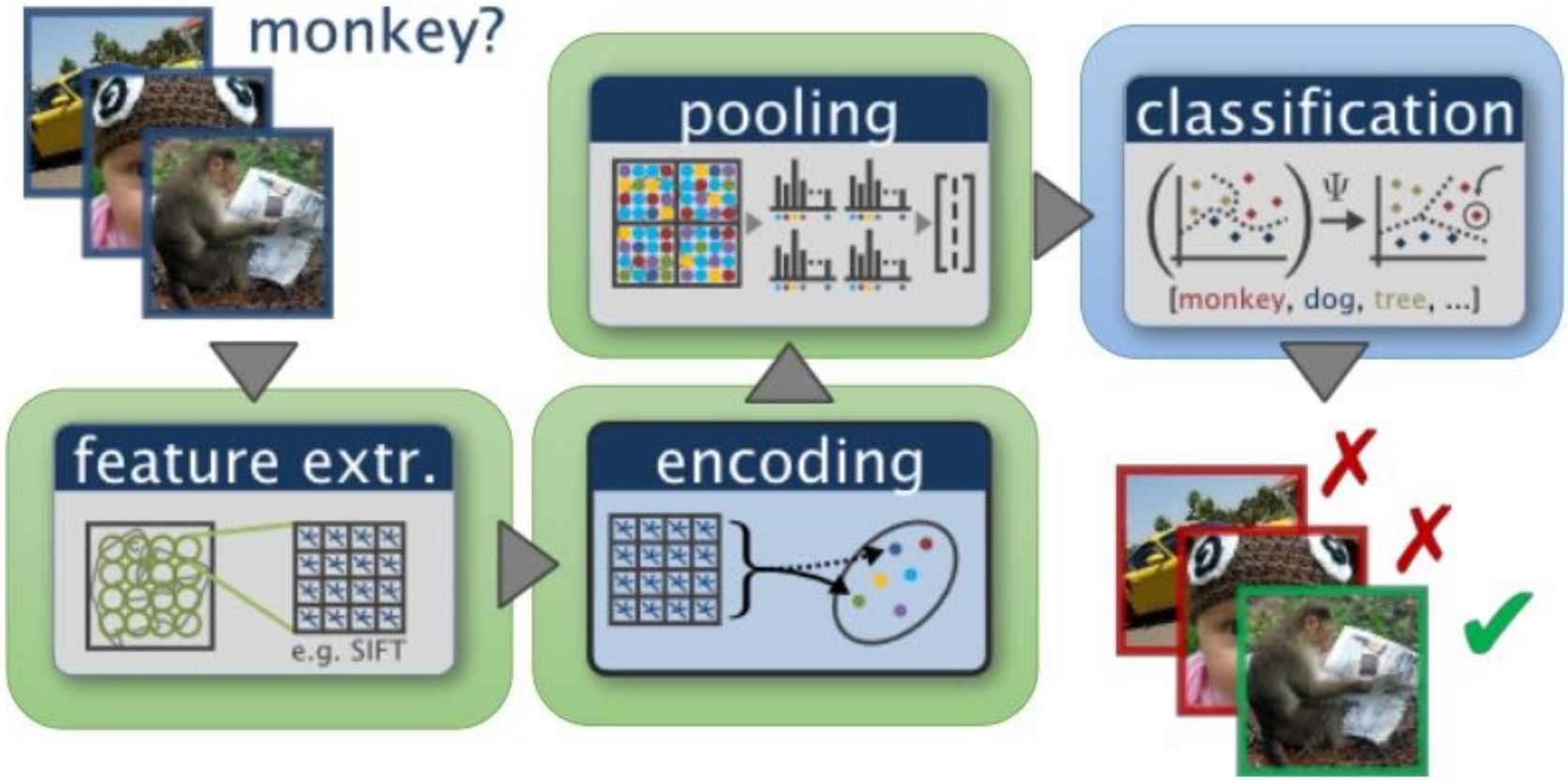


Features / Representations

Representation Learning attempts to learn good features/representations
Deep Learning attempts to learn (multiple levels of) representations and an output

Deep v.s. Shallow: Image Recognition

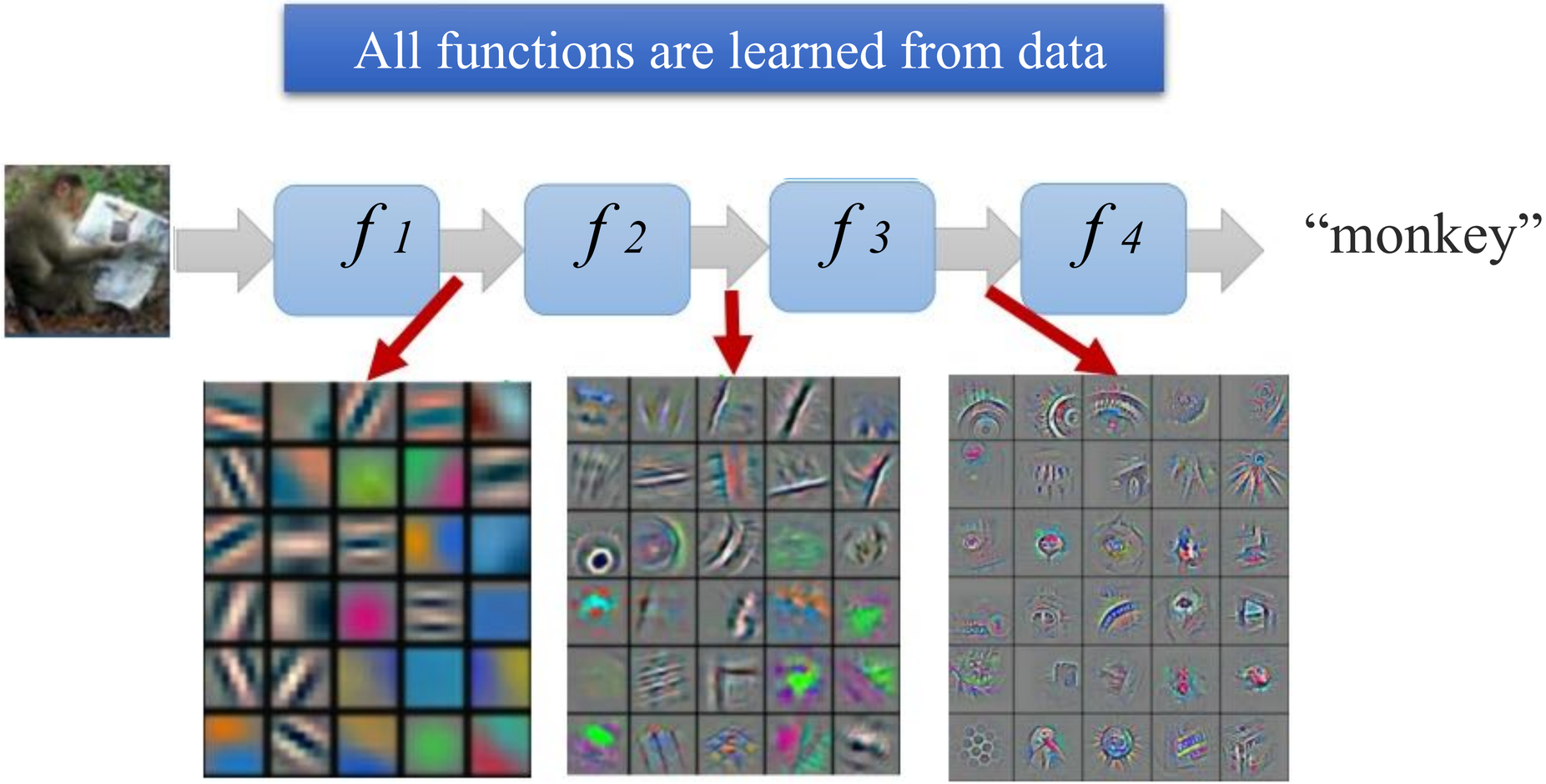
- Shallow model



Deep v.s. Shallow: Image Recognition

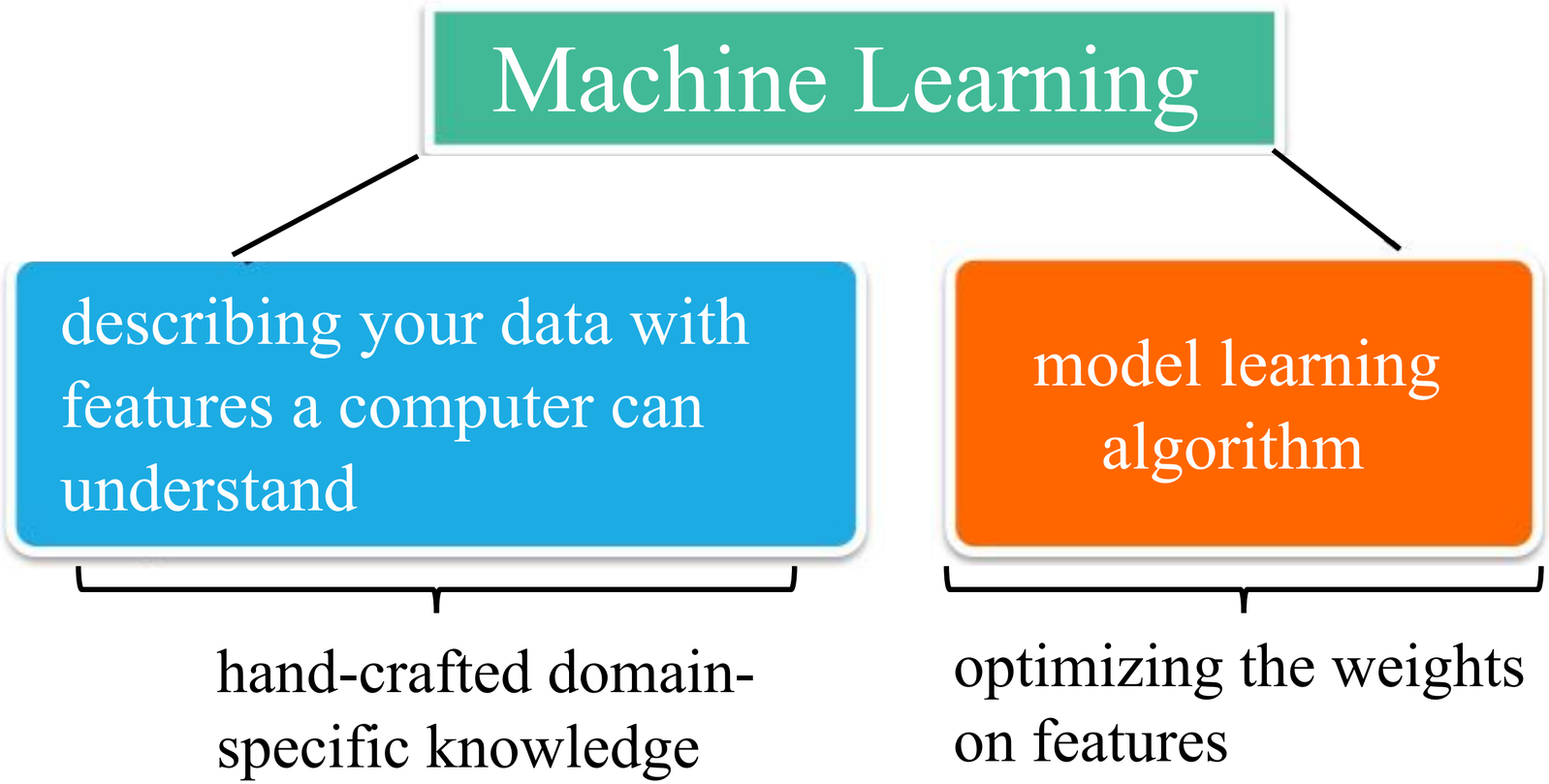
- Deep model

Reference: Zeiler , M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision –ECCV 2014* (pp. 818- 833)

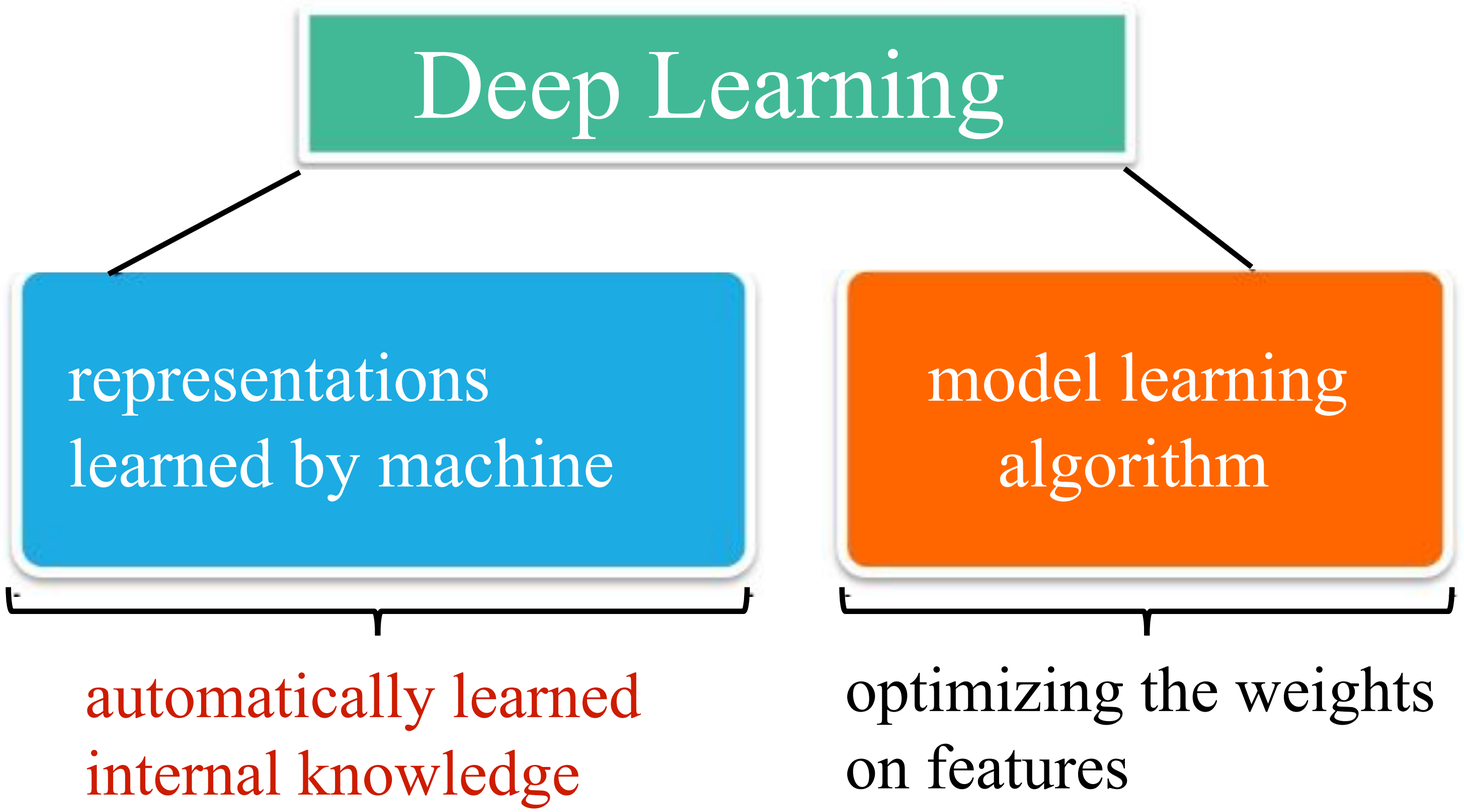


Features / Representations

Machine Learning v.s. Deep Learning

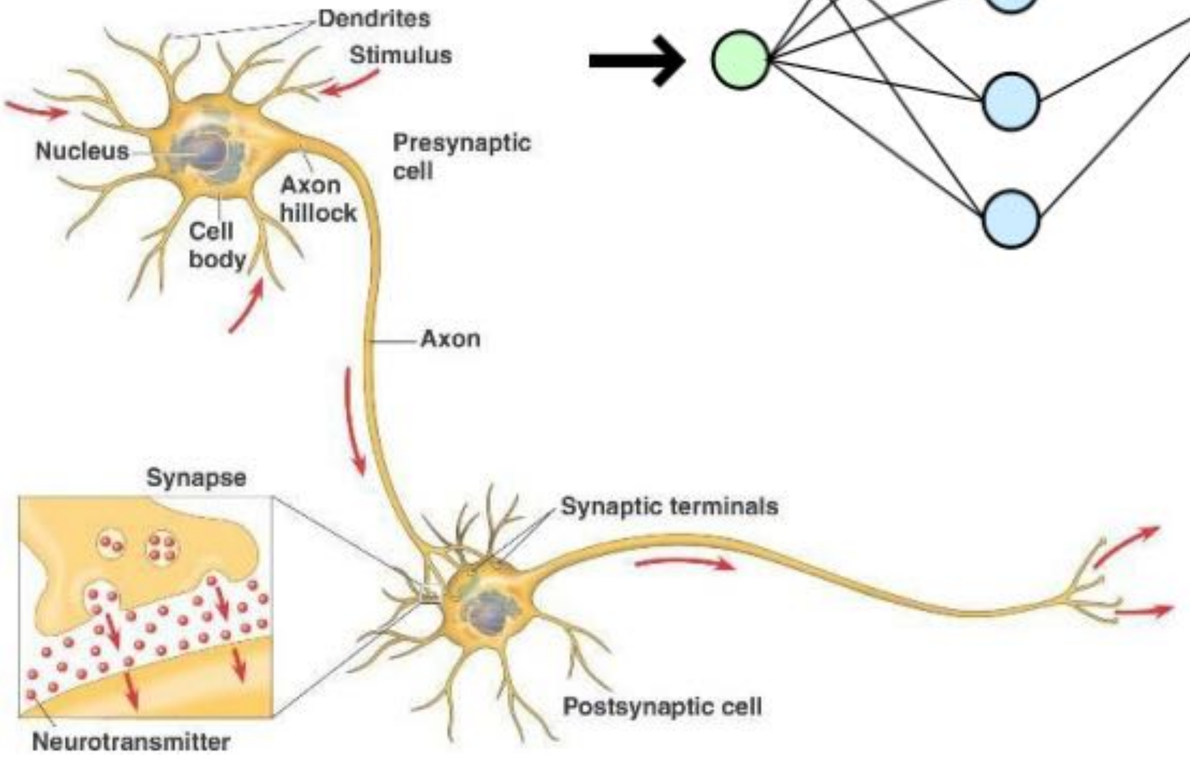


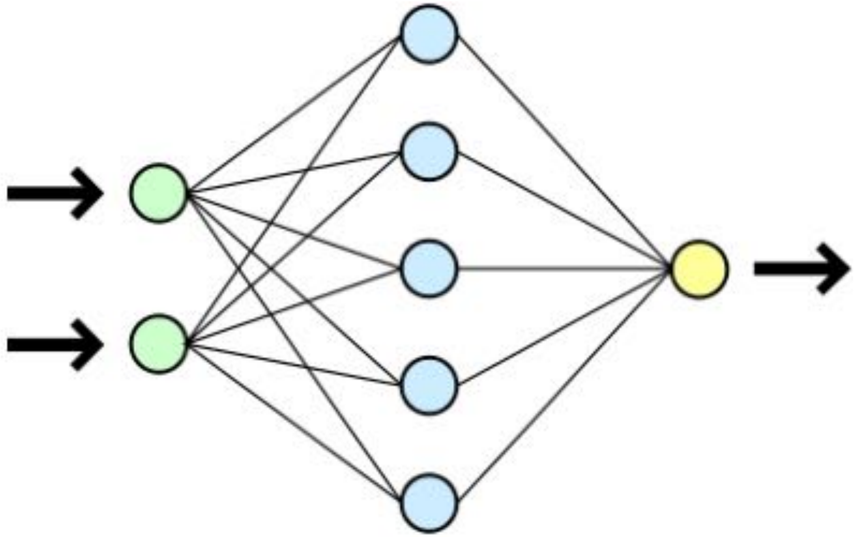
Machine Learning v.s. Deep Learning



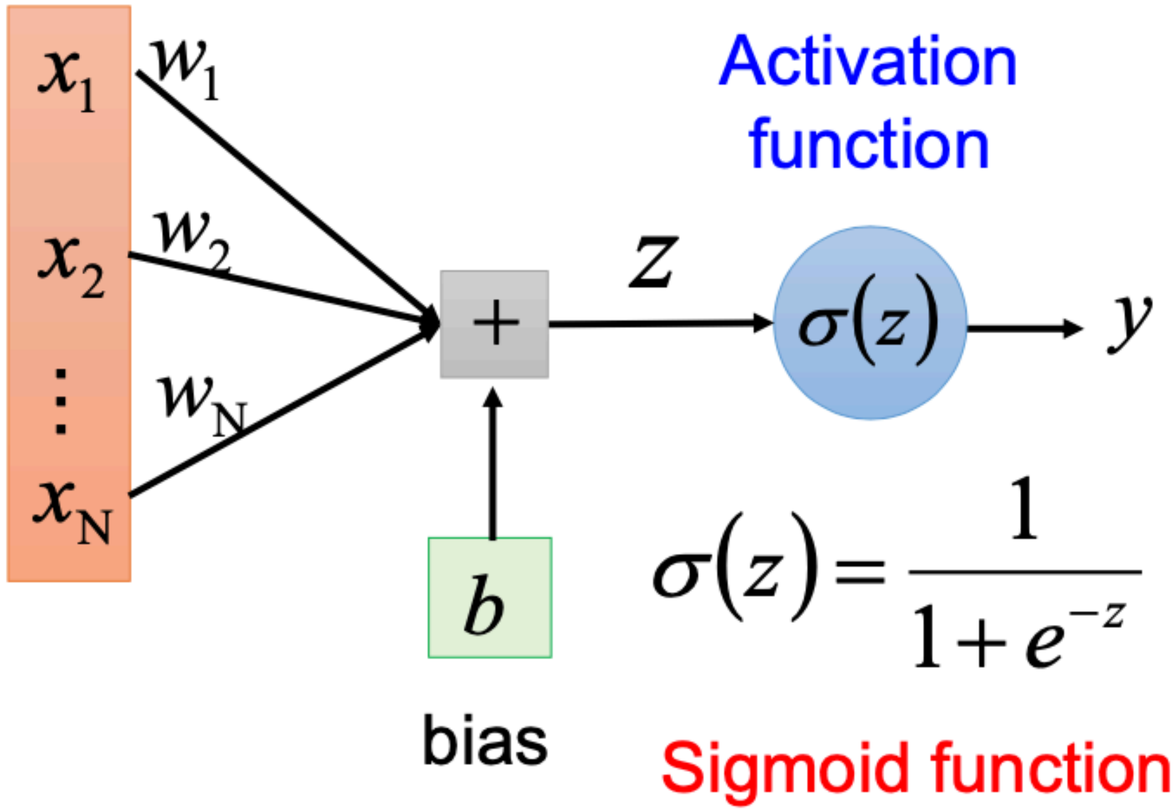
Deep learning usually refers to neural network based model

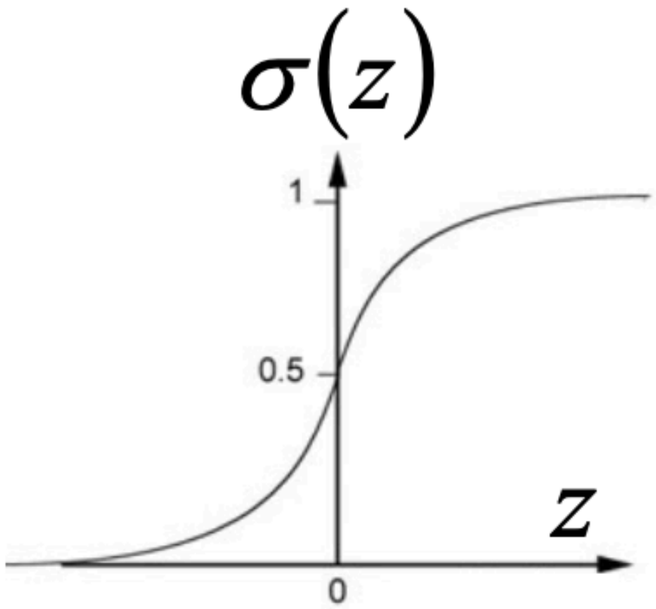
62 Inspired by Human Brain





A Single Neuron



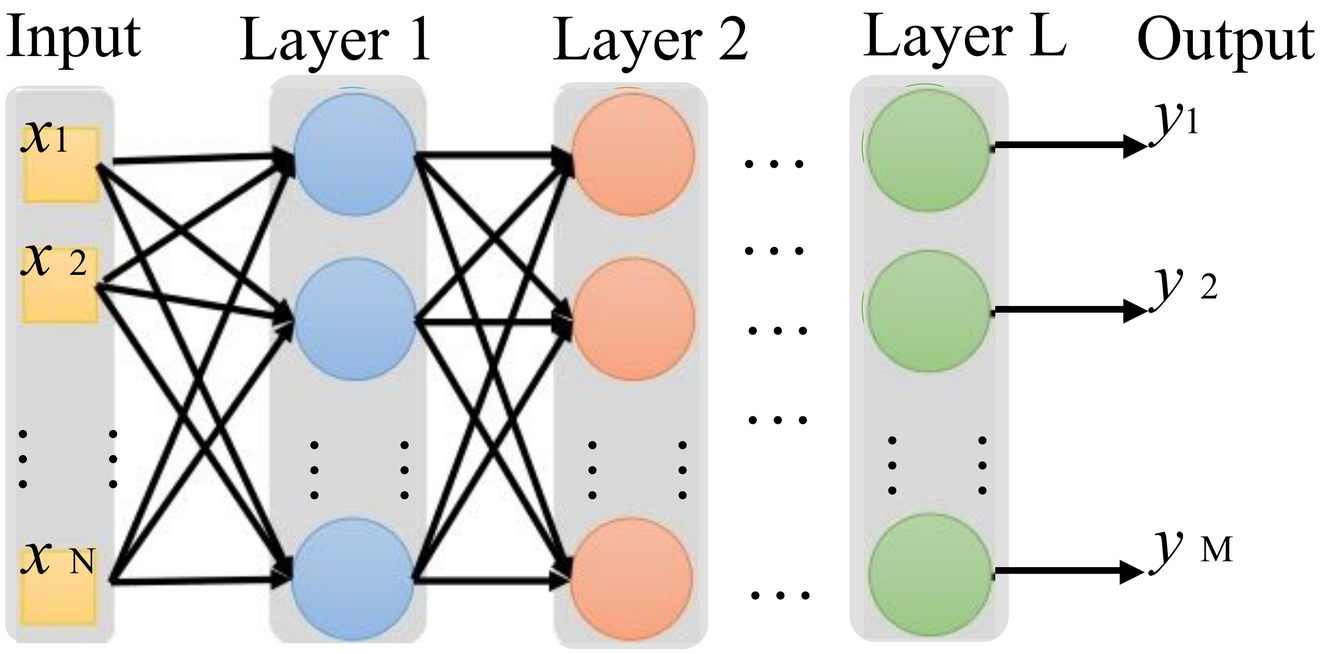


Each neuron is a very simple function

Deep Neural Network

- Cascading the neurons to form a neural network

A neural network is a complex function: $f : R^N \rightarrow R^M$



Each layer is a simple function in the production line

Why Deep Learning Works

Big Data

GPU

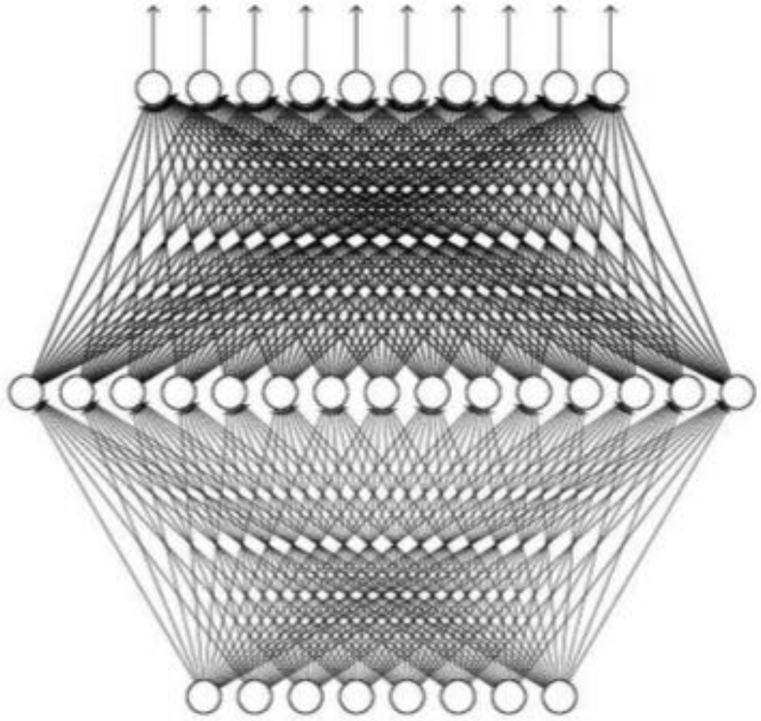
Universal Approximation Theorem

- Any continuous function f

$$f : R^N \rightarrow R^M$$

- can be realized by a network with one hidden layer

http://neuralnetworksanddeeplearning.com/chap4.html



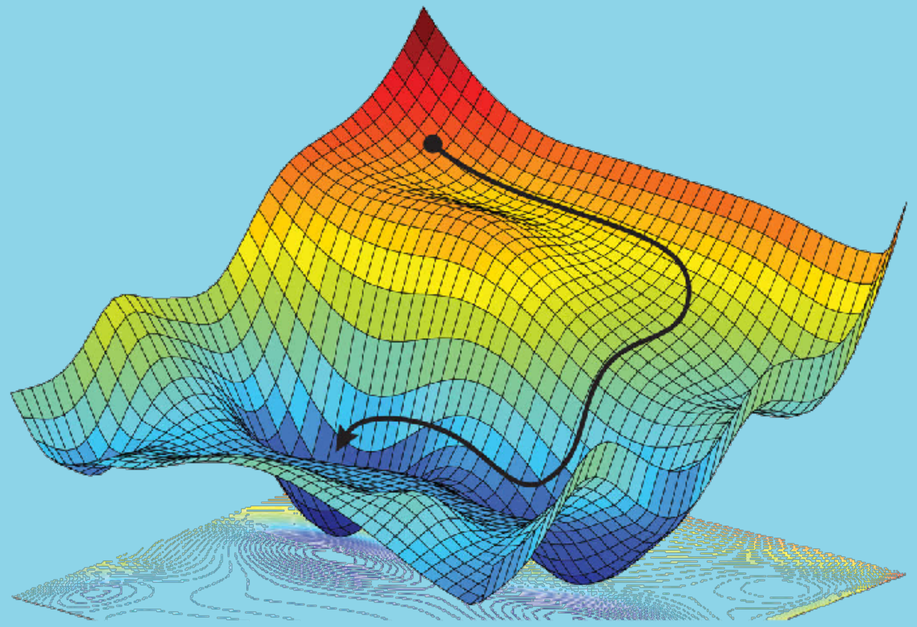
How to Frame the Learning Problem

- The **learning algorithm** f is to **map** the input domain X to output domain Y

$$f : X \rightarrow Y$$

- **Input domain:** word, word sequence, image, audio, video, click logs, brain signal …
- **Output domain:** single label, tag sequence, tree structure, probabilistic distribution …

Training Deep Neural Networks by Gradient Descent



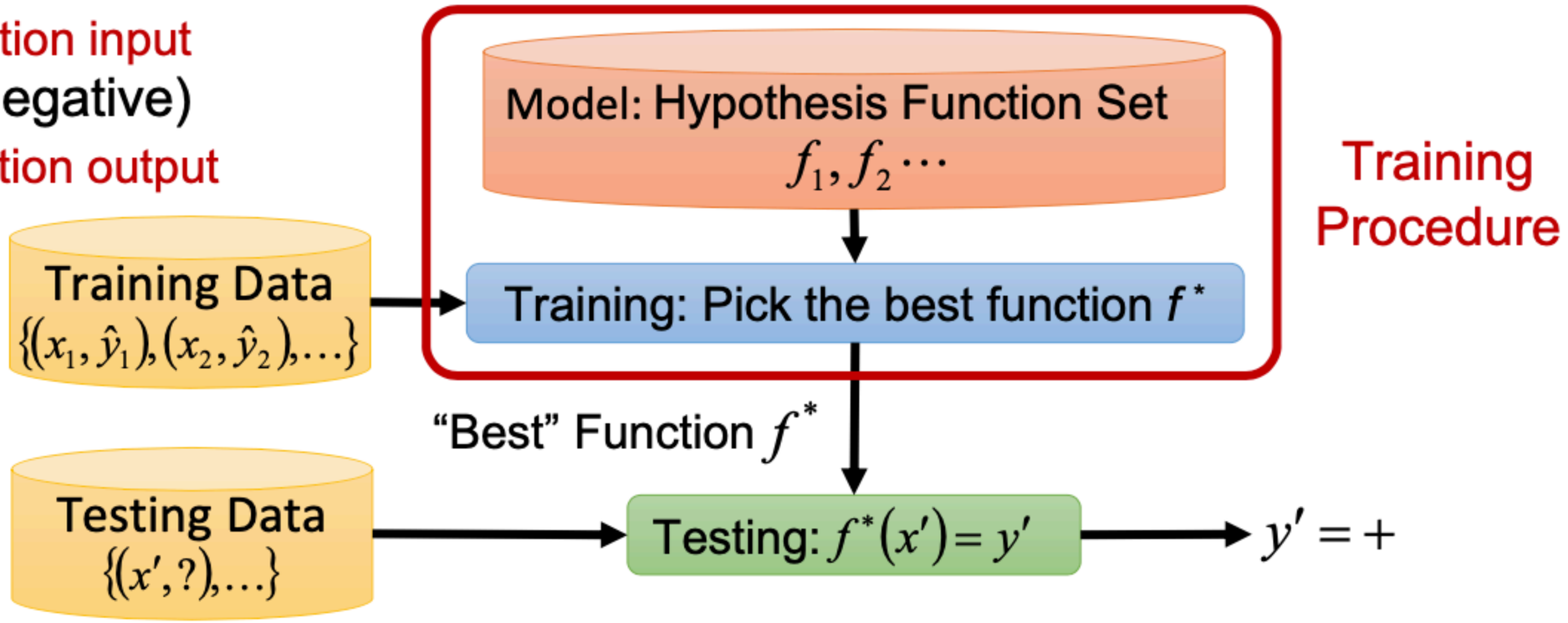
Machine Learning Framework

x : "It claims too much."

function input

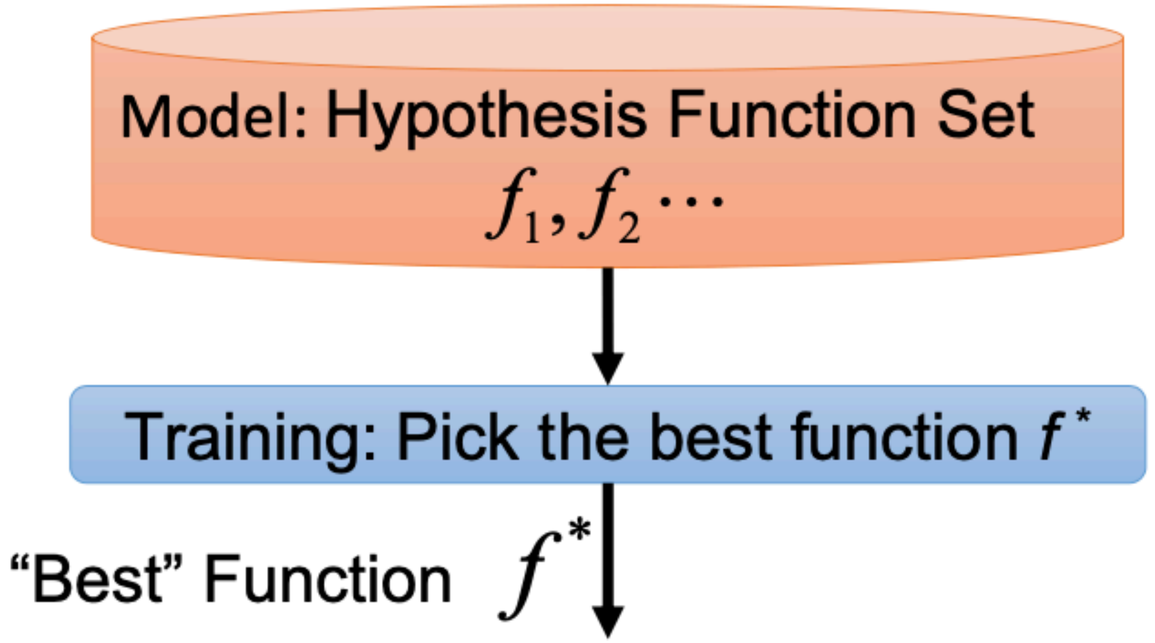
$\hat{y}$: - (negative)

function output



Training is to pick the best function given the observed data
Testing is to predict the label using the learned function

Training Procedure

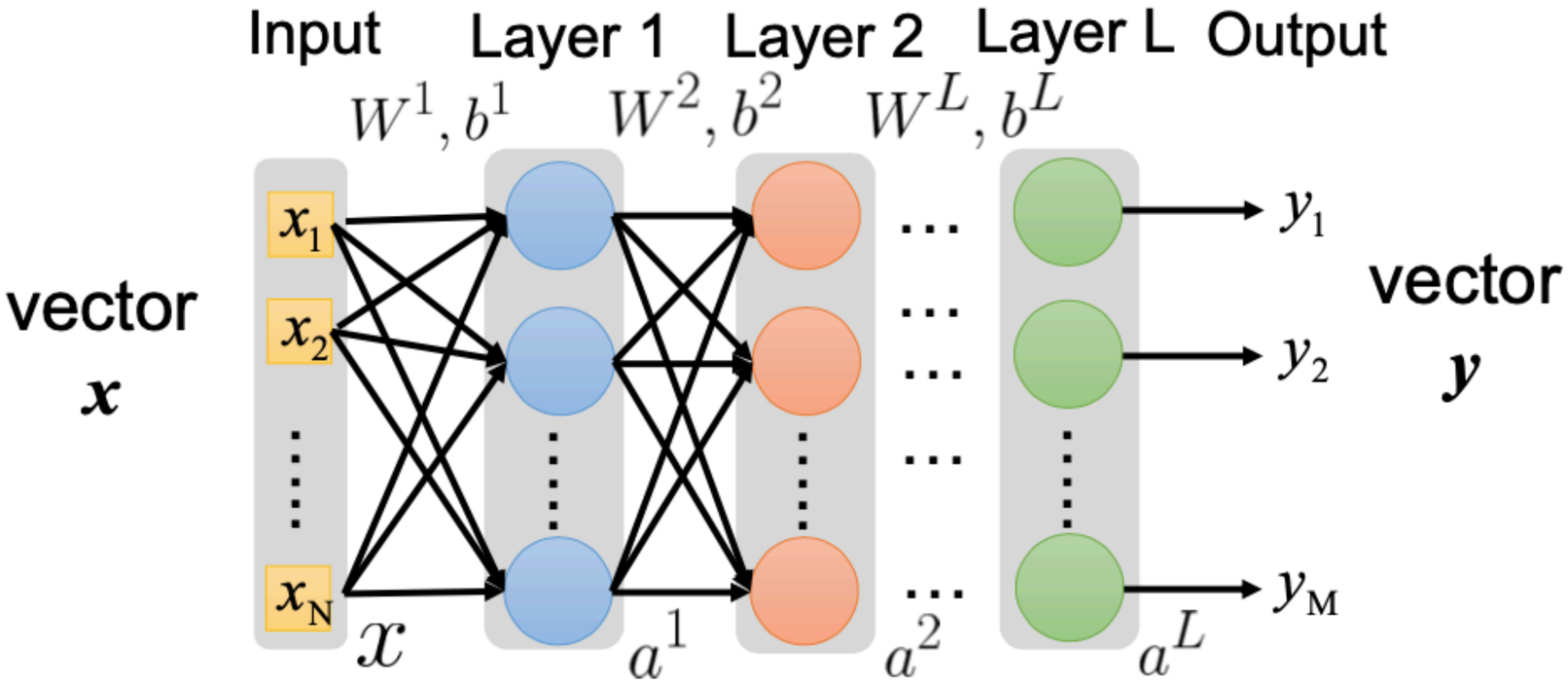


- Q1: What is the **model**? (function hypothesis set)
- Q2: What does a **good function** mean?
- Q3: How do we **pick** the "best" function?

What is the model?
Neural Networks

Neural Network Formulation

Fully connected feedforward network $f : R^N \rightarrow R^M$



$$y = f(x) = \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$

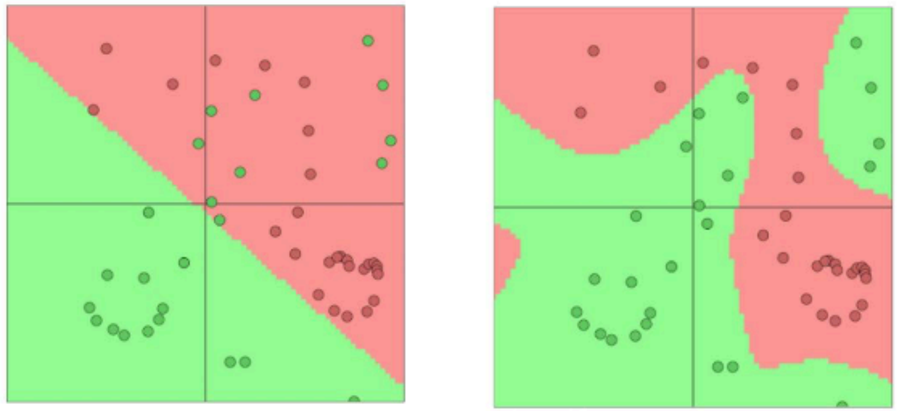
Why Non-Linearity?

Function approximation

- ***Without non-linearity***, deep neural networks work the same as linear transform

$$W_1(W_2 \cdot x) = (W_1 W_2)x = Wx$$

- ***With non-linearity***, networks with more layers can approximate more complex functions



What does the “good” function mean?

Loss Function

Function = Model Parameters

$$y = f(x) = \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$

function set

different parameters W and b → different functions

- Formal definition

$f(x;\theta)$ model parameter set

$$\theta = \left\{ W^1, b^1, W^2, b^2, \cdots W^L, b^L \right\}$$

pick a function f = pick a set of model parameters θ

Model Parameter Measurement

- Define a function to measure the quality of a parameter set θ
  - Evaluating by a loss/cost/error function $C(\theta)$ → how bad θ is
  - Best model parameter set

$$\theta^* = \arg\min_{\theta} C(\theta)$$

  - Evaluating by an objective/reward function $O(\theta)$ → how good θ is
  - Best model parameter set

$$\theta^* = \arg\max_{\theta} O(\theta)$$

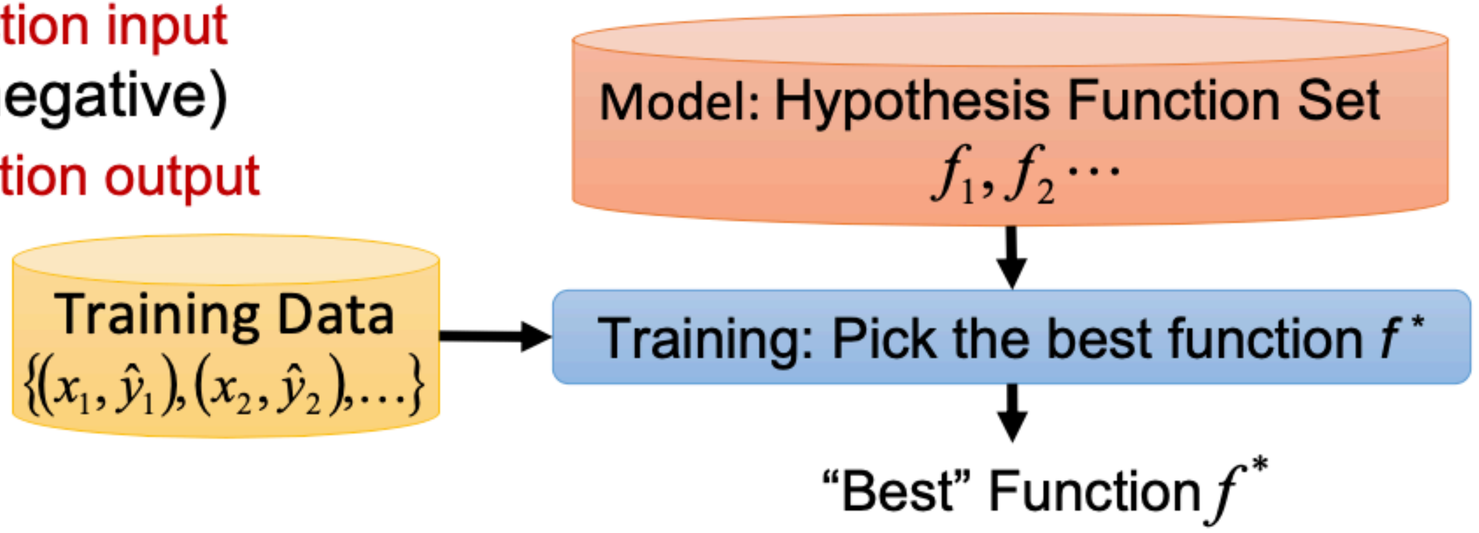
Loss Function Example

x : "It claims too much."

function input

$\hat{y}$: - (negative)

function output



A "Good" function: $f(x;\theta) \sim \hat{y}$ ➡ $\|\hat{y} - f(x;\theta)\| \approx 0$

Define an example loss function: $C(\theta) = \sum_k \|\hat{y}_k - f(x_k;\theta)\|$

sum over the error of all training samples

Frequent Loss Functions

- Squared Error Loss

$$L = \left(y - f(x)\right)^2$$

- Hinge Loss

$$L = \max\left(0, 1 - y * f(x)\right)$$

- Binary Cross Entropy Loss

$$L = -y * \log(p) - (1 - y) * \log(1 - p)$$

- Multi-Class Cross Entropy Loss

$$L(X_i, Y_i) = -\sum_{j=1}^{c} y_{ij} * \log(p_{ij})$$

where Y_i is one − hot encoded target vector $(y_{i1}, y_{i2}, \ldots, y_{ic})$,

$$y_{ij} = \begin{cases} 1, & \text{if } i_{th} \text{ element is in class } j \\ 0, & \text{otherwise} \end{cases}$$

$p_{ij} = f(X_i) = Probability\ that\ i_{th}\ element\ is\ in\ class\ j$

https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/

How can we pick the “best” function?

Optimization

Problem Statement

- Given a loss function and several model parameter sets
  - Loss function: $C(\theta)$
  - Model parameter sets: $\{\theta_1, \theta_2, \cdots\}$
- Find a model parameter set that minimizes $C(\theta)$

How to solve this optimization problem?

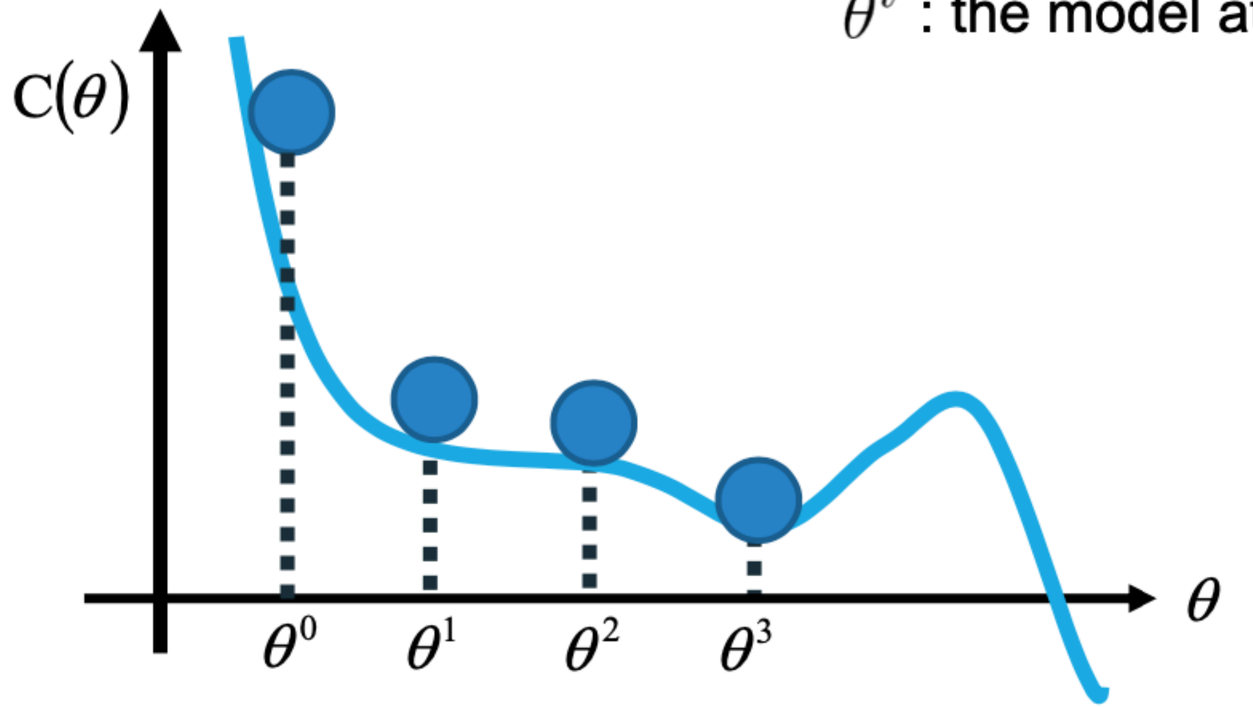
- 1) Brute force – enumerate all possible θ
- 2) Calculus – $\frac{\partial C(\theta)}{\partial \theta} = 0$

Issue: whole space of $C(\theta)$ is unknown

Gradient Descent for Optimization

- Assume that θ has only one variable

θ^i : the model at the i-th iteration



Idea: drop a ball and find the position where the ball stops rolling (local minima)

Gradient Descent for Optimization

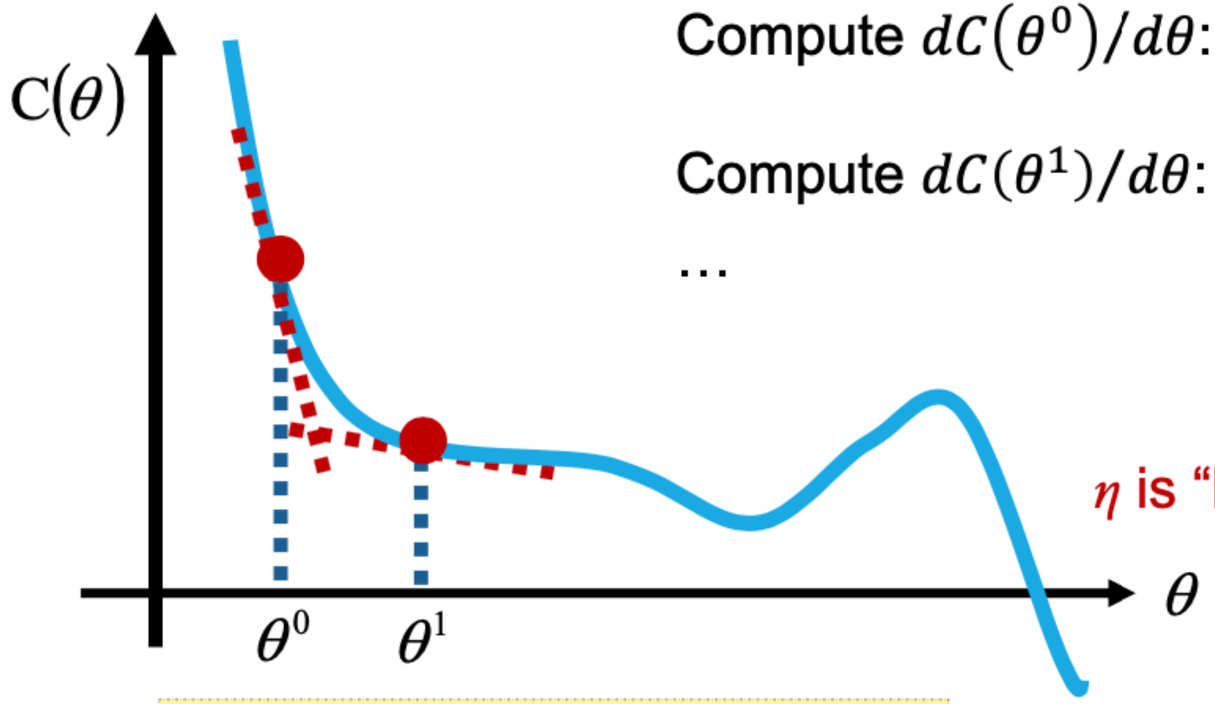
- Assume that θ has only one variable

Randomly start at θ^0

Compute $dC(\theta^0)/d\theta$: $\theta^1 \leftarrow \theta^0 - \eta \frac{\partial C(\theta^0)}{\partial \theta}$

Compute $dC(\theta^1)/d\theta$: $\theta^2 \leftarrow \theta^1 - \eta \frac{\partial C(\theta^1)}{\partial \theta}$

…

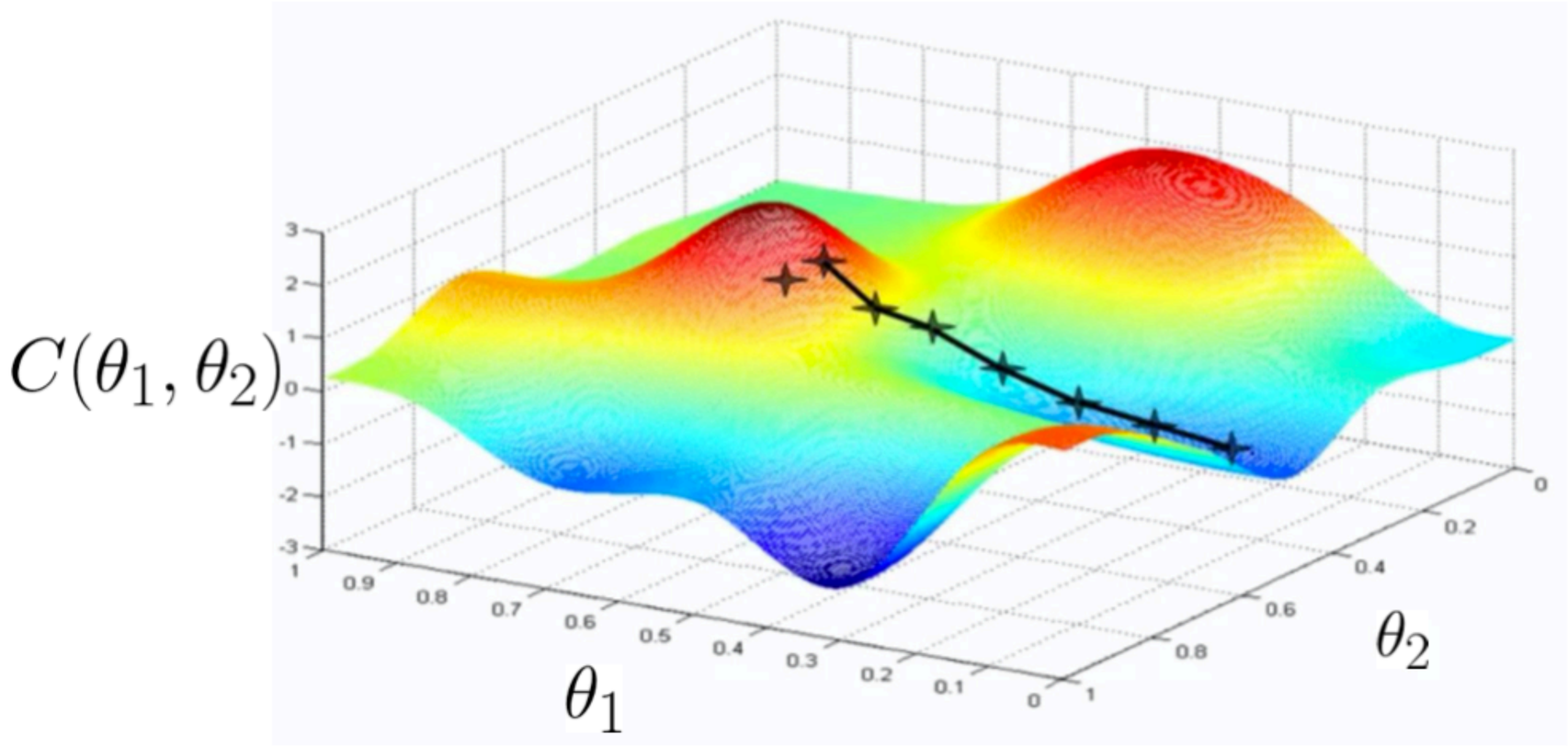


η is "learning rate"

$$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$$

Gradient Descent for Optimization

- Assume that θ has two variables $\{\theta_1, \theta_2\}$



Gradient Descent for Optimization

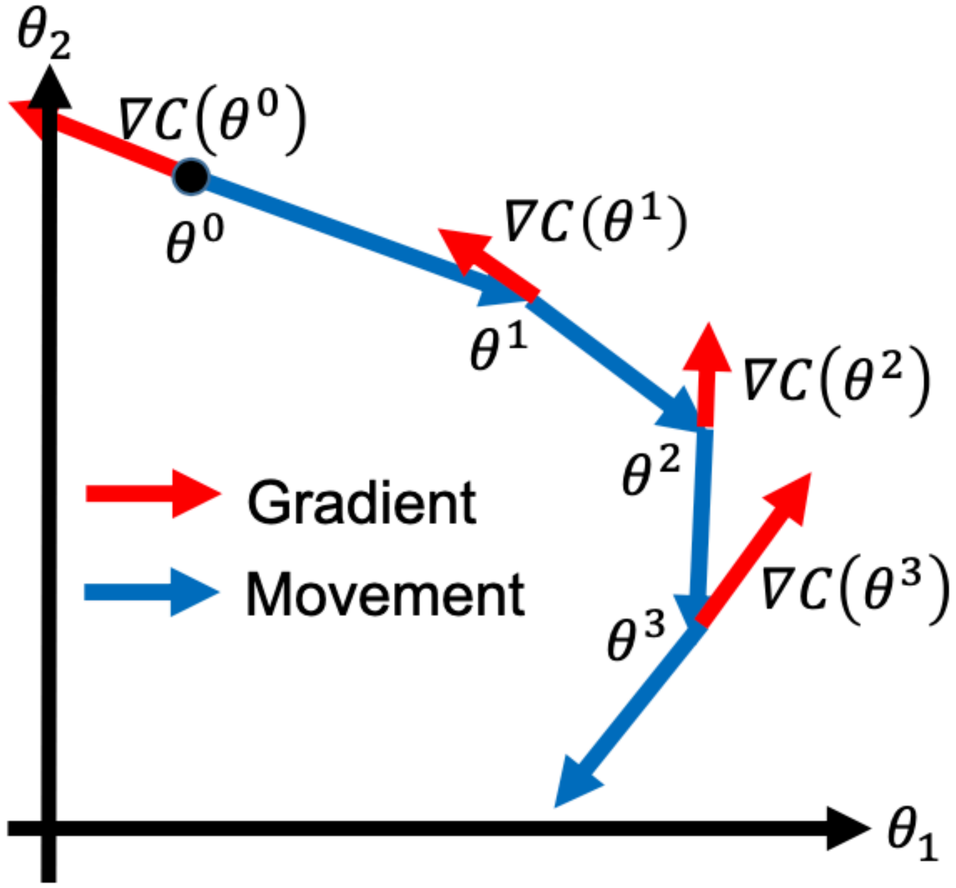
- Assume that θ has two variables $\{\theta_1, \theta_2\}$
  - Randomly start at θ^0: $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$
  - Compute the gradients of $C(\theta)$ at θ^0: $\nabla_\theta C(\theta^0) = \begin{bmatrix} \frac{\partial C(\theta_1^0)}{\partial \theta_1} \\ \frac{\partial C(\theta_2^0)}{\partial \theta_2} \end{bmatrix}$
  - Update parameters:

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial C(\theta_1^0)}{\partial \theta_1} \\ \frac{\partial C(\theta_2^0)}{\partial \theta_2} \end{bmatrix}$$

$$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$$

  - Compute the gradients of $C(\theta)$ at θ^1: $\nabla_\theta C(\theta^1) = \begin{bmatrix} \frac{\partial C(\theta_1^1)}{\partial \theta_1} \\ \frac{\partial C(\theta_2^1)}{\partial \theta_2} \end{bmatrix}$

Gradient Descent for Optimization



Algorithm

Initialization: start at θ^0

while($\theta^{(i+1)} \neq \theta^i$)

{

compute gradient at θ^i

update parameters

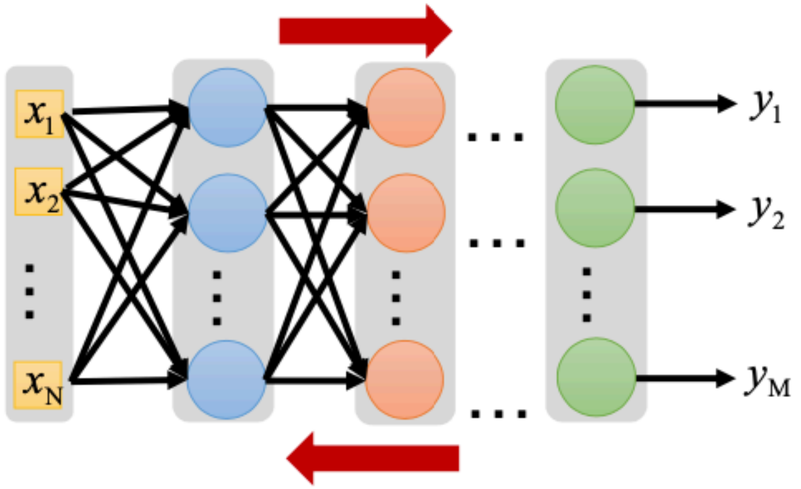
$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$

}

How to efficiently compute the gradients? Backpropagation

Forward v.s. Backpropagation

- In a feedforward neural network
  - forward propagation
    - from input x to output y information flows forward through the network
    - during training, forward propagation can continue onward until it produces a scalar cost $C(\theta)$
  - back-propagation
    - allows the information from the cost to then flow backwards through the network, in order to compute the **gradient**
    - can be applied to any function

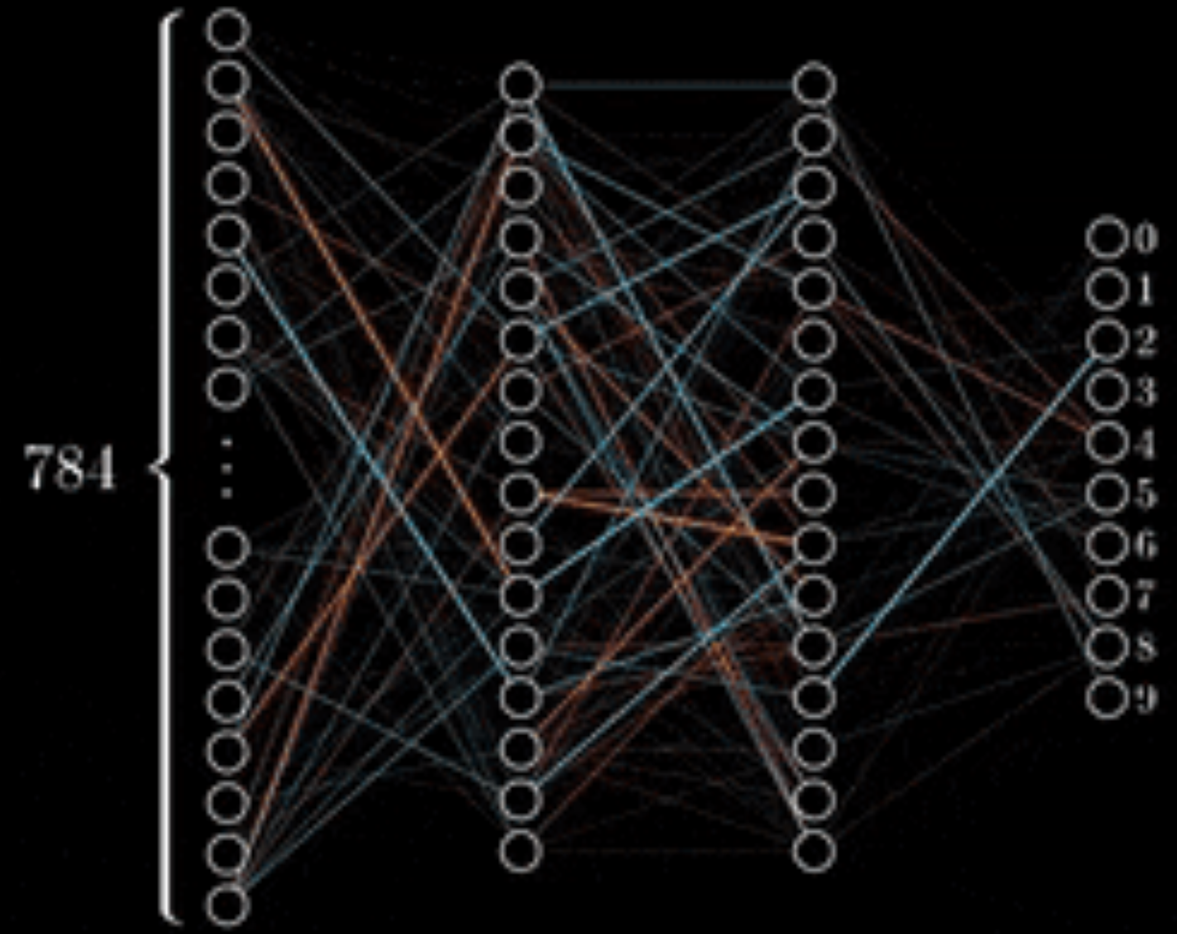


Why backward?

Reduce the redundant computations

(same computation repreated for different parameters. e.g., w, b)

Training in progress...



Chain Rule

$$\Delta w \rightarrow \Delta x \rightarrow \Delta y \rightarrow \Delta z$$

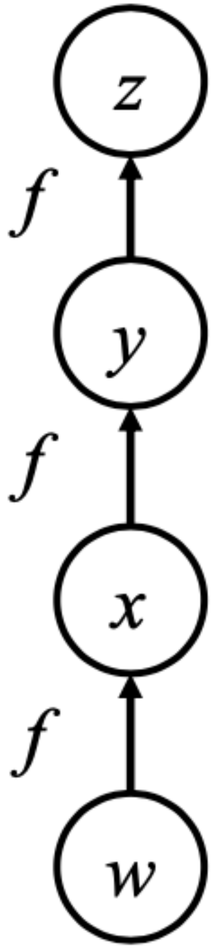
$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$

$$= f'(y)f'(x)f'(w)$$

$$= f'(f(f(w)))f'(f(w))f'(w)$$

forward propagation for cost

back-propagation for gradient



Gradient Descent for Neural Networks

$$y = f(x) = \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$

$$\theta = \left\{ W^1, b^1, W^2, b^2, \cdots W^L, b^L \right\}$$

$$W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \quad b^l = \begin{bmatrix} \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$\nabla C(\theta) = \begin{bmatrix} \vdots \\ \frac{\partial C(\theta)}{\partial w_{ij}^l} \\ \vdots \\ \frac{\partial C(\theta)}{\partial b_i^l} \end{bmatrix}$$

Algorithm

Initialization: start at θ^0

while($\theta^{(i+1)} \neq \theta^i$)

{

compute gradient at θ^i

update parameters

$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$

}

Computing the gradient includes millions of parameters.
To compute it efficiently, we use **backpropagation**.

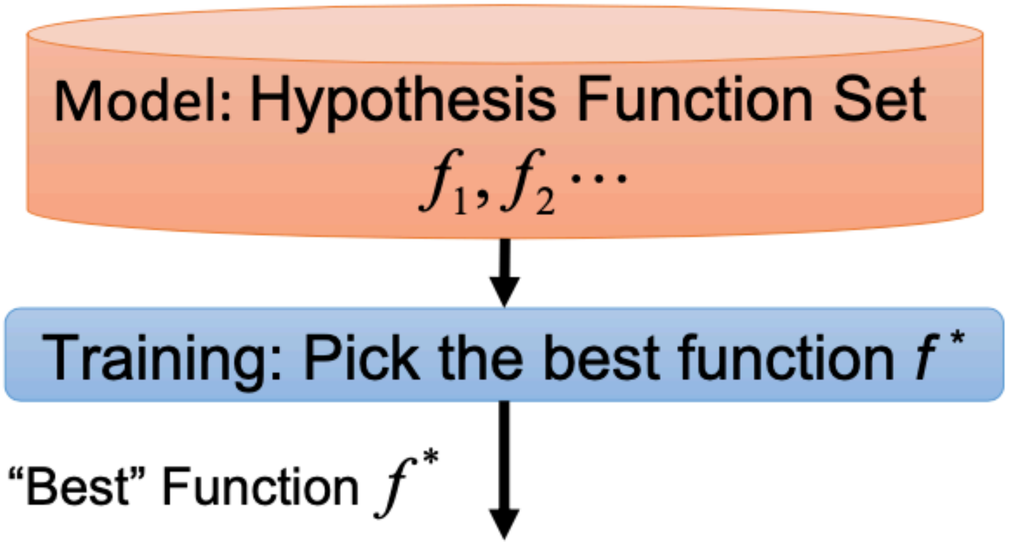
Concluding Remarks

- Q1. What is the model?
- Q2. What does a “good” function mean?
- Q3. How do we pick the “best” function?

Model Architecture

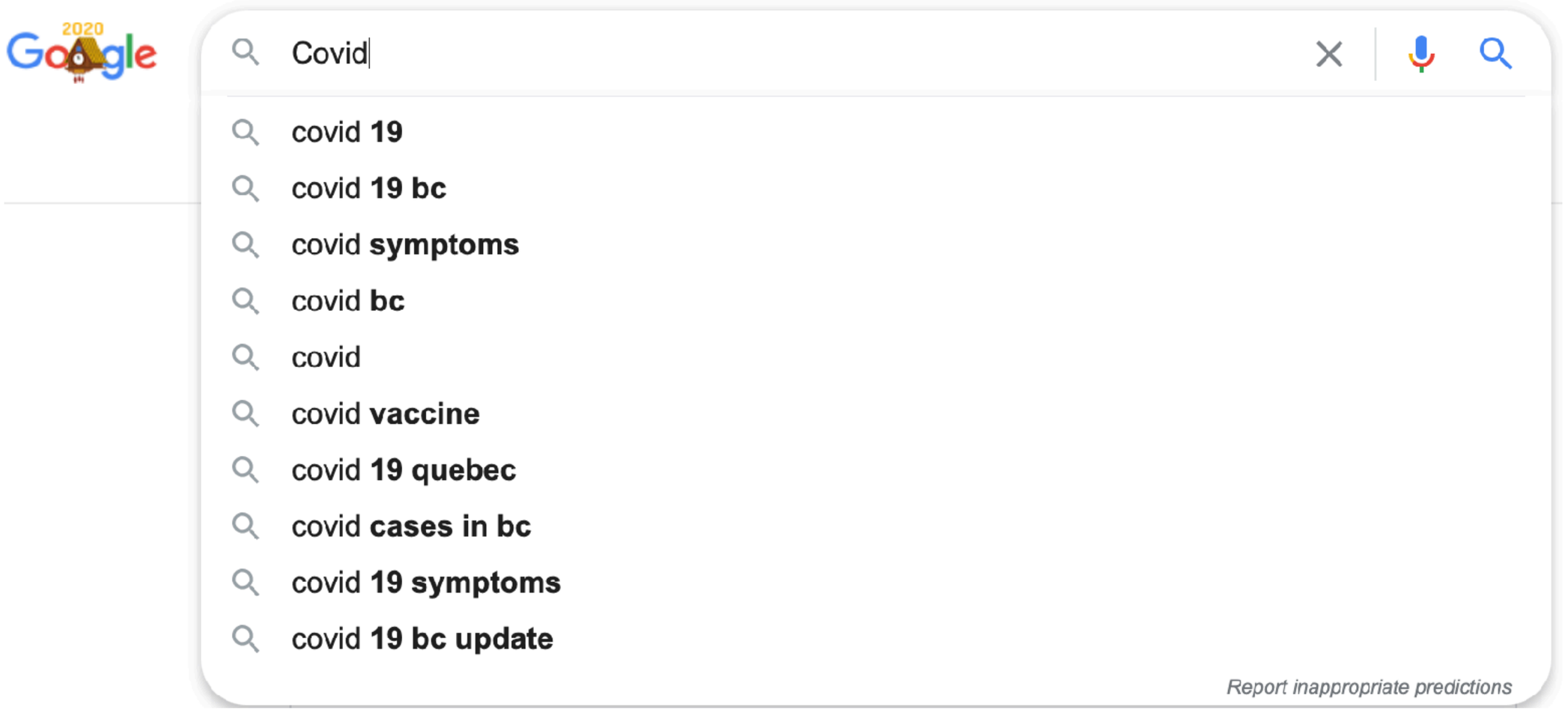
Loss Function Design

Optimization



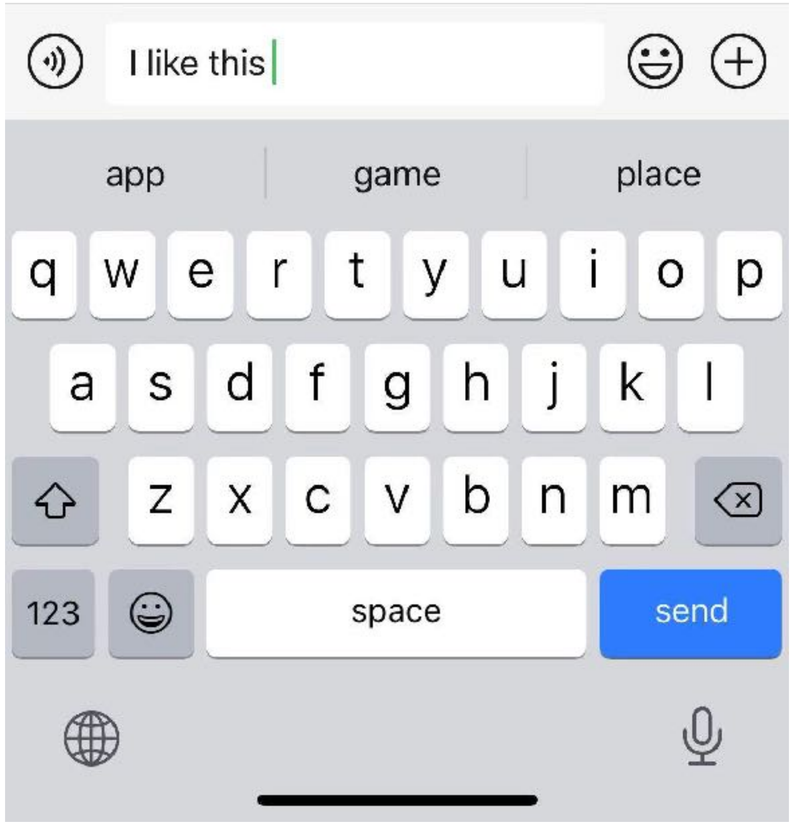
Language Modeling

You Use Language Models Everyday



What is Language Modeling

- **Language Modeling** is the task of predicting what word comes next.



What is Language Modeling

- **Language Model (LM)** is a system that estimates the probability of a piece of text (a word sequence)

Given a sequence of words:

$$x^{(1)}, x^{(2)}, \cdots, x^{(T)}$$

where $x^{(i)}$ can be any word in a vocabulary

$$V = \{w_1, \cdots, w_{|V|}\}$$

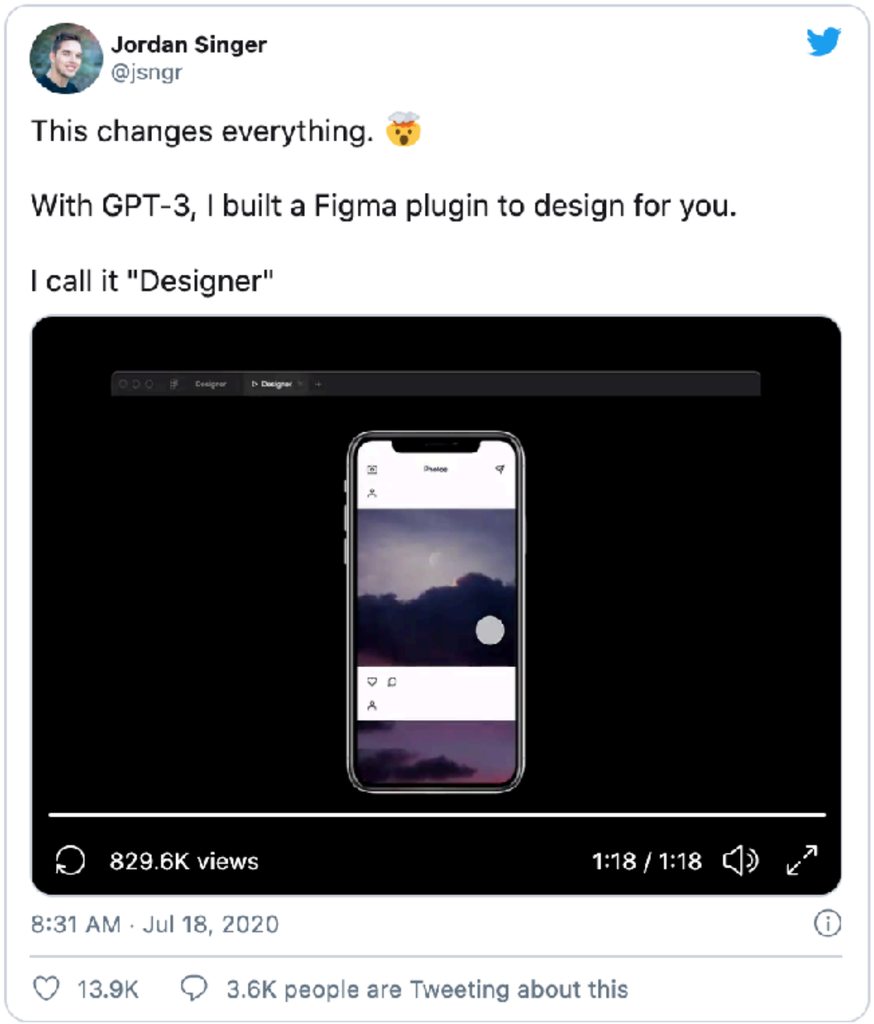
Estimate the probability:

$$P(x^{(1)}, x^{(2)}, \cdots, x^{(T)}) = P(x^{(1)}) \times P(x^{(2)}|x^{(1)}) \times \cdots \times P(x^{(T)}|x^{(T-1)}, \cdots, x^{(1)})$$

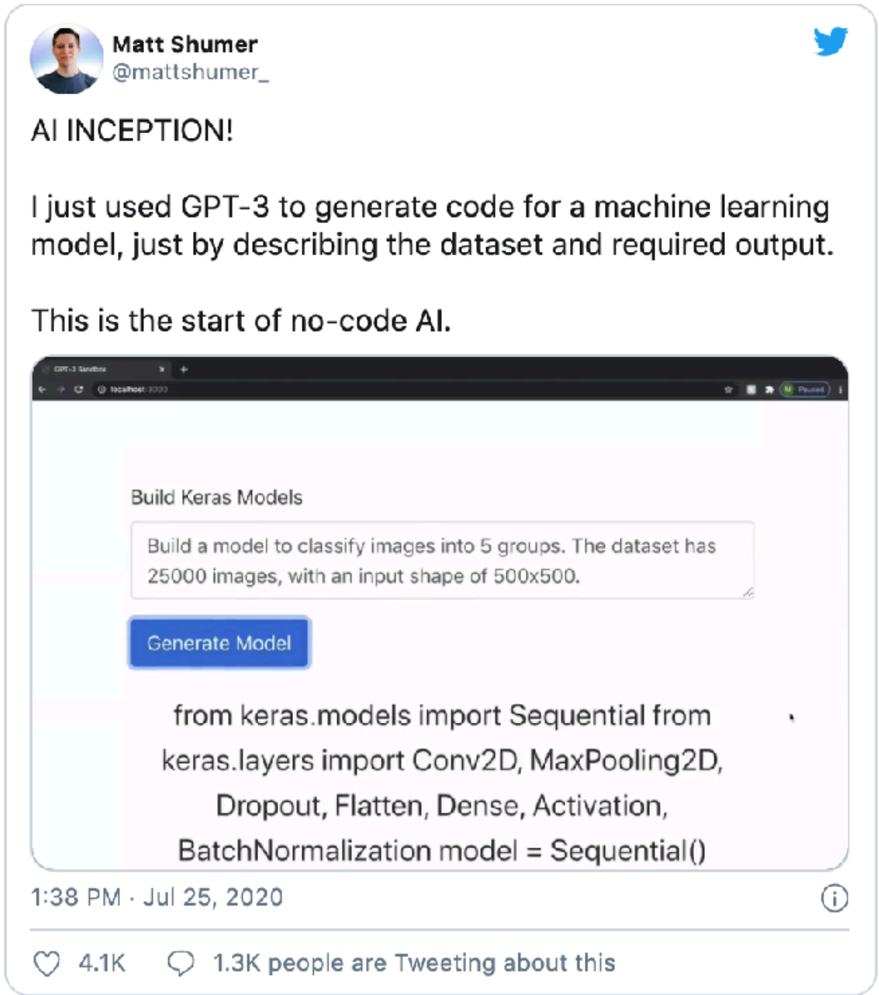
$$= \prod_{t=1}^{T} P(x^{(t)}|x^{(t-1)}, \cdots, x^{(1)})$$

This is what a LM provides

Language Models are Changing Everything



Design



Machine Learning Code

https://machinelearningtokyo.com/2020/07/26/10-cool-gpt-3-demos/

How to Learn a Language Model?

N-gram Language Models

N-gram Language Models

- A **n-gram** is a chunk of n consecutive words.

  Given a piece of text: "the students opened their …"
  - **uni**grams: "the", "students", "opened", "their"
  - **bi**grams: "the students", "students opened", "opened their"
  - **tri**grams: "the students opened", "students opened their"
  - **4**-grams: "the students opened their"
- **Idea (before deep learning)**: Collect statistics about how frequent different n-grams are, and use these to predict next word.

N-gram Language Models

- First we make a **Markov assumption**: $x^{(t+1)}$ depends only on the preceding (n-1) words:

n-1 words

$$
\begin{aligned}
P(x^{(t+1)}|x^{(t)},\cdots,x^{(1)}) &= P(x^{(t+1)}|x^{(t)},\cdots,x^{(t-n+2)}) && \textbf{(assumption)}\\
&= \frac{P(x^{(t+1)},x^{(t)},\cdots,x^{(t-n+2)})}{P(x^{(t)},\cdots,x^{(t-n+2)})} && \textbf{(conditional probability)}\\
&\approx \frac{\text{count}(x^{(t+1)},x^{(t)},\cdots,x^{(t-n+2)})}{\text{count}(x^{(t)},\cdots,x^{(t-n+2)})} && \textbf{(statistical approximation)}
\end{aligned}
$$

- We get these n-gram and (n-1)-gram probabilities by **counting** them in some large corpus of text!

100 N-gram Language Models: Example

- Suppose we are learning a 4-gram Language Model

~~as the proctor started the clock, the stu~~dents opened their ____

$$P(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}$$

- For example, suppose that in the corpus:
  - "students opened their" occurred 1000 times
  - If "students opened their books" occurred 400 times

$$P(\text{books}|\text{students opened their}) = 0.4$$

  - "students opened their exams" occurred 100 times

$$P(\text{exams}|\text{students opened their}) = 0.1$$

Shall we discard the context "proctor"?

101 N-gram Language Models in Practice

You can build a simple trigram Language Model over a
1.7 million word corpus (Reuters) in a few seconds on your laptop*

Business and financial news

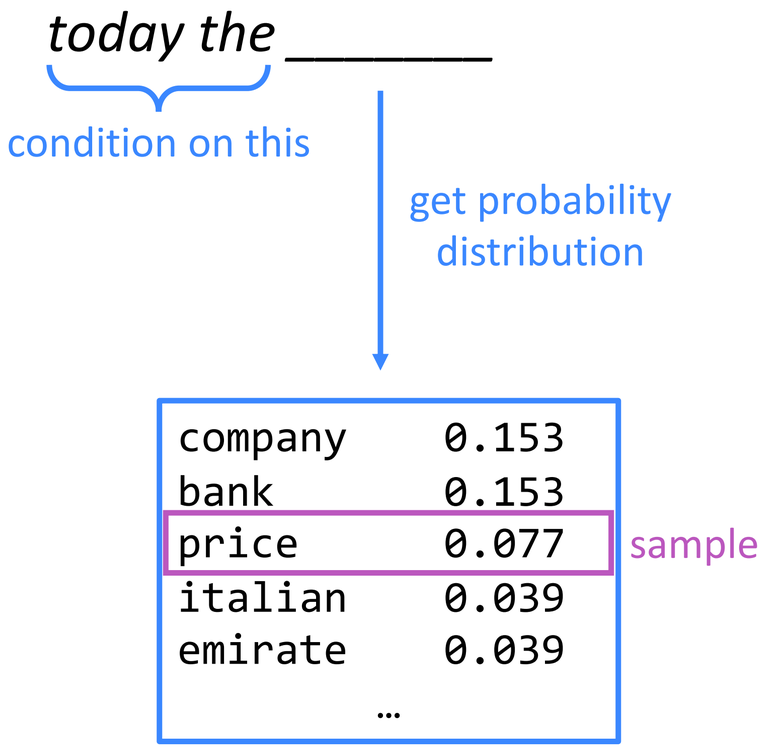
today the _____

get probability distribution

Sparsity problem:
not much granularity
in the probability
distribution

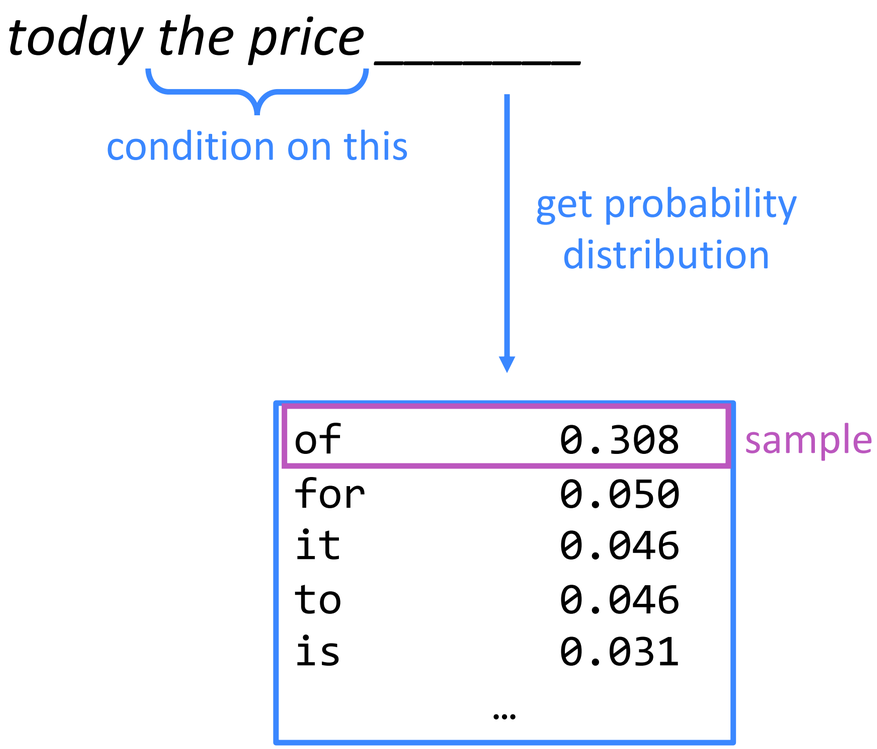
Generating Text with a N-gram Language Model

- You can also use a Language Model to generate text.



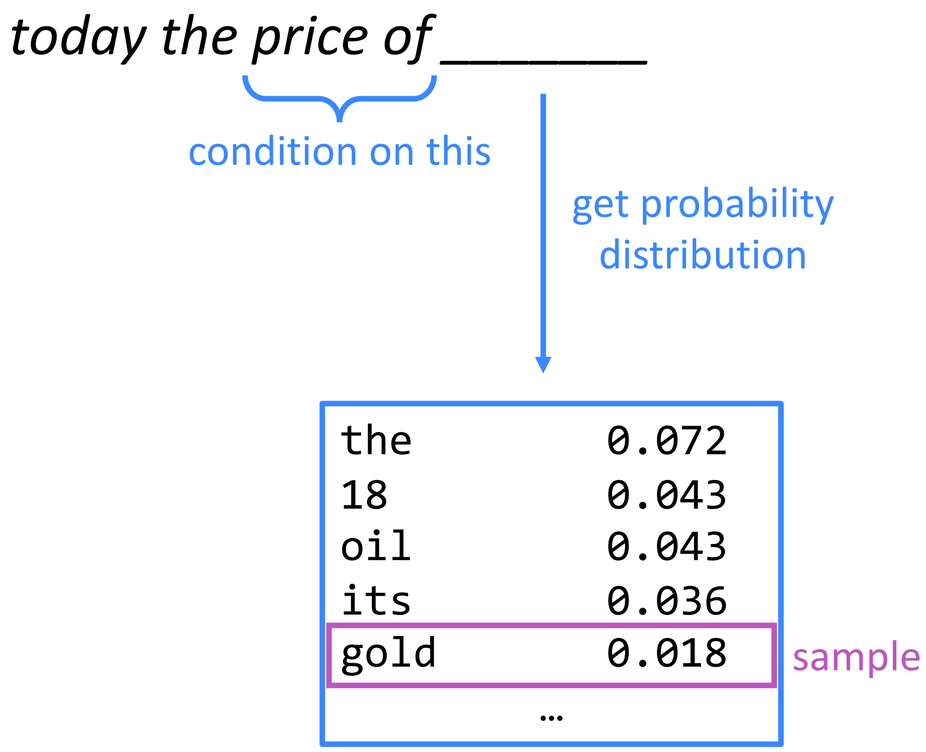
Generating Text with a N-gram Language Model

You can also use a Language Model to generate text.



Generating Text with a N-gram Language Model

You can also use a Language Model to generate text.



Generating Text with a N-gram Language Model

You can also use a Language Model to generate text.

today the price of gold _____

Generating Text with a N-gram Language Model

You can also use a Language Model to generate text.

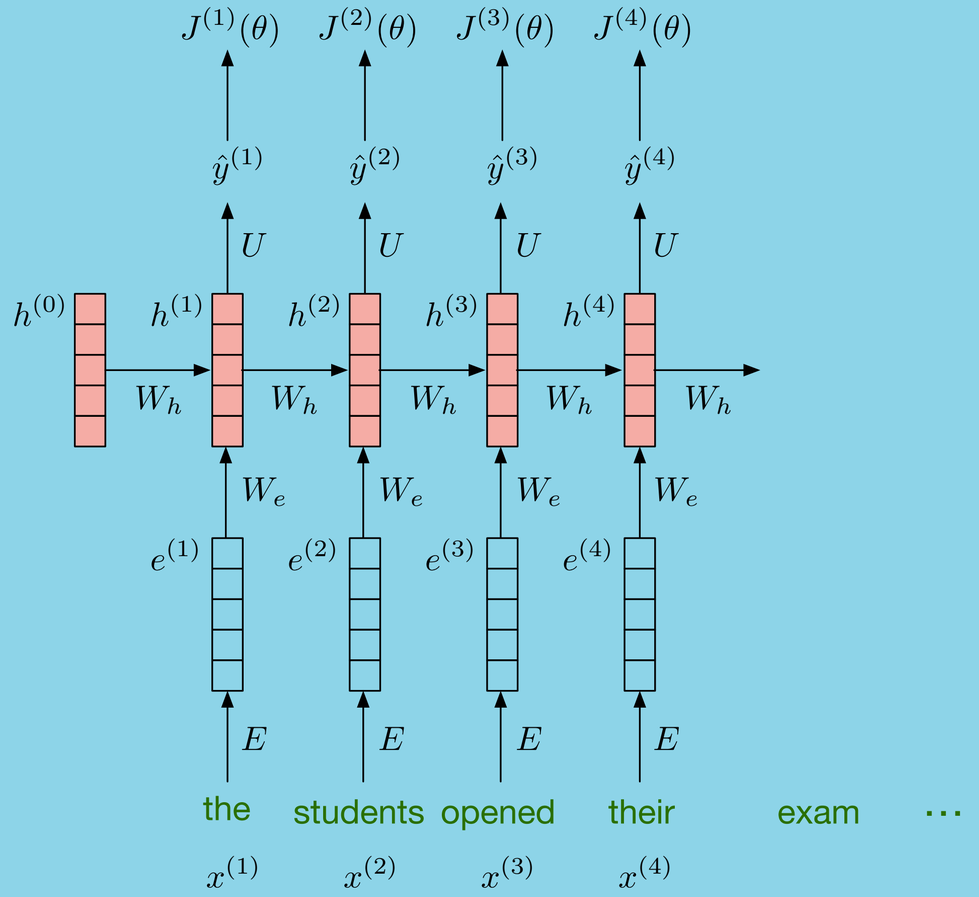
today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

Surprisingly grammatical!

…but **incoherent.** We need to consider more than three words at a time if we want to model language well.

But increasing *n* worsens sparsity problem, and increases model size…

Neural Language Modeling and RNN



A Fixed-Window Neural Language Model

- Estimate $P(x^{(t+1)}|x^{(t)},\cdots,x^{(t-n+2)})$ not from **count**, but from **neural network prediction**

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

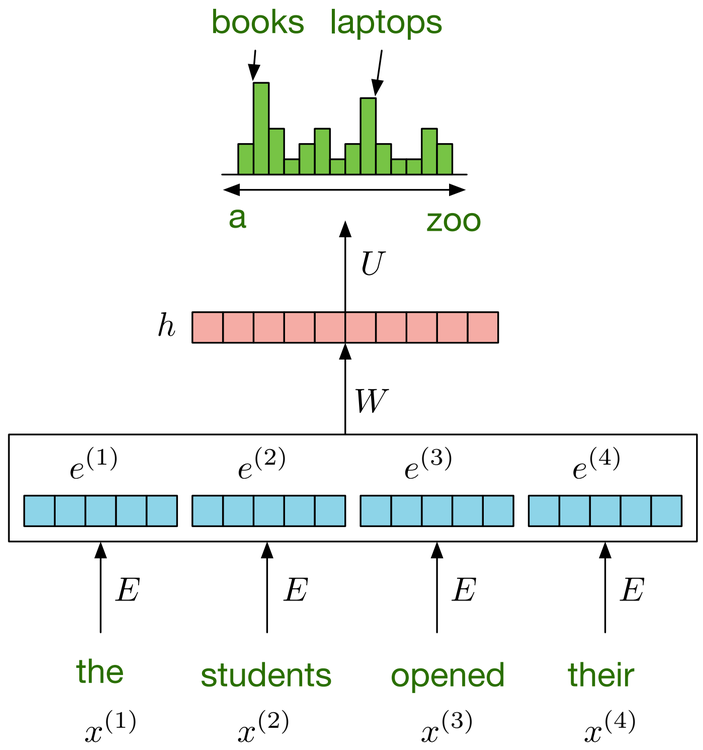
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

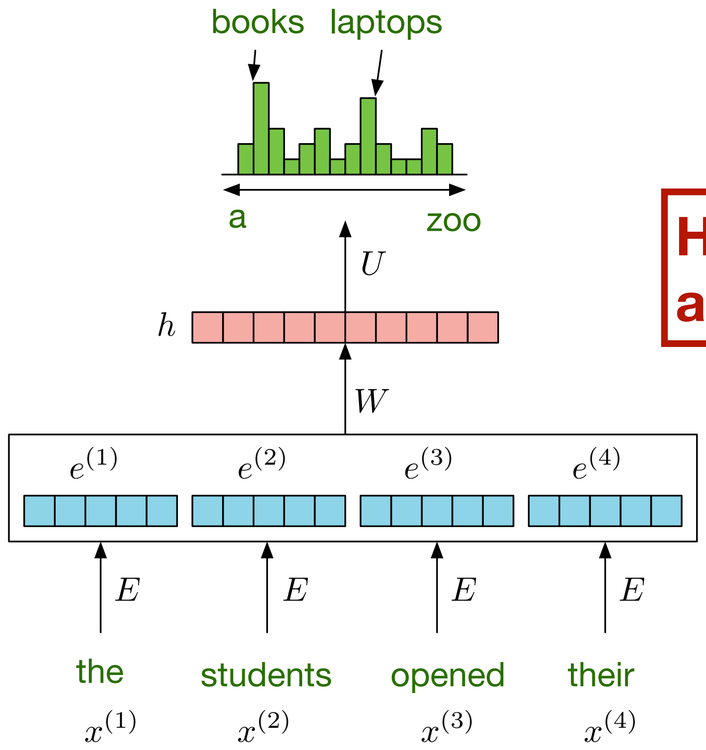
$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



A Fixed-Window Neural Language Model

- **Improvements** to n-gram LM
  - No sparsity problem
  - Don't need to store all observed n-grams

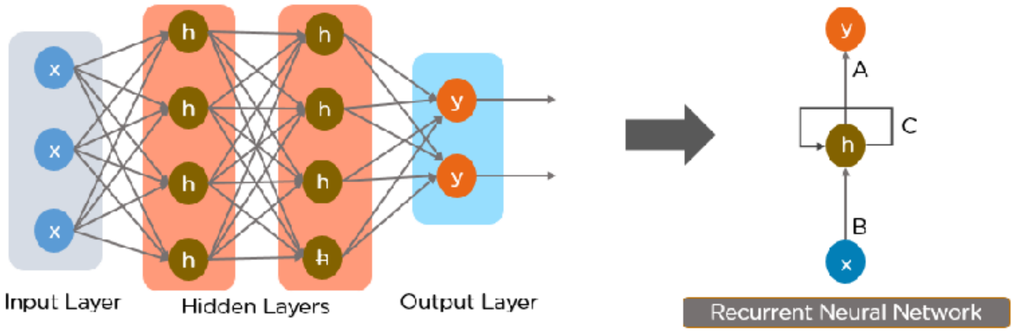
- **Remaining problems**
  - Fixed window is too small
  - Enlarging window enlarges W
  - Words in different positions are multiplied by different W, no symmetry in how the inputs are processed.

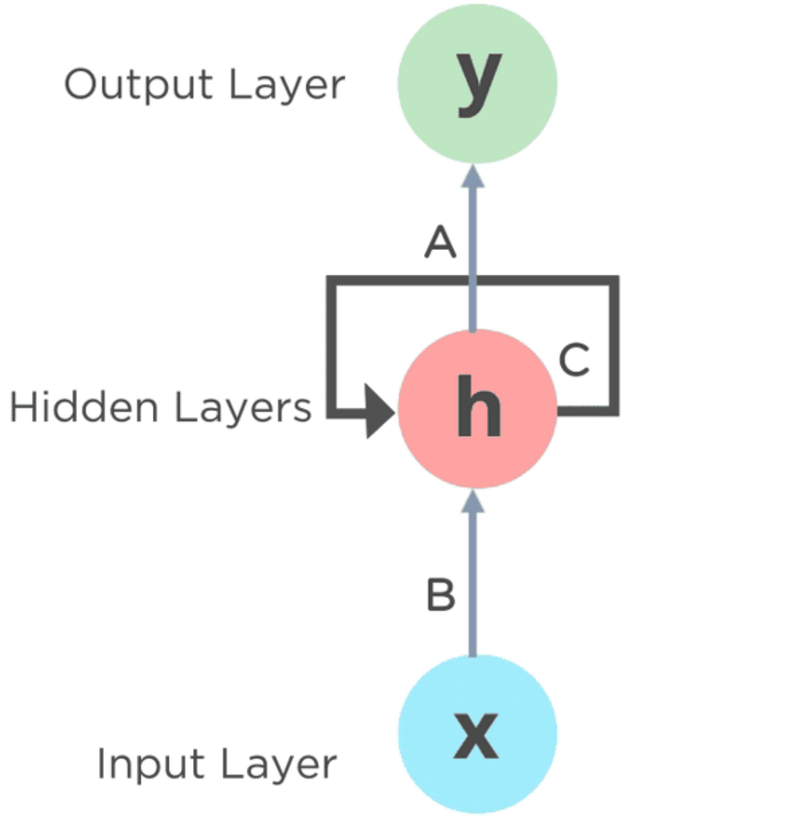


How to process any length input?

Recurrent Neural Networks

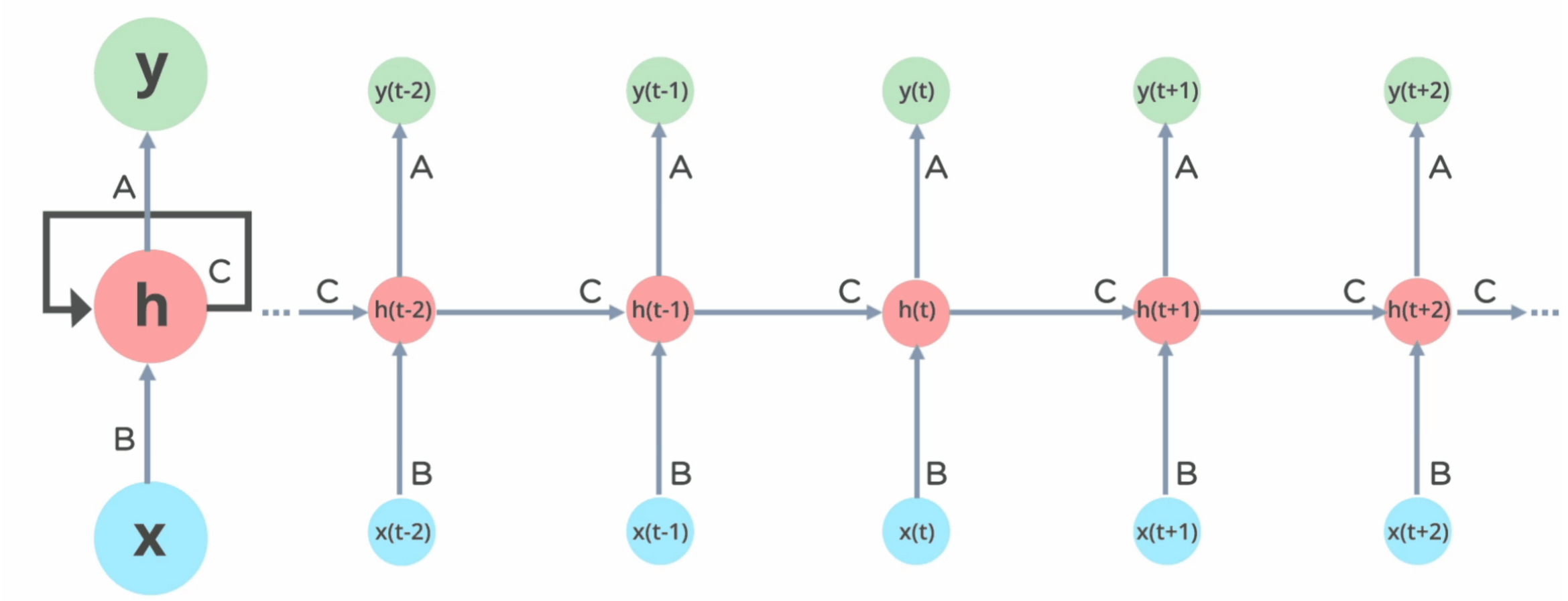
111 Recurrent Neural Networks (RNN)





https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

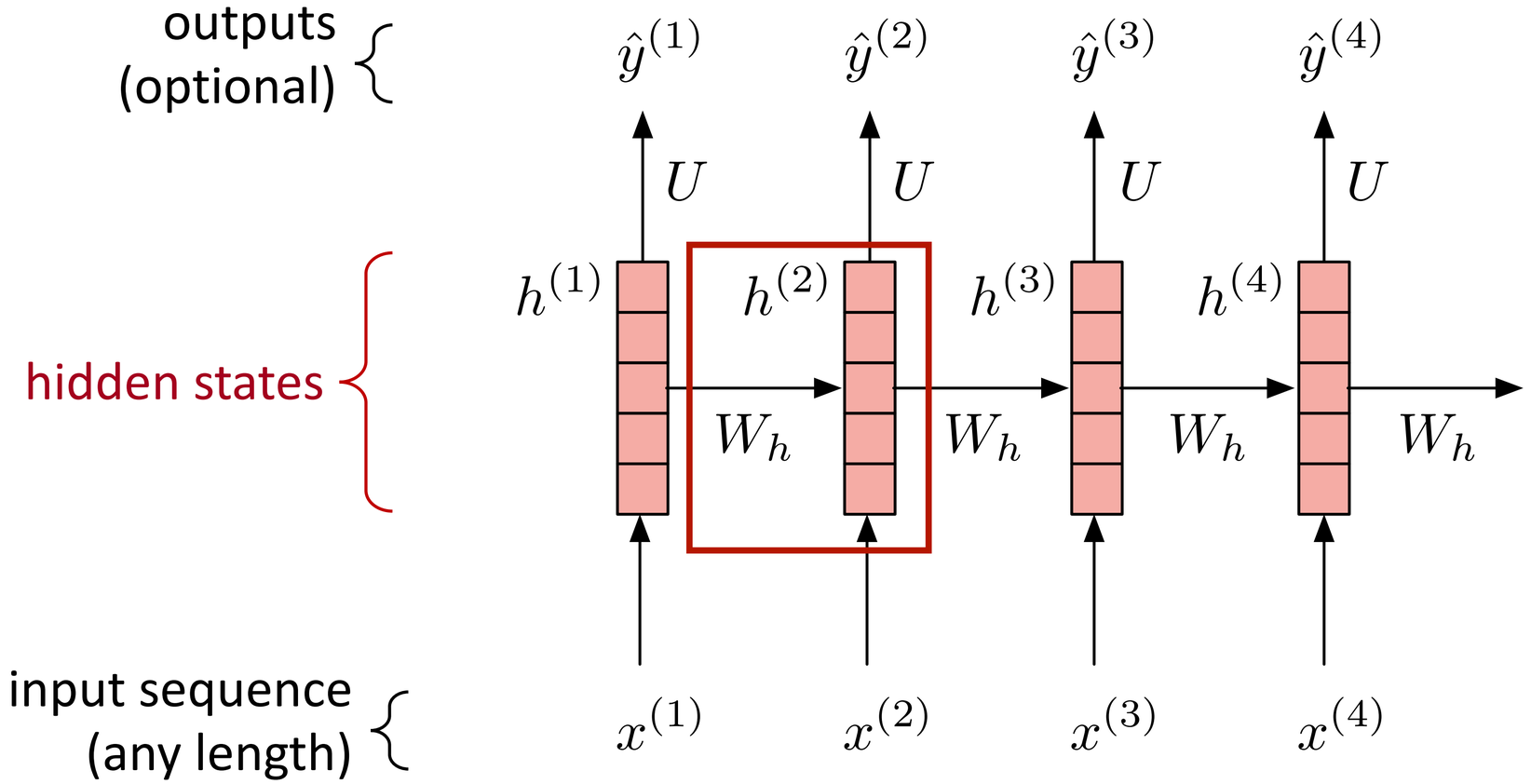
Recurrent Neural Networks (RNN)



https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Recurrent Neural Networks (RNN)

- RNN is a family of neural architectures



Core idea: Apply the same weights W repeatedly

A Simple RNN Language Model

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

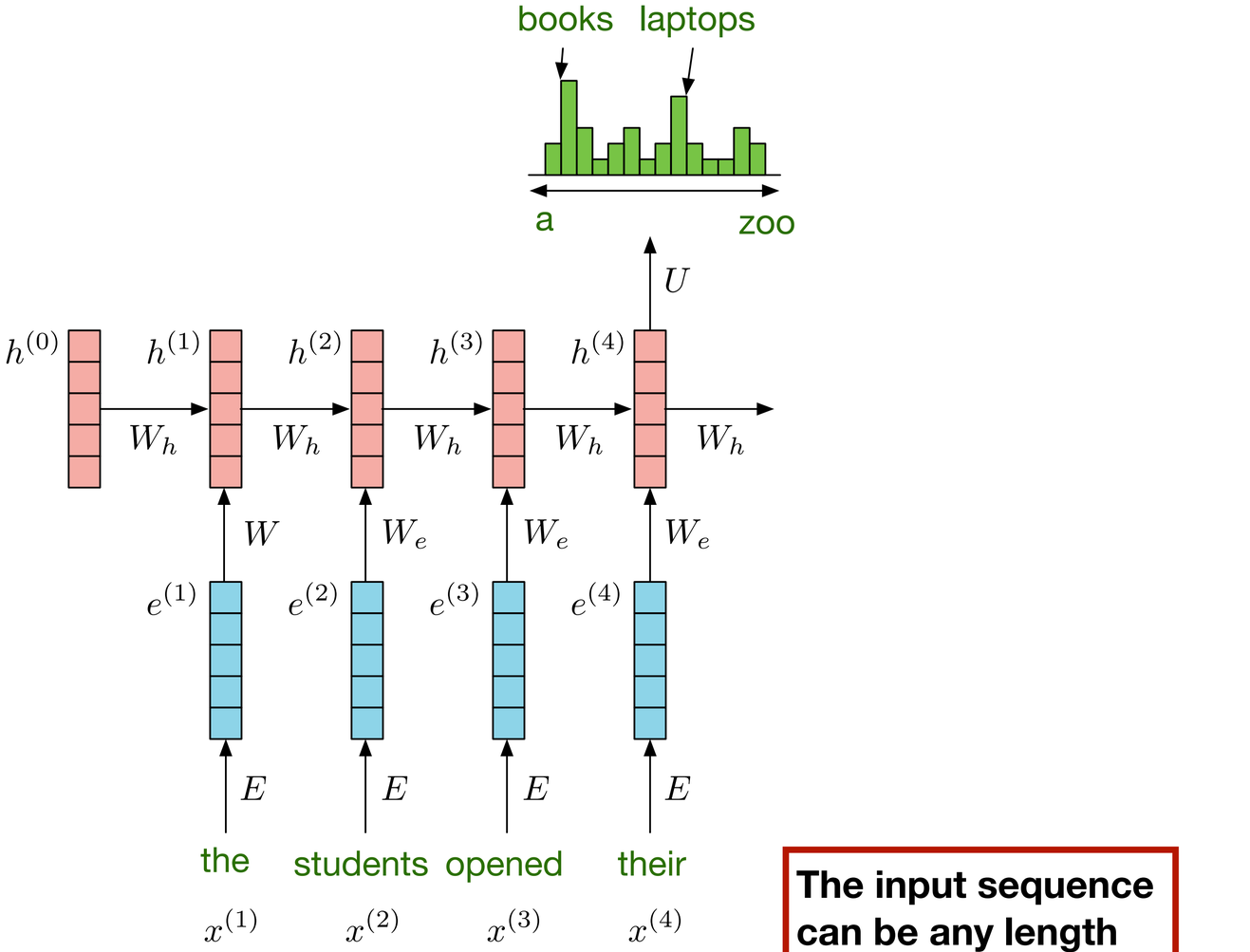
word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$

$$\hat{y}^{(4)} = P(x^{(5)}|\text{the students opened their})$$

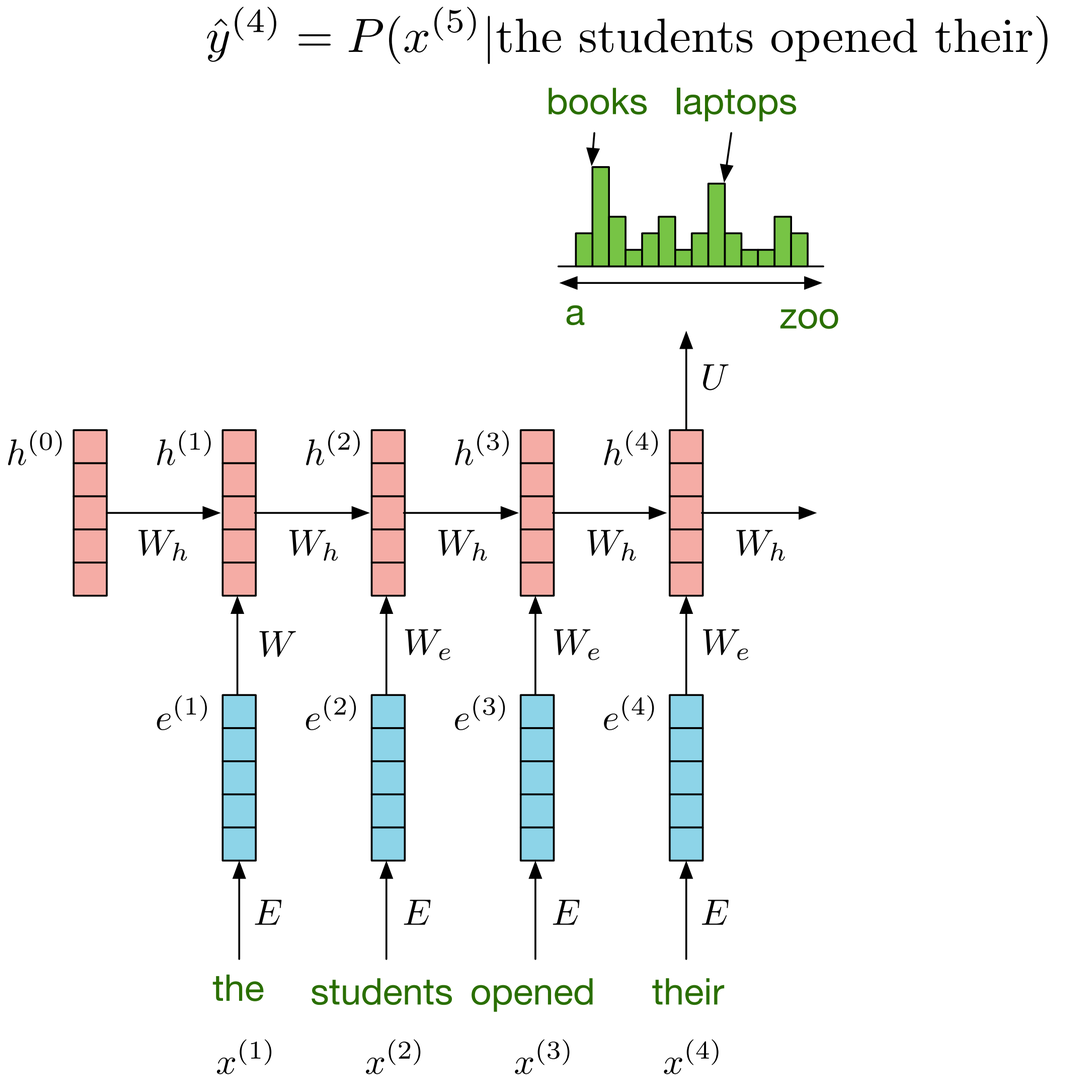


The input sequence can be any length

RNN Language Models

- **RNN advantages**
  - Can process any length input
  - Computation in step *t* can (in theory) use information from many steps back
  - Model size doesn't increase for longer inputs
  - Same weights applied on each time step, so there is symmetry on how inputs are processed

- **RNN problems**
  - RNN computations are slow
  - In practice, it is difficult to access information from many steps back



RNN Applications

Generating Text with a RNN Language Model

RNN-LM trained on *Harry Potter*:

"Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"

"No idea," said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry's shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn't felt it seemed. He reached the teams too.

Source: https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

Generating Text with a RNN Language Model

RNN-LM trained on recipes:

```
    Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
    Yield: 6 Servings

    2 tb Parmesan cheese -- chopped
    1 c  Coconut milk
    3    Eggs, beaten

Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer
until firm. Serve hot in bodied fresh, mustard, orange and cheese.

Combine the cheese and salt together the dough in a large skillet; add the ingredients
and stir in the chocolate and pepper.
```

Source: https://gist.github.com/nylki/1efbaa36635956d35bcc

Evaluating Language Models

The standard evaluation metric for Language Models is perplexity.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

Lower perplexity is better!

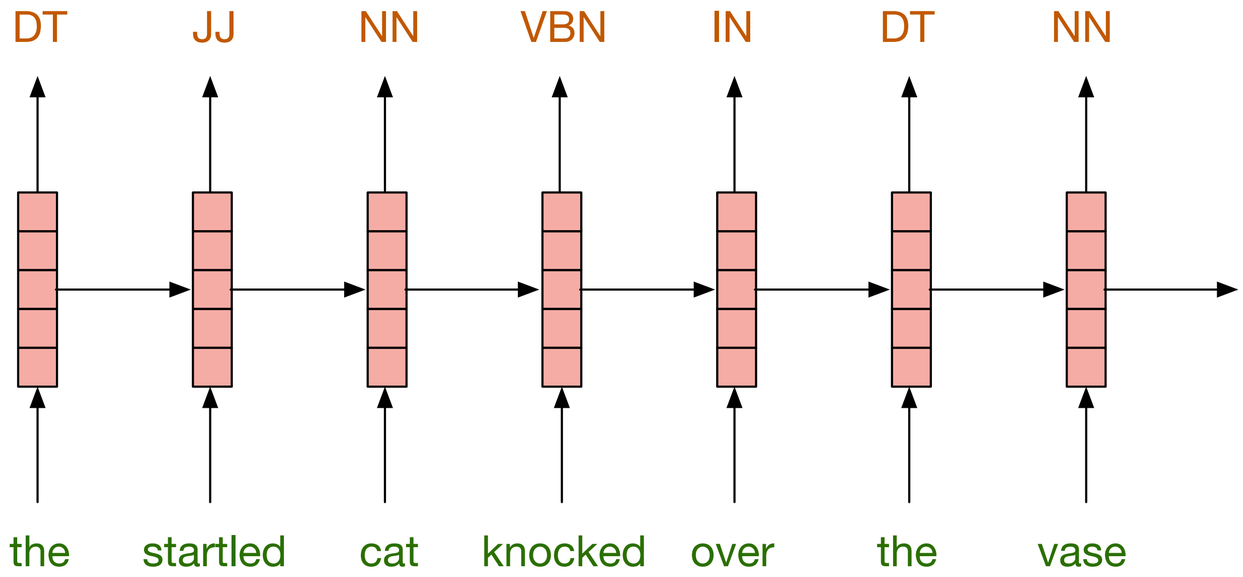
Why Should We Care about Language Modeling

Language Modeling is a benchmark task that helps us measure our progress on understanding language

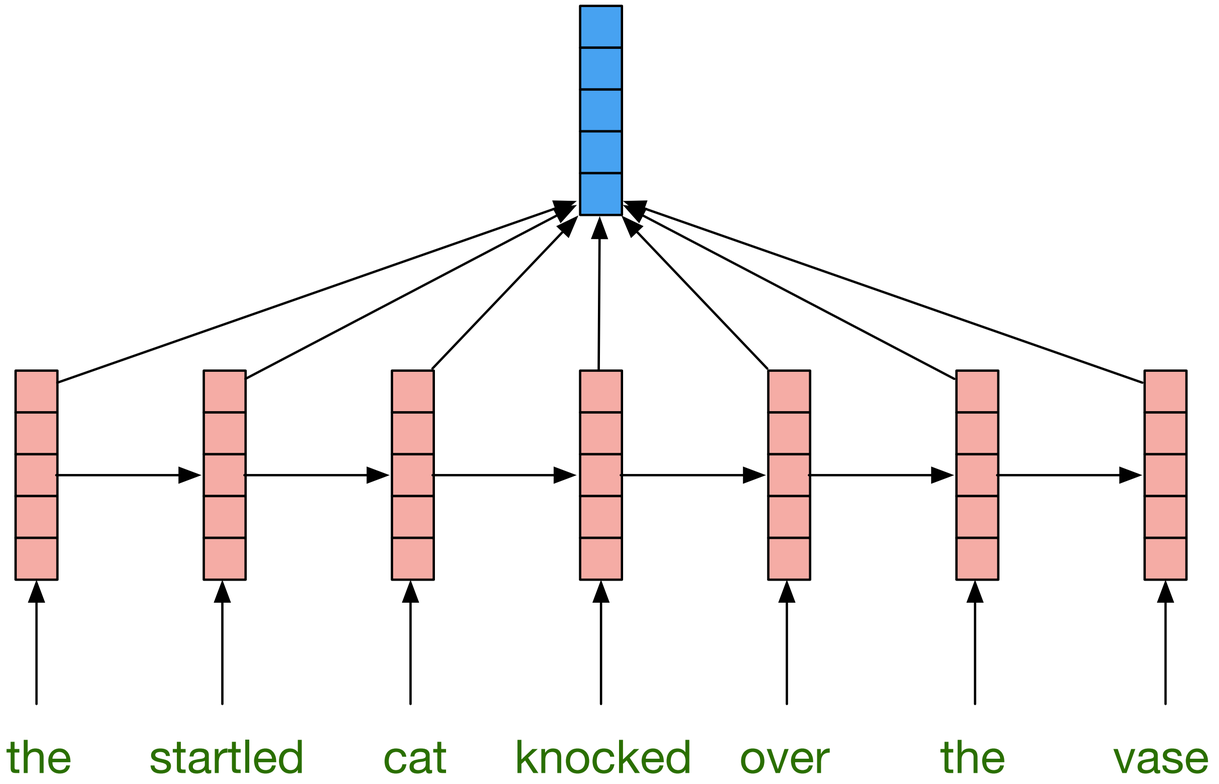
Language Modeling is a subcomponent of many NLP tasks, especially those involving generating text or estimating the probability of text:

- Predictive typing
- Speech recognition
- Handwriting recognition
- Spelling/grammar correction
- Authorship identification
- Machine translation
- Summarization
- Dialogue
- etc.

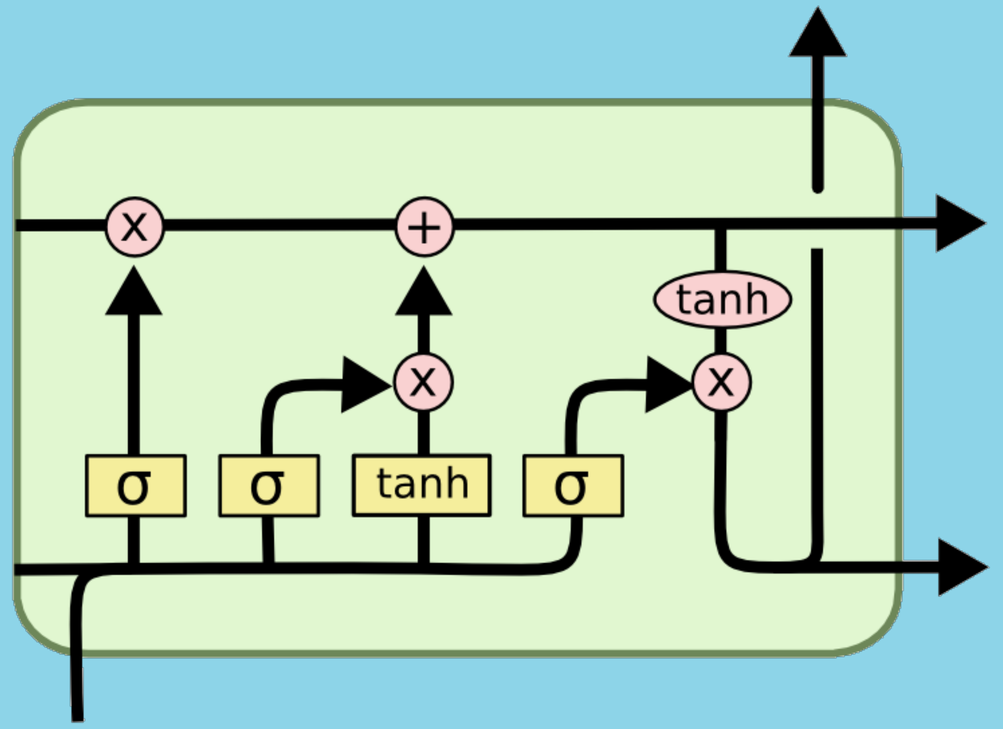
RNNs Can Be Used for Tagging



RNNs Can Be Used as an Encoder Module

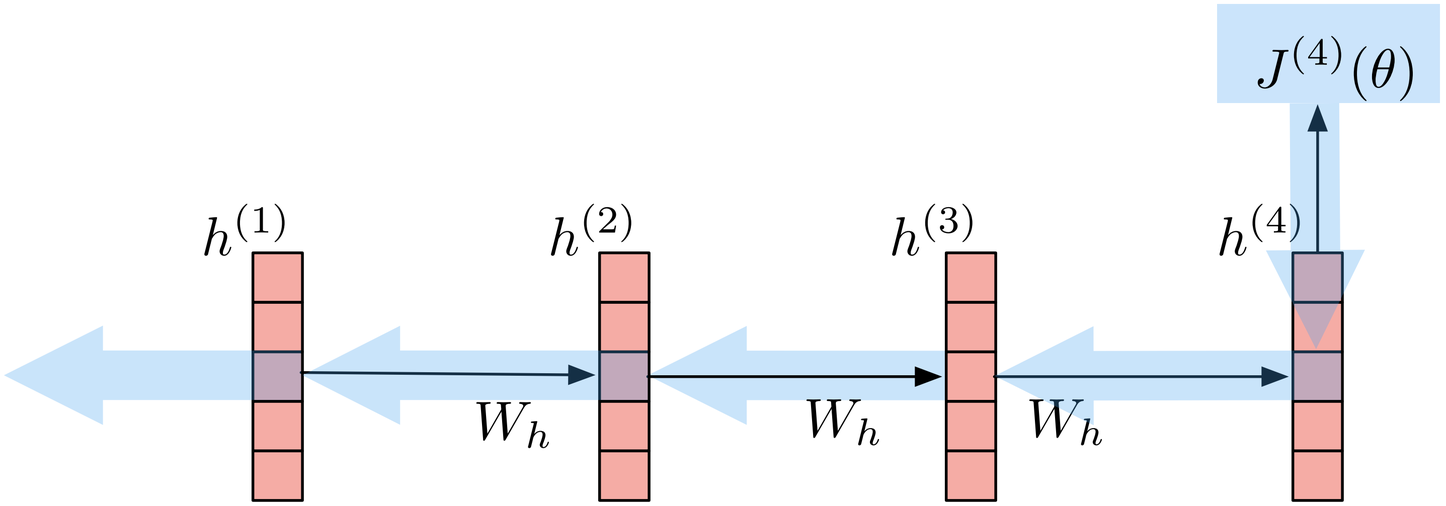


RNN Variants



Problems with RNNs: Vanishing Gradient and Exploding Gradient Problem

Vanishing Gradient Intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
the gradient signal gets smaller and smaller as it backpropagates further

Long Short-Term Memory (LSTM)

- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem.

- On step *t*, there is a hidden state $\boldsymbol{h}^{(t)}$ *and* a cell state $\boldsymbol{c}^{(t)}$
  - Both are vectors length *n*
  - The cell stores long-term information
  - The LSTM can erase, write and read information from the cell

- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors length *n*
  - On each timestep, each element of the gates can be open (1), closed (0), or somewhere in-between.
  - The gates are dynamic: their value is computed based on the current context

"Long short-term memory", Hochreiter and Schmidhuber, 1997. https://www.bioinf.jku.at/publications/older/2604.pdf

Long Short-Term Memory (LSTM)

We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t:

Sigmoid function: all gate values are between 0 and 1

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

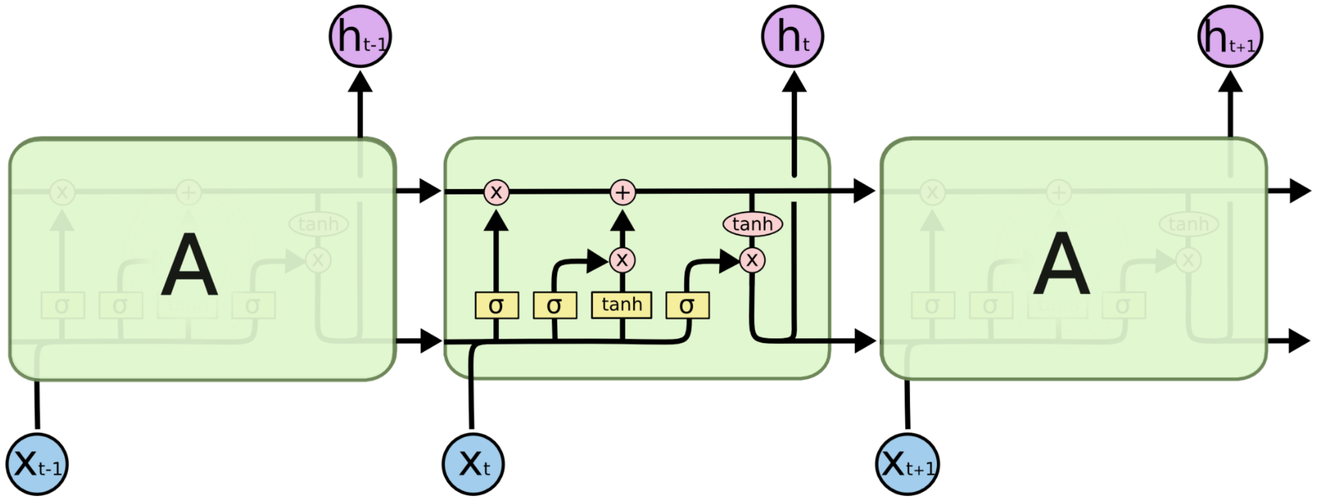
$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}$$

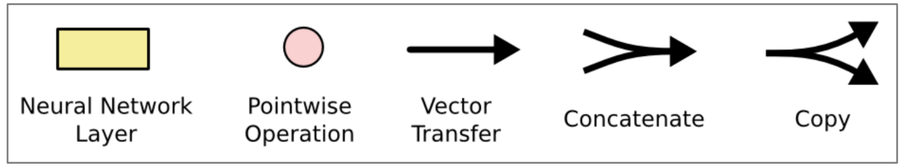
Gates are applied using element-wise product

All these are vectors of same length *n*

Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:

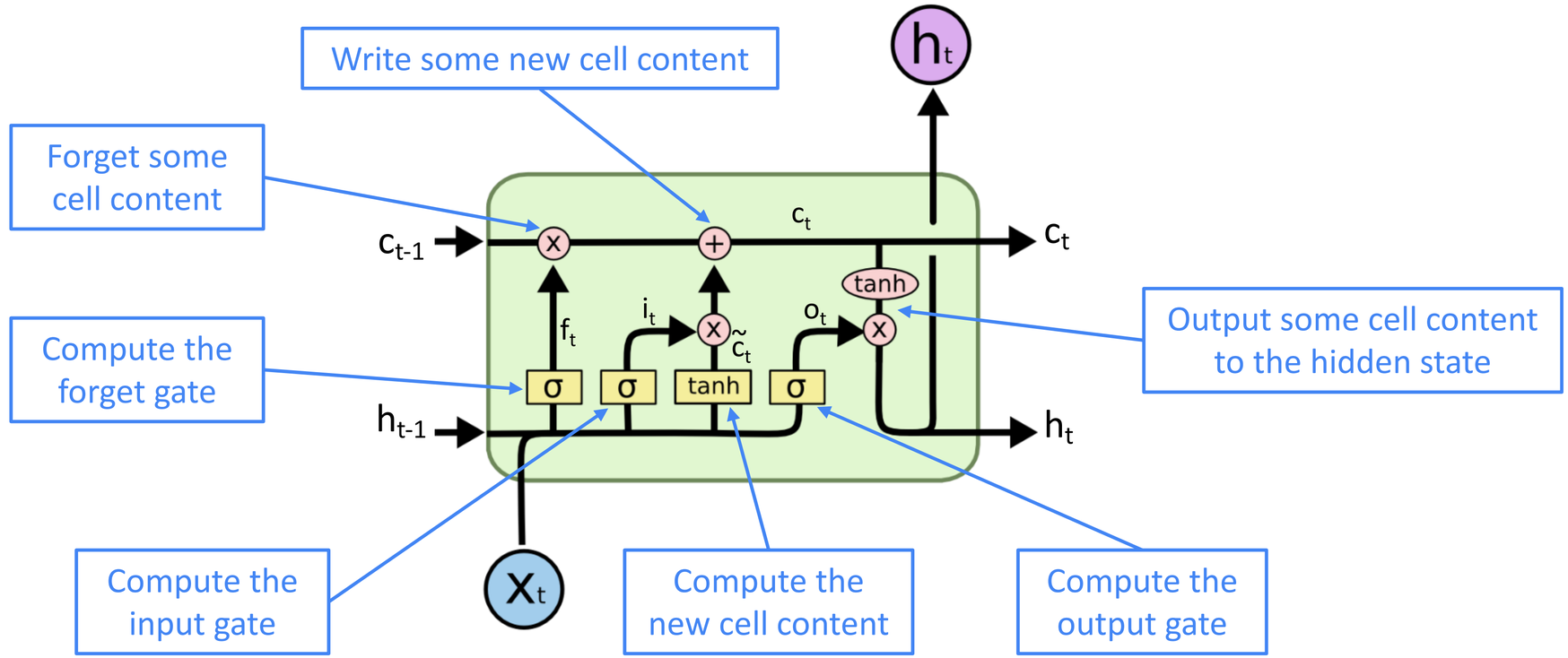


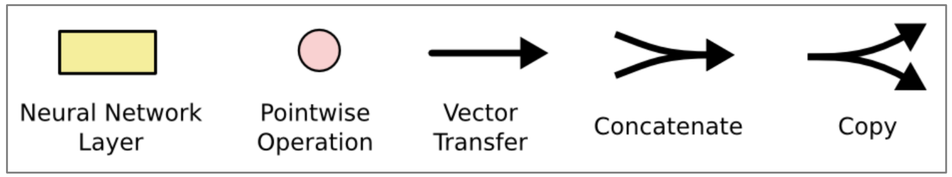


Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:





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

Gated Recurrent Unites (GRU)

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep *t* we have input $\boldsymbol{x}^{(t)}$ and hidden state $\boldsymbol{h}^{(t)}$ (no cell state).

Update gate: controls what parts of hidden state are updated vs preserved

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_u\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

Reset gate: controls what parts of previous hidden state are used to compute new content

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_r\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r\boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$

New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_h(\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{U}_h\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{u}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$$

How does this solve vanishing gradient?
Like LSTM, GRU makes it easier to retain info long-term (e.g. by setting update gate to 0)

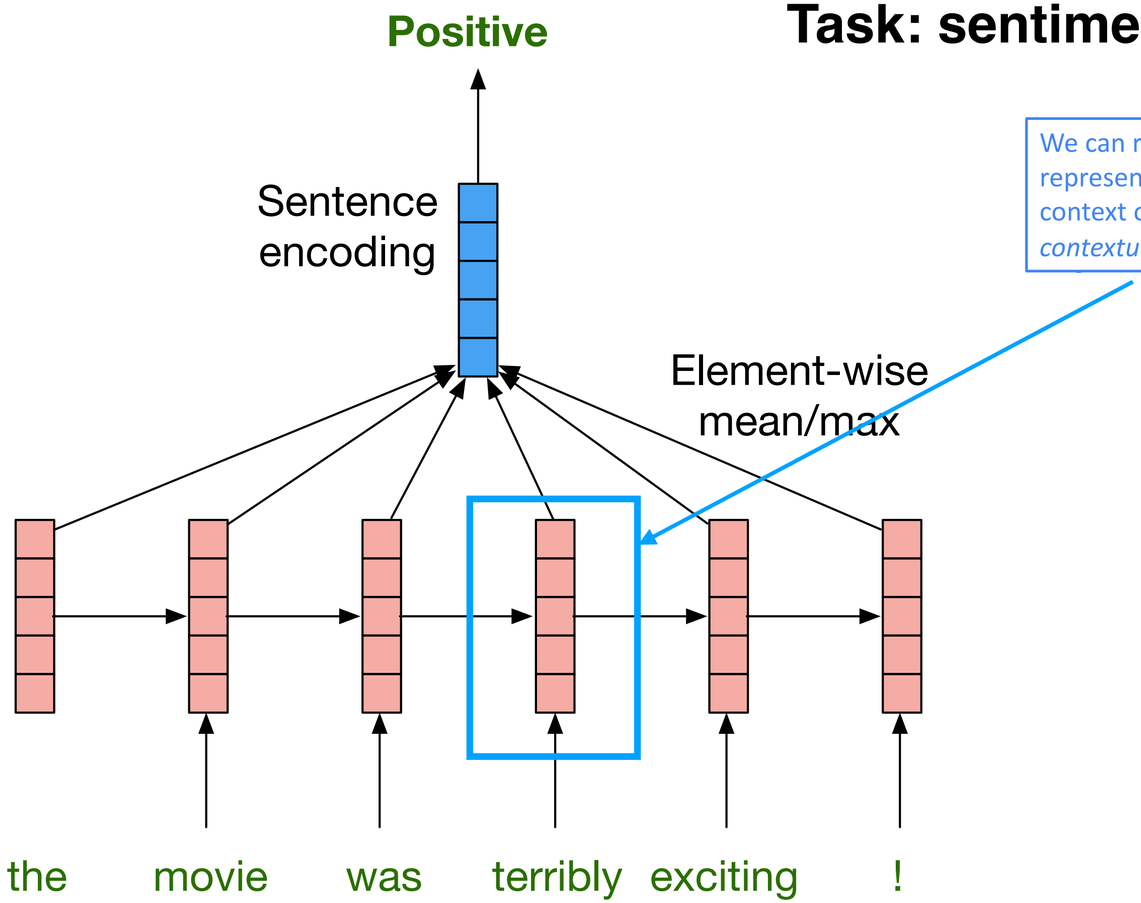
"Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", Cho et al. 2014, https://arxiv.org/pdf/1406.1078v3.pdf

LSTM v.s. GRU

- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
- Rule of thumb: LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data); Switch to GRUs for speed and fewer parameters.

Bi-RNN

Bidirectional RNNs: Motivation



Task: sentiment classification

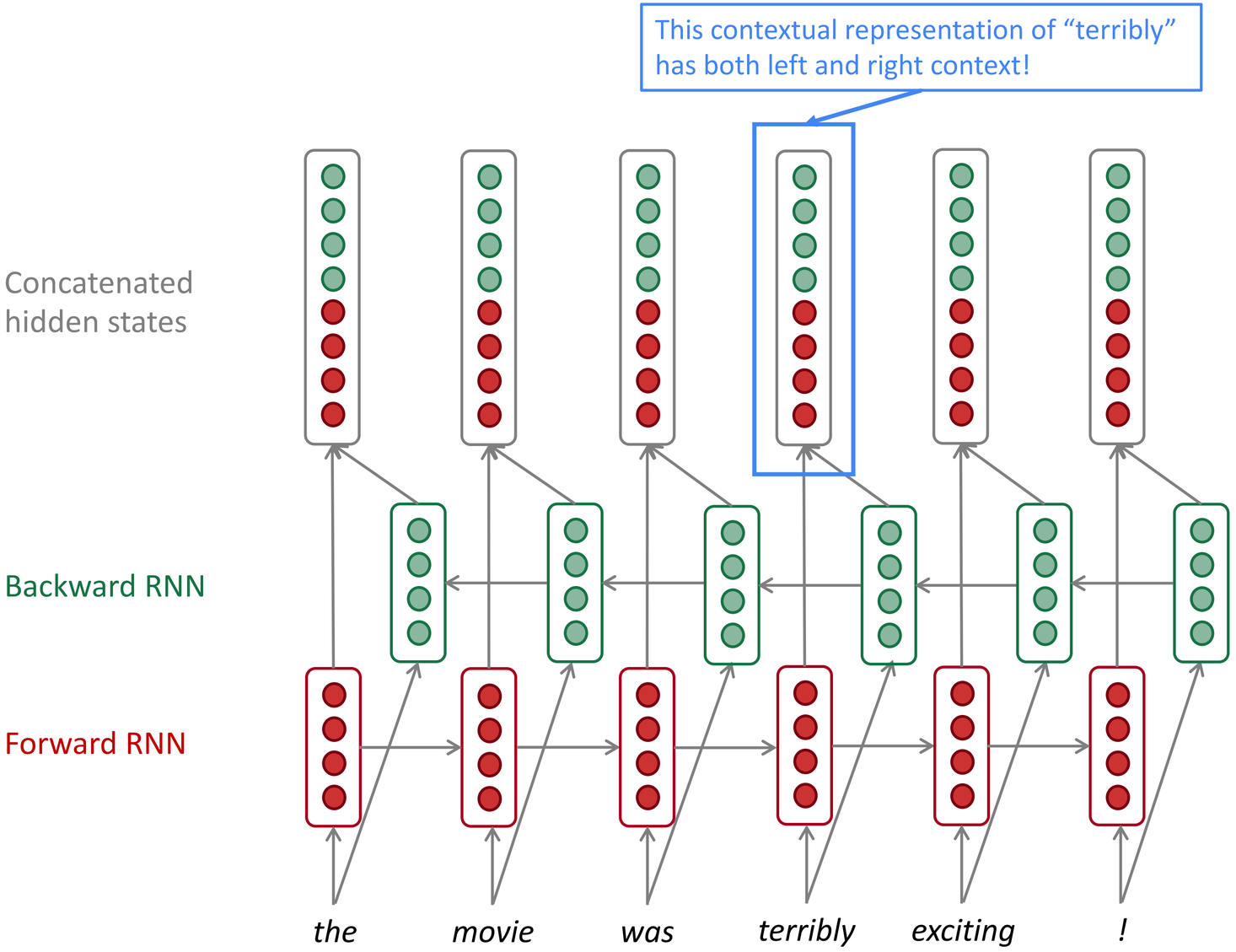
We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation.*

These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)

Bidirectional RNNs



Bidirectional RNNs

On timestep *t*:

This is a general notation to mean "compute one forward step of the RNN" – it could be a vanilla, LSTM or GRU computation.

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

Generally, these two RNNs have separate weights

Concatenated hidden states $\boldsymbol{h}^{(t)} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

We regard this as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network.

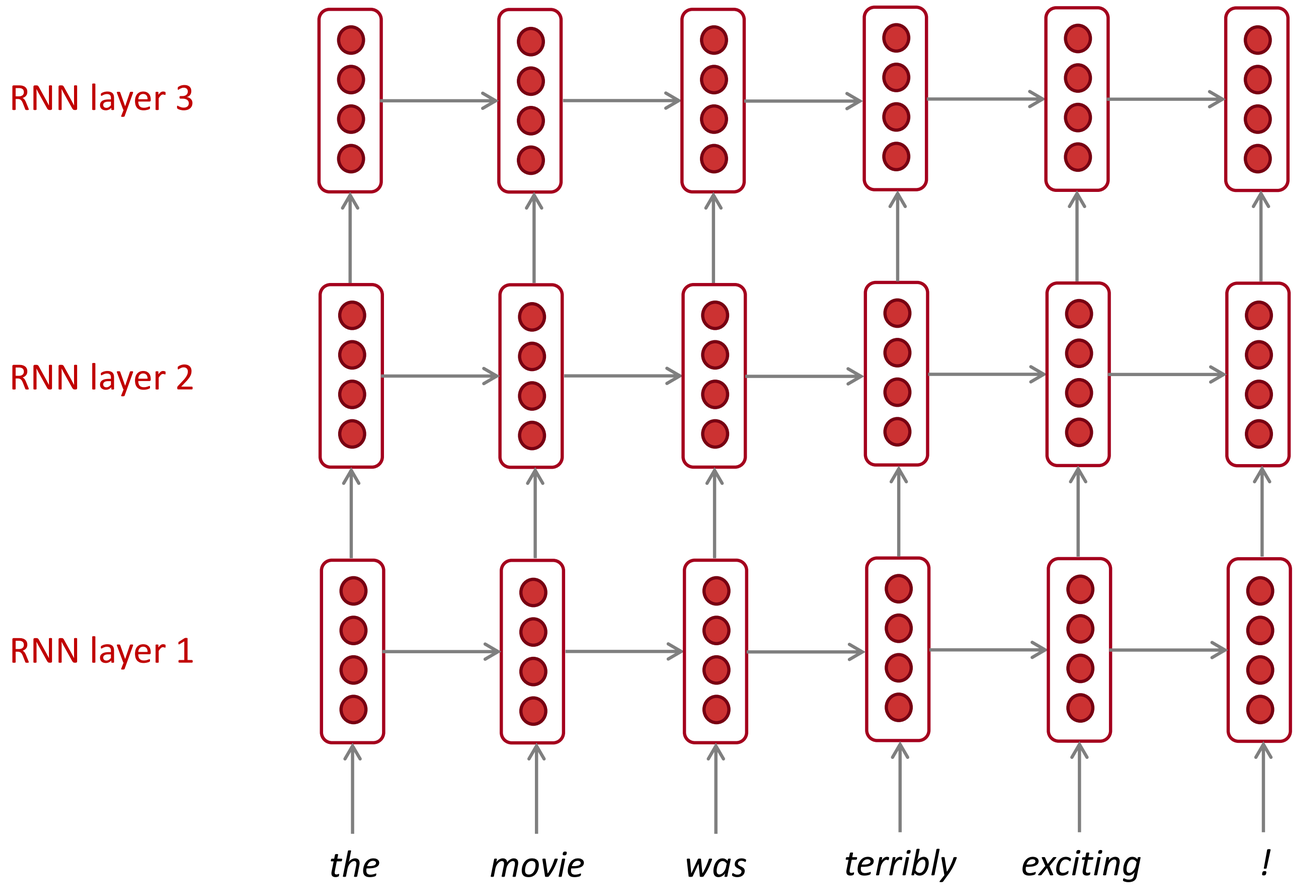
Deep-RNN

Multi-layer RNNs

- RNNs are already “deep” on one dimension (they unroll over many timesteps)
- We can also make them “deep” in another dimension by applying multiple RNNs – this is a multi-layer RNN.
- This allows the network to compute more complex representations
  - The lower RNNs should compute lower-level features and the higher RNNs should compute higher-level features.
- Multi-layer RNNs are also called *stacked RNNs*.

Multi-layer RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i+1*



Thanks! Q&A


Université
de Montréal


rali
recherche appliquée en
linguistique informatique


Mila