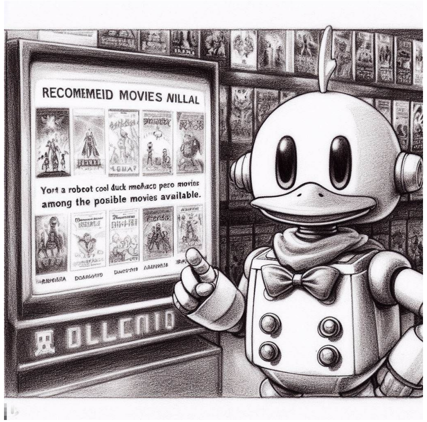
Introduction to Natural Language Processing



AI frameworks

Course: 8 videos

- Introduction
- Tokenization
- Bag-of-words
- Co-occurrence Matrix
- Word embeddings
- Sequential modeling
- Transformers
- Large language models

Introduction

What is a language

A language is a structured system of communication.

The structure of a language is its grammar and the free components are its vocabulary.

Languages are the primary means of communication of humans, and can be conveyed through speech (spoken language), sign, or writing.

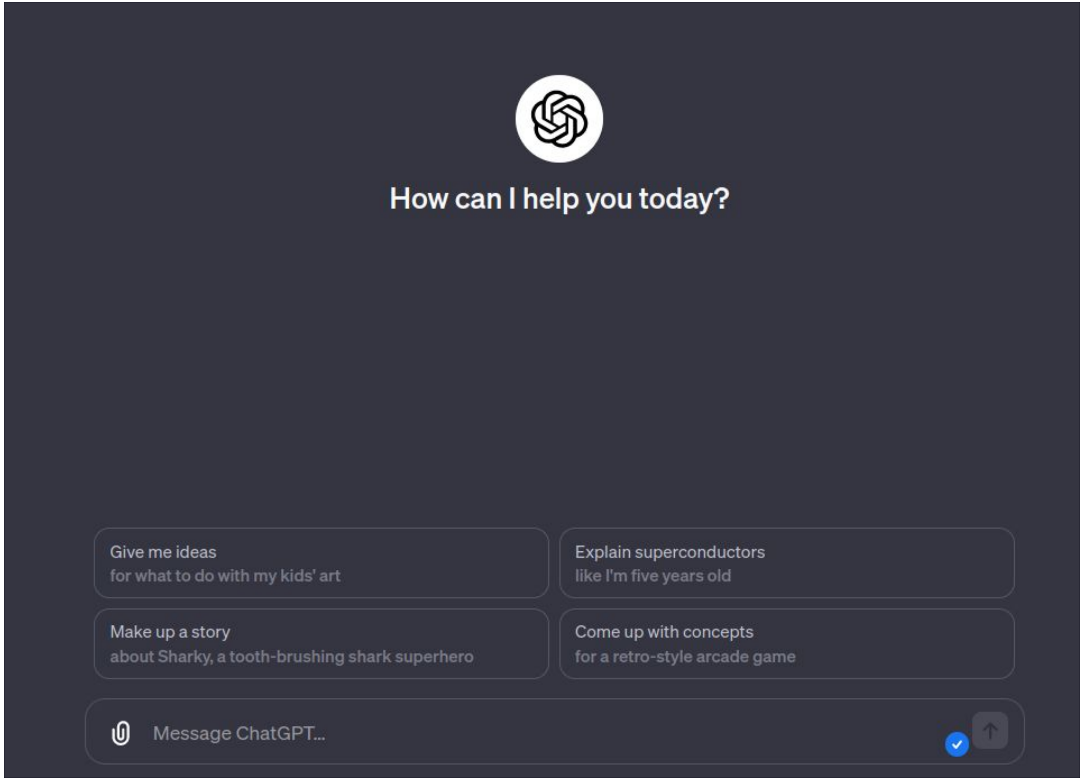
(Source Wikipedia)

Natural Language Processing

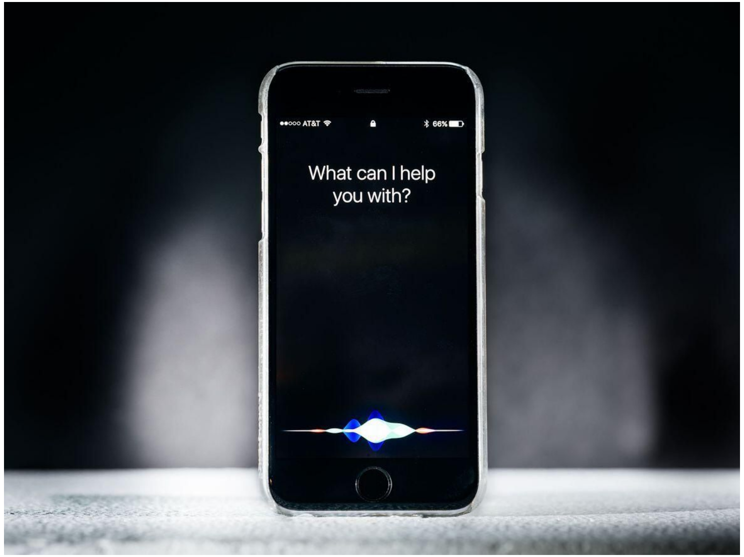
Objectives:

- Design programs able to understand human language as it is spoken and written
- Extract insightful information
- Generate language

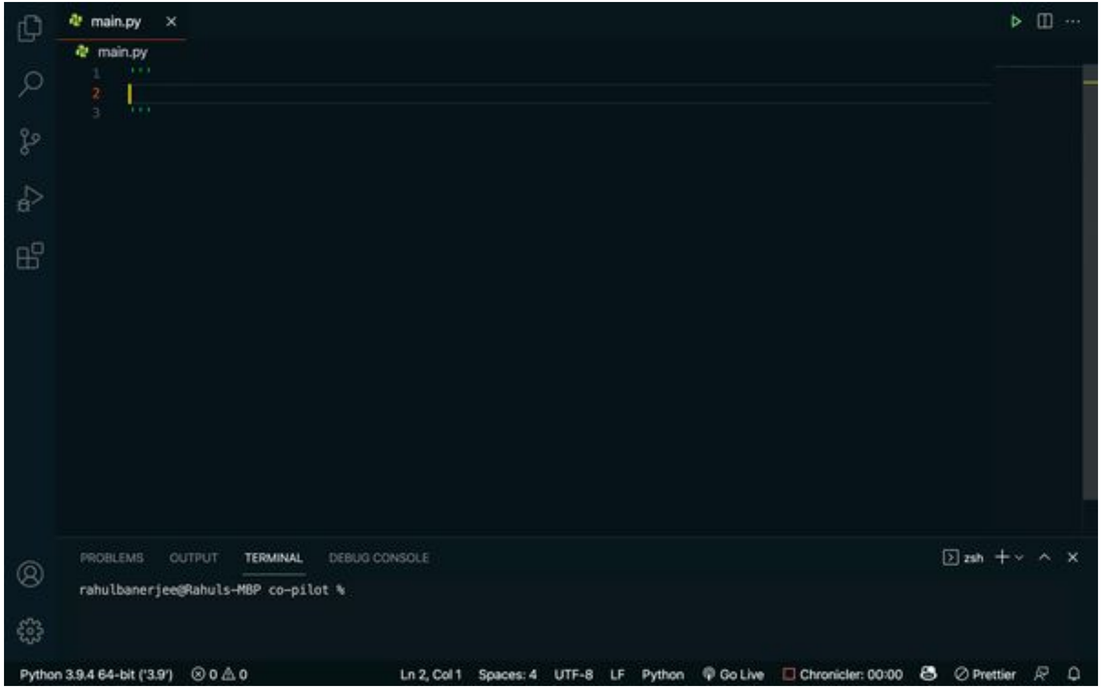
Examples


How can I help you today?
Give me ideas
for what to do with my kids' art
Explain superconductors
like I'm five years old
Make up a story
about Sharky, a tooth-brushing shark superhero
Come up with concepts
for a retro-style arcade game
Message ChatGPT...

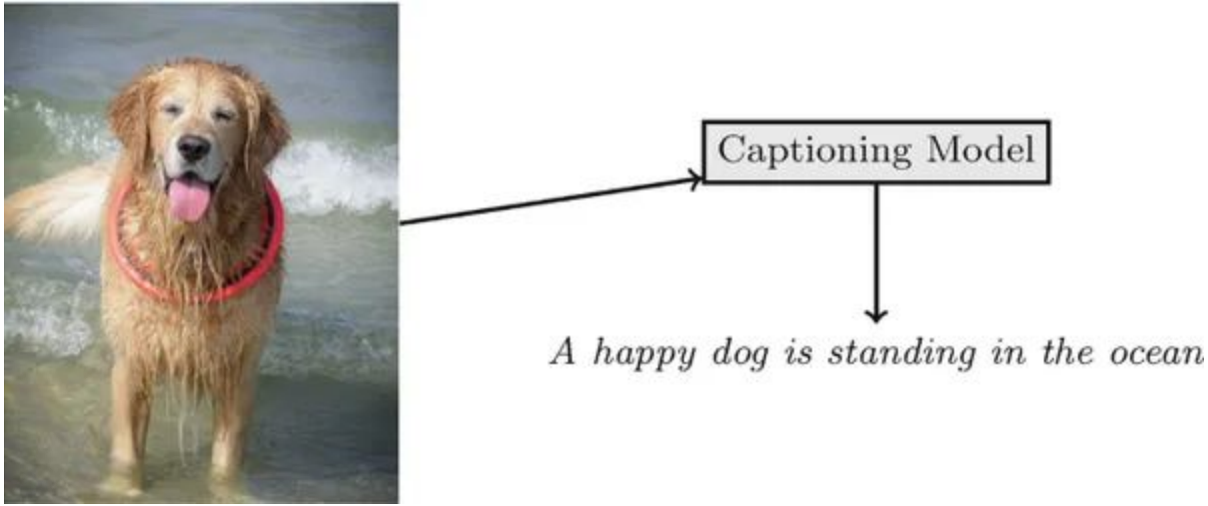
Examples



Examples


main.py
main.py
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rahulbanerjee@Rahuls-MBP co-pilot %
Python 3.9.4 64-bit ('3.9')
Ln 2, Col 1
Spaces: 4
UTF-8
LF
Python
Go Live
Chronicler: 00:00
Prettier

Exemples



Example

DALL·E 2

“A manga-style drawing of the cool duck listening to music while coding.”

NLP tasks

(X,Y) random variables

Estimate the conditional probability $\mathbf{P(Y|\boxed{X=x})}$

- **Word**: *chaise*
- **Word sequence**: [*il, porte, une, chemise*]
- **Document**: *Il porte une chemise jaune. Ses chaussures sont noires …*

NLP tasks

(X,Y) random variables

Estimate the conditional probability **P(Y|X=x)**

- **Word**: *chaise*
- **Word sequence**: [*il, porte, une, chemise*]
- **Document**: *Il porte une chemise jaune. Ses chaussures sont noires …*

- Classification

NLP tasks

(X,Y) random variables

Estimate the conditional probability **P(Y|X=x)**

- **Word**: *chaise*
- **Word sequence**: [*il, porte, une, chemise*]
- **Document**: *Il porte une chemise jaune. Ses chaussures sont noires …*

- Classification
- Translation, Text generation, Question answering, Text summarization, …
- Document ranking, recommendations, …

Representations

Machine learning algorithms need vectorial representations

- Words -> vectors: Voiture -> (0.2, 4 .1, …, -2.3,)
- Documents -> vectors La voiture roule -> (3.3, -2.1, …, 2.5)

We call these vector representation **embeddings**

Two approaches:

- Statistical representations -> Bag of Words
- Learned representations -> Word embeddings

Textual data

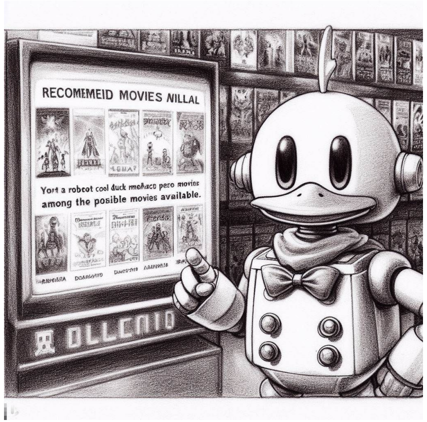
- A **corpus** is a collection of:
  - **documents** which are sequences of :
    - **tokens**

Token: basic unit of discrete data indexed from a vocabulary

- Word
- Sub-word
- A sequence of words or sub-words
- A character
- A symbol

How to identify tokens?

Introduction to Natural Language Processing



AI frameworks

Tokenization

Textual data

- A **corpus** is a collection of:
  - **documents** which are sequences of :
    - **tokens**

Token: basic unit of discrete data indexed from a vocabulary

- Word
- Sub-word
- A sequence of words or sub-words
- A character
- A symbol

How to identify tokens?

Tokenization

Consists in segmenting a document into tokens

Not so trivial!

This is an example of tokenization.

Use whitespace?

[“This”, “is”, “an”, “example”, “of”, “tokenization.”]

Tokenization

Consists in segmenting a document into tokens

Not so trivial!

This is an example of tokenization.

Use whitespace and punctuation?

[“This”, “is”, “an”, “example”, “of”, “tokenization”, “.”]

That’s a problem…

[“That”, “‘“, “s”, “a”, “problem”, “.”, “.”, “.”]

Tokenization

Many other problems:

- Compound words (e.g. pick-pocket, German, …)
- No separators (Chinese, Japanese)
- Casual language: This is a cooool #dummysmiley: :-) :-P <3
- …

Many partial solutions:

- Character level tokenization
- Regular Expression tokenization
- Dictionary based tokenization
- Rule Based Tokenization (Penn TreeBank, Spacy, Moses, TweetTokenizer …)

Lemmatization

Lemmatization is the process of grouping inflected forms together as a single base form

Example:

"**builds**", "**building**", or "**built**" => "**build**"

Stemming

Stemming is the process of reducing inflected words to their word stem, base or root form

Example:

“**programming**”, “**programs**”, “**programmed**”   => “**program**”

Information loss

Stemming and Lemmatization may lead to crucial information loss

Example: sentiment analysis

objective, objection -> object

compliment, complicate -> comply

Subwords tokenization

Principle:

- Frequently used words are unit tokens
- Less frequent words should be decomposed into meaningful subwords
  e.g. “annoyingly” => "annoying" and "ly"
- Rely on model training to discover the most frequent occurring pairs of symbols

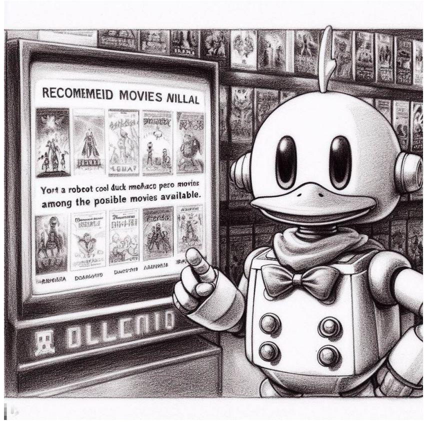
Advantages:

- Reasonable vocabulary size
- Process unknown words

Text Cleaning

- Remove noise (**HTML, punctuation, specific symbols like #, nouns, references, ...**)
- Remove stop words (**and, or, the, ...**)
- Pass a spell checker on your data
- Convert to lowercase

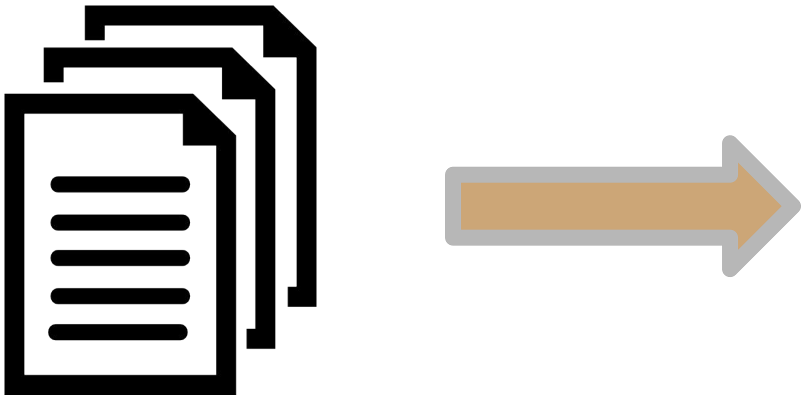
Introduction to Natural Language Processing



AI frameworks

Bag of Words

Vectorizing documents



[0.2, 5.1, ..., -4.1, 2.2]
[-0.8, 2.1, ..., 2.1, 0.7]
[2.1, -3.3, ..., 0.8, 2.5]

Principle :

Use tokens statistics

One hot encoding

(1) I am going to the supermarket

(2) The post office is close to the supermarket

I	am	going	to	the	supermarket	post	office	is	close
1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	0	1	1	1	1

Term-frequency

(1) I am going to the supermarket
(2) The post office is close to the supermarket

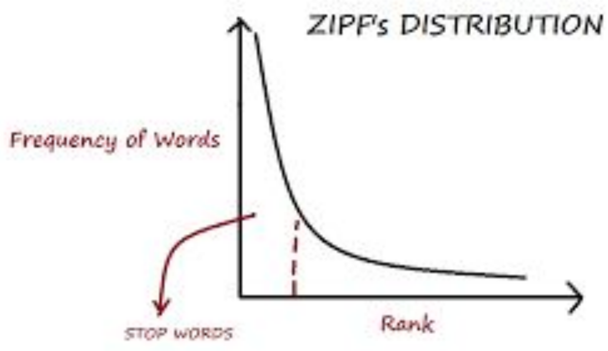
I	am	going	to	the	supermarket	post	office	is	close
1	1	1	1	1	1	0	0	0	0
0	0	0	1	2	0	1	1	1	1

Term-frequency matrix: $tf_{t,d} = |\{t \in d\}|$

TF-IDF

Count based encoding is sensitive to frequent words

- A word present in all documents is not very informative
- A word present in few documents is very informative



Weight term frequency with the **Inverse Document Frequency**:

$$idf_{t,C} = \log\left(\frac{|C|}{|\{d \in C, s.t. t \in d\}|}\right)$$

TF-IDF

(1) This is a big supermarket

(2) This post office is close to a supermarket

This	is	a	to	supermarket	post	office	big	close
0	0	0	0	0	0	0	0.3	0
0	0	0	0.3	0	0.3	0.3	0	0.3

$$tf_{t,d}.idf_{t,C} = |\{t \in d\}| \log\left(\frac{|C|}{|\{d \in C, s.t.t \in d\}|}\right)$$

Bag of words

How to deal with negation or context:

(1) **I do not like carrots**
(2) I do like carrots

I	**do**	**not**	**like**	**carrots**
1	1	1	1	1
1	1	0	1	1

Bag of words

How to deal with negation or context:

(1) **I do not like carrots**
(2) I do like carrots

I	do	not	like	carrots
1	1	1	1	1
1	1	0	1	1

Using n-grams

I	I do	do not	not like	like carrots	carrots
1	1	1	1	1	1

Bag of words pipeline:

<body>I was thinking of eating all the fruits :-)<body>

1. Text cleaning, removing noise, lower case:
   i was thinking of eating all the fruits
2. Remove stop words:
   i was thinking eating all fruits
3. Stemming or lemmatization:
   i was think eat fruit
4. Tokenize:
   [i, was, think, eat, fruit]

Bag of words pipeline:

<body>I was thinking of eating all the fruits :-)<body>

1. Text cleaning, removing noise, lower case:
   i was thinking of eating all the fruits
2. Remove stop words:
   i was thinking eating all fruits
3. Stemming or lemmatization:
   i was think eat fruit
4. Tokenize:
   [i, was, think, eat, fruit]
5. Vectorize (One-hot encoding, count, tf-idf ...)

I	**potato**	**think**	**drink**	**...**	**fruit**
0.1	0	0.5	0	...	0.73
0	0.11	0.4	0.22	...	0.1

Bag of words pipeline:

<body>I was thinking of eating all the fruits :-)<body>

1. Text cleaning, removing noise, lower case:
   i was thinking of eating all the fruits
2. Remove stop words:
   i was thinking eating all fruits
3. Stemming or lemmatization:
   i was think eat fruit
4. Tokenize:
   [i, was, think, eat, fruit]
5. Vectorize (One-hot encoding, count, tf-idf ...)

I	potato	think	drink	...	fruit
0.1	0	0.5	0	…	0.73
0	0.11	0.4	0.22	…	0.1

6. Apply any ML algorithm

Other features

- Number of words, characters, …
- Grammatical categories
- Number of capital letters
- Mean word length
- …

Bag of words

Advantages:

- Easy to set up
- No training required

Drawbacks

- Does not scale with vocabulary size
- Very sparse representation
- No semantic information (synonyms are treated like different words)

Bag of words

Advantages:

- Easy to set up
- No training required

Drawbacks

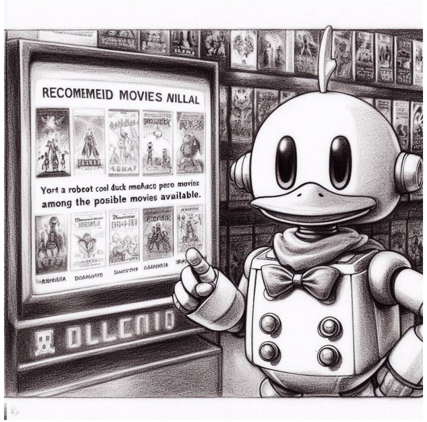
- Does not scale with vocabulary size
- Very sparse representation
- No semantic information (synonyms are treated like different words)

1/ I like this movie

2/ I love this film

I	like	this	movie	love	that	film
1	1	1	1	0	0	0
1	0	0	0	1	1	1

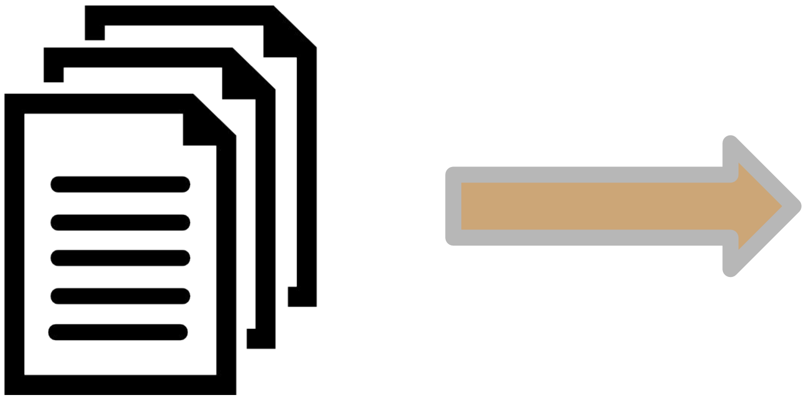
Introduction to Natural Language Processing



AI frameworks

Co-occurrence Matrix

Vectorizing documents



[0.2, 5.1, ..., -4.1, 2.2]
[-0.8, 2.1, ..., 2.1, 0.7]
[2.1, -3.3, ..., 0.8, 2.5]

Principle :

Use tokens statistics

Vectorizing words

car
house
fruit

[0.2, 5.1, ..., -4.1, 2.2]
[-0.8, 2.1, ..., 2.1, 0.7]
[2.1, -3.3, ..., 0.8, 2.5]

Principle :

Use words contexts

A word is defined by its context

Tous les matins je mets du “___” dans mes céréales

-> lait, sucre, …

Tu joues toujours du Mayuri?

A word is defined by its context

Tous les matins je mets du "___" dans mes céréales

-> lait, sucre, ...

Tu joues toujours du Mayuri?

Two main approaches:

- Frequency-based
- Prediction-based

A word is defined by its context

Tous les matins je mets du "___" dans mes céréales

-> lait, sucre, …

Tu joues toujours du Mayuri?

Two main approaches:

- Frequency-based
- Prediction-based

Co-occurrence Matrix

(1) **For breakfast I eat eggs and fruits**
(2) **Go buy some eggs at the supermarket**
(3) **There were no eggs left at the supermarket**

	egg	fruit	...	supermarket
egg	-	1	..	2
fruit	1	-	...	0
...	...	...	...	...
supermarket	2	0	...	-

Co-occurrence Matrix

(1) For breakfast I eat eggs and fruits
(2) Go buy some eggs at the supermarket
(3) There were no eggs left at the supermarket

	egg	fruit	…	supermarket
egg	-	1	..	2
fruit	1	-	…	0
…	…	…	…	…
supermarket	2	0	…	-

Co-occurrence Matrix

- Embedding magnitudes depend on the corpus size
- Frequent words have a strong impact

Co-occurrence Matrix

$$\mathrm{PMI}(\overbrace{w_1}^{\text{A word } w_1}, \underbrace{w_2}_{\text{A word } w_2}) = \log \frac{\overbrace{p(w_1, w2)}^{\text{Probability of } w_1 \text{ and } w_2 \text{ co-occurring}}}{\underbrace{p(w_1)}_{\text{Probability of } w_1 \text{ occurring}}\underbrace{p(w_2)}_{\text{Probability of } w_2 \text{ occurring}}}$$

- Embedding magnitudes depend on the corpus size
- Frequent words have a strong impact

-> Pointwise Mutual Information (PMI)

source: https://medium.com/pew-research-center-decoded/analyzing-text-for-distinctive-terms-using-pointwise-mutual-information-d4c178fa69f2

Co-occurrence Matrix

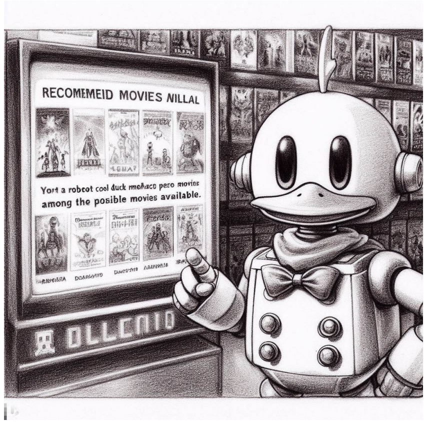
$$PMI(w_1, w_2) = log \frac{\frac{1}{n_{pairs}}\#\{(w_1, w_2)\}}{\frac{1}{n_{word}}\#\{w_1\}\frac{1}{n_{word}}\#\{w_2\})}$$

- Embedding magnitudes depend on the corpus size
- Frequent words have a strong impact

-> Pointwise Mutual Information (PMI)

- Large embedding space **O(nb_words^2)**
- Singular Value Decomposition (SVD)

Introduction to Natural Language Processing



AI frameworks

Word embeddings

Vectorizing words

car
house
fruit

[0.2, 5.1, ..., -4.1, 2.2]
[-0.8, 2.1, ..., 2.1, 0.7]
[2.1, -3.3, ..., 0.8, 2.5]

Principle :

Use words contexts

A word is defined by its context

Tous les matins je mets du "___" dans mes céréales

-> lait, sucre, …

Tu joues toujours du Mayuri?

Two main approaches:

- Frequency-based
- Prediction-based

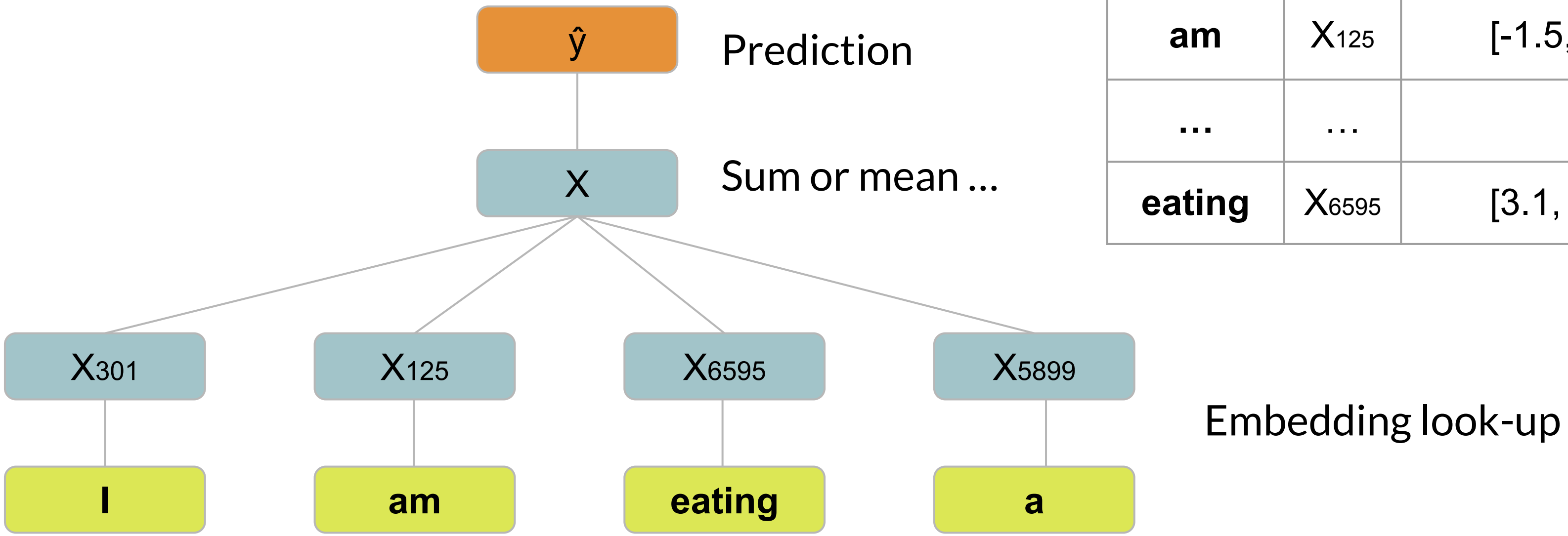
Word2Vec

- Words are mapped to embeddings (**e.g. eating -> [3.1, -2.5,...,1.8]**)

Word	idx	Embeddings
I	X_{301}	[0.2, 1.1,...,-1.2]
...	...	
am	X_{125}	[-1.5, 2.3,...,0.2]
...	...	
eating	X_{6595}	[3.1, -2.5,...,1.8]

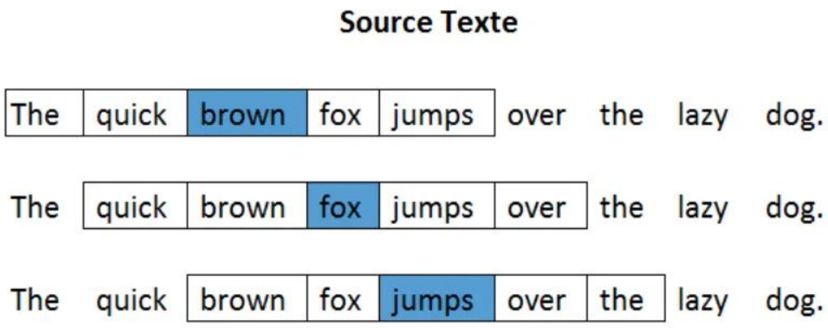
Word2Vec

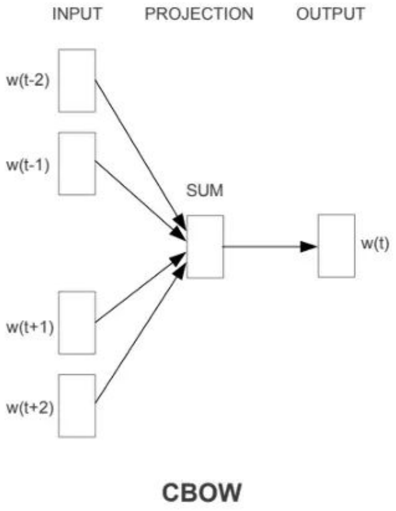
Embeddings are built through shallow neural networks trained to reconstruct linguistic contexts of words



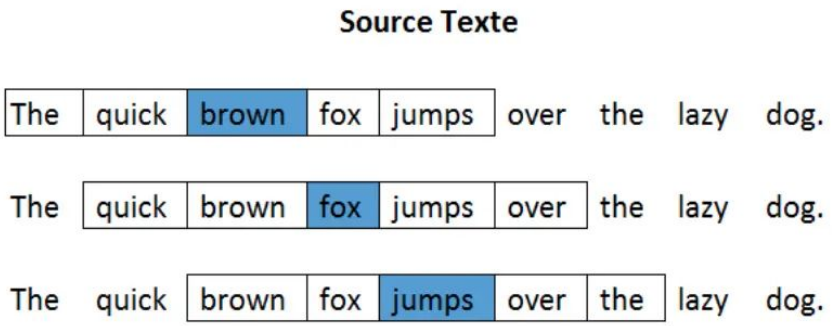
Word	idx	Embeddings
I	X_{301}	[0.2, 1.1,...,-1.2]
...	...	
am	X_{125}	[-1.5, 2.3,...,0.2]
...	...	
eating	X_{6595}	[3.1, -2.5,...,1.8]

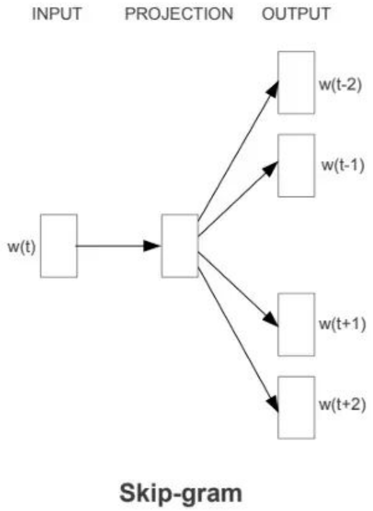
Word2Vec


Source Texte
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW

Word2Vec


Source Texte
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.


INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

Negative sampling

- Predict word given a context is a classification task

  Softmax: $p\left(w_O \mid c_I\right)=\frac{\exp \left({v'_{w_O}}^T v_{c_I}\right)}{\sum_{w=1}^W \exp \left(v_w'^T v_{c_I}\right)}$

Negative sampling

- Predict word given a context is a classification task

  Softmax: $p\left(w_O \mid c_I\right)=\frac{\exp\left({v'_{w_O}}^T v_{c_I}\right)}{\sum_{w=1}^{W}\exp\left(v'^T_w v_{c_I}\right)}$

- Denominator is expensive to compute

Negative sampling

- Predict word given a context is a classification task

  Softmax: $p\left(w_O \mid c_I\right) = \dfrac{\exp\left({v'_{w_O}}^T v_{c_I}\right)}{\sum_{w=1}^{W} \exp\left(v'^T_w v_{c_I}\right)}$

- Denominator is expensive to compute

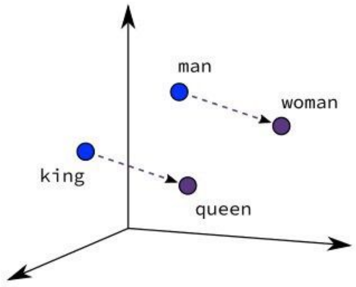
  Solution: **Negative Sampling:**

$$\underbrace{\log \sigma\left({v'_{w_O}}^T v_{w_I}\right)}+\underbrace{\sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)}\left[\log \sigma\left(-{v'_{w_i}}^T v_{w_I}\right)\right]}$$

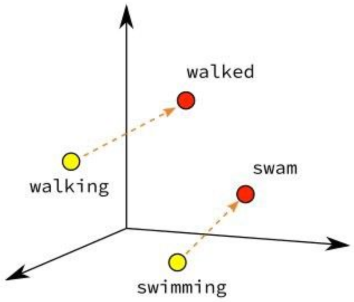
- Maximize the target probability
- Sample k negative samples and minimize their probability

Word2Vec

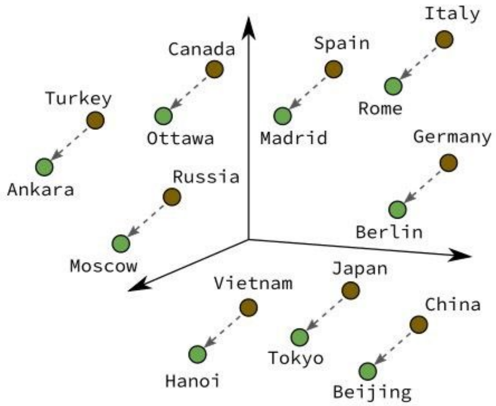
Word with similar **semantic** are located close to one another in the **latent space**



Male-Female



Verb Tense



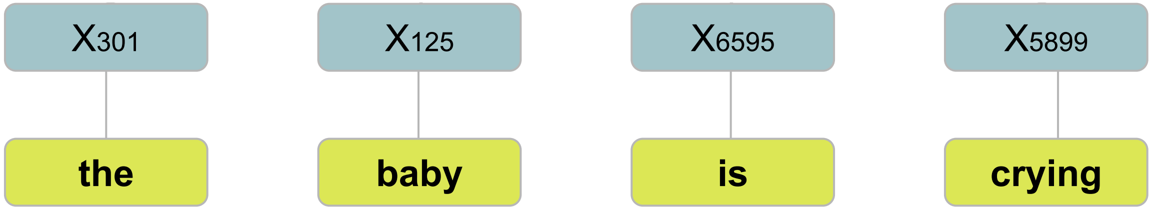
Country-Capital

Word2Vec: Analogies

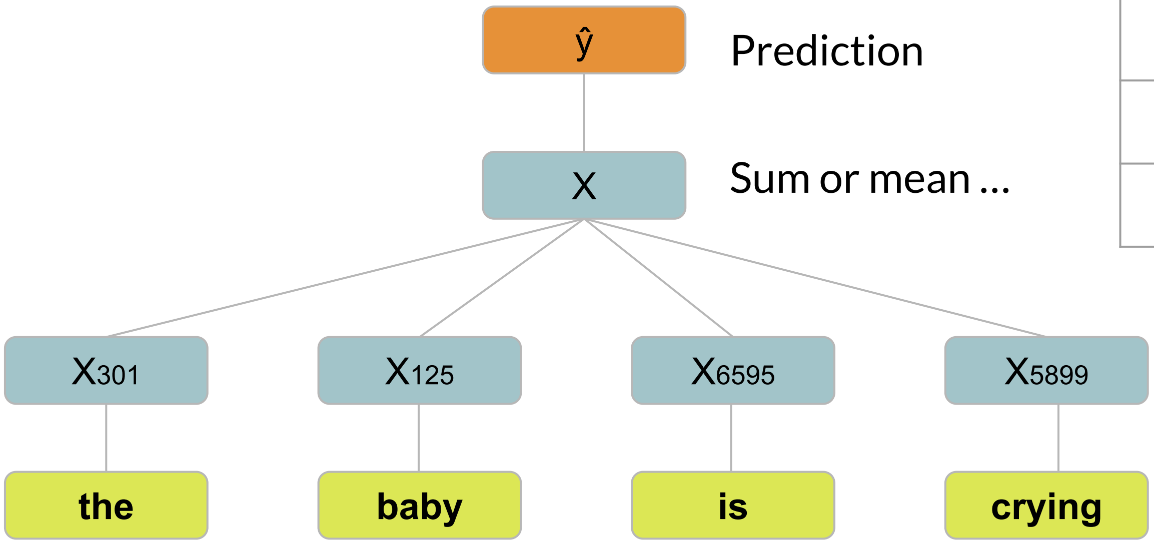
source:https://jalammar.github.io/illustrated-word2vec/

How to get document embedding?

Word	idx	Embeddings
the	X_{301}	[0.2, 1.1,...,-1.2]
...	...	
baby	X_{125}	[-1.5, 2.3,...,0.2]
...	...	
is	X_{6595}	[3.1, -2.5,...,1.8]

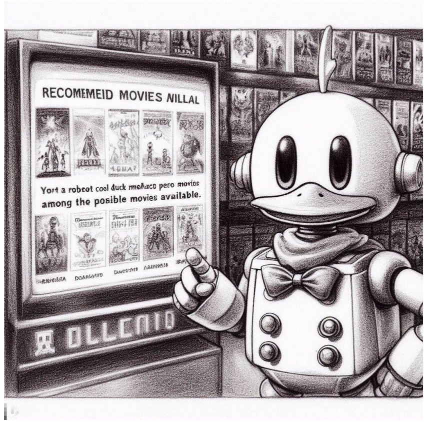


How to get document embedding?



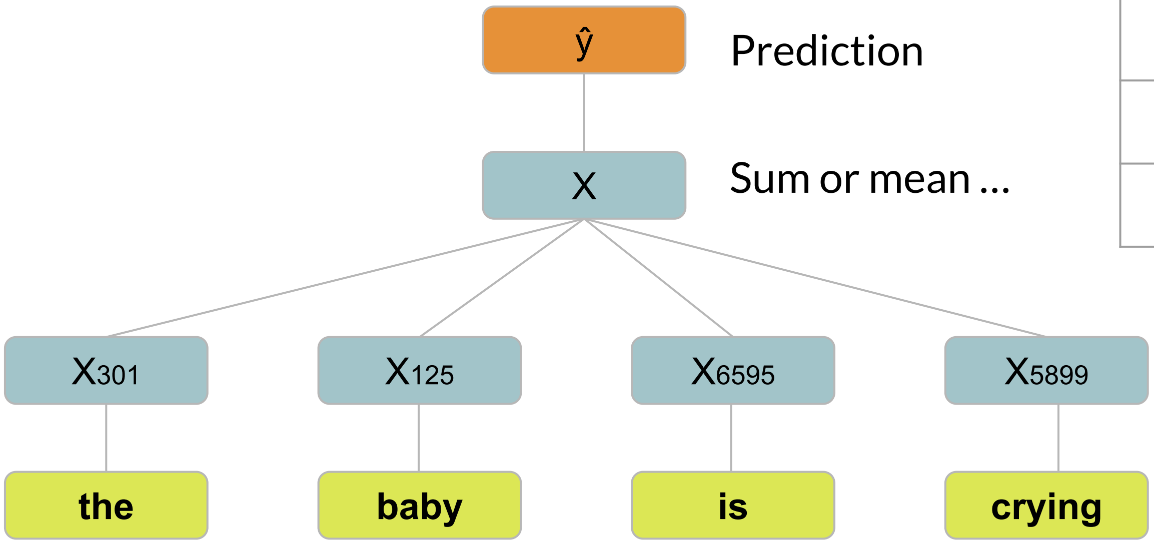
Word	idx	Embeddings
the	X_{301}	[0.2, 1.1,...,-1.2]
...	...	
baby	X_{125}	[-1.5, 2.3,...,0.2]
...	...	
is	X_{6595}	[3.1, -2.5,...,1.8]

Introduction to Natural Language Processing



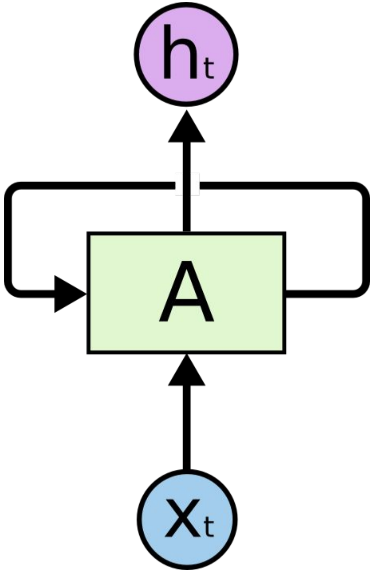
AI frameworks

Sequence modeling



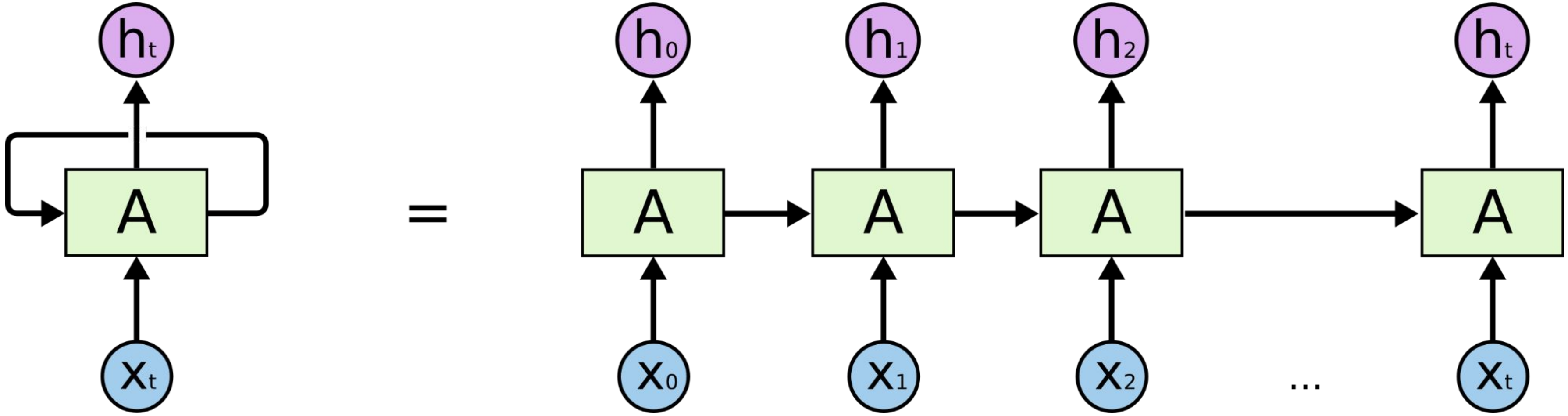
Word	idx	Embeddings
the	X_{301}	[0.2, 1.1,...,-1.2]
...	...	
baby	X_{125}	[-1.5, 2.3,...,0.2]
...	...	
is	X_{6595}	[3.1, -2.5,...,1.8]

Recurrent layer



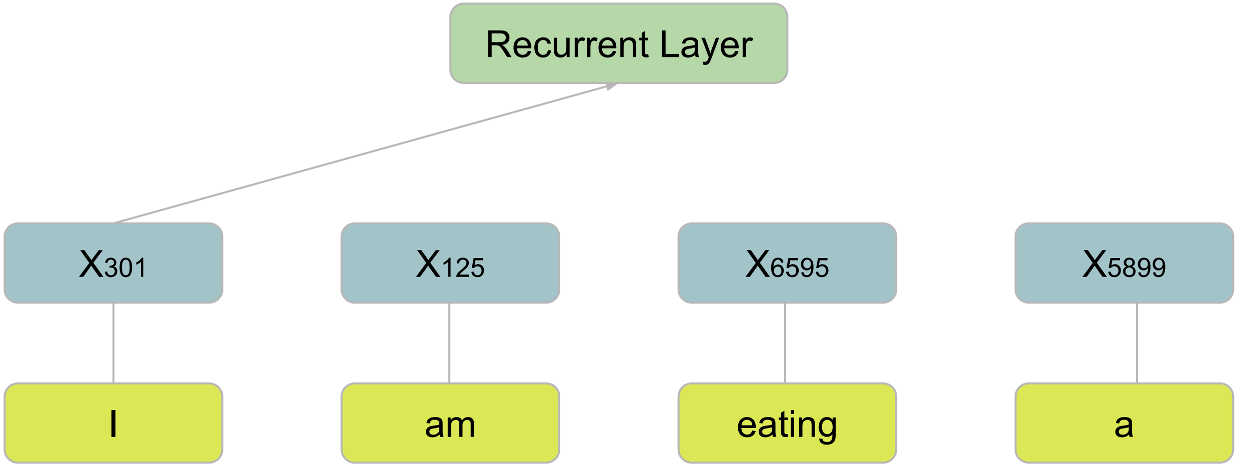
Understanding LSTM Networks by Christopher Olah.

Recurrent layer

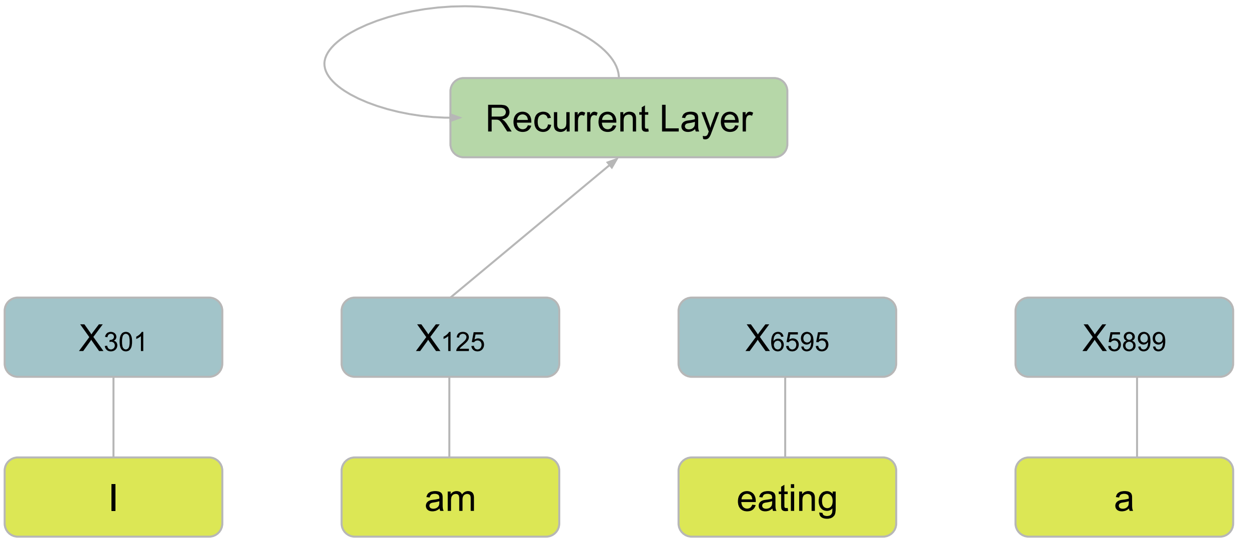


Understanding LSTM Networks by Christopher Olah.

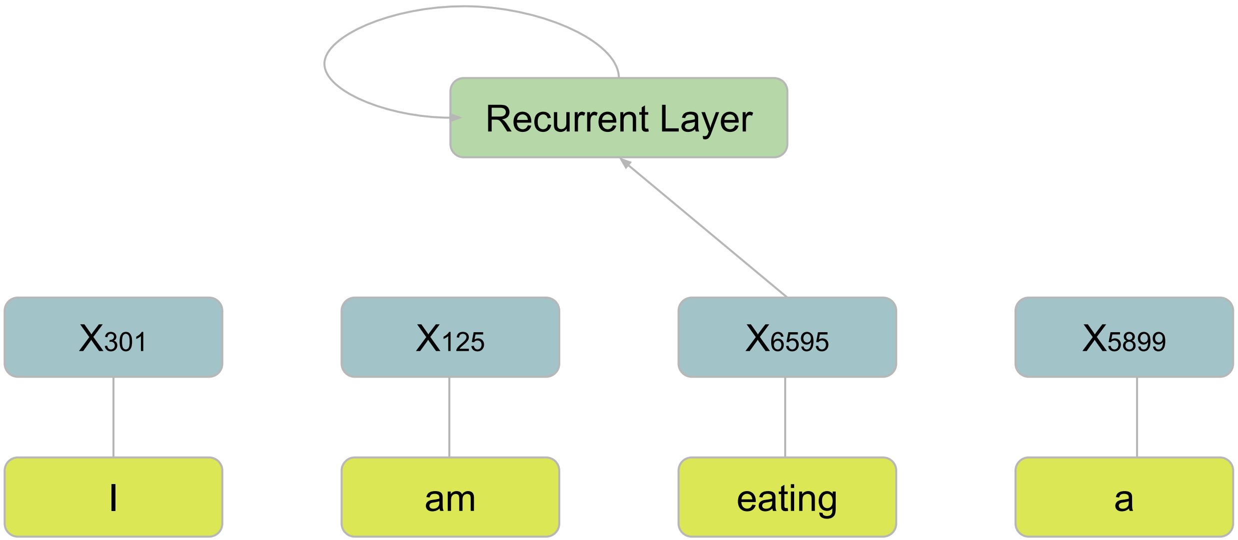
Sequence modeling



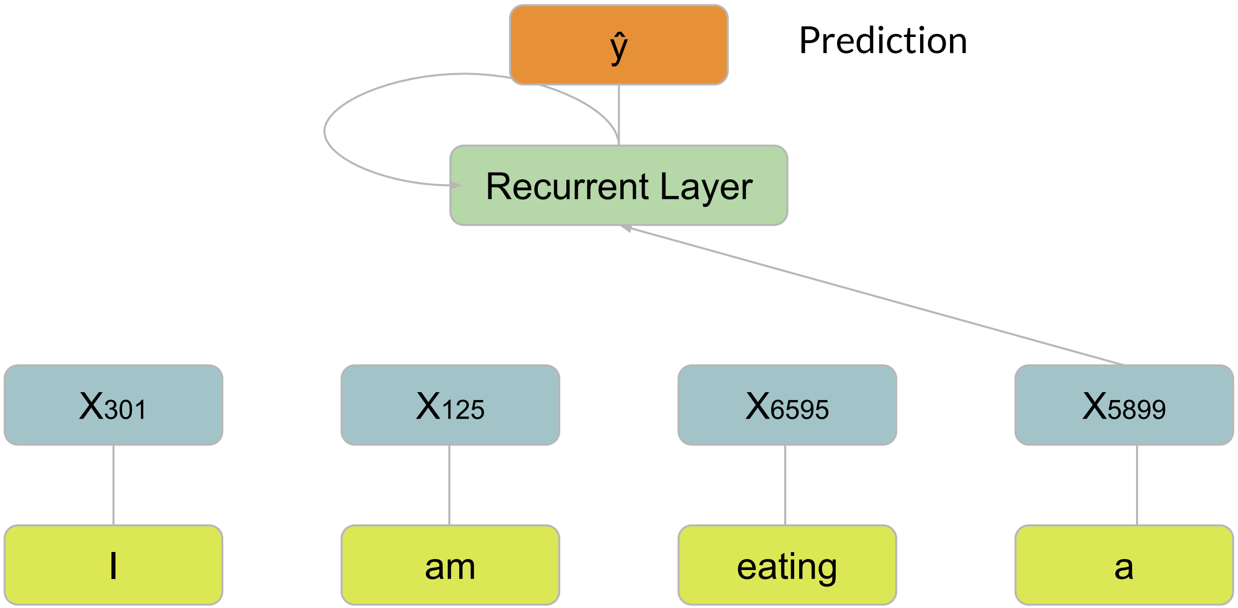
Sequence modeling



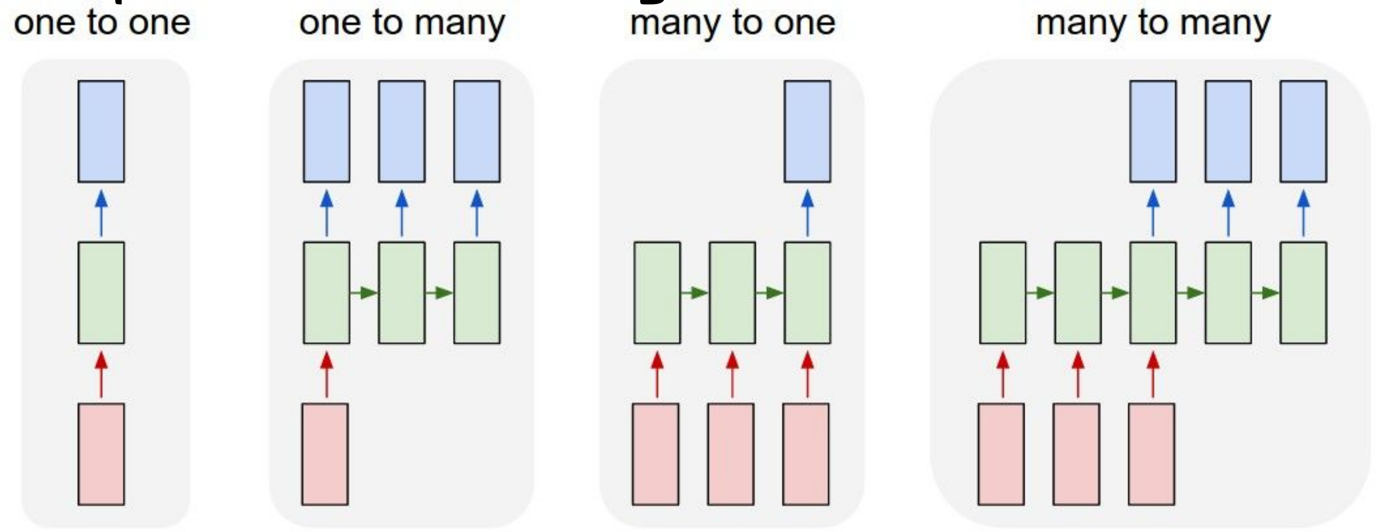
Sequence modeling



Sequence modeling

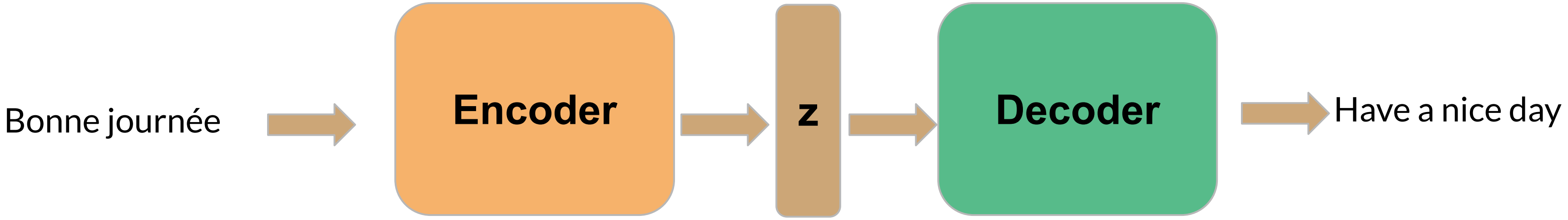


Sequence modeling

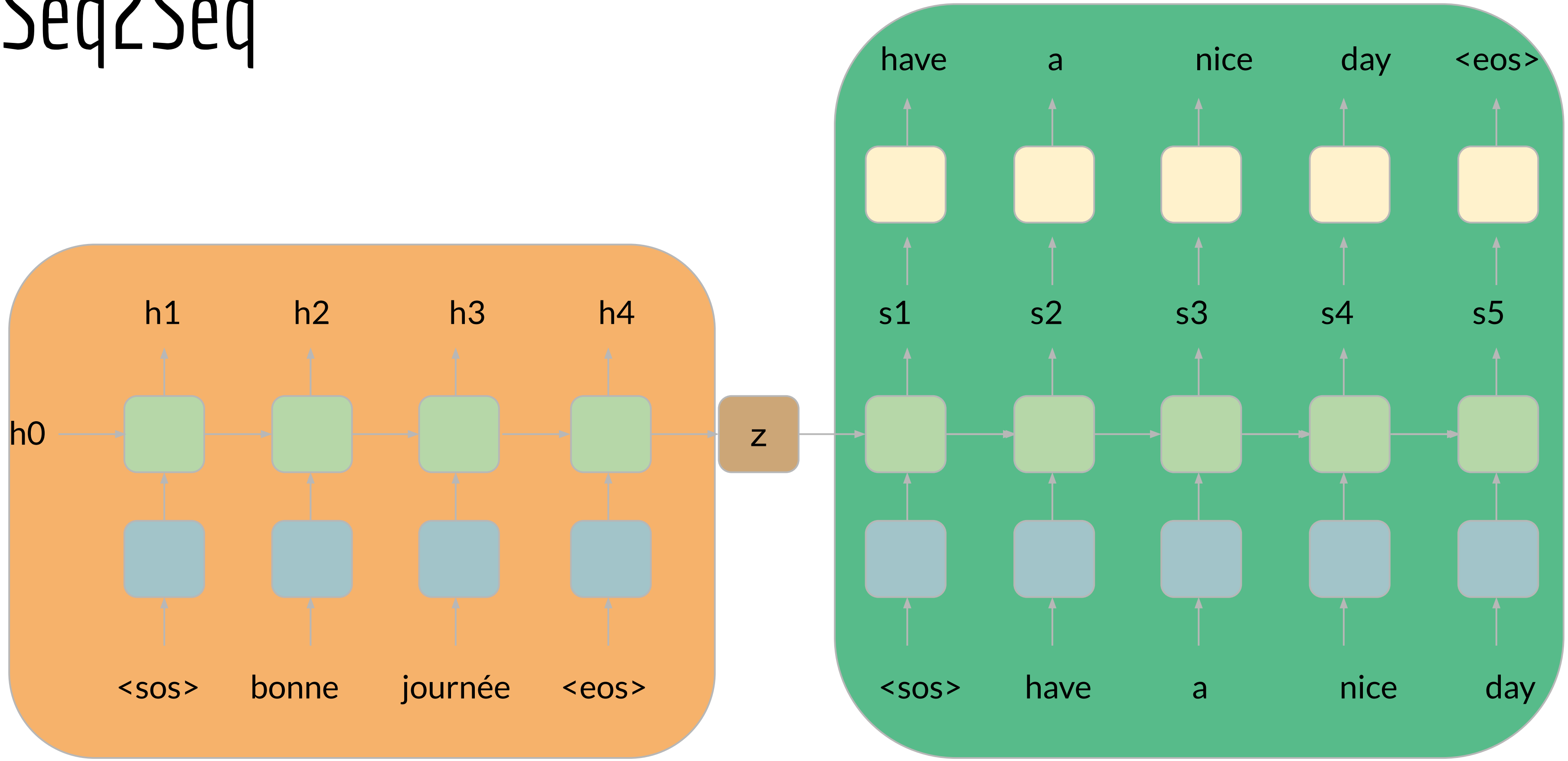


The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy

Seq2Seq



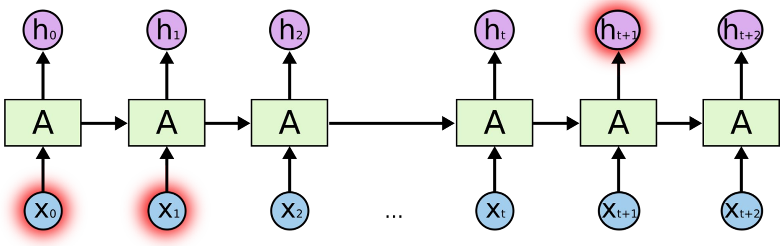
Seq2Seq



have
a
nice
day
<eos>
s1
s2
s3
s4
s5
h1
h2
h3
h4
h0
z
<sos>
bonne
journée
<eos>
<sos>
have
a
nice
day

Sequence modeling

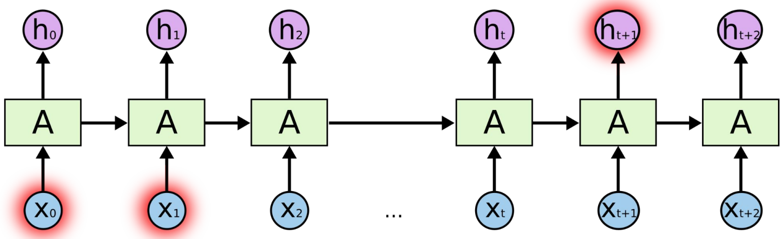
- Plain RNNs tend to perform poorly with very long sequences
- As information flows back through the network, it is lost or distorted



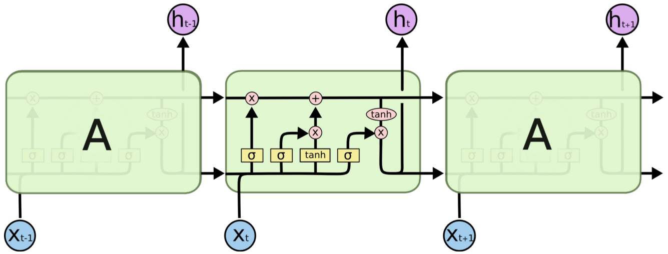
Understanding LSTM Networks by Christopher Olah.

Sequence modeling

- Plain RNNs tend to perform poorly with very long sequences
- As information flows back through the network, it is lost or distorted



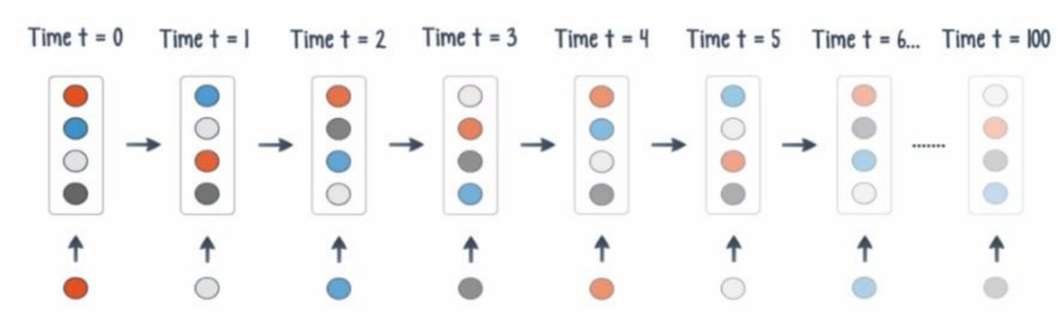
- LSTM cells introduce mechanisms that control the flow of information.



Understanding LSTM Networks by Christopher Olah.

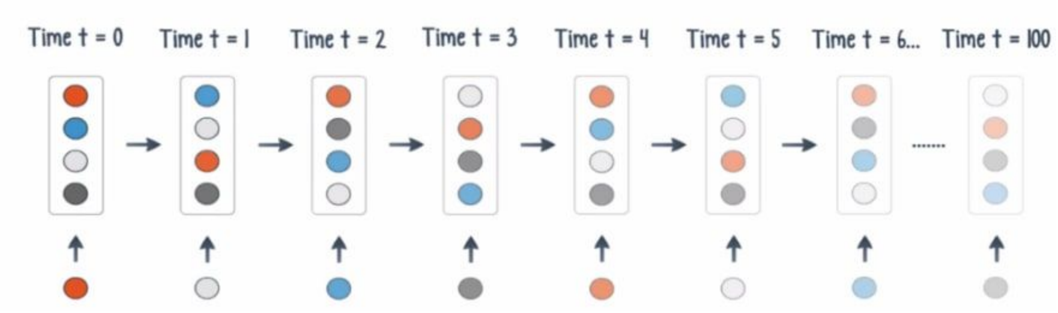
Problems with RNN

- Long Range Dependencies

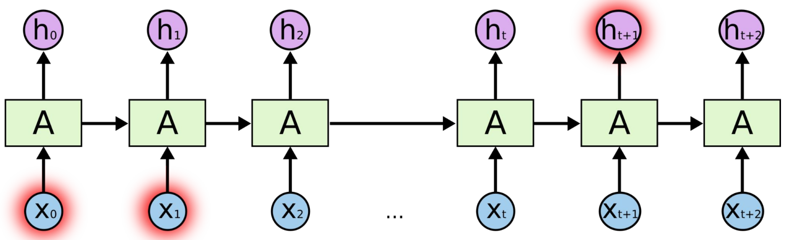


Problems with RNN

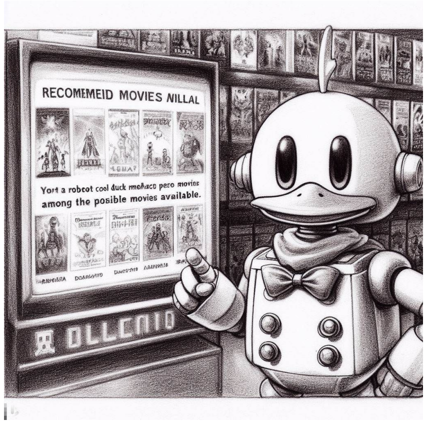
- Long Range Dependencies



- Parallelization



Introduction to Natural Language Processing



AI frameworks

Transformers

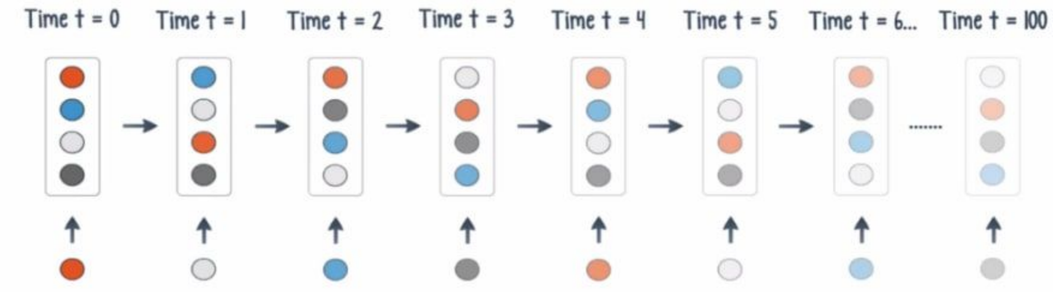
Disclaimer

Most of the illustrations and animations come from the great Jay Alammar blog post

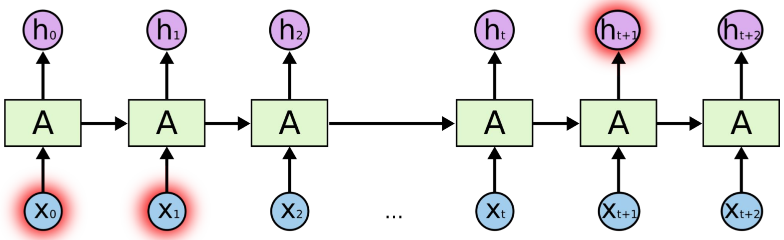
http://jalammar.github.io/illustrated-transformer/

Problems with RNN

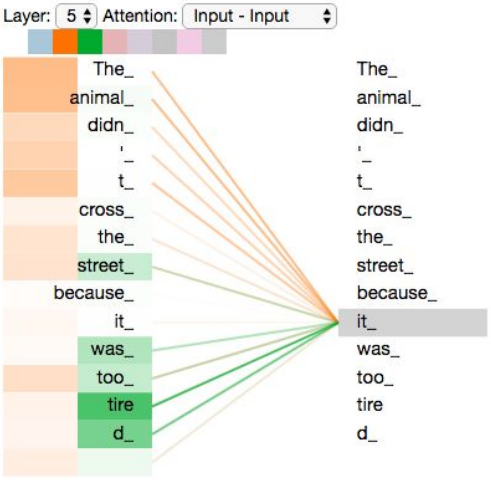
- Long Range Dependencies

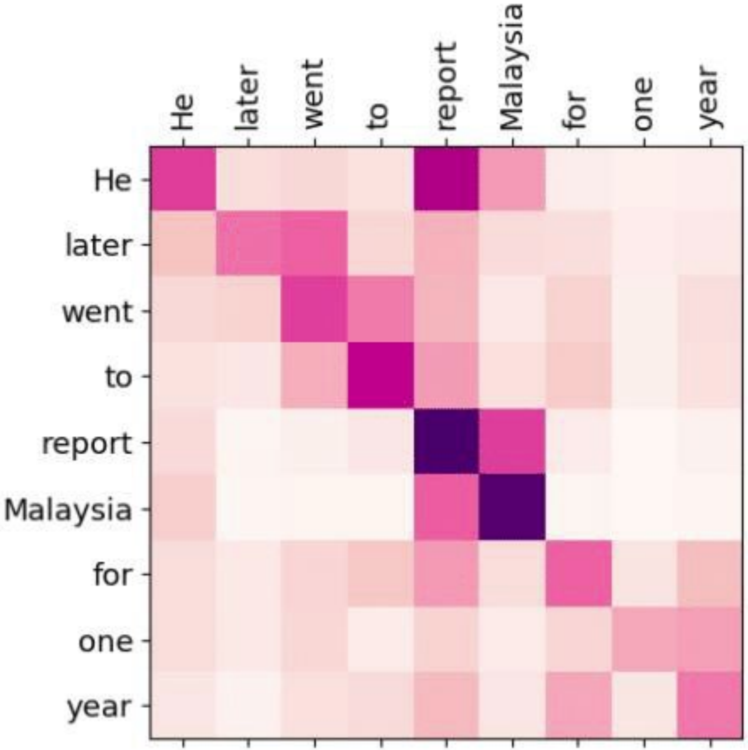


- Parallelization

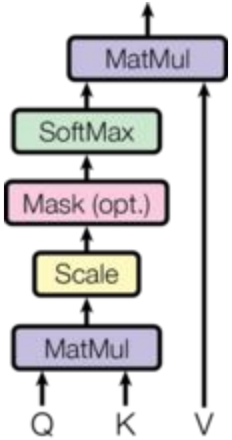


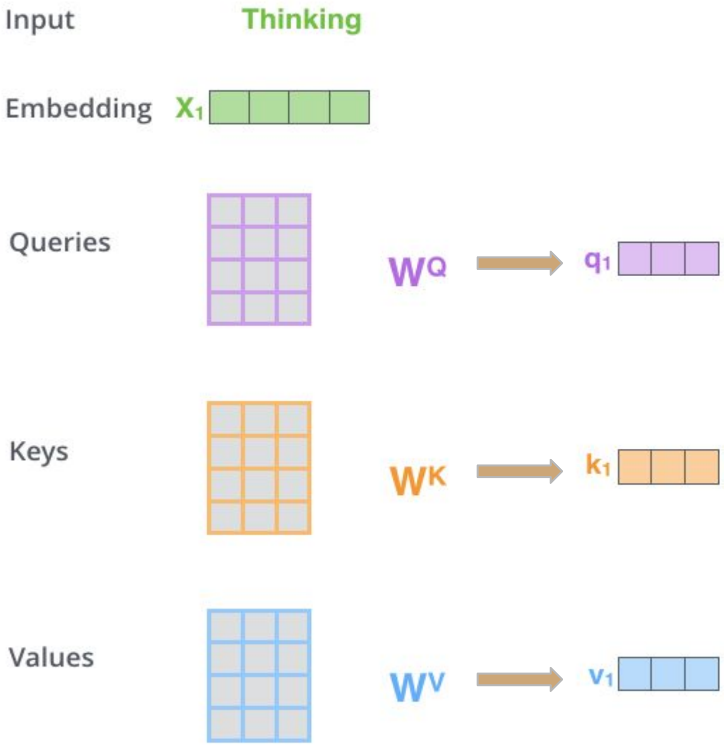
Solution: Attention and self attention



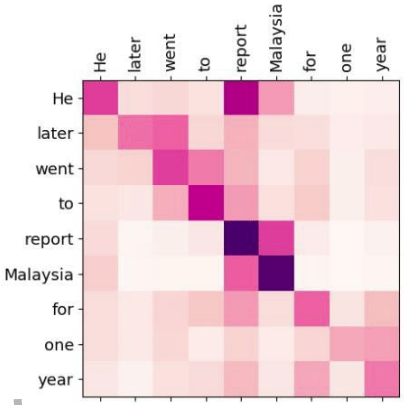


Self attention

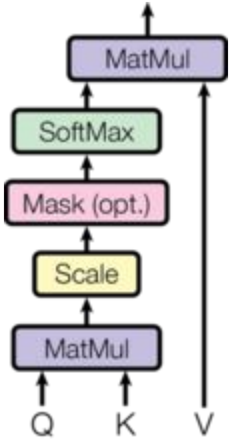


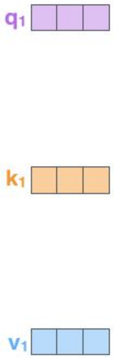


Self attention

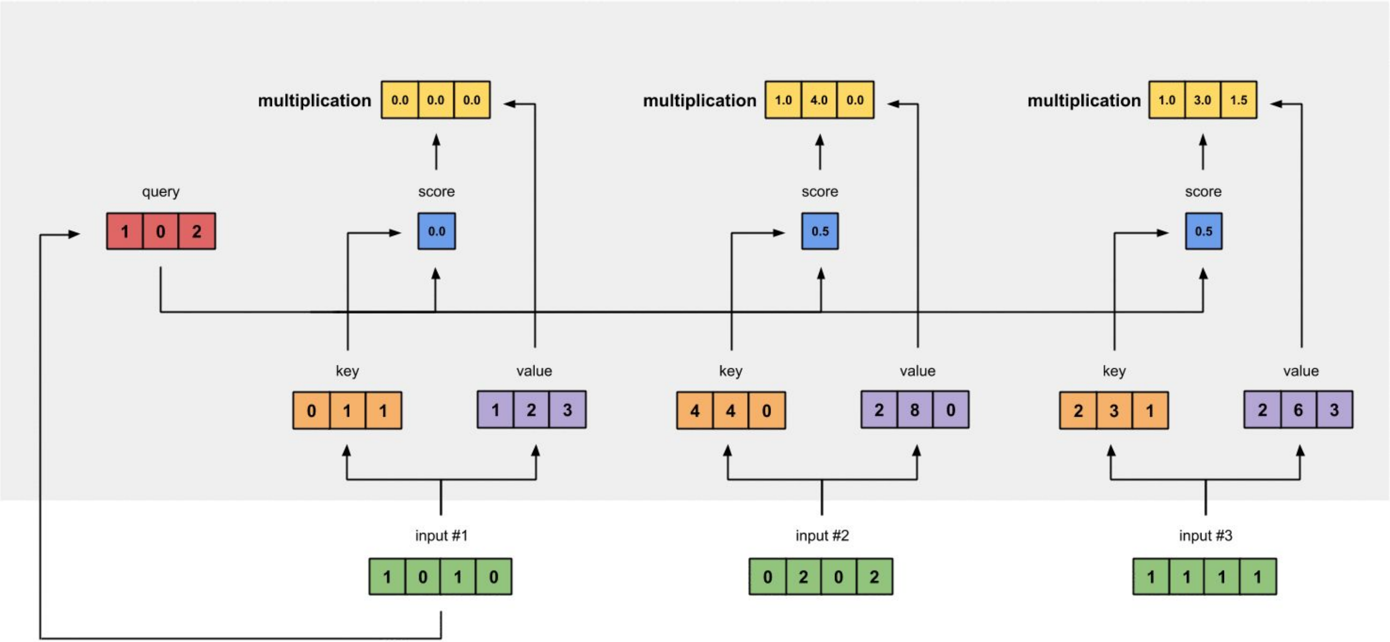


$$\mathrm{softmax}\left(\frac{Q \times K^{T}}{\sqrt{d_k}}\right) V = Z$$

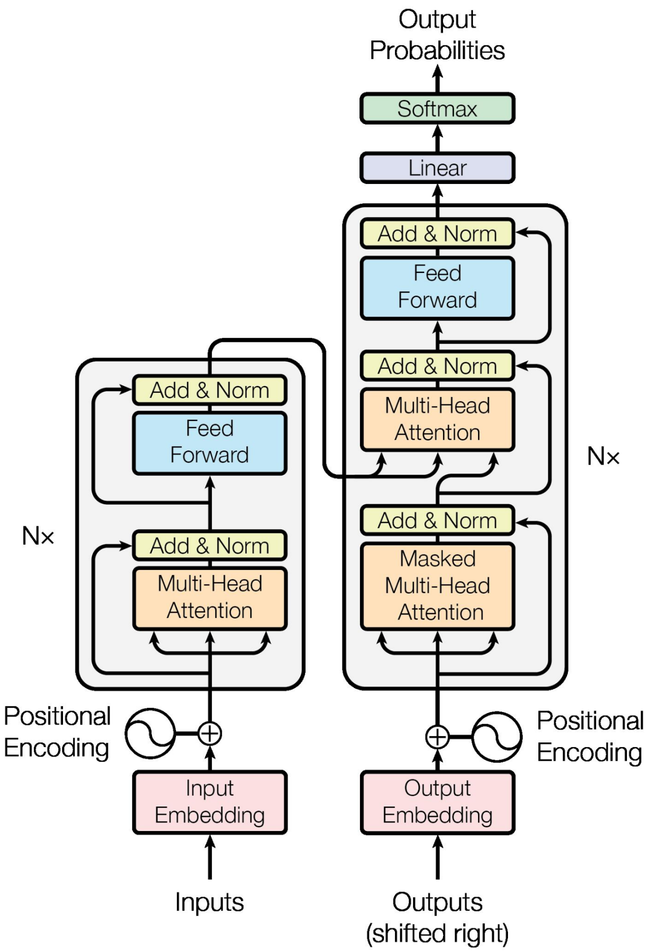




Self attention

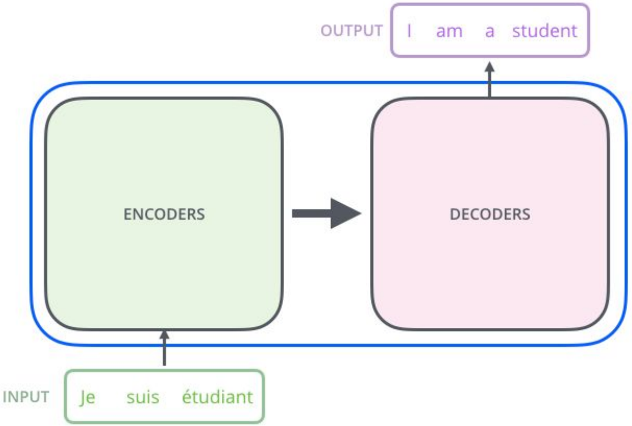


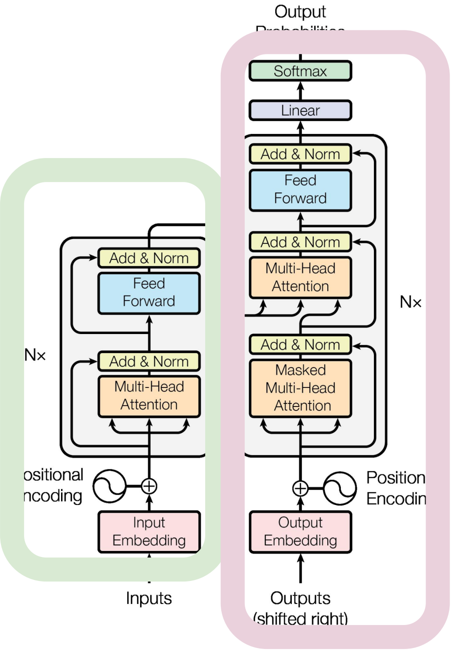
Transformers



Transformers

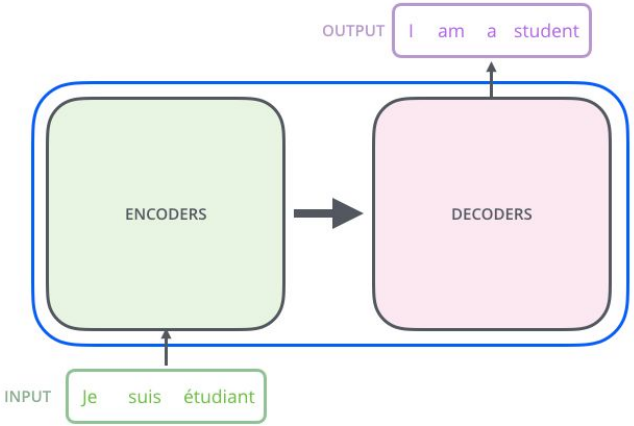
- Encoder-decoder architecture

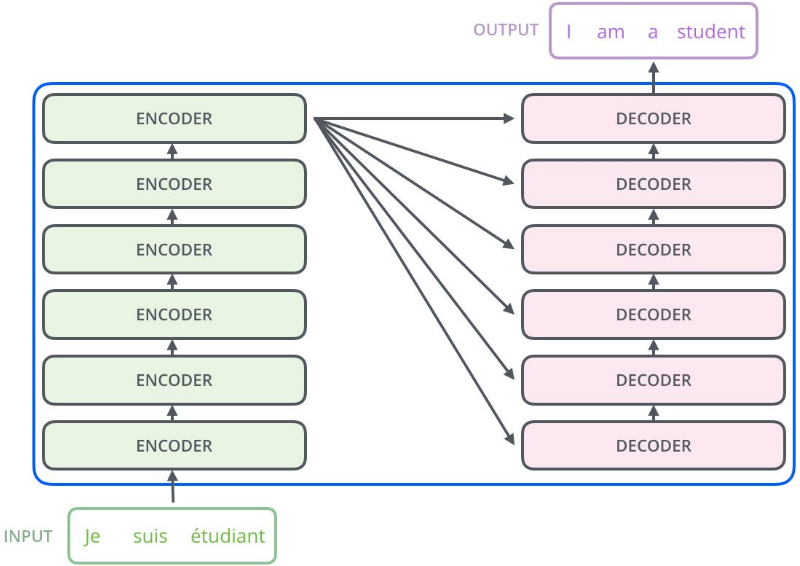




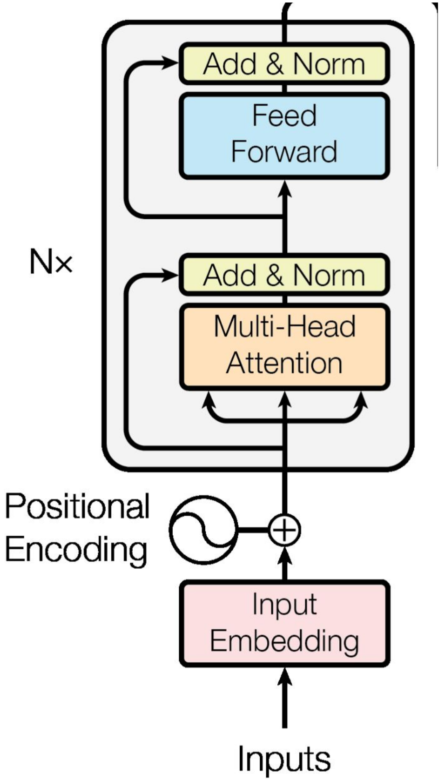
Transformers

- Encoder-decoder architecture





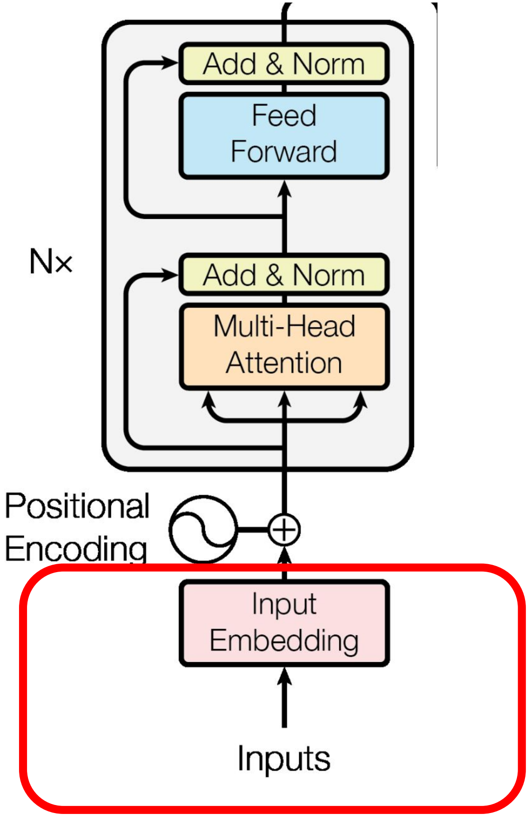
Encoder



Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

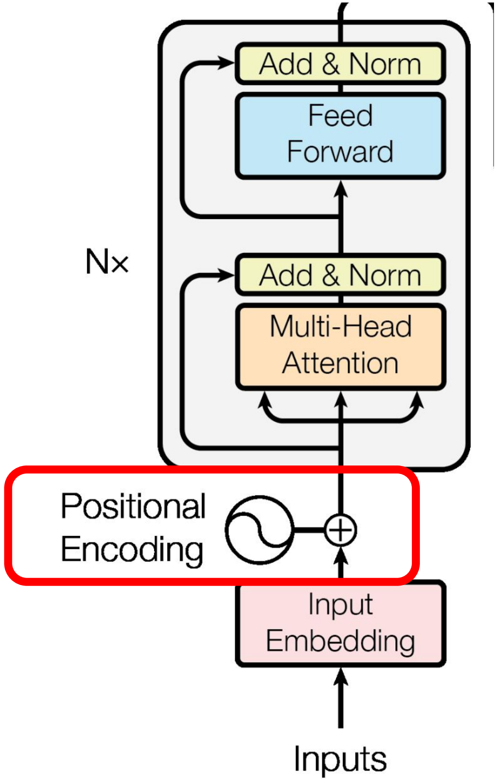
Encoder

- Input embeddings

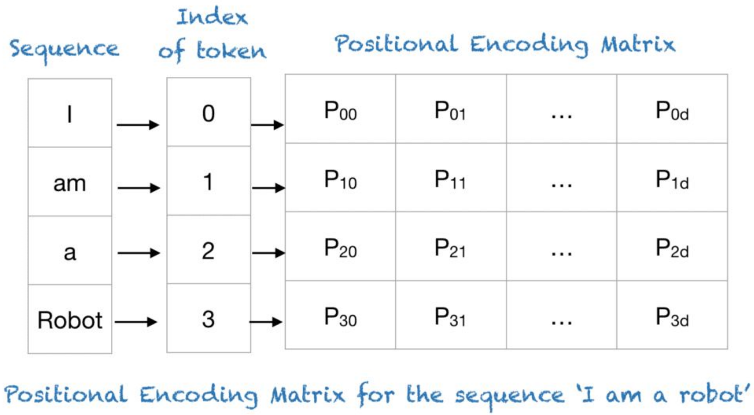


Encoder

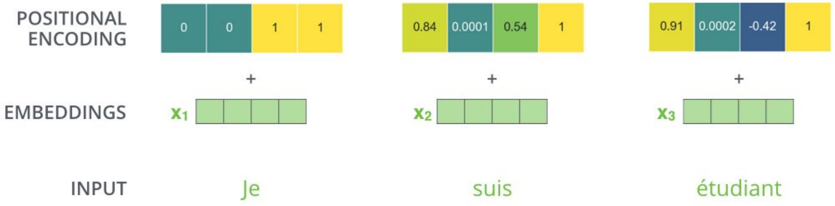
- Input embeddings
- Positional encoding



Encoder



- Input embeddings
- Positional encoding



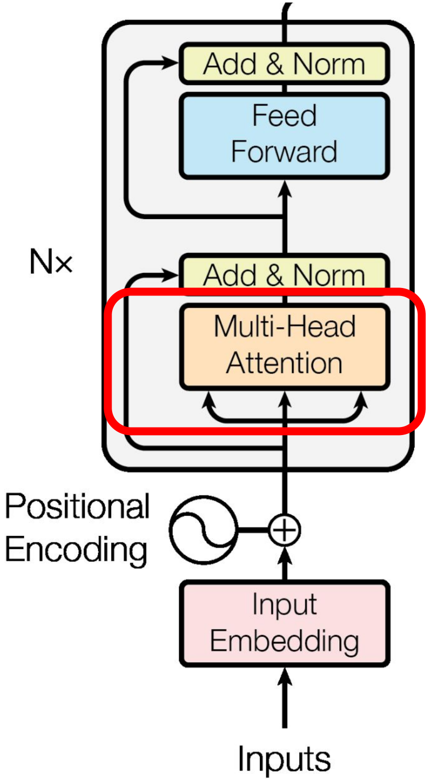
$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

Source: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

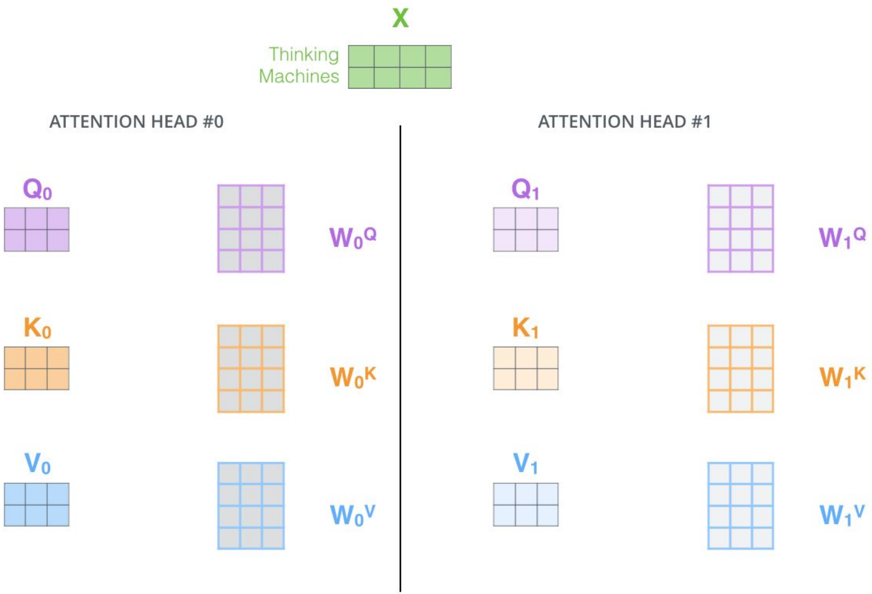
Encoder

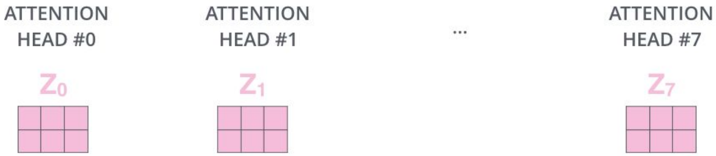
- Input embeddings
- Positional encoding
- Multi-head self attention



Encoder

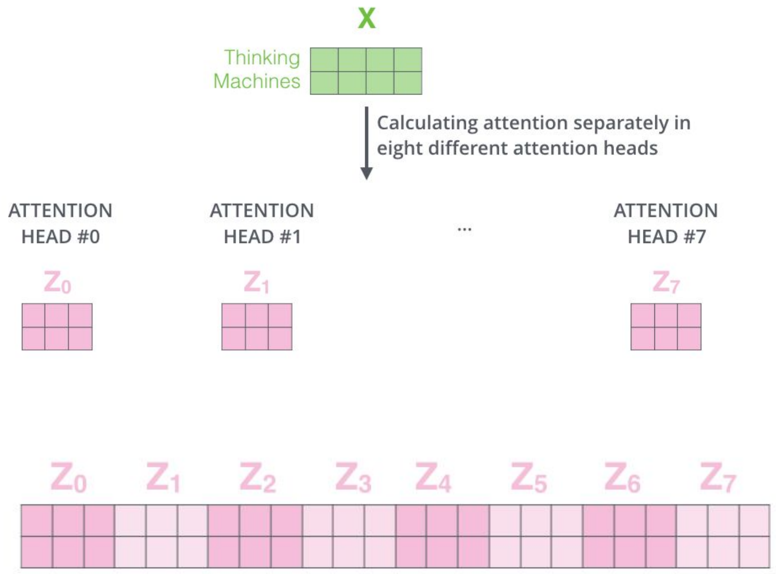
- Input embeddings
- Positional encoding
- Multi-head self attention





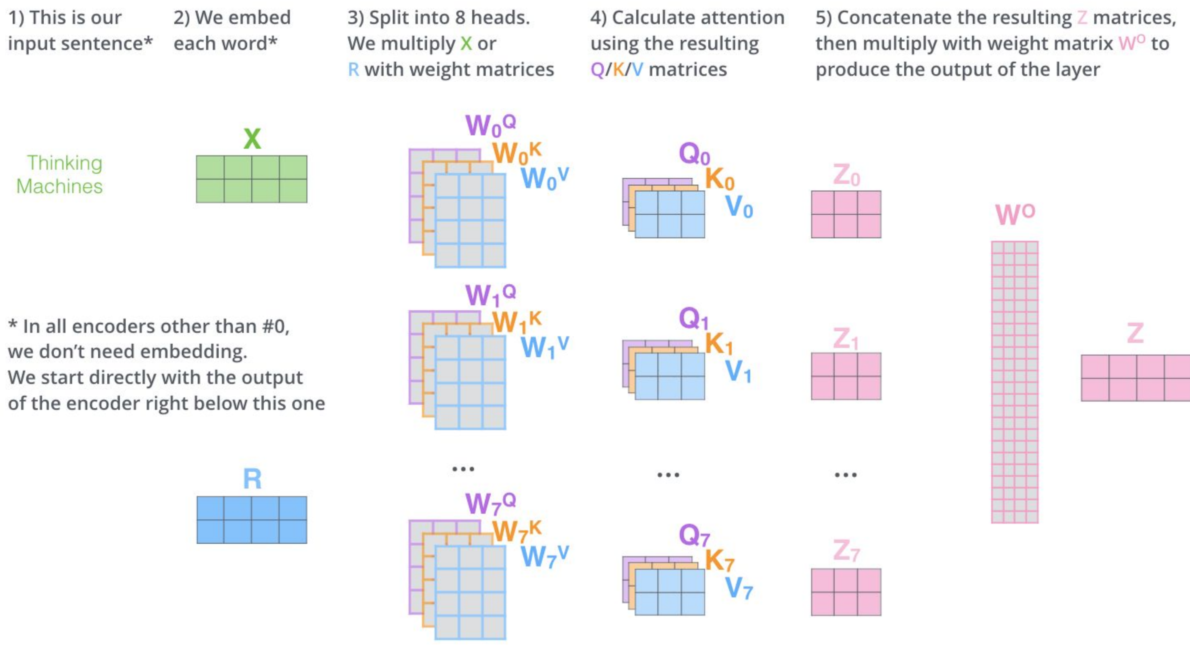
Encoder

- Input embeddings
- Positional encoding
- Multi-head self attention



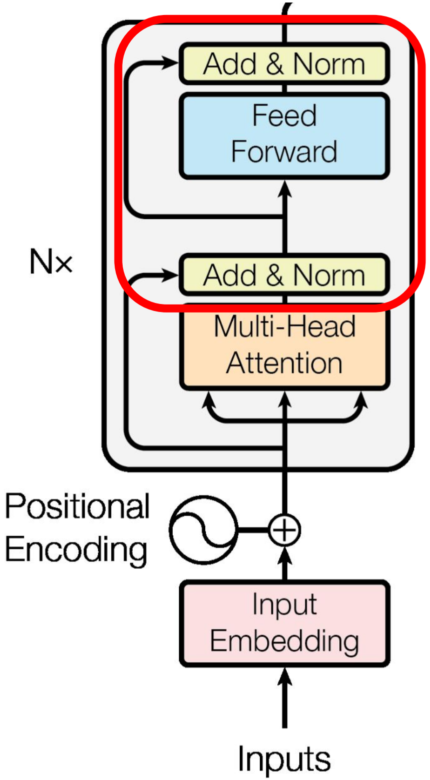
Encoder

- Input embeddings
- Positional encoding
- Multi-head self attention

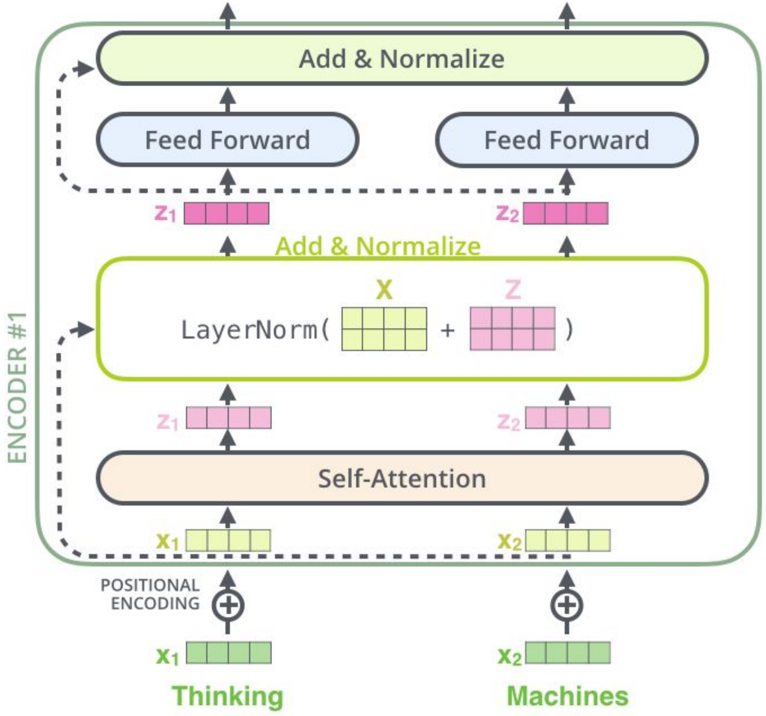


Encoder

- Input embeddings
- Positional encoding
- Multi-head self attention
- Residual connections and normalization
- Feed Forward neural net
- Residual connections and normalization



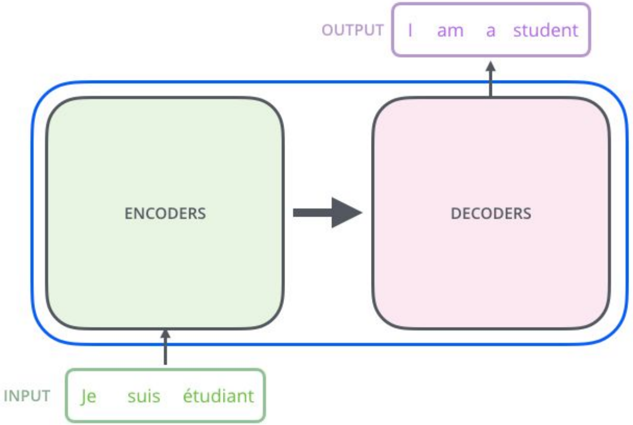
Encoder

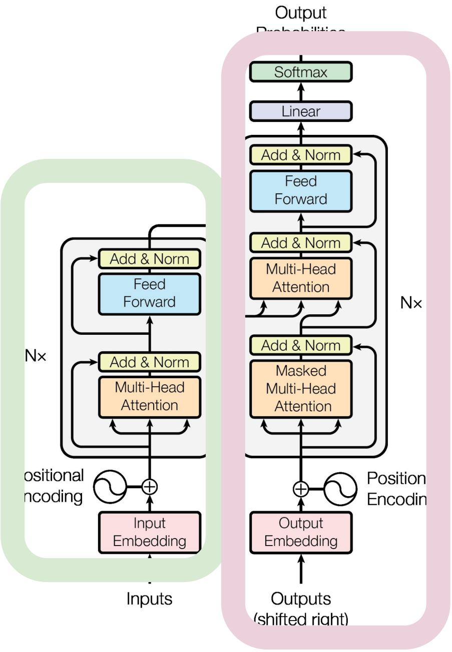


- Input embeddings
- Positional encoding
- Multi-head self attention
- Residual connections and normalization
- Feed Forward neural net
- Residual connections and normalization

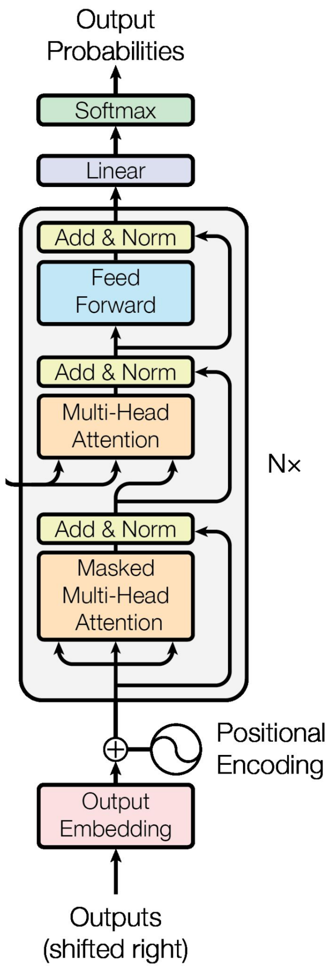
Transformers

- Encoder-decoder architecture





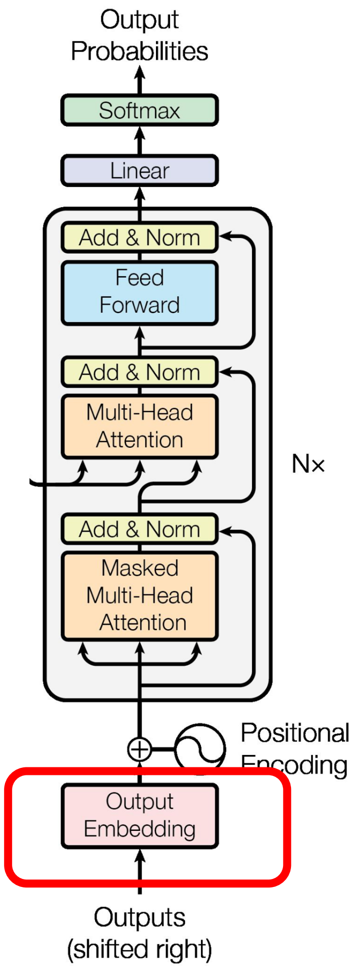
Decoder



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

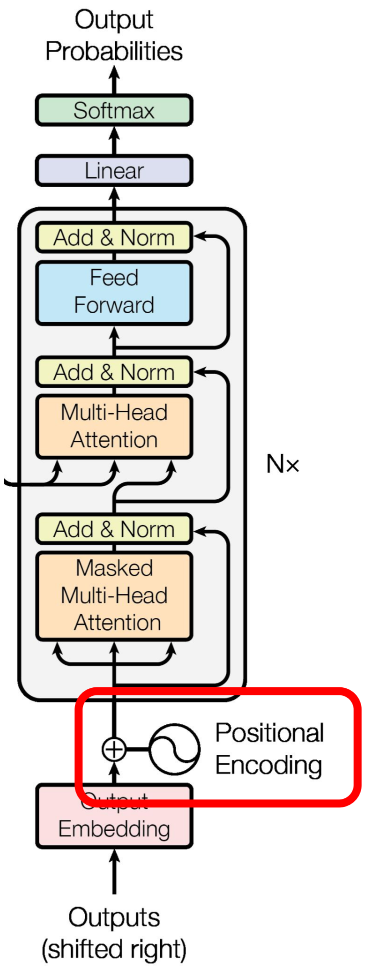
Decoder

- Output embeddings



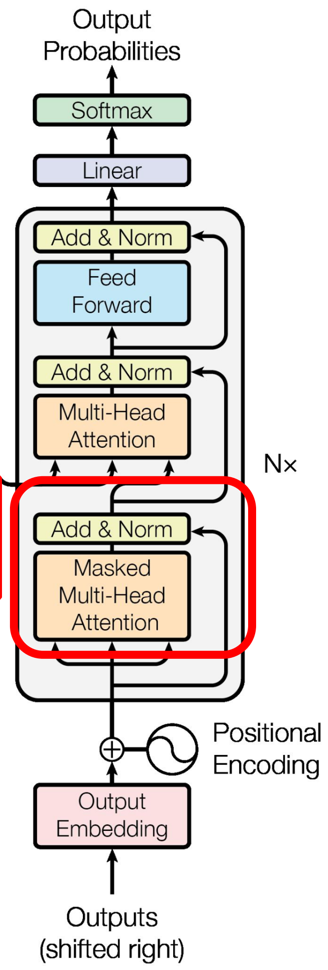
Decoder

- Output embeddings
- Positional encoding



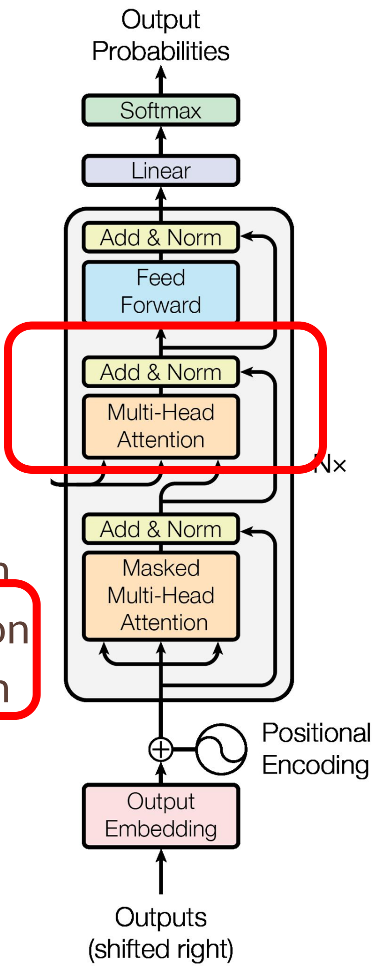
Decoder

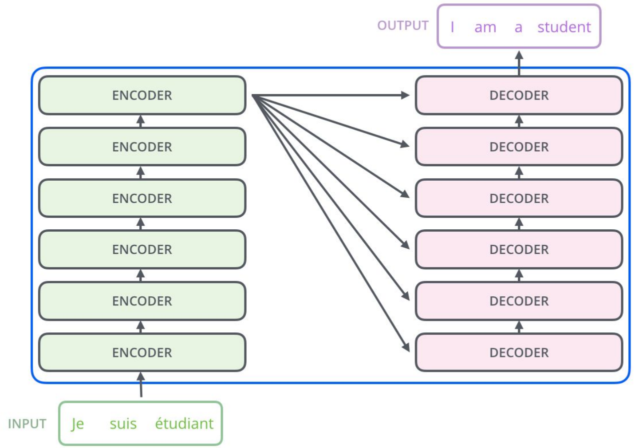
- Output embeddings
- Positional encoding
- Masked Multi-head self attention
- Residual connections & normalization



Decoder

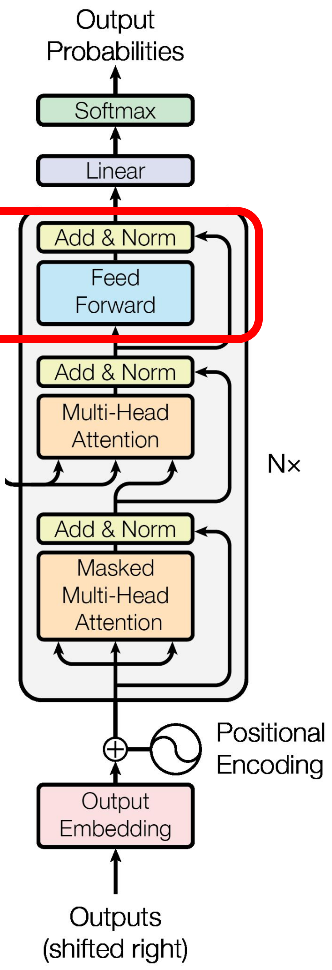
- Output embeddings
- Positional encoding
- Masked Multi-head self attention
- Residual connections & normalization
- Multi-head encoder-decoder attention
- Residual connections & normalization





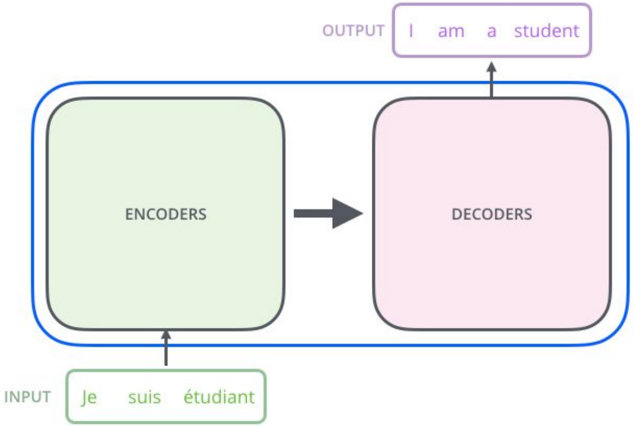
Decoder

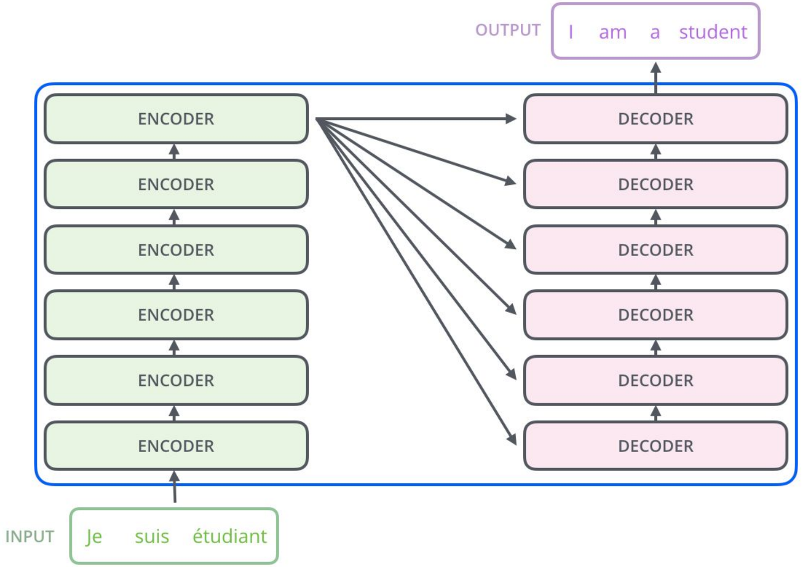
- Output embeddings
- Positional encoding
- Masked Multi-head self attention
- Residual connections & normalization
- Multi-head encoder-decoder attention
- Residual connections & normalization
- Feed forward neural network
- Residual connections & normalization



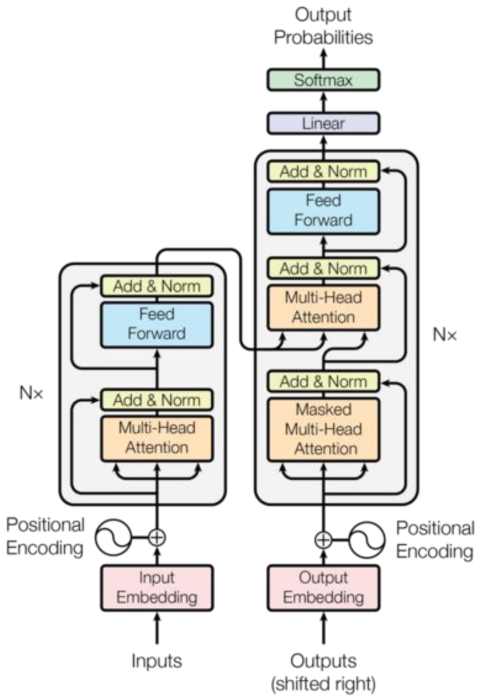
Transformers

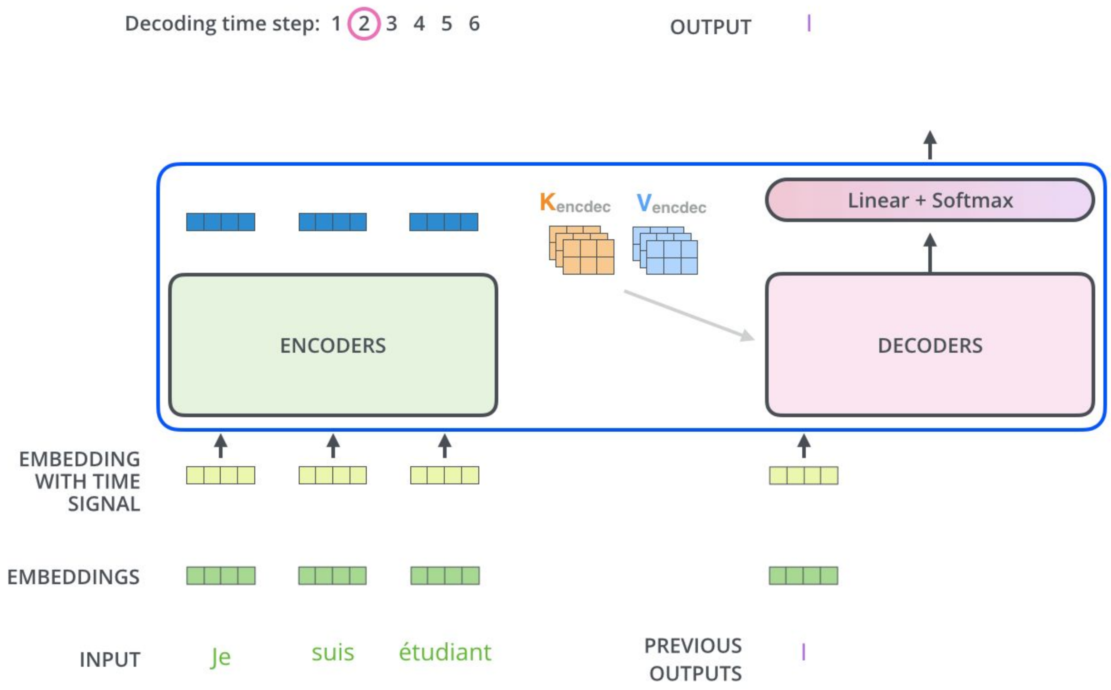
- Encoder-decoder architecture





Transformers



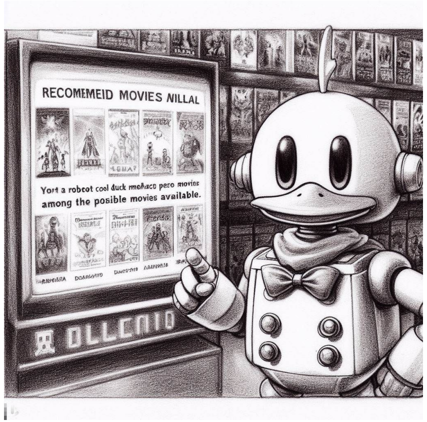


Go to this blog

Jay Alammar blog post: The illustrated transformer

http://jalammar.github.io/illustrated-transformer/

Introduction to Natural Language Processing



AI frameworks

Large Language models

Disclaimer

All of these slides are directly modified from ones created by Adil Zouitine

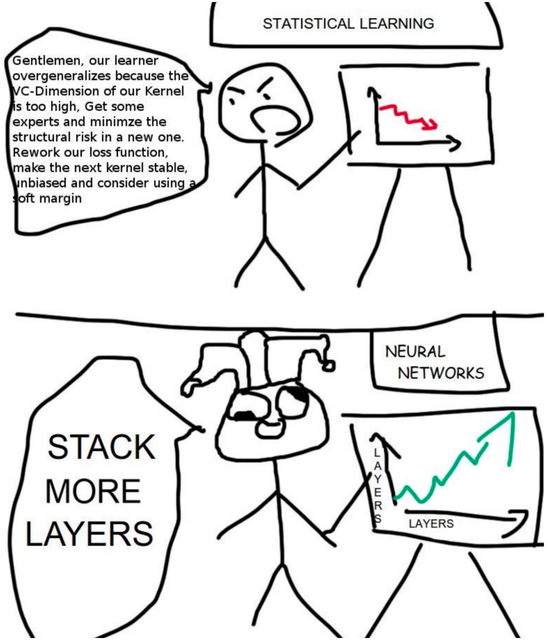
Large Language Model

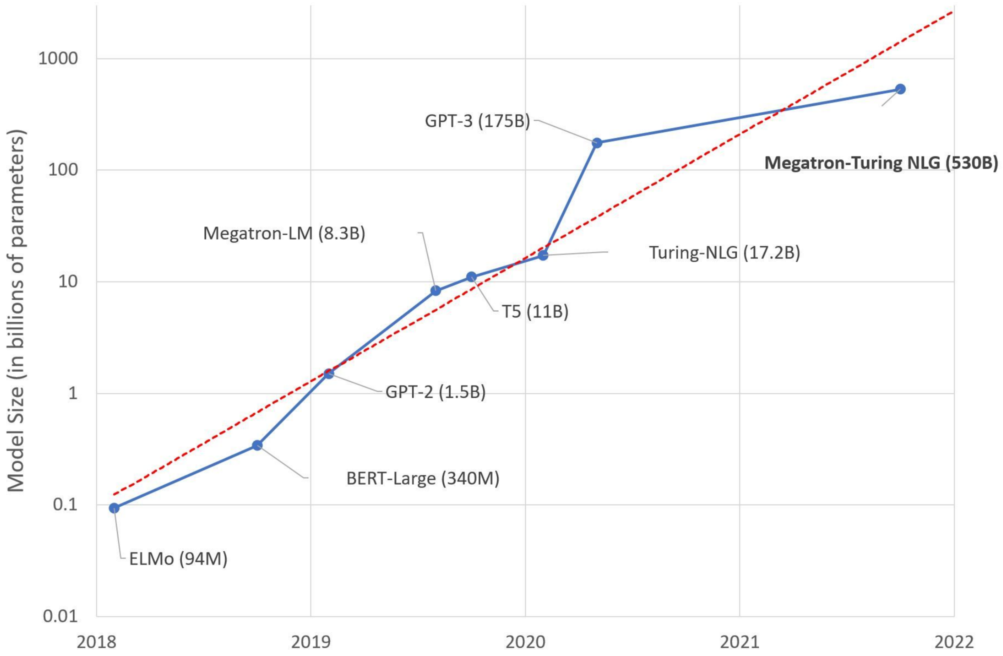
- **BERT**:
  - Masked Language Modeling (MLM)
  - Next Sentence Prediction (NSP)
  - Needs fine tuning!

- **GPT** :
  - Language modeling: predict the next word

- **T5 (Text-to-Text Transfer Transformer)**:
  - denoising auto-encoding task

Large Language Model





Generative Pretrain Transformer aka GPT

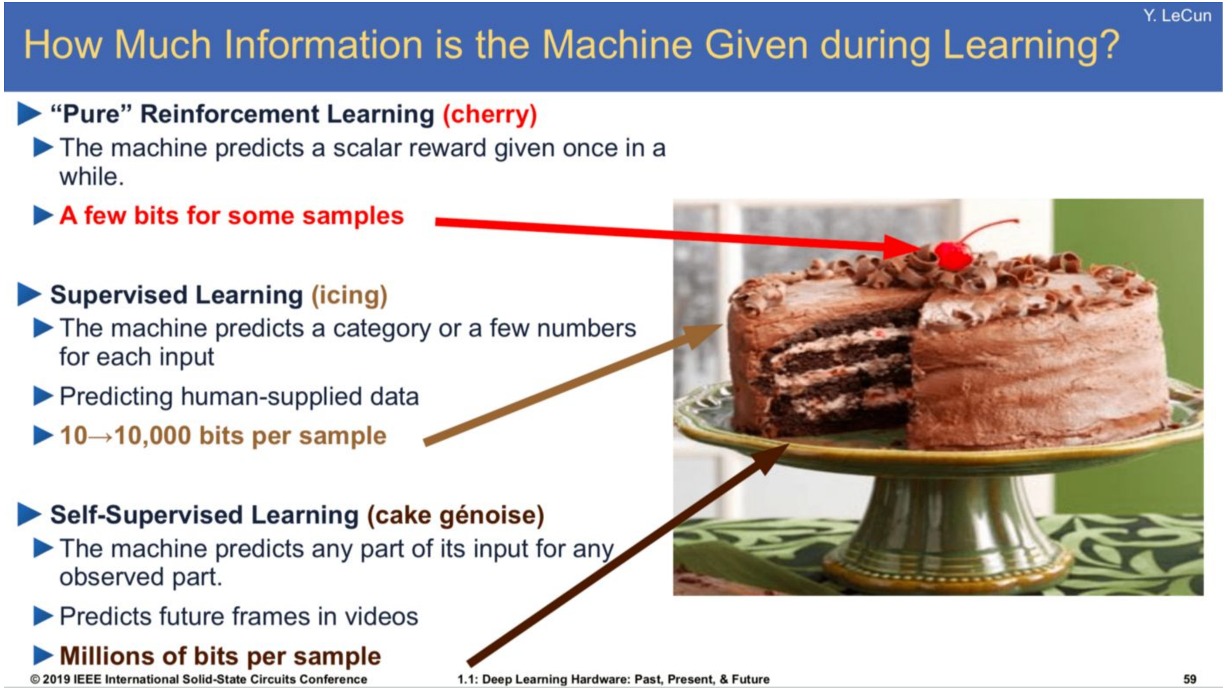
GPT (2018) : Improving Language Understanding by Generative Pre-Training
TLDR: Generative LLM + Academic dataset

GPT 2 (2019): Language Models are Unsupervised Multitask Learners
TLDR: 100 x Bigger than GPT + Generative LLM + Common crawl dataset

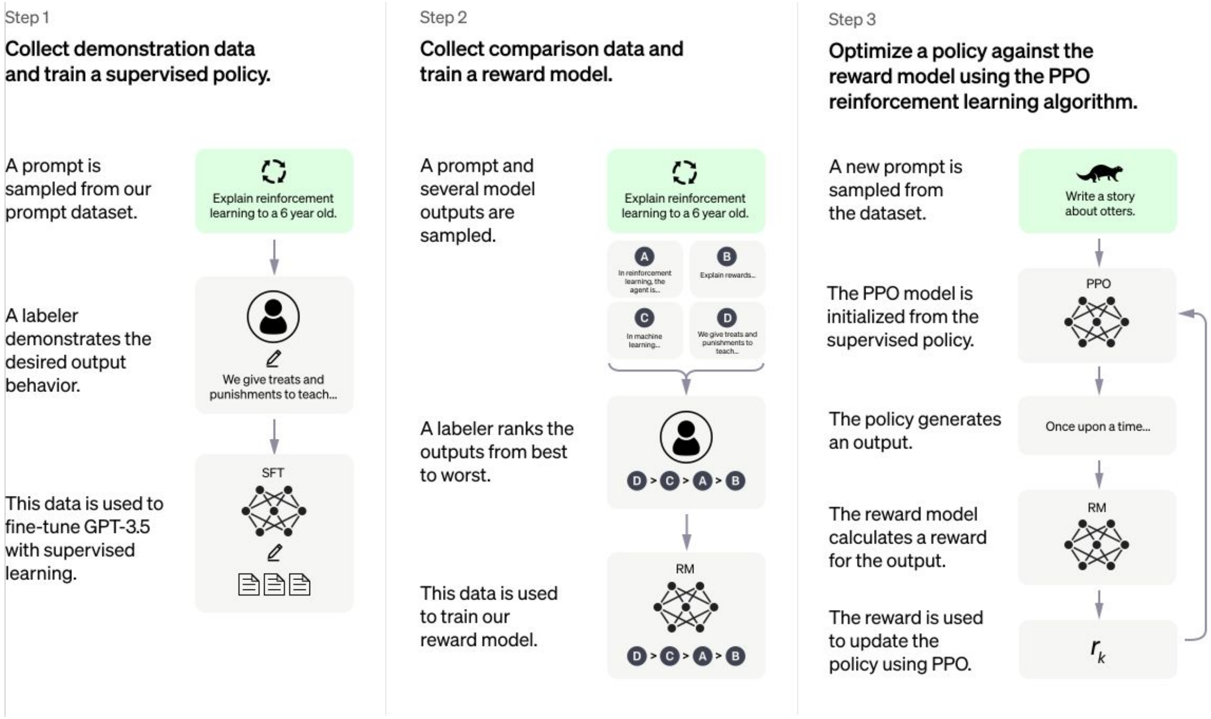
GPT 3 (2020): Language Models are Few-Shot Learners
TLDR: 100 x Bigger than GPT 2 + Generative LLM + Common crawl dataset

GPT 4 (2023) :
TLDR: 400 x Bigger than GPT 3 + Generative LLM + Common crawl (img + txt)

Chat GPT is a cake



Reinforcement learning from human feedback



Future of LLM

- Small open source foundation models: LLaMA2 , Mistral, …
- Fit on single computers
- Fine-tune for specific usages
- Comparable performance than ChatGPT 3.5