Natural Language Processing

Presented to you by Mohamed Arbi Nsibi

14/02/2026

Agenda & Timeline

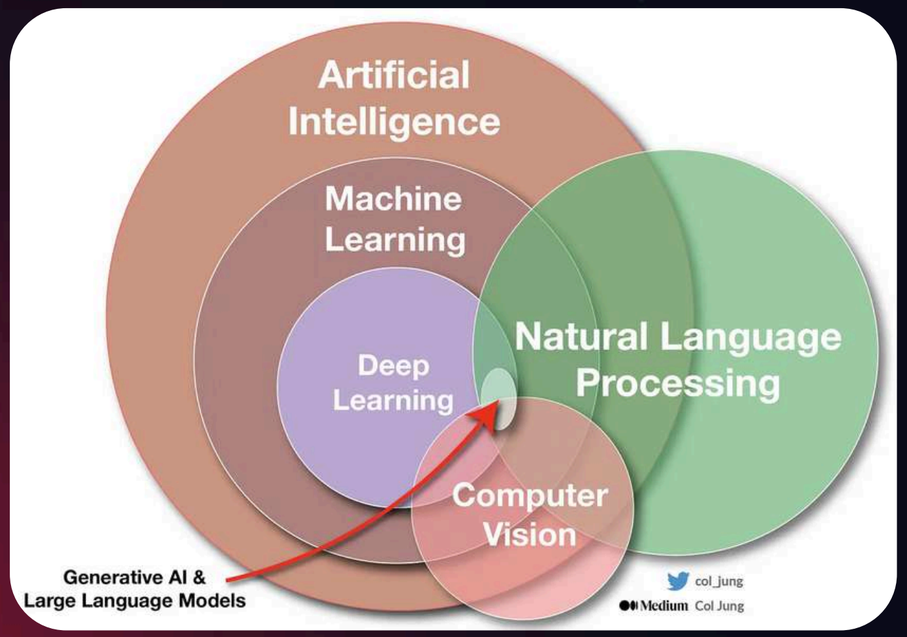
- What is NLP & history
- NLP Pipeline
- RNN & LSTM
- Attention & transformer
- Demo
- QUIZ

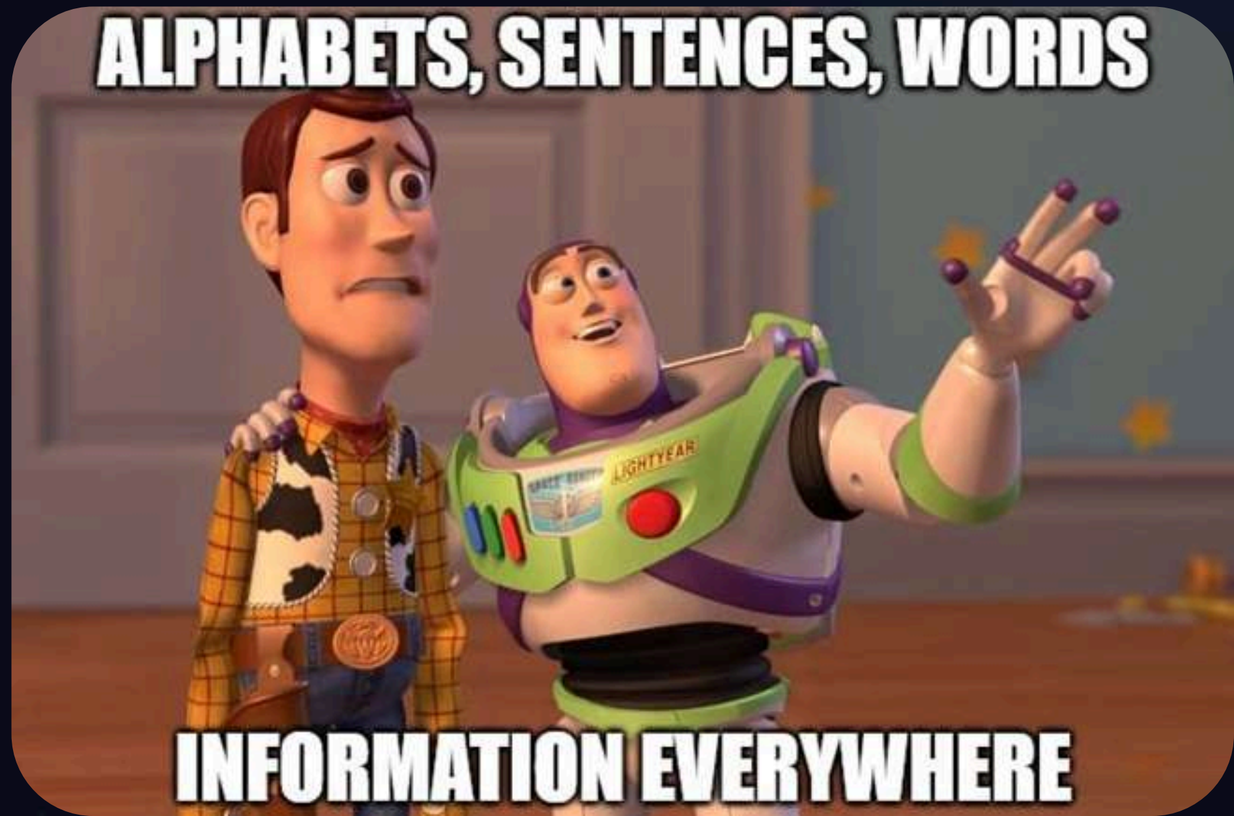
Mohamed Arbi Nsibi

- ML engineer
- Qdrant Star ⭐
- Former GDSC Lead 23/24

WHAT IS NLP ?

is referred to as NLP. It is a subset of AI that enables machines to comprehend and analyze human languages. Text or audio can be used to represent human languages.





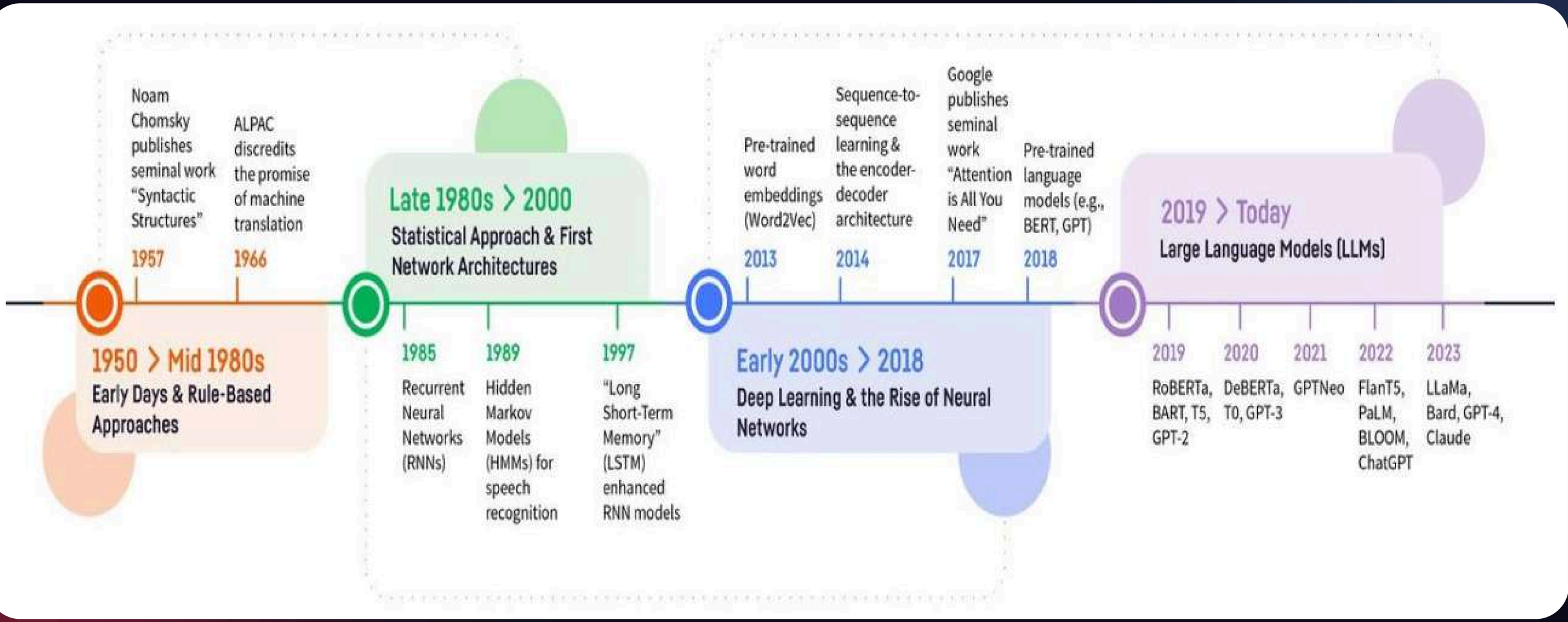
Why natural language processing is important?

- **Facilitates Human-Computer Interaction:** users to communicate with devices, applications, and systems using spoken or written language

- **Automates Routine Tasks**: asks such as text summarization, sentiment analysis, document classification, and information extraction, improving efficiency and productivity.

- **Empowers Chatbots and Virtual Assistants:** understand user queries, provide information, perform tasks, and offer personalized recommendations, enhancing customer service and user experience.

NLP History


1950 > Mid 1980s
Early Days & Rule-Based Approaches
1957
Noam Chomsky publishes seminal work "Syntactic Structures"
1966
ALPAC discredits the promise of machine translation
Late 1980s > 2000
Statistical Approach & First Network Architectures
1985
Recurrent Neural Networks (RNNs)
1989
Hidden Markov Models (HMMs) for speech recognition
1997
"Long Short-Term Memory" (LSTM) enhanced RNN models
Early 2000s > 2018
Deep Learning & the Rise of Neural Networks
2013
Pre-trained word embeddings (Word2Vec)
2014
Sequence-to-sequence learning & the encoder-decoder architecture
2017
Google publishes seminal work "Attention is All You Need"
2018
Pre-trained language models (e.g., BERT, GPT)
2019 > Today
Large Language Models (LLMs)
2019
RoBERTa, BART, T5, GPT-2
2020
DeBERTa, T0, GPT-3
2021
GPTNeo
2022
FlanT5, PaLM, BLOOM, ChatGPT
2023
LLaMa, Bard, GPT-4, Claude

Components of NLP

Natural Language Understanding (NLU)

- NLU focuses on interpreting and extracting meaning from human language input.
- It involves techniques : text parsing, entity recognition, sentiment analysis, and intent detection.
- NLU systems aim to comprehend the content of text or speech input to extract relevant information and understand the user's intentions or queries
- applications : chatbots that understand user queries, sentiment analysis tools that analyze emotions in text, and voice assistants.

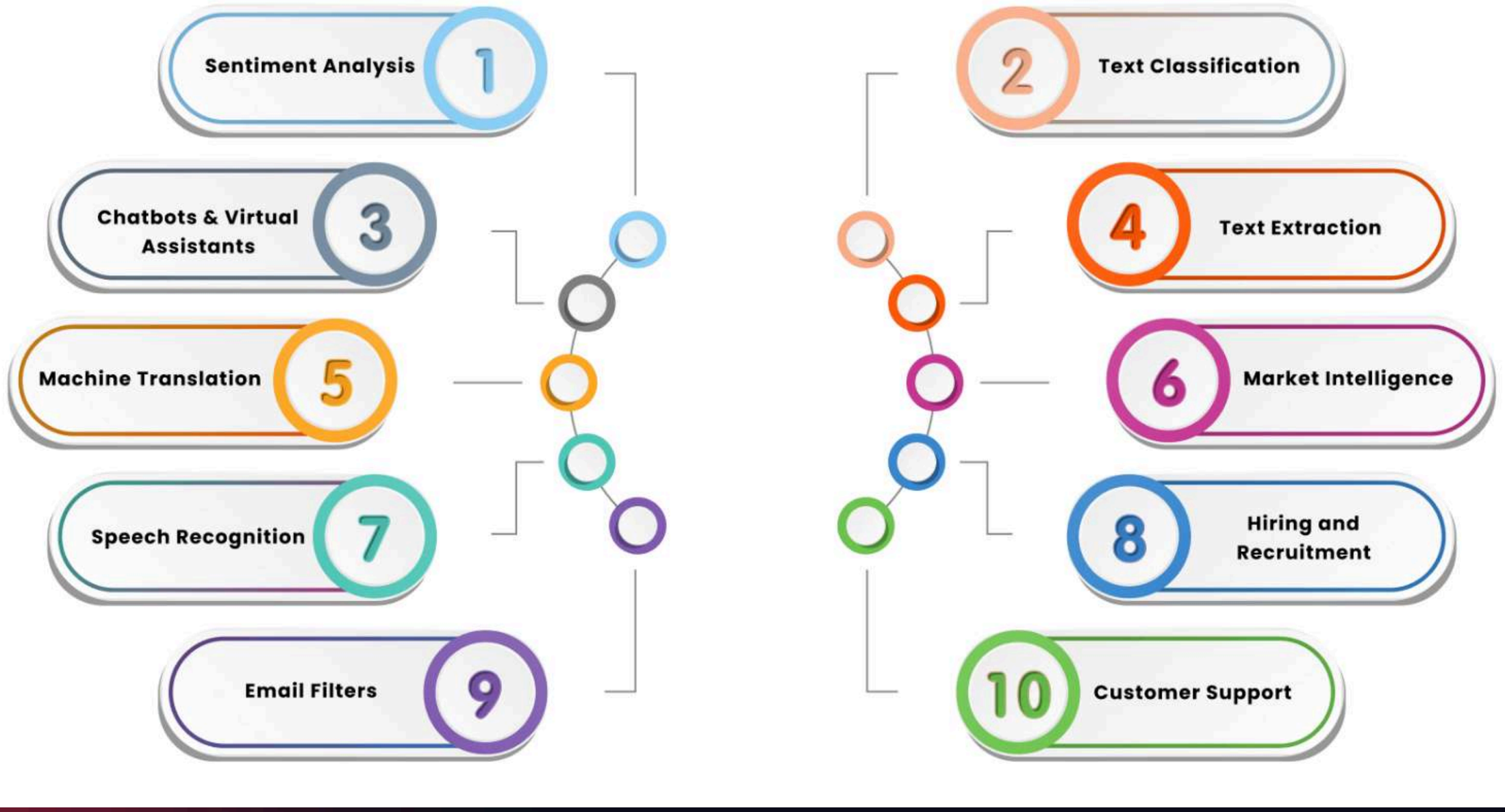
By Mohammed Arbi Nsibi

Components of NLP

Natural Language Generation (NLG)

- NLG, on the other hand, deals with creating human-like text or speech output based on structured data or input from NLU systems
- NLG systems generate coherent and contextually relevant text or speech by combining linguistic rules, templates, and sometimes machine learning models
- NLG applications include text summarization, language translation, chatbot responses, and content generation for news articles or reports

NLP applications



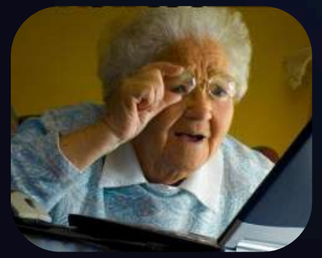
NLP Pipeline

Data Acquisition → Text Cleaning → Text Preprocessing → Feature Engineering → Model Building → Evaluation → Deployment

Text processing

Text Preprocessing Pipeline

- Chunking
- Word Tokenization
- Stemming
- Lemmatization
- Identifying Stop Words
- POS tags



- Lowercasing
- Stop word removal
  - Removing digit/punctuation
  - Named Entity Recognition (NER)

By Mohammed Arbi Nsibi

Text Cleaning

Regex or Regular Expression

Spelling corrections

- used for searching the string of specific patterns. Suppose our data contain phone number, email-Id, and URL.
- we can find such text using the regular expression. After that either we can keep or remove such text patterns as per requirements.

Text Cleaning

Stopword Removal

Input Text: "The quick brown fox jumps over the lazy dog."

Output Text: "quick brown fox jumps lazy dog."

Stemming vs Lemmatization

Stemming

Input Text: "running"

Output Text: "run"

Lemmatization

Input Text: "better"

Output Text: "good"

"Elon Musk founded SpaceX in 2002."

Named Entity Recognition (NER)

[Elon Musk: PERSON],
[SpaceX: ORG],
[2002: DATE].

Entity Recognition and Masking

Input Text: "John Smith works at Google."

Output Text: "[PERSON] works at [ORGANIZATION]."

POS tagging

Elon/NNP (Proper noun)
Musk/NNP
founded/VBD
SpaceX/NNP
in/IN
2002/CD (Cardinal number)

Feature Engineering = Text Representation = Text Vectorization.

Our main agenda is to represent the text in the numeric vector in such a way that the ML algorithm can understand the text attribute.

Traditional Approach

- One Hot Encoding

Traditional Approach

- One Hot Encoding

id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Traditional Approach

- TF-IDF (Term Frequency – Inverse Document Frequency) 1972

  Give more weight to rare words and less to common terms

$$TF(t,d) = \frac{(Number\ of\ occurrences\ of\ term\ t\ in\ document\ d)}{(Total\ number\ of\ terms\ in\ the\ document\ d)}$$

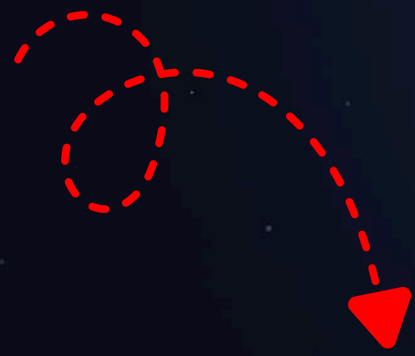
$$IDF(t,D) = \log_e \frac{(Total\ number\ of\ documents\ in\ the\ corpus)}{(Number\ of\ documents\ with\ term\ t\ in\ them)}$$

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

- Neural Approach (Word embedding)

car =[0.8, 0.9, 0.9, 0.01, 0.75]
bike =[0.8, 0.7, 0.2, 0.01, 0.5]

not interpretable for humans



try to incorporate the contextual meaning of the words.

Word	car	bike
Road	0.8	0.8
Speed	0.9	0.7
Fuel	0.9	0.2
Animal	0.01	0.01
Price	0.75	0.5

How can we get these word embedding vectors?

How can we get these word embedding vectors?

Train our own embedding layer

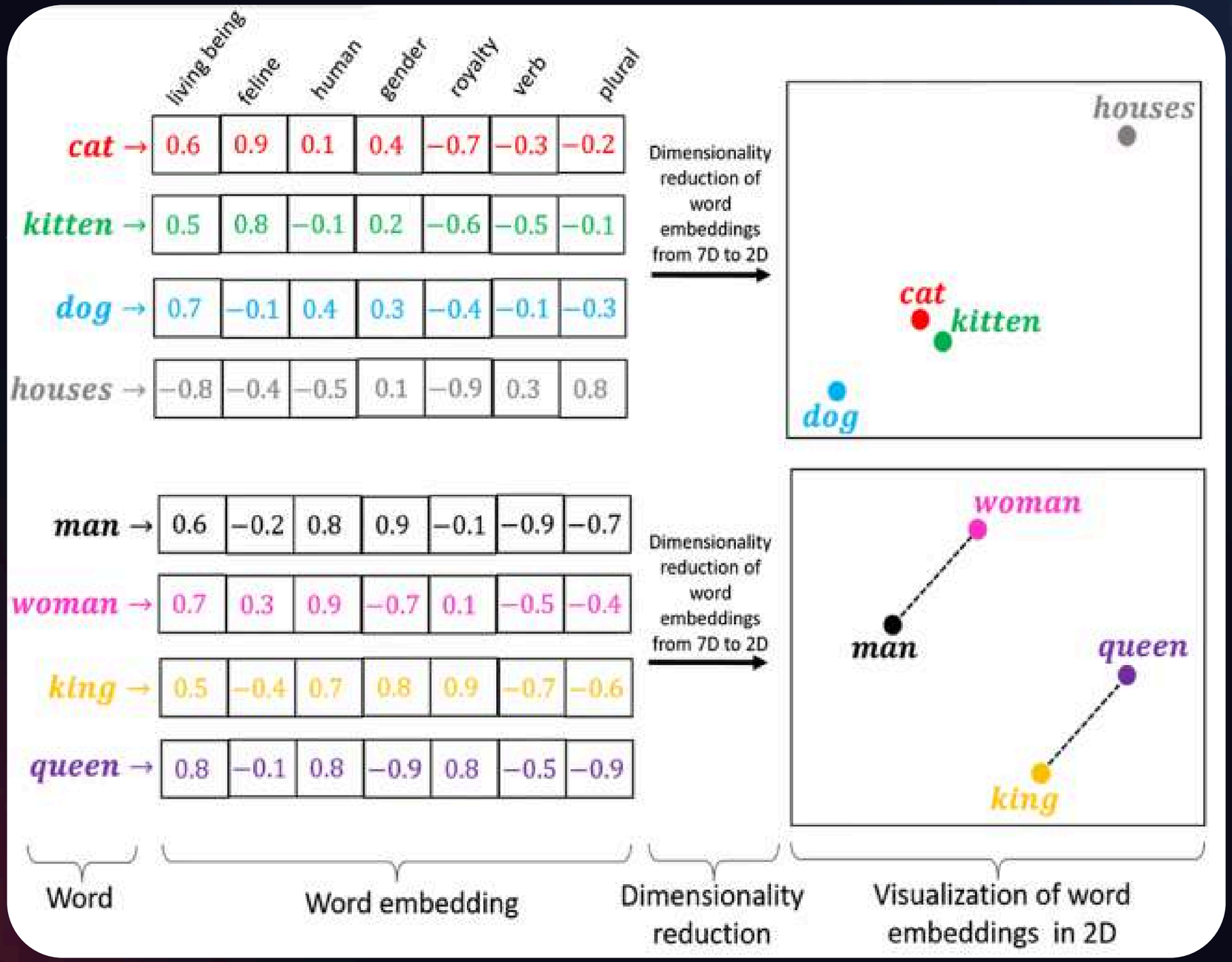
- CBOW (Continuous Bag of Words)
- SkipGram

Pre-Trained Word Embeddings

- Word2vec by Google 2013
- GloVe by Stanford (Global Vectors for Word Representation)
- fasttext by Facebook
- BERT (Bidirectional Encoder Representations from Transformers)

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

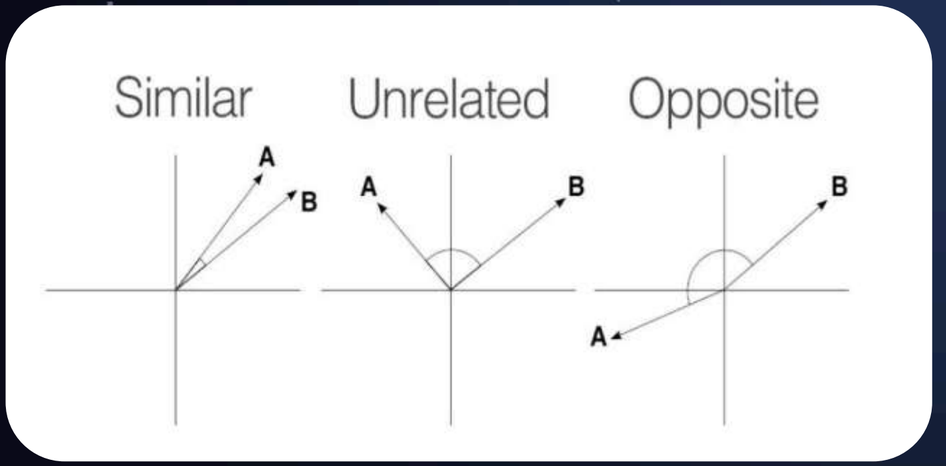
Visualizing High-Dimensional Space: https://www.youtube.com/watch?v=wvsE8jm1GzE&ab_channel=GoogleforDevelopers


living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Feature Extraction Techniques summary

Techniques	Main Feature	Use case
CountVectorizer	Coverts text to matrix of word counts	Text classification, topic modeling
TF-IDF (Term Frequency-Inverse Document Frequency)	Assign weights to words based on importance	Information retrieval, text classification
Word embeddings	Vector representation of words based on semantics and syntax	Text classification, Information Retrieval
Bag of words	Represents text as a vector of word frequencies	Text classification, Sentimental Analysis
Bag of n-grams	Capture frequency of sequences of n words	Text classification, Sentimental Analysis
Principal Component Analysis (PCA) , t-SNE	Reduces dimensionality of documents-term matrix	Text visualization, text compression
Part-of-speech (POS) tagging	Assigns parts of speech tag to each word in text	Named entity recognition, text classification
N-grams	sequences of contiguous words or characters, capturing local word dependencies	Captures context and word order information, useful in language modeling, machine translation, and text generation
Named Entity Recognition (NER)	identifies and classifies named entities (e.g., person names, organizations, locations) in text	Information extraction, entity linking, and improving search engine results

Text simmilarity



The dog bites the man == The man bites the dog?

- According to the lexical similarity, those two phrases are very close and almost identical because they have the same word set.
- For semantic similarity, they are completely different because they have different meanings despite the similarity of the word set.

Techniques are commonly used to compute text similarity

Cosine Similarity

Measures the cosine of the angle between two vectors representing the text.

Used with word embeddings or TF-IDF vectors to compute similarity.

Jaccard Similarity

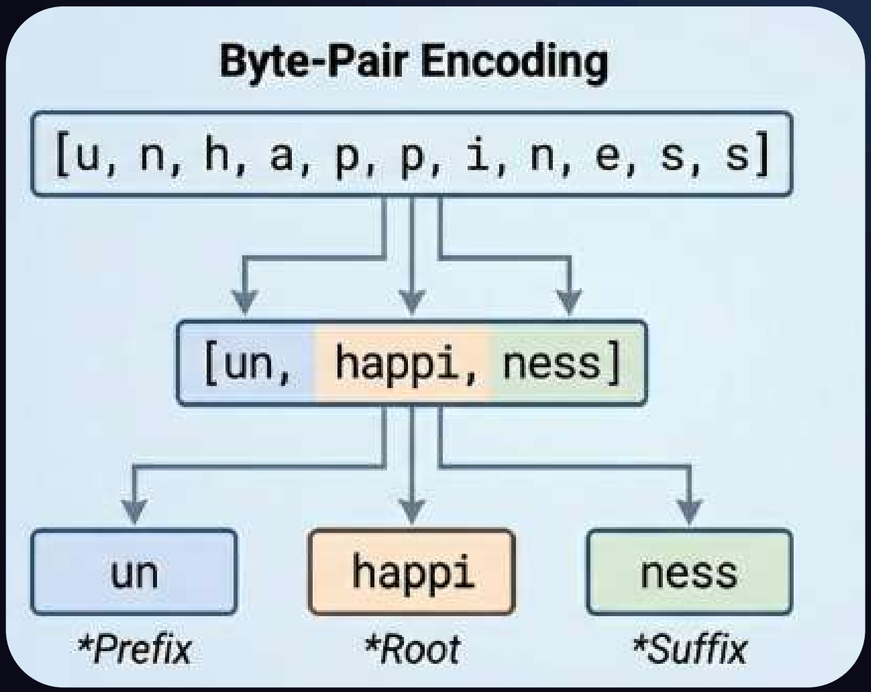
Measures the similarity between two sets by dividing the size of their intersection by the size of their union.

BERT and Similar Models

Pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) can be fine-tuned for text similarity tasks.

Vocab bottelneck

- Prob: Static dictionary fail on unseen words or misspellings
- Solution : BPE /wordPiece algos balance character level flexibility with word-level efficiency
- Impact : eliminates the destructive [UNK] token, enables handling of morphologically rich language



The Tokenizer

Scenario: A noisy Reddit dataset contains slang and emojis.

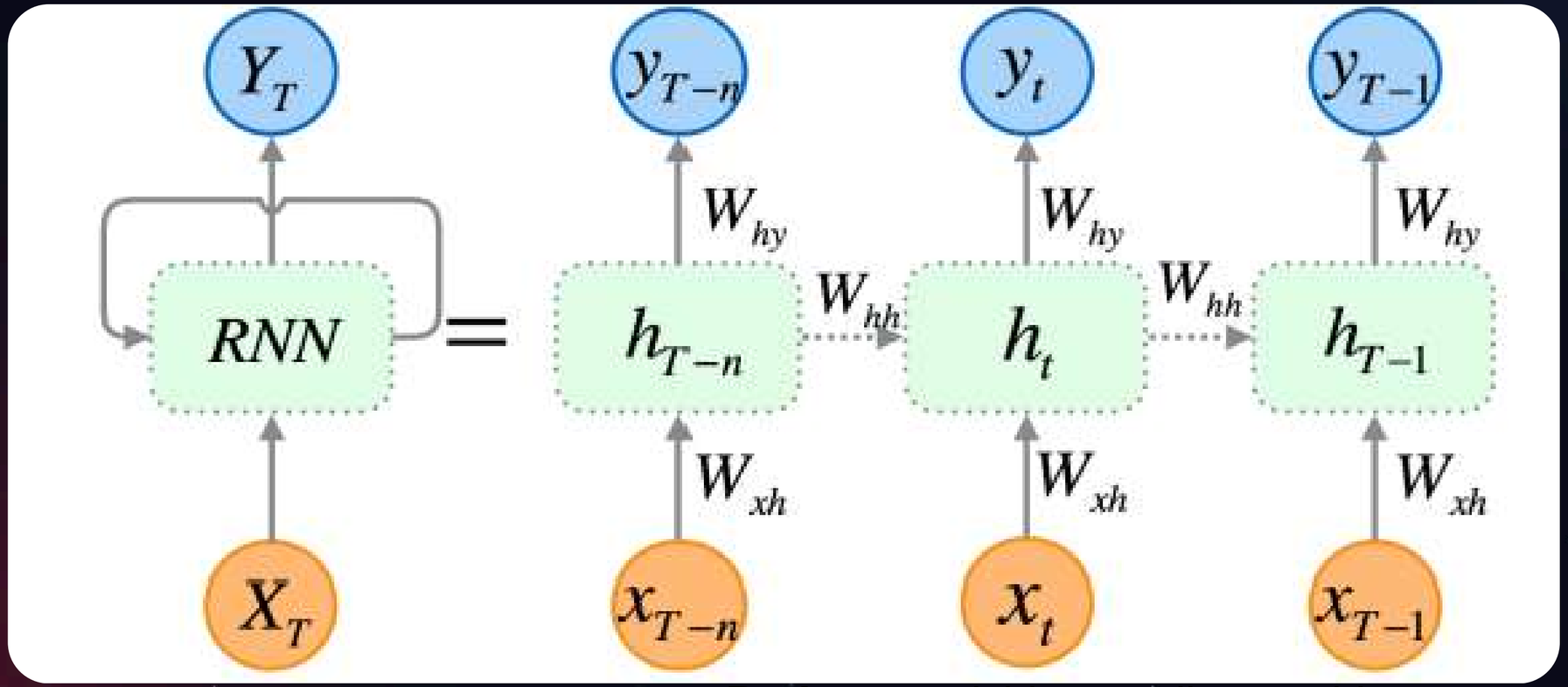
input text	Output log

`text = 'Captain Smurgleblorp 😜'`	**BertTokenizer Output**
	`>>> ['Captain', '[UNK]', '[UNK]']`
	
	GPT2Tokenizer (Byte-Level) Output
	`>>> ['Captain', 'ĠSm', 'urg', 'le', 'bl', 'orp', 'Ġ😜']`

CRITICAL FAILURE: 'Smurgleblorp' and Emoji are ost.

SUCCESS: Neologism decomposed into subwords. Emoji preserved.

Actionable Insight: For user-generated content, Byte-Level BPE is mandatory to prevent feature collapse on rare tokens.



Prolem: vanishing gradient problem

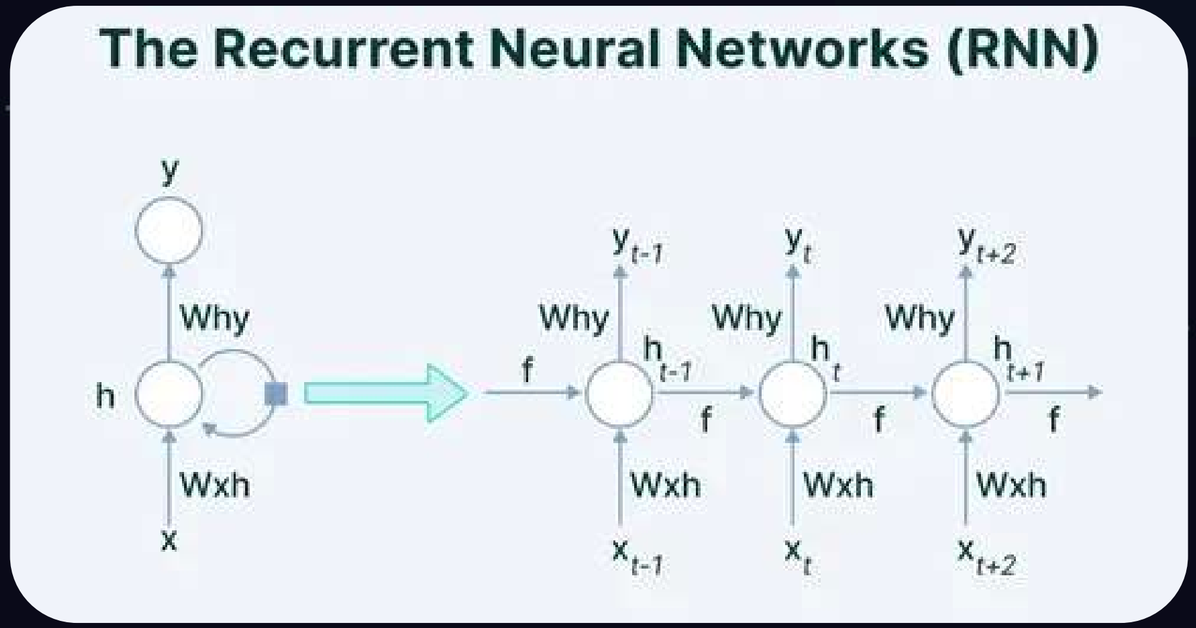
RNN

Reccurent Neural Networks (RNN) :

The RNN has is highly preferred method , especially for sequential data.

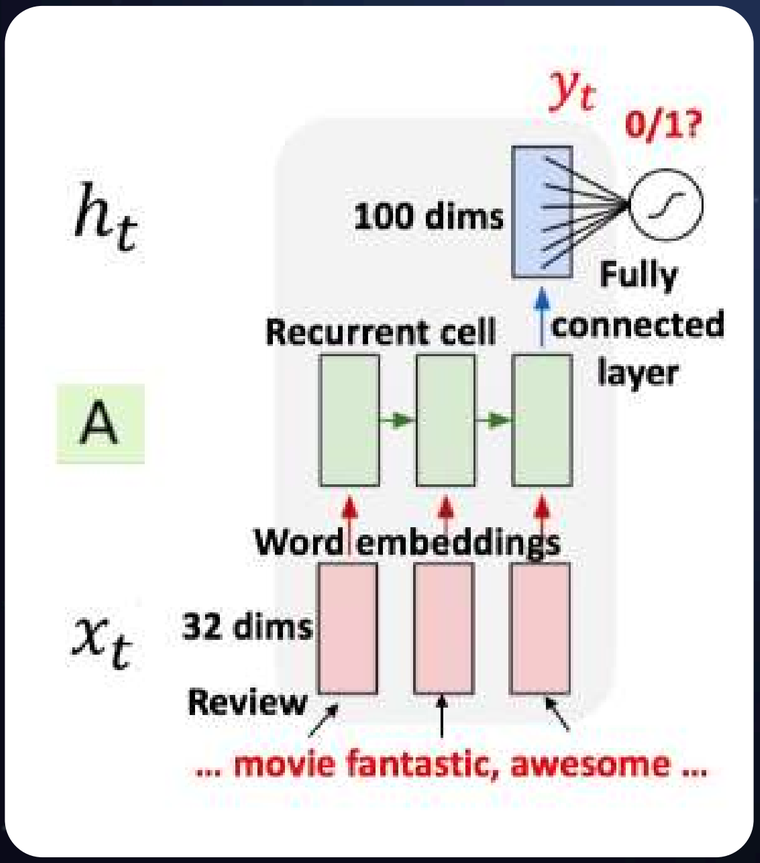
Every node at a time step consists of an input from the previous node, and it proceeds using a feedback loop.

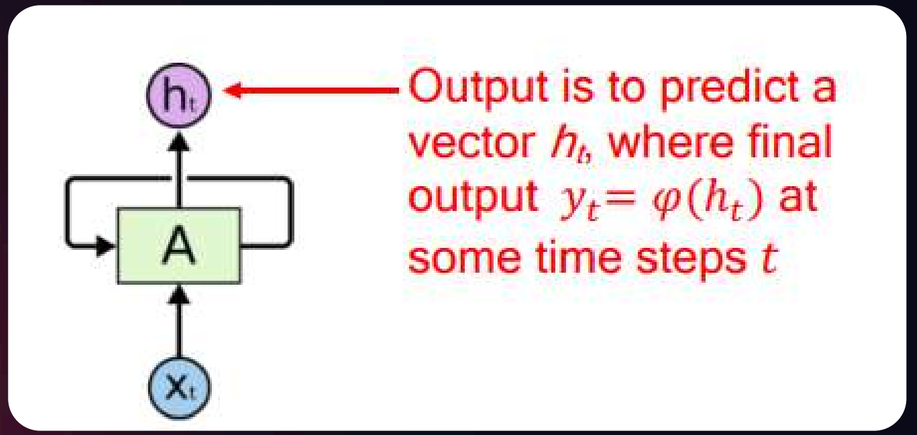
In RNN, each node generates a current hidden state and its output is obtained by using the given input and previous hidden state as follows:

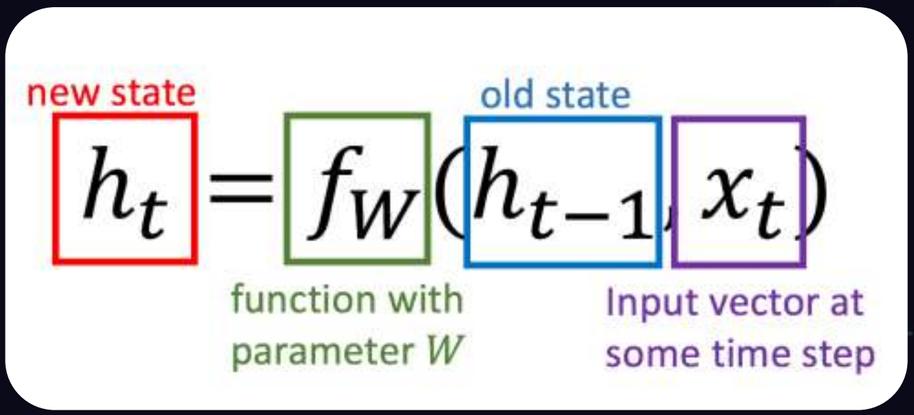


RNN

are networks with loops, allowing information to persist. In the above diagram, a chunk of neural network, A = fw, looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next.



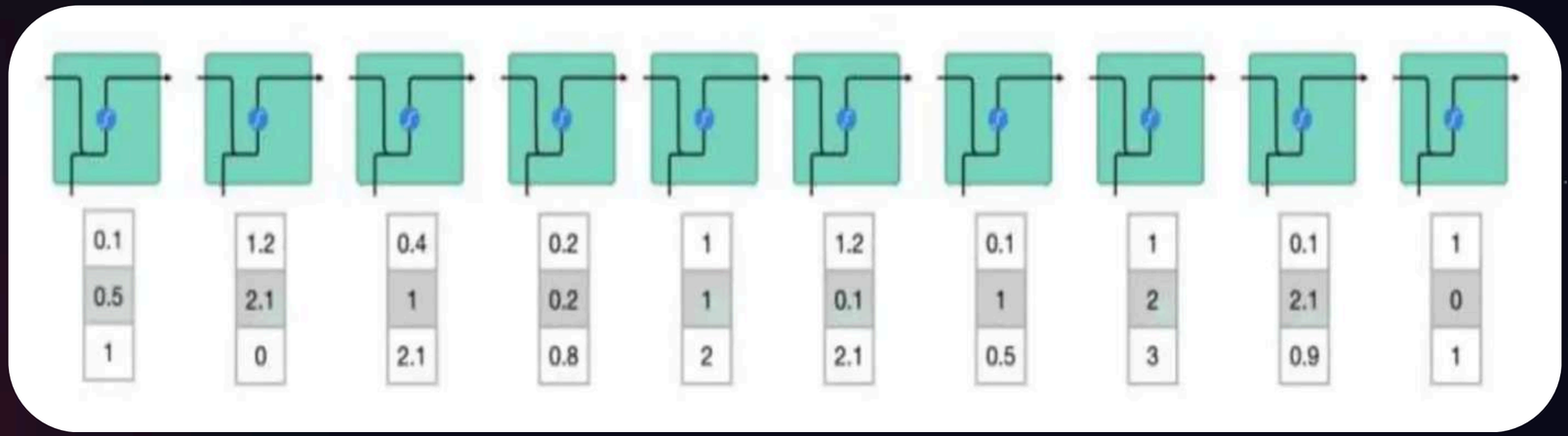




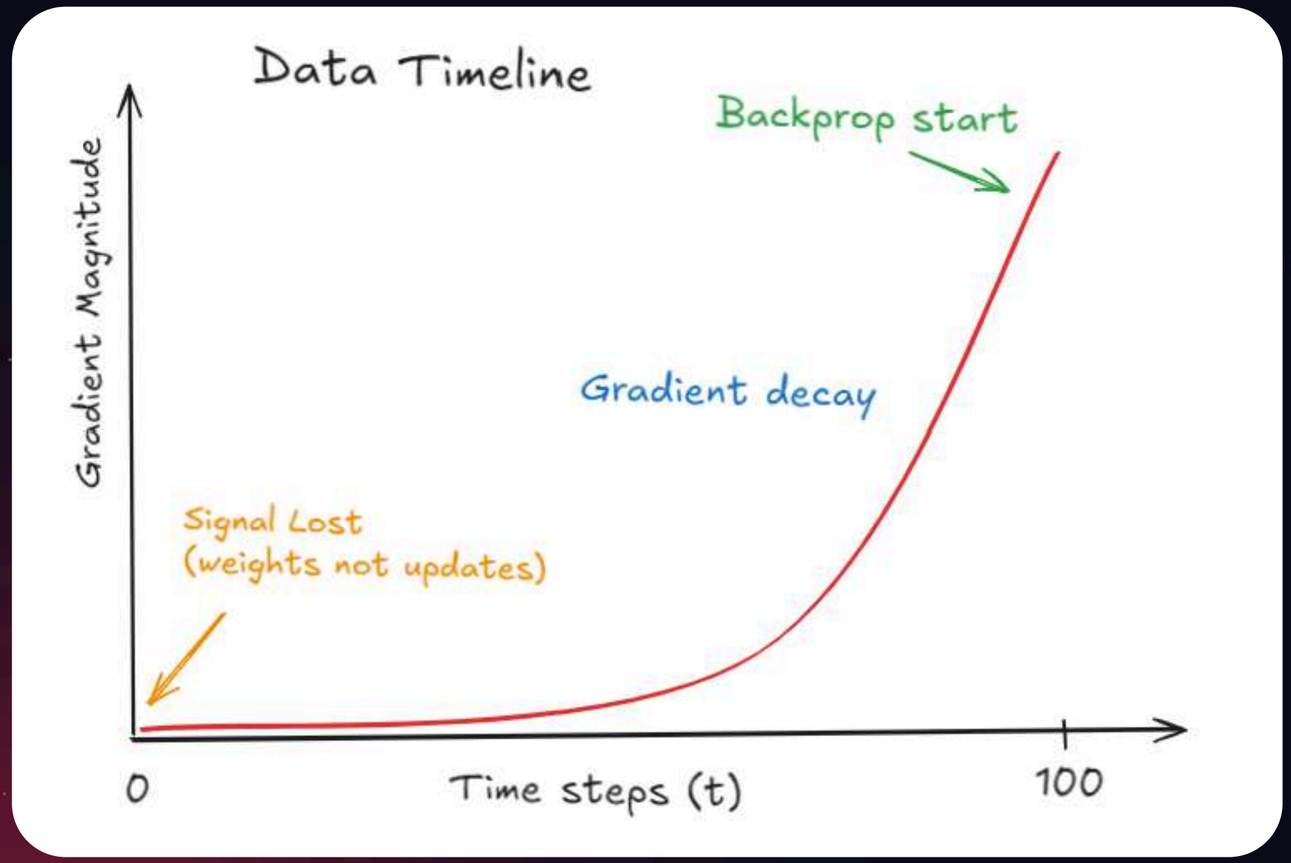
RNN how it works

- Sequential Processing
- processes the sequence of vectors one by one.
- While processing, it passes the previous hidden state to the next step of the sequence.
- The hidden state acts as the neural network's memory, holding information on previous data the network has seen before.

Visual: the processing of a sequence one by one with a series of vector inputs.



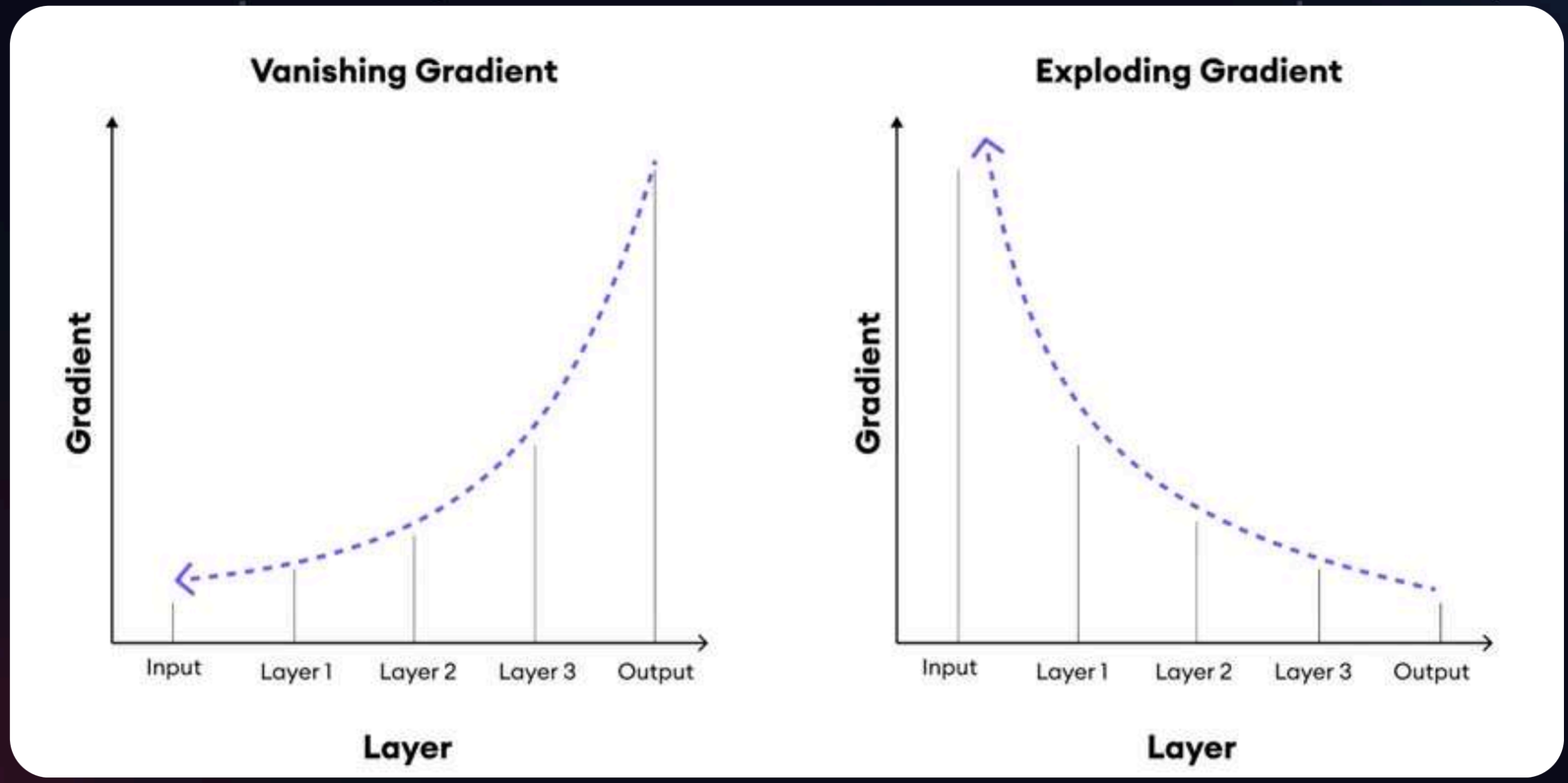
Problems with RNN



RNN be like :

Backpropagation Through Time (BPTT) is the standard algorithm used to train RNNs :
gradients must low backwards from T to 0

Problems with RNN

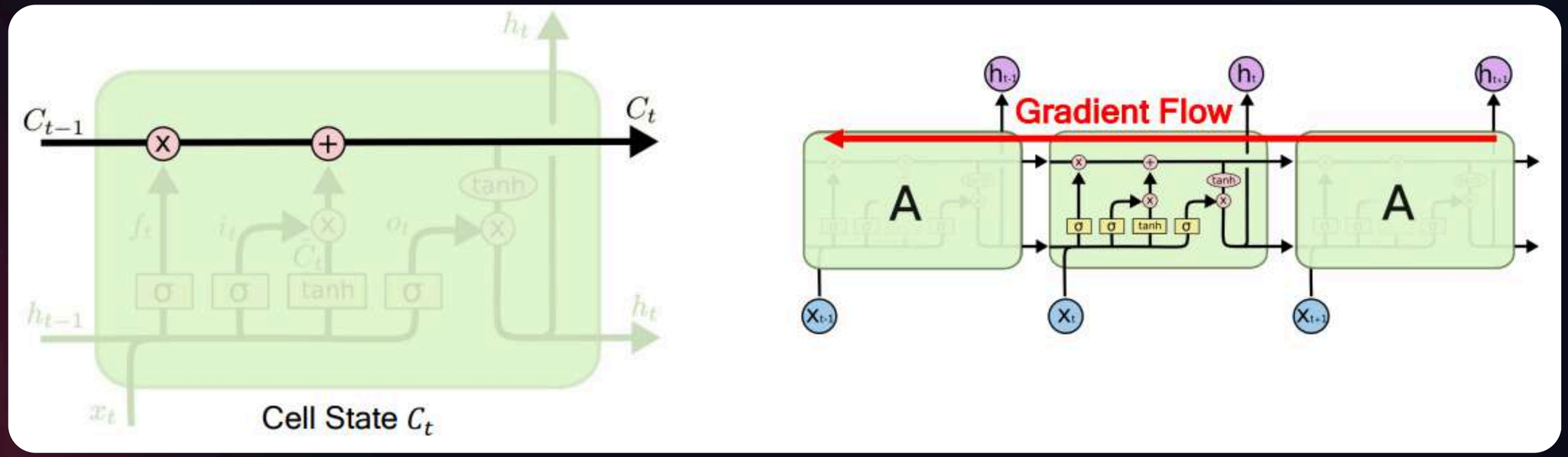


sometimes solved using :
Gradient Clipping,

LSTM

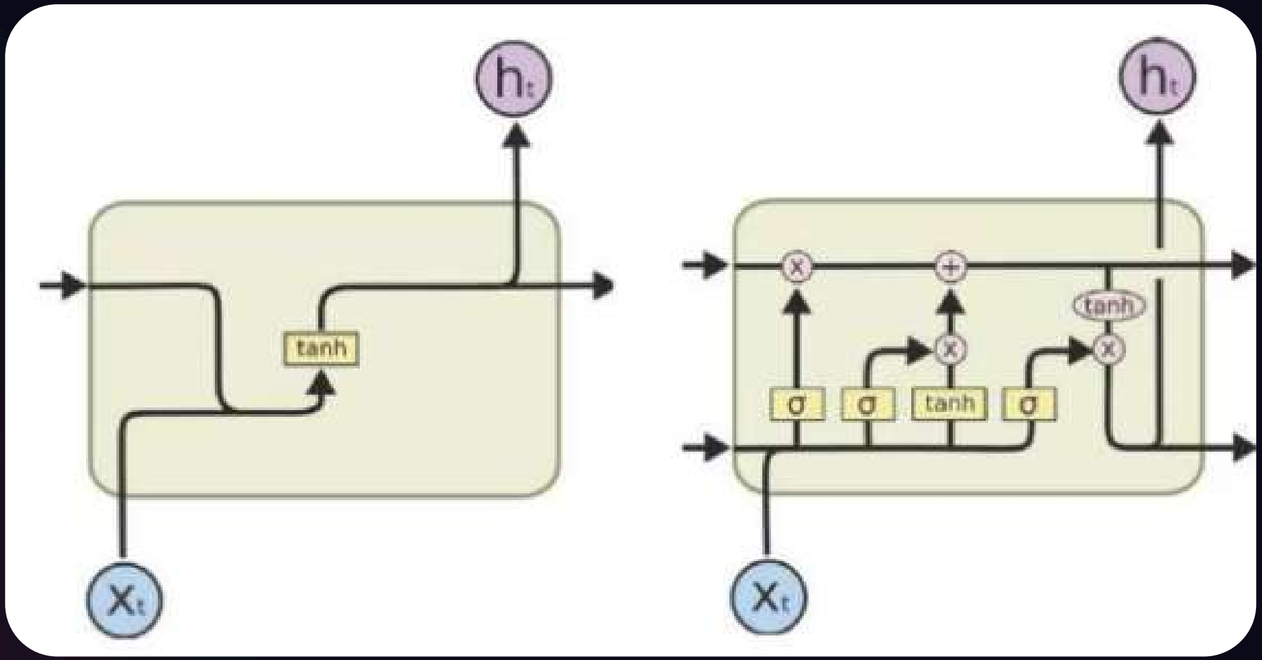
A special type of RNN designed to solve the problem of remembering things over long sequences.
It has a more complex internal structure that lets it **decide what to remember and what to forget** more effectively.

Gating mechanism



Drawbacks of LSTM

- LSTM is even more slower to train as compared to RNN.
- Vanishing gradient problem still persists.
- Sequential computation inhibits parallelization.
- Distance between positions is linear



These models were the **standard** for sequential data but struggled with long-range dependencies because gradients would "**vanish**" before reaching the beginning of the sentence.

These models were the **standard** for sequential data but struggled with long-range dependencies because gradients would "**vanish**" before reaching the beginning of the sentence.

Researchers realized forcing an entire sentence into one **fixed-length** vector created a "**bottleneck**". As sentences got **longer**, the model's performance dropped because it couldn't "**remember**" the specific details of the input while generating the output.

Transformer

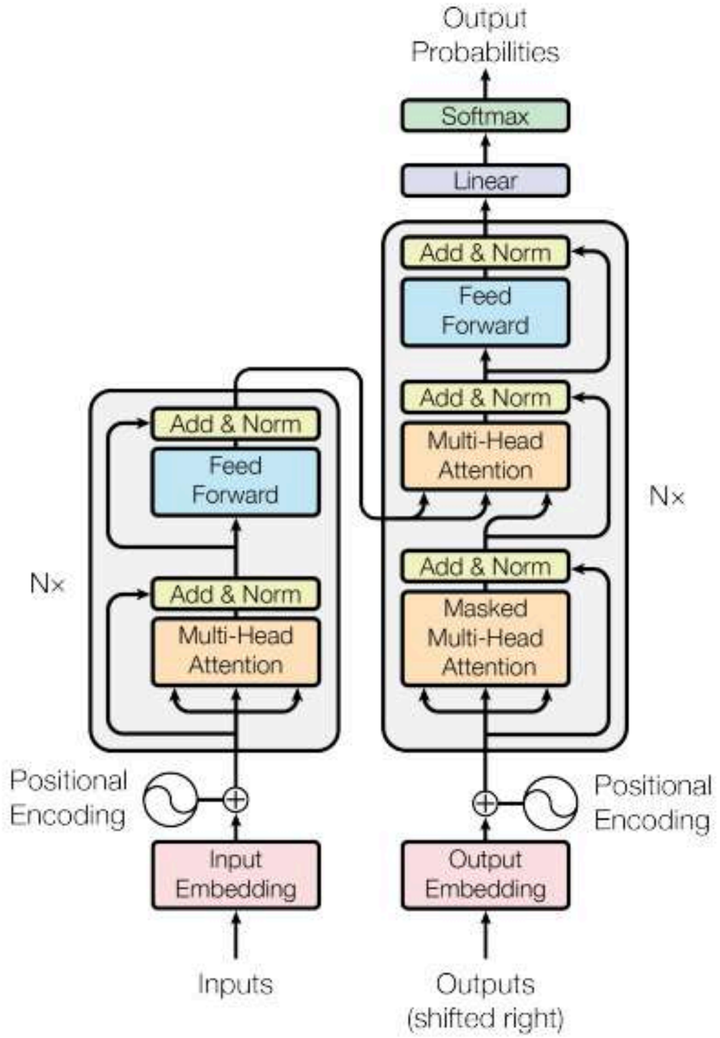
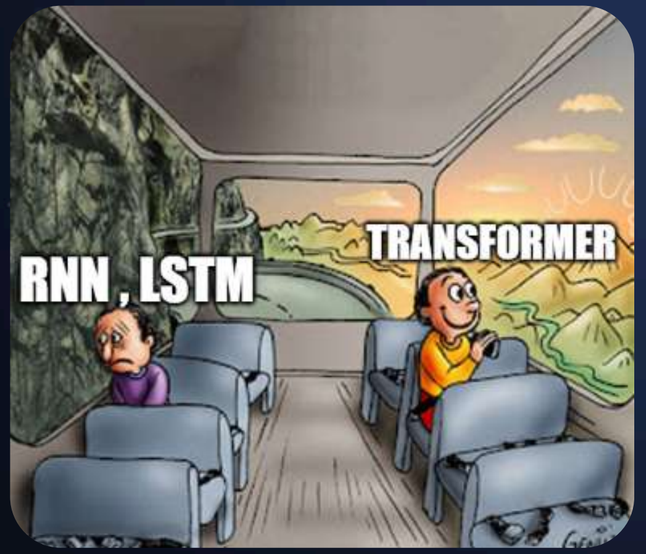


Figure 1: The Transformer - model architecture.



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Transformer

The Interaction Matrix: Measures the raw similarity between the query and Key vectors.

The Probability Distnibtion: Detrmines how such focus to place on each token. Controlled by Temperature (T).

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

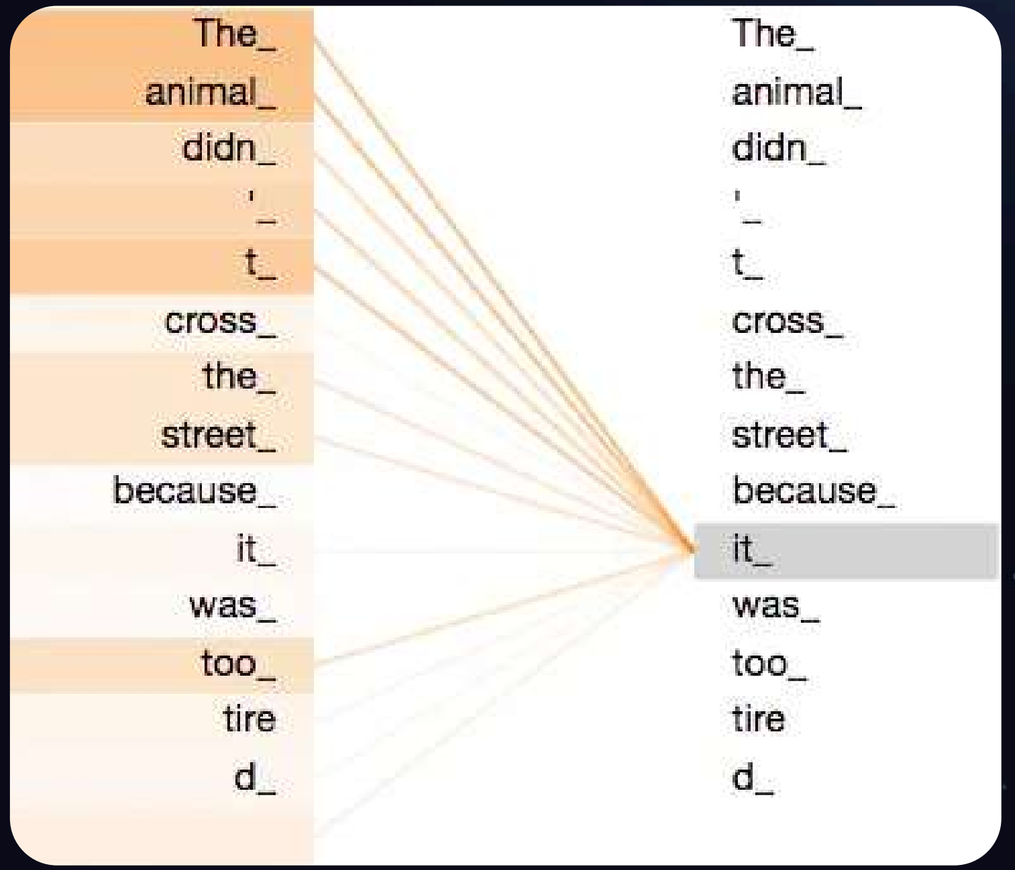
The Scaling Factor:
Prevents dot products from growing too large. Without this, the Softmax function enters regions with extremely small gradients (vanishing gradient problem).

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Query (Q): Represents what the token is searching for
- Key (K): Represents what information the token offers.
- Value (V): The actual semantic content to be aggregated.

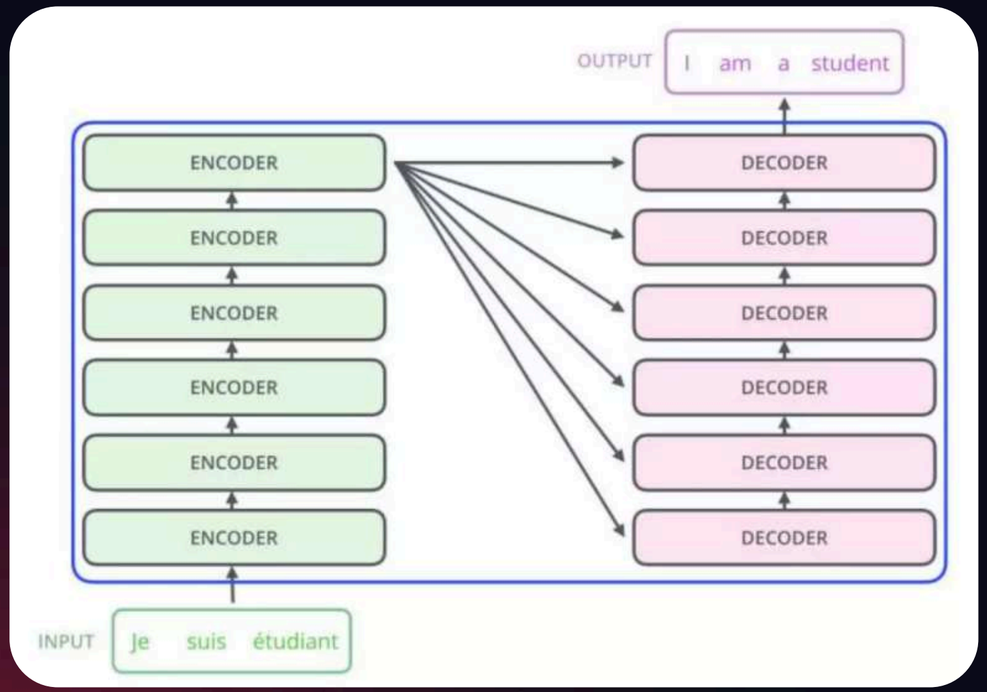
Attention idea

- Allowing the decoder to access the entire encoded input sequence
- Induce attention weights over the input sequence to prioritize the set of positions where relevant information is present for generating the next output token



When encoding the word “it”, the attention mechanism is focusing on “The Animal

Transformer



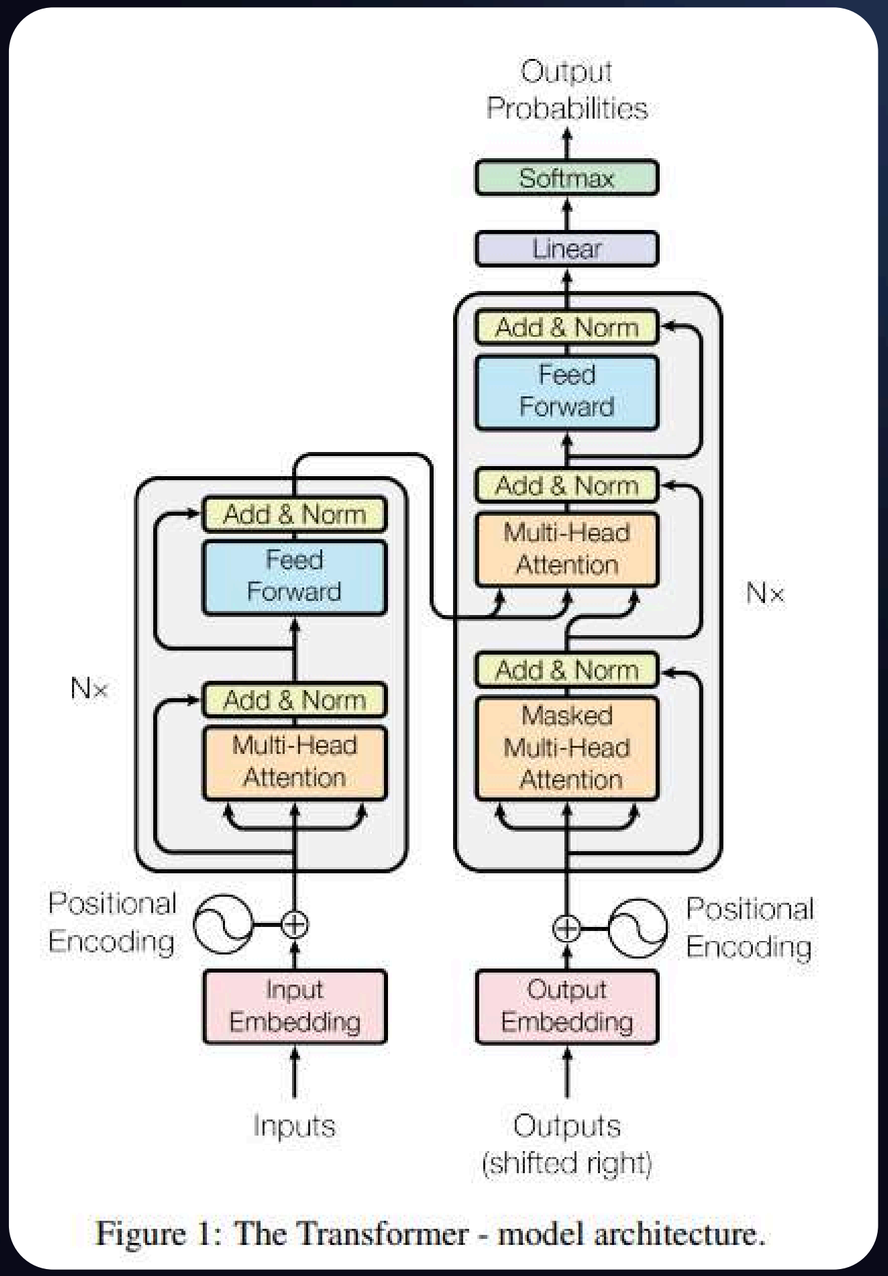
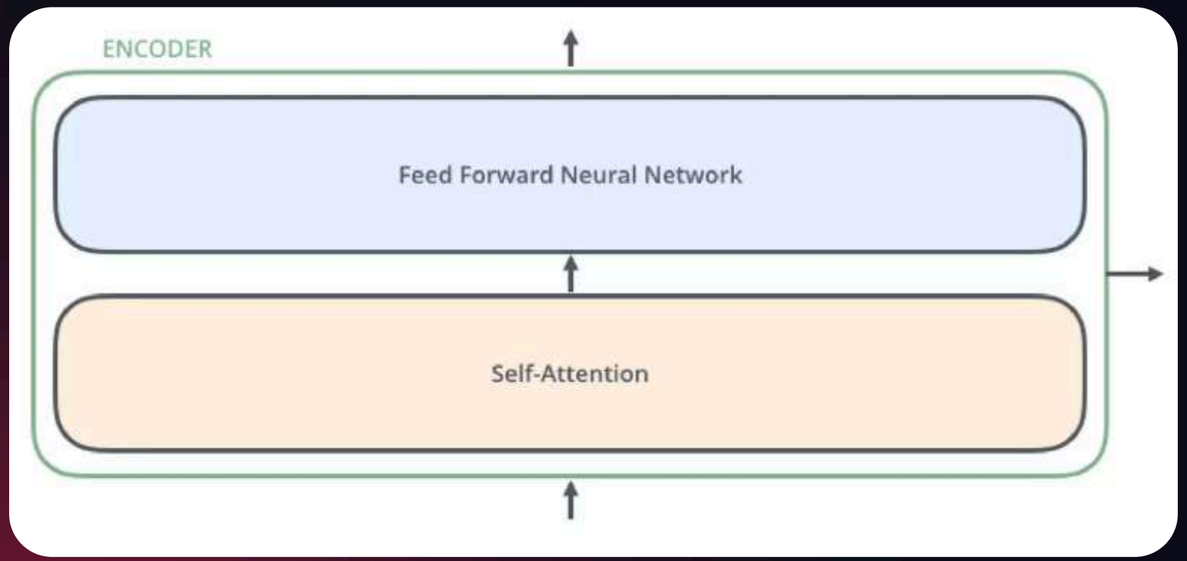
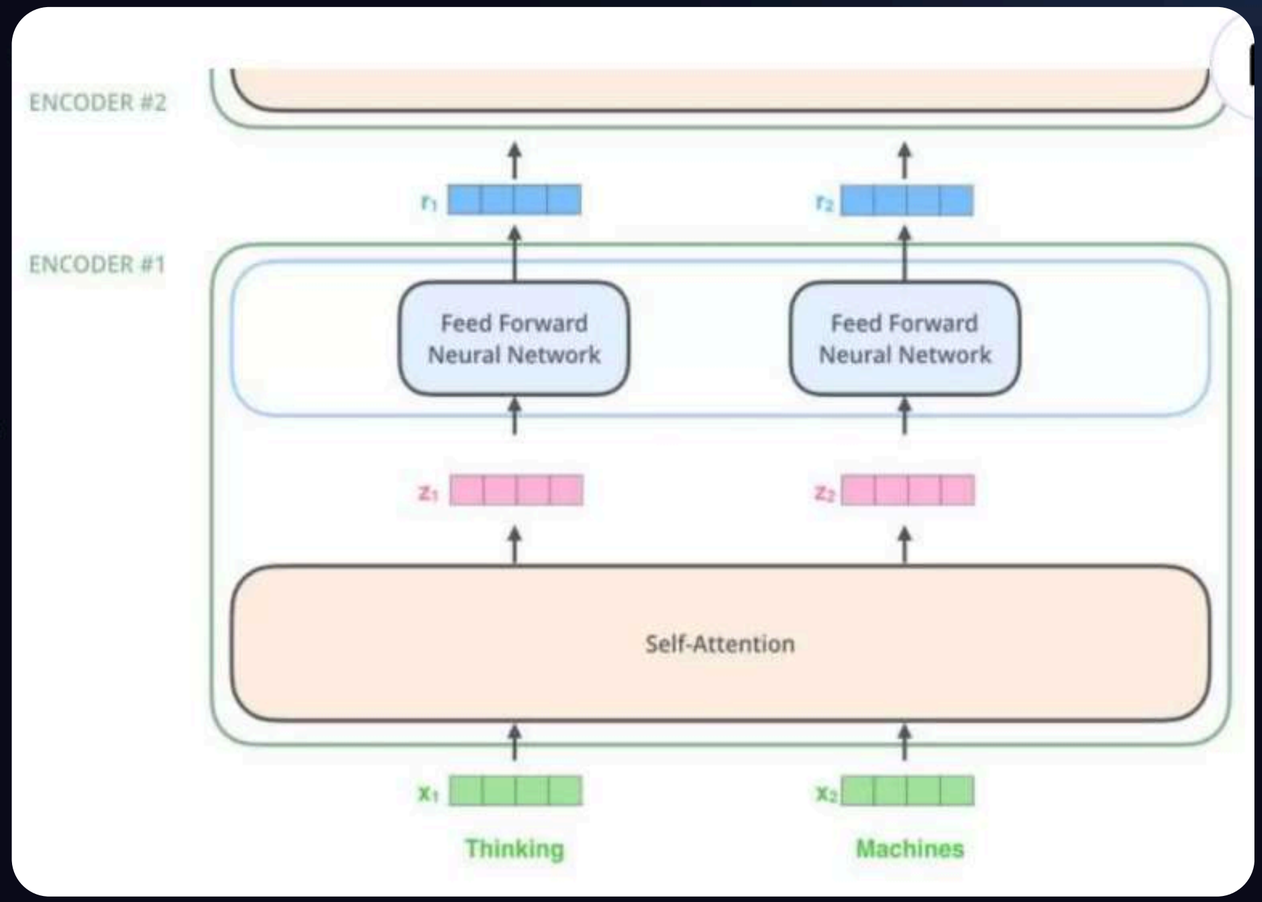


Figure 1: The Transformer - model architecture.

Encoder Block

All identical in structure (yet they do not share weights).
Each one is broken down into two sub-layers
Word → Embedding → Positional Embedding → Final Vector, framed as Context.

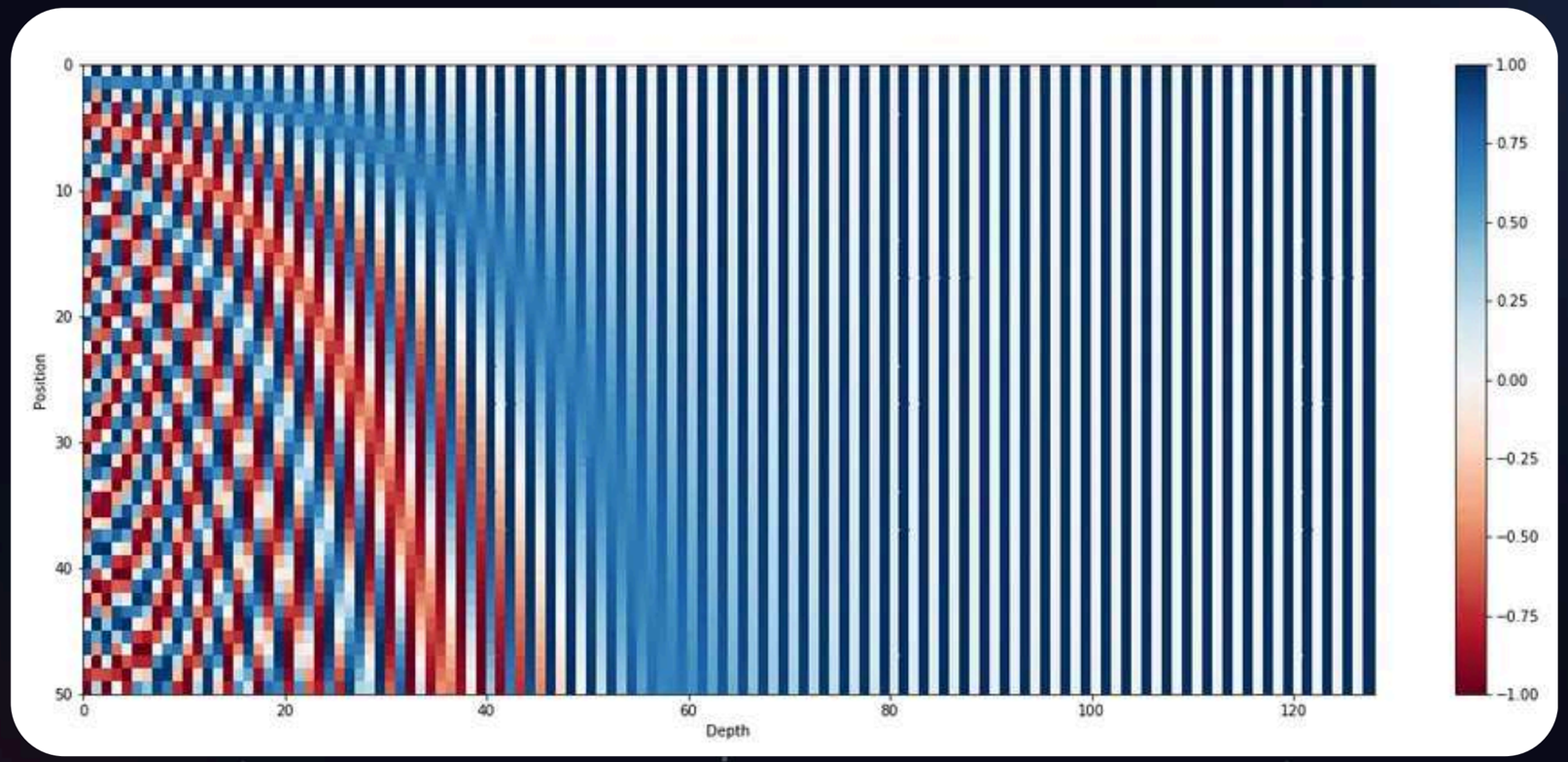




Transformers process all words in a sentence simultaneously, they are permutation-invariant

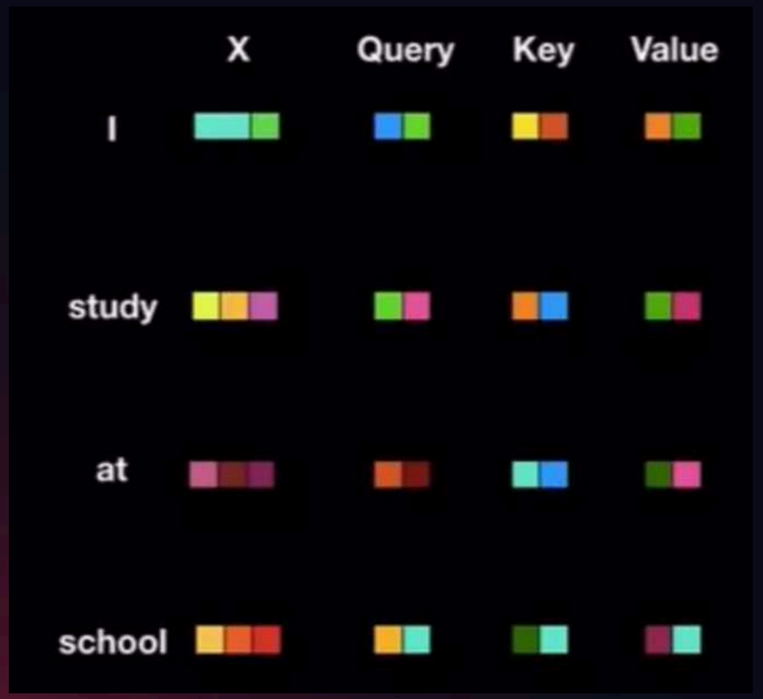
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

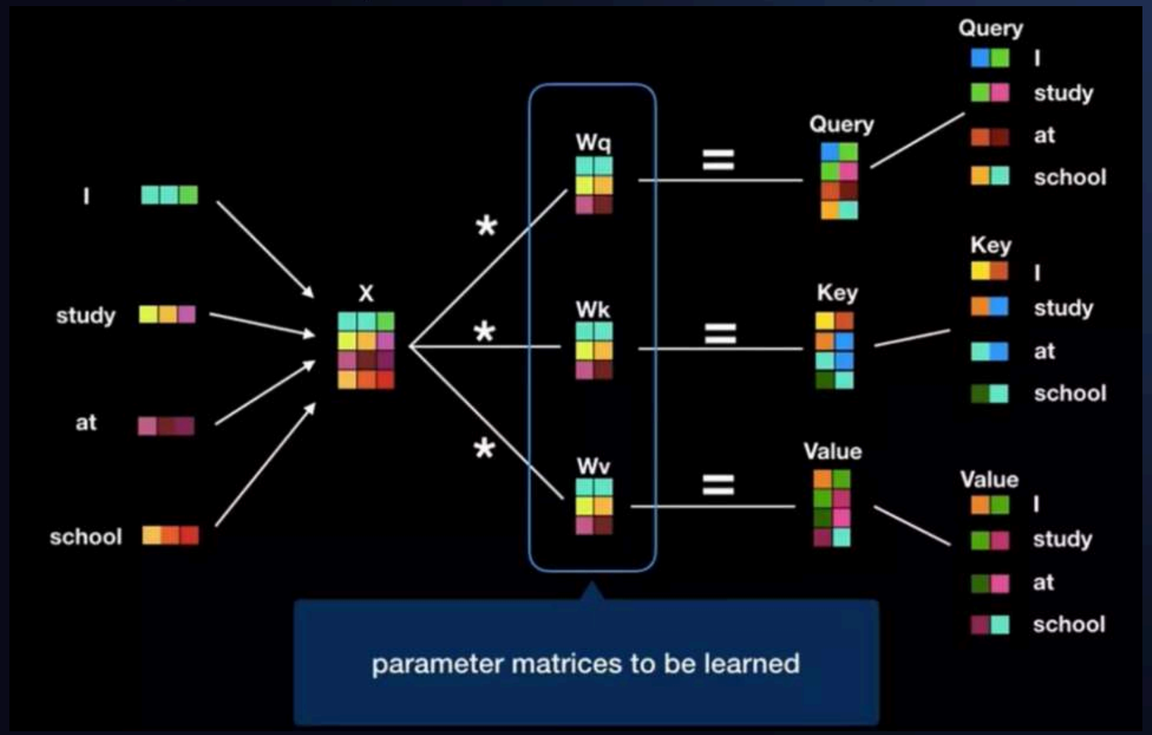
sin(A + B) = sin(A).cos(B) + cos(A).sin(B)



Dimension (i)	Denominator	Speed of Change	Frequency	Visual in Heatmap
0 (Low)	Small (1)	Fast	**High**	Flickering/Noisy
511 (High)	Large (10,000)	Slow	**Low**	Solid/Smooth

Self attention

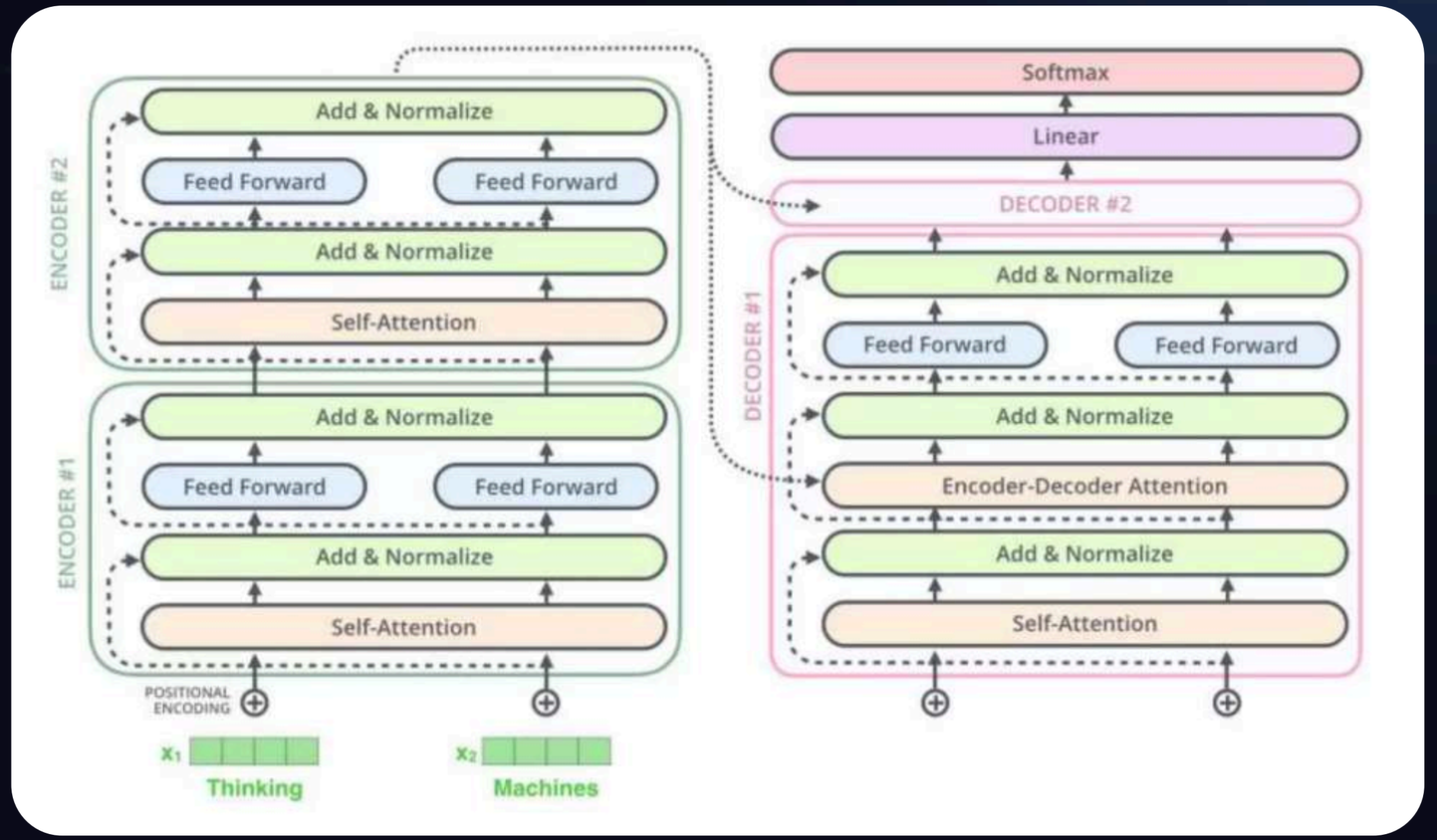


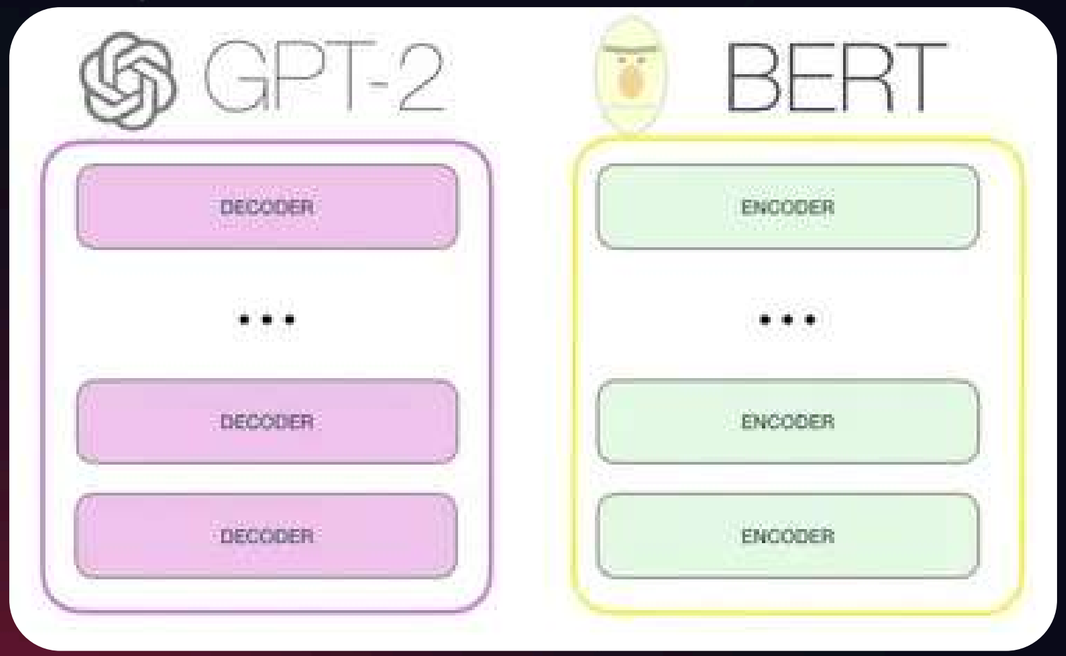


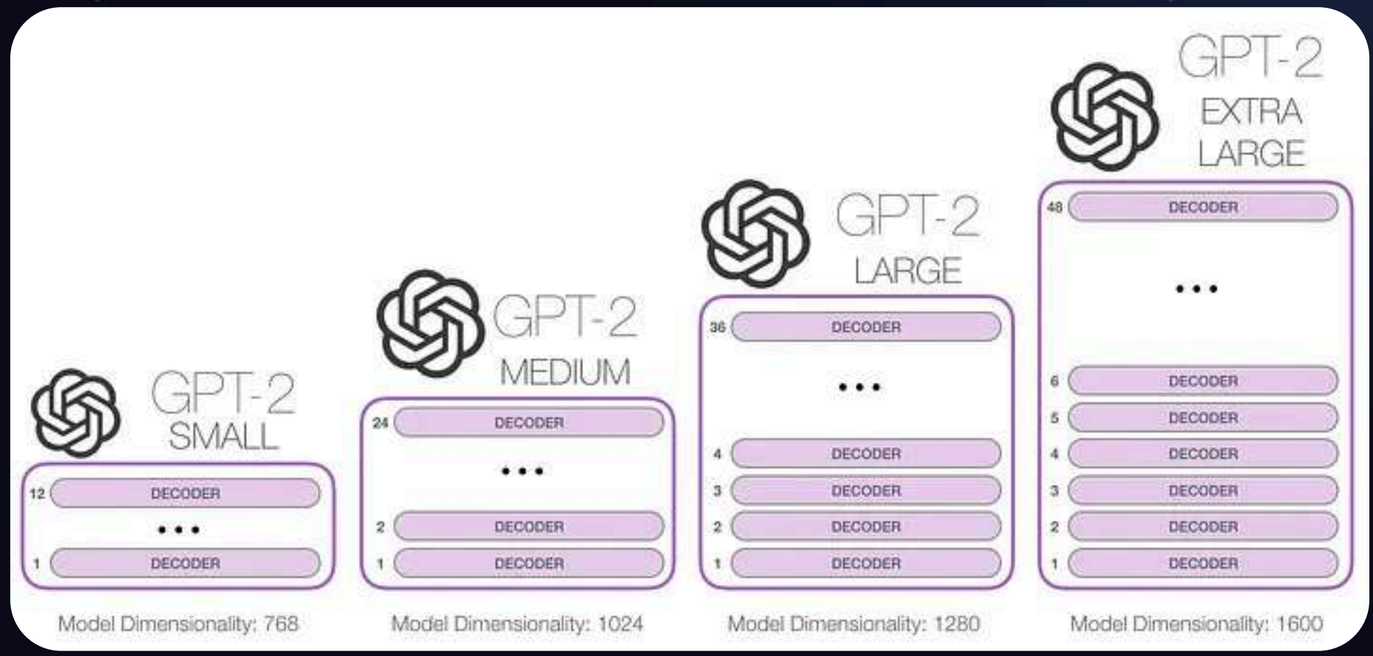
	Query * Key^T	Score	Softmax	Value	Softmax * Value	Σ Softmax * Value (Attention layer output)
I	I * I	= 130	0.92	I		
	I * study	= 50	0.05	study		
	I * at	= 20	0.02	at		
	I * school	= 10	0.01	school		
study	study * I	= 30	0.02			
	study * study	= 110	0.70			
	study * at	= 20	0.03			
	study * school	= 70	0.25			

Complete architecture of transformer

Transformers, GPT-2, and BERT
A transformer uses Encoder stack to model input and uses Decoder stack to model output (using input information from encoder side). But if we do not have input, we just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us GPT. If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.




GPT-2
DECODER
DECODER
DECODER
BERT
ENCODER
ENCODER
ENCODER


GPT-2
SMALL
12
DECODER
1
DECODER
Model Dimensionality: 768
GPT-2
MEDIUM
24
DECODER
2
DECODER
1
DECODER
Model Dimensionality: 1024
GPT-2
LARGE
36
DECODER
4
DECODER
3
DECODER
2
DECODER
1
DECODER
Model Dimensionality: 1280
GPT-2
EXTRA
LARGE
48
DECODER
6
DECODER
5
DECODER
4
DECODER
3
DECODER
2
DECODER
1
DECODER
Model Dimensionality: 1600

Encoder decoder

Feature	Encoder (The Reader) Bert	Decoder (The Writer) GPT
Attention Type	Bidirectional (Looks everywhere)	Masked (Looks only at the past)
Goal	Understand the Input	Generate the Output
Task Example	Classifying an email as "Spam"	Writing a response to an email
Logic	"What does this mean?"	"What word comes next?"

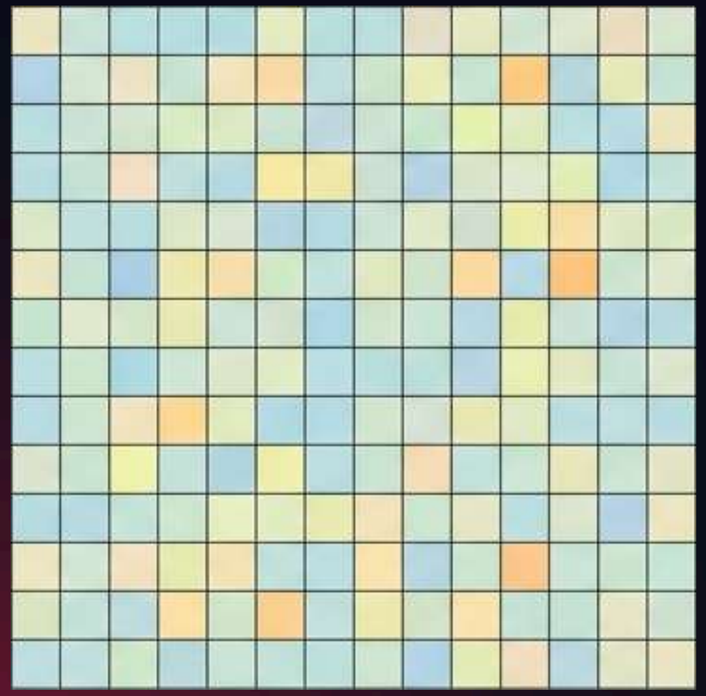
GPT-3

- OpenAI's third-generation Generative Pretrained Transformer
- GPT-3, is a general-purpose language algorithm that uses machine learning to translate text, answer questions, and write text predictively.
- It analyzes a series of terms, text and other data, then elaborates on those examples in order to generate fully original production in the form of an article or an image.

The Temperature

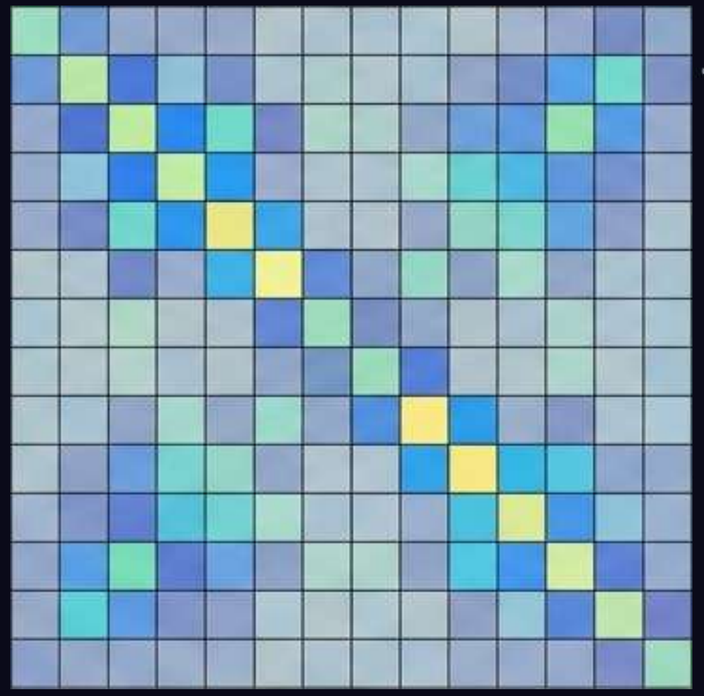
Visualizing Softmax behavior on attention scores.

High Temperature (T=2.0)



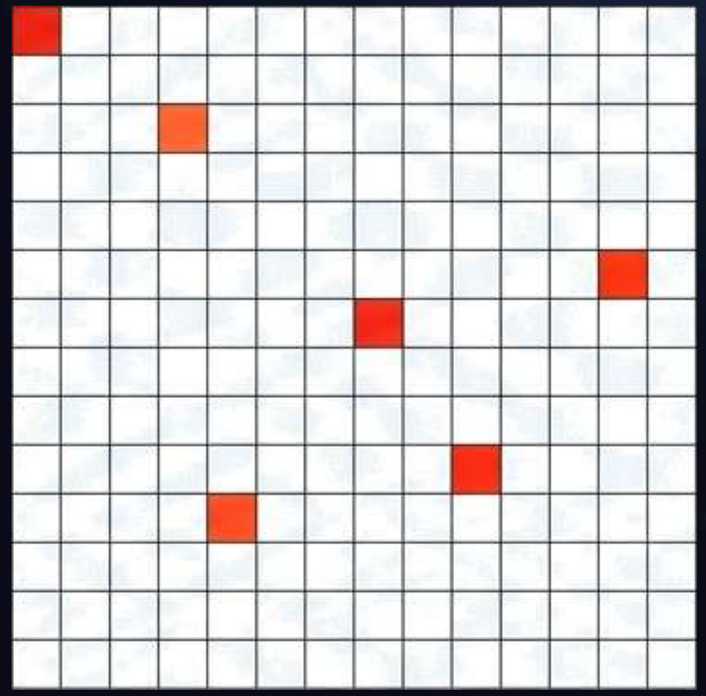
Uniform Distribution (Blurry) . The mode1 is "uncertain" and attends to everything equally.

Standard Tempenatune (T=1.0)



Balanced Focus. The model attends to relevant semantic relationships.

Low Temperature (T=0.5)



Sharp Distribution (Peaked) . The model becomes deterministic and may overf1t to single tokens.

Where to find pretrained LLMs ?

NLP vs LLM

Hugging Face

Model types

Text-to-text

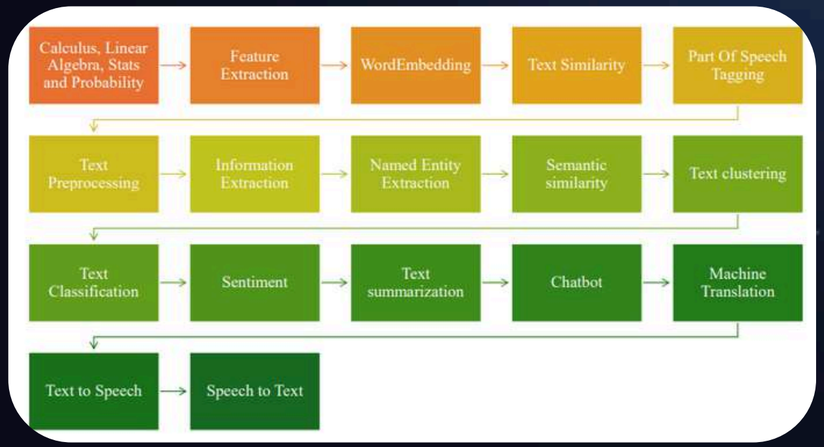
Text-to-image

Text-to-video

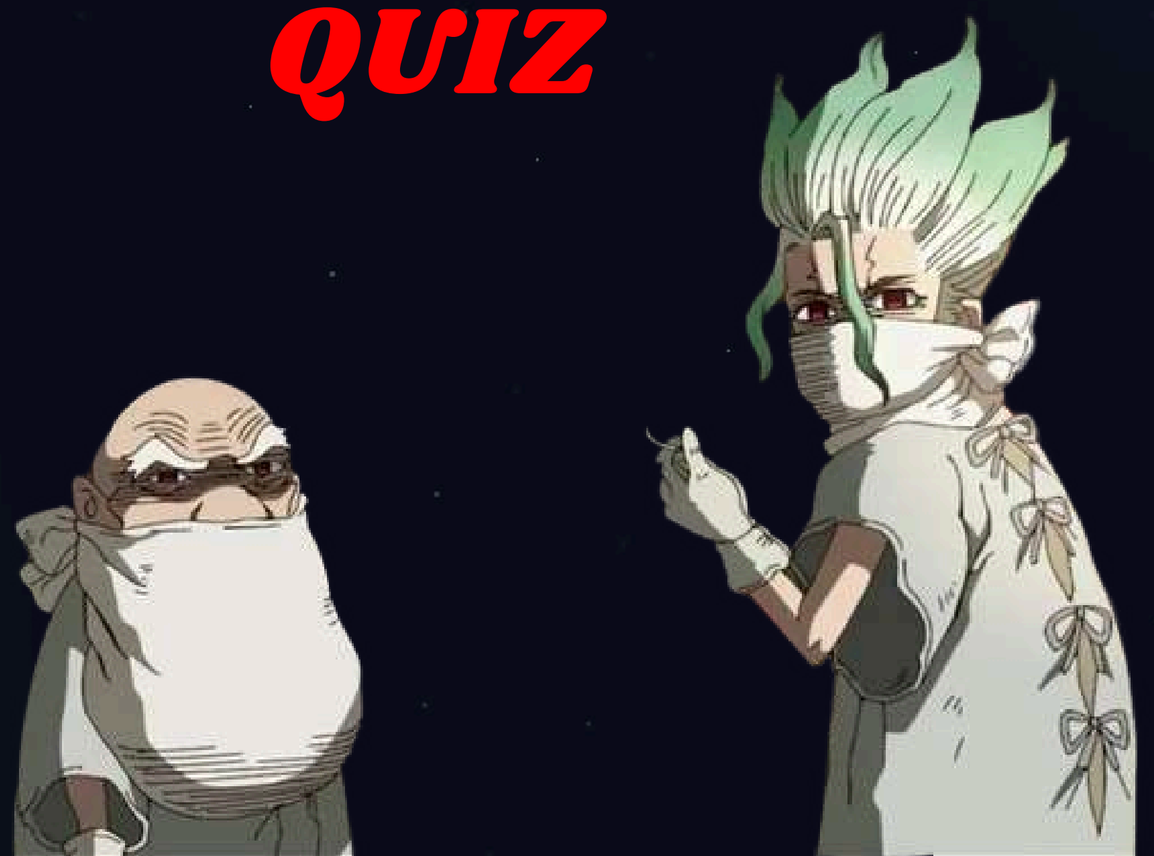
Sentence-similarity

Projects we should try for a better understanding of NLP

- Sentiment Analysis
- Question Answering System
- Named Entity Recognition
- Fake News Detection
- Topic Modeling
- Text Similarity
- Text summarization and machine translation
- Next word Prediction
- LLM applications using RAG
- Fine-tune a model for a specific NLP task
- Constructing your own LLM, inspired by models like Llama 2



QUIZ



Resources

- RNN presentation
- Illustrated transformer

Important Reading to know about NLP approaches

Level	Topic	Weblink
Beginning Level	RNN and LSTM	https://arxiv.org/pdf/1808.03314.pdf
	Word2Vec	https://arxiv.org/pdf/1301.3781
	Attention for Translation Task	https://arxiv.org/abs/1409.0473
	Attention is all you Need	https://arxiv.org/abs/1706.03762
	BERT	https://arxiv.org/abs/1810.04805
	GPT-1	https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf
	GPT-2	https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
	Research Papers	https://www.kaggle.com/discussions/general/236973
Advanced Level	RLHF	https://arxiv.org/abs/2203.02155
	LIama2	https://arxiv.org/abs/2307.09288
	PaLM2	https://blog.google/technology/ai/google-palm-2-ai-large-language-model/
	Mistral : Mixture of Experts	https://mistral.ai/news/mixtral-of-experts/
	GPT -4	https://openai.com/research/gpt-4
	Gemini AI	https://blog.google/technology/ai/google-gemini-ai

THANK YOU FOR YOUR ATTENTION!!

Where to find me?

Goodnight

Linkedin Profile

MedArbiNsibi