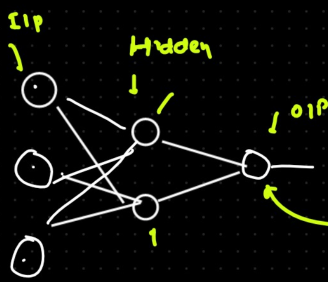
GEN-AI History

1. Languge Modelling ✓
2. RNN ⟶ Transformer
3. LLM
4. BERT
5. GPT
7. ChatGPT training
8. Transfer Learning & Fine tuning NLP

DL
Type of NN [NN is a fundamental unit of DL]

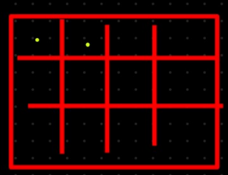
1. ANN
2. CNN
3. RNN



Convolution + Pooling + {fully connected network}
ANN

ANN + Feedback loop

- Structure Data



Col → Num, Cat
Num → Cont, Discl
Cat → Nom, Ord

- Image / videos

Sequence data
text / Audio

Discriminative Model v/s Generative Model

Discriminative Model

Layer

Model → Y, 1

1. Specific task
2. Limited data

Generative Model

Model →

1. genral task
2. huge data

Generative Model type

1. image to image
2. text to text

} homogenous Model

3. image to text
4. text to image

} hetrogenous Model or (Multimodel) or (Diffusion model)

Visual + Seq

GENAI → LLM ⇒ text to text

Transformer ⇒ Bert & gpt

vision based task also (GPT 4 v, gemini Pro, Donut, CLIP, whisper)

CNN → Computer vision ⇒

1. image classification
2. Object Detection
3. " " Segmentation
4. " " tracking
5. OCR

1. Efficent Det
2. VGG
3. Resnet
4. Densenet

1. YOLO
2. SSD
3. fasterRCNN

1. MaskRCNN
2. U-net
3. V-net

Transfer Learning & Finetuning

Use the Previous Learning for the future task

Modified Previous Learning According to requirement

Ex:-

teenage → transferring → Adult

Cycle → ① balance ② Traffic

Motorcycle

balancing traffic | Gearbox ACC

Transfer Learning & Finetuning (CNN) (Computer vision)

1. Imagenet
2. COCO
3. CIFAR
4. Pascalvoc

CNN → Same knowledge classification

Finetuning — Weight

living, non-living, human, animal, bird — shape, face, edge

Data generation

1. GANs → ANN
2. Autoencoder → ANN

Unsupervied

1

2

NLP ⇒ Sequence data, text data

1. Summrization
2. Generation (word, sentence)
3. translation (
4. classification

Model

1. RNN → 1987
2. LSTM → 1997
3. GRU → 2014

training ↓

text → Model → Prediction

RNN | LSTM

↑ supervised

text	target
Sunny is good	+ve
sunny is bad	-ve

$\boxed{\text{Languge Modelling}}$

complete timeline of the evalution of LLM

RNN (1987) → LSTM (1997) → GRU (2014) → Encoder & Decoder (2014) → E&D Attention (2016) → ULMFIT (2018) → Transformer (2018) → BERT (2019) → GPT (2019)

1) Mapping (Seq to Seq mapping)

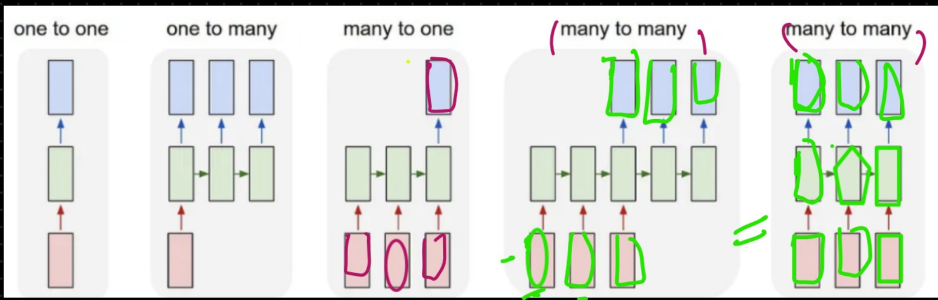


image classification

Image caption

(CNN+RNN)

Sentiment

(RNN, LSTM)

{language translation}, NER, POS

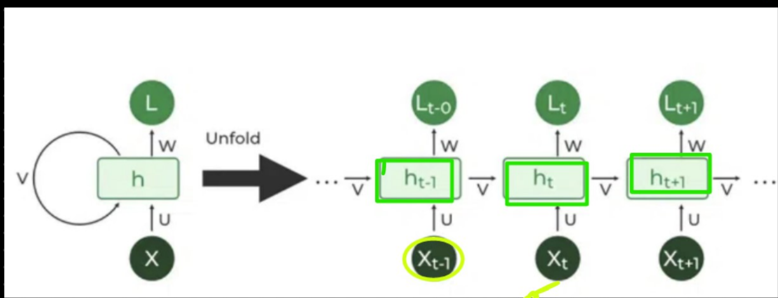
Same length

Diff length

I am sunny

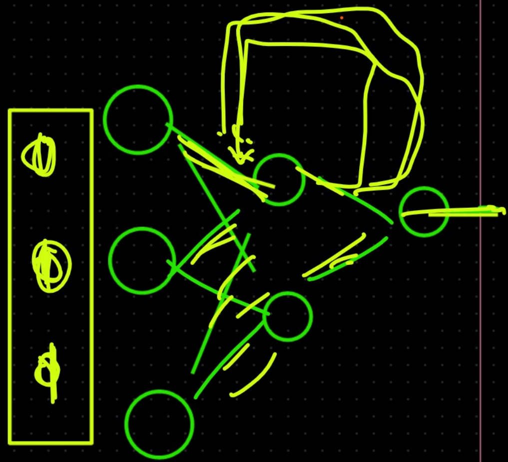
{ Switch off the fan

2) RNN



I am Sunny

100 010 001



Working ⇒ time stamp

Sequential Process

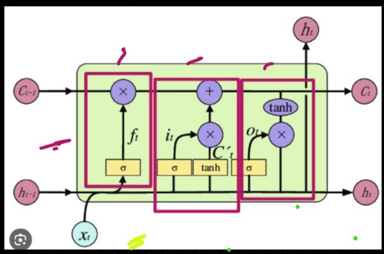
optimization

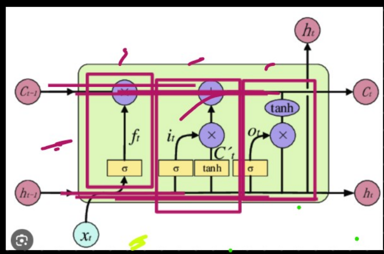
1) vanishing gradient
2) exploding " "

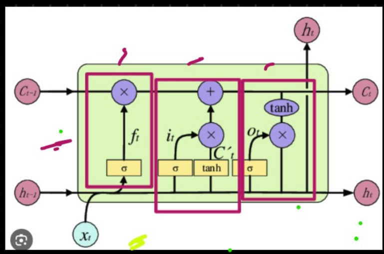
Advantage ⇒ Seq

Disadvantage ⇒ (Long context)

3 LSTM (Long short term memory)







t=1

t=2

t=3

Little longer sentence

Long - Short

Sunny, Sandeep

2000-2010

Short

Sandeep

Developer

mentor

2020

Sunny → Mentor → AI

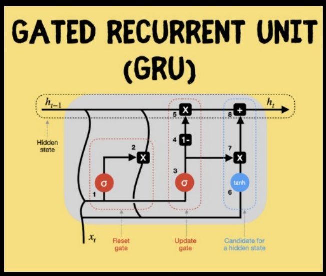
Who is the father:

RNN ✗

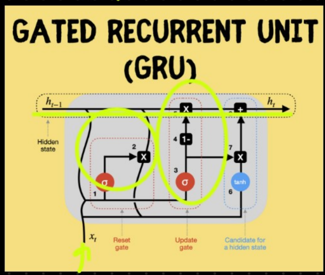
LSTM

4 GRU (Gated Recurrent Unit)

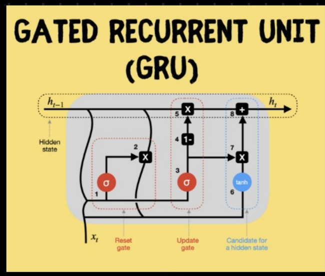
t=0



t=1



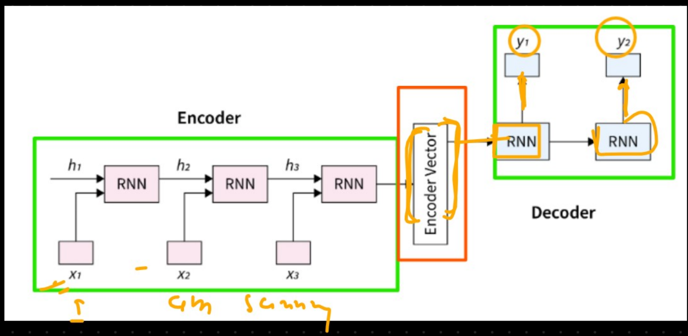
t=2



Feature	LSTM	GRU
Architecture Complexity	More complex with three gates (input, forget, output) and a separate memory cell.	Simpler with two gates (update, reset) and a combined hidden state/memory cell.
Memory Handling	Separate memory cell explicitly manages what information to keep or discard.	Combines hidden state and memory cell into a single state vector.
Gating Mechanism	Explicit forget, input, and output gates for controlling information flow.	Single update gate for combining past and new information, and a reset gate.
Parameter Count	Higher due to the additional parameters from the complex architecture.	Lower, as GRUs have fewer parameters, making them computationally more efficient.
Training Convergence	May require more time to converge during training due to the complexity.	Often converges faster during training, making them easier to train in some scenarios.

Choosing between LSTM and GRU depends on the specific task, dataset, and computational resources. LSTMs might be more effective in capturing longer dependencies due to their more intricate structure, but GRUs often require fewer parameters and may be computationally more efficient in certain scenarios. It's common practice to experiment with both architectures to determine which one performs better for a particular use case.

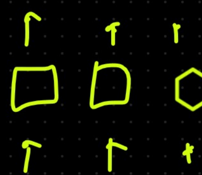
6 Encoder & Decoder



1. Encoder
2. decoder
3. RNN | LSTM | GRU
   → Birdirectional
4. Encoder → Content vector → Decoder

Why?

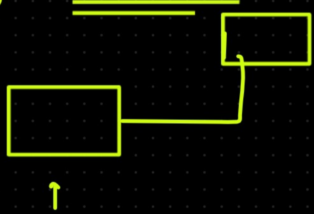
RNN | LSTM / GRU   many to many



fail ⇒

Differeur length

Asynchronized Problem ⇒ translation



Content

translation (E&D)

Content vector

perfect → Error

30-35

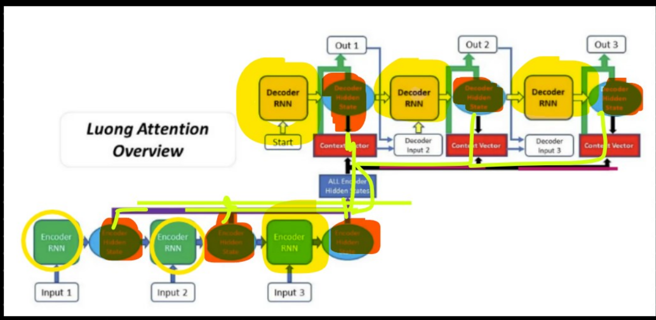
{ पंखा बंद कर दो }
switch off the fan

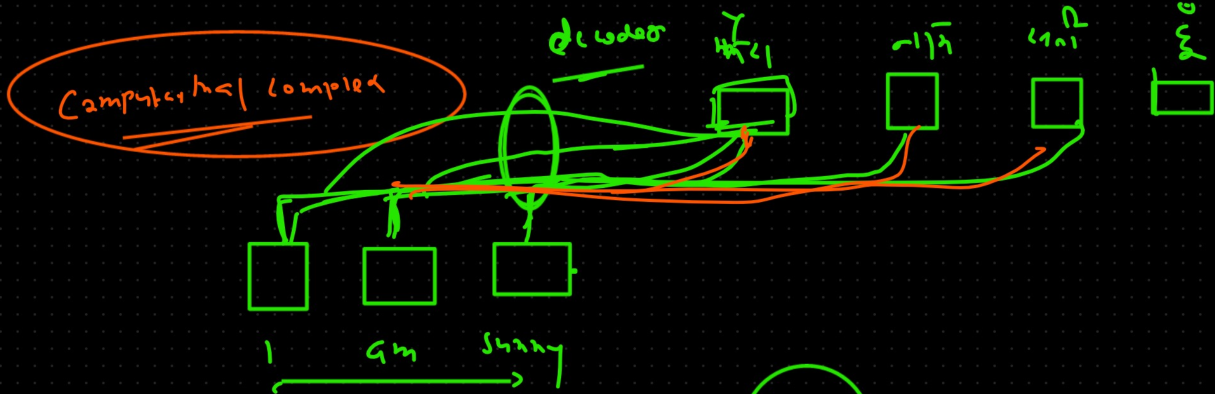
Attention

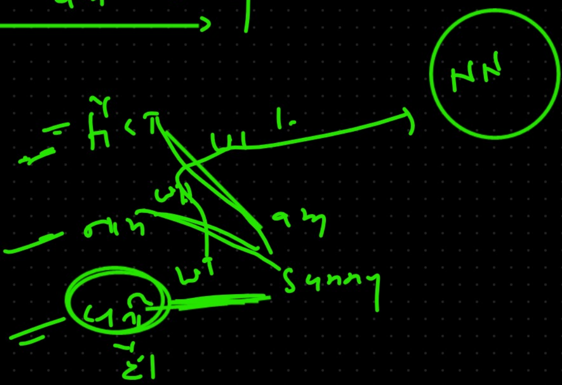
Problem

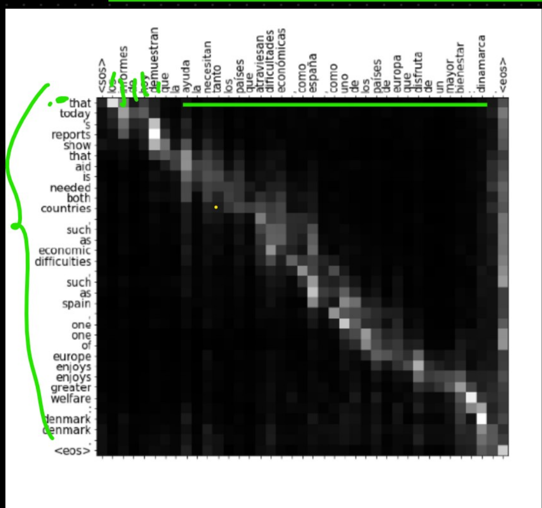
Switch
off
th
fan

7 Encoder & Decoder with attention









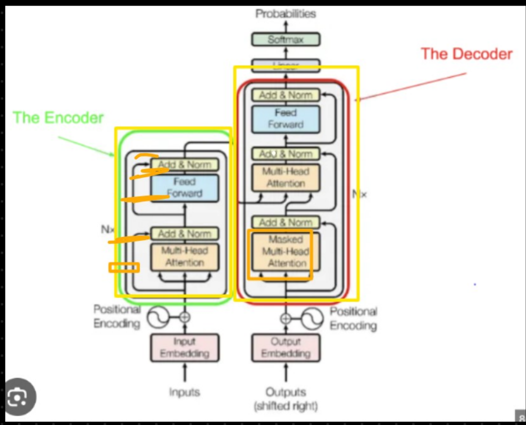
Encoder – Decoder Attention

Base LSTM ⇒ Parallel Processing

transformer

8 Transformer (Attention all you need)

Milestone



1. Remove LSTM/RNN
2. Self attention → Multihead attention
3. Positional encoding (to sustain the seq)
4. Parallel processing
5. Normalization
6. Artifical Neural Network
7. Skip Connection

Language modelling   Image generation   Computer

9 ULMFIT =>

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al.,

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs

give language

fine tuning
transfer learning

understood
→ finetune

Language modelling

1 text classification
2 ~~generation~~ → next word word
3 summarization
4 ~~translation~~

train

text → RNN/LS → pred

supervised

I am sunny saving .... →

I	am
am	sunny
sunny	?

training

⇓
next word prediction

Language modelling (supervised)

→ specific task / Limited

Discriminative

Generative => Data general

1 universal language modelling
2 fine tuning for the text classification

Model → Transformer

↑↑↑

← GPT

huge
1B

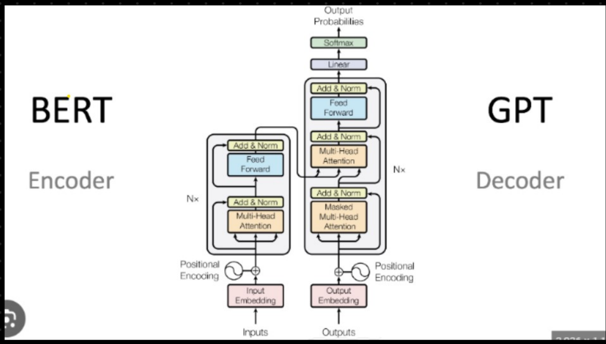
unsupervised language model

1 statical pattern → Dist
2 semantic word
3 contextual relationship

word - word

they were able to teach to the model

10 **BERT**



BERT
Encoder
GPT
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
N×
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs

11 GPT ⇒ Generative Pre training

GPT1 :Improving Language Understanding by Generative Pre-Training

Generative Pre-training:

The term "generative" refers to the model's capability to generate coherent and contextually relevant text. "Pre-training" indicates that the model is initially trained on a large corpus of text data in an unsupervised manner before being fine-tuned for specific tasks.
During the pretraining phase, the model learns to predict the next word in a sequence, capturing language patterns and semantics.

Key Objectives:

The paper outlines the methodology of training a transformer-based language model on a diverse range of tasks without task-specific labelled data.
The generative pretraining approach allows the model to acquire a broad understanding of language, enabling it to perform well on various downstream tasks.

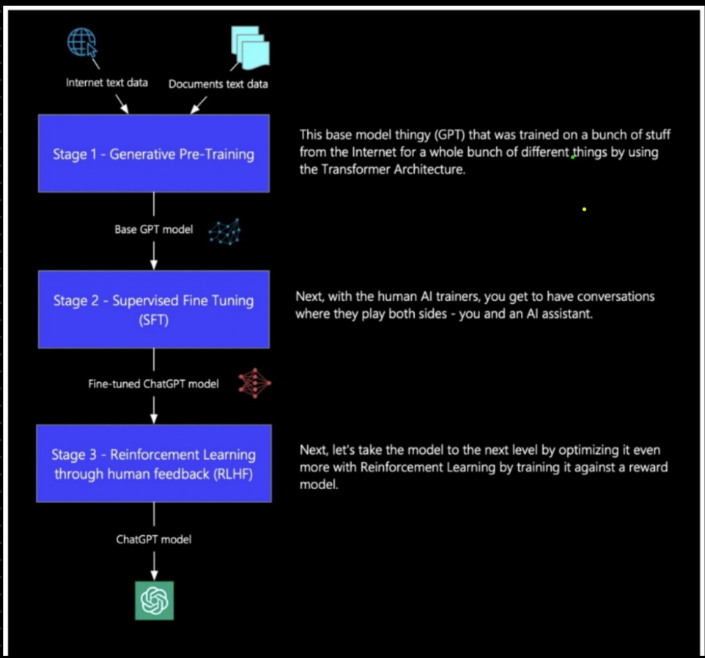
GPT2: language models are unsupervised multitask leaner

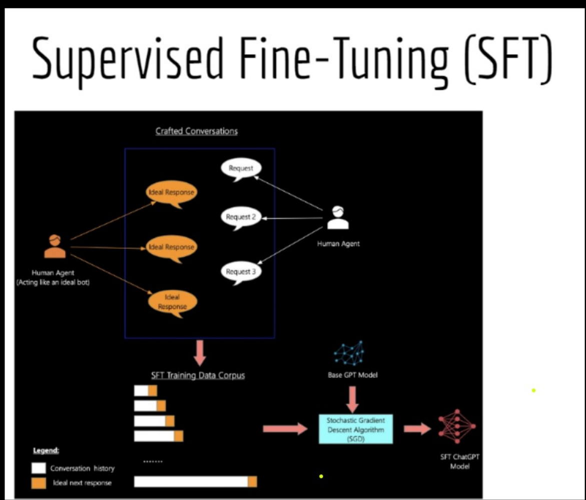
In the context of the GPT-2 (Generative Pre-trained Transformer 2) paper, the term "unsupervised multitask learner" refers to the model's ability to perform a variety of natural language processing (NLP) tasks without task-specific supervision during the pretraining phase. GPT-2 is designed as a large-scale language model that is pretrained on a diverse corpus of text data without explicit annotations for specific tasks. given to it at runtime

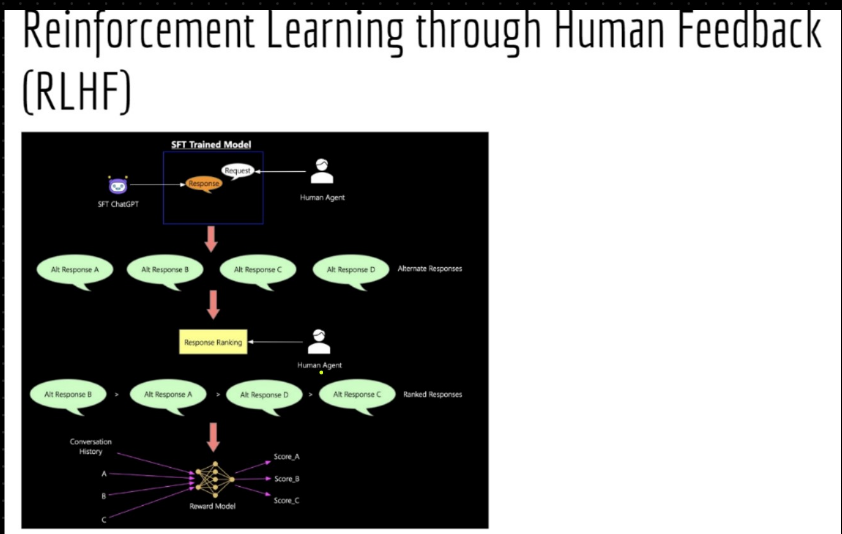
GPT3: language model are fewshot learner

In the context of the GPT-3 research paper, the term "few-shot learner" refers to the model's ability to perform a task or answer questions with only a few examples or prompts provided during inference. GPT-3 is known for its remarkable ability to generalise from a small number of examples or demonstrations given to it at runtime.

(12) Training of chatgpt







13 Conclusion