FourthBrain

MLE Program, Cohort 11 (MLE11)

Week 11: Transformers, Encoder/Decoder, BERT, GPT-3

Last Week!

Concepts

- Natural Language Benchmarks
- Dealing with Text
- Named Entity Recognition
- Bag of Words, Term Frequency Inverse Document Frequency
- Tokenization & Word Embeddings

🤖 This Week!

Concepts

- Encoder and Decoder Networks
- Bidirectional Encoder Representations from Transformers (BERT)
- General Pre-Trained Transformers (GPT-3)
- Fine-Tuning of Pre-Trained Transformers

Hands on

- Fine Tuning a GPT question answering chatbot

What questions do you have?

Encoder and Decoder Networks

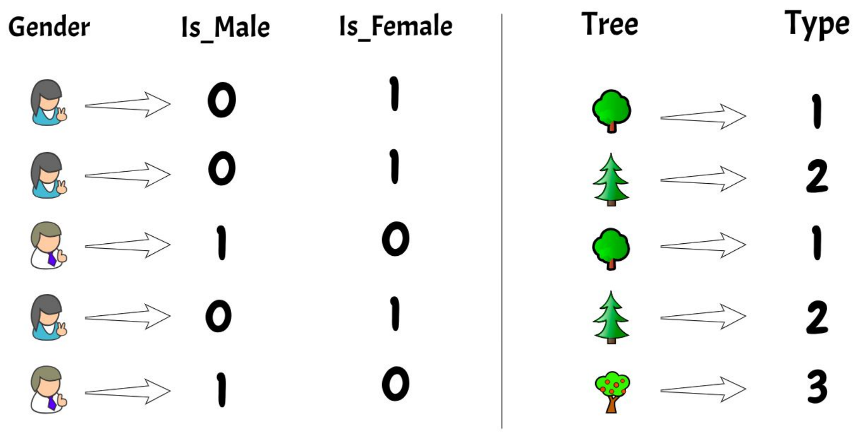
What is Encoding?

- Encoding is to convert the data into a required format
- It is sometimes also known as serialization
- For example:
    - Convert a word into an embedding
    - Convert a sentence into a 2D vector
    - Convert text into a drawing

Note: Encoding is not Encrypting

Encoding | Applications

- Program Compiling
- Program Execution
- Data Transmission
- Data Storage
- Data Processing
- File Conversion
- Compression/Decompression
- Machine Learning

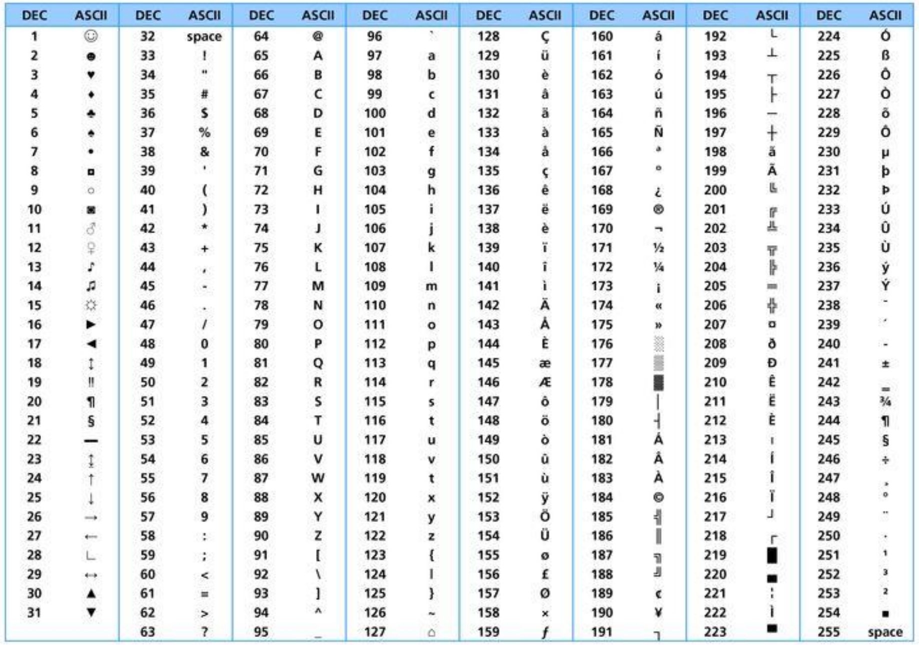


Encoding

- Encoding is the usage of a code to change the original data into a form that can be used by an external process
- ASCII – American Standard Code for Information Interchange
- ASCII is the most commonly used encoding scheme for files that contain text
- Encoding is also used to reduce the size of audio and video files

ASCII

- ASCII contains:
  - Printable characters
  - Non-printable characters
  - Uppercase letters
  - Lowercase letters
  - Symbols
  - Punctuation marks
  - Numbers

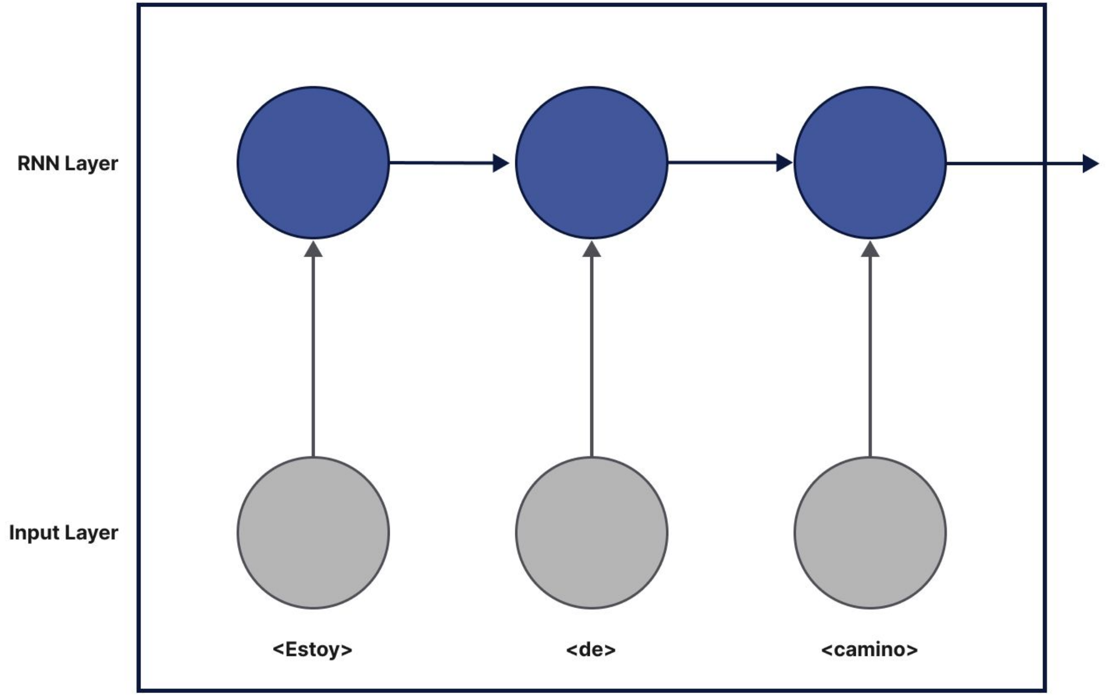


DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII
1	☺	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
2	☻	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
3	♥	34	"	66	B	98	b	130	è	162	ó	194	┬	226	Ô
4	♦	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
5	♣	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
6	♠	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
7	•	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
8	◘	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
9	○	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
10	◙	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
11	♂	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
12	♀	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
13	♪	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
14	♫	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
15	☼	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
16	►	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
17	◄	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
18	↕	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
19	‼	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
20	¶	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
21	§	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
22	▬	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
23	↨	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
24	↑	55	7	87	W	119	t	151	ù	183	À	215	Î	247	¸
25	↓	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
26	→	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
27	←	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
28	∟	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
29	↔	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
30	▲	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
31	▼	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
		63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	space

Encoding | ML

- Encoders are usually built by stacking Recurrent Neural Networks RNNs
- This type of layer is used because its structure allows the model to understand the context and temporal dependencies of the sequences
- The output of the encoder, the hidden state, is the state of the last RNN timestep

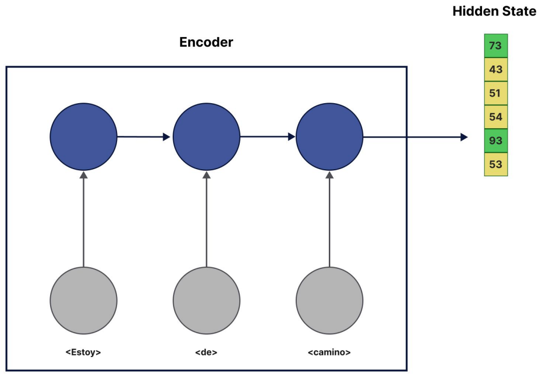
Encoder | ML



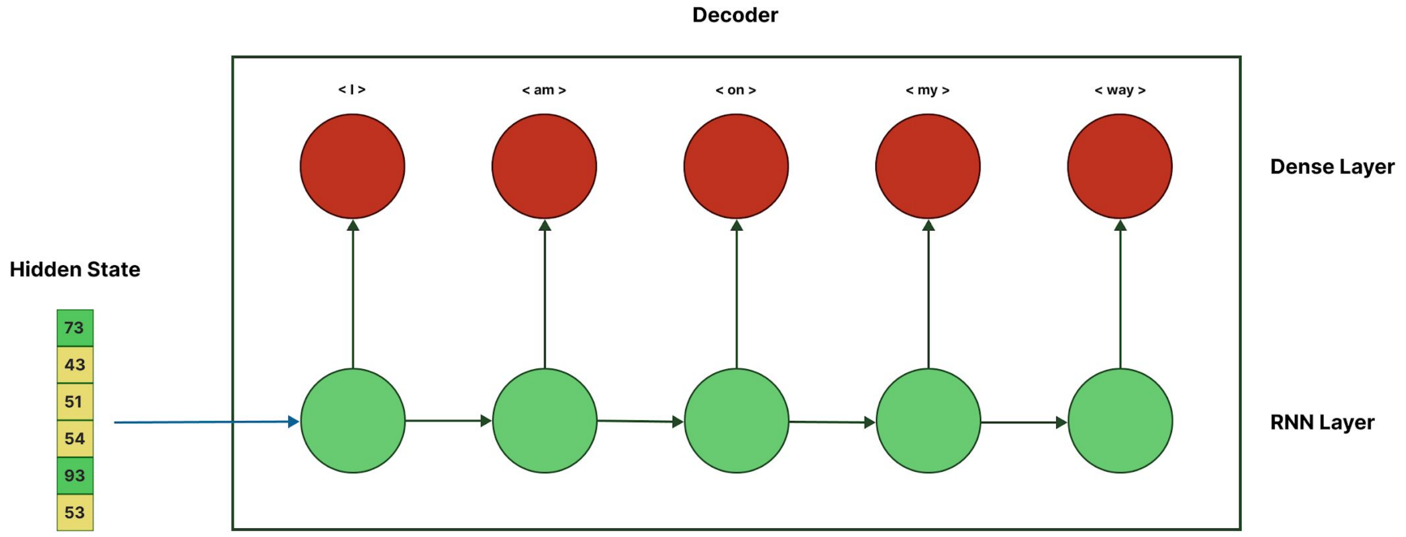
Hidden State | Sketch

- The output of the encoder, a two-dimensional vector that encapsulates the whole meaning of the input sentence

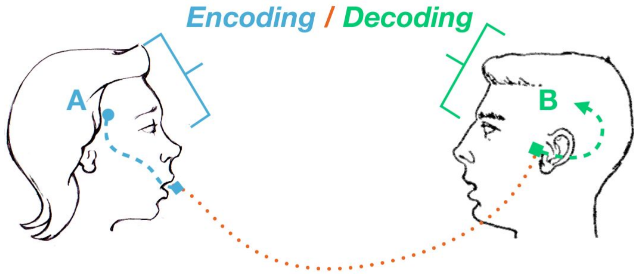
- The length of the vector depends on the number of cells in the RNN



Decoder


Decoder
< I >
< am >
< on >
< my >
< way >
Dense Layer
Hidden State
73
43
51
54
93
53
RNN Layer

Decoder

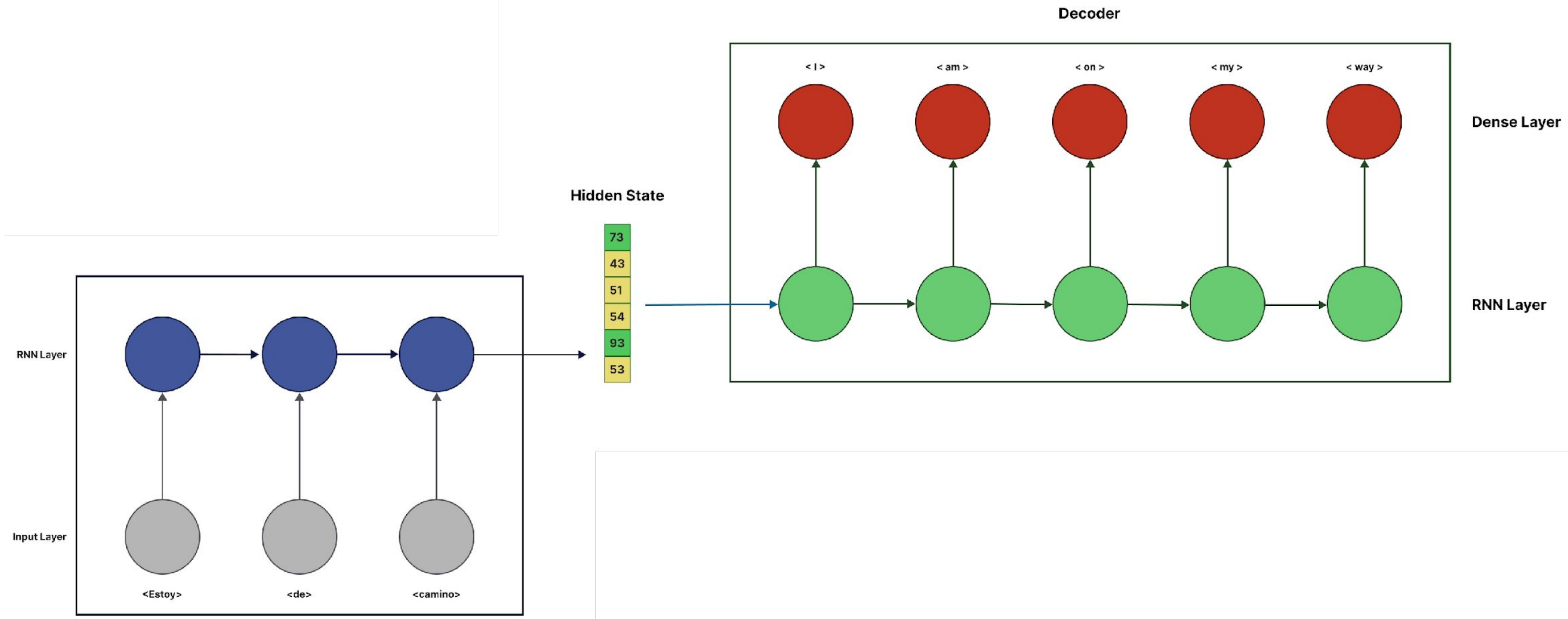


- A decoder converts the coded message into an intelligible language
- It has a multiple-input, multiple-output logic
- In a human conversation, the encoder is the person who develops and sends the message, while decoding is the process of turning this communication into thoughts.

Decoder

- In a machine learning model, the role of the decoder is to convert the two-dimensional vector into the output sequence.
- Taking a machine translation problem, with a Spanish sentence as input, the decoder will be outputting the English sentence translated.
- It is also built with RNN layers and a dense layer to predict the English word.

Encoder-Decoder | Putting it all together



Breakout

15 min
(3-4 per room)

What applications of Encoder-Decoder networks can you think of / have you used ?

Designate *one person to share* from your breakout room

Encoder Decoder | Image Captioning

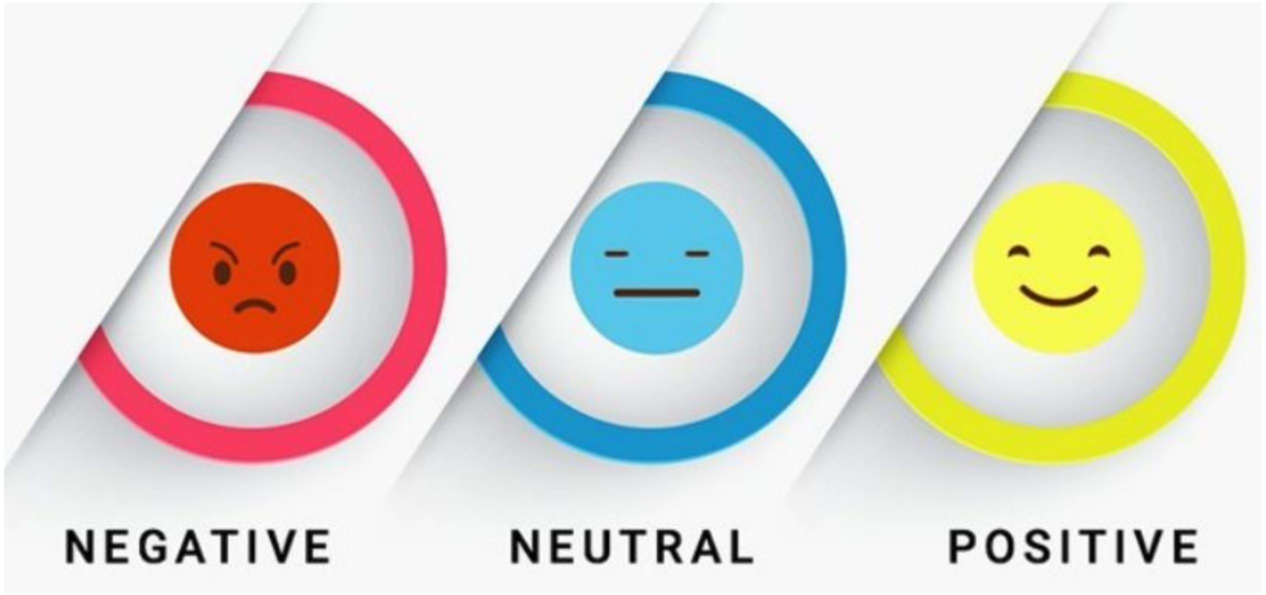
A young boy is playing basketball.	Two dogs play in the grass.	A dog swims in the water.	A little girl in a pink shirt is swinging.
A group of people walking down a street.	A group of women dressed in formal attire.	Two children play in the water.	A dog jumps over a hurdle.

Encoder Decoder | Image Captioning

- Encoder decoder models allow a machine learning model to generate a sentence describing an image
- It receives the image as the input and outputs a sequence of words
- It also works with videos

A young boy is playing basketball.	Two dogs play in the grass.	A dog swims in the water.	A little girl in a pink shirt is swinging.
A group of people walking down a street.	A group of women dressed in formal attire.	Two children play in the water.	A dog jumps over a hurdle.

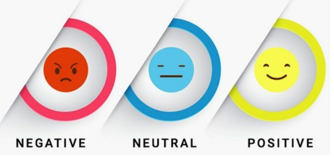
Encoder Decoder | Sentiment Analysis



Totally dissatisfied with the service.
Worst customer care ever.

Good Job but I will expect a lot more in the future

Brilliant effort guys!
Loved your work.



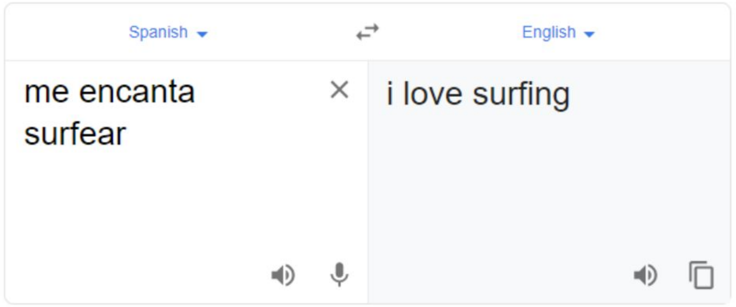
Encoder Decoder | Sentiment Analysis

- These models understand the meaning and emotions of the input sentence and output a sentiment score.
- It is usually rated between -1 (negative) and 1 (positive) where 0 is neutral
- It is mostly used in call centers to analyze the evolution of the client's emotions and their reactions to certain keywords or the company discount

Encoder-Decoder | Translation

Steps:

- Reads the input sentence
- Understands the message
- Generalize the concepts
- Translates it into a second language



me encanta surfear

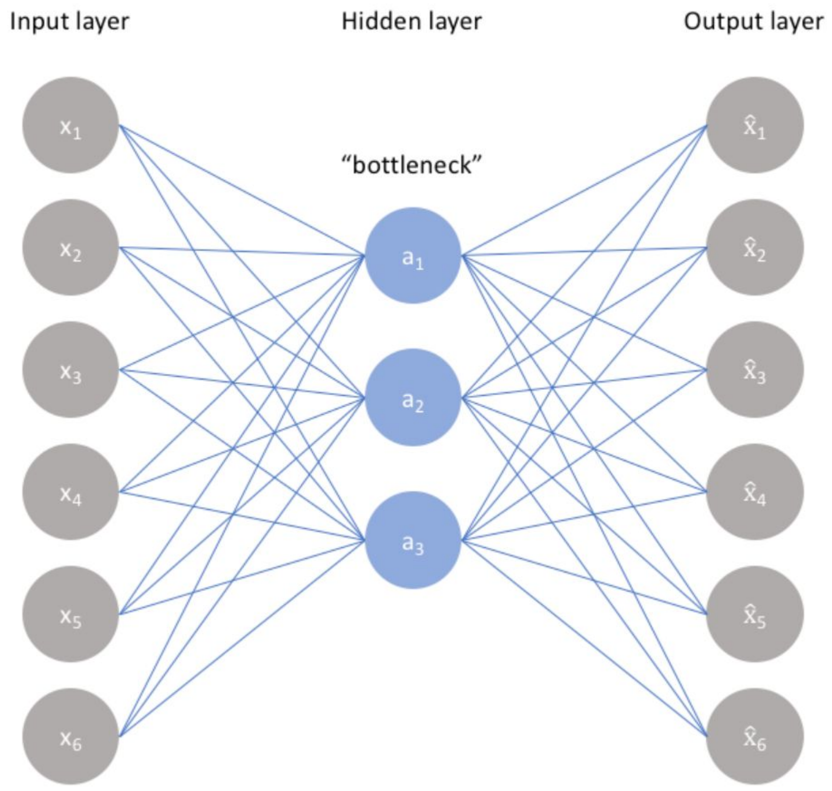
i love surfing

- As an example, Google Translate is built upon an encoder-decoder structure.

Autoencoders

- Autoencoders are an unsupervised technique
- It leverages neural networks for the task of representation learning
- It usually imposes a bottleneck in the network which forces a compressed knowledge representation of the original input

Autoencoders



Autoencoders

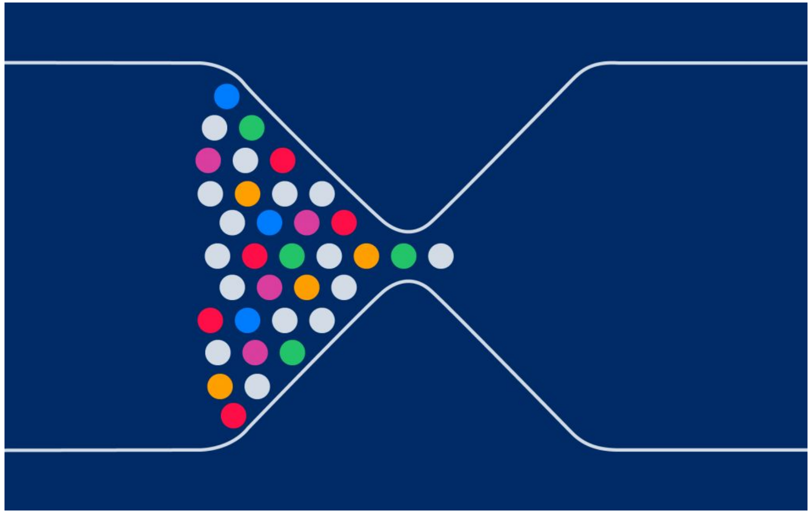
- It best works if some sort of structure exists in the data
  - Correlations between the input features

- This structure can be learned and consequently leveraged when forcing the input through the network's bottleneck.

- This compression will become very hard if the input features were independent

Autoencoders | why use a bottleneck?

- The bottleneck is a key attribute of the network design



Autoencoders | why use a bottleneck?

- The bottleneck is a key attribute of the network design
- Without the presence of an information bottleneck, the network could easily learn to simply memorize the input values by passing these values along through the network
- A Bottleneck constrains the amount of information that can traverse the full network, forcing a learned compression of the input data

Autoencoders | Tradeoff

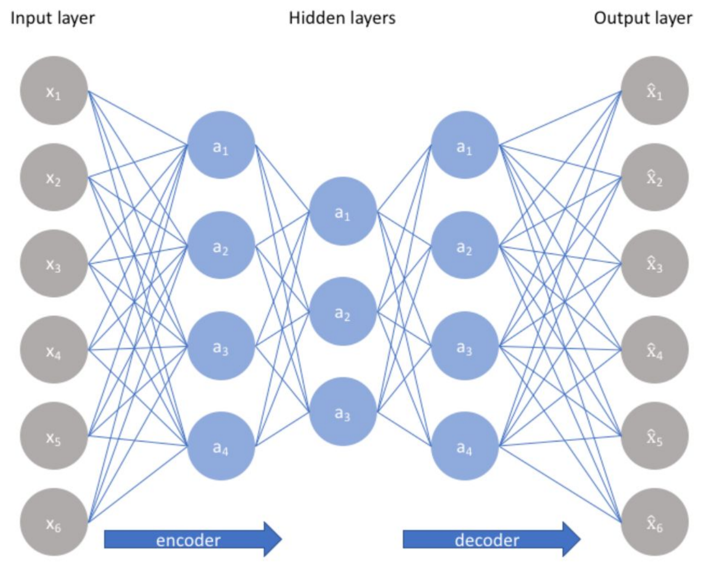
- The ideal Autoencoder model balances the following:
  - Sensitive to the inputs enough to accurately build a reconstruction
  - Insensitive enough to the inputs that the model doesn’t simply memorize or overfit the training data

- This trade-off forces the model to maintain only the variations in the data required to reconstruct the input without holding on to redundancies within the input

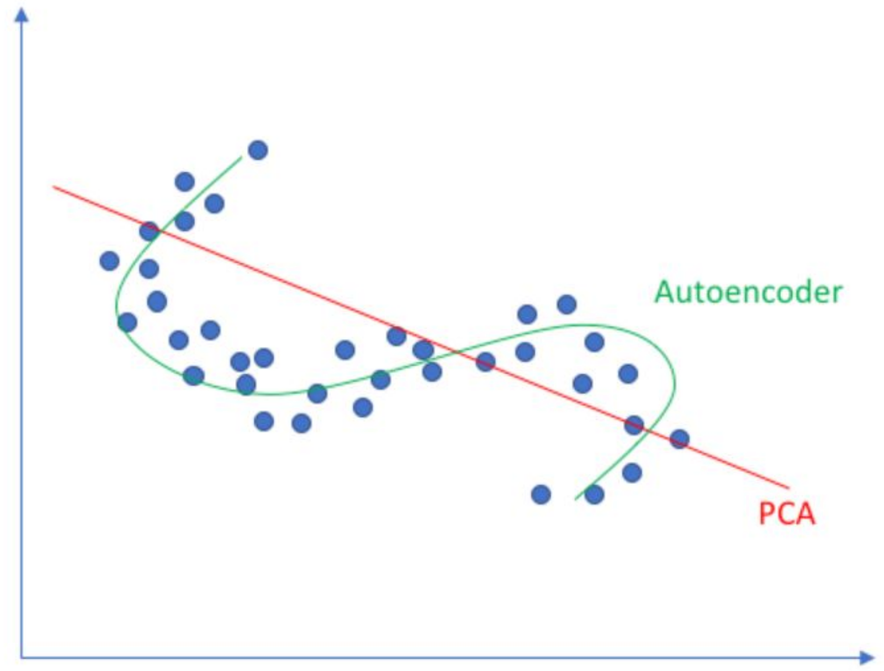
Autoencoders | dimensionality reduction

- By penalizing the network according to the reconstruction error, the model can learn the most important attributes of the input data and how to best reconstruct the original input from an "encoded" state

- This encoding will learn and describe latent attributes of the input data

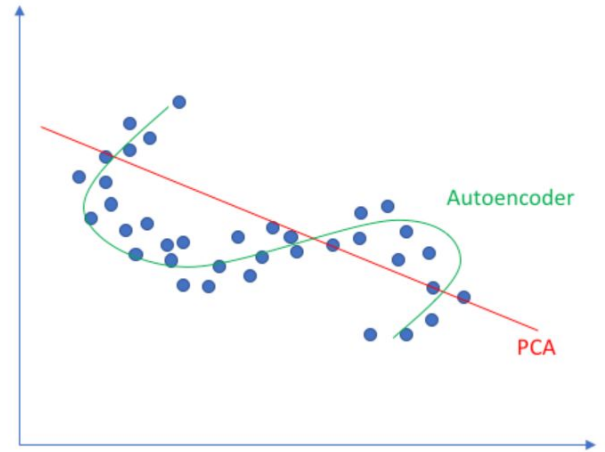


Autoencoder vs PCA



Autoencoder vs PCA

- Neural Networks (NN) are capable of learning nonlinear relationships
- This makes NN a much more powerful generalization than PCA
- PCA attempts to discover a lower dimensional hyperplane which describes the original data



What's an Auto-Encoder?
IBM Cloud
CODE

Foundation Models

About

- Stanford HAI popularized the name
- What are the foundation models?
  - ‘models trained on broad data (generally using self-supervision at scale) that can be adapted to a wide range of downstream tasks’ (source)
- Application of foundation models
  - through self-supervised learning and transfer learning

The Application Layer of AI

DALL·E 2

- Relationship = images <> text
- Uses “diffusion,”
  - Starts with a pattern of random dots and gradually alters that pattern towards an image when it recognizes specific aspects of that image.
- https://openai.com/dall-e-2/
- https://stability.ai/blog/stable-diffusion-public-release
- https://midjourney.com/home

VALL-E

Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers

- Relationship = speech <> text
- Can be used to synthesize high-quality personalized speech with only a 3-second enrolled recording of an unseen speaker as an acoustic prompt
- https://valle-demo.github.io/

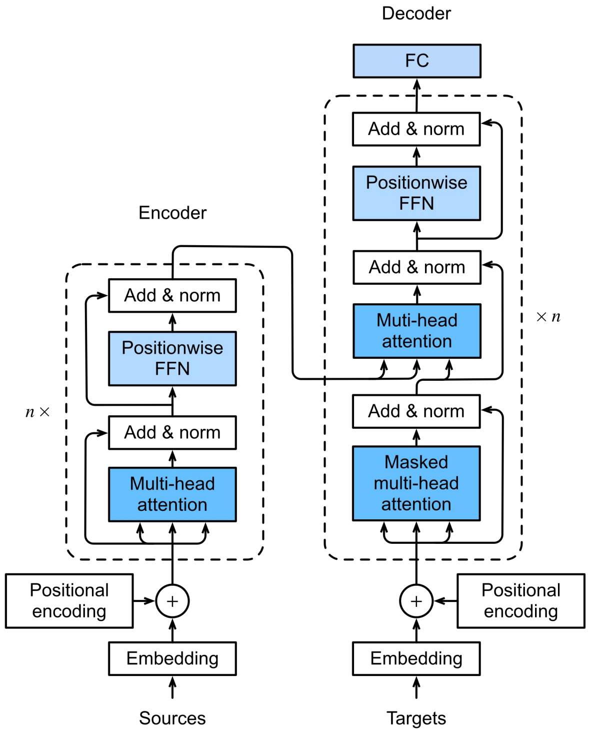
Large Language Models (LLMs)

- transformers are neural networks that understand context from the sequential data
  - Foundation models trained on MASSIVE data (aka open source internet - the whole internet, the whole Wikipedia, YouTube videos, publicly available images from social-media etc)
- LLMs billions parameters models that can write poems in any language and have a conversation
- Examples: BERT, GPT-3, Bloom, LaMDA, Whisper, NLLB-200

Ameca robot

BERT

BERT | Transformers



BERT

- BERT – Bidirectional encoder representations of transformers
- It is a open-source state-of-the-art language model for Natural Language Processing
- Published in 2018 by researchers at Google AI Language
- Designed to pre-train deep bidirectional representations from unlabeled text

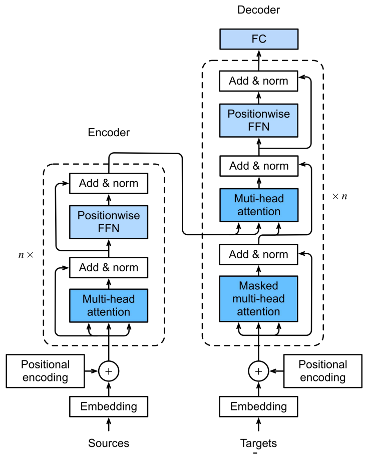
BERT

- It is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context.

- The BERT framework was pre-trained on Wikipedia (~2.5B words) and Google's BooksCorpus (~800M words)
  - 64 TPUs trained BERT over the course of 4 days

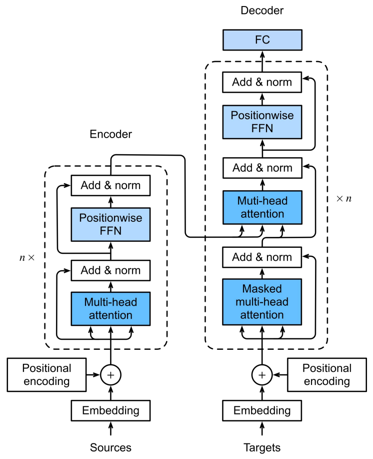
- BERT is different than historical language models – it is designed to read in both directions at once, hence the word "bidirectional"

BERT | Transformers



- BERT is based on Transformers
- A deep learning model in which every output element is connected to every input element
- The weightings between these elements are dynamically calculated based on their connection
- In NLP, this process is called “attention”

BERT | Transformers



- Transformers were first introduced by Google in 2017
    - Attention is All You Need Paper
- At the time of their introduction, language models primarily used RNN and CNN to handle NLP tasks
- Transformers are considered a significant improvement because they don't require data sequences to be processed in any fixed order, whereas RNNs and CNNs do.

BERT | pre-training tasks

- Using its bidirectional capability, BERT is pre-trained on two different but related NLP tasks:
  - Masked Language Modeling
  - Next Sentence Prediction

- The objective of the Masked Language Model (MLM) is to hide a word in a sentence and then have the program predict what word has been hidden (masked) based on the hidden word’s context

BERT | Functions

- Sequence-to-sequence-based language generation tasks:
    - Question Answering
    - Abstract Summarization
    - Sentence Prediction
    - Conversational Response Generation
- Natural Language Understanding tasks
    - Word Sense Disambiguation
    - Natural Language Inference
    - Sentiment Classification
    - Polysemy and Coreference resolution

BERT | Models

- patentBERT – fine-tuned to perform patent classification
- docBERT – fine-tuned for document classification
- bioBERT – pre-trained biomedical language representation model
- VideoBERT – a joint visual-linguistic model for processing unsupervised learning of an abundance of unlabeled data from Youtube
- SciBERT – a pre-trained BERT model for scientific text
- G-BERT – pre-trained model using medical codes
- TinyBERT – “smaller” BERT to improve efficiency (by Huawei)
- DistillBERT – faster, smaller, cheaper version (by HyggingFace)

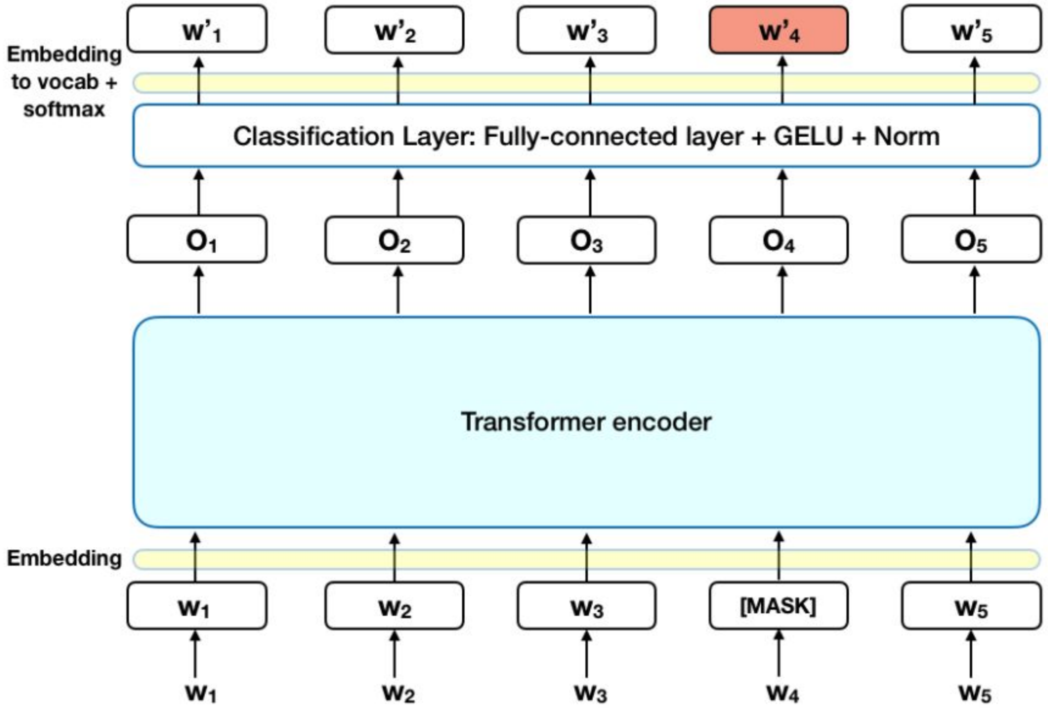
BERT | How it works?

- As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once

- When training language models, there is a challenge of defining a prediction goal

- Many models predict the next word in a sequence, which is directional, and BERT overcomes this as mentioned with MLM and NSP

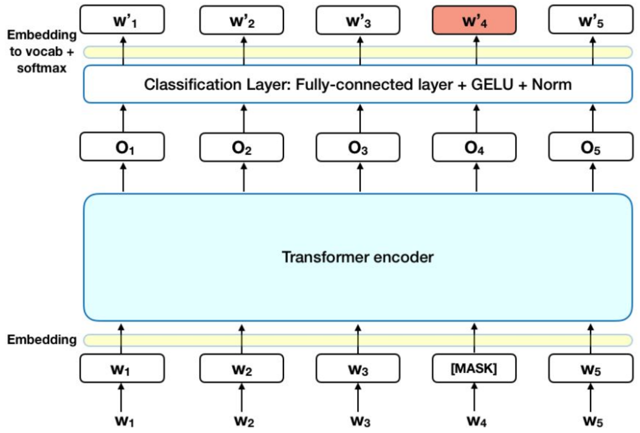
BERT | MLM


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

BERT | MLM

- Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a MASK token

- The model then attempts to predict the original value of the masked words, based on the context provided by the other non-masked words in the sequence

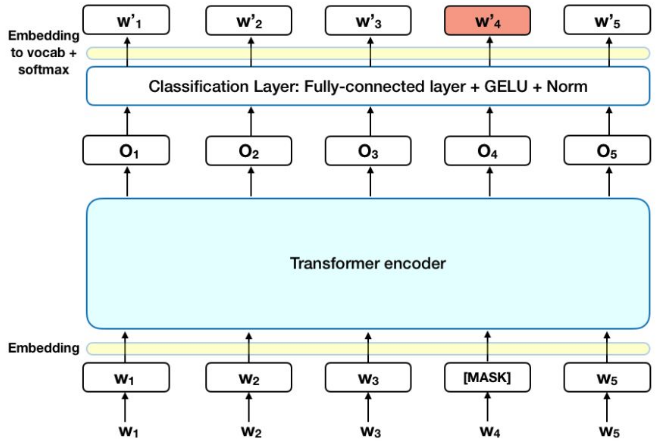


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

BERT | MLM

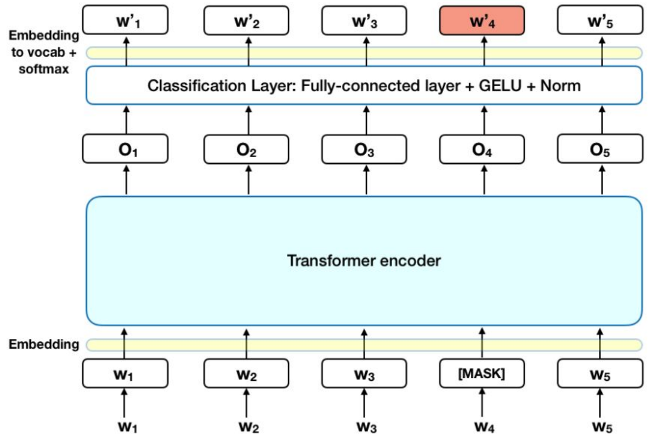
The prediction of the output words requires:

- Adding a classification layer on top of the encoder output
- Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension
- Calculating the probability of each word in the vocabulary with SoftMax


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

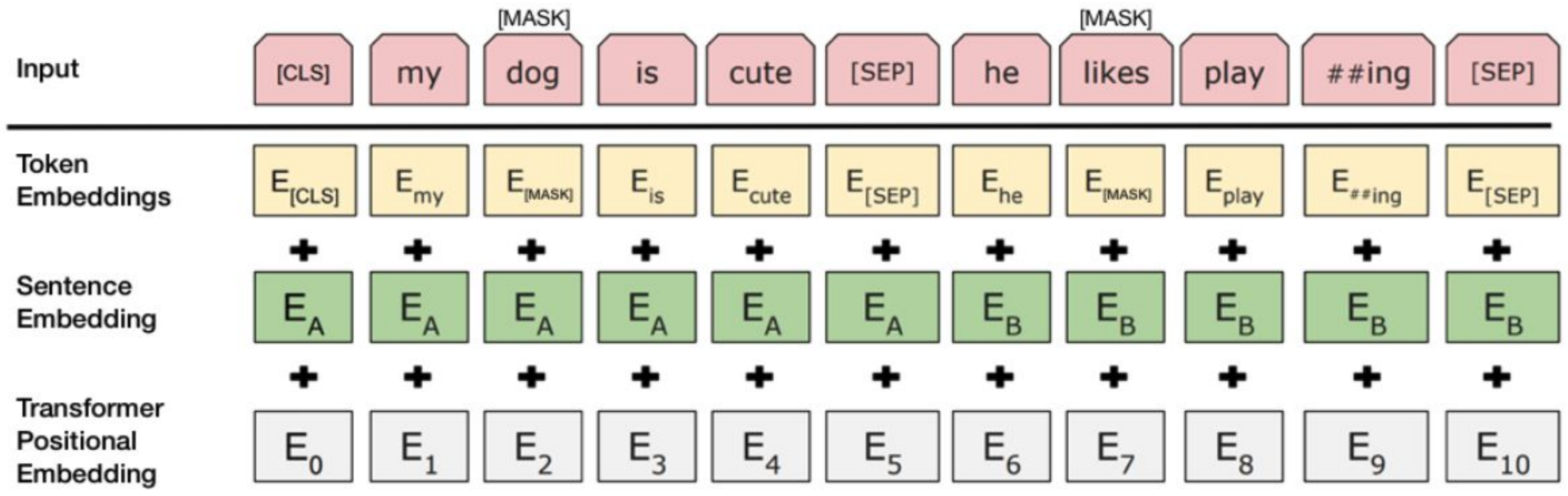
BERT | MLM

- The BERT loss function takes into consideration only the prediction of the masked values
- It ignores the prediction of the non-masked words
- The model converges slower than directional models
- This characteristic is offset by its increased context awareness

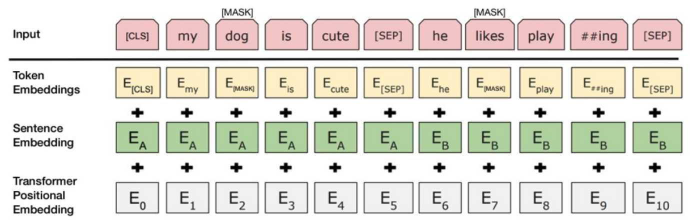


W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

BERT | NSP


Input
[MASK]
[MASK]
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$
E_{my}
$E_{[MASK]}$
E_{is}
E_{cute}
$E_{[SEP]}$
E_{he}
$E_{[MASK]}$
E_{play}
$E_{\#\#ing}$
$E_{[SEP]}$
Sentence Embedding
E_A
E_B
Transformer Positional Embedding
E_0
E_1
E_2
E_3
E_4
E_5
E_6
E_7
E_8
E_9
E_{10}

BERT | NSP

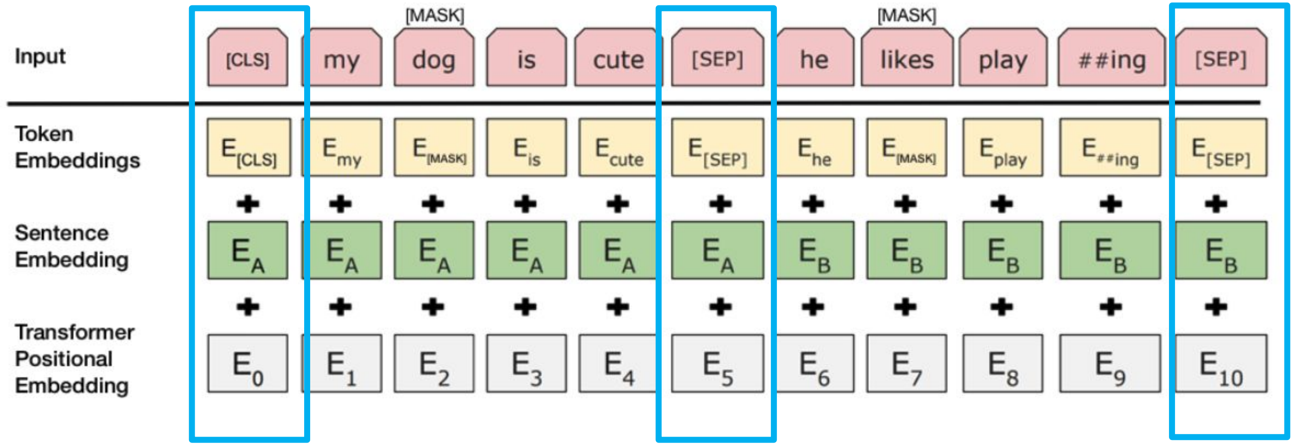

Input
[MASK]
[MASK]
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

- The model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document

- During training
  - 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document
  - 50% of the inputs are a random sentence from the corpus is chosen as the second sentence

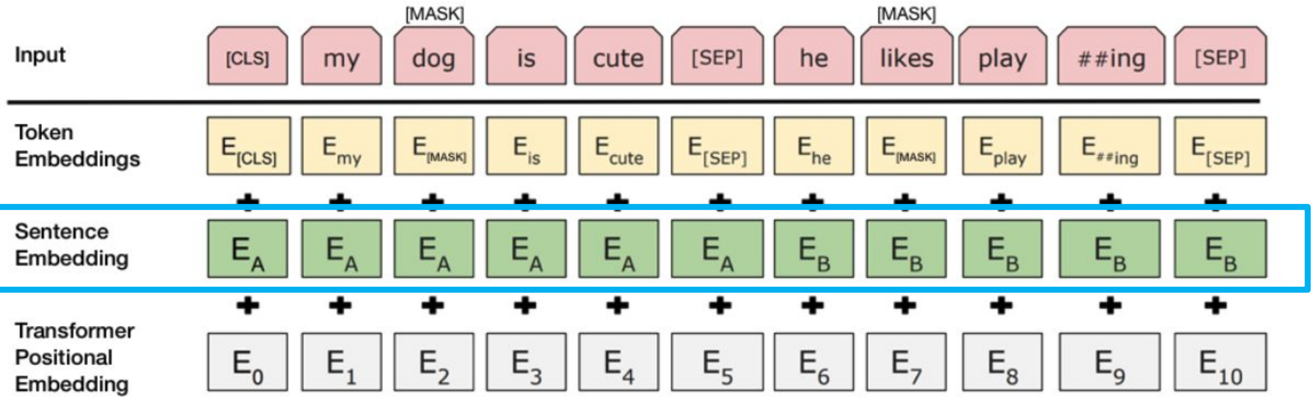
BERT | NSP – Step 1

- To help the model distinguish between the 2 sentences in training, the input is processed in the following way before entering the model:
    - A CLS token is inserted at the beginning of the first sentence and a SEP token is inserted at the end of each sentence


Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

BERT | NSP – Step 2

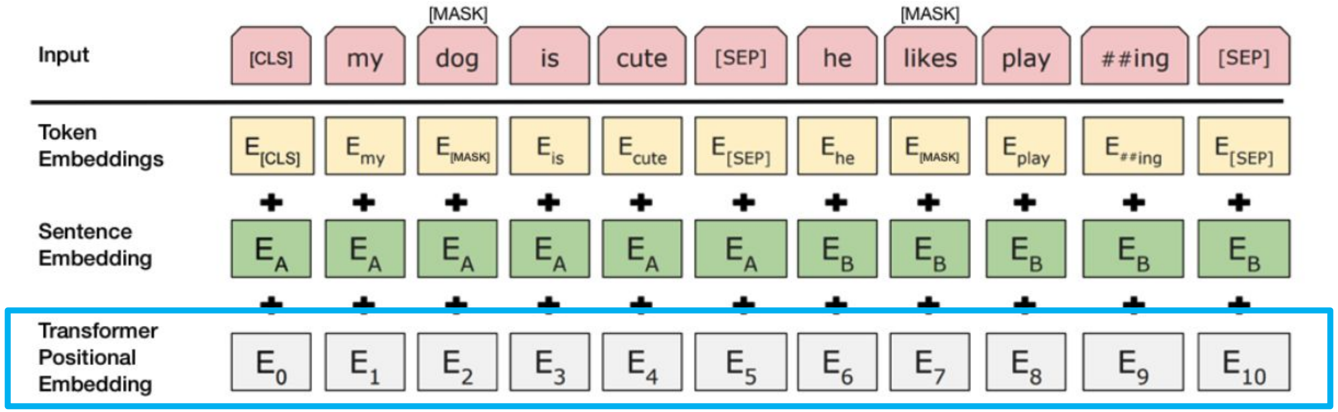
- A sentence embedding indicating sentence A or Sentence B is added to each token
- Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2



[MASK]
[MASK]
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

BERT | NSP – Step 3

- A positional embedding is added to each token to indicate its position in the sequence
- The concept and implementation of positional embedding are presented in the Transformer paper


Input
[MASK]
[MASK]
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

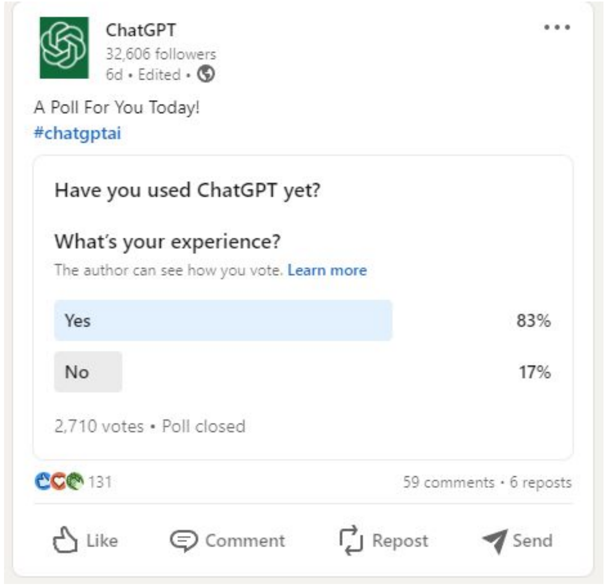
BERT | NSP

To predict if the second sentence is indeed connected to the first:

- The entire input sequence goes through the Transformer model
- The output of the CLS token is transformed into a 2x1 shaped vector, using a simple classification layer
- Calculating the probability of the next sequence with softmax

Generative Pre trained Transformer -3 (GPT-3)

What's the deal with ChatGPT?!


ChatGPT
32,606 followers
6d • Edited •

A Poll For You Today!
#chatgptai

Have you used ChatGPT yet?

What's your experience?
The author can see how you vote. Learn more

Yes	83%
No	17%

2,710 votes • Poll closed

131 · 59 comments • 6 reposts

- [19 Use Cases for ChatGPT prompts]
- [Use Cases for ChatGPT in marketing]

Time it took to reach **1 million users**:

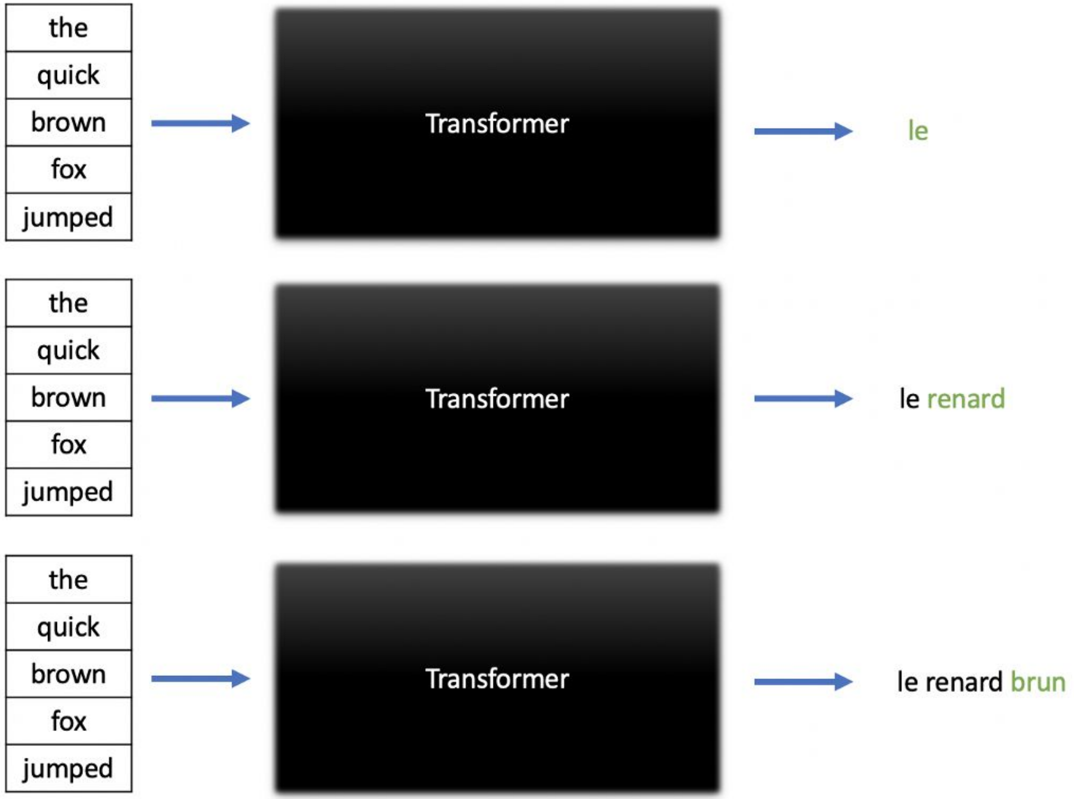
Netflix - 3.5 years
Airbnb - 2.5 years
Facebook - 10 months
Spotify - 5 months
Instagram - 2.5 months
iPhone - 74 days
ChatGPT - 5 days

ChatGPT is one of those rare moments in technology that **will reshape everything** going forward.

GPT-3

- GPT-3 – Generative Pre-Trained Transformer 3
- It is a language model created by OpenAI in San Francisco
- It’s a deep learning model that has 175 billion parameter
- The algorithm is capable of producing human-like text
- It was trained on large text datasets with hundreds of billions of words

GPT-3 | How it works



GPT-3 | How it works?

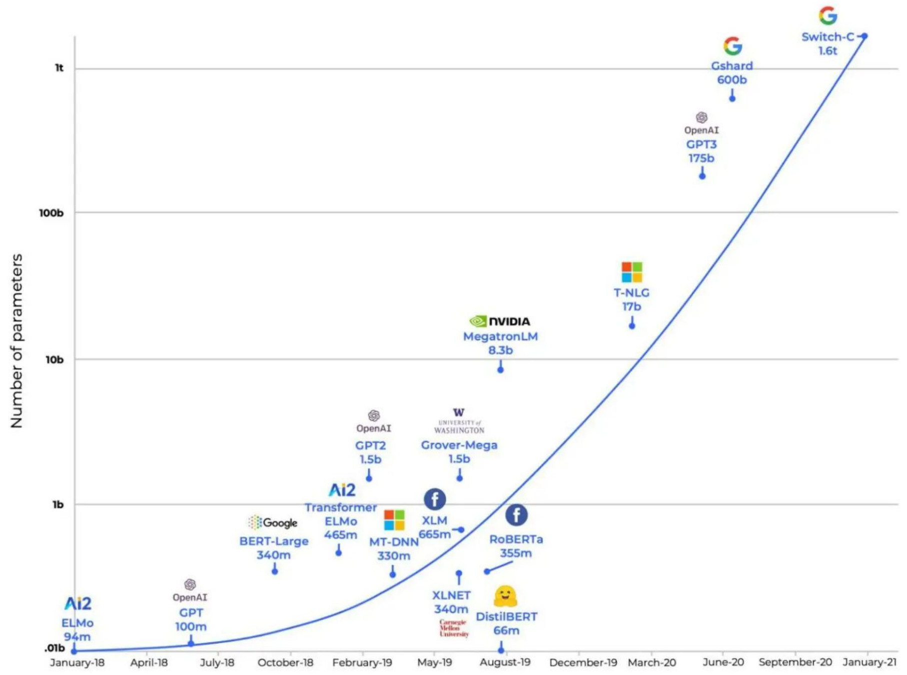
- GPT-3 is a transformer model
- Transformer models are sequence-to-sequence deep learning models that can produce a sequence of text given an input sequence
- This model is designed for text generation tasks such as:
  - Question Answering
  - Text Summarization
  - Machine Translation

GPT | Accessibility

- GPT-3 is not open-source
- OpenAI decided to make GPT-3 model available through commercial API
- OpenAI has a special program for academic researchers that want to use GPT-3
- GPT-2 is open-source and accessible through Hugging Face's transformers library

GPT-3 | Size

- GPT-3 is the 3rd generation of the GPT language models all created by OpenAI.
- The main difference that sets GPT-3 apart from previous models is its size



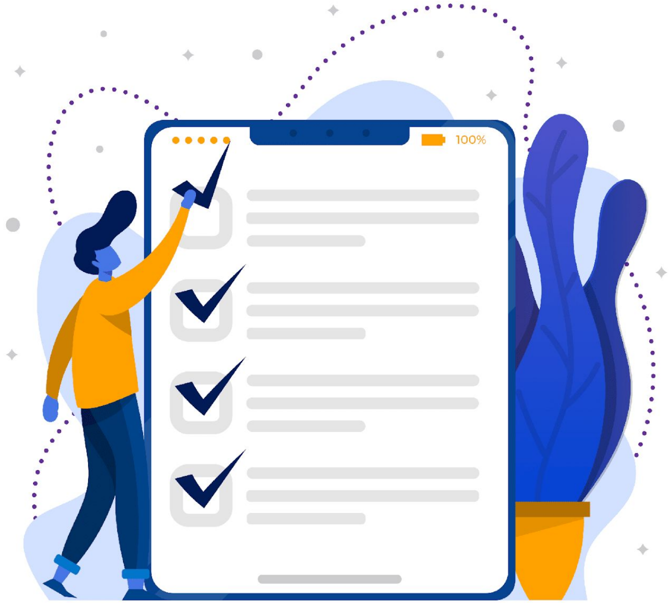
GPT-3 | Dataset

- GPT-3 was trained using a combination of large datasets:
    - Common Crawl
    - WebText2
    - Books1
    - Books2
    - Wikipedia Corpus

- The final dataset contained hundreds of billions of words to train GPT-3 to generate text in English and several other languages.

GPT-3 | Tasks

- Text Classification
- Sentiment Analysis
- Question Answering
- Text Generation
- Text Summarization
- Named-Entity Recognition
- Language Translation



GPT-3 | Power

- Entire startups have been created with GPT-3 because it could be thought of as a “general-purpose swiss army knife” for solving a wide variety of NLP problems

- We can think of it as a model that can perform reading comprehension and writing tasks at a near-human level

- After all, it has seen more text than any human will ever read in their lifetime

GPT-3 | Limitations

- Lacks long-term memory
  - The model does not learn anything from long-term interactions like humans
- Lack of interpretability
  - This is a problem that affects extremely large and complex models
  - GPT-3 is so large that it is so difficult to interpret or explain the output that it produces
- Slow inference time
  - Because GPT-3 is so large, it takes more time for the model to produce predictions

GPT-3 | Limitations

- Limited input size
  - Transformers have a fixed maximum input size
  - This means that prompts that GPT-3 can deal with cannot be longer than a few sentences
- Suffers from bias
  - All models are only as good as the data that was used to train them, including GPT-3

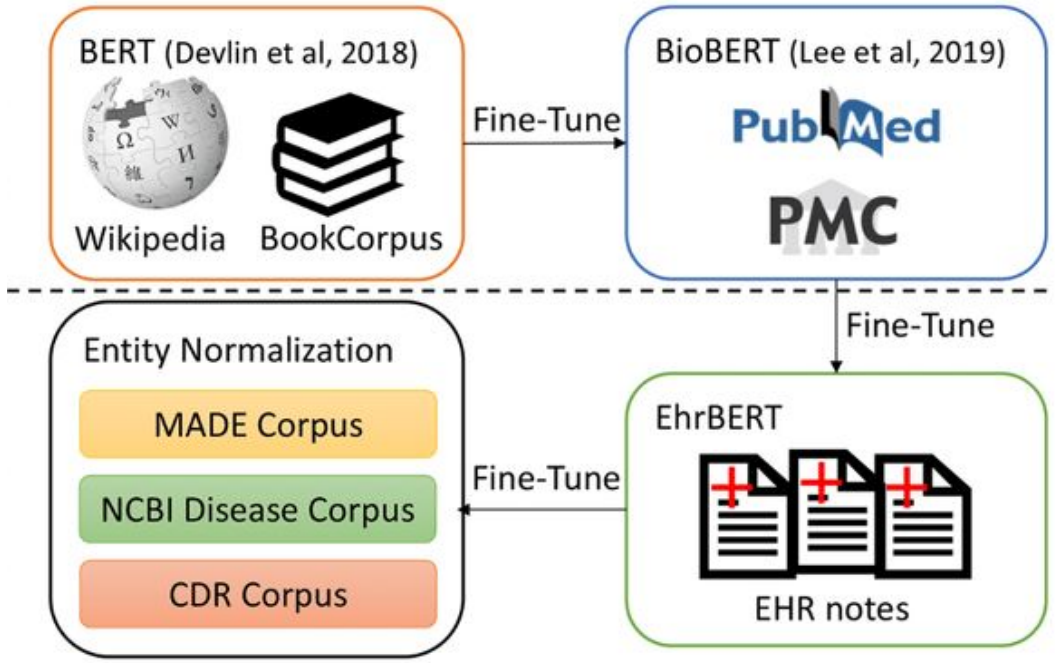
While being so powerful, these limitations stop GPT-3 from being an example of Artificial General Intelligence (AGN)

Fine-tuning of pre-trained Transformers

Fine-tuning Transformers

- Transformers have outperformed recurrent neural networks (RNN) in natural language generation
- This comes with a significant computational cost
- The attention mechanism complexity scales quadratically with the sequence length
- Efficient transformers variants are receiving increasing interest recently

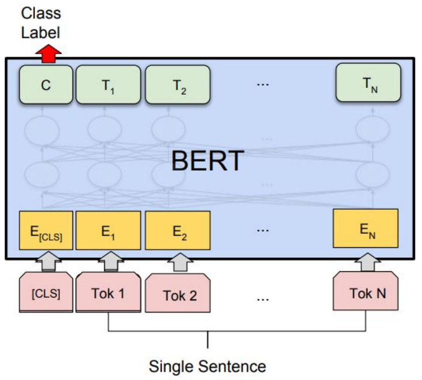
Fine-tuning Transformers | BERT



Fine-tuning Transformers | BERT

- BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model

- Classification tasks such as sentiment analysis are done similarly to Next Sentence Classification, by adding a classification layer on top of the Transformer output of the CLS token



Fine-tuning Transformers | BERT

- BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model

- Question answering tasks (i.e. SQuAD v1.1) software receive a question regarding a text sequence and is required to mark the answer in the sequence
- Using BERT, a Q&A model can be trained by learning 2 extra vectors that mark the beginning and the end of the answer

Fine-tuning Transformers | BERT

- BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model

- Named Entity Recognition (NER) software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc.) that appear in the text
- Using BERT, a NER model can be trained by feeding the output vector of each token into a classification layer that predicts the NER label

Fine-tuning Transformers | BERT

- In the fine-tuning training, most hyper-parameters stay the same as in BERT training
- The BERT paper gives specific guidance (Section 3.5) on the hyper-parameters that require tuning
- The BERT team has used this technique to achieve state-of-the-art results on a wide variety of challenging natural language tasks.

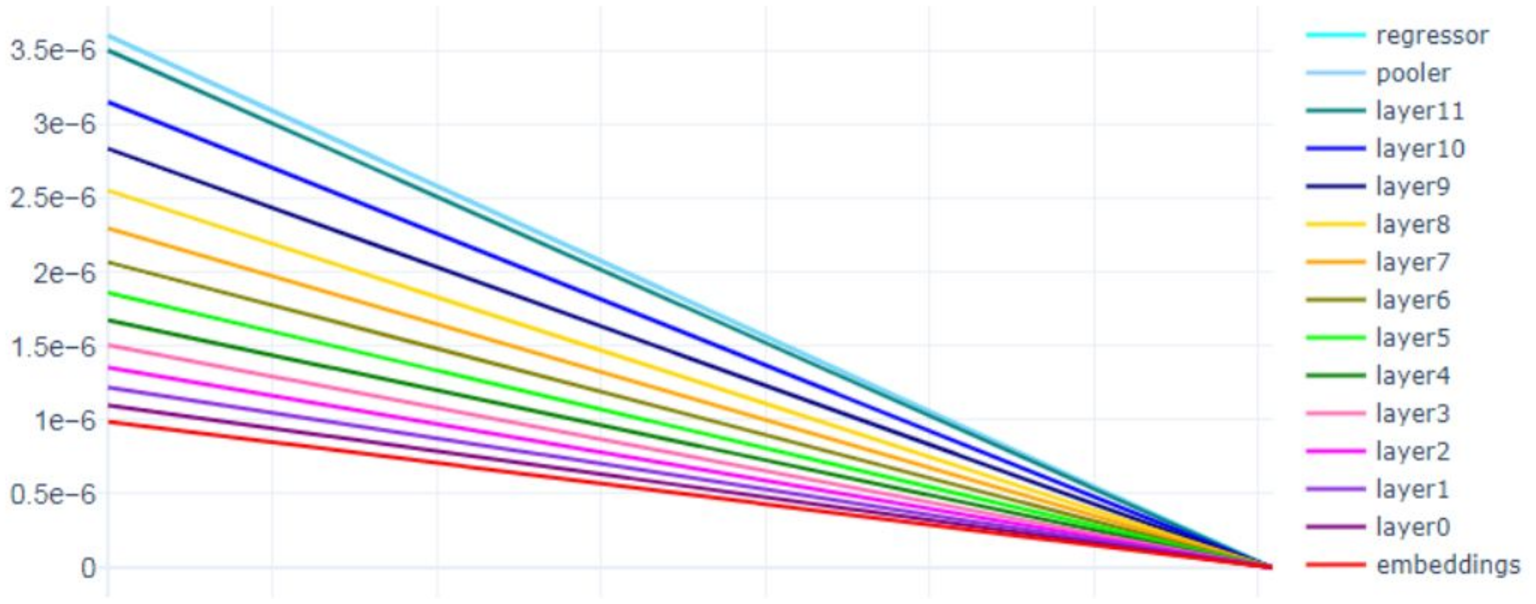
Fine-tuning Transformers | LLRD

- LLRD – Layer-wise Learning Rate Decay
- It is a method that applies higher learning rates for top layers and lower learning rates for bottom layers
- This is accomplished by setting the learning rate of the top layer and using a multiplicative decay rate to decrease the learning rate layer by layer from top to bottom

Fine-tuning Transformers | LLRD

- Different layers in the Transformer model usually capture different kinds of information
- Bottom layers often encode more common, general, and broad-based information
- Top layer, closer to the output, encodes information more localized and specific to the task on hand

Fine-tuning Transformers | LLRD



Re-Initializing Pretrained Layers

- Not training from scratch saves substantial resources and time
- Transformers usually have been pre-trained on a large corpus of text data, and they contain pre-trained weights that we could use
- However, to achieve better fine-tuning results, sometimes we need to discard some of these weights and re-initialize them during the fine-tuning process

Re-Initializing Pretrained Layers

This is done by:

- Preserve the useful low-level representations that encode more general information (bottom layers)

- Adapting the top layers (closer to the output) to our training task

- This way we would re-initialize the layers to suit our task, but keep the important general tasks intact

Fine-tuning Transformers | SWA

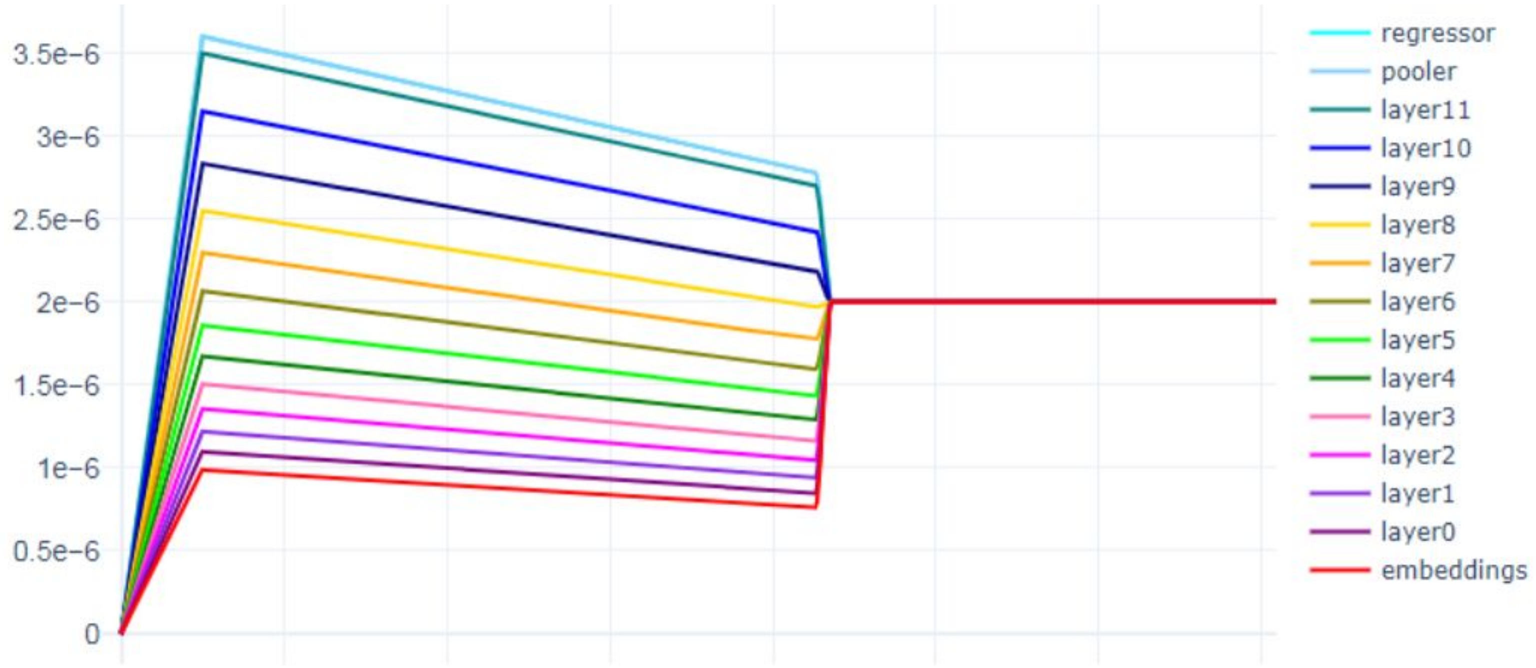
- SWA – Stochastic Weight Averaging
- It is a deep neural network training technique
- SWA is extremely easy to implement and has virtually no computational overhead compared to the conventional training schemes

Fine-tuning Transformers | SWA

How Does it work?

- Use a modified learning rate schedule
  - I.e. use standard decaying learning rate for the first 75% of training time
  - Then set the learning rate to a reasonably high constant value for the remaining 25% of the time
- Take an equal average of weights of the networks traversed
  - I.e. maintain a running average of the weights obtained at the end, within the last 25% of training time
  - After training is complete, we then set the weights of the network to the computed SWA averages

Fine-tuning Transformers | SWA



Breakout

15 min
(3-4 per room)

What applications can you think of using pre-trained models?
You can think both NLP & CV!
Even think about your Capstone :)

Designate *one person to share* from your breakout room

What questions do you have?

Feedback on Lecture and Concepts?

See you next on Thursday!