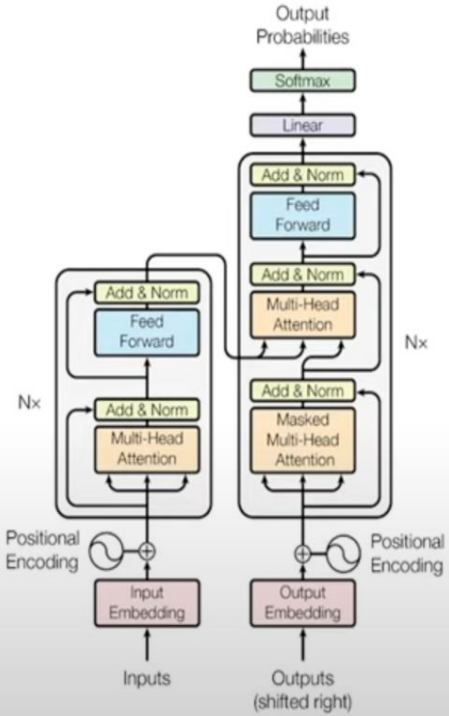
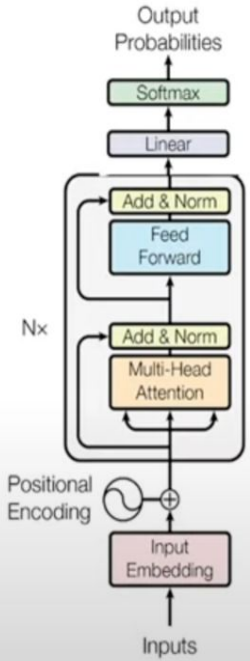
BERT & Autoencoder

Transformer Encoder architecture



Transformer



Transformer Encoder

Introducing BERT

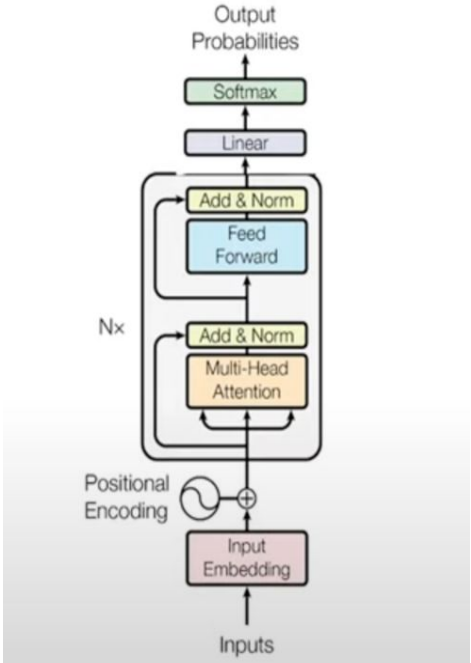
Stack of Encoders:
BERT-BASE: 12 encoder layers with 12 attention heads
BERT-LARGE: 24 encoder layers with 16 attention heads

Finer and finer input processing after each encoder

The size of embedding vector for two models is 768 and 1024, respectively

Use WordPiece embeddings (around 30k tokens) during training

Final linear layer changes according to the application



Tasks in GPT/LLaMA vs BERT

Question Answering in GPT/LLaMA: **Prompt Engineering**

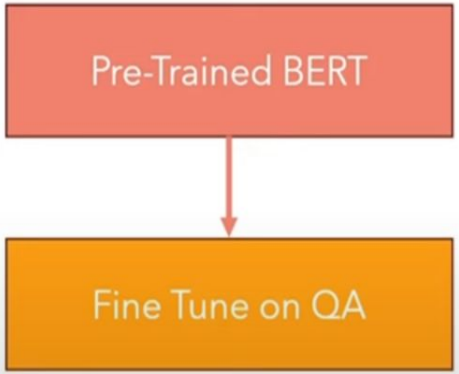
Default (GPT-3.5)

Context: "Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city."
Question: "What is the fashion capital of China? Answer with one word."
Answer: Shanghai.

Context: "Shenzhen is the tech capital of China, it is also close to Hong Kong."
Question: "What city is close to Hong Kong"?
Answer:

Shenzhen.

Question Answering in BERT: **Fine Tuning**



Let's focus on BERT in this lecture

Left and right context in BERT

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

softmax $\left(\frac{Q \times K^T}{\sqrt{768}}\right) =$

Q (10, 768) × K^T (768, 10)

This is the reason it is a **Bidirectional Encoder.**
Each token "attends" token to its left and tokens to its right in a sentence.

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

Masked Language Model (MLM): details

Rome is the capital of Italy, which is why it hosts many government buildings.

The pre-training procedure selects 15% of the tokens from the sentence to be masked.
When a token is selected to be masked (suppose the word "capital" is selected):

- 80% of the time it is replaced with the [MASK] token → Rome is the [MASK] of Italy, which is why it hosts many government buildings.
- 10% of the time it is replaced with a random token → Rome is the zebra of Italy, which is why it hosts many government buildings.
- 10% of the time it is not replaced → Rome is the capital of Italy, which is why it hosts many government buildings.

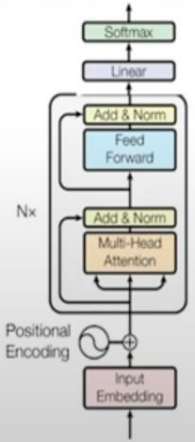
Masked Language Model (MLM): training

Target (1 token): capital

Loss → Run **backpropagation** to update the weights

Output (14 tokens):

TK1	TK2	TK3	TK4	TK5	TK6	TK7	TK8	TK9	TK10	TK11	TK12	TK13	TK14

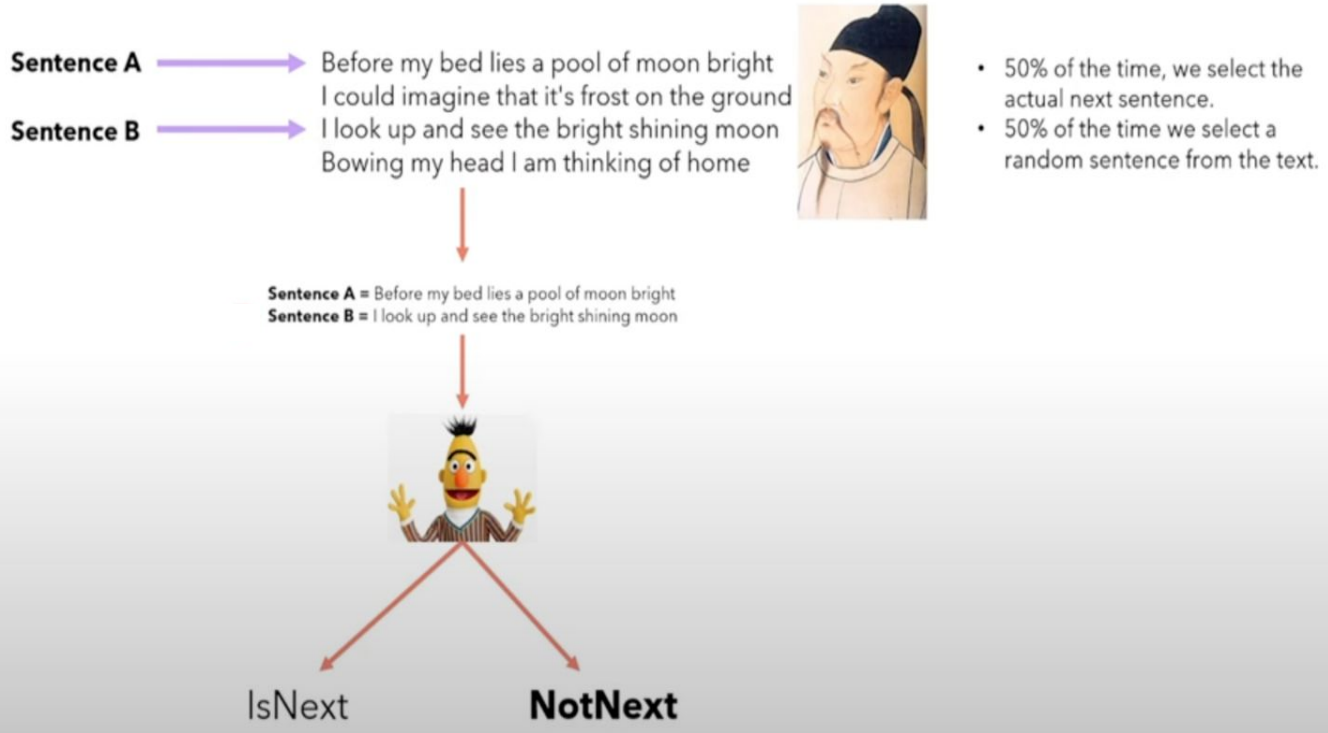


Also use segment embeddings (explained in upcoming slides)

Input (14 tokens): Rome is the [mask] of Italy, which is why it hosts many government buildings.

Next Sentence Prediction (NSP)

Many downstream applications (for example choosing the right answer given 4 choices) require learning relationships between sentences rather than single tokens, that's why BERT has been pre-trained on the Next Sentence Prediction task.

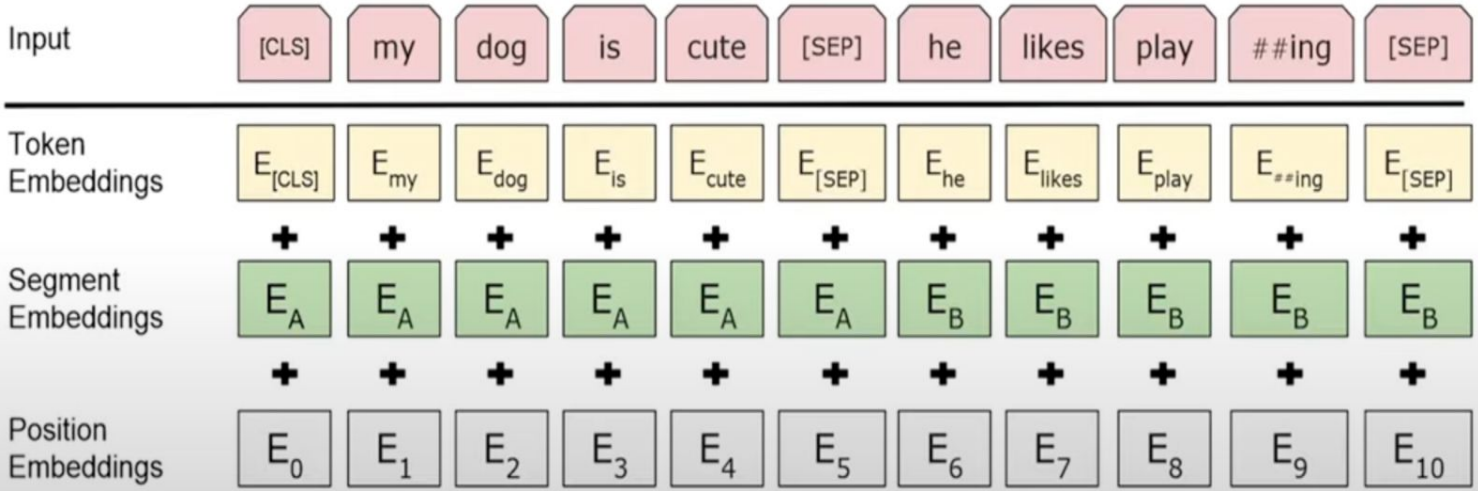


- 50% of the time, we select the actual next sentence.
- 50% of the time we select a random sentence from the text.

Next Sentence Prediction (NSP): segmentation embedding

Given the sentence A and the sentence B, how can BERT understand which tokens belongs to the sentence A and which to the sentence B? We introduce the segmentation embeddings!

We also introduce two special tokens: **[CLS]** and **[SEP]**



Next Sentence Prediction (NSP): training

Target (1 token):

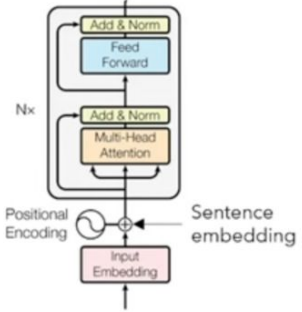
NotNext

Loss → Run **backpropagation** to update the weights

Linear Layer (2 output features) + **Softmax**

Output (20 tokens):

TK 1	TK 2	TK 3	TK 4	TK 5	TK 6	TK 7	TK 8	TK 9	TK 10	TK 11	TK 12	TK 13	TK 14	TK 15	TK 16	TK 17	TK 18	TK 19	TK 20



Before my bed lies a pool of moon bright
I could imagine that it's frost on the ground
I look up and see the bright shining moon
Bowing my head I am thinking of home

Input (20 tokens):

[CLS] Before my bed lies a pool of moon bright [SEP] I look up and see the bright shining moon

Sentence A **Sentence B**

Umar Jamil - https://github.com/hkproj/bert-from-scratch

[CLS] token in BERT

The **[CLS]** token always interacts with all the other tokens, as we do not use any mask.
So, we can consider the **[CLS]** token as a token that "captures" the information from all the other tokens.

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

	[CLS]	Before	my	bed	lies	a	pool	of	moon	bright
[CLS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

BERT does not need **manually** labeled data

Performs MLM and NSP together

Text Classification

Text classification is the task of assigning a label to a piece of text. For example imagine we are running an internet provider and we receive complaints from our customers. We may want to classify requests coming from users as hardware problems, software problems or billing issues.

My router's led is not working, I tried changing the power socket but still nothing.

Hardware

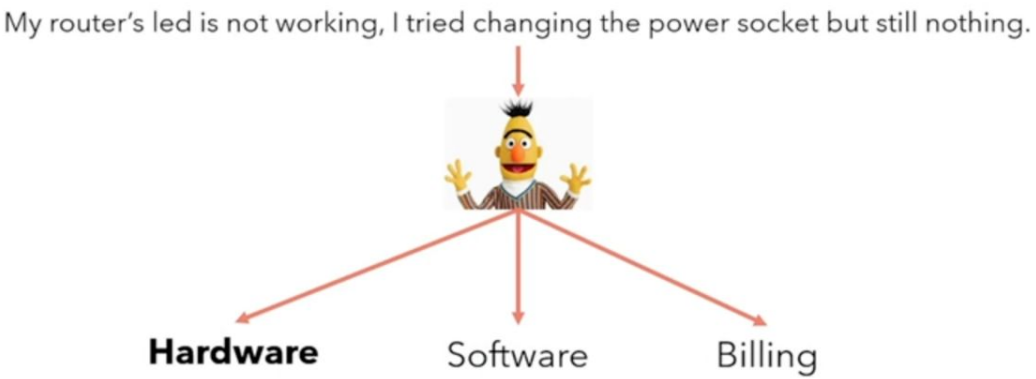
My router's web page doesn't allow me to change password anymore… I tried restarting it but nothing.

Software

In this month's bill I have been charged 100$ instead of the usual 60$, why is that?

Billing

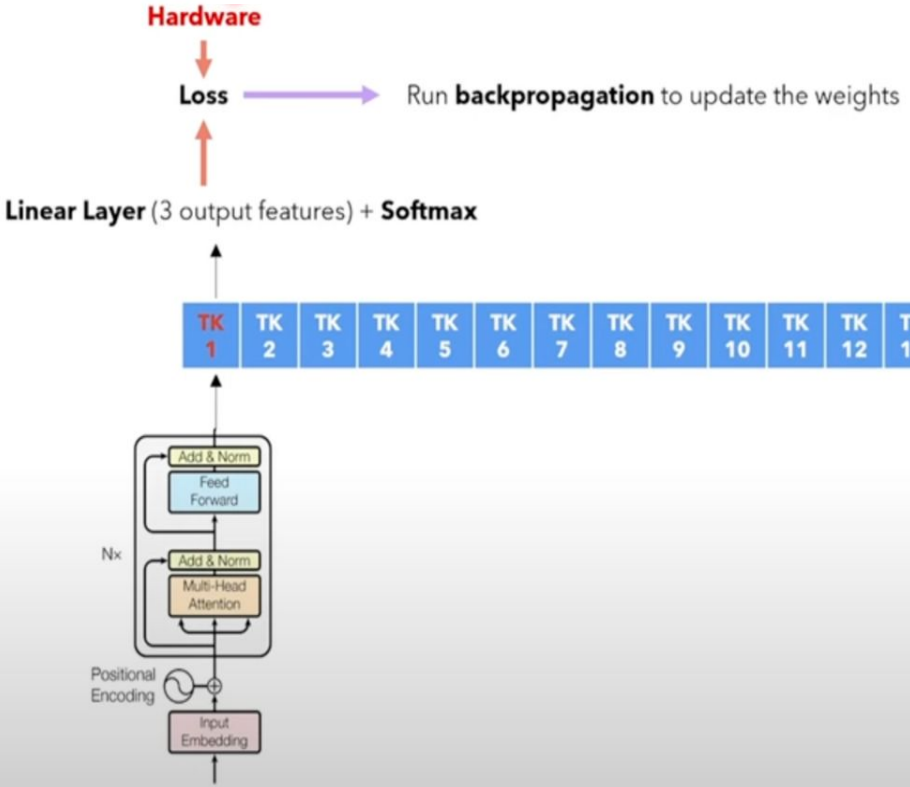
Text Classification: training



Let's discuss!

Text Classification: training

Target (1 token):



Transfer learning!

Input (16 tokens): [CLS] My router's led is not working, I tried changing the power socket but still nothing.

Question Answering

Question answering is the task of answering questions given a context.

Context: “Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.”

Question: "What is the fashion capital of China?"

Answer: “Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.”

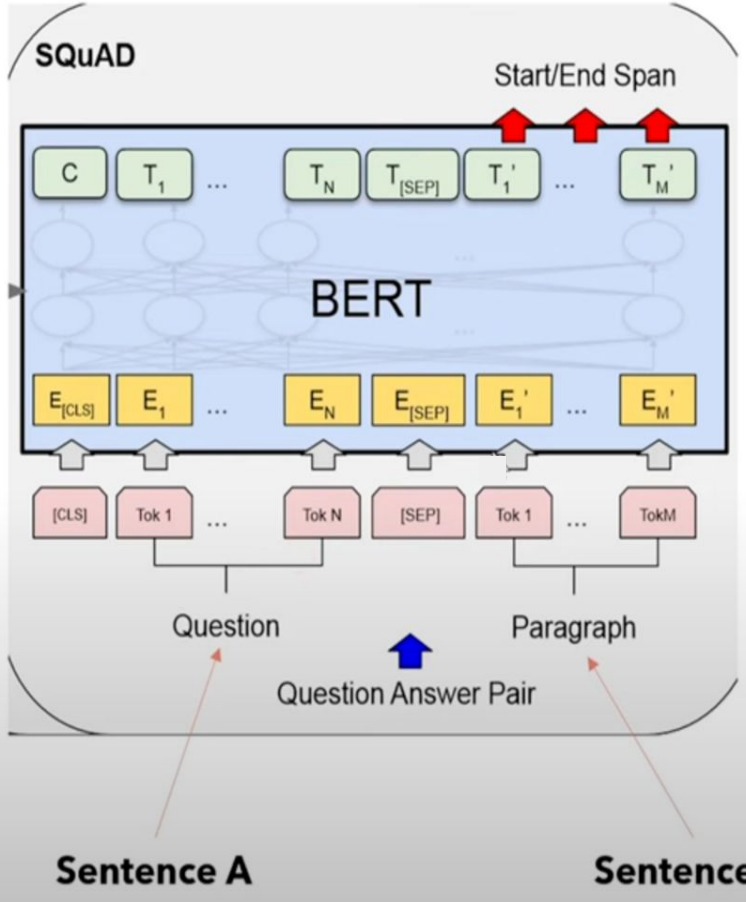
The model has to highlight the part of text that contains the answer.

Two problems:

1. We need to find a way for BERT to understand which part of the input is the context, which one is the question.
2. We also need to find a way for BERT to tell us where the answer starts and where it ends in the context provided.

Question Answering: sentence A and B

Fine-tuned on
Stanford Question Answering Dataset



Let's discuss!

Question Answering: **start** and **end** positions

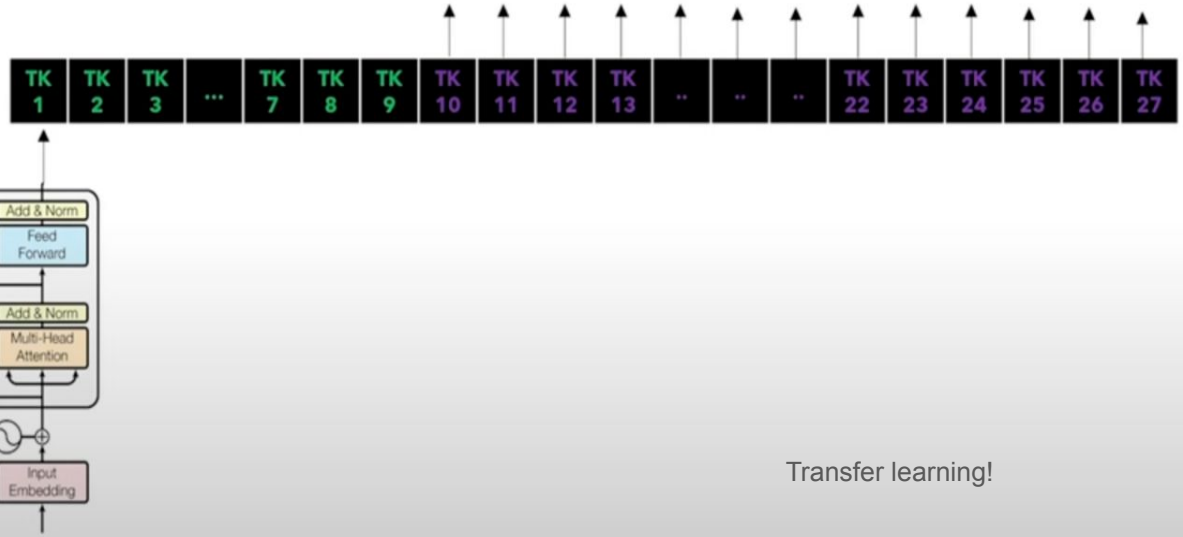
Target (1 token):

start=TK10, end=TK10

Loss → Run **backpropagation** to update the weights

Linear Layer (2 output features) + **Softmax**

Output (27 tokens):



Transfer learning!

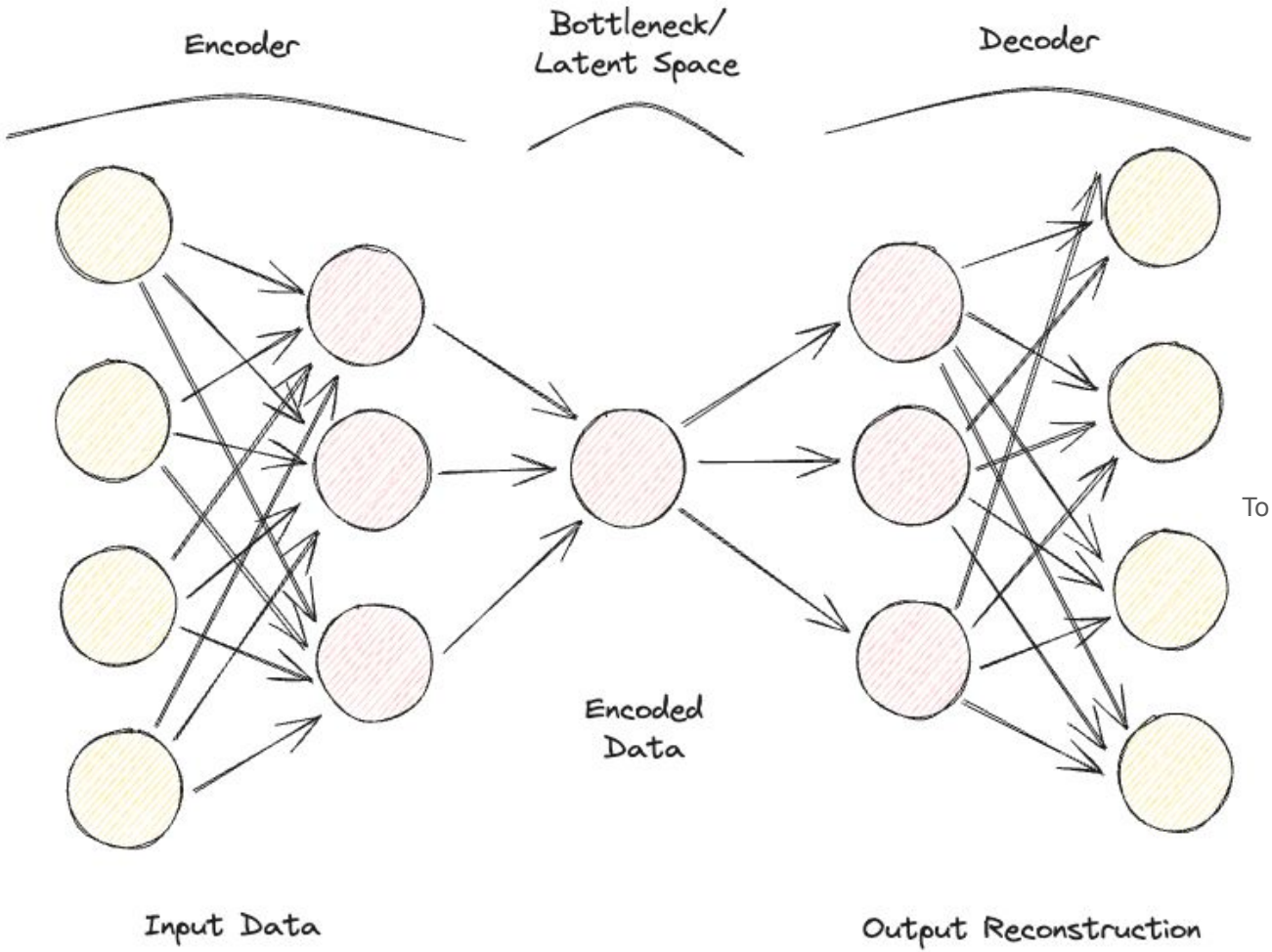
Input (27 tokens):

[CLS] What is the fashion capital of China? [SEP] Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.

Class Imbalance Problem

In reality, hate speech is about 5-10% of the actual data

Let’s discuss!



One class learning using Autoencoder framework

To solve class-imbalance problem

Applications: Anomaly detection and dimensionality reduction

Acknowledgments

Umar Jamil: https://github.com/hkproj/bert-from-scratch