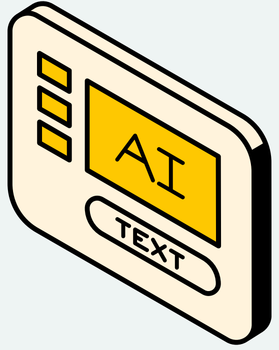
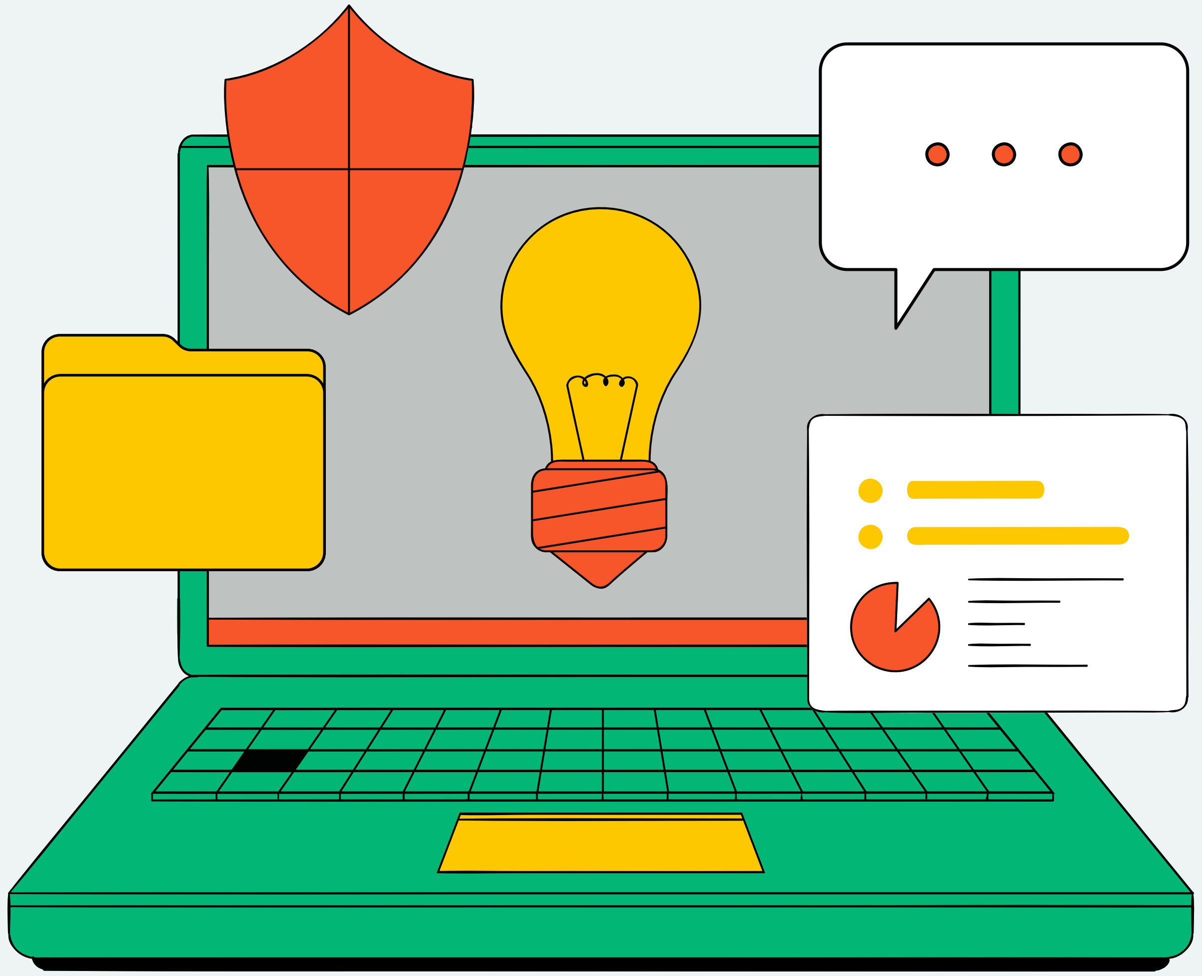
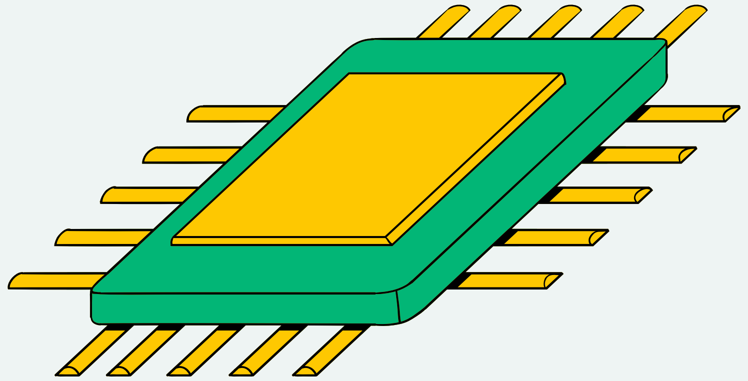




UNDERSTANDING TRANSFORMERS



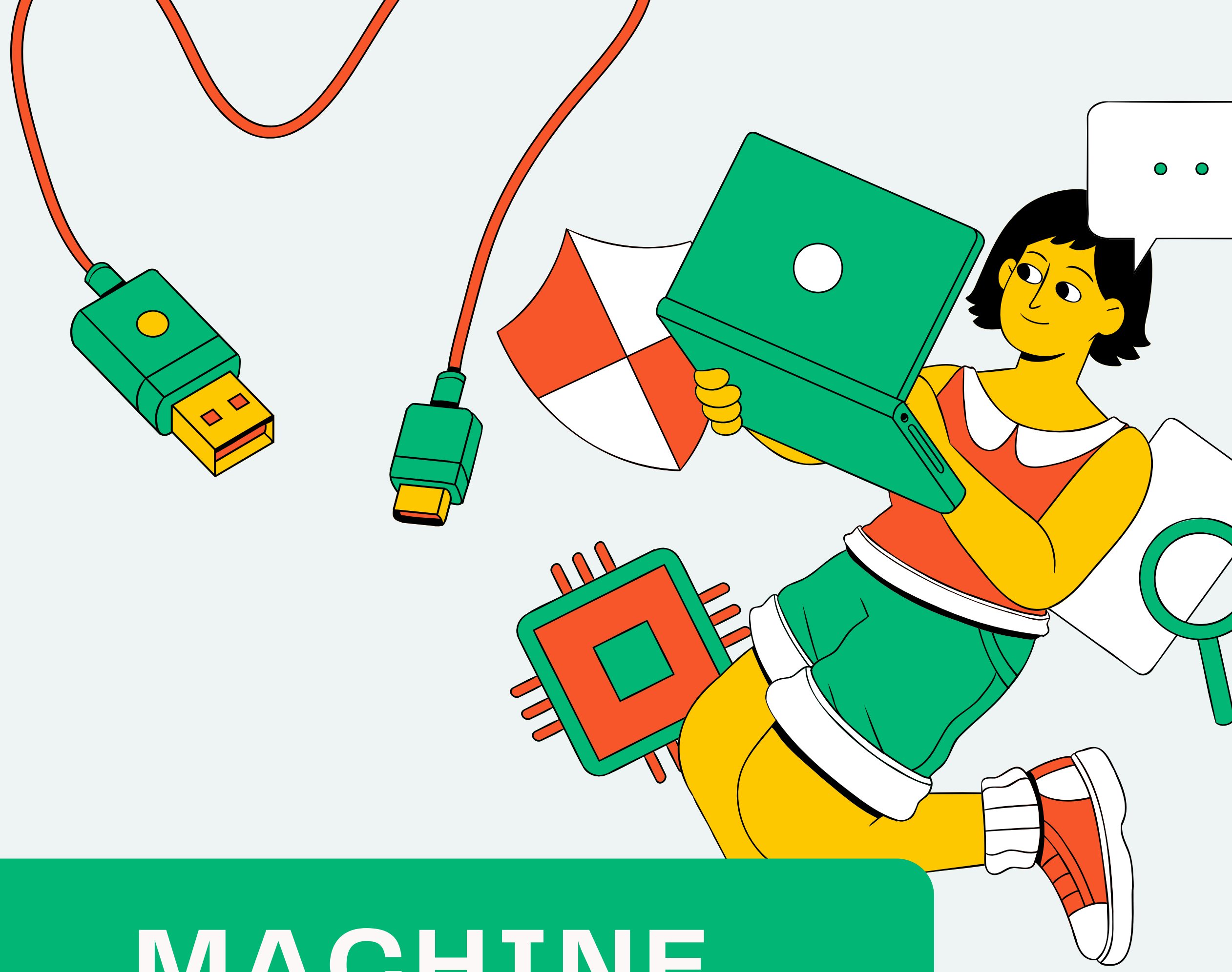
RIYA BHUTADA

ADVISOR: DR. MANAR MOHAISEN

04/22/2024

OUTLINE

- Historical Context
- Machine Learning & Deep Learning Algorithms
- Why Large Language Models?
- What are Transformers?
- Transformer Architecture
- Case Studies
- Cybersecurity Dataset
- Limitations
- Preparing for the Future
- Questions and Answers



DEALING WITH PROBLEMS

	TRADITIONAL PROGRAMMING	MACHINE LEARNING
GIVEN	INPUT & RULES	INPUT FEATURES & OUTPUT LABELS
FIGURES OUT	OUTPUT LABELS	RULES/PATTERNS

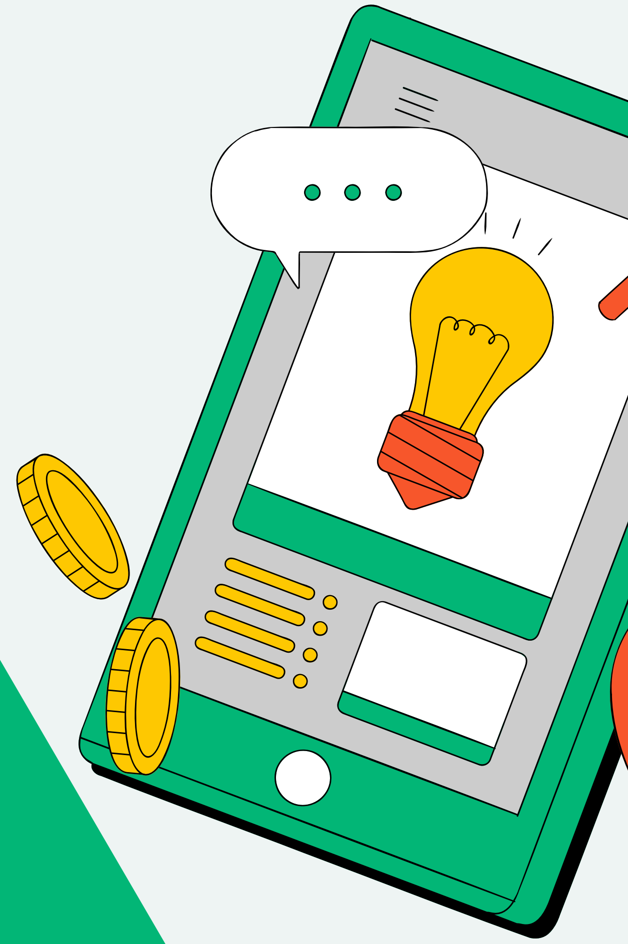
MACHINE LEARNING

- Structured Data
- XGBoost, Gradient Boosted Machine, Random Forest, Naïve Bayes, SVM, Nearest Neighbours, Tree Based, etc.
- Applications – Classification and Clustering problems

DEEP LEARNING

- Unstructured Data
- Neural Networks, Fully connected NNs, CNN, RNN, Transformers
- Applications – Recommendation, Seq2Seq (Translation, Speech Recognition), Classification/ Regression (Computer Vision, NLP)

NATURAL LANGUAGE PROCESSING TASKS



Text summarization

QA, chatbot/AI assistant

Text classification

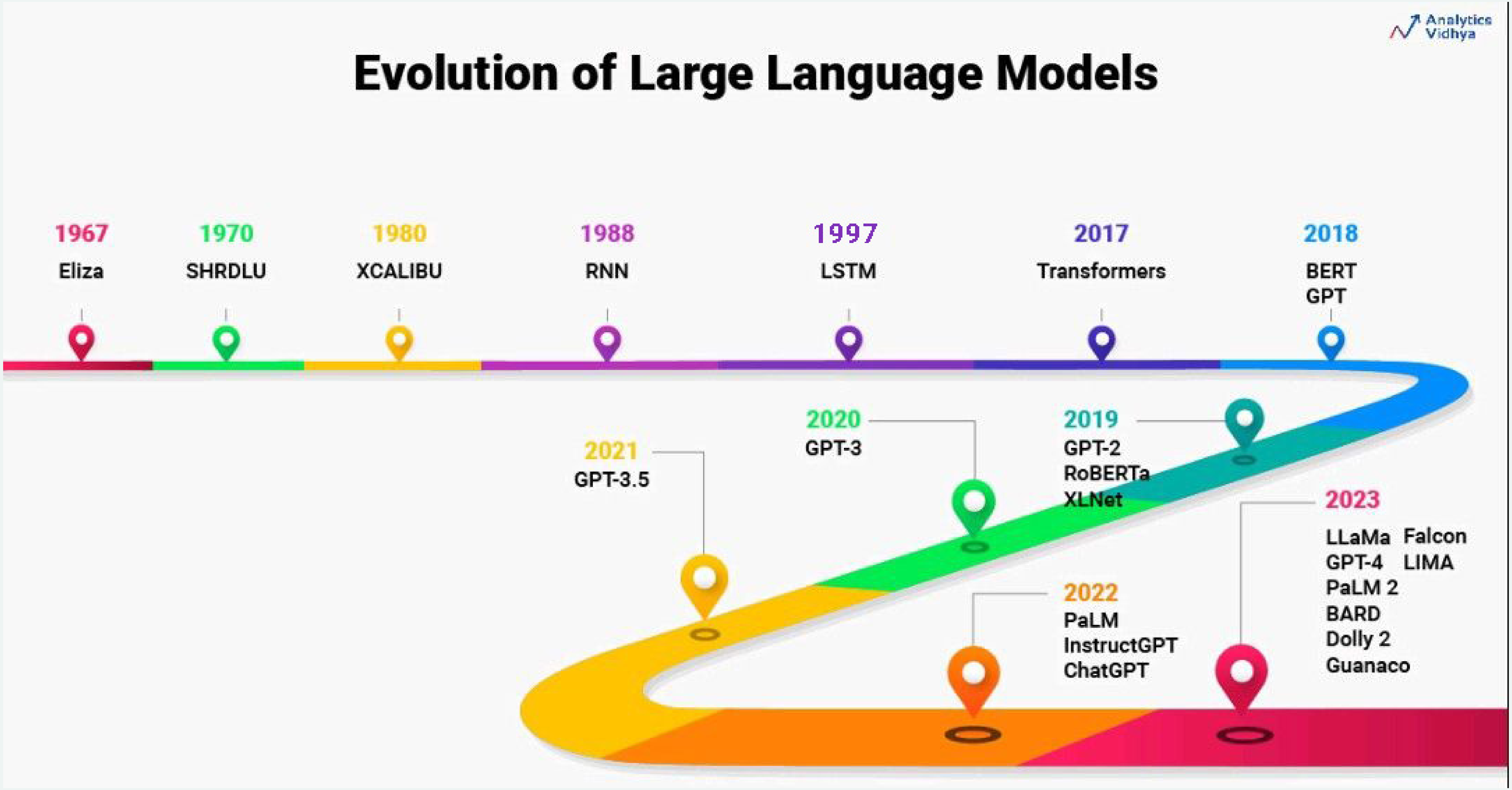
Text generation

Language translation

Sentiment analysis

WHAT ARE LARGE LANGUAGE MODELS?

- A deep learning algorithm that translates, predicts, summarizes, generates text, etc.
- Use Transformer architecture.
- Trained using massive datasets.
- The models typically work by being trained in a couple of phases, the first of which involves 'masking' different words within sentences.
- Pre-trained, then fine-tuned.


Analytics Vidhya
Evolution of Large Language Models
1967
Eliza
1970
SHRDLU
1980
XCALIBU
1988
RNN
1997
LSTM
2017
Transformers
2018
BERT
GPT
2019
GPT-2
RoBERTa
XLNet
2020
GPT-3
2021
GPT-3.5
2022
PaLM
InstructGPT
ChatGPT
2023
LLaMa
Falcon
GPT-4
LIMA
PaLM 2
BARD
Dolly 2
Guanaco

WHY TRANSFORMERS ?

USING LARGE LANGUAGE MODELS FOR NLP TASKS

NO LABELS, MORE PERFORMANCE !

Before transformers arrived, users had to train neural networks with large, labeled datasets that were costly and time-consuming to produce.

- Finds patterns between elements mathematically, making available the trillions of images and petabytes of text data on the web and in corporate databases.
- Parallel processing possible because of the math, so models run fast

TRANSFORMER ARCHITECTURES

Tokenize input and with math equations discover relations between tokens.

ENCODER ONLY

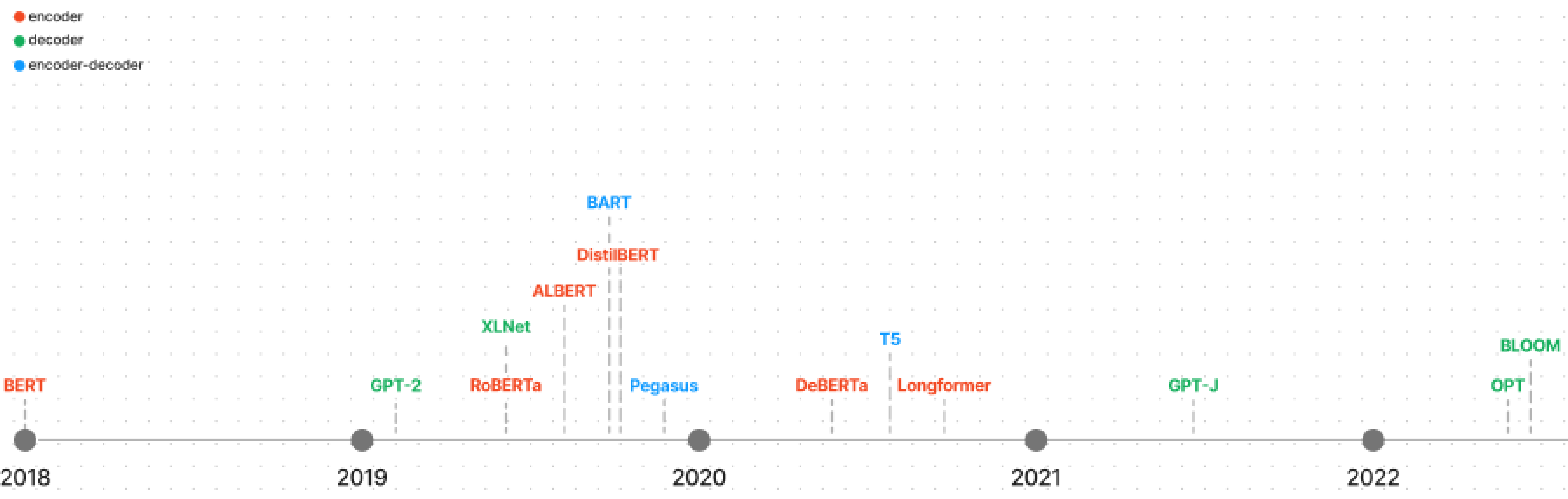
Tasks that require understanding of the input, such as sentence classification and named entity recognition.

DECODER ONLY

Generative tasks such as text generation.

ENCODER-DECODER

Generative tasks that require an input, such as translation or summarization.



encoder
decoder
encoder-decoder
BART
DistilBERT
ALBERT
XLNet
T5
BLOOM
BERT
GPT-2
RoBERTa
Pegasus
DeBERTa
Longformer
GPT-J
OPT
2018
2019
2020
2021
2022

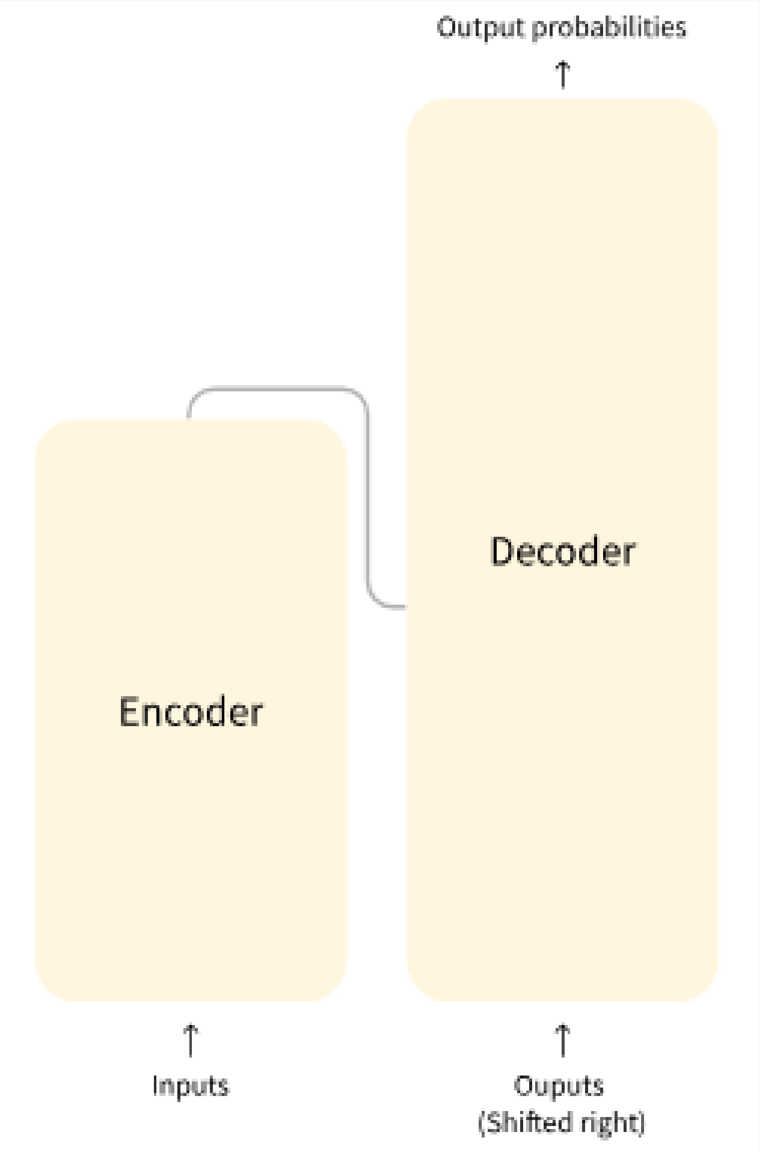
TRANSFORMER

ENCODER

Receives input and builds a representation.
Optimized to acquire understanding from the input.

DECODER

Uses the encoder's representation (features) along with other inputs to generate a target sequence.
Optimized for generating outputs.



Word embedding

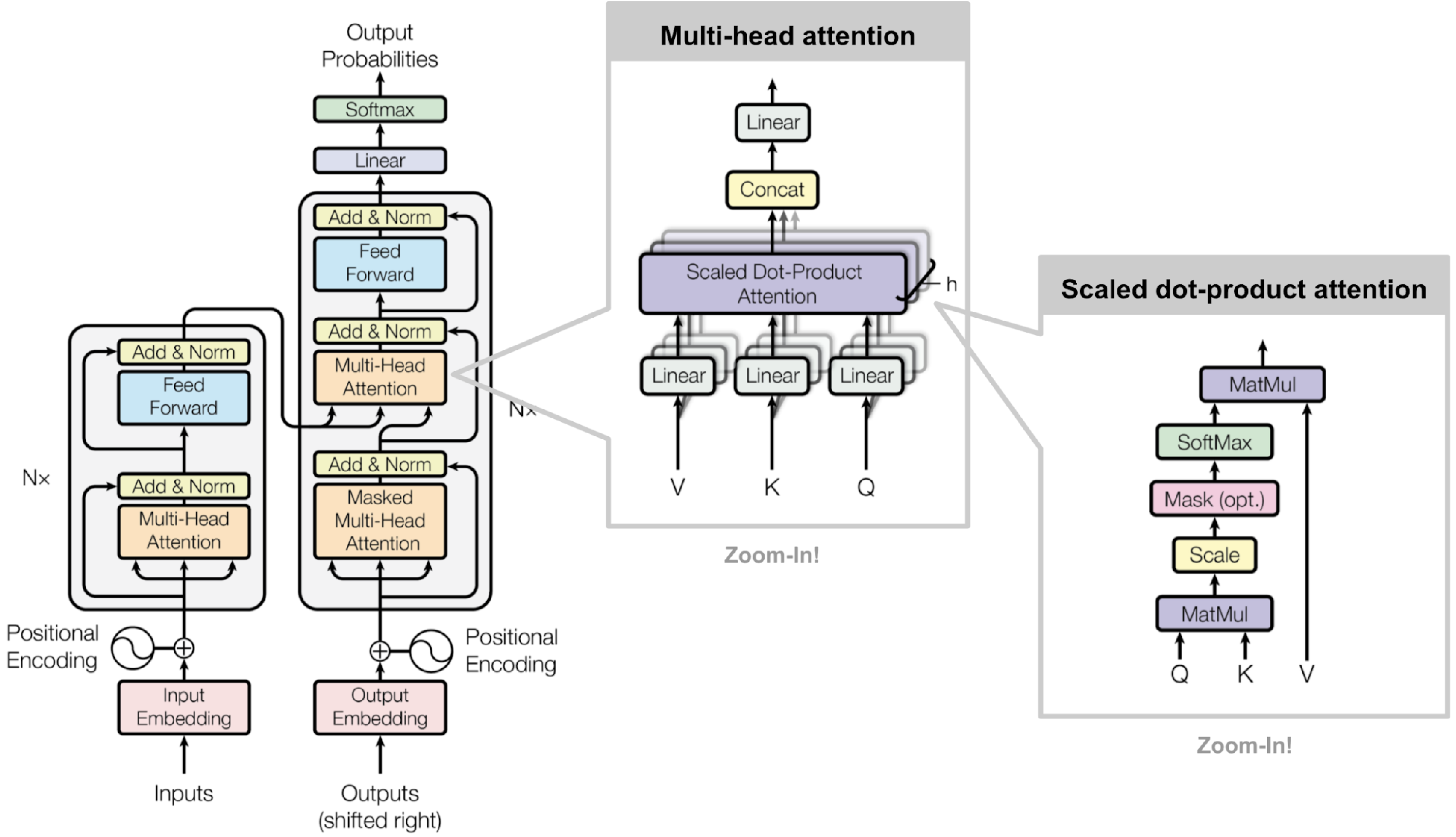
Positional encoding

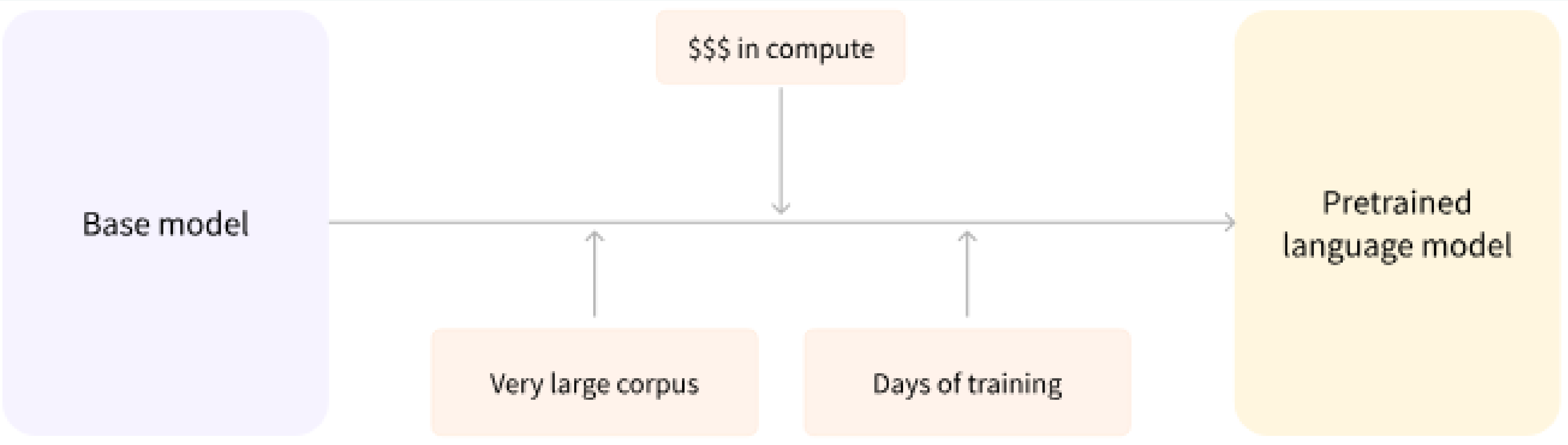
Self-attention

Multi-head self attention

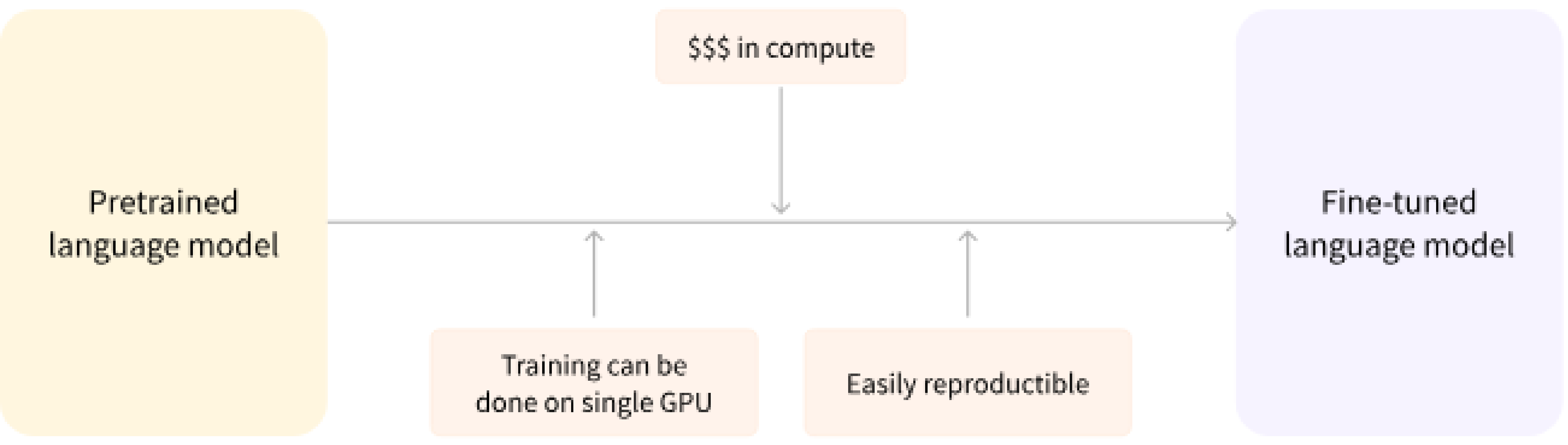
Feed-forward layer

Normalization


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Multi-head attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
Zoom-In!
Scaled dot-product attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Zoom-In!



OpenAI gave a disclaimer that "ChatGPT sometimes writes plausible-sounding but incorrect or nonsensical answers."



CASE STUDY - YELP DATASET

```
Downloading readme: 100%  6.72k/6.72k [00:00<00:00, 194kB/s]
Downloading data: 100%  299M/299M [00:06<00:00, 56.4MB/s]
Downloading data: 100%  23.5M/23.5M [00:01<00:00, 19.1MB/s]
Generating train split: 100%  650000/650000 [00:04<00:00, 63723.99 examples/s
Generating test split: 100%  50000/50000 [00:00<00:00, 178020.39 examples/s]
{'label': 0,
 'text': 'My expectations for McDonalds are t rarely high. But for one to still fail so spectacularly...
something special!\\nThe cashier took my friends\'s order, then promptly ignored me. I had to force myse
cashier who opened his register to wait on the person BEHIND me. I waited over five minutes for a gigant
included precisely one kid\'s meal. After watching two people who ordered after me be handed their food,
mine was. The manager started yelling at the cashiers for \\"serving off their orders\\" when they didn'
food. But neither cashier was anywhere near those controls, and the manager was the one serving food to
clearing the boards.\\nThe manager was rude when giving me my order. She didn\'t make sure that I had ev
RECEIPT, and never even had the decency to apologize that I felt I was getting poor service.\\nI\'ve eat
McDonalds restaurants for over 30 years. I\'ve worked at more than one location. I expect bad days, bad
occasional mistake. But I have yet to have a decent experience at this store. It will remain a place I a
someone in my party needs to avoid illness from low blood sugar. Perhaps I should go back to the raciall
of Steak n Shake instead!'}
```

- Text Classification problem
- 5 labels – 1 star, 2 star, 3 star, 4 star, 5 star
- 700, 000 rows. (For case study, using 2000 train and 1000 test samples.)

Hugging Face

- Offers many open-source ML models for various tasks.
- Many datasets.
- Task-specific fine-tuned models ready to use.

Natural Language Processing

Text Classification · Token Classification · Table Question Answering · Question Answering · Zero-Shot Classification · Translation · Summarization · Feature Extraction · Text Generation · Text2Text Generation · Fill-Mask · Sentence Similarity

VECTOR/WORD EMBEDDING

- Convert words into numbers that capture their meaning.
- Similar data points are clustered together in multi-dimensional space.

```
from datasets import load_dataset

dataset = load_dataset("yelp_review_full")
```

```
dataset
```

```
DatasetDict({
    train: Dataset({
        features: ['label', 'text'],
        num_rows: 650000
    })
    test: Dataset({
        features: ['label', 'text'],
        num_rows: 50000
    })
})
```

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("google-bert/bert-base-cased")

def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True)

tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

tokenizer_config.json: 100% 49.0/49.0 [00:00<0

POSITIONAL ENCODING

- Encode positions of the word.
- Use this to keep track of word order.

FEED-FORWARD LAYER

- Multiple fully connected layers that transform the input embedding.
- Find high level abstractions. Understand user’s intent.

SELF ATTENTION

e.g. The animal didn’t cross the street because it was too tired.

“it” refers to what? “animal” or “street”?

Self-attention determines the relevance of each nearby word to “it”

- Encode the relationships among the words.
- Tells model to learn different parts of sequence or entire context of sentence.
- Focus on parts of text relevant to task at hand

Training using Transformer's Trainer class

- Load your model and specify the number of expected labels.
- Create a TrainingArguments class and initialize all the flags and hyperparameters to tune
- Create an evaluation method

```
from transformers import AutoModelForSequenceClassification
from transformers import TrainingArguments


model = AutoModelForSequenceClassification.from_pretrained("google-bert/bert-base-cased", num_labels=5)
training_args = TrainingArguments(output_dir="./test_trainer", evaluation_strategy="epoch")
```

Using Evaluate library's **accuracy** function

Convert the logits to predictions.

Logits are the raw, unnormalized predictions generated by a model before applying any activation function

```
import numpy as np
import evaluate

metric = evaluate.load("accuracy")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)
```

Trainer

```
[ ]  from transformers import Trainer

     trainer = Trainer(
         model=model,
         args=training_args,
         train_dataset=small_train_dataset,
         eval_dataset=small_eval_dataset,
         compute_metrics=compute_metrics,
     )

     trainer.train()
```

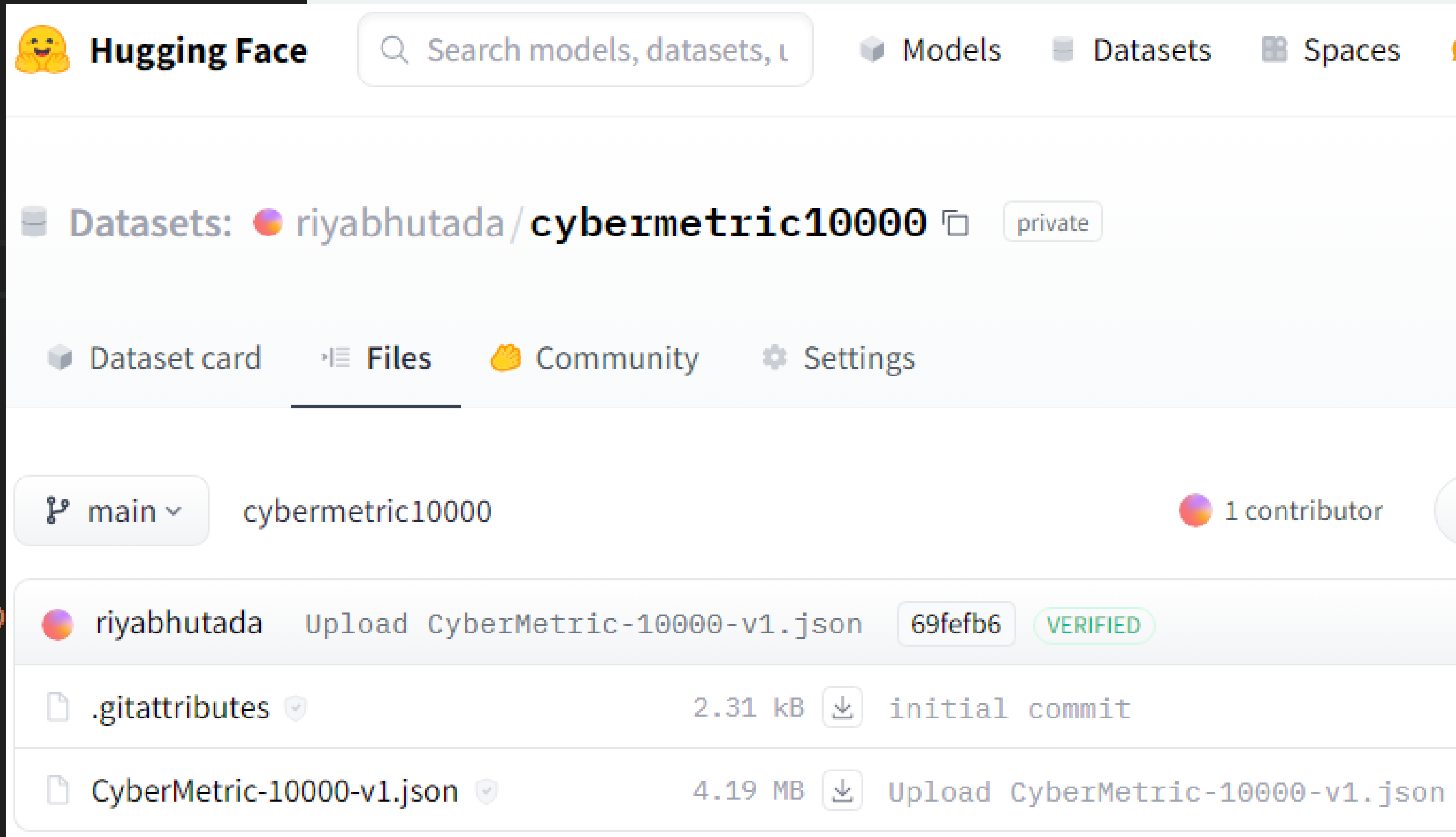
[750/750 11:16, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	1.069502	0.561000
2	1.050700	0.964069	0.584000
3	1.050700	1.067736	0.585000

```
TrainOutput(global_step=750, training_loss=0.874600809733073, metrics={'train_runtime': 678.5136,
'train_samples_per_second': 8.843, 'train_steps_per_second': 1.105, 'total_flos': 1578708854784000.0, 'train_loss':
0.874600809733073, 'epoch': 3.0})
```

USING CYBERSECURITY DATA

```json
"questions": [
    {
        "question": "Which of the following is a desirable property of a biometric
        system?",
        "answers": {
            "A": "Permanent",
            "B": "Transferability",
            "C": "Uniformity",
            "D": "Forgiveness"
        },
        "solution": "A"
    },
    {
        "question": "In TCP/IP networking, which protocol is used to ho
        addresses and routing information in a packet?",
        "answers": {
            "A": "HTTP",
            "B": "IP",
            "C": "Routing Information Protocol (RIP)",
            "D": "TCP"
        },
        "solution": "B"
    },
```




```
dataset
```

```
Dataset({
    features: ['answers', 'question', 'label'],
    num_rows: 10211
})
```

PREPARING DATA FOR TOKENIZATION

```
dataset
```

```
Dataset({
    features: ['answers', 'question', 'label'],
    num_rows: 10211
})
```

```
dataset
```

```
Dataset({
    features: ['label', 'question', 'A', 'B', 'C', 'D'],
    num_rows: 10211
})
```

AFTER TOKENIZATION

```
tokenized_dataset_train:  Dataset({
    features: ['question', 'label', 'A', 'B', 'C', 'D', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 6534
})
        okenized_dataset_eval:  Dataset({
    features: ['question', 'label', 'A', 'B', 'C', 'D', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 1634
})
        okenized_dataset_test:  Dataset({
    features: ['question', 'label', 'A', 'B', 'C', 'D', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 2043
})
```

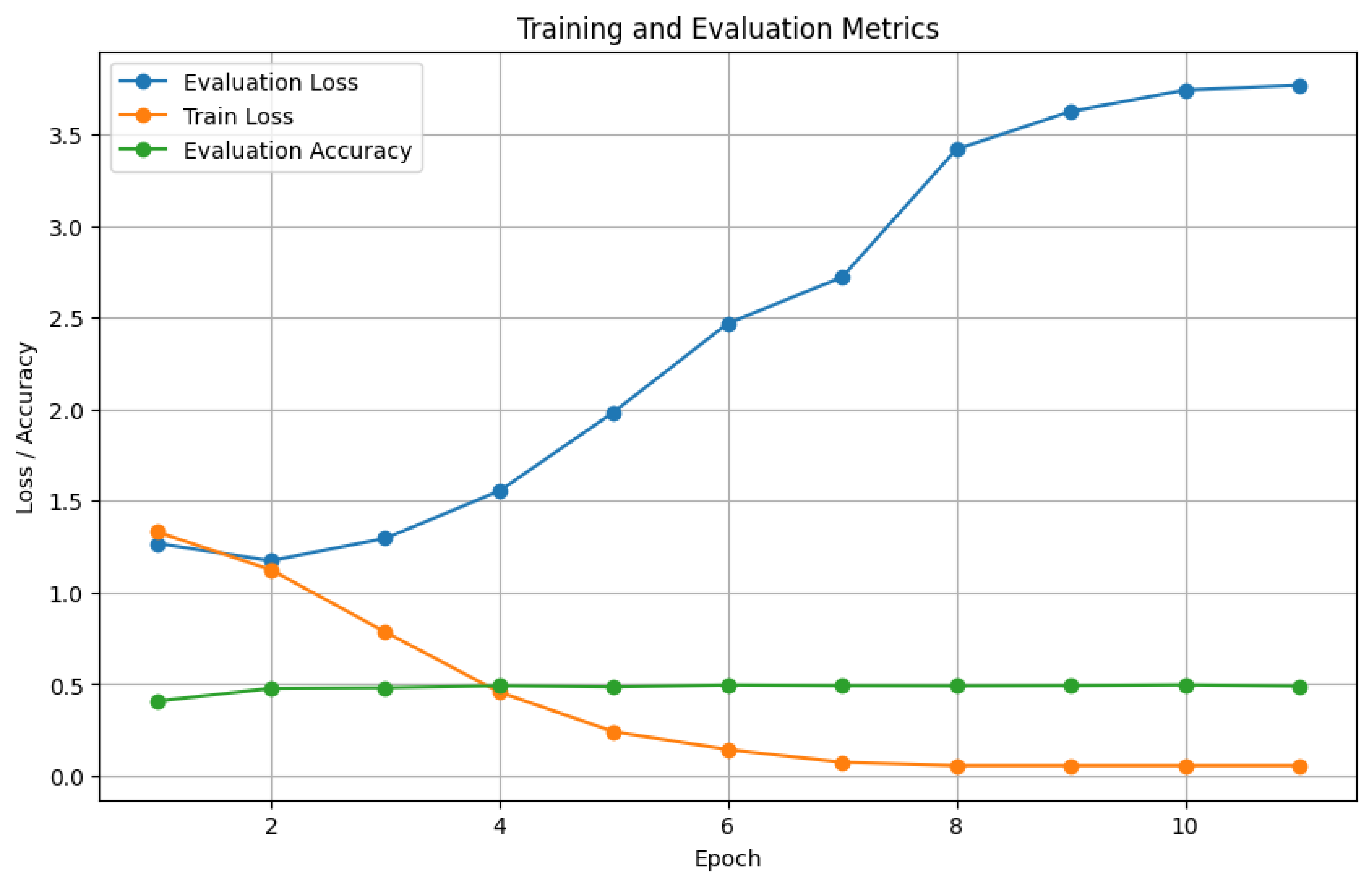
TRAINING THE MULTIPLE CHOICE MODEL

```
[ ]  from transformers import BertForMultipleChoice, TrainingArguments, Trainer

     # model = BertForMultipleChoice.from_pretrained("google-bert/bert-base-uncased")
     training_args = TrainingArguments(
         output_dir="my_cybermetric_model_10000_bert4",
         evaluation_strategy="epoch",
         save_strategy="epoch",
         load_best_model_at_end=True,
         #learning_rate=5e-5,
         per_device_train_batch_size=16,
         per_device_eval_batch_size=16,
         num_train_epochs=11,
         # weight_decay=0.01,
         # push_to_hub=True,
     )

     trainer = Trainer(
         model=model,
         args=training_args,
         train_dataset=tokenized_dataset_train,
         eval_dataset=tokenized_dataset_eval,
         tokenizer=tokenizer,
         data_collator=DataCollatorForMultipleChoice(tokenizer=tokenizer),
         compute_metrics=compute_metrics,
     )

     trainer.train()
```



Problem

- Decrease in evaluation loss
- Decrease in training loss
- Overfitting ?

Possible solutions?

- Training Arguments
- Dataset size
- Injecting Cyber-security related vocabulary in the tokenizer
- Other models

REVAMP DATASET FOR A DIFFERENT TASK

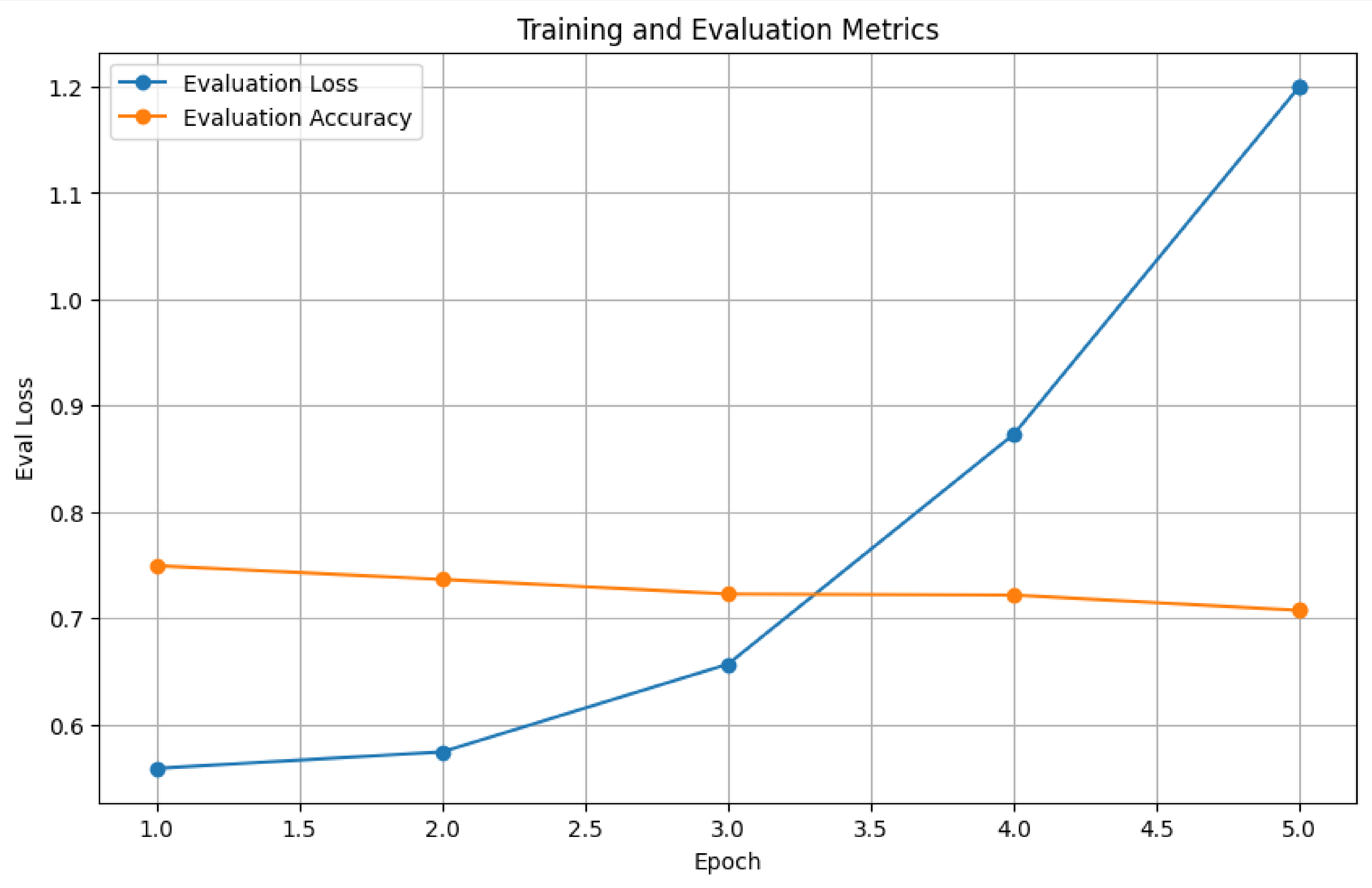
SEQUENCE CLASSIFICATION !

```
dataset
```

```
Dataset({
    features: ['new_statement', 'label'],
    num_rows: 20452
})
```

```
id2label = {0: "FALSE", 1: "TRUE"}
label2id = {"FALSE": 0, "TRUE": 1}
```

```
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert/distilbert-base-uncased", num_labels=2, id2label=id2label, label2id=label2id
)
```



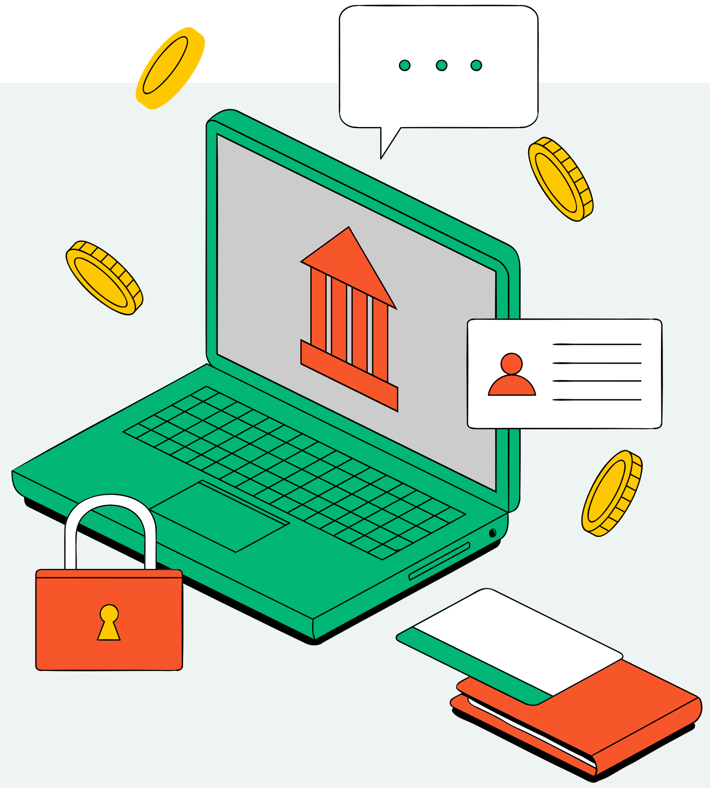

```
{'loss': 0.5571, 'grad_norm': 2.331603527069092, 'learning_rate': 4.755620723362659e-05, 'epoch': 0.24}
{'loss': 0.5565, 'grad_norm': 1.544159173965454, 'learning_rate': 4.511241446725318e-05, 'epoch': 0.49}
{'loss': 0.5736, 'grad_norm': 4.510655403137207, 'learning_rate': 4.266862170087977e-05, 'epoch': 0.73}
{'loss': 0.5582, 'grad_norm': 1.5657628774642944, 'learning_rate': 4.0224828934506356e-05, 'epoch': 0.98}
```

```
{'eval_loss': 0.5587450265884399, 'eval_accuracy': 0.7494500122219506, 'eval_runtime': 257.0491, 'eval_samples_per_se
15.915, 'eval_steps_per_second': 1.992, 'epoch': 1.0}
```

LIMITATIONS

```
---------------------------------------------------------------------------
OutOfMemoryError                          Traceback (most recent call last)
<ipython-input-35-33be74765a41> in <cell line: 15>()
     13 )
     14 
---> 15 trainer = Trainer(
     16     model=model,
     17     args=training_args,
```

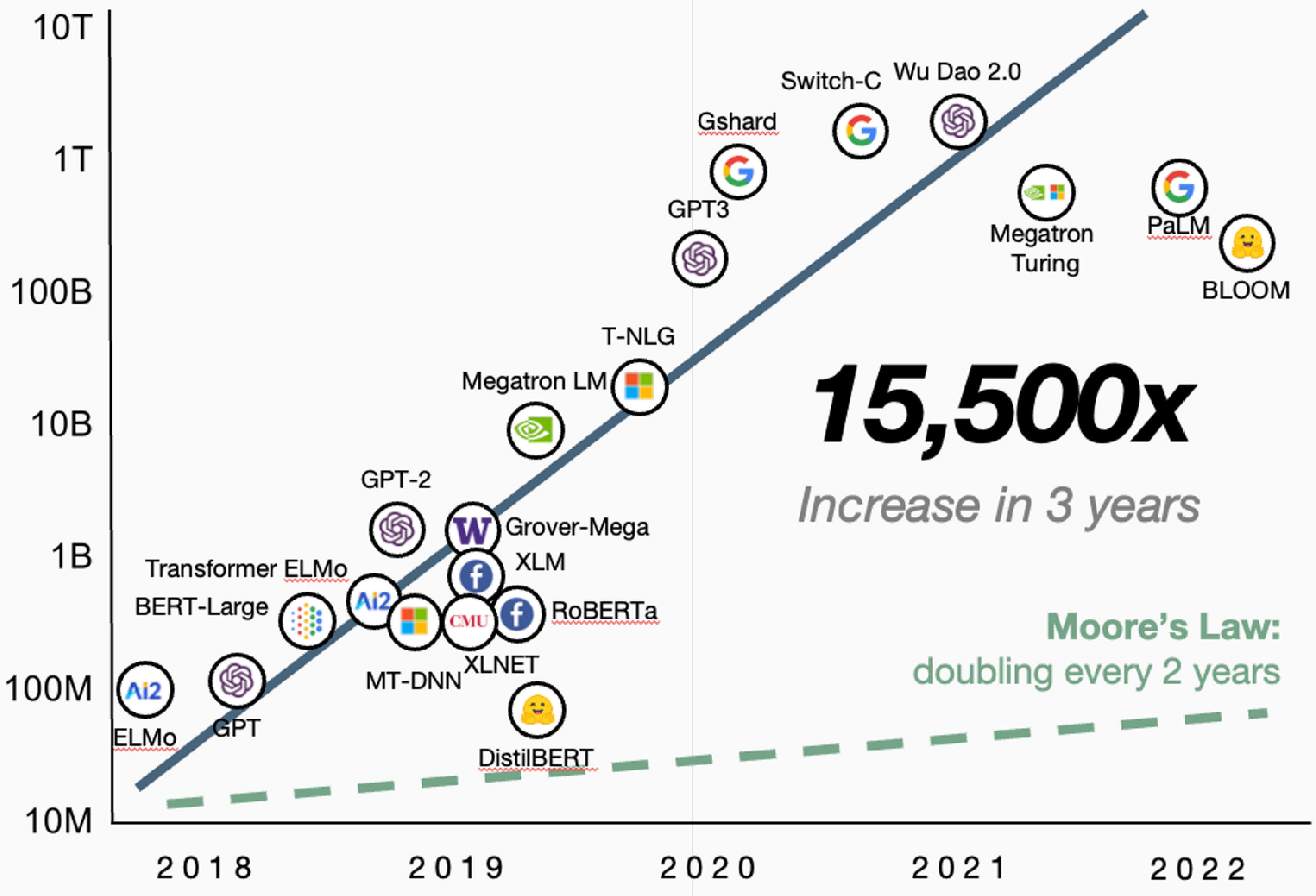
- SPACE
- TIME
- Cost for hardware like GPU (Compute units)
- Domain Specificity
- Considerable amount of data



```
OutOfMemoryError: CUDA out of memory. Tried to allocate 64.00 MiB. GPU 0 has a total capacity of 22.17 GiB of
which 34.88 MiB is free. Process 38344 has 22.13 GiB memory in use. Of the allocated memory 21.94 GiB is
allocated by PyTorch, and 9.11 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory
is large try setting PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to avoid fragmentation.  See
documentation for Memory Management  (https://pytorch.org/docs/stable/notes/cuda.html#environment-variables)
```

PREPARING FOR THE FUTURE

- Choose a pre-trained model more suitable for your dataset. In this case, more close to Cybersecurity, or IT domain?
- Explore models for different NLP tasks
- Improving model efficiency (model compression techniques, distillation methods, etc/).
- Deployment and pipelines.
- Learning about transfer learning techniques.



10T
1T
100B
10B
1B
100M
10M
2018
2019
2020
2021
2022
ELMo
GPT
BERT-Large
Transformer ELMo
MT-DNN
XLNET
DistilBERT
GPT-2
Grover-Mega
XLM
RoBERTa
Megatron LM
T-NLG
GPT3
Gshard
Switch-C
Wu Dao 2.0
Megatron Turing
PaLM
BLOOM
15,500x
Increase in 3 years
Moore's Law:
doubling every 2 years

THANK YOU!

QUESTIONS ?