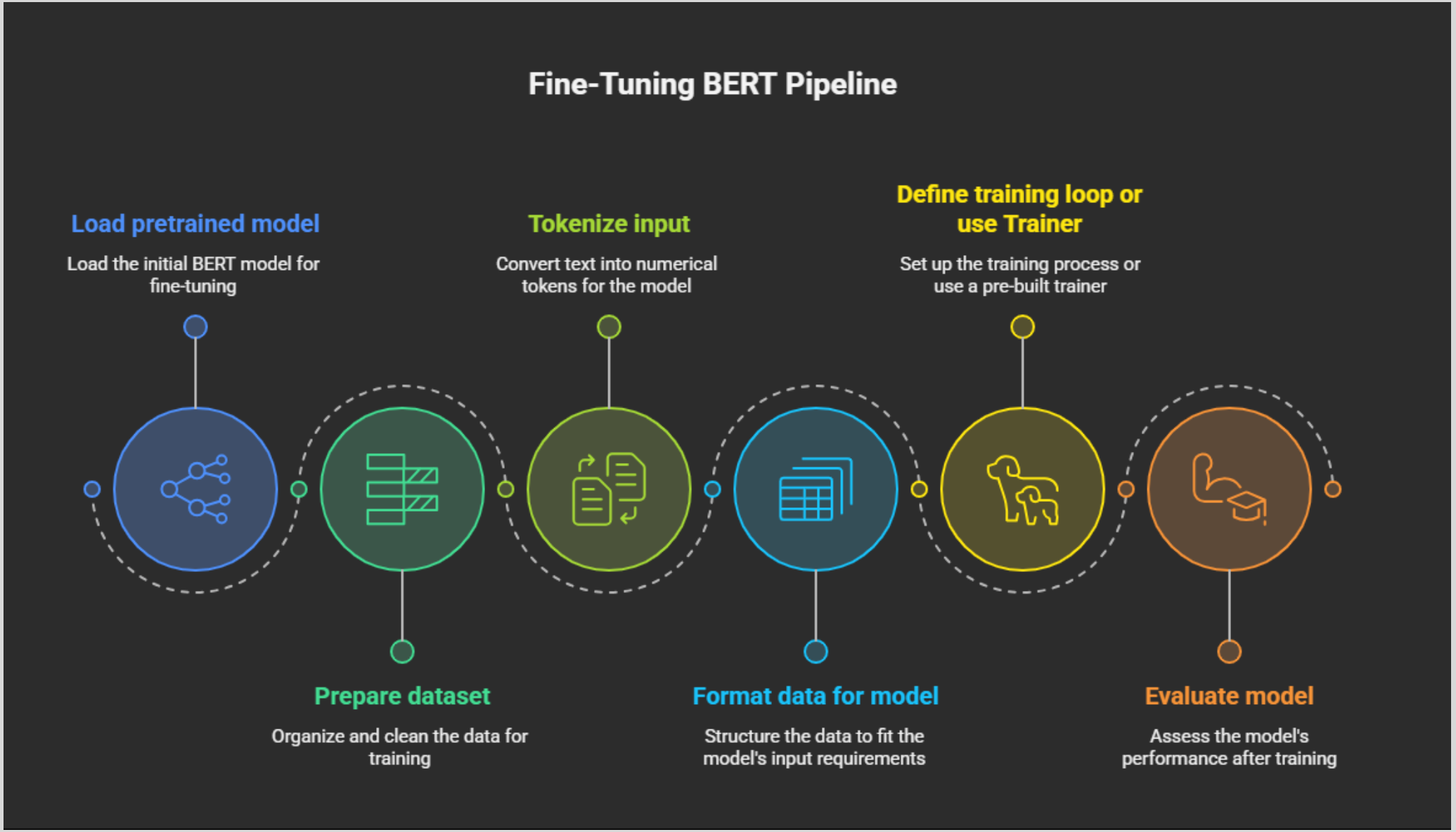
FINE-TUNING PLMS

Day 4 08/05/2025

Quick Recap

- What is a Pre-trained Language Model?
- Examples of PLMs
- Why use pretrained models?
- What is fine-tuning?
- Why Fine-Tune Instead of Training from Scratch?
- What do you know about HuggingFace?


Fine-Tuning BERT Pipeline
Load pretrained model
Load the initial BERT model for fine-tuning
Prepare dataset
Organize and clean the data for training
Tokenize input
Convert text into numerical tokens for the model
Format data for model
Structure the data to fit the model's input requirements
Define training loop or use Trainer
Set up the training process or use a pre-built trainer
Evaluate model
Assess the model's performance after training

Hugging Face

- It is a leading platform and library for NLP and AI models.
- It provides pretrained models, datasets, and tools to easily build, fine-tune, and deploy NLP and vision models.
- Their most popular library is 🤗 **Transformers** – used widely in research and industry.

It Offers:

- **Pretrained Models** (NLP, vision, audio, multimodal)
  - BERT, GPT, T5, RoBERTa, BLOOM, Whisper, etc.
  - Models for 100+ languages (including Indic languages)
- **Datasets**
  - Ready-to-use datasets for classification, QA, translation, summarization, etc.
- **Trainer API**
  - Simplifies training and fine-tuning of models.
- **Model Hub**
  - A centralized repository to browse, download, or share models
- **Spaces**
  - A free platform to deploy ML apps using Gradio or Streamlit.

Key Libraries from Hugging Face

- transformers
- datasets
- tokenizers
- evaluate
- accelerate
- PEFT

NVIDIA CUDA

- Compute Unified Device Architecture
- Parallel computing platform and **API**
- allows to use GPUs for general-purpose computing.
- Faster computations , Better Performance
- PyTorch and TensorFlow support CUDA

General Errors when we use CUDA

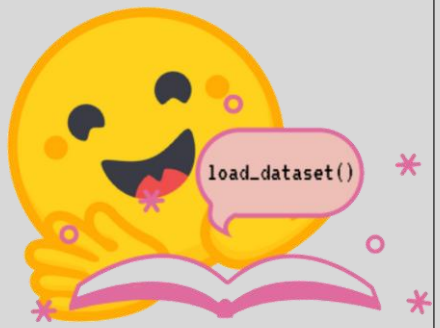
- Expected all tensors to be on the same device
- CUDA out of memory
- RuntimeError: CUDA error: device-side assert triggered
- RuntimeError: cuDNN error: CUDNN_STATUS_EXECUTION_FAILED
- CUDA device not found

Always define a device at the top of your script:

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

- Use .to(device) for everything: models, inputs, and targets

load_dataset()



- This method is part of the Hugging Face datasets library
- used to load datasets either from the Hugging Face Hub or from a local/custom source.
- Basic Syntax

```
from datasets import load_dataset
dataset = load_dataset("dataset_name")
```

- We can also load specific splits

```
dataset = load_dataset("ag_news", split="train")
```

- loading from a csv file

```
dataset = load_dataset("csv", data_files="data.csv")
dataset = load_dataset("csv", data_files={"train": "train.csv", "test": "test.csv"})
```

- It loads
  - A DatasetDict object with splits like "train", "test", and sometimes "validation".
  - Each split is a Dataset object that behaves like a list of dictionaries.

Load Tokenizer

- AutoTokenizer is a generic class that automatically loads the correct tokenizer for any pretrained model – like BERT, RoBERTa, T5, etc.
- It figures out whether to load BertTokenizer, RobertaTokenizer, etc., based on the model name.

```
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
```

- Example

```
text = "Hugging Face makes NLP easy!"
inputs = tokenizer(text, padding=True, truncation=True, return_tensors="pt")
print(inputs['input_ids'])
print(inputs['attention_mask'])
```

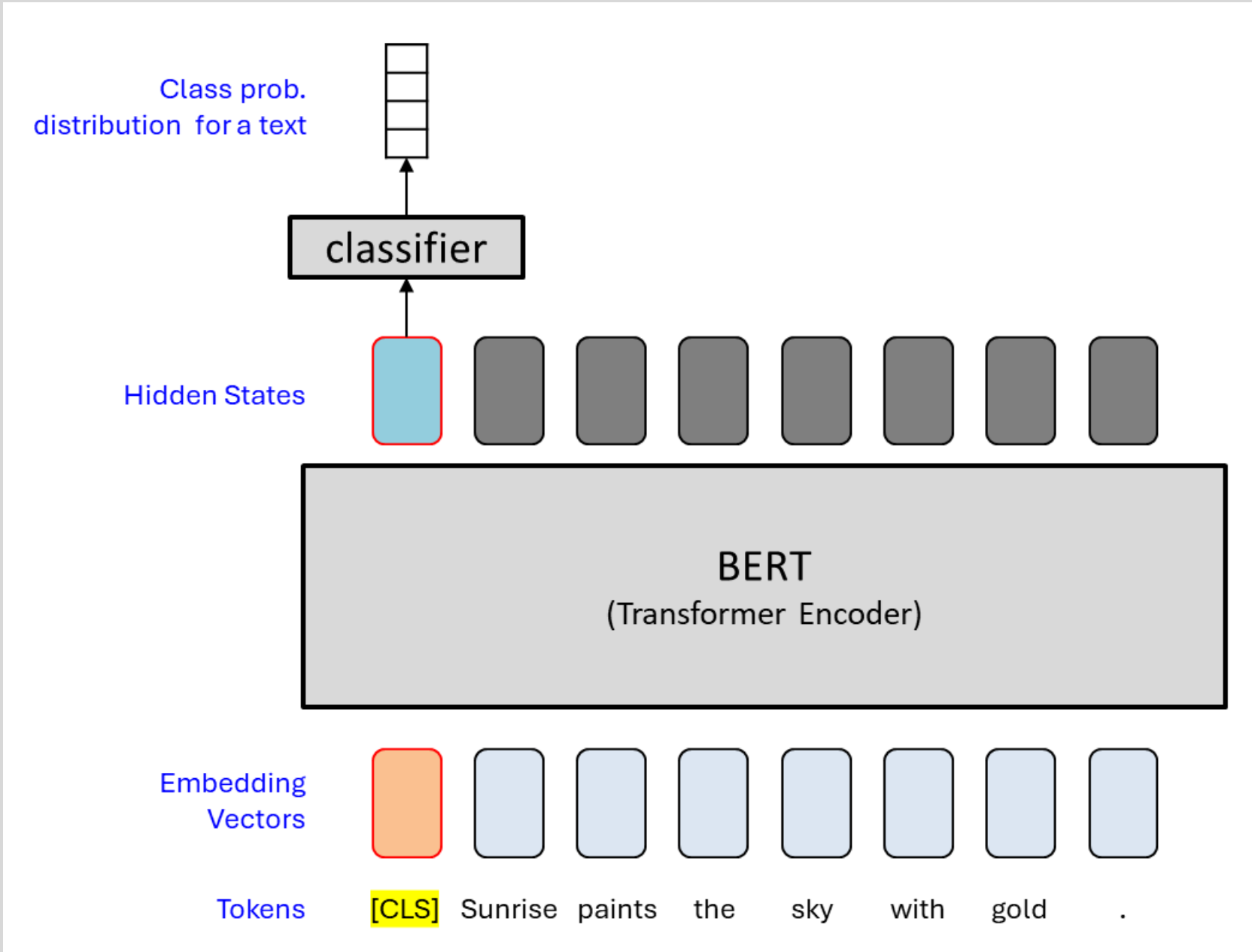
- input_ids: Numerical token IDs
- attention_mask: Tells model which tokens to attend to (1 = real token, 0 = padding)

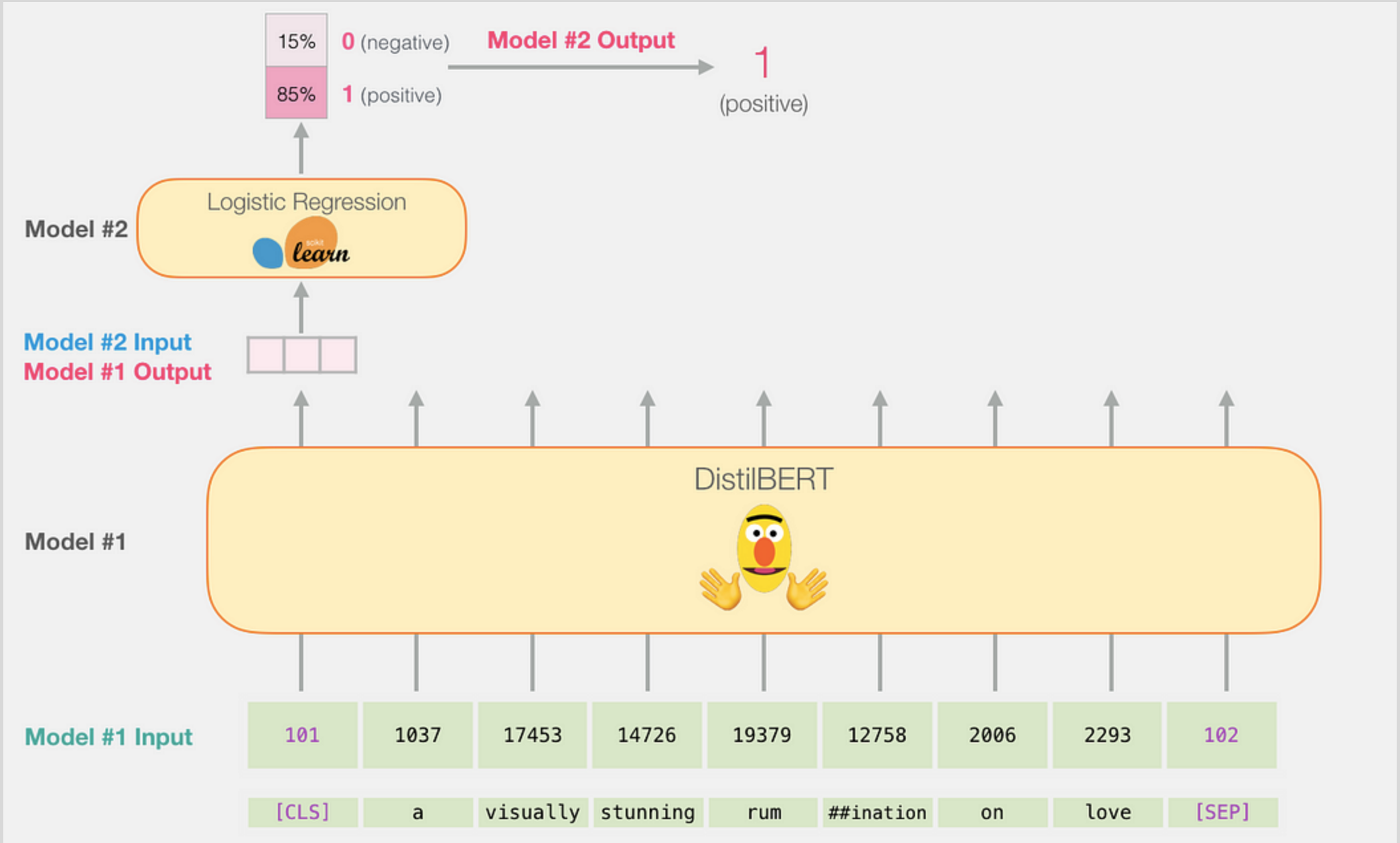
DataLoader

- Wraps a dataset.
- Helps in batching, shuffling, and loading data efficiently during training.

```
from torch.utils.data import DataLoader
train_loader = DataLoader(dataset, batch_size=16, shuffle=True)
```

```
from transformers import AutoTokenizer
from datasets import load_dataset
from torch.utils.data import DataLoader

dataset = load_dataset("imdb", split="train[:5000]")                    # Load dataset

tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")     # Load tokenizer

# Tokenize dataset
def tokenize_fn(batch):
    return tokenizer(batch["text"], padding="max_length", truncation=True)

tokenized_dataset = dataset.map(tokenize_fn, batched=True)

tokenized_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])     # Set format for PyTorch

train_loader = DataLoader(tokenized_dataset, batch_size=16, shuffle=True)                    # Create DataLoader
```


Class prob.
distribution for a text
classifier
Hidden States
BERT
(Transformer Encoder)
Embedding
Vectors
Tokens
[CLS]
Sunrise
paints
the
sky
with
gold
.


15%
0 (negative)
85%
1 (positive)
Model #2 Output
1
(positive)
Model #2
Logistic Regression
learn
Model #2 Input
Model #1 Output
Model #1
DistilBERT
Model #1 Input
101
1037
17453
14726
19379
12758
2006
2293
102
[CLS]
a
visually
stunning
rum
##ination
on
love
[SEP]

BertForSequenceClassification

- If you are building a model to classify news into 4 categories:

```
MODEL_NAME = 'bert-base-uncased'
num_classes = 4
model = BertForSequenceClassification.from_pretrained(MODEL_NAME, num_labels=num_classes)
```

- BERT will be loaded with a final linear layer like:

```
nn.Linear(hidden_size, 4)
```

Examine the Fine_Tuning___Emotion_Detection.ipynb

- What dataset is being used in this code?
- How many emotion classes are there in this dataset?
- Which model is being fine-tuned here?
- Why do we move the batch to the device using .to(device)?
- What is the purpose of loss.backward()?
- What happens during optimizer.step()?
- Why do we call optimizer.zero_grad() after each step?
- Why is a learning rate scheduler used?
- What does torch.argmax(outputs.logits, dim=-1) do?
- What will happen if you forget to rename 'label' to 'labels'?

Objective – 3

Domain Adaptation using Textbook Corpora

Importance of Domain-specific Models

Methodology

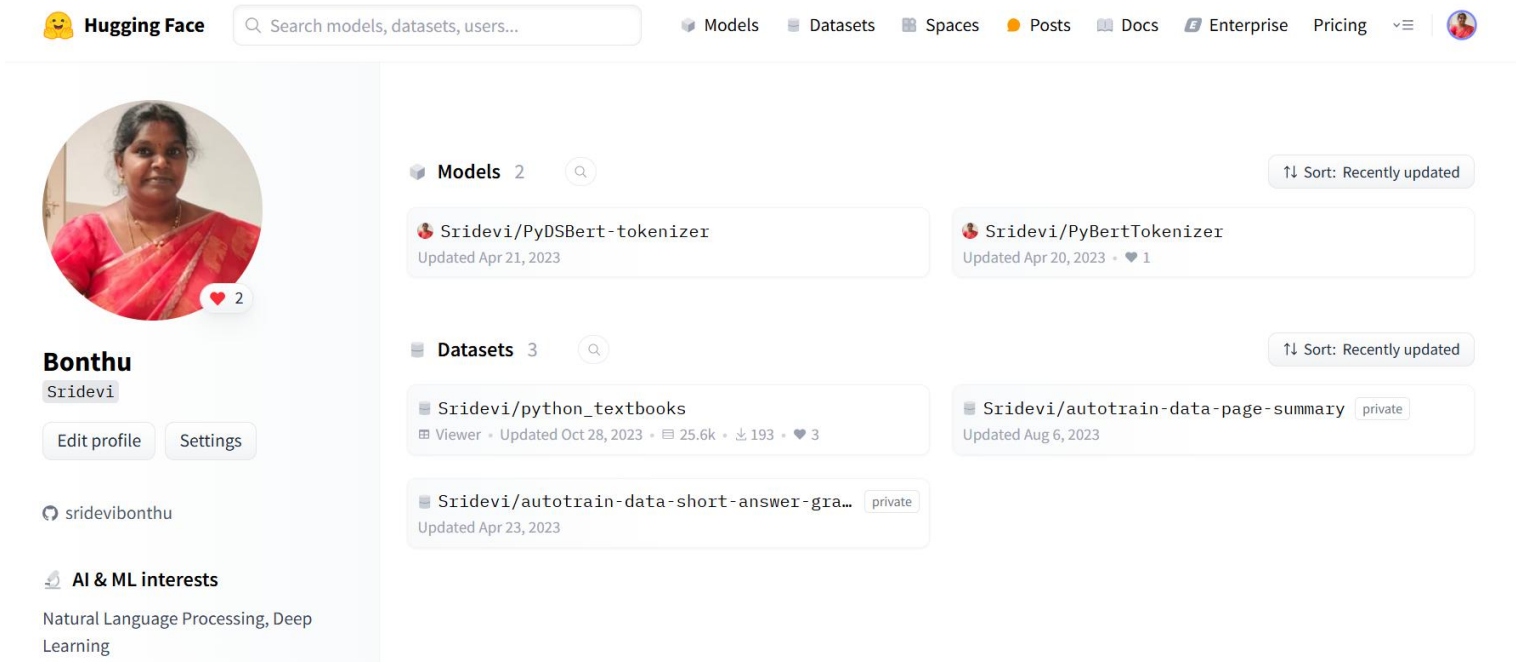
Domain-specific Corpus

Tokenizers

Domain Adaptation

Results

Summary of the findings



Bonthu, Sridevi, S. Rama Sree, and MHM Krishna Prasad. "**Framework for automation of short answer grading based on domain-specific pre-training.** Engineering Applications of Artificial Intelligence 137 (2024): 109163. https://doi.org/10.1016/j.engappai.2024.109163

Effectiveness of Fine-tuned BERT Model in Cl of Helpful and Unhelpful Online Customer Re

Muhammad Bilal[1] · Abdulwahab Ali Almazroi[2]

Accepted: 5 April 2022 / Published online: 29 April 2022
© The Author(s), under exclusive licence to Springer Science+Business Medi 2022

Abstract

The problem of information overload in online review pla pered many customers' ability to evaluate the quality when making purchasing decisions. A large body of lite to predict the helpfulness of online customer reviews ar tory findings on the effectiveness of various approaches solutions use traditional machine learning techniques and iting generalization. Therefore, this study aims to propose fine-tuning the BERT (Bidirectional Encoder Represent base model. The performance of BERT-based classifiers of bag-of-words approaches to determine the effectiven fiers. The evaluations performed using Yelp shopping re BERT-based classifiers outperform bag-of-words approa and unhelpful reviews. In addition, it is found that the seq based classifier has a significant impact on classification p

Keywords Electronic Word-Of-Mouth · Online Reviews · Sequence Classification · Bag-of-Words · Deep Learning

BERT-ERC: Fine-Tuning BERT Is Enough for Emotion Recognition in Conversation

Xiangyu Qin[1,2*], Zhiyu Wu[1*], Tingting Zhang[1], Yanran Li[2], Jian Luan[2†], Bin Wang[2], Li Wang[3], Jinshi Cui[1‡],

[1]School of Intelligence Science and Technology, Peking University
[2]Xiaomi AI Lab
[3]School of Psychological and Cognitive Sciences and Beijing Key Laboratory of Behavior and Mental Health, Peking University
2001213087@stu.pku.edu.cn, wuzhiyu@pku.edu.cn, zhangtingting3412@gmail.com, yanranli.summer@gmail.com, luanjian78@hotmail.com, wangbin11@xiaomi.com, liwang@pku.edu.cn, cjs@cis.pku.edu.cn,

Abstract

Previous works on emotion recognition in conversation (ERC) follow a two-step paradigm, which can be summarized as first producing context-independent features via fine-tuning pretrained language models (PLMs) and then analyzing contextual information and dialogue structure information among the extracted features. However, we discover that this paradigm has several limitations. Accordingly, we propose a novel paradigm, i.e., exploring contextual information and dialogue structure information in the fine-tuning step, and adapting the PLM to the ERC task in terms of input text, classification structure, and training strategy. Furthermore, we develop our model BERT-ERC according to the proposed paradigm, which improves ERC performance in three aspects, namely suggestive text, fine-grained classification module, and two-stage training. Compared to existing methods, BERT-ERC achieves substantial improvement on four datasets, indicating its effectiveness and generalization capability. Besides, we also set up the limited resources scenario and the online prediction scenario to approximate real-world scenarios. Extensive experiments demonstrate that the proposed paradigm significantly outperforms the previous

Method		MELD
PLM	classifier	
RoBERTa-large	RGAT	62.80
	DialogGCN	63.02
	DAGNN	63.12
	DialogRNN	63.61
	DAG-ERC	63.65
RoBERTa-large	MLP	63.39

Table 1: Pilot experiment on MELD (%).

structure information consisting of non-textual information of the conversation, such as the speaker information, the emotion states, and the relative position of the utterances. To exploit these three kinds of information, previous works (Ghosal et al. 2019b; Majumder et al. 2019; Ishiwatari et al. 2020; Shen et al. 2021) commonly follow a two-step paradigm of first extracting context-independent features via fine-tuning pretrained language models (PLMs)

Dataset Distillation with Attention Labels for Fine-tuning BERT

Aru Maekawa[a], Naoki Kobayashi[b], Kotaro Funakoshi[a] and Manabu Okumur

Dataset distillation aims to create a small dataset of informative synthetic sam
to rapidly train neural networks that retain the performance of the original data
In this study, we focus on constructing distilled few-shot datasets for natural
guage processing (NLP) tasks to fine-tune pre-trained transformers. Specifically,
propose introducing attention labels, which can efficiently distill knowledge from
original dataset and transfer it to transformer models via attention probabilities.
evaluated our dataset distillation methods in four NLP tasks and demonstrated t
it is possible to create distilled few-shot datasets with attention labels, yielding
impressive performance for fine-tuning BERT. Specifically, in AGNews, which
four-class news classification task, our distilled few-shot dataset achieved up to 93
accuracy, which is 98.5% that of the original dataset, even with only one sample
class and only one gradient step.

Key Words: *Dataset Distillation, Dataset Condensation, Attention Labels*

Patterns

CellPress
OPEN ACCESS

Article

Fine-tuning large neural language models for biomedical natural language processing

Robert Tinn,[1,2] Hao Cheng,[1,2] Yu Gu,[1] Naoto Usuyama,[1] Xiaodong Liu,[1] Tristan Naumann,[1] Jianfeng Gao,[1] and Hoifung Poon[1,3,*]
[1]Microsoft Research, Redmond, WA, USA
[2]These authors contributed equally
[3]Lead contact
*Correspondence: hoifung@microsoft.com
https://doi.org/10.1016/j.patter.2023.100729

THE BIGGER PICTURE Large neural language models have transformed modern natural language processing (NLP) and have recently become a focus of public attention. However, fine-tuning these models for specific tasks of interest remains challenging as model size increases, especially with small labeled datasets, which are common in biomedical NLP.

This study conducts a systematic exploration of fine-tuning stability in biomedical NLP and identifies techniques that address instability and improve performance. The findings highlight the importance of domain-specific vocabulary and pretraining for creating robust models and establish a new state of the art on a wide range of biomedical NLP applications in the Biomedical Language Understanding and Reasoning Benchmark (BLURB).

Proof-of-Concept: Data science output has been formulated, implemented, and tested for one domain/problem