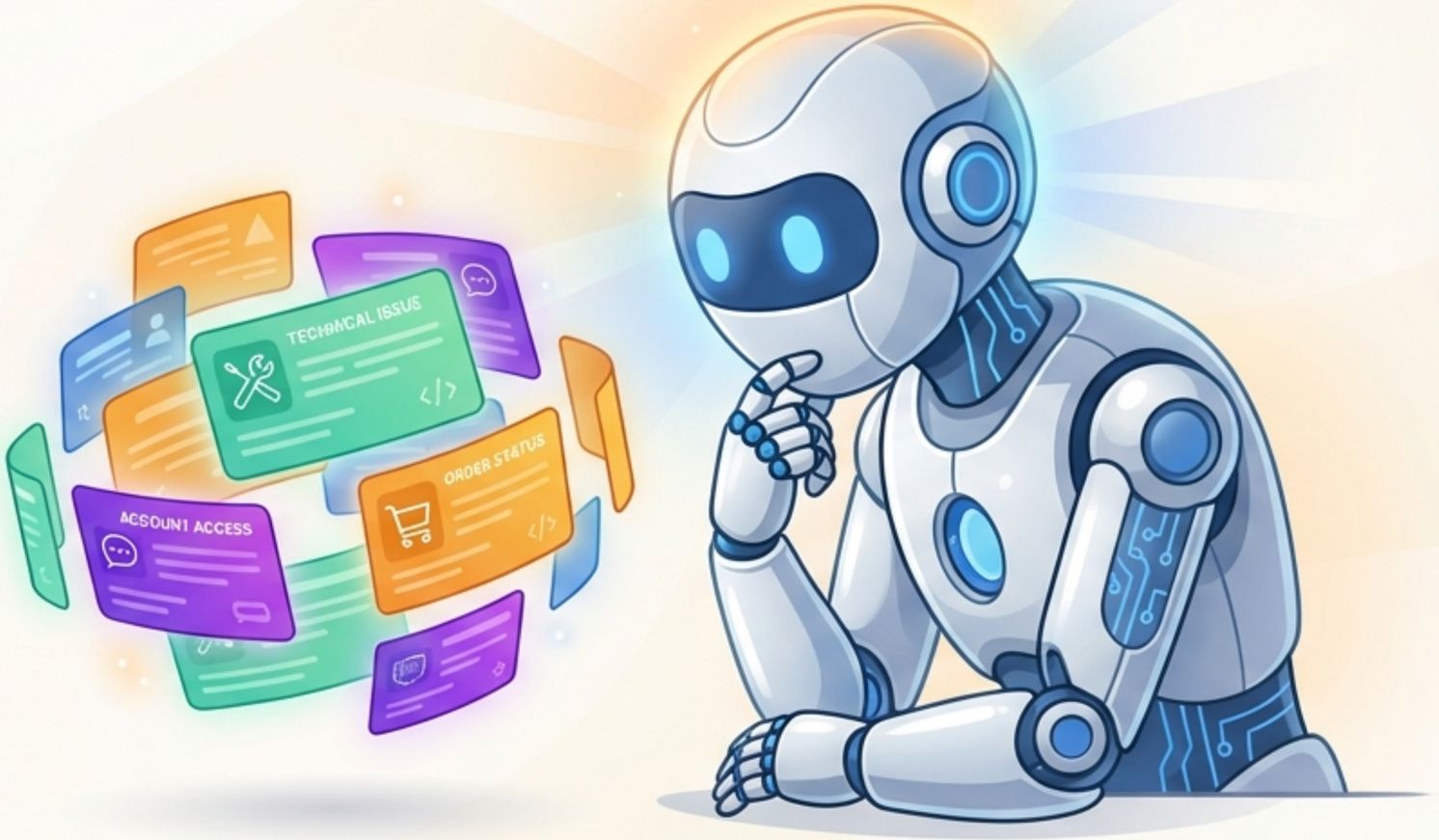
Teaching an AI to Understand Our Customers

The journey of training a language expert to sort support tickets, step by step.



The Challenge: A Flood of Customer Questions

Every day, we receive dozens of customer support tickets. They cover everything from billing issues and technical problems to sales questions.

Sorting them manually is slow and inefficient.

Our mission is to automate this process.

Technical Support

Billing

Sales

Our Raw Material: The Customer's Voice

We begin with a collection of 50 real (but anonymized) customer tickets. Each one is a plain text sentence, and for our training, we've already manually categorized them. This is the textbook our apprentice will learn from.


```
texts = [
    "My internet connection drops every few minutes",
    "My payment failed but money was deducted",
    "How can I upgrade my subscription plan?",
    "The app crashes when I try to log in",
    "I need a refund for my last payment",
    "Is there a discount for annual billing?",
    # ... (more examples) ...
]
```

Assembling Our Specialist Team

We don’t build this from scratch. We assemble a team of world-class specialists—powerful code libraries designed for data science and AI. Each has a specific role to play in educating our apprentice.

Hugging Face Transformers

The AI School - Provides the Apprentice & Tutor

Scikit-learn

The Organizer - Helps prepare the curriculum

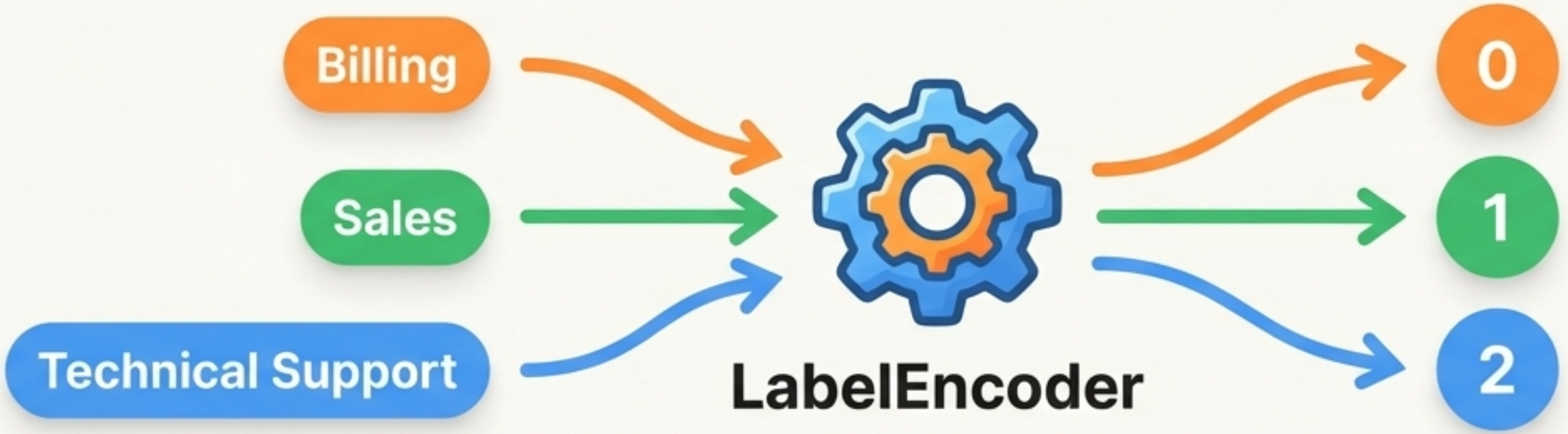
PyTorch & Datasets

The Foundation - The infrastructure and language of learning

```
from datasets import Dataset
from transformers import BertTokenizer, BertForSequenceClassification, TrainingArguments, Trainer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, f1_score
import torch
```

A First Lesson: Translating Categories into Code

Our apprentice thinks in numbers, not words. Our first task is to translate the ticket categories—'Billing', 'Sales', 'Technical Support'—into a simple numerical code. We use a tool called a LabelEncoder to do this automatically.

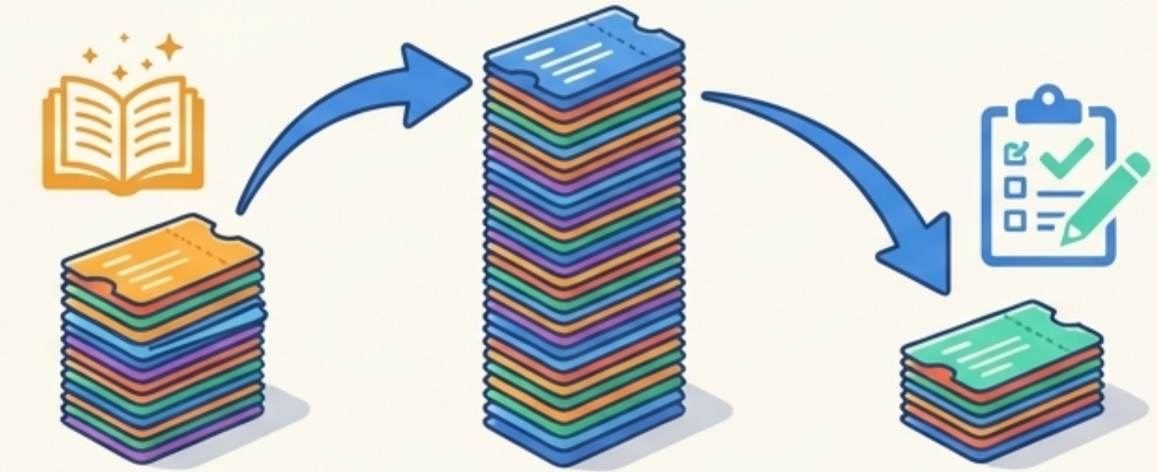



```
label_encoder = LabelEncoder() \
labels_encoded = label_encoder.fit_transform(labels)
```

Creating a Curriculum and a Practice Exam

To teach effectively, we can't show the apprentice all the answers at once. We split our 50 tickets into two piles:

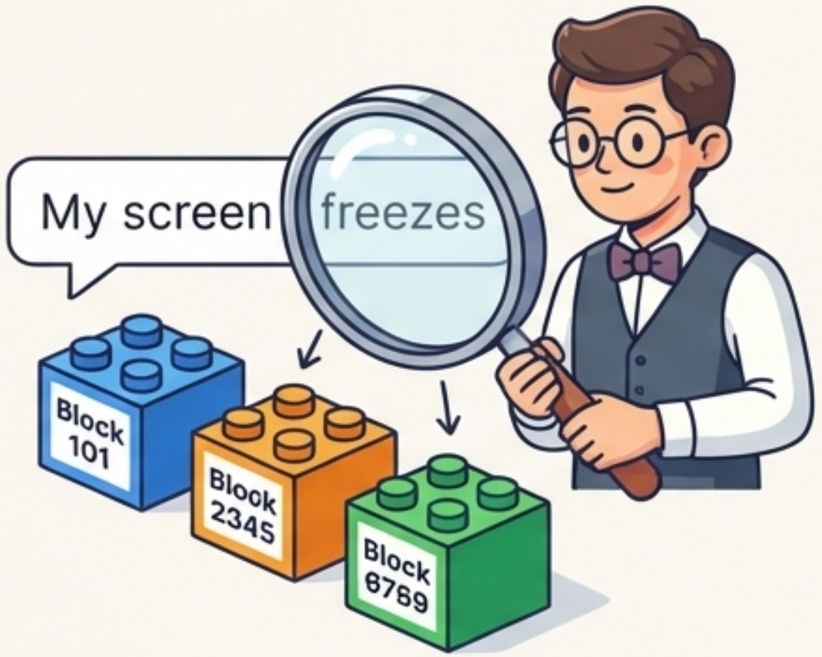
- **Training Set (40 tickets)**: The main textbook for learning.
- **Validation Set (10 tickets)**: A practice exam to check its understanding along the way.




```
X_train, X_val, y_train, y_val = train_test_split(
    texts,
    labels_encoded,
    test_size=0.2,
    stratify=labels_encoded,
    random_state=42
)
```

Hiring the Translator

Just like with categories, our apprentice doesn't read English sentences. It needs a special translator, called a **Tokenizer**, to break down each ticket into a sequence of numbers (tokens) it can understand. Each number represents a word or a piece of a word.




```
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
```

The Great Translation: From Words to Numbers

The Translator now processes our entire curriculum. It reads every ticket in our training and validation sets and converts them into the special numerical format the apprentice requires. This process also standardizes the length of every ticket to ensure consistency.

Before

My screen freezes

I cannot connect

Password reset issue

.map(tokenize)

After

Ticket 1

input_ids	101	2023	3423	13642	102	0	0	0
attention_mask	1	1	1	1	1	0	0	0

Ticket 2

input_ids	101	1045	3685	4568	102	0	0	0
attention_mask	1	1	1	1	1	0	0	0

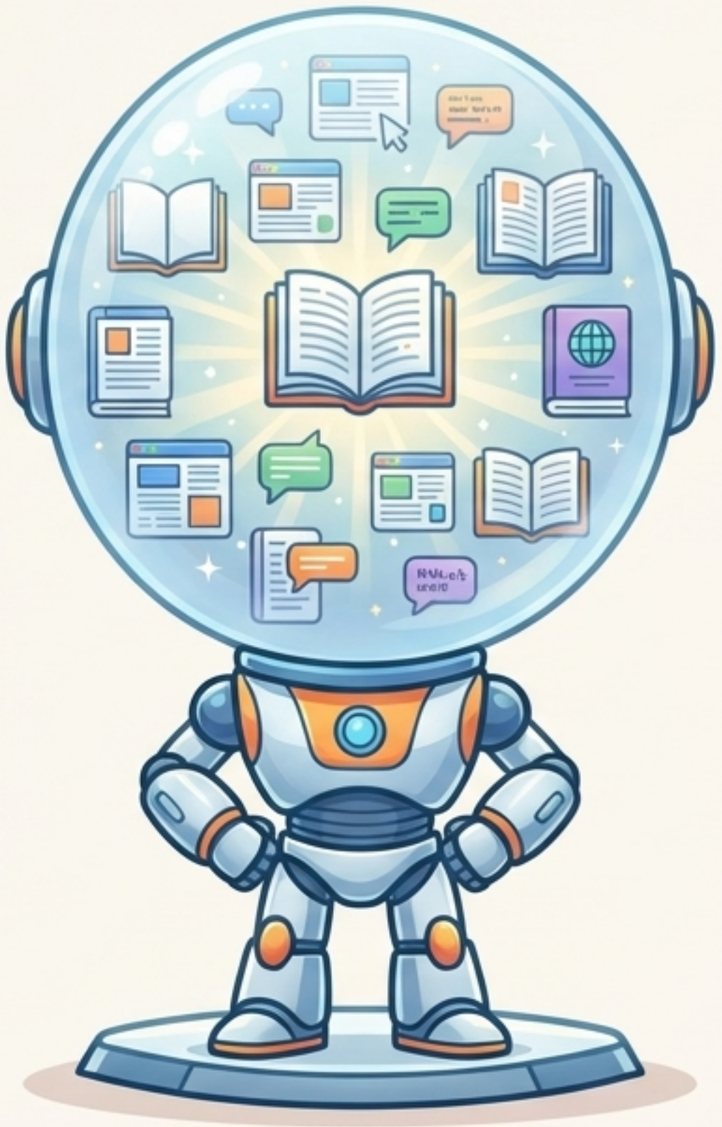
Ticket 3

input_ids	101	9853	5146	4354	102	0	0	0
attention_mask	1	1	1	1	1	0	0	0

Meet the Apprentice: A Genius Linguist

We don’t need to teach our AI English from scratch. We hire BERT, a world-class model that has already been “pre-trained” on a massive library of books and websites. It understands grammar, context, and nuance. Our job is simply to fine-tune its general knowledge for our specific task.

```
model = BertForSequenceClassification.
    from_pretrained("bert-base-uncased",
    num_labels=3)
```



Setting the Rules for School

Every good training program needs structure. We provide two things:

1. **The Grading Rules:** We define metrics like `accuracy` and `f1_score` to measure how well it's learning.
2. **The Lesson Plan:** We set arguments like learning rate and batch size, which control the pace and intensity of the study session.

Grading Rules
(`compute_metrics`)

Lesson Plan
(`TrainingArguments`)

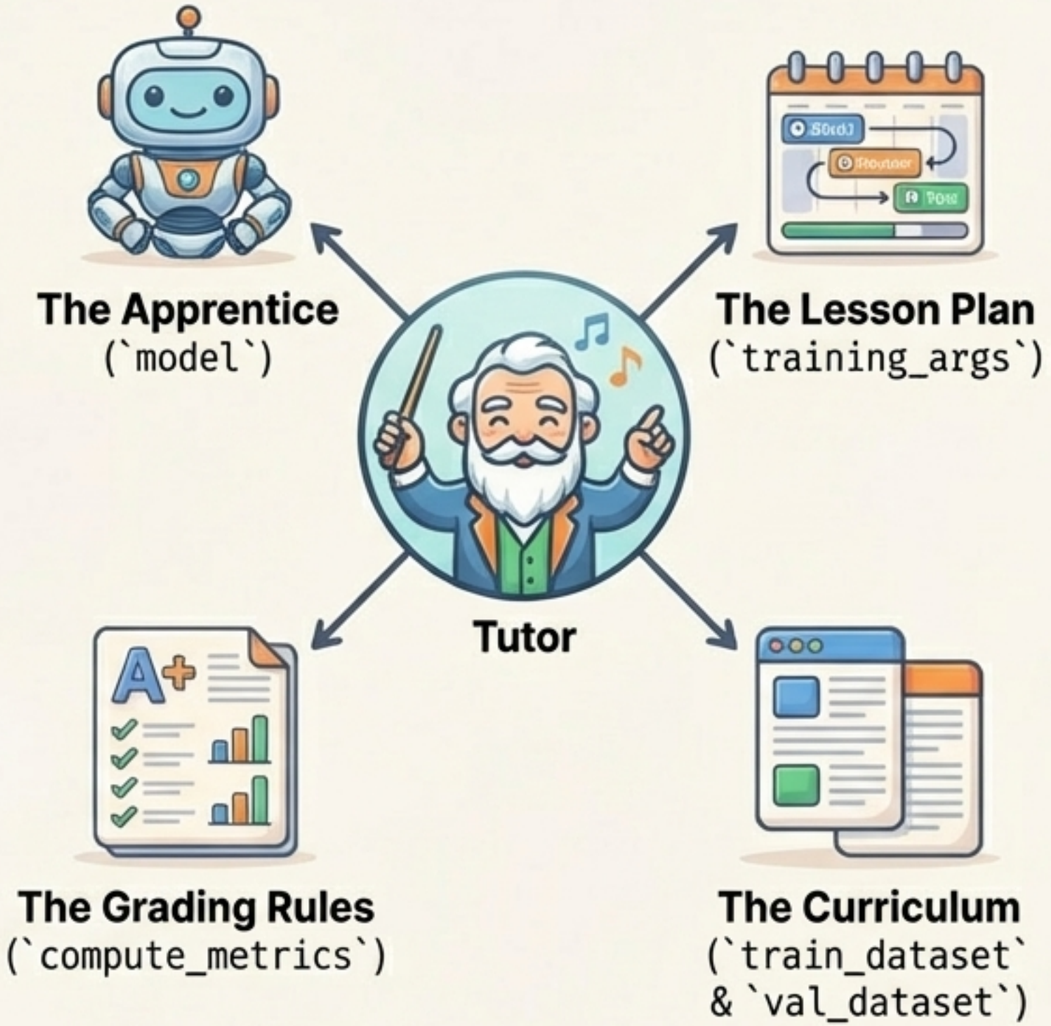
Bringing in the Tutor

The `Trainer` is the expert tutor who manages the entire education process. It brings everything together:

- The Apprentice (our `model`)
- The Lesson Plan (`training_args`)
- The Curriculum (`train_dataset` & `val_dataset`)
- The Grading Rules (`compute_metrics`)

The Tutor will guide the apprentice, give it the material to study, and test its knowledge.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```



The Study Session Begins

With a single command—`trainer.train()`—the learning begins. The Tutor guides the apprentice through the training data multiple times (called “epochs”). In each pass, the apprentice makes predictions, compares them to the correct answers, and adjusts its internal wiring to get better.

Graduation Day: A Graduation: A New Specialist is Ready

The training is complete! Our apprentice, once a general linguist, is now a **specialist** in classifying our customer support tickets.

It has learned the specific patterns and vocabulary related to our business needs.

Now, it's time to put it to the test on a new, unseen ticket.

The First Day on the Job

We give our newly trained specialist a ticket it has never seen before: ***“My card was billed twice for the same order”***. The model processes the text and instantly makes its prediction.


```
test_text = "My card was billed twice for the same order"
predict_ticket(test_text)
```

From Raw Text to Automated Insight

The prediction is **Technical Support**. In seconds, our system has read, understood, and categorized a new customer query.

We have successfully transformed a pile of unstructured data into an intelligent tool that can work for us. This is the power of fine-tuning.