Practical Exercise — Lesson 1: “Turn Scores into Decisions

Overview

In this practical, you will turn raw model outputs into decisions that stakeholders can trust. You will start with a simple text classifier and compute core metrics—accuracy, precision, recall, F1—reading the confusion matrix as a diagnostic tool rather than a scoreboard. You will then align your **primary metric** to the harm you most want to avoid and, if scores are available, select a deployment threshold using a precision–recall curve. Next, you will practice **boundary-aware evaluation** for two structure-sensitive tasks: entity-level F1 for NER and EM/Span-F1 for extractive QA, explaining what partial matches reveal about your model’s behavior. You will close by sketching a lightweight “extrinsic” success metric (e.g., minutes of moderator time saved) and a guardrail you would monitor in a live setting.

By the end, you should be able to compute the right metrics, interpret them by slice, justify a threshold, and articulate a clear recommendation—what to optimize first, what to fix next, and how you will know the change truly helps users.

Tools: Google Colab, Python, `datasets`, `transformers`, `evaluate`/`scikit-learn`
What you will produce: a short metrics report (tables + 1–2 charts) and a one-page decision memo.

Learning outcomes

By the end, you will be able to:

1. Compute and interpret **accuracy, precision, recall, F1** (micro/macro/weighted) with a confusion matrix.
2. Align the **primary metric** to the harm you want to avoid and defend that choice.
3. Evaluate **boundary-sensitive** tasks using **entity-level F1** (NER) and **EM/Span-F1** (extractive QA).
4. Make an actionable recommendation (threshold choice, next fix, or ship/no-ship).

Part A — Classification metrics in practice (≈30 min)

Data (choose one):

- Sentiment: `glue/sst2` (dev set) or a 5–10k subset of IMDb.
  Model: `text-classification` pipeline with any BERT-like checkpoint.

Tasks

1. **Baseline run.** Score the dev set; save for each item: gold label, predicted label, predicted score.
2. **Core metrics.** Compute overall **accuracy, precision, recall, F1** (micro/macro/weighted). Show a **confusion matrix**.
3. **Slice analysis.** Create two non-sensitive slices (e.g., **short vs. long texts**, **neutral words count high/low**). Report P/R/F1 per slice.
4. **Threshold choice (if model gives scores).** Plot **precision–recall curve** for the positive class. Pick a threshold that optimizes your **primary metric** (see Decision prompt below).
5. **Decision prompt (2–3 sentences).** Given (a) a cost of false positives vs false negatives you specify, and (b) your slices, **state which metric you optimize first**, what threshold you choose, and why.

Expected artifacts

- Table: overall and per-slice P/R/F1 (+ confusion matrix image).
- One chart: PR curve with your chosen threshold marked.
- 2–3 sentence justification tying metric choice to user harm/business risk.

Part B — Boundary-aware evaluation (NER) (≈15 min)

Data: 100–200 sentences from `conll2003` (validation).
Model: `token-classification` pipeline.

Tasks

1. Run predictions; compute **token-level accuracy** (for context only) and **entity-level P/R/F1** (primary).
2. **Boundary error drill.** List the top 5 boundary/type mistakes. For **one** case, paste the sentence and show **gold vs. predicted span/type**.
3. **One-line takeaway:** what annotation or modeling tweak would most reduce boundary errors?

Expected artifacts

- Table: entity-level P/R/F1 and 5 example errors with a one-line fix each.

Part C — Extractive QA scoring (EM & Span-F1) (≈15 min)

Data: 50–100 examples from `squad` (validation).
Model: `question-answering` pipeline.

Tasks

1. Score the subset; compute **Exact Match (EM)** and **Span-F1**.
2. Show **3 examples** where EM=0 but Span-F1>0; write a one-line interpretation of what the model tends to over/under-include (e.g., titles, prefixes).
3. Propose **one decoding or post-processing tweak** (e.g., strip titles, cap answer length) and **predict** qualitatively how it would move EM/Span-F1.

Expected artifacts

- Table: EM, Span-F1 + three illustrative examples with your interpretation.

Part D — Mini "extrinsic" bridge (design only; no coding) (≈5–10 min)

Task
In 4–5 sentences, define a **real-world success metric** for deploying this model (e.g., "reduced moderator review minutes," "fewer payment delays"). Describe:

- the unit of measurement,
- how you would run an **A/B or pre/post** check,
- how long you need to detect a meaningful change,
- the **guardrail** you would monitor (e.g., max acceptable FPR gap on a critical slice).

Deliverables (upload to Blackboard)

1. **Metrics notebook** (Colab):

- A. Classification tables + confusion matrix + PR curve with chosen threshold.

- B. NER entity-level metrics + error list.
- C. QA EM/Span-F1 + three “partial-match” examples.

2. **One-page decision memo** (≈250–350 words):

- **Primary metric** you optimized and why.
- **Threshold** (if applicable) and trade-offs.
- **Top error pattern** you’ll fix next and the concrete action (data, labeling, decoding).
- The **extrinsic metric** you propose for a small live test and the guardrail you would enforce.

Starter code hints (optional)

```
from datasets import load_dataset
from transformers import pipeline
from sklearn.metrics import classification_report, confusion_matrix,
precision_recall_curve
import numpy as np, pandas as pd

# A. Classification — SST-2
ds = load_dataset("glue", "sst2")
clf = pipeline("text-classification", model="distilbert-base-uncased-
finetuned-sst-2-english", return_all_scores=True)
preds = [clf(x["sentence"])[0] for x in ds["validation"]]
# Convert to labels/scores, then compute metrics & PR curve

# B. NER — CoNLL 2003 (validation)
ner_ds = load_dataset("conll2003")["validation"]
ner = pipeline("token-classification", aggregation_strategy="simple")
# Run on a small slice, then compute entity-level P/R/F1

# C. QA — SQuAD (validation)
qa_ds = load_dataset("squad")["validation"].select(range(100))
qa = pipeline("question-answering")
# For each example: qa(question=..., context=...), then EM & Span-F1
```