

## Assignment 1

```
# === IMPORTS AND SETUP ===
import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datasets import load_dataset
import transformers
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
import evaluate
from sklearn.metrics import confusion_matrix, classification_report
```

```
# === EVALUATION METRICS DEFINITION ===
# Load accuracy and F1 metrics from evaluate library
acc_metric = evaluate.load("accuracy")
f1_metric = evaluate.load("f1")

def compute_metrics(eval_pred):
    """Computes accuracy, macro F1, and weighted F1."""
    logits, labels = eval_pred.predictions, eval_pred.label_ids
    preds = np.argmax(logits, axis=1)

    # Calculate metrics
    acc = acc_metric.compute(predictions=preds, references=labels)["accuracy"]
    f1_macro = f1_metric.compute(predictions=preds, references=labels, average="macro")["f1"]
    f1_weighted = f1_metric.compute(predictions=preds, references=labels, average="weighted")["f1"]
    return {"accuracy": acc, "f1_macro": f1_macro, "f1_weighted": f1_weighted}
```

Show hidden output

```
# === DATA LOADING AND PREPARATION ===
DATASET_NAME = "ag_news"
raw = load_dataset(DATASET_NAME)

# Split the training data to create a validation set
split = raw["train"].train_test_split(test_size=0.08, seed=42)
train_ds = split["train"]
val_ds = split["test"]
test_ds = raw["test"]

# Get label names and number of labels
label_names = train_ds.features["label"].names
num_labels = len(label_names)
```

Show hidden output

```
# === TOKENIZATION ===
def tokenize_datasets(model_ckpt, train_ds, val_ds, test_ds, max_length=128):
    """Loads tokenizer and tokenizes all datasets."""
    tokenizer = AutoTokenizer.from_pretrained(model_ckpt, use_fast=True)
    def _tok(batch):
        """Tokenization helper function."""
        return tokenizer(batch["text"], truncation=True, padding='max_length', max_length=max_length)

    # Map tokenization function across all datasets
    t_train = train_ds.map(_tok, batched=True, batch_size=1000)
    t_val = val_ds.map(_tok, batched=True, batch_size=1000)
    t_test = test_ds.map(_tok, batched=True, batch_size=1000)

    # Set format to torch
    t_train.set_format(type="torch", columns=tokenizer.model_input_names + ["label"])
    t_val.set_format(type="torch", columns=tokenizer.model_input_names + ["label"])
    t_test.set_format(type="torch", columns=tokenizer.model_input_names + ["label"])
    return tokenizer, t_train, t_val, t_test
```

```
# === FINE-TUNING FUNCTION ===
def fine_tune_resume(model_ckpt, outdir, tokenizer, train_dataset, val_dataset, test_dataset,
                     num_labels, epochs, per_device_batch, lr, resume_from_checkpoint=None):
    """Initializes and fine-tunes a model."""

    # Load model for sequence classification
    model = AutoModelForSequenceClassification.from_pretrained(model_ckpt, num_labels=num_labels)

    # Define training arguments
```

```

    training_args = TrainingArguments(
        output_dir=outdir,
        eval_strategy="epoch",
        save_strategy="epoch",
        save_total_limit=5,
        load_best_model_at_end=True,
        metric_for_best_model="eval_accuracy",
        per_device_train_batch_size=per_device_batch,
        per_device_eval_batch_size=per_device_batch,
        learning_rate=lr,
        weight_decay=0.01,
        num_train_epochs=epochs,
        fp16=True,
        logging_dir=f"{outdir}/logs",
        report_to="none"
    )

    # Initialize Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        compute_metrics=compute_metrics
    )

    # Check if resuming from checkpoint
    if resume_from_checkpoint:
        print(f"Resuming training from checkpoint: {resume_from_checkpoint}")
    else:
        print("Starting training from scratch")

    # Start training
    trainer.train(resume_from_checkpoint=resume_from_checkpoint)

    # Evaluate on the test set
    pred = trainer.predict(test_dataset)
    print("Final Test metrics:", pred.metrics)

    return trainer, model, pred

```

```

# === ANALYSIS FUNCTIONS ===
def plot_confusion_matrix(predictions, labels, label_names, title="Confusion Matrix"):
    cm = confusion_matrix(labels, predictions)
    plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt="d", xticklabels=label_names, yticklabels=label_names, cmap="Blues")
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.title(title)
    plt.show()

def error_analysis(preds, labels, dataset, text_column="text", label_names=None, n_samples=10):
    import random

    errors = []

    # Text
    if text_column in dataset.column_names:
        texts = dataset[text_column]
    else:
        texts = [f"No {text_column}, sample #{i}" for i in range(len(dataset))]

    # Iterate over predictions and labels
    for i, (p, l) in enumerate(zip(preds, labels)):
        if p != l:
            errors.append({
                "text": texts[i],
                "true_label": label_names[l] if label_names else l,
                "pred_label": label_names[p] if label_names else p
            })

    # Show random samples of errors
    sample_errors = random.sample(errors, min(n_samples, len(errors)))
    for e in sample_errors:
        print(f"Text: {e['text']}")
        print(f" True: {e['true_label']} | Pred: {e['pred_label']}")
        print("-" * 50)
    return errors

```

```

# === EXPERIMENT 1: DISTILBERT ===
# Define constants for DistilBERT

```

```

DISTIL_CKPT = "distilbert-base-uncased"
DISTIL_LR = 2e-5
DISTIL_BATCH = 32
DISTIL_EPOCHS = 3

# Tokenize datasets for DistilBERT
tokenizer_distil, t_train_distil, t_val_distil, t_test_distil = tokenize_datasets(
    DISTIL_CKPT, train_ds, val_ds, test_ds
)

# Run fine-tuning
trainer_distil, model_distil, pred_distil = fine_tune_resume(
    DISTIL_CKPT, "./checkpoints/distilbert", tokenizer_distil,
    t_train_distil, t_val_distil, t_test_distil, num_labels,
    epochs=DISTIL_EPOCHS, per_device_batch=DISTIL_BATCH, lr=DISTIL_LR,
    resume_from_checkpoint=None # First Training
)

# --- Task 1: DistilBERT Error Analysis ---
print("\n--- DistilBERT Analysis ---")
distil_preds = np.argmax(pred_distil.predictions, axis=1)
plot_confusion_matrix(distil_preds, pred_distil.label_ids, label_names, title="DistilBERT Confusion Matrix")

```

tokenizer_config.json: 100%	48.0/48.0 [00:00<00:00, 2.97kB/s]
config.json: 100%	483/483 [00:00<00:00, 30.1kB/s]
vocab.txt: 100%	232k/232k [00:00<00:00, 1.44MB/s]
tokenizer.json: 100%	466k/466k [00:00<00:00, 2.86MB/s]
Map: 100%	110400/110400 [00:43<00:00, 1639.14 examples/s]
Map: 100%	9600/9600 [00:03<00:00, 2662.38 examples/s]
Map: 100%	7600/7600 [00:04<00:00, 2099.08 examples/s]
model.safetensors: 100%	268M/268M [00:07<00:00, 35.6MB/s]

Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are newly initialized. You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

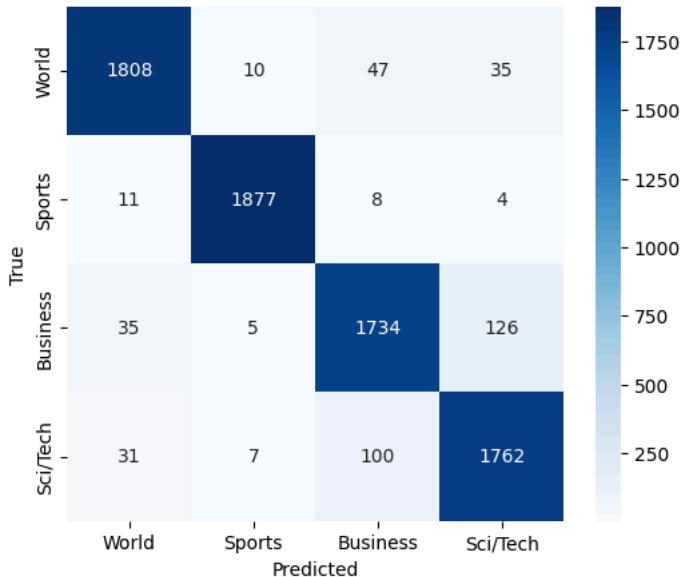
Starting training from scratch

[10350/10350 16:59, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro	F1 Weighted
1	0.187300	0.184443	0.938021	0.937609	0.937883
2	0.133700	0.178672	0.944375	0.944121	0.944379
3	0.090400	0.197675	0.944896	0.944653	0.944894

Final Test metrics: {'test\_loss': 0.18460433185100555, 'test\_accuracy': 0.9448684210526316, 'test\_f1\_macro': 0.9449029081176249, 'test\_f1\_weighted': 0.9449029081176249}

DistilBERT Confusion Matrix



```

# Perform qualitative error analysis
errors_distil = error_analysis(distil_preds, pred_distil.label_ids, test_ds, label_names=label_names)

```

Text: Press review 0850CET--The seizure of fake Nike sportswear by the Customs Department was one of the main stories on Wednesday #39;s news  
True: Sci/Tech | Pred: Sports

Text: UBS pays 265 million dollars for Schwab capital markets business (AFP) AFP - Swiss banking group UBS said that it had paid 265 million

True: Business | Pred: Sci/Tech

Text: Caterpillar snaps up another remanufacturer of engines PEORIA - Caterpillar Inc. said Wednesday it will acquire a South Carolina reman  
True: Sci/Tech | Pred: Business

Text: Bush Signs Internet Access Tax Ban State and local governments will be barred from taxing connections that link people to the Internet  
True: Sci/Tech | Pred: Business

Text: CA, Partners Move On As Kumar Faces Charges Concern over the fate of former Computer Associates International chairman and CEO Sanjay K  
True: Business | Pred: Sci/Tech

Text: Spread of GM grass raises fears of crossbreeding Pollen from a genetically modified grass was found 21 kilometres from where it was pla  
True: Business | Pred: Sci/Tech

Text: Bush Scraps Most U.S. Sanctions on Libya (Reuters) Reuters - President Bush on Monday formally\ended the U.S. trade embargo on Libya to  
True: Sci/Tech | Pred: World

Text: World briefs LONDON - A man wielding a machete and a knife attacked two security guards at the building housing the headquarters of the  
True: Business | Pred: World

Text: Iraq's Neighbors to Get Little for Environment Loss (Reuters) Reuters - Iraq's neighbors want tens of billions\of dollars for environment  
True: Sci/Tech | Pred: World

Text: Bryant Makes First Appearance at Trial (AP) AP - NBA star Kobe Bryant arrived at his sexual assault trial Monday as attorneys in the ca  
True: Sci/Tech | Pred: Sports

```
# === EXPERIMENT 2: MODERNBERT ===
# --- Task 2: Fine-tune ModernBERT ---
import torch._dynamo
torch._dynamo.config.suppress_errors = True

MODERN_CKPT = "answerdotai/ModernBERT-base"
MODERN_LR = 3e-5
MODERN_BATCH = 16
MODERN_EPOCHS = 3

# Tokenize datasets for ModernBERT
tokenizer_modern, t_train_modern, t_val_modern, t_test_modern = tokenize_datasets(
    MODERN_CKPT, train_ds, val_ds, test_ds
)

# Run fine-tuning
trainer_modern, model_modern, pred_modern = fine_tune_resume(
    MODERN_CKPT, "./checkpoints/modernbert", tokenizer_modern,
    t_train_modern, t_val_modern, t_test_modern, num_labels,
    epochs=MODERN_EPOCHS, per_device_batch=MODERN_BATCH, lr=MODERN_LR,
    resume_from_checkpoint=None # First Training
)

# --- Task 4: ModernBERT Error Analysis ---
print("\n--- ModernBERT Analysis ---")
modern_preds = np.argmax(pred_modern.predictions, axis=1)
plot_confusion_matrix(modern_preds, pred_modern.label_ids, label_names, title="ModernBERT Confusion Matrix")
```

```

tokenizer_config.json:      20.8k/? [00:00<00:00, 1.96MB/s]
tokenizer.json:      2.13M/? [00:00<00:00, 48.8MB/s]
special_tokens_map.json: 100%                                         694/694 [00:00<00:00, 82.2kB/s]
Map: 100%                                         110400/110400 [00:27<00:00, 3979.70 examples/s]
Map: 100%                                         9600/9600 [00:02<00:00, 4382.74 examples/s]
Map: 100%                                         7600/7600 [00:01<00:00, 4949.50 examples/s]
config.json:     1.19k/? [00:00<00:00, 91.6kB/s]
model.safetensors: 100%                                         599M/599M [00:13<00:00, 51.2MB/s]

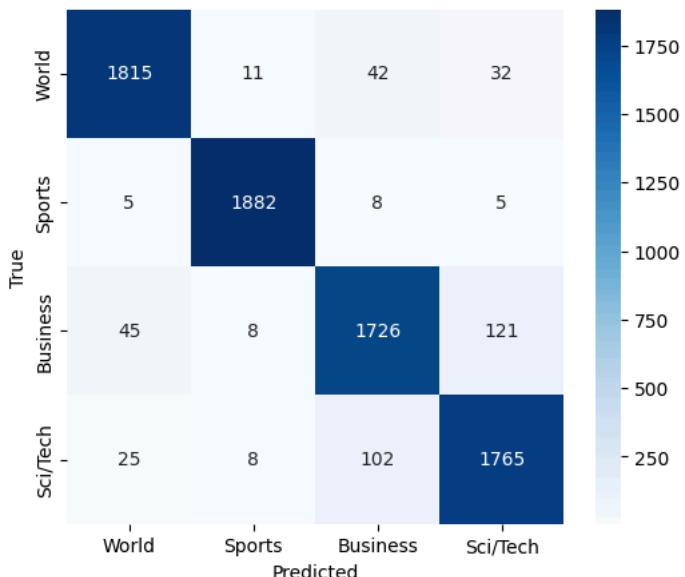
Some weights of ModernBertForSequenceClassification were not initialized from the model checkpoint at answerdotai/ModernBERT-base and are new.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Starting training from scratch
W0912 16:02:41.962000 880 torch/_inductor/utils.py:1436] [1/0_1] Not enough SMs to use max_autotune_gemm mode
[20700/20700 1:01:08, Epoch 3/3]

```

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro	F1 Weighted
1	0.171300	0.179866	0.942187	0.941885	0.942137
2	0.093800	0.203881	0.947500	0.947280	0.947526
3	0.045300	0.345240	0.947292	0.947078	0.947321

Final Test metrics: {'test\_loss': 0.20002830028533936, 'test\_accuracy': 0.9457894736842105, 'test\_f1\_macro': 0.9457597734458655, 'test\_f1\_weighted': 0.9457597734458655}

**ModernBERT Confusion Matrix**



```

# Perform qualitative error analysis
errors_modern = error_analysis(modern_preds, pred_modern.label_ids, test_ds, label_names=label_names)

Text: No tax on Internet access until at least 2007 Consumers won #39;t have to pay a tax to log on to the Internet until at least 2007, after
True: Business | Pred: Sci/Tech
-----
Text: Software Service Aims to Outfox Caller ID A new computerized service enables customers to create phony outbound phone numbers in order
True: Business | Pred: Sci/Tech
-----
Text: Caterpillar snaps up another remanufacturer of engines PEORIA - Caterpillar Inc. said Wednesday it will acquire a South Carolina remanufacturer
True: Sci/Tech | Pred: Business
-----
Text: Prime minister says Australia faces tough economic year in 2005 (AFP) AFP - Prime Minister John Howard warned that Australia's strong economy faces
True: World | Pred: Business
-----
Text: Gas prices up 5 cents after Hurricane Ivan CAMARILLO, Calif. - Gas prices jumped more than 5 cents a gallon in the past two weeks, largely due to
True: Sci/Tech | Pred: Business
-----
Text: For Jobs, Brazilians Desert Their Cities A growing number of Brazilians are finding it increasingly difficult to get good jobs in big cities
True: Business | Pred: World
-----
Text: Ask Jeeves retools search engine in bid to catch Google, Yahoo SAN FRANCISCO -- Hoping to emerge from the shadow of its more popular rivals, Ask Jeeves
True: Business | Pred: Sci/Tech
-----
Text: Israel to close Erez industrial zone before March Israel would start liquidating the Erez industrial zone in the northern Gaza Strip because
True: Business | Pred: World
-----
Text: Without Batting An Eyelash : A very French welcome for a new &lt;b&gt;...&lt;/b&gt; THAT night, it seemed as if three or four parties were
True: Business | Pred: World
-----
```

Text: China's Lenovo in talks with 'major IT firm' for acquisition (AFP) AFP - China's largest manufacturer of personal computers Lenovo Grou  
True: Sci/Tech | Pred: World

```
# === COMPARISON (TASK 3 & 4) ===
from sklearn.metrics import precision_recall_fscore_support, accuracy_score, f1_score

# Get validation predictions for comparison (using best model)
val_pred_distil = trainer_distil.predict(t_val_distil)
y_val_true = val_pred_distil.label_ids
y_val_pred = np.argmax(val_pred_distil.predictions, axis=1)

val_pred_modern = trainer_modern.predict(t_val_modern)
y_val_true_mod = val_pred_modern.label_ids
y_val_pred_mod = np.argmax(val_pred_modern.predictions, axis=1)

# Use test predictions already gathered
y_true = pred_distil.label_ids
y_pred_distil = np.argmax(pred_distil.predictions, axis=1)
y_true_mod = pred_modern.label_ids
y_pred_modern = np.argmax(pred_modern.predictions, axis=1)

# --- Task 3: Compare Test Set Performance ---
#print("\n--- Model Performance Comparison ---")
summary = pd.DataFrame([
    {
        "model": "DistilBERT",
        "val_accuracy": accuracy_score(y_val_true, y_val_pred),
        "val_f1_macro": f1_score(y_val_true, y_val_pred, average="macro"),
        "test_accuracy": accuracy_score(y_true, y_pred_distil),
        "test_f1_macro": f1_score(y_true, y_pred_distil, average="macro"),
    },
    {
        "model": "ModernBERT",
        "val_accuracy": accuracy_score(y_val_true_mod, y_val_pred_mod),
        "val_f1_macro": f1_score(y_val_true_mod, y_val_pred_mod, average="macro"),
        "test_accuracy": accuracy_score(y_true_mod, y_pred_modern),
        "test_f1_macro": f1_score(y_true_mod, y_pred_modern, average="macro"),
    }
])
display(summary)

# Per-class metrics comparison
def per_class_table(y_true, y_pred, labels):
    p, r, f, s = precision_recall_fscore_support(y_true, y_pred, labels=list(range(len(labels))))
    df = pd.DataFrame({"label": labels, "precision": p, "recall": r, "f1": f, "support": s})
    return df

#print("\nDistilBERT Per-Class Metrics (Test Set):")
df_distil_cls = per_class_table(y_true, y_pred_distil, label_names)
display(df_distil_cls)

#print("\nModernBERT Per-Class Metrics (Test Set):")
df_modern_cls = per_class_table(y_true_mod, y_pred_modern, label_names)
display(df_modern_cls)
```

	model	val_accuracy	val_f1_macro	test_accuracy	test_f1_macro
<b>0</b>	DistilBERT	0.944896	0.944653	0.944868	0.944903
<b>1</b>	ModernBERT	0.947500	0.947280	0.945789	0.945760
	label	precision	recall	f1	support
<b>0</b>	World	0.959151	0.951579	0.955350	1900
<b>1</b>	Sports	0.988415	0.987895	0.988155	1900
<b>2</b>	Business	0.917946	0.912632	0.915281	1900
<b>3</b>	Sci/Tech	0.914375	0.927368	0.920826	1900
	label	precision	recall	f1	support
<b>0</b>	World	0.960317	0.955263	0.957784	1900
<b>1</b>	Sports	0.985856	0.990526	0.988186	1900
<b>2</b>	Business	0.919063	0.908421	0.913711	1900
<b>3</b>	Sci/Tech	0.917837	0.928947	0.923359	1900

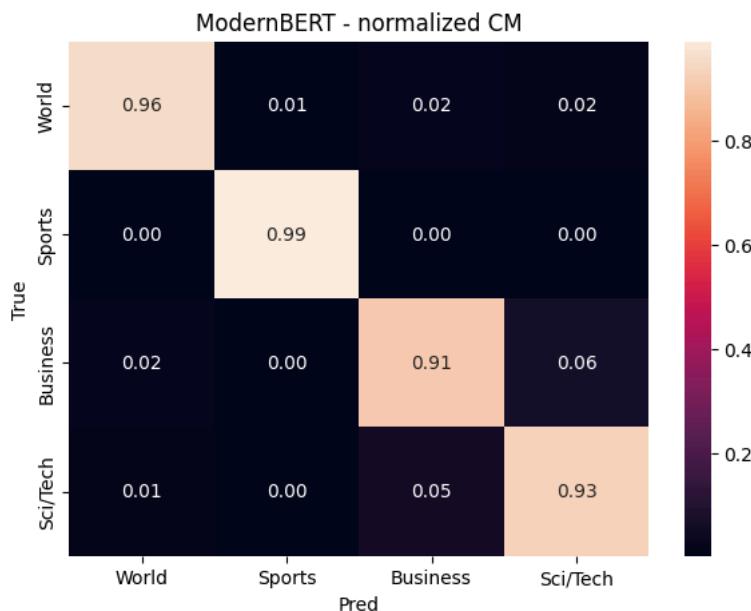
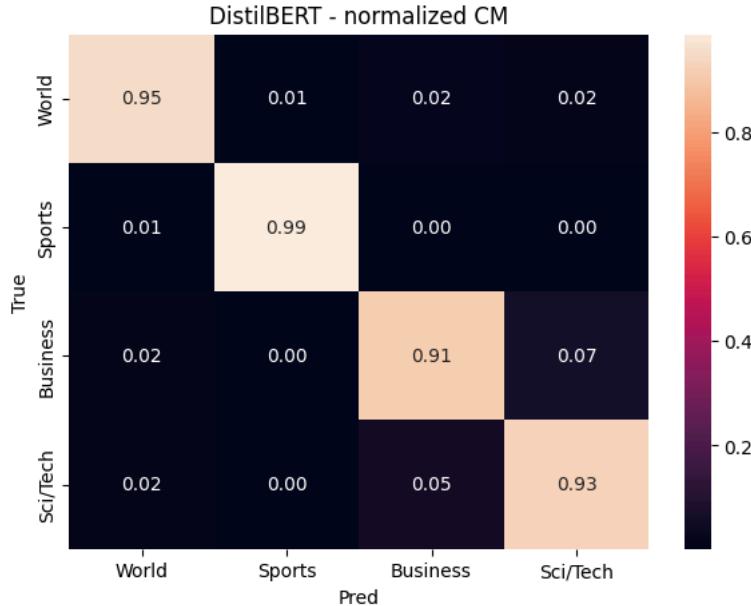
```
# Normalized Confusion Matrix Comparison
def plot_cm(y_true, y_pred, labels, title):
```

```

cm = confusion_matrix(y_true, y_pred)
cmn = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(7, 5))
sns.heatmap(cmn, annot=True, fmt=".2f", xticklabels=labels, yticklabels=labels)
plt.xlabel("Pred")
plt.ylabel("True")
plt.title(title)
plt.show()

plot_cm(y_true, y_pred_distil, label_names, "DistilBERT - normalized CM")
plot_cm(y_true_mod, y_pred_modern, label_names, "ModernBERT - normalized CM")

```



```

# --- Task 4: Compare Error Analyses ---
import torch.nn.functional as F
def make_error_df(pred_output, tokenizer, raw_test_ds):
    """Creates a DataFrame of misclassified samples with confidence scores."""
    logits = pred_output.predictions
    probs = F.softmax(torch.tensor(logits), dim=1).numpy()
    preds = np.argmax(logits, axis=1)
    labels = pred_output.label_ids
    texts = raw_test_ds["text"]

    df = pd.DataFrame({"text": texts, "true_id": labels, "pred_id": preds})
    df["true"] = df["true_id"].map(lambda x: label_names[x])
    df["pred"] = df["pred_id"].map(lambda x: label_names[x])
    df["pred_conf"] = np.max(probs, axis=1)

    # Filter for errors and sort by confidence
    df_err = df[df["true_id"] != df["pred_id"]].copy()
    df_err = df_err.sort_values("pred_conf", ascending=False)
    return df_err

```

```

err_distil = make_error_df(pred_distil, tokenizer_distil, test_ds)
err_modern = make_error_df(pred_modern, tokenizer_modern, test_ds)

# Save top errors to CSV
err_distil.head(200).to_csv("errors_distil_top200.csv", index=False)
err_modern.head(200).to_csv("errors_modern_top200.csv", index=False)
print("Saved error CSVs.")

# Display high-confidence errors
print("DistilBERT High-Confidence Misclassification Examples:")
display(err_distil.head(5)[["text", "true", "pred", "pred_conf"]])
print("ModernBERT High-Confidence Misclassification Examples:")
display(err_modern.head(5)[["text", "true", "pred", "pred_conf"]])

# Compare error types
print("DistilBERT Error type statistics:")
display(err_distil.groupby(['true', 'pred']).size().sort_values(ascending=False).to_frame(name='count'))
print("ModernBERT Error type statistics:")
display(err_modern.groupby(['true', 'pred']).size().sort_values(ascending=False).to_frame(name='count'))

# Compare unique and common errors
set_distil_err_texts = set(err_distil["text"])
set_modern_err_texts = set(err_modern["text"])
unique_distil = set_distil_err_texts - set_modern_err_texts
unique_modern = set_modern_err_texts - set_distil_err_texts
common_err = set_distil_err_texts & set_modern_err_texts
print("DistilBERT Unique Misclassifications:", len(unique_distil))
print("ModernBERT Unique Misclassifications:", len(unique_modern))
print("Common Misclassifications:", len(common_err))

```

Saved error CSVs.

DistilBERT High-Confidence Misclassification Examples:

		text	true	pred	pred_conf
<b>912</b>	Bryant Makes First Appearance at Trial (AP) AP...	Sci/Tech	Sports	0.999592	
<b>2126</b>	Bush Scraps Most U.S. Sanctions on Libya (Reut...	Sci/Tech	World	0.999454	
<b>3688</b>	Ljubicic Downs Hanesic at Open De Moselle Top...	World	Sports	0.999177	
<b>1110</b>	EU foreign ministers hope to break deadlock ov...	Business	World	0.999058	
<b>1197</b>	World briefs LONDON - A man wielding a machete...	Business	World	0.999026	

ModernBERT High-Confidence Misclassification Examples:

		text	true	pred	pred_conf
<b>1197</b>	World briefs LONDON - A man wielding a machete...	Business	World	0.999992	
<b>248</b>	Caterpillar snaps up another remanufacturer of...	Sci/Tech	Business	0.999912	
<b>4396</b>	Revolving Door &lt;em&gt; IN&lt;/em&gt;&lt;br...	World	Business	0.999908	
<b>4540</b>	Miss Peru takes Miss World crown Twenty-year-o...	Sports	World	0.999901	
<b>6185</b>	Special to ESPN.com It #39;s the age old quest...	Business	Sports	0.999897	

DistilBERT Error type statistics:

	count	
true	pred	
<b>Business</b>	<b>Sci/Tech</b>	126
<b>Sci/Tech</b>	<b>Business</b>	100
<b>World</b>	<b>Business</b>	47
<b>Business</b>	<b>World</b>	35
<b>World</b>	<b>Sci/Tech</b>	35
<b>Sci/Tech</b>	<b>World</b>	31
<b>Sports</b>	<b>World</b>	11
<b>World</b>	<b>Sports</b>	10
<b>Sports</b>	<b>Business</b>	8
<b>Sci/Tech</b>	<b>Sports</b>	7
<b>Business</b>	<b>Sports</b>	5
<b>Sports</b>	<b>Sci/Tech</b>	4

ModernBERT Error type statistics:

	count	
true	pred	
<b>Business</b>	<b>Sci/Tech</b>	121
<b>Sci/Tech</b>	<b>Business</b>	102
<b>Business</b>	<b>World</b>	45
<b>World</b>	<b>Business</b>	42
	<b>Sci/Tech</b>	32
<b>Sci/Tech</b>	<b>World</b>	25
<b>World</b>	<b>Sports</b>	11
<b>Business</b>	<b>Sports</b>	8
<b>Sci/Tech</b>	<b>Sports</b>	8
<b>Sports</b>	<b>Business</b>	8
	<b>Sci/Tech</b>	5
	<b>World</b>	5

DistilBERT Unique Misclassifications: 122

ModernBERT Unique Misclassifications: 115

Common Misclassifications: 297

DistilBERT Unique Examples (2 samples):

1 BearingPoint CFO Out; Error Found McLean technology consulting company BearingPoint Inc. said yesterday that its chief financial officer, F  
2 eBay gets into rentals eBay plans to buy the apartment and home rental service Rent.com for \$415 million, adding to its already exhaustive  
ModernBERT Unique Examples (2 samples):

1 Ryder Cup: Europe close on victory Another good day at Oakland Hills sees Europe move 11-5 clear of the USA going into Sunday's Ryder Cup s  
2 EU Head Office Trims 2005 Growth Forecast (AP) AP - The European Union's head office issued a bleak economic report Tuesday, warning that t

```
# Word Cloud Analysis  
from wordcloud import WordCloud
```

```

def plot_wordcloud(err_df, true_label, pred_label, model_name):
    """Generates a word cloud for a specific error type."""
    target_errors_df = err_df[(err_df['true'] == true_label) & (err_df['pred'] == pred_label)]
    if not target_errors_df.empty:
        text_corpus = ' '.join(target_errors_df['text'].tolist())
        wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text_corpus)
        plt.figure(figsize=(10, 5))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis('off')
        plt.title(f'Word Cloud for {true_label} -> {pred_label} Misclassifications ({model_name})')
        plt.show()
        print(f"Found {len(target_errors_df)} examples where '{true_label}' was misclassified as '{pred_label}'.")
    else:
        print(f"No error samples found for {true_label} -> {pred_label} in {model_name}.")
```

```
# Plot word clouds for the most common error types  
# DistilBERT: Business -> Sci/Tech  
plot_wordcloud(err_distil, "Business", "Sci/Tech", "DistilBERT")
```



Found 121 examples where 'Business' was misclassified as 'Sci/Tech'.

```
# DistilBERT: Sci/Tech → Business  
plot_wordcloud(err_distil, "Sci/Tech", "Business", "DistilBERT")
```

