```
!pip install transformers
!pip install evaluate
```

显示隐藏的输出项

```
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
# ==============================
print(transformers.__version__)
acc_metric = evaluate.load("accuracy")
f1_metric = evaluate.load("f1")

def compute_metrics(eval_pred):
    logits, labels = eval_pred.predictions, eval_pred.label_ids
    preds = np.argmax(logits, axis=1)
    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}
```

显示隐藏的输出项

```
# ==============================
# Loading data
# ==============================
DATASET_NAME = "ag_news"
raw = load_dataset(DATASET_NAME)
split = raw["train"].train_test_split(test_size=0.08, seed=42)
train_ds = split["train"]
val_ds = split["test"]
test_ds = raw["test"]

label_names = train_ds.features["label"].names
num_labels = len(label_names)
```

显示隐藏的输出项

```
# ==============================
# Tokenization
# ==============================
def tokenize_datasets(model_ckpt, train_ds, val_ds, test_ds, max_length=128):
    tokenizer = AutoTokenizer.from_pretrained(model_ckpt, use_fast=True)
    def _tok(batch):
        return tokenizer(batch["text"], truncation=True, padding='max_length', max_length=max_length)
    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)
    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 model
# ==============================
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):
    model = AutoModelForSequenceClassification.from_pretrained(model_ckpt, num_labels=num_labels)

    training_args = TrainingArguments(
        output_dir=outdir,
        eval_strategy="epoch",                    # Automatic verification at each epoch
        save_strategy="epoch",                    # Save checkpoints at each 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"
    )

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

    # ==============================
    # determine whether to resume
    # ==============================
    if resume_from_checkpoint:
        print(f"Resuming training from checkpoint: {resume_from_checkpoint}")
    else:
        print("Starting training from scratch")

    trainer.train(resume_from_checkpoint=resume_from_checkpoint)

    # ==============================
    # Test set evaluation
    # ==============================
    pred = trainer.predict(test_dataset)
    print("Final Test metrics:", pred.metrics)

    return trainer, model, pred
```

```
# ==============================
# Confusion Matrix & Error Analysis
# ==============================
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))]

    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
            })

    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
```

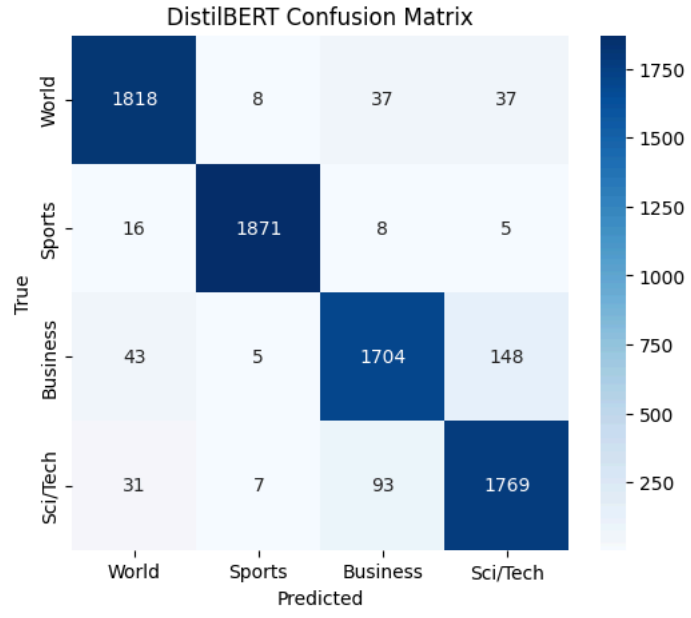
```
# ==============================
# DistilBERT Fine-tuning
# ==============================
DISTIL_CKPT = "distilbert-base-uncased"
```

```
DISTIL_LR = 2e-5
DISTIL_BATCH = 32
DISTIL_EPOCHS = 3

tokenizer_distil, t_train_distil, t_val_distil, t_test_distil = tokenize_datasets(
        DISTIL_CKPT, train_ds, val_ds, test_ds
)

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
)

# Confusion Matrix & Error 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, 2.01kB/s]
config.json: 100%  483/483 [00:00<00:00, 12.5kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 3.08MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 7.39MB/s]
Map: 100%  110400/110400 [00:57<00:00, 1445.83 examples/s]
Map: 100%  9600/9600 [00:04<00:00, 3419.33 examples/s]
Map: 100%  7600/7600 [00:01<00:00, 5264.16 examples/s]
model.safetensors: 100%  268M/268M [00:05<00:00, 43.8MB/s]
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are newly
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 18:10, Epoch 3/3]
```

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro	F1 Weighted
1	0.189900	0.182267	0.939271	0.938873	0.939144
2	0.135000	0.180944	0.942708	0.942408	0.942675
3	0.089800	0.202472	0.942500	0.942258	0.942517

```
Final Test metrics: {'test_loss': 0.1727038323879242, 'test_accuracy': 0.9423684210526316, 'test_f1_macro': 0.9423668324849056, 'test_f1_we
```




```
errors_distil = error_analysis(distil_preds, pred_distil.label_ids, test_ds, label_names=label_names)
```

```
Text: Microsoft signs two Indian deals Microsoft is to form multi-million pound partnerships with two Indian software firms, and is expecte
 True: Sci/Tech | Pred: Business
----------------------------------------------------------------------
Text: Paris Tourists Search for Key to 'Da Vinci Code' (Reuters) Reuters - A funny thing happened on the way to the\Mona Lisa. Visitors to
 True: World | Pred: Sci/Tech
----------------------------------------------------------------------
Text: Ali gives Iraq fighting chance The back of his shirt told the story last night at the Peristeri Olympic Boxing Hall.
 True: Sports | Pred: World
----------------------------------------------------------------------
```

```
Text: Is this the end of IT as we know it? Halsey Minor, CEO of hosted integration provider grand central Communications, has a powerful me
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: CA, Partners Move On As Kumar Faces Charges Concern over the fate of former Computer Associates International chairman and CEO Sanjay
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: Vodafone targets Japan with 3G offensive Vodafone has unveiled plans for 10 new third-generation handsets for Christmas to help shore
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: The battle for DR Congo's wildlife As donors pledge \$40m to save DR Congo's wildlife, life in Virunga Park remains a battle ground.
 True: Sci/Tech | Pred: World
--------------------------------------------------
Text: Labor situation deserves honest talk For a moment last week, President Bush escaped the White House spin chamber and was the plainspo
 True: Business | Pred: World
--------------------------------------------------
Text: FDA OKs Scientist Publishing Vioxx Data (AP) AP - The Food and Drug Administration has given a whistle-blower scientist permission to
 True: World | Pred: Sci/Tech
--------------------------------------------------
Text: Stocks Dip on Consumer Income Report News (AP) AP - An unsettling report on consumer incomes set off a spate of profit-taking on Wall
 True: Business | Pred: World
--------------------------------------------------
```

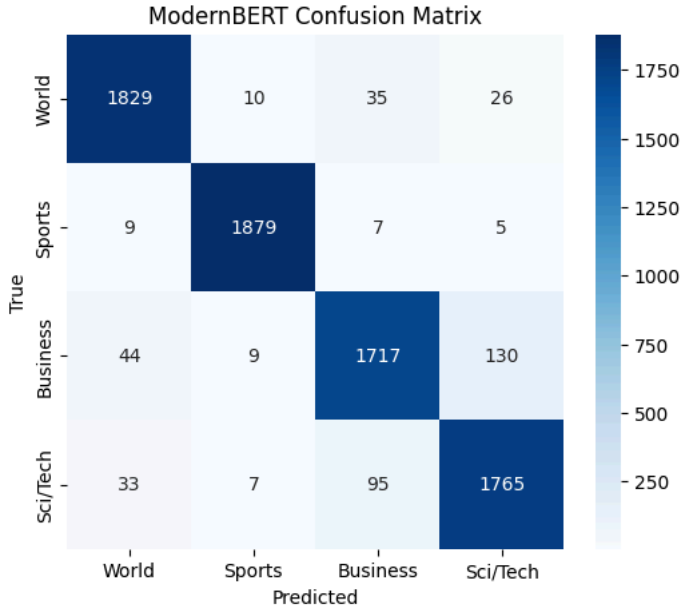
```
# ==============================
# ModernBERT Fine-tuning
# ==============================
import torch._dynamo
torch._dynamo.config.suppress_errors = True

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

tokenizer_modern, t_train_modern, t_val_modern, t_test_modern = tokenize_datasets(
        MODERN_CKPT, train_ds, val_ds, test_ds
)

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
)

# Confusion Matrix & Error 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, 1.33MB/s]
tokenizer.json:          2.13M/? [00:00<00:00, 29.6MB/s]
special_tokens_map.json: 100%                                                694/694 [00:00<00:00, 70.1kB/s]
Map: 100%                                                    110400/110400 [00:27<00:00, 3987.21 examples/s]
Map: 100%                                                    9600/9600 [00:02<00:00, 3482.09 examples/s]
Map: 100%                                                    7600/7600 [00:02<00:00, 4238.01 examples/s]
config.json:          1.19k/? [00:00<00:00, 98.1kB/s]
model.safetensors: 100%                                                599M/599M [01:08<00:00, 14.4MB/s]
Some weights of ModernBertForSequenceClassification were not initialized from the model checkpoint at answerdotai/ModernBERT-base and are n
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
W0913 21:55:14.971000 413 torch/_inductor/utils.py:1436] [1/0_1] Not enough SMs to use max_autotune_gemm mode
```

[20700/20700 1:05:30, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro	F1 Weighted
1	0.174800	0.179104	0.942500	0.942170	0.942427
2	0.089600	0.203006	0.947604	0.947357	0.947595
3	0.038400	0.350692	0.945625	0.945384	0.945650

```
Final Test metrics: {'test_loss': 0.20180293917655945, 'test_accuracy': 0.9460526315789474, 'test_f1_macro': 0.9459716717408502, 'test_f1_w
```




```
errors_modern = error_analysis(modern_preds, pred_modern.label_ids, test_ds, label_names=label_names)
```

```
Text: TheStreet.com May Be Up for Sale -- Report (Reuters) Reuters - TheStreet.com Inc. , the\financial news and commentary Web site, may b
 True: Sci/Tech | Pred: Business
--------------------------------------------------
Text: China Reports Births of Two Giant Pandas (AP) AP - For pandas, it's practically a baby boom. Two giant pandas were born this week, an
 True: Sci/Tech | Pred: World
--------------------------------------------------
Text: Court: File-swapping software not liable for copyright violations The makers of two leading file-sharing programs are not legally lia
 True: Sci/Tech | Pred: Business
--------------------------------------------------
Text: Amazon #39;s UK DVD Rentals Might Presage War with Netflix Analysts said Amazon #39;s launch of the DVD rental service in the UK migh
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: Adobe updates RAW plug-in with digital negative format Adobe has updated Photoshop #39;s support for digital cameras #39; RAW image f
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: PeopleSofts big bash See you next year in Las Vegas , proclaimed a marquee at the PeopleSoft user conference in San Francisco in late
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: Ask Jeeves retools search engine in bid to catch Google, Yahoo SAN FRANCISCO -- Hoping to emerge from the shadow of its more popular
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: Retailers Looking to Move Plasma TV's (AP) AP - Hanging stockings by the chimney with care? Retailers hope that St. Nicholas soon wil
 True: Business | Pred: Sci/Tech
--------------------------------------------------
Text: English 'world language' forecast A third of people on the planet will be learning English in the next decade, \says a report.
 True: World | Pred: Sci/Tech
--------------------------------------------------
Text: Hall of Shame  Hall of Fame We spotlight people and products that pester us...and the heroes saving us from annoyances.
```

```
True: Sci/Tech | Pred: Business
--------------------------------------------------
```

```python
# %% Evaluation and comparison (accuracy / macro F1 / per-class)
# Distil
from sklearn.metrics import precision_recall_fscore_support
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)
y_true = pred_distil.label_ids
y_pred_distil = np.argmax(pred_distil.predictions, axis=1)

# Modern
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)
y_true_mod = pred_modern.label_ids      # Should be consistent with y_true (same test set)
y_pred_modern = np.argmax(pred_modern.predictions, axis=1)

from sklearn.metrics import accuracy_score, f1_score
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
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

df_distil_cls = per_class_table(y_true, y_pred_distil, label_names)
df_modern_cls = per_class_table(y_true_mod, y_pred_modern, label_names)
display(df_distil_cls)
display(df_modern_cls)
```

	model	val_accuracy	val_f1_macro	test_accuracy	test_f1_macro
0	DistilBERT	0.942708	0.942408	0.942368	0.942367
1	ModernBERT	0.947604	0.947357	0.946053	0.945972

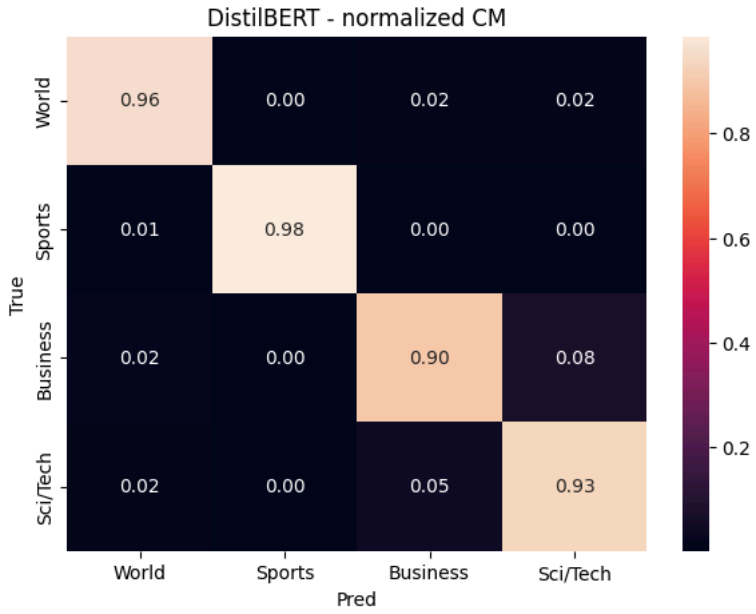
	label	precision	recall	f1	support
0	World	0.952830	0.956842	0.954832	1900
1	Sports	0.989424	0.984737	0.987075	1900
2	Business	0.925081	0.896842	0.910743	1900
3	Sci/Tech	0.903012	0.931053	0.916818	1900

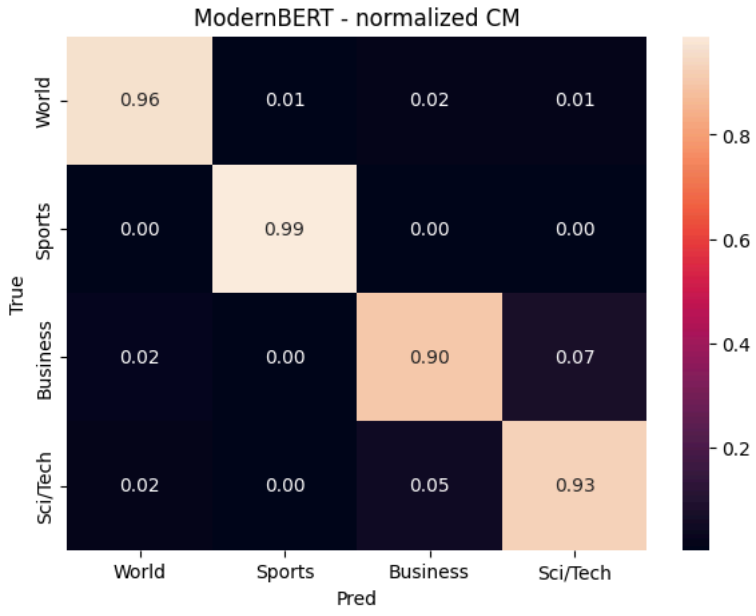
	label	precision	recall	f1	support
0	World	0.955091	0.962632	0.958847	1900
1	Sports	0.986352	0.988947	0.987648	1900
2	Business	0.926106	0.903684	0.914758	1900
3	Sci/Tech	0.916407	0.928947	0.922635	1900

后续步骤：


```python
# %% Confusion Matrix (normalized)
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")
```






```
import torch.nn.functional as F
# %% Error analysis: save misclassified samples and increase confidence level of misclassified samples
def make_error_df(pred_output, tokenizer, raw_test_ds):
        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"]    # Original 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])
        # Confidence
        df["pred_conf"] = np.max(probs, axis=1)
        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 the top 200 error samples as csv
err_distil.head(200).to_csv("errors_distil_top200.csv", index=False)
```

```
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.")

#  Print  several  high-confidence  but  wrong  examples
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"]])

#  %%  Counting  the  types  and  number  of  errors  made  by  DistilBERT
print("DistilBERT  Error  type  statistics：")
display(err_distil.groupby(['true',  'pred']).size().sort_values(ascending=False).to_frame(name='count'))

#  %%  Counting  the  types  and  number  of  errors  made  by  ModernBERT
print("ModernBERT  Error  type  statistics：")
display(err_modern.groupby(['true',  'pred']).size().sort_values(ascending=False).to_frame(name='count'))

#  %%  Compare  the  misclassified  sets  of  two  models  (intersection/difference)
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))

#  Printing  Example
print("DistilBERT  Unique  Examples  (2  samples):")
for  i,t  in  enumerate(list(unique_distil)[:2]):  print(i+1,  t)
print("ModernBERT  Unique  Examples  (2  samples):")
for  i,t  in  enumerate(list(unique_modern)[:2]):  print(i+1,  t)
```

```
Saved error CSVs.
DistilBERT High-Confidence Misclassification Examples:
```

	text	true	pred	pred_conf
2126	Bush Scraps Most U.S. Sanctions on Libya (Reut...	Sci/Tech	World	0.999053
912	Bryant Makes First Appearance at Trial (AP) AP...	Sci/Tech	Sports	0.999017
1269	Philippines mourns dead in Russian school sieg...	Sci/Tech	World	0.998799
7228	Indonesian diplomats asked to help improve RI ...	Business	World	0.998725
1197	World briefs LONDON - A man wielding a machete...	Business	World	0.998257

```
ModernBERT High-Confidence Misclassification Examples:
```

	text	true	pred	pred_conf
7467	Daily Mail hits back at Blunkett The Daily Mai...	World	Sports	0.999990
2660	Gas prices up 5 cents after Hurricane Ivan CAM...	Sci/Tech	Business	0.999981
1197	World briefs LONDON - A man wielding a machete...	Business	World	0.999975
4540	Miss Peru takes Miss World crown Twenty-year-o...	Sports	World	0.999951
3077	Brewers buyer expected to step out of the shad...	Business	Sports	0.999946

```
DistilBERT Error type statistics:
```

		count
true	pred	
Business	**Sci/Tech**	148
Sci/Tech	**Business**	93
Business	**World**	43
Sci/Tech	**World**	31
Sports	**World**	16
Business	**Sports**	8
World	**Sports**	8
Sci/Tech	**Sports**	7
Business	**Sports**	5
Sports	**Sci/Tech**	5
Sci/Tech	**Business**	95

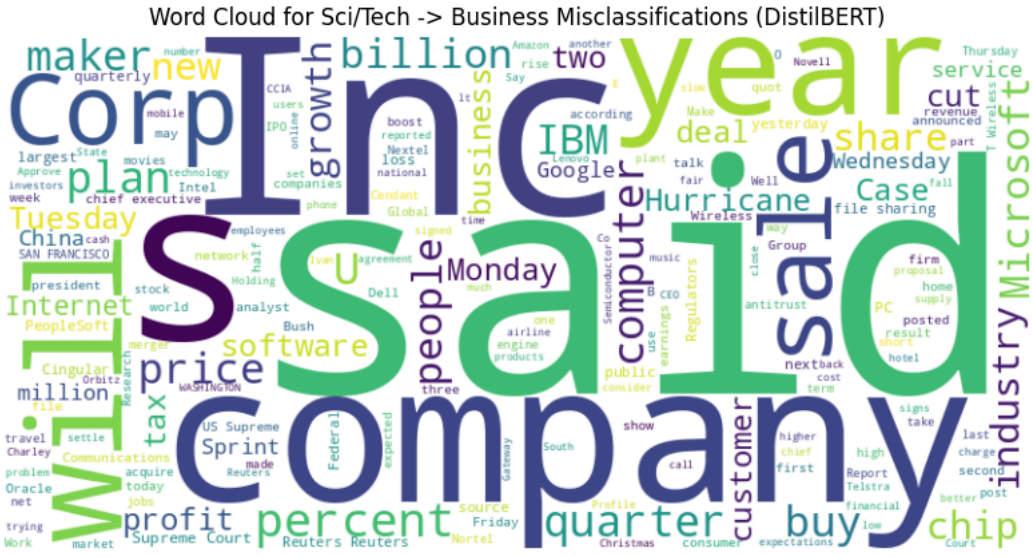
```
# %% Install word cloud library
!pip install wordcloud

Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.12/dist-packages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-packages (from wordcloud) (11.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (4.59.2)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.17
```

```
# %% Generate a word cloud for a specific error type

import matplotlib.pyplot as plt
from wordcloud import WordCloud

def plot_wordcloud(err_df, true_label, pred_label, model_name):

    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}.")
ModernBERT Unique Misclassifications: 106
```

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

2 E.U. Regulators Approve Oracle's PeopleSoft Bid European Union antitrust regulators on Tuesday cleared Oracle Corp.'s hostile \$7.7 billi
ModernBERT Unique Examples (2 samples):
1 Mazu scores VC lifeblood from Symantec, others Mazu Networks, the Cambridge, Massachusetts, network intrusion prevention system (IPS) tec
2 SBC calling on cable SBC is teaming up with Microsoft to provide consumers with a new way to view television -- a move that puts it in di
```

Found 148 examples where 'Business' was misclassified as 'Sci/Tech'.

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



Found 93 examples where 'Sci/Tech' was misclassified as 'Business'.

```
# ModernBERT: Business -> Sci/Tech
plot_wordcloud(err_modern, "Business", "Sci/Tech", "ModernBERT")
```

Found 130 examples where 'Business' was misclassified as 'Sci/Tech'.
Found 0 texts containing 'Swill'.

```
# ModernBERT: Sci/Tech -> Business
plot_wordcloud(err_modern, "Sci/Tech", "Business", "ModernBERT")
```