## Import libraries and dependencies for data, models, and evaluation

```
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sentence_transformers import SentenceTransformer, util,
InputExample, losses, CrossEncoder
from sklearn.metrics import f1_score
from torch.utils.data import DataLoader
os.environ["WANDB_DISABLED"] = "true"
```

## Loading the dataset, creating a 50k stratified subset, and splitting it

```
df = pd.read_csv("train.csv")

subset_size = 50000
df_subset, _ = train_test_split(
    df,
    train_size=subset_size,
    stratify=df["is_duplicate"],
    random_state=42
)

train, temp = train_test_split(
    df_subset,
    test_size=0.2,
    stratify=df_subset["is_duplicate"],
    random_state=42
)
valid, test = train_test_split(
    temp,
    test_size=0.5,
    stratify=temp["is_duplicate"],
    random_state=42
)

os.makedirs("splits", exist_ok=True)

train.to_csv("splits/train.csv", index=False)
valid.to_csv("splits/valid.csv", index=False)
test.to_csv("splits/test.csv", index=False)

print("Final sizes:", len(train), len(valid), len(test))
```

```
Final sizes: 40000 5000 5000
```

## Baseline Model: Encode test questions with a pre-trained bi-encoder

```
test = pd.read_csv("splits/test.csv")

model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2")

emb1 = model.encode(test["question1"].tolist(), batch_size=128,
convert_to_numpy=True)
emb2 = model.encode(test["question2"].tolist(), batch_size=128,
convert_to_numpy=True)

sims = util.cos_sim(emb1, emb2).diagonal()

best_f1, best_thr = 0, 0
for thr in [i/100 for i in range(-100, 101)]:
    sims = util.cos_sim(emb1, emb2).diagonal().cpu().numpy()
    preds = (sims >= thr).astype(int)
    f1 = f1_score(test["is_duplicate"], preds)
    if f1 > best_f1:
        best_f1, best_thr = f1, thr

print(f"[Baseline] Test F1={best_f1:.4f} at threshold={best_thr:.2f}")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"abdd357e3e9b430cbfc5aed81e94ea68","version_major":2,"vers
ion_minor":0}

{"model_id":"192dc55ed03f4a4ea4efce2a32f3b041","version_major":2,"vers
ion_minor":0}

{"model_id":"91a5e7d5213c4c27aa3fdc27a2d3b2ca","version_major":2,"vers
ion_minor":0}

{"model_id":"feef575d302b4b0a9bde712813106199","version_major":2,"vers
ion_minor":0}
```

{"model_id":"81ea1c556d2f4483b79a71e144534cd7","version_major":2,"version_minor":0}

{"model_id":"13f01499efad4d1c8a2c04d0c487e69f","version_major":2,"version_minor":0}

{"model_id":"2859cfa25852435d981a64128c0bc4cc","version_major":2,"version_minor":0}

{"model_id":"98027bd8c31e4ca5a79c8b44255c9ce4","version_major":2,"version_minor":0}

{"model_id":"79a7542365134c9f8645eb7ef19ac69a","version_major":2,"version_minor":0}

{"model_id":"3ab08a21a71b48afb0ed18b65d3aff28","version_major":2,"version_minor":0}

{"model_id":"6a563f069ebd4d2ebaf2f79809355d1c","version_major":2,"version_minor":0}

```
[Baseline] Test F1=0.7345 at threshold=0.75
```

## Training and Evaluating Bi-Encoders

Fine-tuning three different bi-encoders with specific loss functions and base models:

- **Cosine Similarity Loss** using `all-MiniLM-L6-v2`
- **Contrastive Loss** using `paraphrase-MiniLM-L6-v2`
- **Multiple Negatives Ranking Loss (MNR)** using `all-mpnet-base-v2`

```
train = pd.read_csv("splits/train.csv")
valid = pd.read_csv("splits/valid.csv")
test = pd.read_csv("splits/test.csv")

def to_examples(df):
    return [InputExample(texts=[row["question1"], row["question2"]],
label=float(row["is_duplicate"])) for _, row in df.iterrows()]

train_examples = to_examples(train)
valid_examples = to_examples(valid)
test_examples = to_examples(test)

def run_biencoder(loss_type, base_model, epochs=2, batch_size=32,
lr=2e-5):
    model = SentenceTransformer(base_model)

    if loss_type == "mnr":
        train_loss = losses.MultipleNegativesRankingLoss(model)
    elif loss_type == "cos":
        for ex in train_examples:
```

```
            ex.label = 1.0 if ex.label == 1.0 else -1.0
        train_loss = losses.CosineSimilarityLoss(model)
    elif loss_type == "contrastive":
        train_loss = losses.OnlineContrastiveLoss(
            model,

distance_metric=losses.SiameseDistanceMetric.COSINE_DISTANCE,
            margin=0.5
        )
    else:
        raise ValueError("loss_type must be one of: mnr, cos,
contrastive")

    train_dataloader = DataLoader(train_examples, shuffle=True,
batch_size=batch_size)

    model.fit(
        train_objectives=[(train_dataloader, train_loss)],
        epochs=epochs,
        warmup_steps=100,
        optimizer_params={'lr': lr},
        show_progress_bar=True
    )

    # Evaluate
    def evaluate(model, examples, lt):
        q1 = [ex.texts[0] for ex in examples]
        q2 = [ex.texts[1] for ex in examples]
        labels = [int(ex.label) if lt != "cos" else (1 if ex.label ==
1.0 else 0) for ex in examples]

        emb1 = model.encode(q1, batch_size=128, convert_to_numpy=True)
        emb2 = model.encode(q2, batch_size=128, convert_to_numpy=True)
        sims = util.cos_sim(emb1, emb2).diagonal().cpu().numpy()

        best_f1, best_thr = 0, 0
        for thr in [i/100 for i in range(-100, 101)]:
            preds = (sims >= thr).astype(int)
            f1 = f1_score(labels, preds)
            if f1 > best_f1:
                best_f1, best_thr = f1, thr
        return best_f1, best_thr

    val_f1, thr = evaluate(model, valid_examples, loss_type)
    test_f1, _ = evaluate(model, test_examples, loss_type)
    print(f"[{loss_type}] Validation F1={val_f1:.4f} | Test
F1={test_f1:.4f} at threshold={thr:.2f}")
    return model, test_f1

cos_model, cos_f1 = run_biencoder("cos", "sentence-transformers/all-
```

```
MiniLM-L6-v2")
contrast_model, contrast_f1 = run_biencoder("contrastive", "sentence-transformers/paraphrase-MiniLM-L6-v2")
mnr_model, mnr_f1 = run_biencoder("mnr", "sentence-transformers/all-mpnet-base-v2")
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging result (for instance --report_to none).
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging result (for instance --report_to none).
```

{"model_id":"9408564c463a437eaf70d840b5bfc826","version_major":2,"version_minor":0}

```
<IPython.core.display.HTML object>
```

```
[cos] Validation F1=0.6915 | Test F1=0.6816 at threshold=0.23
```

{"model_id":"afdcd8f3f249418e8de6cfb275a0c7b8","version_major":2,"version_minor":0}

{"model_id":"adb53a50bdf94a38a743b7239c52dc4f","version_major":2,"version_minor":0}

{"model_id":"33f9f9b789a04687b166d885ba6dfb84","version_major":2,"version_minor":0}

{"model_id":"9971e1a2187f412492b11a107fe81109","version_major":2,"version_minor":0}

{"model_id":"279e766d62174a94b1c9dd23b59f3756","version_major":2,"version_minor":0}

{"model_id":"1fccd68a35d948cea347e57a293cbf0d","version_major":2,"version_minor":0}

{"model_id":"11b86c5cd9c54399b9a4a8920d911ff5","version_major":2,"version_minor":0}

{"model_id":"5d653830f776400abb0a84a5ed00a4c9","version_major":2,"version_minor":0}

{"model_id":"b5a1ff7966a84017bbd10d024b1f11b4","version_major":2,"version_minor":0}

{"model_id":"e1ce2aa39e144fd184793e404fbba07a","version_major":2,"version_minor":0}

{"model_id":"ea8f5aa1b66c4210a7097fa148d66eff","version_major":2,"version_minor":0}

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging result (for instance --report_to none).
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging result (for instance --report_to none).
```


```
{"model_id":"53127f6620f74821840c1a33621f2605","version_major":2,"version_minor":0}
```


```
<IPython.core.display.HTML object>
```

```
[contrastive] Validation F1=0.5396 | Test F1=0.5400 at threshold=0.99
```


```
{"model_id":"b49c9420314a4dc3acf102a8142f8c2c","version_major":2,"version_minor":0}

{"model_id":"1695aa776c0742e8a27eca7ebd9d4f23","version_major":2,"version_minor":0}

{"model_id":"3e002aaa778844eb8db46a1ae6cb23d7","version_major":2,"version_minor":0}

{"model_id":"a626c80f663d40218371f7ee75b5d0c5","version_major":2,"version_minor":0}

{"model_id":"65177903420a43d482e88b3796e6d020","version_major":2,"version_minor":0}

{"model_id":"9ee4ed6053cf44e69bfc6e1dbb972237","version_major":2,"version_minor":0}

{"model_id":"04211cd33ff146c488e4a912e54b52e8","version_major":2,"version_minor":0}

{"model_id":"ab0220e7f1224ceeadee773e7c20a127","version_major":2,"version_minor":0}

{"model_id":"a916d315bd844e8fa585c09f8eb63301","version_major":2,"version_minor":0}

{"model_id":"eae283b7294b4b4f8c97731dcce73d09","version_major":2,"version_minor":0}

{"model_id":"3b333974b6c54f69bad0e7f822709f10","version_major":2,"version_minor":0}
```



```
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging result (for instance --report_to none).
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging result (for instance --report_to none).
```

{"model_id":"40fcf231c4994b9db141a9ea5095f20a","version_major":2,"version_minor":0}

```
<IPython.core.display.HTML object>

[mnr] Validation F1=0.7194 | Test F1=0.7117 at threshold=0.74
```

## Cross-Encoder Training & Evaluation

Fine-tuning `ms-marco-MiniLM-L-6-v2` on the training split

```
train_samples = [
    InputExample(texts=[row["question1"], row["question2"]],
label=float(row["is_duplicate"]))
    for _, row in train.iterrows()
]
valid_samples = [
    (row["question1"], row["question2"], int(row["is_duplicate"]))
    for _, row in valid.iterrows()
]
test_samples = [
    (row["question1"], row["question2"], int(row["is_duplicate"]))
    for _, row in test.iterrows()
]

train_dataloader = DataLoader(train_samples, shuffle=True,
batch_size=16)


ce_model = CrossEncoder("cross-encoder/ms-marco-MiniLM-L-6-v2",
num_labels=1)

ce_model.fit(
    train_dataloader=train_dataloader,
    epochs=1,
    warmup_steps=100,
    show_progress_bar=True
)


def evaluate_ce(model, samples):
    texts = [(q1, q2) for q1, q2, _ in samples]
    labels = [lbl for _, _, lbl in samples]

    scores = model.predict(texts)

    best_f1, best_thr = 0, 0
    for thr in np.linspace(0, 1, 101):
        preds = (scores >= thr).astype(int)
        f1 = f1_score(labels, preds)
```

```
        if f1 > best_f1:
            best_f1, best_thr = f1, thr
    return best_f1, best_thr

val_f1, thr = evaluate_ce(ce_model, valid_samples)
test_f1, _ = evaluate_ce(ce_model, test_samples)

print(f"[CrossEncoder] Validation F1={val_f1:.4f} | Test
F1={test_f1:.4f} at threshold={thr:.2f}")
```


```
{"model_id":"add3a28da9d14aaabb00932b696821ed","version_major":2,"vers
ion_minor":0}

{"model_id":"b2df5a0f64fc4d18910786bceec034e8","version_major":2,"vers
ion_minor":0}

{"model_id":"9ea90c3b9ba446b3864134c4206b30ba","version_major":2,"vers
ion_minor":0}

{"model_id":"f81797d0e44c45bebc3bc7da639f30eb","version_major":2,"vers
ion_minor":0}

{"model_id":"c50fbe015703432c86034161d8715848","version_major":2,"vers
ion_minor":0}

{"model_id":"d17b626516234d948dcec18cca362c31","version_major":2,"vers
ion_minor":0}

{"model_id":"752de9b07acf43daa7adfc8c99bcf40a","version_major":2,"vers
ion_minor":0}
```


```
Using the `WANDB_DISABLED` environment variable is deprecated and will
be removed in v5. Use the --report_to flag to control the integrations
used for logging result (for instance --report_to none).
Using the `WANDB_DISABLED` environment variable is deprecated and will
be removed in v5. Use the --report_to flag to control the integrations
used for logging result (for instance --report_to none).

<IPython.core.display.HTML object>

[CrossEncoder] Validation F1=0.8038 | Test F1=0.7999 at threshold=0.02
```

## F1 Score Eval

```
results = {
    "Baseline": "-",
    "Bi-encoder (Cosine)": cos_f1,
    "Bi-encoder (Contrastive)": contrast_f1,
    "Bi-encoder (MNR)": mnr_f1,
    "Cross-encoder": test_f1
```

```
}
print(results)
```

```
{'Baseline': '-', 'Bi-encoder (Cosine)': 0.6816165598817151, 'Bi-encoder (Contrastive)': 0.5400029252596168, 'Bi-encoder (MNR)': 0.7117158671586716, 'Cross-encoder': 0.7998996990972919}
```