Bengali NLP: Dataset Preparation, Training, Validation & Packaging

- Sentiment classification example (Bengali)
- NER dataset example (BIO format)
- Translation dataset example (parallel corpus)
- Code templates in Python (TensorFlow/Keras & PyTorch/HuggingFace)
- How to save datasets as CSV/JSONL/HDF5 (.h5) and models as .h5 / .pt
- Notes on ideal sample sizes

1. Dataset Preparation Overview (Bengali)

General pipeline:
1. Define task & schema
2. Collect raw text data
3. Annotate / label (if supervised)
4. Clean & normalize (Unicode, whitespace, noise)
5. Tokenize (Indic-aware / subword)
6. Split into train/validation/test
7. Save datasets in convenient formats (CSV, JSONL, HDF5)

2. Sentiment Classification Example (Bengali)

Task: Classify Bengali reviews into {positive, negative, neutral}.

2.1 Example Dataset Schema & Sample

id	text	label
1	এই ফোনটা দারুণ! আমি খুব খুশি।	positive
2	ব্যাটারি লাইফ একদমই বাজে।	negative
3	ফোনটা মোটামুটি, বিশেষ ভালো না খারাপও না।	neutral

2.2 Data Loading & Basic Preprocessing (Python)

```
import pandas as pd

# Load from CSV (UTF-8)
df = pd.read_csv("bengali_reviews.csv")  # columns: id, text, label
```

```
def normalize_bn(text: str) -> str:
    if not isinstance(text, str):
        return ""
    # Strip and normalize whitespace
    text = text.strip()
    text = " ".join(text.split())
    # TODO: add Unicode normalization for Bengali (e.g. using indic-nlp-
library)
    return text

df["text"] = df["text"].apply(normalize_bn)

# Drop empty texts
df = df[df["text"].str.len() > 0].reset_index(drop=True)

# Encode labels to integers
label_list = sorted(df["label"].unique())
label2id = {lab: i for i, lab in enumerate(label_list)}
id2label = {i: lab for lab, i in label2id.items()}
df["label_id"] = df["label"].map(label2id)
```

2.3 Train/Validation/Test Split

```
from sklearn.model_selection import train_test_split

train_df, temp_df = train_test_split(
    df, test_size=0.2, random_state=42, stratify=df["label_id"]
)
valid_df, test_df = train_test_split(
    temp_df, test_size=0.5, random_state=42, stratify=temp_df["label_id"]
)

print(len(train_df), len(valid_df), len(test_df))
```

2.4 Training with TensorFlow/Keras (and saving as .h5)

```
import tensorflow as tf
from tensorflow.keras import layers
```

```
# Simple TF text vectorization (for demo; in practice use a pretrained
model)
max_tokens = 20000
max_len = 64

vectorizer = layers.TextVectorization(
    max_tokens=max_tokens,
    output_mode="int",
    output_sequence_length=max_len
)
vectorizer.adapt(train_df["text"].values)

# Prepare tf.data datasets
def df_to_tfdata(df, batch_size=32, shuffle=False):
    texts = df["text"].values
    labels = df["label_id"].values
    ds = tf.data.Dataset.from_tensor_slices((texts, labels))
    if shuffle:
        ds = ds.shuffle(buffer_size=len(df))
    ds = ds.batch(batch_size).map(
        lambda x, y: (vectorizer(x), y),
        num_parallel_calls=tf.data.AUTOTUNE
    ).prefetch(tf.data.AUTOTUNE)
    return ds

train_ds = df_to_tfdata(train_df, shuffle=True)
valid_ds = df_to_tfdata(valid_df)
test_ds  = df_to_tfdata(test_df)

# Build a simple model
model = tf.keras.Sequential([
    layers.Embedding(input_dim=max_tokens, output_dim=128,
mask_zero=True),
    layers.Bidirectional(layers.LSTM(64)),
    layers.Dense(64, activation="relu"),
    layers.Dense(len(label_list), activation="softmax"),
])

model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

```
history = model.fit(
    train_ds,
    validation_data=valid_ds,
    epochs=5
)

# Evaluate on test set
model.evaluate(test_ds)

# Save model as .h5
model.save("bengali_sentiment_model.h5")
```

2.5 Training with HuggingFace Transformers (PyTorch, .pt)

```
from datasets import Dataset, DatasetDict
from transformers import AutoTokenizer,
AutoModelForSequenceClassification, TrainingArguments, Trainer
import torch

model_name = "bert-base-multilingual-cased"  # or a Bengali-specific
model

tokenizer = AutoTokenizer.from_pretrained(model_name)

def to_hf_dataset(df):
    return Dataset.from_pandas(df[["text", "label_id"]])

hf_dataset = DatasetDict({
    "train": to_hf_dataset(train_df),
    "validation": to_hf_dataset(valid_df),
    "test": to_hf_dataset(test_df),
})

def tokenize_batch(batch):
    return tokenizer(
        batch["text"],
        truncation=True,
        padding="max_length",
        max_length=64
    )

hf_dataset = hf_dataset.map(tokenize_batch, batched=True)
hf_dataset = hf_dataset.rename_column("label_id", "labels")
hf_dataset.set_format(type="torch", columns=["input_ids",
"attention_mask", "labels"])

model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=len(label_list),
    id2label=id2label,
    label2id=label2id,
)

training_args = TrainingArguments(
    output_dir="./bn-sentiment-model",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_steps=50
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=hf_dataset["train"],
    eval_dataset=hf_dataset["validation"],
)

trainer.train()
trainer.evaluate(hf_dataset["test"])

# Save as PyTorch model (.pt)
torch.save(model.state_dict(), "bengali_sentiment_model.pt")
```

---

3. NER Dataset Example (BIO Format)

Named Entity Recognition (NER) labels each token with a tag, usually in BIO format: B-ORG, I-ORG, B-PER, I-PER, B-LOC, I-LOC, O, etc.

3.1 Example Annotated Sentence

Sentence (Bengali): 'রবীন্দ্রনাথ ঠাকুর কলকাতায় জন্মগ্রহণ করেন।'

Token-level BIO tags:

রবীন্দ্রনাথ	B-PER
ঠাকুর	I-PER
কলকাতায়	B-LOC
জন্মগ্রহণ	O
করেন।	O

3.2 NER Dataset Format

Common text format: one token per line, with tag; sentences separated by blank lines.

```
রবীন্দ্রনাথ B-PER
ঠাকুর I-PER
কলকাতায় B-LOC
জন্মগ্রহণ O
করেন। O

আমি O
ঢাকায় B-LOC
থাকি O
```

4. Translation Dataset Example (Parallel Corpus)

Machine Translation datasets contain parallel sentence pairs, e.g. English ↔ Bengali.

4.1 Example Parallel Data

Format: TSV (tab-separated) with columns: src_text (English), tgt_text (Bengali)

```
src_text	tgt_text
How are you?	তুমি কেমন আছো?
This phone is very good.	এই ফোনটা খুব ভালো।
I live in Dhaka.	আমি ঢাকায় থাকি।
```

4.2 Loading Parallel Data with Pandas

```
import pandas as pd

df_mt = pd.read_csv("en_bn_parallel.tsv", sep="\t", names=["src_text",
"tgt_text"])

print(df_mt.head())
```

5. Packaging Datasets as HDF5 (.h5) and Other Formats

You can package processed datasets into different formats depending on your pipeline:
- CSV: Easy to inspect, widely supported.
- JSONL: One JSON per line, good for streaming.
- HDF5 (.h5): Efficient binary format for large arrays; ideal for pre-tokenized datasets.
- TFRecord: Efficient for TensorFlow pipelines.

5.1 Save Tokenized Dataset as HDF5 (.h5)

```
import h5py
import numpy as np

# Example: assume you already have numpy arrays for tokenized inputs
# X_train: shape (num_train, max_len)
# y_train: shape (num_train,)
# likewise X_valid, y_valid, X_test, y_test

with h5py.File("bengali_sentiment_dataset.h5", "w") as f:
   f.create_dataset("X_train", data=X_train, compression="gzip")
   f.create_dataset("y_train", data=y_train, compression="gzip")
   f.create_dataset("X_valid", data=X_valid, compression="gzip")
   f.create_dataset("y_valid", data=y_valid, compression="gzip")
   f.create_dataset("X_test", data=X_test, compression="gzip")
   f.create_dataset("y_test", data=y_test, compression="gzip")

# Loading later:
with h5py.File("bengali_sentiment_dataset.h5", "r") as f:
   X_train = np.array(f["X_train"])
   y_train = np.array(f["y_train"])
```

5.2 Save Dataset with Pandas as HDF

```
# Save a DataFrame to HDF store using Pandas
train_df.to_hdf("bengali_dataset_store.h5", key="train", mode="w")
valid_df.to_hdf("bengali_dataset_store.h5", key="valid")
test_df.to_hdf("bengali_dataset_store.h5", key="test")

# Load later
train_df_loaded = pd.read_hdf("bengali_dataset_store.h5", key="train")
```

---

6. Ideal Dataset Size (Rules of Thumb)

These are general guidelines; actual needs depend on task complexity, label noise, and model size:

• Tiny experiment / prototype:
  - 1,000 – 5,000 labeled examples total
  - Useful for proving pipeline, not for production

• Small-to-medium production sentiment model:
  - 10,000 – 50,000 labeled examples (balanced across classes)
  - E.g. ~5k–15k per class for 3-way sentiment

• Strong production model / multiple domains:
  - 50,000 – 200,000+ labeled examples

• NER and complex sequence labeling tasks:
  - Often need more data because each sentence yields many labels
  - 20,000+ sentences with good entity coverage is a reasonable starting point

• Translation (MT) parallel corpora:
  - Basic MT: >= 100,000 sentence pairs
  - Robust MT: millions of sentence pairs

Always monitor validation performance; if the model is underfitting/overfitting badly, adjust data size, model complexity, or regularization.

7. Validation & Packaging Checklist

Before publishing / deploying:
- [ ] Data is UTF-8, normalized, and free from major encoding issues
- [ ] Train/validation/test split is strict (no leakage)
- [ ] Label distribution across splits is reasonable (no extreme imbalance)
- [ ] Model has been evaluated on validation and final test sets
- [ ] Metrics: accuracy, precision, recall, F1 (for classification); token-level F1 for NER; BLEU/COMET for MT
- [ ] Dataset formats exported as needed: CSV/JSONL + HDF5 if required
- [ ] Model saved in standard formats (.h5 for Keras, .pt for PyTorch) with accompanying label maps and tokenizer configs