```
from datasets import load_dataset

ds = load_dataset("nielsr/funsd")
```

```
/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/se
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(
README.md: 100%                                                 755/755 [00:00<00:00, 42.1kB/s]
data/train-00000-of-00001.parquet: 100%                         12.3M/12.3M [00:02<00:00, 7.33MB/s]
data/test-00000-of-00001.parquet: 100%                          4.38M/4.38M [00:02<00:00, 2.06MB/s]
Generating train split: 100%                                    149/149 [00:00<00:00, 772.34 examples/s]
Generating test split: 100%                                     50/50 [00:00<00:00, 942.38 examples/s]
```

```
pip install transformers datasets torch pillow pytesseract
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.57.3)
Requirement already satisfied: datasets in /usr/local/lib/python3.12/dist-packages (4.0.0)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.9.0+cpu)
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-packages (11.3.0)
Collecting pytesseract
  Downloading pytesseract-0.3.13-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (
```

```
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transfor
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transf
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from trans
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from datas
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (from datasets) (3.6.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.12/dist-packages (from data
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.1
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch) (3
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.12/dist-packages (fs
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from hug
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->t
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from reque
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from reque
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pa
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->data
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->da
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from a
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohtt
```

```
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aioh
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.12/dist-packages (from aiohtt
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil
Downloading pytesseract-0.3.13-py3-none-any.whl (14 kB)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.13
```

```
from datasets import load_dataset

dataset = load_dataset("nielsr/funsd")

train_dataset = dataset["train"]
test_dataset = dataset["test"]

print(train_dataset[0])
```

```
{'id': '0', 'words': ['R&D', ':', 'Suggestion:', 'Date:', 'Licensee', 'Yes', 'No', '597005708', 'R&D', '(
```

```
LABELS = [
    "O",
    "B-QUESTION", "I-QUESTION",
    "B-ANSWER", "I-ANSWER",
    "B-HEADER", "I-HEADER"
]

label2id = {label: i for i, label in enumerate(LABELS)}
id2label = {i: label for label, i in label2id.items()}
```

```
from transformers import LayoutLMv3Processor
import torch
processor = LayoutLMv3Processor.from_pretrained(
    "microsoft/layoutlmv3-base",
    apply_ocr=False
)

def preprocess(example):
    image = example["image"].convert("RGB")
    encoding = processor(
        image,
        example["words"],
        boxes=example["bboxes"],
        word_labels=example["ner_tags"],
        truncation=True,
        padding="max_length",
        return_tensors="pt"
    )

    # The processor returns tensors with a batch dimension of 1 for a single input.
```

```
    # We need to remove this for each individual example before batching.
    for k, v in encoding.items():
        if isinstance(v, torch.Tensor) and v.ndim > 1 and v.shape[0] == 1:
            encoding[k] = v.squeeze(0)

    return encoding

encoded_train = train_dataset.map(
    preprocess,
    batched=False,
    remove_columns=train_dataset.column_names
)

encoded_test = test_dataset.map(
    preprocess,
    batched=False,
    remove_columns=test_dataset.column_names
)
```

```
Map: 100%  149/149 [00:05<00:00, 40.66 examples/s]

Map: 100%  50/50 [00:01<00:00, 20.76 examples/s]
```

```
from transformers import LayoutLMv3ForTokenClassification
from transformers import Trainer, TrainingArguments, DefaultDataCollator

model = LayoutLMv3ForTokenClassification.from_pretrained(
    "microsoft/layoutlmv3-base",
    num_labels=len(LABELS),
    id2label=id2label,
    label2id=label2id
```

```
)

training_args = TrainingArguments(
    output_dir="./funsd_model",
    learning_rate=2e-5,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    num_train_epochs=3,
    eval_strategy="steps",
    logging_steps=100,
    save_steps=500,
    save_total_limit=2,
    report_to="none"
)

data_collator = DefaultDataCollator()

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=encoded_train,
    eval_dataset=encoded_test,
    data_collator=data_collator # Using DefaultDataCollator for preprocessed data
)

trainer.train()
```

```
Some weights of LayoutLMv3ForTokenClassification were not initialized from the model checkpoint at micros
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and infer
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:668: UserWarning: 'pin_memory' arg
  warnings.warn(warn_msg)
/usr/local/lib/python3.12/dist-packages/transformers/modeling_utils.py:1621: FutureWarning: The `device`
  warnings.warn(
```

warnings.warn(

[225/225 1:36:49, Epoch 3/3]

Step	Training Loss	Validation Loss
100	1.112700	0.671565
200	0.661500	0.585828

```
TrainOutput(global_step=225, training_loss=0.8458343972100152, metrics={'train_runtime': 5841.0771,
'train_samples_per_second': 0.077, 'train_steps_per_second': 0.039, 'total_flos': 117831588424704.0,
'train_loss': 0.8458343972100152, 'epoch': 3.0})
```

```
import torch

def infer(example):
    model.eval()

    # Convert the image to RGB before processing
    image_rgb = example["image"].convert("RGB")

    encoding = processor(
        image_rgb,
        example["words"],
        boxes=example["bboxes"],
        return_tensors="pt",
        truncation=True,
        padding="max_length"
    )

    with torch.no_grad():
        outputs = model(**encoding)

    predictions = outputs.logits.argmax(-1).squeeze().tolist()

    results = []
    for word, label_id in zip(example["words"], predictions):
        results.append({
            "word": word,
            "label": id2label[label_id]
        })

    return results
```

```
import torch
from PIL import Image
import numpy as np

def infer(example):
    model.eval()

    image = example["image"]

    # ensure image is RGB (3 channels)
    if isinstance(image, Image.Image):
        image = image.convert("RGB")
    elif isinstance(image, np.ndarray):
        if image.ndim == 2:  # grayscale
            image = np.stack([image]*3, axis=-1)
        image = Image.fromarray(image).convert("RGB")

    encoding = processor(
        image,
        example["words"],
        boxes=example["bboxes"],
        return_tensors="pt",
        truncation=True,
        padding="max_length"
    )

    with torch.no_grad():
        outputs = model(**encoding)

    predictions = outputs.logits.argmax(-1).squeeze().tolist()
```

```
    results = []
    for word, label_id in zip(example["words"], predictions):
        results.append((word, id2label[label_id]))

    return results



sample = test_dataset[0]
predictions = infer(sample)

for w, l in predictions:
    print(w, "->", l)
```

```
/usr/local/lib/python3.12/dist-packages/transformers/modeling_utils.py:1621: FutureWarning: The `device
  warnings.warn(
TO: -> O
DATE: -> B-ANSWER
3 -> B-ANSWER
Fax: -> B-ANSWER
NOTE: -> B-ANSWER
82092117 -> I-ANSWER
614 -> B-HEADER
-466 -> O
-5087 -> O
Dec -> O
10 -> B-ANSWER
'98 -> I-ANSWER
17 -> O
:46 -> O
P. -> O
01 -> O
ATT. -> O
```

```
GEN. -> O
ADMIN. -> I-HEADER
OFFICE -> O
Attorney -> I-HEADER
General -> I-HEADER
Betty -> O
D. -> O
Montgomery -> O
CONFIDENTIAL -> O
FACSIMILE -> O
TRANSMISSION -> O
COVER -> O
SHEET -> O
(614) -> O
466- -> O
5087 -> O
FAX -> O
NO. -> O
George -> I-QUESTION
Baroody -> I-QUESTION
(336) -> I-QUESTION
335- -> I-QUESTION
7392 -> I-QUESTION
FAX -> I-QUESTION
NUMBER: -> B-HEADER
PHONE -> I-HEADER
NUMBER: -> B-HEADER
(336) -> I-HEADER
335- -> I-HEADER
7363 -> I-HEADER
NUMBER -> B-QUESTION
OF -> I-QUESTION
PAGES -> I-QUESTION
INCLUDING -> I-QUESTION
COVER -> I-QUESTION
```

```
SHEET: -> I-QUESTION
June -> I-QUESTION
Flynn -> I-QUESTION
for -> I-OUESTION
```