Notebook

October 20, 2025

[ ]:
```
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.3.2-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.17.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-
packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading
xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in
/usr/local/lib/python3.11/dist-packages (from
fsspec[http]<=2024.12.0,>=2023.1.0->datasets) (2024.10.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.11.13)
Requirement already satisfied: huggingface-hub>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.28.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.4.6)
Requirement already satisfied: aiosignal>=1.1.2 in
```

```
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0->datasets)
(4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
Downloading datasets-3.3.2-py3-none-any.whl (485 kB)
                        485.4/485.4 kB
10.9 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
                        116.3/116.3 kB
9.6 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
                        143.5/143.5 kB
9.8 MB/s eta 0:00:00
Downloading
xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                        194.8/194.8 kB
14.3 MB/s eta 0:00:00
Installing collected packages: xxhash, dill, multiprocess, datasets
Successfully installed datasets-3.3.2 dill-0.3.8 multiprocess-0.70.16
```

```
xxhash-3.5.0
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
from datasets import load_dataset
from datasets import Dataset
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from transformers import AdamWeightDecay
from sklearn.preprocessing import LabelEncoder
```

[ ]:
```
dataset = load_dataset('bitext/
  ↪Bitext-retail-banking-llm-chatbot-training-dataset')

print(dataset)
```

```
/usr/local/lib/python3.11/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(

README.md:   0%|          | 0.00/11.7k [00:00<?, ?B/s]

(…)ing-llm-chatbot-training-dataset.parquet:   0%|          | 0.00/7.87M [00:00<?
  ↪, ?B/s]

Generating train split:   0%|          | 0/25545 [00:00<?, ? examples/s]

DatasetDict({
    train: Dataset({
        features: ['tags', 'instruction', 'category', 'intent', 'response'],
        num_rows: 25545
    })
})
```

[ ]:
```
df = pd.DataFrame(dataset['train'])
df.head()
```

[ ]:
```
     tags                                          instruction category  \
0   BCIPZ   I would like to acivate a card, can you help me?     CARD
```

```
1   BCILZ  I have to activate an Visa online, how can I d…     CARD
2  BCIPQZ        I'd like to actiate a card where do i do it     CARD
3  BCLPQZ  I'd likke to activate a visa on mobile i need …     CARD
4  BCILPZ  I would ilke to activate a credit card online,…     CARD

          intent                                              response
0  activate_card  I'm here to assist you with that! Activating y…
1  activate_card  I'm here to assist you with activating your {{…
2  activate_card  I can help you with that! Activating your card…
3  activate_card  I'm here to assist you with activating your {{…
4  activate_card  I'm here to assist you with activating your cr…
```

0.1 Intent Classification

[ ]:
```
df['intent'].unique()
```

[ ]:
```
array(['activate_card', 'activate_card_international_usage',
       'apply_for_loan', 'apply_for_mortgage', 'block_card',
       'cancel_card', 'cancel_loan', 'cancel_mortgage', 'cancel_transfer',
       'check_card_annual_fee', 'check_current_balance_on_card',
       'check_fees', 'check_loan_payments', 'check_mortgage_payments',
       'check_recent_transactions', 'close_account', 'create_account',
       'customer_service', 'dispute_ATM_withdrawal', 'find_ATM',
       'find_branch', 'get_password', 'human_agent', 'make_transfer',
       'recover_swallowed_card', 'set_up_password'], dtype=object)
```

[ ]:
```
encoder = LabelEncoder()
df['intent_label'] = encoder.fit_transform(df['intent'])
```

[ ]:
```
intent_mapping = dict(zip(encoder.transform(encoder.classes_), encoder.
 ↪classes_))
intent_mapping
```

[ ]:
```
{0: 'activate_card',
 1: 'activate_card_international_usage',
 2: 'apply_for_loan',
 3: 'apply_for_mortgage',
 4: 'block_card',
 5: 'cancel_card',
 6: 'cancel_loan',
 7: 'cancel_mortgage',
 8: 'cancel_transfer',
 9: 'check_card_annual_fee',
 10: 'check_current_balance_on_card',
 11: 'check_fees',
 12: 'check_loan_payments',
 13: 'check_mortgage_payments',
```

```
 14: 'check_recent_transactions',
 15: 'close_account',
 16: 'create_account',
 17: 'customer_service',
 18: 'dispute_ATM_withdrawal',
 19: 'find_ATM',
 20: 'find_branch',
 21: 'get_password',
 22: 'human_agent',
 23: 'make_transfer',
 24: 'recover_swallowed_card',
 25: 'set_up_password'}
```

[ ]:
```
unique_intents = df['intent_label'].nunique()
```

[ ]:
```
df['instruction_word_count'] = df['instruction'].apply(lambda x: len(str(x).
 ↪split()))
max(df['instruction_word_count'])
```

[ ]: 23

[ ]:
```
from transformers import DistilBertTokenizer

bert_tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]
```

[ ]:
```
tokenized_datasets = df[['instruction', 'intent_label']].copy()
train_df, test_df = train_test_split(tokenized_datasets, test_size=0.2,␣
 ↪random_state=42)
```

[ ]:
```
train_df = Dataset.from_pandas(train_df)
test_df = Dataset.from_pandas(test_df)
```

[ ]:
```
train_df
```

[ ]:
```
Dataset({
    features: ['instruction', 'intent_label', '__index_level_0__'],
    num_rows: 20436
})
```

[ ]:
```
def tokenize(batch):
```

```
    return bert_tokenizer(batch['instruction'], padding='max_length',␣
 ↪truncation=True, max_length=40)

train_df = train_df.map(tokenize, batched=True)
test_df = test_df.map(tokenize, batched=True)
```

```
Map:   0%|          | 0/20436 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/5109 [00:00<?, ? examples/s]
```

[ ]:
```
train_df
```

[ ]:
```
Dataset({
    features: ['instruction', 'intent_label', '__index_level_0__', 'input_ids',
'attention_mask'],
    num_rows: 20436
})
```

[ ]:
```
intent_train_df = train_df.to_tf_dataset(
    columns=['input_ids', 'attention_mask'],
    label_cols=['intent_label'],
    shuffle=True,
    batch_size=32
)

intent_test_df = test_df.to_tf_dataset(
    columns=['input_ids', 'attention_mask'],
    label_cols=['intent_label'],
    shuffle=False,
    batch_size=32
)
```

```
/usr/local/lib/python3.11/dist-packages/datasets/arrow_dataset.py:405:
FutureWarning: The output of `to_tf_dataset` will change when a passing single
element list for `labels` or `columns` in the next datasets version. To return a
tuple structure rather than dict, pass a single string.
Old behaviour: columns=['a'], labels=['labels'] -> (tf.Tensor, tf.Tensor)
             : columns='a', labels='labels' -> (tf.Tensor, tf.Tensor)
New behaviour: columns=['a'],labels=['labels'] -> ({'a': tf.Tensor}, {'labels':
tf.Tensor})
             : columns='a', labels='labels' -> (tf.Tensor, tf.Tensor)
  warnings.warn(
```

[ ]:
```
intent_train_df
```

[ ]:
```
<_PrefetchDataset element_spec=({'input_ids': TensorSpec(shape=(None, 40),
dtype=tf.int64, name=None), 'attention_mask': TensorSpec(shape=(None, 40),
dtype=tf.int64, name=None)}, TensorSpec(shape=(None,), dtype=tf.int64,
name=None))>
```

[ ]:
```
intent_test_df
```

[ ]:
```
<_PrefetchDataset element_spec=({'input_ids': TensorSpec(shape=(None, 40),
dtype=tf.int64, name=None), 'attention_mask': TensorSpec(shape=(None, 40),
dtype=tf.int64, name=None)}, TensorSpec(shape=(None,), dtype=tf.int64,
name=None))>
```

[ ]:
```
for batch in intent_train_df.take(1):
    print(batch)
```

```
({'input_ids': <tf.Tensor: shape=(32, 40), dtype=int64, numpy=
array([[ 101, 1045, 2288, …,    0,    0,    0],
       [ 101, 2043, 2003, …,    0,    0,    0],
       [ 101, 1045, 2031, …,    0,    0,    0],
       …,
       [ 101, 2393, 2033, …,    0,    0,    0],
       [ 101, 1045, 1005, …,    0,    0,    0],
       [ 101, 1045, 2031, …,    0,    0,    0]])>, 'attention_mask':
<tf.Tensor: shape=(32, 40), dtype=int64, numpy=
array([[1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       …,
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0]])>}, <tf.Tensor: shape=(32,), dtype=int64, numpy=
array([12, 13,  0, 10, 10, 12, 25, 19,  7, 24, 10,  9,  6, 11, 24,  7, 18,
       14, 10,  1, 17, 15, 17,  2, 14, 12,  8, 22, 25, 12,  9, 10])>)
```

[ ]:
```
from transformers import TFDistilBertForSequenceClassification

bert_model = TFDistilBertForSequenceClassification.
 ↪from_pretrained('distilbert-base-uncased', num_labels=unique_intents)
```

```
model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]

Some weights of the PyTorch model were not used when initializing the TF 2.0
model TFDistilBertForSequenceClassification: ['vocab_transform.weight',
'vocab_transform.bias', 'vocab_layer_norm.weight', 'vocab_layer_norm.bias',
'vocab_projector.bias']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from a PyTorch model trained on another task or with another architecture (e.g.
initializing a TFBertForSequenceClassification model from a BertForPreTraining
model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from a PyTorch model that you expect to be
exactly identical (e.g. initializing a TFBertForSequenceClassification model
from a BertForSequenceClassification model).
Some weights or buffers of the TF 2.0 model
```

```
TFDistilBertForSequenceClassification were not initialized from the PyTorch
model and are newly initialized: ['pre_classifier.weight',
'pre_classifier.bias', 'classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[ ]:
```
optimizer = AdamWeightDecay(learning_rate=5e-5, weight_decay_rate=0.01)

bert_model.compile(
    optimizer=optimizer,
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

[ ]:
```
intent_history = bert_model.fit(intent_train_df,␣
 ↪validation_data=intent_test_df, epochs=3)
```

```
Epoch 1/3
639/639 [==============================] - 118s 161ms/step - loss: 0.3614 -
accuracy: 0.9508 - val_loss: 0.0092 - val_accuracy: 0.9982
Epoch 2/3
639/639 [==============================] - 107s 167ms/step - loss: 0.0097 -
accuracy: 0.9985 - val_loss: 0.0057 - val_accuracy: 0.9990
Epoch 3/3
639/639 [==============================] - 106s 166ms/step - loss: 0.0050 -
accuracy: 0.9989 - val_loss: 0.0199 - val_accuracy: 0.9959
```

[ ]:
```
import numpy as np
def get_intent(user_input):
  input = bert_tokenizer(user_input, return_tensors="tf", padding=True,␣
 ↪truncation=True)
  output = bert_model(input)

  logits = output.logits
  predicted_label = np.argmax(logits.numpy(), axis=1).item()
  return intent_mapping[predicted_label]

user = 'I am very annoyed when will I get my card'
print(get_intent(user))
```

```
cancel_card
```

[ ]:
```
bert_model.save_pretrained('/content/distilbert_model_tf1', save_format='tf')
bert_tokenizer.save_pretrained('/content/distilbert_model_tf1')
```

[ ]:
```
('/content/distilbert_model_tf1/tokenizer_config.json',
 '/content/distilbert_model_tf1/special_tokens_map.json',
 '/content/distilbert_model_tf1/vocab.txt',
 '/content/distilbert_model_tf1/added_tokens.json')
```

[ ]:
```
from transformers import TFAutoModelForSequenceClassification, AutoTokenizer

model_path = '/content/distilbert_model_tf1'

model = TFAutoModelForSequenceClassification.from_pretrained(model_path)

tokenizer = AutoTokenizer.from_pretrained(model_path)

print("Model and tokenizer loaded successfully!")
```

```
Some layers from the model checkpoint at /content/distilbert_model_tf1 were not
used when initializing TFDistilBertForSequenceClassification: ['dropout_19']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at /content/distilbert_model_tf1 and are newly initialized:
['dropout_39']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
Model and tokenizer loaded successfully!
```

[ ]:
```
sample_text = "What is my account balance?"

inputs = tokenizer(sample_text, return_tensors='tf', padding=True,␣
 ↪truncation=True, max_length=40)

outputs = model(inputs)

logits = outputs.logits

predicted_class = tf.argmax(logits, axis=1).numpy()[0]

print("Predicted class index:", intent_mapping[predicted_class])
```

```
Predicted class index: check_recent_transactions
```

0.2 T5

[ ]:
```
df['combined_input'] = 'Intent:' + df['intent'] + ' Context:' +␣
 ↪df['instruction']
df.head()
```

[ ]:

```
     tags                                        instruction category  \
0   BCIPZ   I would like to acivate a card, can you help me?     CARD
1   BCILZ  I have to activate an Visa online, how can I d…     CARD
2  BCIPQZ        I'd like to actiate a card where do i do it     CARD
3  BCLPQZ  I'd likke to activate a visa on mobile i need …     CARD
4  BCILPZ  I would ilke to activate a credit card online,…     CARD

           intent                                              response  \
0  activate_card  I'm here to assist you with that! Activating y…
1  activate_card  I'm here to assist you with activating your {{…
2  activate_card  I can help you with that! Activating your card…
3  activate_card  I'm here to assist you with activating your {{…
4  activate_card  I'm here to assist you with activating your cr…

   intent_label  instruction_word_count  \
0             0                      11
1             0                      12
2             0                      11
3             0                      11
4             0                      13

                                    combined_input
0  Intent:activate_card Context:I would like to a…
1  Intent:activate_card Context:I have to activat…
2  Intent:activate_card Context:I'd like to actia…
3  Intent:activate_card Context:I'd likke to acti…
4  Intent:activate_card Context:I would ilke to a…
```

[ ]:

```
df = df.drop(columns=['intent', 'instruction', 'tags', 'category',␣
 ↪'intent_label', 'instruction_word_count'])
df.head()
```

[ ]:

```
                                          response  \
0  I'm here to assist you with that! Activating y…
1  I'm here to assist you with activating your {{…
2  I can help you with that! Activating your card…
3  I'm here to assist you with activating your {{…
4  I'm here to assist you with activating your cr…

                                    combined_input
0  Intent:activate_card Context:I would like to a…
1  Intent:activate_card Context:I have to activat…
2  Intent:activate_card Context:I'd like to actia…
3  Intent:activate_card Context:I'd likke to acti…
4  Intent:activate_card Context:I would ilke to a…
```

[ ]:
```
X = df['combined_input']
y = df['response']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[ ]:
```
df['word_size'] = df['response'].apply(lambda x: len(str(x).split()))
max(df['word_size'])
```

[ ]: 430

[ ]:
```
from transformers import T5Tokenizer

t5_tokenizer = T5Tokenizer.from_pretrained('t5-small')
```

```
tokenizer_config.json:   0%|          | 0.00/2.32k [00:00<?, ?B/s]

spiece.model:   0%|          | 0.00/792k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.39M [00:00<?, ?B/s]

You are using the default legacy behaviour of the <class
'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, and
simply means that the `legacy` (previous) behavior will be used so nothing
changes for you. If you want to use the new behaviour, set `legacy=False`. This
should only be set if you understand what it means, and thoroughly read the
reason why this was added as explained in
https://github.com/huggingface/transformers/pull/24565
```

[ ]:
```
def tokenize_data(tokenizer, inputs, outputs, batch_size=32, max_length=440):
    # Tokenize inputs
    input_tokens = tokenizer(
        list(inputs),
        padding='max_length',
        truncation=True,
        max_length=max_length,
        return_tensors='tf'
    )

    # Tokenize outputs (labels)
    output_tokens = tokenizer(
        list(outputs),
        padding='max_length',
        truncation=True,
        max_length=max_length,
        return_tensors='tf'
    )

    return input_tokens, output_tokens
```

```
X_train_token, y_train_tokens = tokenize_data(t5_tokenizer, X_train, y_train)
X_test_token, y_test_tokens = tokenize_data(t5_tokenizer, X_train, y_train)
```

[ ]:
```
def shift_labels(labels, pad_token_id):

    decoder_input_ids = tf.concat(
        [tf.fill((tf.shape(labels)[0], 1), pad_token_id), labels[:, :-1]],
        axis=1
    )
    return decoder_input_ids
```

[ ]:
```
# Create attention masks: 1 for non-padding tokens, 0 for padding tokens
def create_attention_mask(input_ids, pad_token_id):
    return tf.where(
        tf.equal(input_ids, pad_token_id),
        tf.constant(0, dtype=tf.int32),
        tf.constant(1, dtype=tf.int32)
    )

# Get padding token ID
pad_token_id = t5_tokenizer.pad_token_id

train_dataset = {
    "input_ids": X_train_token['input_ids'],
    "attention_mask": create_attention_mask(X_train_token['input_ids'],␣
 ↪pad_token_id),
    "labels": y_train_tokens['input_ids'],
    "decoder_input_ids": shift_labels(y_train_tokens['input_ids'],␣
 ↪pad_token_id),
    "decoder_attention_mask":␣
 ↪create_attention_mask(y_train_tokens['input_ids'], pad_token_id)
}

test_dataset = {
    "input_ids": X_test_token['input_ids'],
    "attention_mask": create_attention_mask(X_test_token['input_ids'],␣
 ↪pad_token_id),
    "labels": y_test_tokens['input_ids'],
    "decoder_input_ids": shift_labels(y_test_tokens['input_ids'], pad_token_id),
    "decoder_attention_mask": create_attention_mask(y_test_tokens['input_ids'],␣
 ↪pad_token_id)
}
```

[ ]:
```
train_dataset = tf.data.Dataset.from_tensor_slices(train_dataset)
train_dataset = train_dataset.shuffle(1000).batch(9).prefetch(tf.data.AUTOTUNE)

test_dataset = tf.data.Dataset.from_tensor_slices(test_dataset)
test_dataset = test_dataset.shuffle(1000).batch(9).prefetch(tf.data.AUTOTUNE)
```

[ ]:
```
for i in train_dataset.take(1):
  print(i)
```

```
{'input_ids': <tf.Tensor: shape=(9, 440), dtype=int32, numpy=
array([[  86, 4669,   10, …,    0,    0,    0],
       [  86, 4669,   10, …,    0,    0,    0],
       [  86, 4669,   10, …,    0,    0,    0],
       …,
       [  86, 4669,   10, …,    0,    0,    0],
       [  86, 4669,   10, …,    0,    0,    0],
       [  86, 4669,   10, …,    0,    0,    0]], dtype=int32)>,
'attention_mask': <tf.Tensor: shape=(9, 440), dtype=int32, numpy=
array([[1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       …,
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0]], dtype=int32)>, 'labels': <tf.Tensor: shape=(9,
440), dtype=int32, numpy=
array([[  27,   54, 2094, …,    0,    0,    0],
       [  27,   31,  195, …,    0,    0,    0],
       [  27,   31,   51, …,    0,    0,    0],
       …,
       [  27,   31,   51, …,    0,    0,    0],
       [  27,   31,   51, …,    0,    0,    0],
       [  27,   54,  199, …,    0,    0,    0]], dtype=int32)>,
'decoder_input_ids': <tf.Tensor: shape=(9, 440), dtype=int32, numpy=
array([[ 0, 27, 54, …,  0,  0,  0],
       [ 0, 27, 31, …,  0,  0,  0],
       [ 0, 27, 31, …,  0,  0,  0],
       …,
       [ 0, 27, 31, …,  0,  0,  0],
       [ 0, 27, 31, …,  0,  0,  0],
       [ 0, 27, 54, …,  0,  0,  0]], dtype=int32)>, 'decoder_attention_mask':
<tf.Tensor: shape=(9, 440), dtype=int32, numpy=
array([[1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
       …,
       [1, 1, 1, …, 0, 0, 0],
       [1, 1, 1, …, 0, 0, 0],
```

```
        [1, 1, 1, …, 0, 0, 0]], dtype=int32)>}
```

[ ]:
```
from transformers import TFAutoModelForSeq2SeqLM

t5_model = TFAutoModelForSeq2SeqLM.from_pretrained('t5-small')
```

```
config.json:   0%|          | 0.00/1.21k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/242M [00:00<?, ?B/s]

All PyTorch model weights were used when initializing
TFT5ForConditionalGeneration.

All the weights of TFT5ForConditionalGeneration were initialized from the
PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFT5ForConditionalGeneration for predictions without further
training.
```

[ ]:
```
from tensorflow.keras.mixed_precision import set_global_policy

# Enable mixed precision
set_global_policy('mixed_float16')
```

[ ]:
```
optimizer = AdamWeightDecay(learning_rate=5e-5, weight_decay_rate=0.01)

t5_model.compile(
        optimizer=optimizer,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=['accuracy']
)
```

[ ]:
```
history = t5_model.fit(train_dataset, validation_data=test_dataset, epochs=2)
```

```
Epoch 1/2
2271/2271 [==============================] - 2466s 1s/step - loss: 1.1329 -
accuracy: 0.7674 - val_loss: 0.6969 - val_accuracy: 0.8349
Epoch 2/2
2271/2271 [==============================] - 2444s 1s/step - loss: 0.7441 -
accuracy: 0.8254 - val_loss: 0.5777 - val_accuracy: 0.8580
```

[ ]:
```
results = t5_model.evaluate(test_dataset)
print(f"Test Loss: {results[0]}")
print(f"Test Accuracy: {results[1]}")
```

[ ]:
```
def generate_response(intent, context):
    # Format the input
    input_text = f"intent: {intent} context: {context}"
```

```
    inputs = t5_tokenizer(input_text, return_tensors="tf", padding=True,␣
 ↪truncation=True, max_length=440)

    outputs = t5_model.generate(
    inputs["input_ids"],
    max_length=430,
    num_beams=5,
    repetition_penalty=2.0,
    no_repeat_ngram_size=3,
    early_stopping=True
    )

    # Decode the output
    response = t5_tokenizer.decode(outputs[0], skip_special_tokens=True)
    return response


context = input('Enter query: ')
intent = get_intent(context)
print(intent)

print("Generated Response:", generate_response(intent, context))
```

```
Enter query: I can apply for a loan can you help what kind of loan you provide
apply_for_loan
Generated Response: I'd be happy to assist you with applying for a loan.
Applying for the loan is a simple process. Here's what you need to do: 1.
Contact our customer support team at Customer Support Phone Number or visit our
website at "Loan Support Website URL. 2. Provide them with your loan details,
such as your name, account number, and any other relevant information they may
require. 3. They will guide you through the application process and provide you
with the necessary information. 4. If you have any specific questions or need
further assistance, feel free to let me know.
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

This notebook was converted with convert.ploomber.io