```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Collecting transformers
  Downloading transformers-4.24.0-py3-none-any.whl (5.5 MB)
     |████████████████████████████████| 5.5 MB 18.8 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dis
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-pack
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x
     |████████████████████████████████| 7.6 MB 9.8 MB/s
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.11.1-py3-none-any.whl (182 kB)
     |████████████████████████████████| 182 kB 19.4 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/di
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/pyth
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/di
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/l
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.11.1 tokenizers-0.13.2 transformers-4.2
```

```
import torch
```

```
!pip uninstall torch-scatter torch-sparse torch-geometric torch-cluster  --y
!pip install torch-scatter -f https://data.pyg.org/whl/torch-{torch.__version__}.html
!pip install torch-sparse -f https://data.pyg.org/whl/torch-{torch.__version__}.html
!pip install torch-cluster -f https://data.pyg.org/whl/torch-{torch.__version__}.html
!pip install git+https://github.com/pyg-team/pytorch_geometric.git
```

```
WARNING: Skipping torch-scatter as it is not installed.
WARNING: Skipping torch-sparse as it is not installed.
WARNING: Skipping torch-geometric as it is not installed.
WARNING: Skipping torch-cluster as it is not installed.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Looking in links: https://data.pyg.org/whl/torch-1.12.1+cu113.html
Collecting torch-scatter
  Downloading https://data.pyg.org/whl/torch-1.12.0%2Bcu113/torch_scatter-2.1.0%
     |████████████████████████████████| 8.9 MB 21.5 MB/s
Installing collected packages: torch-scatter
Successfully installed torch-scatter-2.1.0+pt112cu113
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Looking in links: https://data.pyg.org/whl/torch-1.12.1+cu113.html
Collecting torch-sparse
  Downloading https://data.pyg.org/whl/torch-1.12.0%2Bcu113/torch_sparse-0.6.15%
     |████████████████████████████████| 3.5 MB 18.3 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in /usr/local/lib/python3.7
Installing collected packages: torch-sparse
Successfully installed torch-sparse-0.6.15+pt112cu113
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Looking in links: https://data.pyg.org/whl/torch-1.12.1+cu113.html
Collecting torch-cluster
  Downloading https://data.pyg.org/whl/torch-1.12.0%2Bcu113/torch_cluster-1.6.0%
     |████████████████████████████████| 2.5 MB 17.2 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in /usr/local/lib/python3.7
Installing collected packages: torch-cluster
Successfully installed torch-cluster-1.6.0+pt112cu113
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Collecting git+https://github.com/pyg-team/pytorch_geometric.git
  Cloning https://github.com/pyg-team/pytorch_geometric.git to /tmp/pip-req-buil
  Running command git clone -q https://github.com/pyg-team/pytorch_geometric.git
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-pac
Collecting psutil>=5.8.0
  Downloading psutil-5.9.4-cp36-abi3-manylinux_2_12_x86_64.manylinux2010_x86_64.
     |████████████████████████████████| 280 kB 27.5 MB/s
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/di
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/l
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/
Building wheels for collected packages: torch-geometric
  Building wheel for torch-geometric (setup.py) ... done
  Created wheel for torch-geometric: filename=torch_geometric-2.1.0-py3-none-any
  Stored in directory: /tmp/pip-ephem-wheel-cache-ybyr5rp8/wheels/85/c9/07/7936e
Successfully built torch-geometric
Installing collected packages: psutil, torch-geometric
```

```
from transformers import TapasTokenizer, TapasForQuestionAnswering
import pandas as pd
```

```
  Successfully uninstalled psutil-5.4.8
```

```
tokenizer = TapasTokenizer.from_pretrained("google/tapas-base-finetuned-wtq")
model = TapasForQuestionAnswering.from_pretrained("google/tapas-base-finetuned-wtq",
```

You must restart the runtime in order to use newlv installed versions.

```
from transformers import AutoTokenizer, TFAutoModelForTableQuestionAnswering

tokenizer2 = AutoTokenizer.from_pretrained("google/tapas-mini-finetuned-wtq")

model2 = TFAutoModelForTableQuestionAnswering.from_pretrained("google/tapas-mini-fine
```

```
Downloading: 100%  490/490 [00:00<00:00, 14.6kB/s]

Downloading: 100%  1.65k/1.65k [00:00<00:00, 51.4kB/s]

Downloading: 100%  262k/262k [00:00<00:00, 5.00MB/s]

Downloading: 100%  154/154 [00:00<00:00, 3.79kB/s]

Downloading: 100%  45.9M/45.9M [00:00<00:00, 52.7MB/s]

Some layers from the model checkpoint at google/tapas-mini-finetuned-wtq were nc
- This IS expected if you are initializing TFTapasForQuestionAnswering from the
- This IS NOT expected if you are initializing TFTapasForQuestionAnswering from
Some layers of TFTapasForQuestionAnswering were not initialized from the model c
You should probably TRAIN this model on a down-stream task to be able to use it
```

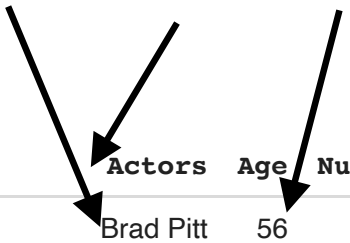
```
data = {
    "Actors": ["Brad Pitt", "Leonardo Di Caprio", "George Clooney"],
    "Age": ["56", "45", "59"],
    "Number of movies": ["87", "53", "34"],
    "Context":["Albania, on Southeastern Europe’s Balkan Peninsula, is a small countr
               "Albert Einstein was a German-born theoretical physicist, widely ackno
               "Apple Inc. is an American multinational technology company headquarte
}
table = pd.DataFrame.from_dict(data)
queries = ["Where is albania?","Who's 56 years old?", "How old is Brad Pitt?"]


inputs = tokenizer2(table=table, queries=queries, padding="max_length", return_tensor
outputs = model(**inputs)


#tokenizer2.save_pretrained("/tapas")

torch.save(inputs, 'tensor.pt')


traced = torch.jit.trace(model,toke)
traced.save()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-18-ec02f8bec862> in <module>
----> 1 traced = torch.jit.trace(model,inputs[0])
      2 traced.save()

/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py
in __getitem__(self, item)
    240         else:
    241             raise KeyError(
--> 242                 "Indexing with integers (to access backend Encoding for
a given batch index) "
    243                 "is not available when using Python based tokenizers"
    244             )

KeyError: 'Indexing with integers (to access backend Encoding for a given batch
index) is not available when using Python based tokenizers'
```

```
logits = outputs.logits
logits_aggregation = outputs.logits_aggregation
```

```
tokenizer2.convert_logits_to_predictions(inputs, logits, logits_agg=logits_aggregatio
```

```
([[(0, 0)], [(0, 0)], [(0, 1)]], [0, 0, 0])
```



```
table
```

	Actors	Age	Number of movies	Context
0	Brad Pitt	56	87	Albania, on Southeastern Europe’s Balkan Penin...
1	Leonardo Di Caprio	45	53	Albert Einstein was a German-born theoretical ...
2	George Clooney	59	34	Apple Inc. is an American multinational techno...

```
import tensorflow as tf
```

```
#m5 = tf.keras.models.load_model("/tf_model.h5")

#converter = tf.lite.TFLiteConverter.from_keras_model(model2) # path to the SavedMode
converter = tf.lite.TFLiteConverter.from_saved_model("/content")

converter.target_spec.supported_ops = [
  tf.lite.OpsSet.TFLITE_BUILTINS, # enable TensorFlow Lite ops.
  tf.lite.OpsSet.SELECT_TF_OPS # enable TensorFlow ops.
]
```

```
tflite_model = converter.convert()

with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-9-01f6551fa585> in <module>
      2
      3 #converter = tf.lite.TFLiteConverter.from_keras_model(model2) # path to
----> 4 converter = tf.lite.TFLiteConverter.from_saved_model("/content")
      5
      6 converter.target_spec.supported_ops = [

                                 4 frames

/usr/local/lib/python3.7/dist-packages/tensorflow/python/saved_model/loader_impl
    114   else:
    115     raise IOError(
--> 116         f"SavedModel file does not exist at: {export_dir}{os.path.sep}"
    117         f"{{{constants.SAVED_MODEL_FILENAME_PBTXT}|"
    118         f"{constants.SAVED_MODEL_FILENAME_PB}}}")

OSError: SavedModel file does not exist at: /content/{saved_model.pbtxt|saved_mo
```

SEARCH STACK OVERFLOW

```
model2.save("/content", save_format="pt")
```

```
WARNING:absl:Found untraced functions such as compute_column_logits_layer_call_f
```

```
from transformers import TableQuestionAnsweringPipeline

tokenizer2.save_pretrained("./")
```

```
('./tokenizer_config.json',
 './special_tokens_map.json',
 './vocab.txt',
 './added_tokens.json')
```

```
from transformers import TapasConfig,TapasTokenizer,TapasForQuestionAnswering
import torch
config = TapasConfig.from_pretrained('google/tapas-mini-finetuned-wtq',from_pt=True)
model = TapasForQuestionAnswering.from_pretrained('google/tapas-mini-finetuned-wtq',
tokenizer=TapasTokenizer.from_pretrained("google/tapas-mini-finetuned-wtq",from_pt=Tr
import sys

outdir = sys.argv[1]

model.save_pretrained(outdir)
```

```
tokenizer.save_pretrained(outdir)
config.save_pretrained(outdir)
```

```
Downloading: 100%                    1.65k/1.65k [00:00<00:00, 25.2kB/s]
Downloading: 100%                    45.8M/45.8M [00:01<00:00, 44.2MB/s]
Downloading: 100%                    262k/262k [00:00<00:00, 783kB/s]
Downloading: 100%                    154/154 [00:00<00:00, 4.37kB/s]
Downloading: 100%                    490/490 [00:00<00:00, 11.2kB/s]
```

```
from transformers import pipeline


generator = pipeline(task="text-generation", model=model, tokenizer=tokenizer)
```

```
The model 'TapasForQuestionAnswering' is not supported for text-generation. Supp
```

```
generator.tokenizer.save_pretrained("./tokens")
```

```
('./tokens/tokenizer_config.json',
 './tokens/special_tokens_map.json',
 './tokens/vocab.txt',
 './tokens/added_tokens.json')
```

```
tokenizer2.save_pretrained("./test", legacy_format=False)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-8bcc34962170> in <module>
----> 1 tokenizer2.save_pretrained("./test", legacy_format=False)

 ------------------------------------ 1 frames ------------------------------------

/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py i
filename_prefix)
   2163         if legacy_format is False:
   2164             raise ValueError(
-> 2165                 "Only fast tokenizers (instances of PreTrainedTokenizerF
   2166             )
   2167

ValueError: Only fast tokenizers (instances of PreTrainedTokenizerFast) can be s

 SEARCH STACK OVERFLOW
```

```
#!/usr/bin/env python
#
```

```
# Copyright 2022 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may not use thi
# except in compliance with the License. A copy of the License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "LICENSE.txt" file accompanying this file. This file is distributed on an
# BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied. See the Li
# the specific language governing permissions and limitations under the License.

import logging
import os.path
import shutil
from argparse import Namespace

import torch
from huggingface_hub import hf_hub_download
from transformers import pipeline

from metadata import HuggingfaceMetadata
from shasum import sha1_sum
from zip_utils import zip_dir


class HuggingfaceConverter:

    def __init__(self):
        self.device = torch.device(
            "cuda:0" if torch.cuda.is_available() else "cpu")
        self.task = None
        self.application = None
        self.translator = None
        self.inputs = None
        self.outputs = None

    def save_model(self, model_info, args: Namespace, temp_dir: str):
        model_id = model_info.modelId
        if not os.path.exists(temp_dir):
            os.makedirs(temp_dir)

        hf_pipeline = self.load_model(model_id)
        # Save tokenizer.json to temp dir
        self.save_tokenizer(hf_pipeline, temp_dir)

        # Save config.json just for reference
        config = hf_hub_download(repo_id=model_id, filename="config.json")
        shutil.copyfile(config, os.path.join(temp_dir, "config.json"))

        # Save jit traced .pt file to temp dir
        include_types = False
```

```
    model_file = self.jit_trace_model(hf_pipeline, model_id, temp_dir,
                                      include_types)
    if not model_file:
        return False, "Failed to trace model", -1

    result, reason = self.verify_jit_model(hf_pipeline, model_file,
                                           include_types, args.cpu_only)
    if not result:
        include_types = True
        model_file = self.jit_trace_model(hf_pipeline, model_id, temp_dir,
                                          include_types)
        if not model_file:
            return False, reason, -1

        result, reason = self.verify_jit_model(hf_pipeline, model_file,
                                               include_types,
                                               args.cpu_only)
        if not result:
            return False, reason, -1

    size = self.save_to_model_zoo(model_info, args.output_dir, temp_dir,
                                  hf_pipeline, include_types)

    return True, None, size

@staticmethod
def save_tokenizer(hf_pipeline, temp_dir: str):
    hf_pipeline.tokenizer.save_pretrained(temp_dir)
    # only keep tokenizer.json file
    for path in os.listdir(temp_dir):
        if path != "tokenizer.json":
            os.remove(os.path.join(temp_dir, path))

def jit_trace_model(self, hf_pipeline, model_id: str, temp_dir: str,
                    include_types: bool):
    logging.info(
        f"Tracing model: {model_id} include_token_types={include_types} ..."
    )
    encoding = self.encode_inputs(hf_pipeline.tokenizer)
    input_ids = encoding["input_ids"]
    attention_mask = encoding["attention_mask"]
    token_type_ids = encoding.get("token_type_ids")
    if include_types and token_type_ids is None:
        return None

    # noinspection PyBroadException
    try:
        if include_types:
            script_module = torch.jit.trace(
                hf_pipeline.model,
                (input_ids, attention_mask, token_type_ids),
```

```
                strict=False)
        else:
            script_module = torch.jit.trace(hf_pipeline.model,
                                            (input_ids, attention_mask),
                                            strict=False)

        model_name = model_id.split("/")[-1]
        logging.info(f"Saving torchscript model: {model_name}.pt ...")
        model_file = os.path.join(temp_dir, f"{model_name}.pt")
        script_module.save(model_file)
    except RuntimeError as e:
        logging.warning(f"Failed to trace model: {model_id}.")
        logging.warning(e, exc_info=True)
        return None

    return model_file

def save_to_model_zoo(self, model_info, output_dir: str, temp_dir: str,
                      hf_pipeline, include_types: bool):
    model_id = model_info.modelId
    model_name = model_id.split("/")[-1]

    repo_dir = f"{output_dir}/model/{self.application}/ai/djl/huggingface/pytorch
    model_dir = f"{repo_dir}/0.0.1"
    if not os.path.exists(model_dir):
        os.makedirs(model_dir)

    # Save serving.properties
    serving_file = os.path.join(temp_dir, "serving.properties")
    arguments = self.get_extra_arguments(hf_pipeline)
    with open(serving_file, 'w') as f:
        f.write(f"engine=PyTorch\n"
                f"option.modelName={model_name}\n"
                f"option.mapLocation=true\n"
                f"translatorFactory={self.translator}\n")
        if include_types:
            f.write(f"includeTokenTypes={include_types}\n")

        for k, v in arguments.items():
            f.write(f"{k}={v}\n")

    # Save model as .zip file
    logging.info(f"Saving DJL model as zip: {model_name}.zip ...")
    zip_file = os.path.join(model_dir, f"{model_name}.zip")
    zip_dir(temp_dir, zip_file)

    # Save metadata.json
    sha1 = sha1_sum(zip_file)
    file_size = os.path.getsize(zip_file)
    metadata = HuggingfaceMetadata(model_info, self.application,
                                   self.translator, sha1, file_size)
```

```
    metadata_file = os.path.join(repo_dir, "metadata.json")
    metadata.save_metadata(metadata_file)

    return file_size

def verify_jit_model(self, hf_pipeline, model_file: str,
                     include_types: bool, cpu_only: bool):
    logging.info(
        f"Verifying torchscript model(include_token_types={include_types}): {mode
    )

    tokenizer = hf_pipeline.tokenizer
    encoding = self.encode_inputs(tokenizer)

    input_ids = encoding["input_ids"]
    attention_mask = encoding["attention_mask"]
    token_type_ids = encoding.get("token_type_ids")
    if torch.cuda.is_available() and not cpu_only:
        traced_model = torch.jit.load(model_file, map_location='cuda:0')
        traced_model.to(self.device)
        input_ids = input_ids.to(self.device)
        attention_mask = attention_mask.to(self.device)
        if token_type_ids is not None:
            token_type_ids = token_type_ids.to(self.device)
    else:
        traced_model = torch.jit.load(model_file)

    traced_model.eval()

    try:
        # test traced model
        if include_types:
            out = traced_model(input_ids, attention_mask, token_type_ids)
        else:
            out = traced_model(input_ids, attention_mask)
    except RuntimeError as e:
        logging.warning(e, exc_info=True)
        return False, "Failed to run inference on jit model"

    return self.verify_jit_output(hf_pipeline, encoding, out)

def get_extra_arguments(self, hf_pipeline) -> dict:
    return {}

def verify_jit_output(self, hf_pipeline, encoding, out):
    if not hasattr(out, "last_hidden_layer"):
        return False, f"Unexpected inference result: {out}"

    return True, None

def load_model(self, model_id: str):
```

```
        logging.info(f"Loading model: {model_id} ...")
        kwargs = {
            "tokenizer": model_id,
            "device": -1  # always use CPU to trace the model
        }
        return pipeline(task=self.task,
                        model=model_id,
                        framework="pt",
                        **kwargs)

    def encode_inputs(self, tokenizer):
        return tokenizer.encode_plus(self.inputs, return_tensors='pt')
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-35-0533e5186ce0> in <module>
     21 from transformers import pipeline
     22
---> 23 from metadata import HuggingfaceMetadata
     24 from shasum import sha1_sum
     25 from zip_utils import zip_dir

ModuleNotFoundError: No module named 'metadata'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

OPEN EXAMPLES SEARCH STACK OVERFLOW

```
!pip install metadata
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Collecting metadata
  Downloading metadata-0.2.tar.gz (1.5 kB)
  Downloading metadata-0.1.1.tar.gz (1.5 kB)
  Downloading metadata-0.1.tar.gz (1.1 kB)
ERROR: Cannot install metadata==0.1, metadata==0.1.1 and metadata==0.2 because t

The conflict is caused by:
    metadata 0.2 depends on hachoir-core==1.3.3
    metadata 0.1.1 depends on hachoir-core==1.3.3
    metadata 0.1 depends on hachoir-core==1.3.3

To fix this you could try to:
1. loosen the range of package versions you've specified
2. remove package versions to allow pip attempt to solve the dependency conflict

ERROR: ResolutionImpossible: for help visit https://pip.pypa.io/en/latest/user_g
```

```
from tokenizers import Tokenizer
from tokenizers.models import WordPiece
bert_tokenizer = Tokenizer(WordPiece(unk_token="[UNK]"))


from tokenizers import normalizers
from tokenizers.normalizers import NFD, Lowercase, StripAccents
bert_tokenizer.normalizer = normalizers.Sequence([NFD(), Lowercase(), StripAccents()]


from tokenizers.pre_tokenizers import Whitespace
bert_tokenizer.pre_tokenizer = Whitespace()


from tokenizers.processors import TemplateProcessing
bert_tokenizer.post_processor = TemplateProcessing(
    single="[CLS] $A [SEP]",
    pair="[CLS] $A [SEP] $B:1 [SEP]:1",
    special_tokens=[
        ("[CLS]", 1),
        ("[SEP]", 2),
    ],
)


from tokenizers.trainers import WordPieceTrainer
trainer = WordPieceTrainer(vocab_size=30522, special_tokens=["[UNK]", "[CLS]", "[SEP]
files = [f"/content/token/vocab.txt"]
bert_tokenizer.train(files, trainer)
bert_tokenizer.save("/content/bert-wiki.json")


bert_tokenizer(table=table, queries=queries, padding="max_length", return_tensors="tf
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-cd533446c372> in <module>
----> 1 bert_tokenizer(table=table, queries=queries, padding="max_length", retur

TypeError: 'tokenizers.Tokenizer' object is not callable
```

SEARCH STACK OVERFLOW

Colab paid products - Cancel contracts here

✓ 0s completed at 12:54 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.