```
!pip install datasets evaluate transformers[sentencepiece]
!apt install git-lfs
```

```
Collecting datasets
  Downloading datasets-2.15.0-py3-none-any.whl (521 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 521.2/521.2 kB 7.1 MB/s eta 0:00:00
Collecting evaluate
  Downloading evaluate-0.4.1-py3-none-any.whl (84 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.1/84.1 kB 8.8 MB/s eta 0:00:00
Requirement already satisfied: transformers[sentencepiece] in /usr/local/lib/python3.10/dist-packages (4.35.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.23.5)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (9.0.0)
Collecting pyarrow-hotfix (from datasets)
  Downloading pyarrow_hotfix-0.6-py3-none-any.whl (7.9 kB)
Collecting dill<0.3.8,>=0.3.0 (from datasets)
  Downloading dill-0.3.7-py3-none-any.whl (115 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 15.7 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from datasets) (3.4.1)
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.15-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 16.6 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<=2023.10.0,>=2023.1.0 in /usr/local/lib/python3.10/dist-packages (from datase
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.8.6)
Requirement already satisfied: huggingface-hub>=0.18.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.19.4
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.1)
Collecting responses<0.19 (from evaluate)
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers[sentencepiece]) (3
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers[sentencep
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers[sente
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers[sentence
Collecting sentencepiece!=0.1.92,>=0.1.91 (from transformers[sentencepiece])
  Downloading sentencepiece-0.1.99-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 26.3 MB/s eta 0:00:00
Requirement already satisfied: protobuf in /usr/local/lib/python3.10/dist-packages (from transformers[sentencepiece]) (3
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (23.1.0
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->da
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->dat
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.9.2
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-h
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->dat
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->dat
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2023.3.p
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->
Installing collected packages: sentencepiece, pyarrow-hotfix, dill, responses, multiprocess, datasets, evaluate
Successfully installed datasets-2.15.0 dill-0.3.7 evaluate-0.4.1 multiprocess-0.70.15 pyarrow-hotfix-0.6 responses-0.18.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git-lfs is already the newest version (3.0.2-1ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
```

```python
import transformers

def get_model_checkpoint():
    return "clincolnoz/LessSexistBERT"

def load_model(checkpoint):
    return transformers.TFAutoModelForMaskedLM.from_pretrained(checkpoint, from_pt=True)

model_checkpoint = get_model_checkpoint()
model = load_model(model_checkpoint)
model.summary()

masked_text = "This is a great [MASK]."

def load_tokenizer(checkpoint):
    return transformers.AutoTokenizer.from_pretrained(checkpoint)

tokenizer = load_tokenizer(model_checkpoint)
tokenizer.mask_token_id
```

```
config.json: 100%                                        697/697 [00:00<00:00, 13.0kB/s]

pytorch_model.bin: 100%                                  441M/441M [00:11<00:00, 49.3MB/s]
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFBertForMaskedLM: ['bert.embeddings.posi
- This IS expected if you are initializing TFBertForMaskedLM from a PyTorch model trained on another task or with another ar
- This IS NOT expected if you are initializing TFBertForMaskedLM from a PyTorch model that you expect to be exactly identica
All the weights of TFBertForMaskedLM 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 TFBertForMaskedLM for pr
Model: "tf_bert_for_masked_lm"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  108986880

 mlm___cls (TFBertMLMHead)   multiple                  24555190

=================================================================
Total params: 109609654 (418.13 MB)
Trainable params: 109609654 (418.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
tokenizer_config.json: 100%                              309/309 [00:00<00:00, 14.4kB/s]

vocab.txt: 100%                                          232k/232k [00:00<00:00, 2.62MB/s]

tokenizer.json: 100%                                     1.05M/1.05M [00:00<00:00, 15.9MB/s]

added_tokens.json: 100%                                  2.35k/2.35k [00:00<00:00, 113kB/s]

special_tokens_map.json: 100%                            125/125 [00:00<00:00, 4.11kB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
103
```

```
import numpy as np
import tensorflow as tf

def prepare_inputs(text, tokenizer):
    return tokenizer(text, return_tensors="np")

def find_mask_token_index(inputs, tokenizer):
    return np.argwhere(inputs["input_ids"] == tokenizer.mask_token_id)[0, 1]

def get_top_tokens(logits, mask_index, num_tokens=5):
    mask_token_logits = logits[0, mask_index, :]
    return np.argsort(-mask_token_logits)[:num_tokens].tolist()
text = "This is a great [MASK]."

inputs = prepare_inputs(text, tokenizer)
token_logits = model(**inputs).logits
mask_token_index = find_mask_token_index(inputs, tokenizer)
top_5_tokens = get_top_tokens(token_logits, mask_token_index)

def display_predictions(tokens, text, tokenizer):
    for token in tokens:
        print(f">>> {text.replace(tokenizer.mask_token, tokenizer.decode([token]))}")

display_predictions(top_5_tokens, text, tokenizer)
```

```
>>> This is a great post.
>>> This is a great question.
>>> This is a great point.
>>> This is a great idea.
>>> This is a great comment.
```

```
from datasets import load_dataset

def load_and_prepare_dataset(dataset_name, num_samples=3, seed=42):
    dataset = load_dataset(dataset_name)
    sample = dataset["train"].shuffle(seed=seed).select(range(num_samples))
    return dataset, sample

def display_samples(sample):
    for row in sample:
        print(f"\n'>>> Review: {row['hypothesis']}'")

def tokenize_dataset(dataset, tokenizer):
    def tokenize_function(examples):
        result = tokenizer(examples["hypothesis"])
        if tokenizer.is_fast:
            result["word_ids"] = [result.word_ids(i) for i in range(len(result["input_ids"]))]
        return result
    return dataset.map(tokenize_function, batched=True, remove_columns=["premise", "hypothesis", "label"])

snli_dataset, sample = load_and_prepare_dataset("snli")
display_samples(sample)
tokenized_datasets = tokenize_dataset(snli_dataset, tokenizer)
print(tokenized_datasets)
```

```
    '>>> Review: the historian is digging with his friend for study.'
```

```
chunk_size = 128
```

```
    '>>> Review: A man is outside on the patio.'
```

```
def print_review_lengths(tokenized_samples):
    for idx, sample in enumerate(tokenized_samples["input_ids"]):
        print(f"'>>> Review {idx} length: {len(sample)}'")

def concatenate_and_print_length(tokenized_samples):
    concatenated = {k: sum(tokenized_samples[k], []) for k in tokenized_samples.keys()}
    total_length = len(concatenated["input_ids"])
    print(f"'>>> Concatenated reviews length: {total_length}'")
    return concatenated, total_length

def create_and_print_chunks(concatenated, total_length, chunk_size):
    chunks = {
        k: [t[i : i + chunk_size] for i in range(0, total_length, chunk_size)]
        for k, t in concatenated.items()
    }
    for chunk in chunks["input_ids"]:
        print(f"'>>> Chunk length: {len(chunk)}'")

tokenized_samples = tokenized_datasets["train"][:3]
print_review_lengths(tokenized_samples)

concatenated, total_length = concatenate_and_print_length(tokenized_samples)
create_and_print_chunks(concatenated, total_length, chunk_size)
```

```
    '>>> Review 0 length: 12'
    '>>> Review 1 length: 17'
    '>>> Review 2 length: 13'
    '>>> Concatenated reviews length: 42'
    '>>> Chunk length: 42'
```

```
# Revised Code
chunked_data = {key: [value[idx:idx + chunk_size] for idx in range(0, total_length, chunk_size)] for key, value in concatenated.

for each_chunk in chunked_data["input_ids"]:
    print(f"'>>> Chunk length: {len(each_chunk)}'")
```

```
    '>>> Chunk length: 42'
```

```
def split_into_chunks(data):
    # Combine all elements
    combined_data = {key: sum(data[key], []) for key in data.keys()}
    # Calculate total combined length
    combined_length = len(combined_data[next(iter(data))])
    # Adjust length to be a multiple of chunk_size
    adjusted_length = (combined_length // chunk_size) * chunk_size
    # Divide into chunks
    chunked_result = {
        key: [chunk[i:i + chunk_size] for i in range(0, adjusted_length, chunk_size)]
        for key, chunk in combined_data.items()
    }
    # Replicate input_ids to labels
    chunked_result["labels"] = chunked_result["input_ids"].copy()
    return chunked_result

processed_datasets = tokenized_datasets.map(split_into_chunks, batched=True)
processed_datasets
```

```
Map: 100%                                    10000/10000 [00:01<00:00, 7373.05 examples/s]

Map: 100%                                    550152/550152 [01:16<00:00, 8686.95 examples/s]

Map: 100%                                    10000/10000 [00:01<00:00, 8856.09 examples/s]

DatasetDict({
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
```

```
decoded_text = tokenizer.decode(processed_datasets["train"][1]["input_ids"])
print(decoded_text)

from transformers import DataCollatorForLanguageModeling

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm_probability=0.15)

selected_samples = [processed_datasets["train"][index] for index in range(2)]
for sample in selected_samples:
    sample.pop("word_ids", None)

for batch in data_collator(selected_samples)["input_ids"]:
    print(f"\n'>>> {tokenizer.decode(batch)}'")
```

```
[SEP] [CLS] an elderly man sit s in a small shop. [SEP] [CLS] some women are hugging on vacation. [SEP] [CLS] the women are

'>>> [CLS] a person is training his horse for [MASK] competition. [SEP] [CLS] a person is at [MASK] diner, order ing an ome

'>>> [SEP] [CLS] an elderly man sit s in a small shop. [SEP] [CLS] some women are hugging on vacation. [SEP] [CLS] the women
```

```
import collections
import numpy as np
from transformers.data.data_collator import tf_default_data_collator

wwm_probability = 0.2

def apply_whole_word_masking(samples):
    for sample in samples:
        word_ids = sample.pop("word_ids")

        # Mapping tokens to their respective word indices
        token_to_word = collections.defaultdict(list)
        word_index = -1
        for idx, word_id in enumerate(word_ids):
            if word_id is not None:
                if word_id != word_index:
                    word_index = word_id
                token_to_word[word_index].append(idx)

        # Masking words based on probability
        random_mask = np.random.binomial(1, wwm_probability, len(token_to_word))
        input_ids = sample["input_ids"]
        labels = sample["labels"]
        updated_labels = [-100] * len(labels)
        for word_idx in np.nonzero(random_mask)[0]:
            for token_idx in token_to_word[word_idx.item()]:
                updated_labels[token_idx] = labels[token_idx]
                input_ids[token_idx] = tokenizer.mask_token_id
        sample["labels"] = updated_labels

    return tf_default_data_collator(samples)

sampled_data = [processed_datasets["train"][i] for i in range(2)]
processed_batch = apply_whole_word_masking(sampled_data)
```

```
train_size = 40_000
test_size = int(0.1 * train_size)

downsampled_dataset = processed_datasets["train"].train_test_split(
    train_size=train_size, test_size=test_size, seed=42
)
downsampled_dataset
```

```
    DatasetDict({
        train: Dataset({
            features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
            num_rows: 40000
        })
        test: Dataset({
            features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
            num_rows: 4000
        })
    })
```

```
tf_train_dataset = model.prepare_tf_dataset(
    downsampled_dataset["train"],
    collate_fn=data_collator,
    shuffle=True,
    batch_size=32
)

tf_test_dataset = model.prepare_tf_dataset(
    downsampled_dataset["test"],
    collate_fn=data_collator,
    shuffle=False,
    batch_size=32
)
```

```
from transformers import create_optimizer
from transformers.keras_callbacks import PushToHubCallback
import tensorflow as tf

# Calculate the number of training steps
num_training_steps = len(tf_train_dataset)
# Setting up the optimizer with warmup and weight decay
optimizer_config, lr_schedule = create_optimizer(
    init_lr=2e-5,
    num_warmup_steps=1_000,
    num_train_steps=num_training_steps,
    weight_decay_rate=0.01
)
# Compiling the model with the configured optimizer
model.compile(optimizer=optimizer_config)

# Enabling mixed-precision training with float16
tf.keras.mixed_precision.set_global_policy('mixed_float16')
```

```
import math

# Evaluating the model on the evaluation dataset and calculating perplexity
initial_eval_loss = model.evaluate(tf_test_dataset)
print(f"Initial Perplexity: {math.exp(initial_eval_loss):.2f}")

# Training the model
model.fit(tf_train_dataset, validation_data=tf_test_dataset)

# Re-evaluating the model to see improvements
final_eval_loss = model.evaluate(tf_test_dataset)
print(f"Final Perplexity: {math.exp(final_eval_loss):.2f}")
```

```
    125/125 [==============================] - 57s 346ms/step - loss: 6.0955
    Initial Perplexity: 443.84
    1250/1250 [==============================] - 1286s 1s/step - loss: 3.4706 - val_loss: 2.0009
    125/125 [==============================] - 44s 350ms/step - loss: 2.0239
    Final Perplexity: 7.57
```

```
from transformers import pipeline

mask_filler = pipeline(
    "fill-mask", model=model, tokenizer=tokenizer
)

preds = mask_filler(text)
```

```
from transformers import AutoTokenizer, AutoModel , AutoModelForMaskedLM
from transformers import pipeline

! pip install honest
! pip install transformers
! pip install sentencepiece

from honest import honest
```

```
    Collecting honest
      Downloading honest-0.2.1-py2.py3-none-any.whl (6.2 kB)
    Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from honest) (1.5.3)
    Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from honest) (1.23.5)
    Collecting Unidecode==1.3.2 (from honest)
      Downloading Unidecode-1.3.2-py3-none-any.whl (235 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 235.7/235.7 kB 4.4 MB/s eta 0:00:00
    Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->honest) (2.8.
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->honest) (2023.3.post1)
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->hon
    Installing collected packages: Unidecode, honest
    Successfully installed Unidecode-1.3.2 honest-0.2.1
    Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.35.2)
    Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.1)
    Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from transformers) (
    Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.23.5)
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.2)
    Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
    Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3)
    Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
    Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.15.0
    Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.0)
    Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.1)
    Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.
    Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1
    Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transform
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
    Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (
    Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/dist-packages (0.1.99)
```

```
name_model = model
lang = "en"
k = 1

evaluator = honest.HonestEvaluator(lang)
masked_templates = evaluator.templates(data_set="binary")
```

```
    /usr/local/lib/python3.10/dist-packages/honest/honest.py:132: UserWarning: DataFrame columns are not unique, some columns wi
      index_col=0, sep='\t').T.to_dict('dict')
```

```
def get_completion(sentence, template):
  start = sentence.find(template[:-4])+len(template[:-4])
  if tokenizer.eos_token != None and tokenizer.eos_token in sentence:
    end = sentence.find(tokenizer.eos_token)
  elif tokenizer.sep_token != None and tokenizer.sep_token in sentence:
    end = sentence.find(tokenizer.sep_token)
  else:
    end = len(sentence)-1

  return sentence[start:end]

tokenizer = tokenizer
```