```
!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 5.0 MB/s eta 0:00:00
Collecting evaluate
  Downloading evaluate-0.4.1-py3-none-any.whl (84 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.1/84.1 kB 9.2 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 9.6 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 9.6 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 datas
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.
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[sent
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 13.8 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->data
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) (1.4
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-hu
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->data
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->data
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-0.18.0
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.
```

```
# from huggingface_hub import notebook_login

# notebook_login()
```

```
Token is valid (permission: write).

Your token has been saved in your configured git credential helpers (store).

Your token has been saved to /root/.cache/huggingface/token

Login successful
```

```
from transformers import TFAutoModelForMaskedLM

model_checkpoint = "clincolnoz/LessSexistBERT"
model = TFAutoModelForMaskedLM.from_pretrained(model_checkpoint, from_pt = True)
```

```
config.json: 100%                                                 697/697 [00:00<00:00, 23.0kB/s]
pytorch_model.bin: 100%                                           441M/441M [00:08<00:00, 62.1MB/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.summary()
```

```
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)
_________________________________________________________________
```

```
text = "This is a great [MASK]."


from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

```
tokenizer_config.json: 100%                                       309/309 [00:00<00:00, 15.8kB/s]
vocab.txt: 100%                                                   232k/232k [00:00<00:00, 5.34MB/s]
tokenizer.json: 100%                                              1.05M/1.05M [00:00<00:00, 11.3MB/s]
added_tokens.json: 100%                                           2.35k/2.35k [00:00<00:00, 175kB/s]
special_tokens_map.json: 100%                                     125/125 [00:00<00:00, 9.22kB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
```

```
tokenizer.mask_token_id
```

```
103
```

```
import numpy as np
import tensorflow as tf

inputs = tokenizer(text, return_tensors="np")
token_logits = model(**inputs).logits
# Find the location of [MASK] and extract its logits
mask_token_index = np.argwhere(inputs["input_ids"] == tokenizer.mask_token_id)[0, 1]
mask_token_logits = token_logits[0, mask_token_index, :]
# Pick the [MASK] candidates with the highest logits
# We negate the array before argsort to get the largest, not the smallest, logits
top_5_tokens = np.argsort(-mask_token_logits)[:5].tolist()

for token in top_5_tokens:
    print(f">>> {text.replace(tokenizer.mask_token, tokenizer.decode([token]))}")
```

```
>>> 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

imdb_dataset = load_dataset("imdb")
imdb_dataset
```

```
Downloading builder script: 100%  4.31k/4.31k [00:00<00:00, 135kB/s]
Downloading metadata: 100%  2.17k/2.17k [00:00<00:00, 154kB/s]
Downloading readme: 100%  7.59k/7.59k [00:00<00:00, 328kB/s]
Downloading data: 100%  84.1M/84.1M [00:12<00:00, 6.45MB/s]
Generating train split: 100%  25000/25000 [00:08<00:00, 5251.42 examples/s]
Generating test split: 100%  25000/25000 [00:07<00:00, 9104.09 examples/s]
Generating unsupervised split: 100%  50000/50000 [00:10<00:00, 9041.30 examples/s]
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 25000
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 25000
    })
    unsupervised: Dataset({
```

```
sample = imdb_dataset["train"].shuffle(seed=42).select(range(3))

for row in sample:
    print(f"\n'>>> Review: {row['text']}'")
    print(f"'>>> Label: {row['label']}'")
```

```
'>>> Review: There is no relation at all between Fortier and Profiler but the fact that both are police series about violent
'>>> Label: 1'

'>>> Review: This movie is a great. The plot is very true to the book which is a classic written by Mark Twain. The movie st
'>>> Label: 1'

'>>> Review: George P. Cosmatos' "Rambo: First Blood Part II" is pure wish-fulfillment. The United States clearly didn't win
'>>> Label: 0'
```

```
def tokenize_function(examples):
    result = tokenizer(examples["text"])
    if tokenizer.is_fast:
        result["word_ids"] = [result.word_ids(i) for i in range(len(result["input_ids"]))]
    return result


# Use batched=True to activate fast multithreading!
tokenized_datasets = imdb_dataset.map(
    tokenize_function, batched=True, remove_columns=["text", "label"]
)
tokenized_datasets
```

```
Map: 100%  25000/25000 [00:30<00:00, 703.66 examples/s]
Token indices sequence length is longer than the specified maximum sequence length for this model (535 > 512). Running this
Map: 100%  25000/25000 [00:25<00:00, 959.10 examples/s]
Map: 100%  50000/50000 [00:55<00:00, 1027.51 examples/s]
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids'],
        num_rows: 25000
    })
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids'],
        num_rows: 25000
    })
    unsupervised: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids'],
        num_rows: 50000
    })
})
```

```
chunk_size = 128
```

```
# Slicing produces a list of lists for each feature
tokenized_samples = tokenized_datasets["train"][:3]

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

```
'>>> Review 0 length: 409'
'>>> Review 1 length: 345'
'>>> Review 2 length: 147'
```

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

```
'>>> Concatenated reviews length: 901'
```

```
chunks = {
    k: [t[i : i + chunk_size] for i in range(0, total_length, chunk_size)]
    for k, t in concatenated_examples.items()
}

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

```
'>>> Chunk length: 128'
'>>> Chunk length: 128'
'>>> Chunk length: 128'
'>>> Chunk length: 128'
'>>> Chunk length: 128'
'>>> Chunk length: 128'
'>>> Chunk length: 128'
'>>> Chunk length: 5'
```

```
def group_texts(examples):
    # Concatenate all texts
    concatenated_examples = {k: sum(examples[k], []) for k in examples.keys()}
    # Compute length of concatenated texts
    total_length = len(concatenated_examples[list(examples.keys())[0]])
    # We drop the last chunk if it's smaller than chunk_size
    total_length = (total_length // chunk_size) * chunk_size
    # Split by chunks of max_len
    result = {
        k: [t[i : i + chunk_size] for i in range(0, total_length, chunk_size)]
        for k, t in concatenated_examples.items()
    }
    # Create a new labels column
    result["labels"] = result["input_ids"].copy()
    return result


lm_datasets = tokenized_datasets.map(group_texts, batched=True)
lm_datasets
```

```
Map: 100%                                                  25000/25000 [01:46<00:00, 268.18 examples/s]

Map: 100%                                                  25000/25000 [01:25<00:00, 296.90 examples/s]

Map: 100%                                                  50000/50000 [02:53<00:00, 284.93 examples/s]
```

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
        num_rows: 67795
    })
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
        num_rows: 66211
    })
    unsupervised: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
        num_rows: 136047
    })
})
```

```
tokenizer.decode(lm_datasets["train"][1]["input_ids"])
```

```
'##y on what the average swede thought about certain political issues such as the vietnam war and race issues in the united
states. in bet ween asking politicians and ordinary denizens of stockholm about their opinions on politics, she has sex wit
h her drama teacher, class mate s, and married men. < br / > < br / > what kill s me about i am curious - yel low is that 4
0 years ago, this was considered porn ographic. really, the sex and nudity scene s are few and far bet ween, even then it's
not shot like some cheap ly made porn o. while my'
```

```
from transformers import DataCollatorForLanguageModeling

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


samples = [lm_datasets["train"][i] for i in range(2)]
for sample in samples:
    _ = sample.pop("word_ids")

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

```
You're using a BertTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster th

'>>> [CLS] i rented i am curious [MASK] yel low from my video store be cause of all the controversy that surrounded it when

'>>> ##y [MASK] what the average swede thought [MASK] certain political issues such as [MASK] vietnam war and race [MASK] in
```

```
import collections
import numpy as np

from transformers.data.data_collator import tf_default_data_collator

wwm_probability = 0.2


def whole_word_masking_data_collator(features):
    for feature in features:
        word_ids = feature.pop("word_ids")

        # Create a map between words and corresponding token indices
        mapping = collections.defaultdict(list)
        current_word_index = -1
        current_word = None
        for idx, word_id in enumerate(word_ids):
            if word_id is not None:
                if word_id != current_word:
                    current_word = word_id
                    current_word_index += 1
                mapping[current_word_index].append(idx)

        # Randomly mask words
        mask = np.random.binomial(1, wwm_probability, (len(mapping),))
        input_ids = feature["input_ids"]
        labels = feature["labels"]
        new_labels = [-100] * len(labels)
        for word_id in np.where(mask)[0]:
            word_id = word_id.item()
            for idx in mapping[word_id]:
                new_labels[idx] = labels[idx]
                input_ids[idx] = tokenizer.mask_token_id
        feature["labels"] = new_labels

    return tf_default_data_collator(features)


samples = [lm_datasets["train"][i] for i in range(2)]
batch = whole_word_masking_data_collator(samples)

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

```
'>>> [CLS] [MASK] rented [MASK] am curious - yel low from my video store be [MASK] [MASK] all the controversy that surrounde

'>>> [MASK] on what the average swede thought about certain political issues such as the vietnam war and race [MASK] in the
```

```
train_size = 10_000
test_size = int(0.1 * train_size)

downsampled_dataset = lm_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: 10000
    })
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'word_ids', 'labels'],
        num_rows: 1000
    })
})
```

```
# from huggingface_hub import notebook_login

# notebook_login()
```

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

tf_eval_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

num_train_steps = len(tf_train_dataset)
optimizer, schedule = create_optimizer(
    init_lr=2e-5,
    num_warmup_steps=1_000,
    num_train_steps=num_train_steps,
    weight_decay_rate=0.01,
)
model.compile(optimizer=optimizer)

# Train in mixed-precision float16
tf.keras.mixed_precision.set_global_policy("mixed_float16")

# model_name = model_checkpoint.split("/")[-1]
# callback = PushToHubCallback(
#     output_dir=f"{model_name}-finetuned-imdb", tokenizer=tokenizer
# )
```

```
import math

eval_loss = model.evaluate(tf_eval_dataset)
print(f"Perplexity: {math.exp(eval_loss):.2f}")
```

```
32/32 [==============================] - 18s 363ms/step - loss: 10.0729
Perplexity: 23693.32
```

```
model.fit(tf_train_dataset, validation_data=tf_eval_dataset)
```

```
312/312 [==============================] - 354s 1s/step - loss: 8.1932 - val_loss: 6.6347
<keras.src.callbacks.History at 0x7f811cffcc40>
```

```
eval_loss = model.evaluate(tf_eval_dataset)
print(f"Perplexity: {math.exp(eval_loss):.2f}")
```

```
32/32 [==============================] - 11s 349ms/step - loss: 6.6304
Perplexity: 757.77
```

```
from transformers import pipeline

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


preds = mask_filler(text)

for pred in preds:
    print(f">>> {pred['sequence']}")
```

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

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


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

```
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 4.3 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)
```

```
from honest import honest


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
# model = AutoModelForMaskedLM.from_pretrained(name_model)

# Define nlp_fill pipeline
nlp_fill = pipeline('fill-mask', model=model, tokenizer=tokenizer, top_k=k)

print("FILL EXAMPLE:",nlp_fill('all women likes to [M].'.replace('[M]',tokenizer.mask_token)))


# Fill templates (please check if the filled words contain any special character)
filled_templates = [[get_completion(fill['sequence'],masked_sentence) for fill in nlp_fill(masked_sentence.replace('[M]',tokeniz

honest_score, honest_df = evaluator.honest_dataframe(filled_templates, masked_templates)
print(name_model, k, honest_score)
```

```
FILL EXAMPLE: [{'score': 0.1534290760755539, 'token': 3604, 'token_str': 'travel', 'sequence': 'all women likes to travel.'}
<transformers.models.bert.modeling_tf_bert.TFBertForMaskedLM object at 0x7f818c2e0370> 1 0.009668508287292817
```

```
model
```

```
<transformers.models.bert.modeling_tf_bert.TFBertForMaskedLM at 0x7f818c2e0370>
```