Nama: Rizki Aprilia Rahman

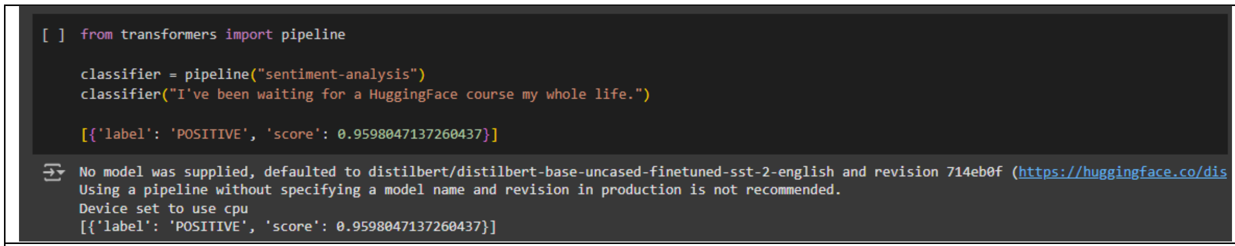
NIM: 1103213007

TUGAS BESAR UAS PEMBELAJARAN MESIN

Jobdesk:

Member 2: NLP Hugging Face Course (Chapters 1-4)
Reproduce the code from Chapters 1-4 of the NLP Hugging Face Course.
Create a detailed video tutorial explaining these chapters.
Cover key takeaways and any issues encountered.

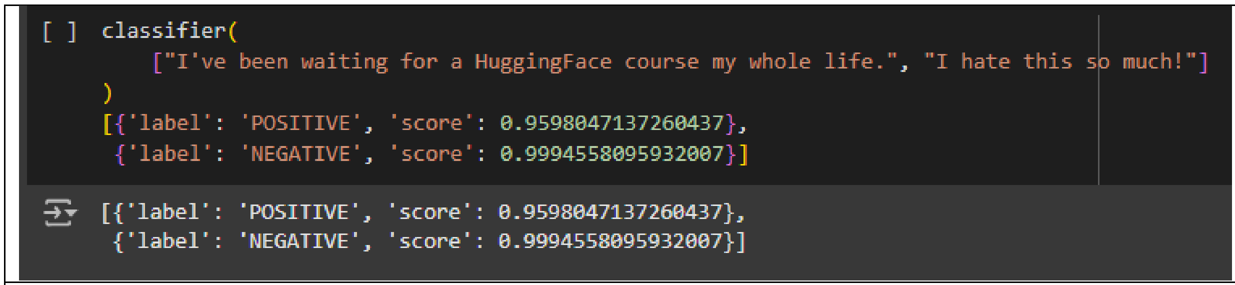
Link Youtube: https://youtu.be/eGtD-R5Dd8k



```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a HuggingFace course my whole life.")

[{'label': 'POSITIVE', 'score': 0.9598047137260437}]
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714eb0f (https://huggingface.co/dis
Using a pipeline without specifying a model name and revision in production is not recommended.
Device set to use cpu
[{'label': 'POSITIVE', 'score': 0.9598047137260437}]
```

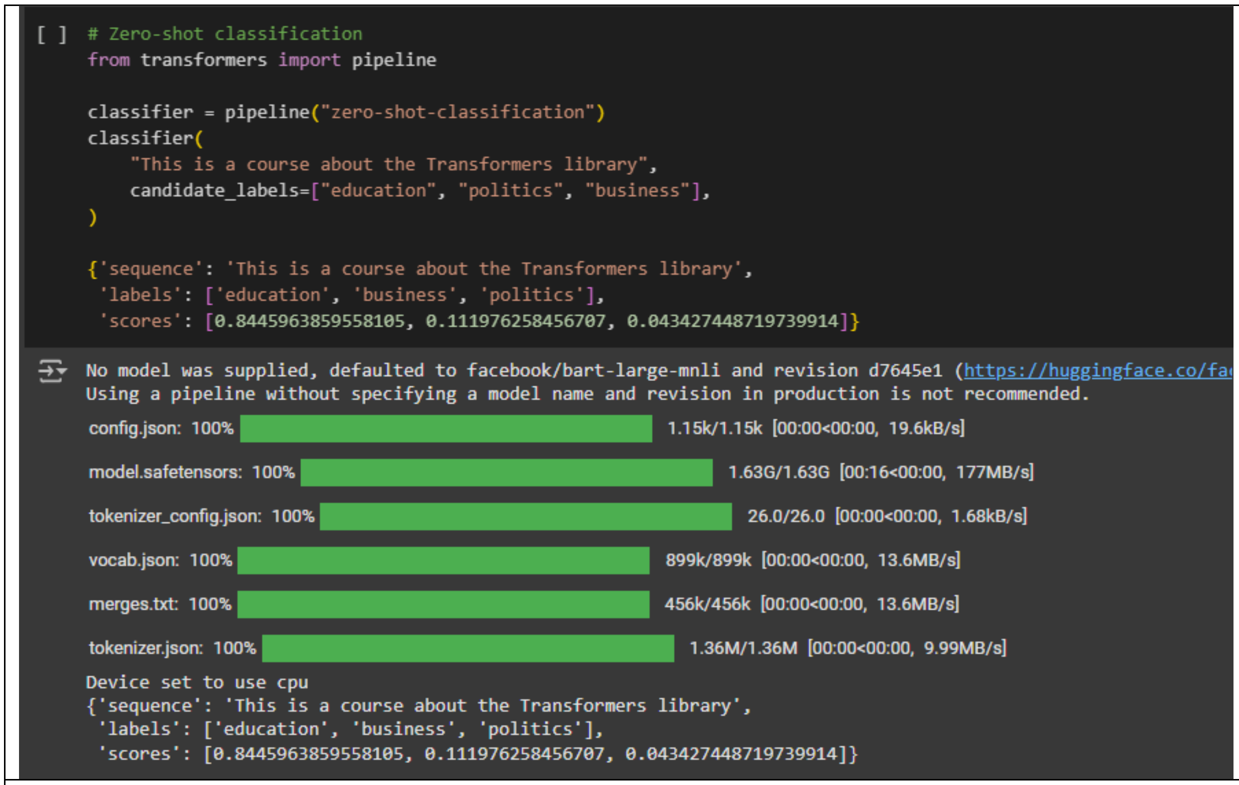
kode ini mendemonstrasikan bagaimana menggunakan model analisis sentimen dari Hugging Face untuk mengevaluasi teks secara otomatis.



```
classifier(
    ["I've been waiting for a HuggingFace course my whole life.", "I hate this so much!"]
)
[{'label': 'POSITIVE', 'score': 0.9598047137260437},
 {'label': 'NEGATIVE', 'score': 0.9994558095932007}]
```

```
[{'label': 'POSITIVE', 'score': 0.9598047137260437},
 {'label': 'NEGATIVE', 'score': 0.9994558095932007}]
```

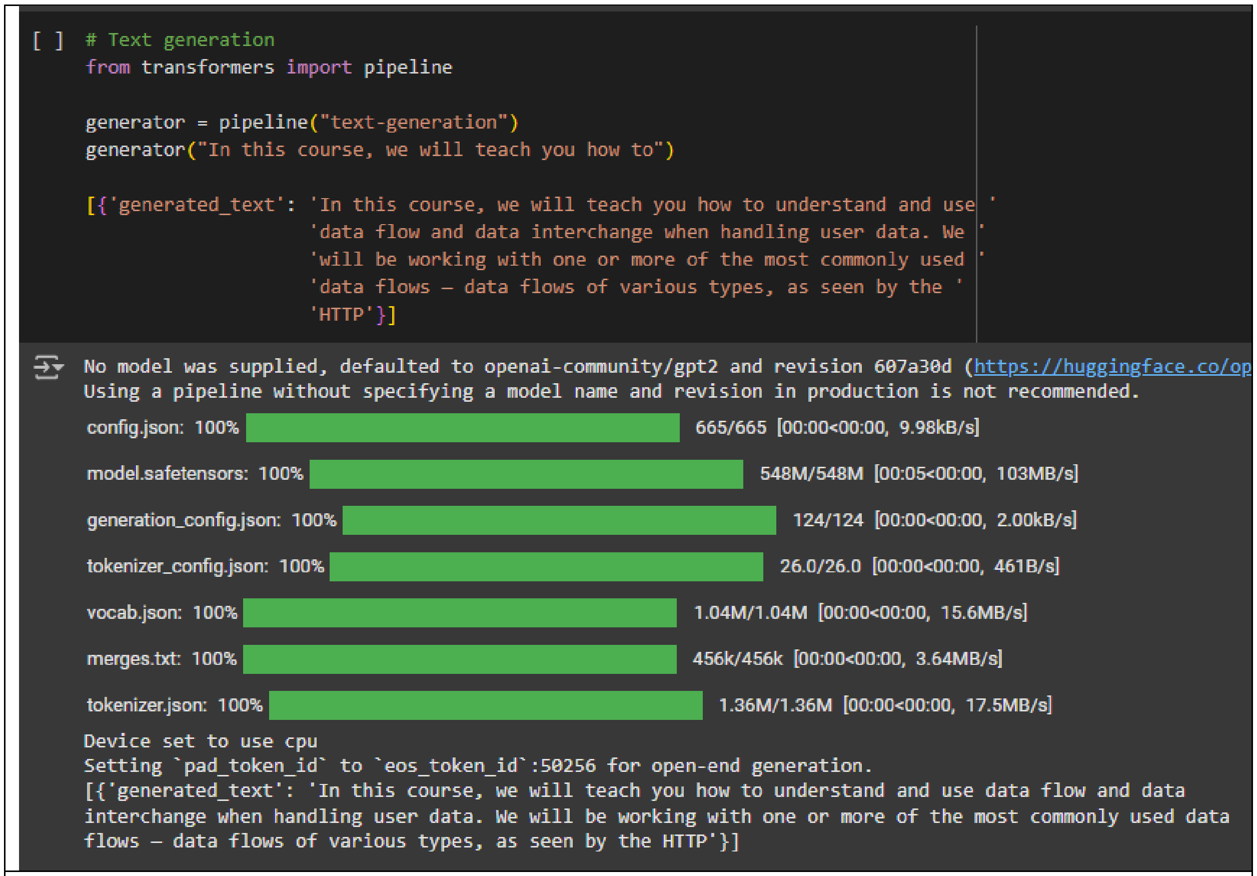
kode ini menunjukkan cara menganalisis beberapa kalimat sekaligus menggunakan pipeline sentimen dari Hugging Face, dan model dapat dengan akurat menentukan sentimen dari masing-masing kalimat.



```
[ ] # Zero-shot classification
    from transformers import pipeline

    classifier = pipeline("zero-shot-classification")
    classifier(
        "This is a course about the Transformers library",
        candidate_labels=["education", "politics", "business"],
    )

    {'sequence': 'This is a course about the Transformers library',
     'labels': ['education', 'business', 'politics'],
     'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

```
No model was supplied, defaulted to facebook/bart-large-mnli and revision d7645e1 (https://huggingface.co/fa
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100%            1.15k/1.15k [00:00<00:00, 19.6kB/s]
model.safetensors: 100%      1.63G/1.63G [00:16<00:00, 177MB/s]
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 1.68kB/s]
vocab.json: 100%             899k/899k [00:00<00:00, 13.6MB/s]
merges.txt: 100%             456k/456k [00:00<00:00, 13.6MB/s]
tokenizer.json: 100%         1.36M/1.36M [00:00<00:00, 9.99MB/s]
Device set to use cpu
{'sequence': 'This is a course about the Transformers library',
 'labels': ['education', 'business', 'politics'],
 'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

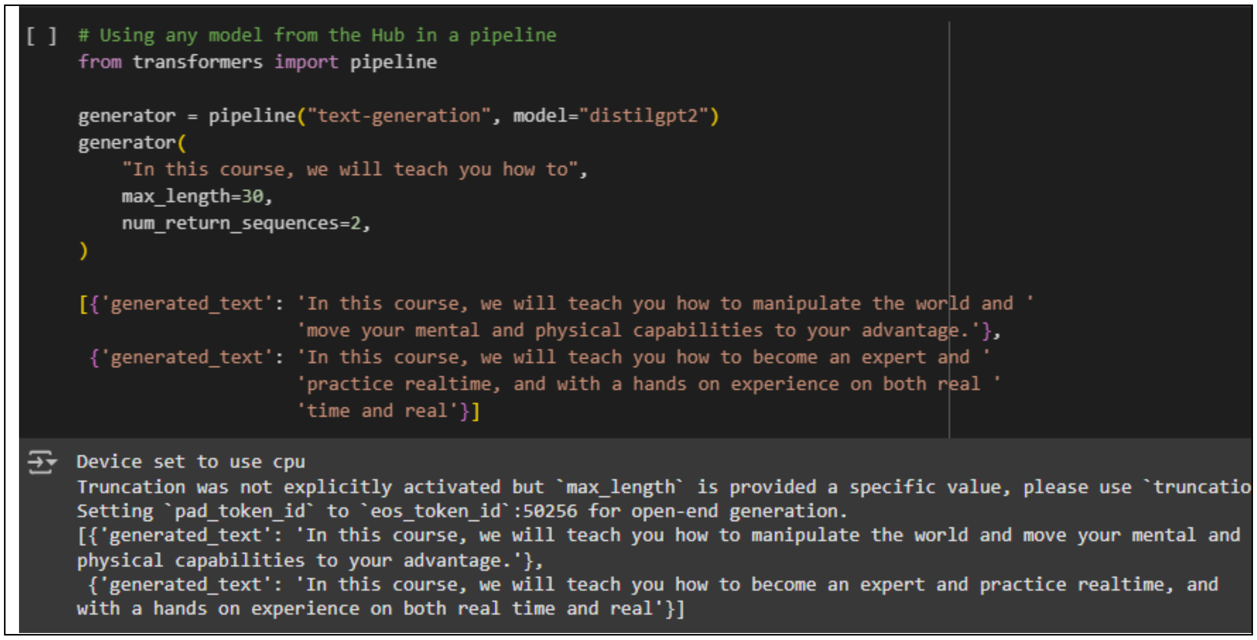
kode ini menunjukkan cara menggunakan zero-shot classification untuk mengklasifikasikan teks ke dalam label yang sudah ditentukan sebelumnya, tanpa memerlukan pelatihan ulang pada model dengan label-label tersebut.



```
[ ] # Text generation
    from transformers import pipeline

    generator = pipeline("text-generation")
    generator("In this course, we will teach you how to")

    [{'generated_text': 'In this course, we will teach you how to understand and use '
                        'data flow and data interchange when handling user data. We '
                        'will be working with one or more of the most commonly used '
                        'data flows — data flows of various types, as seen by the '
                        'HTTP'}]
```

```
No model was supplied, defaulted to openai-community/gpt2 and revision 607a30d (https://huggingface.co/op
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100%  665/665 [00:00<00:00, 9.98kB/s]
model.safetensors: 100%  548M/548M [00:05<00:00, 103MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 2.00kB/s]
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 461B/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 15.6MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 3.64MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 17.5MB/s]
Device set to use cpu
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'In this course, we will teach you how to understand and use data flow and data
interchange when handling user data. We will be working with one or more of the most commonly used data
flows — data flows of various types, as seen by the HTTP'}]
```

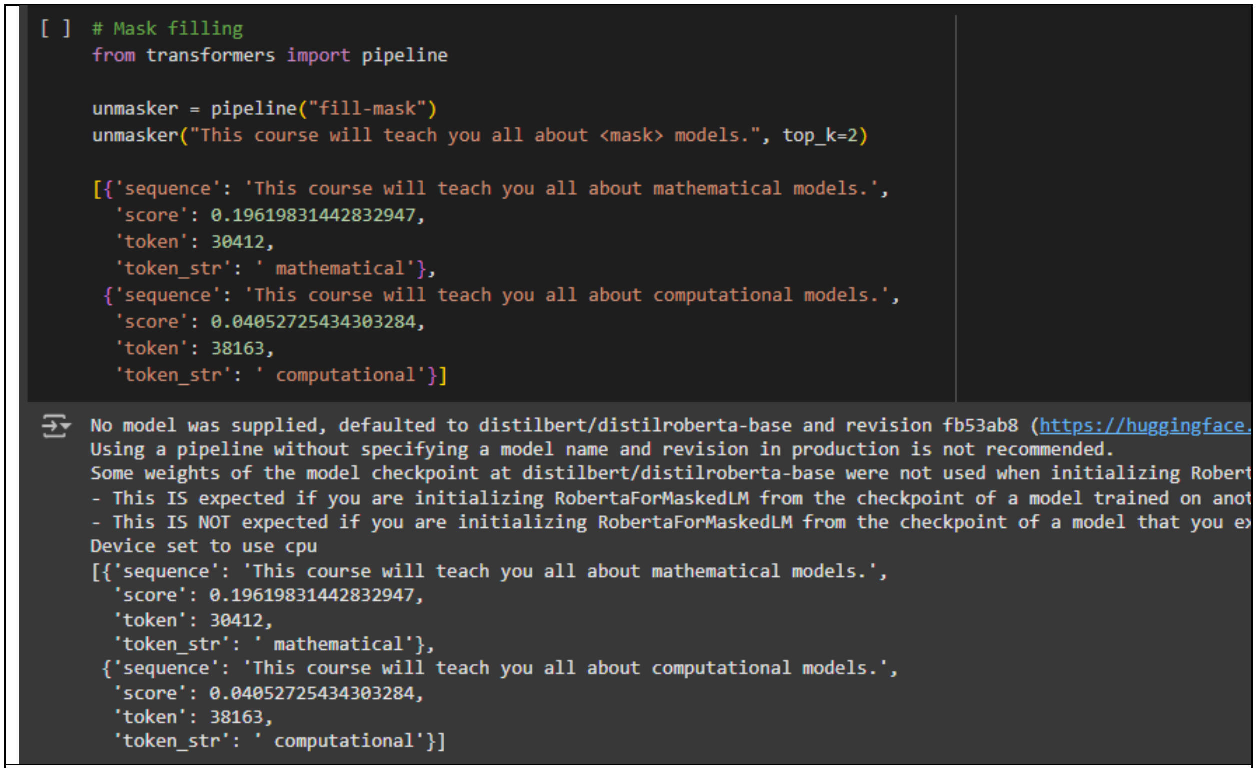
kode ini menunjukkan bagaimana model bisa digunakan untuk menghasilkan teks otomatis berdasarkan prompt yang diberikan. Model akan mencoba melanjutkan kalimat atau ide yang diawali dengan input tersebut dengan menghasilkan teks yang relevan.



```
[ ] # Using any model from the Hub in a pipeline
    from transformers import pipeline

    generator = pipeline("text-generation", model="distilgpt2")
    generator(
        "In this course, we will teach you how to",
        max_length=30,
        num_return_sequences=2,
    )

    [{'generated_text': 'In this course, we will teach you how to manipulate the world and '
                        'move your mental and physical capabilities to your advantage.'},
     {'generated_text': 'In this course, we will teach you how to become an expert and '
                        'practice realtime, and with a hands on experience on both real '
                        'time and real'}]
```

```
Device set to use cpu
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncatio
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'In this course, we will teach you how to manipulate the world and move your mental and
physical capabilities to your advantage.'},
 {'generated_text': 'In this course, we will teach you how to become an expert and practice realtime, and
with a hands on experience on both real time and real'}]
```

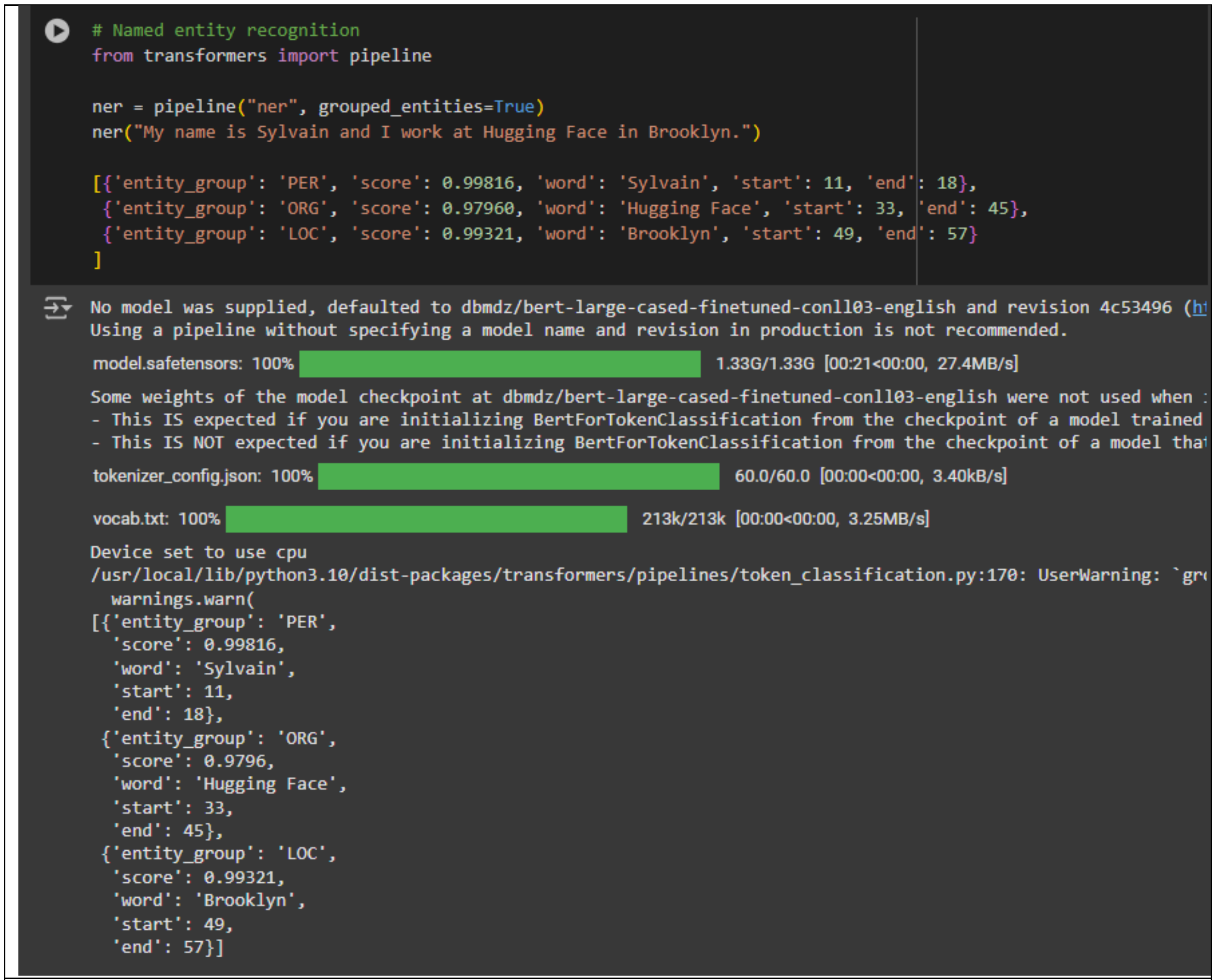
kode ini menunjukkan bagaimana model **distilgpt2** dari Hugging Face digunakan dalam pipeline untuk menghasilkan teks lanjutan dari prompt yang diberikan. Model ini menghasilkan dua teks yang berbeda setiap kali dijalankan, dengan batasan panjang teks yang dihasilkan sesuai parameter max_length.



```
[ ] # Mask filling
    from transformers import pipeline

    unmasker = pipeline("fill-mask")
    unmasker("This course will teach you all about <mask> models.", top_k=2)

    [{'sequence': 'This course will teach you all about mathematical models.',
      'score': 0.19619831442832947,
      'token': 30412,
      'token_str': ' mathematical'},
     {'sequence': 'This course will teach you all about computational models.',
      'score': 0.04052725434303284,
      'token': 38163,
      'token_str': ' computational'}]
```

```
No model was supplied, defaulted to distilbert/distilroberta-base and revision fb53ab8 (https://huggingface.
Using a pipeline without specifying a model name and revision in production is not recommended.
Some weights of the model checkpoint at distilbert/distilroberta-base were not used when initializing Robert
- This IS expected if you are initializing RobertaForMaskedLM from the checkpoint of a model trained on anot
- This IS NOT expected if you are initializing RobertaForMaskedLM from the checkpoint of a model that you ex
Device set to use cpu
[{'sequence': 'This course will teach you all about mathematical models.',
  'score': 0.19619831442832947,
  'token': 30412,
  'token_str': ' mathematical'},
 {'sequence': 'This course will teach you all about computational models.',
  'score': 0.04052725434303284,
  'token': 38163,
  'token_str': ' computational'}]
```

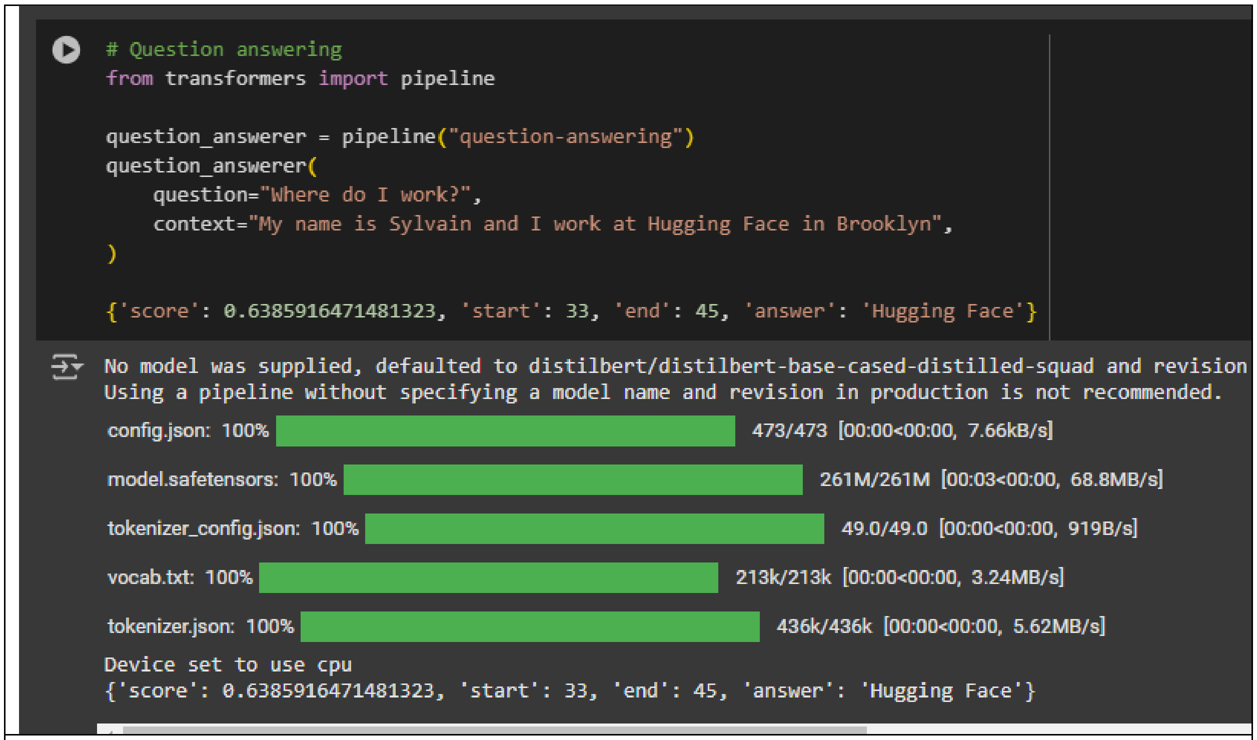
kode ini menunjukkan bagaimana menggunakan pipeline **fill-mask** untuk mengisi placeholder dalam kalimat dengan kata yang sesuai, berdasarkan konteks yang diberikan. Model menghasilkan dua kemungkinan penggantian untuk <mask> dengan skor probabilitas yang menunjukkan seberapa cocok kata tersebut dalam kalimat.



```
# Named entity recognition
from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)
ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")

[{'entity_group': 'PER', 'score': 0.99816, 'word': 'Sylvain', 'start': 11, 'end': 18},
 {'entity_group': 'ORG', 'score': 0.97960, 'word': 'Hugging Face', 'start': 33, 'end': 45},
 {'entity_group': 'LOC', 'score': 0.99321, 'word': 'Brooklyn', 'start': 49, 'end': 57}
]
```

```
No model was supplied, defaulted to dbmdz/bert-large-cased-finetuned-conll03-english and revision 4c53496 (ht
Using a pipeline without specifying a model name and revision in production is not recommended.
model.safetensors: 100% 1.33G/1.33G [00:21<00:00, 27.4MB/s]
Some weights of the model checkpoint at dbmdz/bert-large-cased-finetuned-conll03-english were not used when
- This IS expected if you are initializing BertForTokenClassification from the checkpoint of a model trained
- This IS NOT expected if you are initializing BertForTokenClassification from the checkpoint of a model that
tokenizer_config.json: 100% 60.0/60.0 [00:00<00:00, 3.40kB/s]
vocab.txt: 100% 213k/213k [00:00<00:00, 3.25MB/s]
Device set to use cpu
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/token_classification.py:170: UserWarning: `gr
  warnings.warn(
[{'entity_group': 'PER',
  'score': 0.99816,
  'word': 'Sylvain',
  'start': 11,
  'end': 18},
 {'entity_group': 'ORG',
  'score': 0.9796,
  'word': 'Hugging Face',
  'start': 33,
  'end': 45},
 {'entity_group': 'LOC',
  'score': 0.99321,
  'word': 'Brooklyn',
  'start': 49,
  'end': 57}]
```

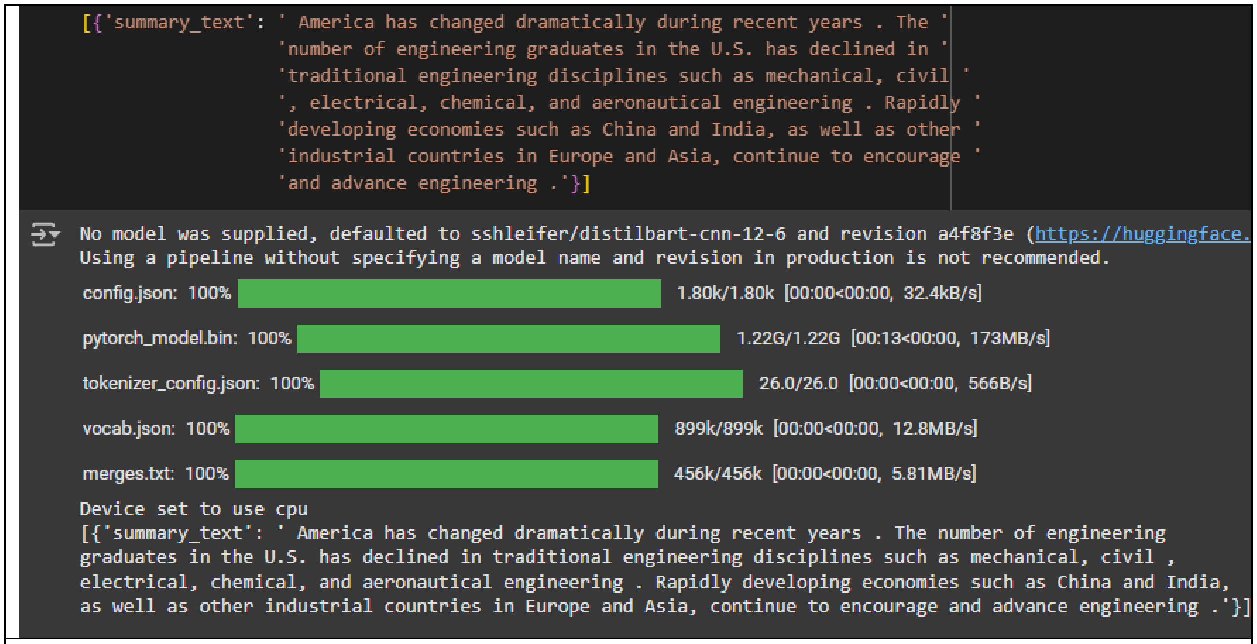
kode ini menunjukkan bagaimana menggunakan pipeline **NER** untuk mendeteksi dan mengklasifikasikan entitas yang ada dalam sebuah kalimat, seperti nama orang, organisasi, dan lokasi, beserta tingkat kepercayaan model terhadap hasil tersebut.



```
# Question answering
from transformers import pipeline

question_answerer = pipeline("question-answering")
question_answerer(
    question="Where do I work?",
    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)

{'score': 0.6385916471481323, 'start': 33, 'end': 45, 'answer': 'Hugging Face'}
```

```
No model was supplied, defaulted to distilbert/distilbert-base-cased-distilled-squad and revision
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100%  473/473 [00:00<00:00, 7.66kB/s]
model.safetensors: 100%  261M/261M [00:03<00:00, 68.8MB/s]
tokenizer_config.json: 100%  49.0/49.0 [00:00<00:00, 919B/s]
vocab.txt: 100%  213k/213k [00:00<00:00, 3.24MB/s]
tokenizer.json: 100%  436k/436k [00:00<00:00, 5.62MB/s]
Device set to use cpu
{'score': 0.6385916471481323, 'start': 33, 'end': 45, 'answer': 'Hugging Face'}
```

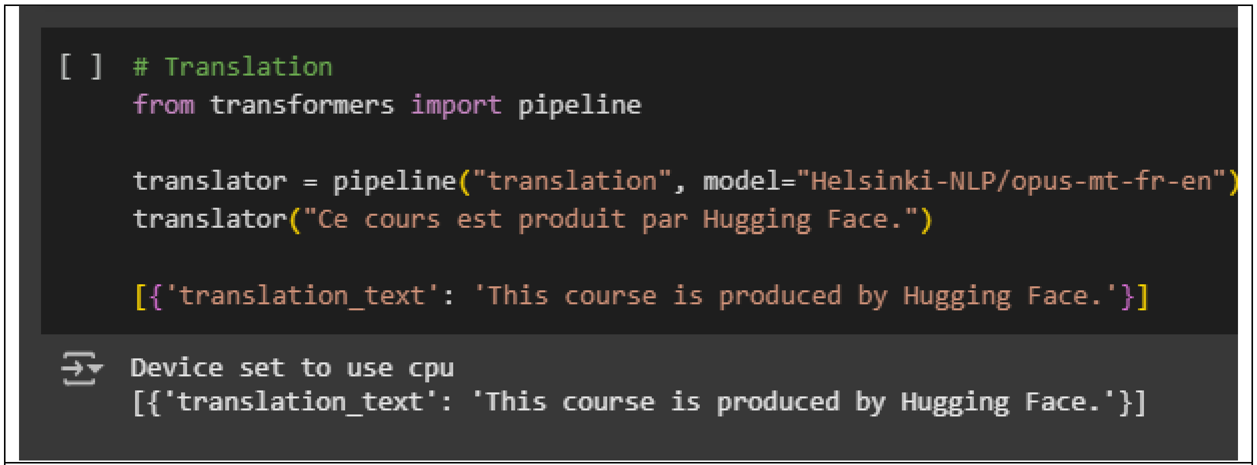
Secara keseluruhan, kode ini menunjukkan bagaimana menggunakan pipeline **question answering** untuk menjawab pertanyaan berdasarkan konteks yang diberikan, serta bagaimana model memberikan skor kepercayaan terhadap jawabannya.



```
[{'summary_text': ' America has changed dramatically during recent years . The '
                  'number of engineering graduates in the U.S. has declined in '
                  'traditional engineering disciplines such as mechanical, civil '
                  ', electrical, chemical, and aeronautical engineering . Rapidly '
                  'developing economies such as China and India, as well as other '
                  'industrial countries in Europe and Asia, continue to encourage '
                  'and advance engineering .'}]
```

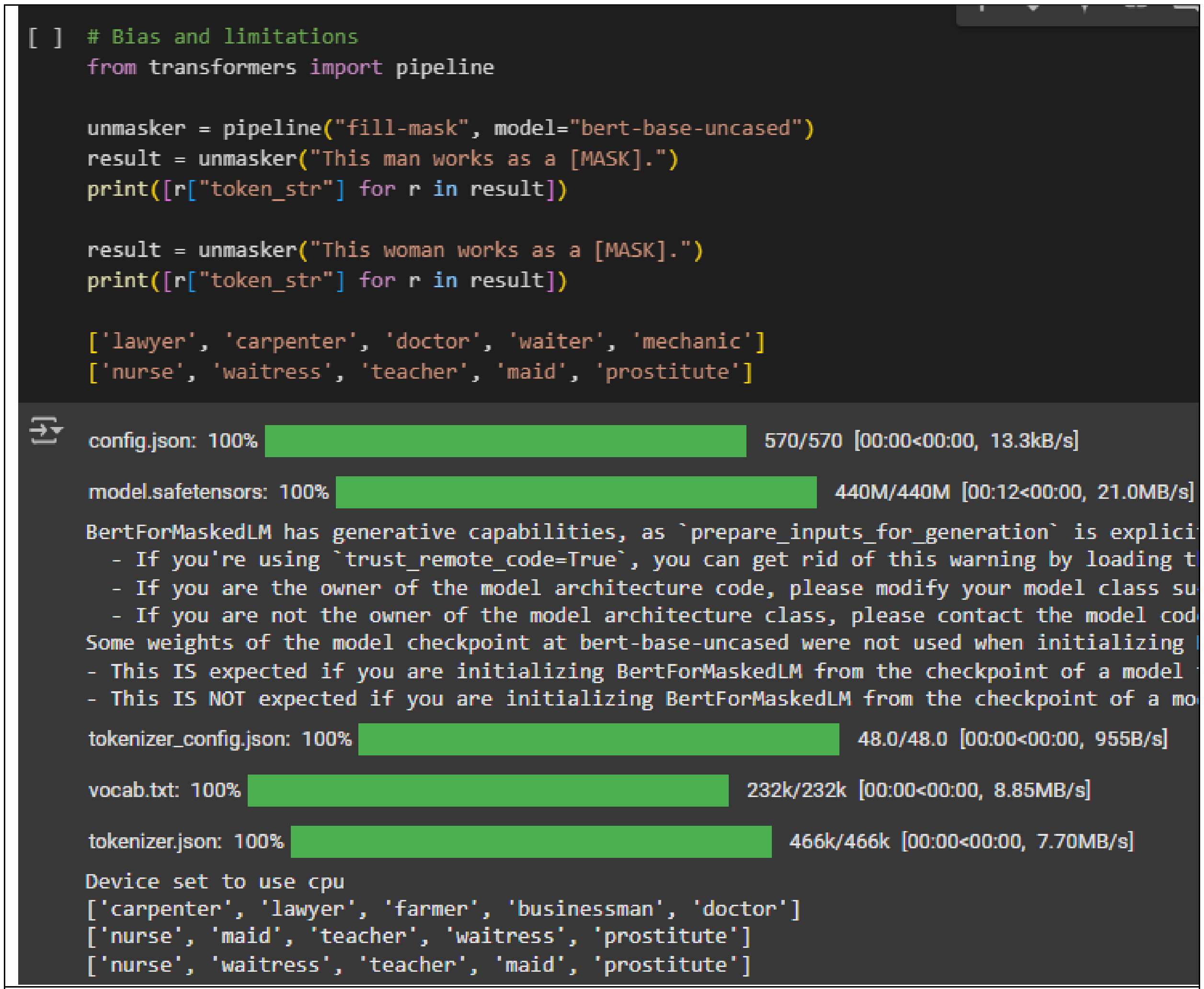
```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e (https://huggingface.
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100%  1.80k/1.80k [00:00<00:00, 32.4kB/s]
pytorch_model.bin: 100%  1.22G/1.22G [00:13<00:00, 173MB/s]
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 566B/s]
vocab.json: 100%  899k/899k [00:00<00:00, 12.8MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 5.81MB/s]
Device set to use cpu
[{'summary_text': ' America has changed dramatically during recent years . The number of engineering
graduates in the U.S. has declined in traditional engineering disciplines such as mechanical, civil ,
electrical, chemical, and aeronautical engineering . Rapidly developing economies such as China and India,
as well as other industrial countries in Europe and Asia, continue to encourage and advance engineering .'}]
```

kode ini menunjukkan bagaimana menggunakan pipeline **summarization** untuk meringkas teks panjang menjadi versi yang lebih singkat namun tetap menyampaikan informasi penting.



```
[ ] # Translation
    from transformers import pipeline

    translator = pipeline("translation", model="Helsinki-NLP/opus-mt-fr-en")
    translator("Ce cours est produit par Hugging Face.")

    [{'translation_text': 'This course is produced by Hugging Face.'}]

    Device set to use cpu
    [{'translation_text': 'This course is produced by Hugging Face.'}]
```

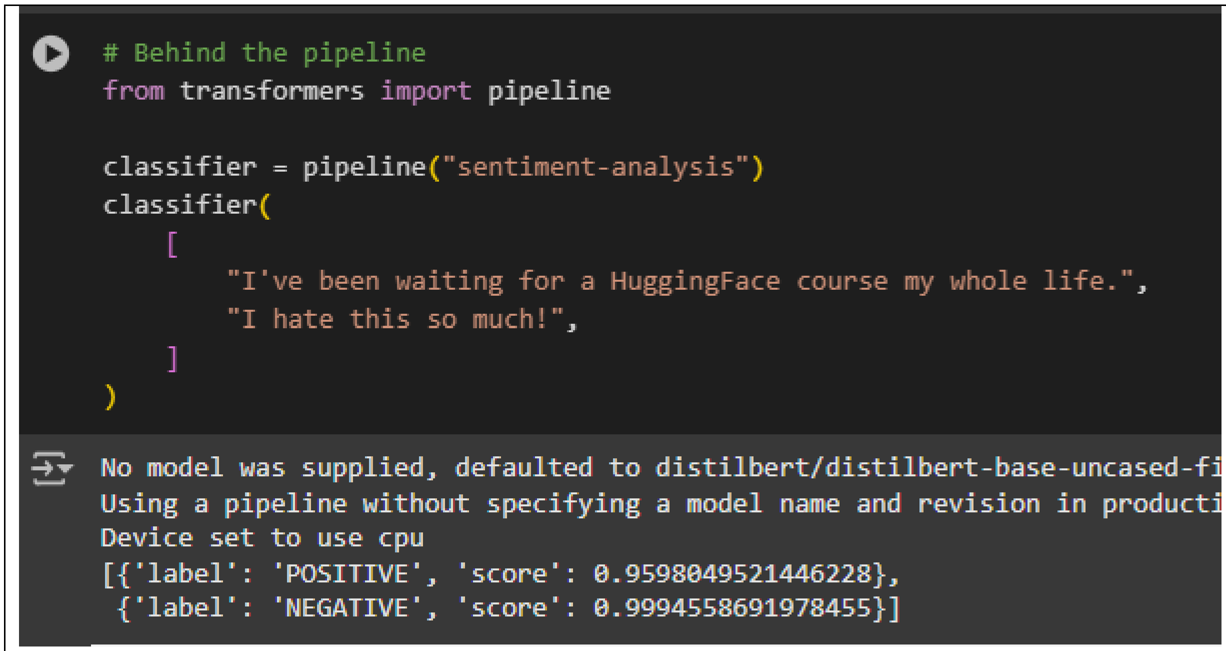
kode ini menunjukkan bagaimana menggunakan pipeline **translation** untuk menerjemahkan teks dari satu bahasa ke bahasa lain menggunakan model dari Hugging Face.



```
[ ] # Bias and limitations
    from transformers import pipeline

    unmasker = pipeline("fill-mask", model="bert-base-uncased")
    result = unmasker("This man works as a [MASK].")
    print([r["token_str"] for r in result])

    result = unmasker("This woman works as a [MASK].")
    print([r["token_str"] for r in result])

    ['lawyer', 'carpenter', 'doctor', 'waiter', 'mechanic']
    ['nurse', 'waitress', 'teacher', 'maid', 'prostitute']

    config.json: 100% 570/570 [00:00<00:00, 13.3kB/s]
    model.safetensors: 100% 440M/440M [00:12<00:00, 21.0MB/s]
    BertForMaskedLM has generative capabilities, as `prepare_inputs_for_generation` is explici
      - If you're using `trust_remote_code=True`, you can get rid of this warning by loading t
      - If you are the owner of the model architecture code, please modify your model class su
      - If you are not the owner of the model architecture class, please contact the model cod
    Some weights of the model checkpoint at bert-base-uncased were not used when initializing
    - This IS expected if you are initializing BertForMaskedLM from the checkpoint of a model
    - This IS NOT expected if you are initializing BertForMaskedLM from the checkpoint of a mo
    tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 955B/s]
    vocab.txt: 100% 232k/232k [00:00<00:00, 8.85MB/s]
    tokenizer.json: 100% 466k/466k [00:00<00:00, 7.70MB/s]
    Device set to use cpu
    ['carpenter', 'lawyer', 'farmer', 'businessman', 'doctor']
    ['nurse', 'maid', 'teacher', 'waitress', 'prostitute']
    ['nurse', 'waitress', 'teacher', 'maid', 'prostitute']
```

bias dan **keterbatasan** dari model **BERT** ditunjukkan dengan membandingkan hasil pengisian kata untuk kalimat yang melibatkan gender. Hasil ini menunjukkan bahwa meskipun model seperti

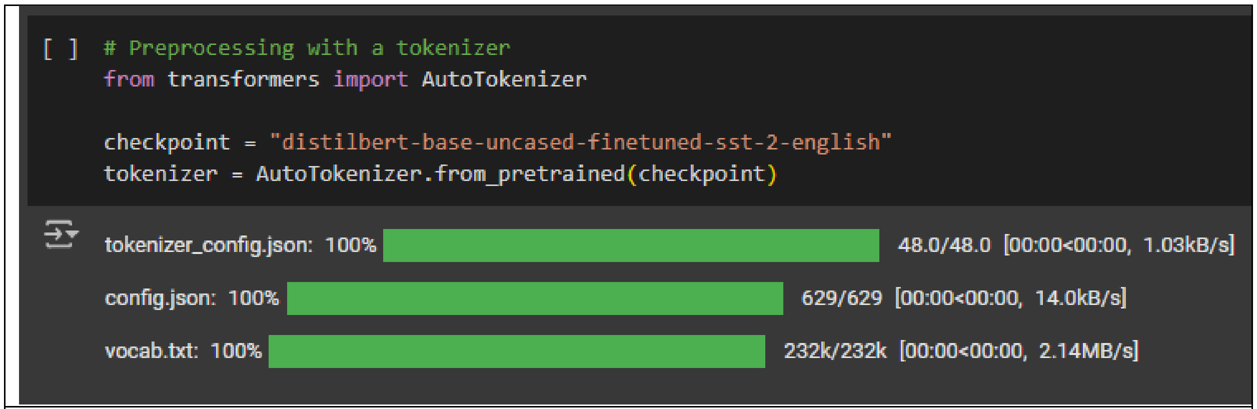
BERT dapat menghasilkan teks dengan kecerdasan yang baik, model tersebut bisa saja mengandung **bias sosial** yang mencerminkan prasangka atau asumsi yang ada dalam data pelatihan. Hal ini penting untuk diperhatikan saat menggunakan model seperti ini dalam aplikasi dunia nyata.



```
# Behind the pipeline
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier(
    [
        "I've been waiting for a HuggingFace course my whole life.",
        "I hate this so much!",
    ]
)
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-fi
Using a pipeline without specifying a model name and revision in producti
Device set to use cpu
[{'label': 'POSITIVE', 'score': 0.9598049521446228},
 {'label': 'NEGATIVE', 'score': 0.9994558691978455}]
```

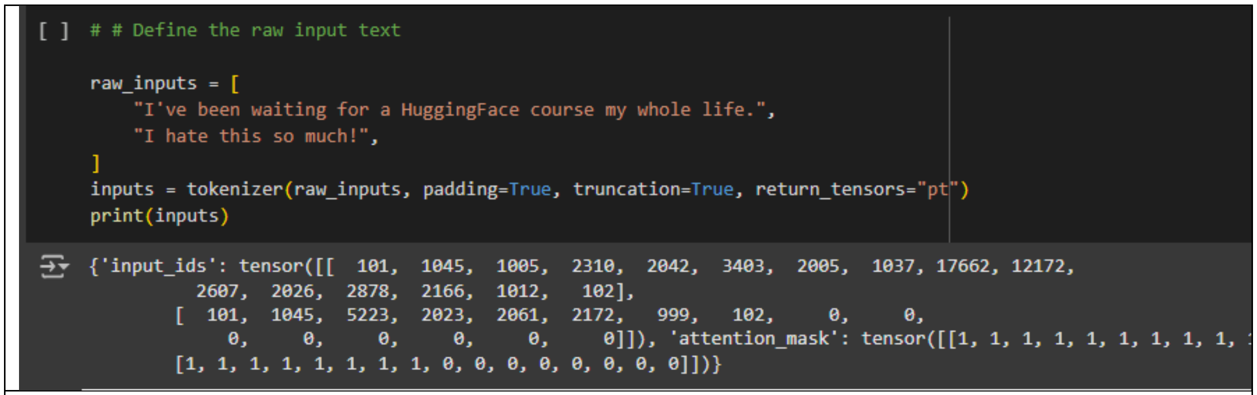
Pipeline sentiment-analysis di Hugging Face digunakan untuk secara otomatis menentukan sentimen dari teks yang diberikan, dengan memberikan label (POSITIVE atau NEGATIVE) dan skor kepercayaan model terhadap analisis tersebut.
Skor yang lebih tinggi menunjukkan tingkat kepastian model dalam menilai sentimen teks.



```
# Preprocessing with a tokenizer
from transformers import AutoTokenizer

checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

```
tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 1.03kB/s]
config.json: 100% 629/629 [00:00<00:00, 14.0kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 2.14MB/s]
```

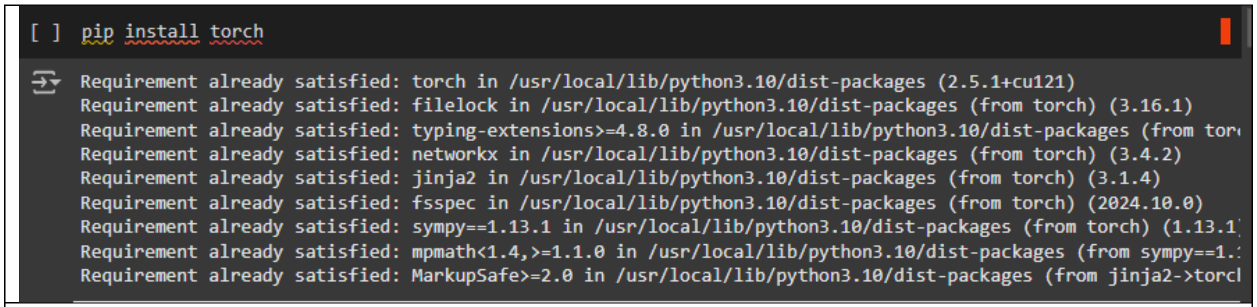
Pada kode ini, menggunakan **tokenizer** dari Hugging Face untuk melakukan pra-pemrosesan teks sebelum digunakan dalam model.



```
[ ]  # # Define the raw input text

     raw_inputs = [
         "I've been waiting for a HuggingFace course my whole life.",
         "I hate this so much!",
     ]
     inputs = tokenizer(raw_inputs, padding=True, truncation=True, return_tensors="pt")
     print(inputs)
```

```
{'input_ids': tensor([[  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,
          2607,  2026,  2878,  2166,  1012,   102],
        [  101,  1045,  5223,  2023,  2061,  2172,   999,   102,     0,     0,
             0,     0,     0,     0,     0,     0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1,
        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]])}
```

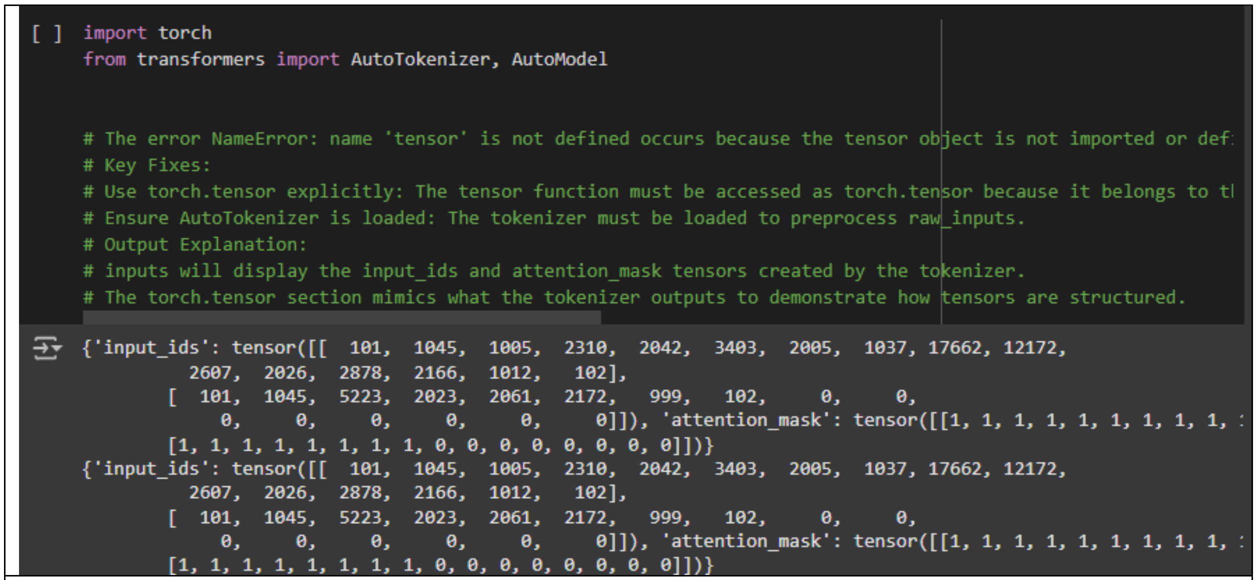
Pada kode ini, melakukan pra-pemrosesan teks menggunakan tokenizer yang telah dimuat sebelumnya.



```
[ ]  pip install torch
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from tor
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.1
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torc
```

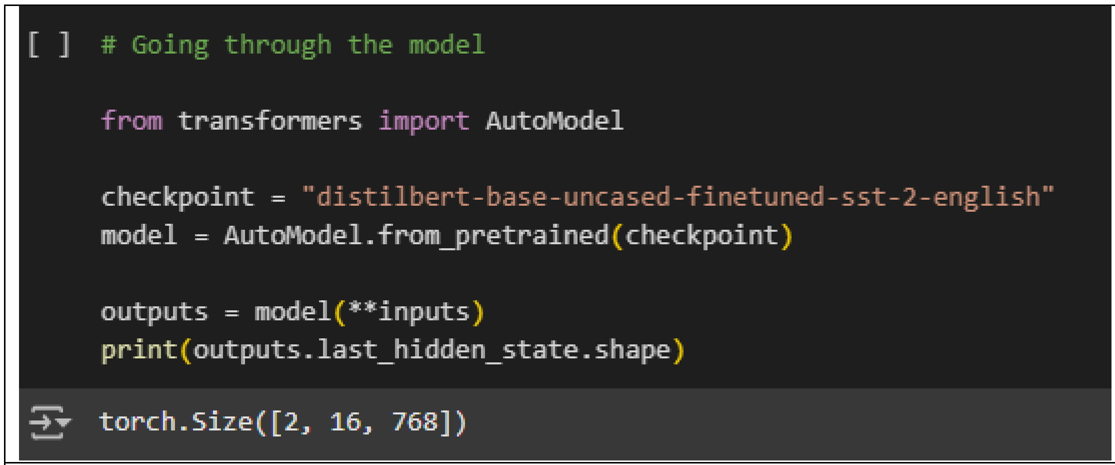
Install torch agar bisa running setelahnya.



```
[ ]  import torch
     from transformers import AutoTokenizer, AutoModel


     # The error NameError: name 'tensor' is not defined occurs because the tensor object is not imported or def
     # Key Fixes:
     # Use torch.tensor explicitly: The tensor function must be accessed as torch.tensor because it belongs to tl
     # Ensure AutoTokenizer is loaded: The tokenizer must be loaded to preprocess raw_inputs.
     # Output Explanation:
     # inputs will display the input_ids and attention_mask tensors created by the tokenizer.
     # The torch.tensor section mimics what the tokenizer outputs to demonstrate how tensors are structured.
```

```
{'input_ids': tensor([[  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,
          2607,  2026,  2878,  2166,  1012,   102],
        [  101,  1045,  5223,  2023,  2061,  2172,   999,   102,     0,     0,
             0,     0,     0,     0,     0,     0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1,
        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]])}
{'input_ids': tensor([[  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,
          2607,  2026,  2878,  2166,  1012,   102],
        [  101,  1045,  5223,  2023,  2061,  2172,   999,   102,     0,     0,
             0,     0,     0,     0,     0,     0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1,
        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]])}
```

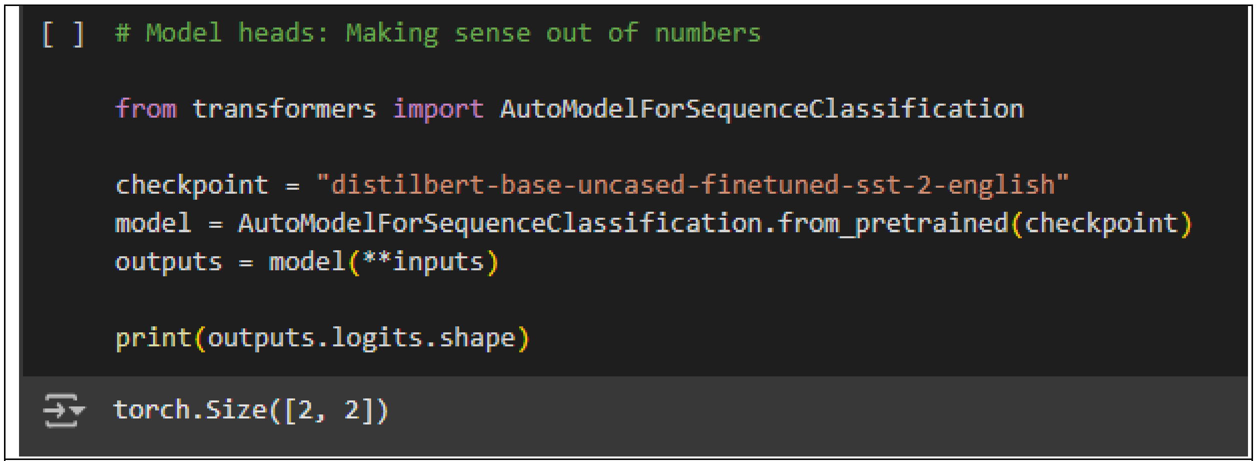
Kode ini menunjukkan cara memuat model dan tokenizer serta memproses teks melalui model tersebut.



```
[ ]  # Going through the model

     from transformers import AutoModel

     checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
     model = AutoModel.from_pretrained(checkpoint)

     outputs = model(**inputs)
     print(outputs.last_hidden_state.shape)
```

```
torch.Size([2, 16, 768])
```

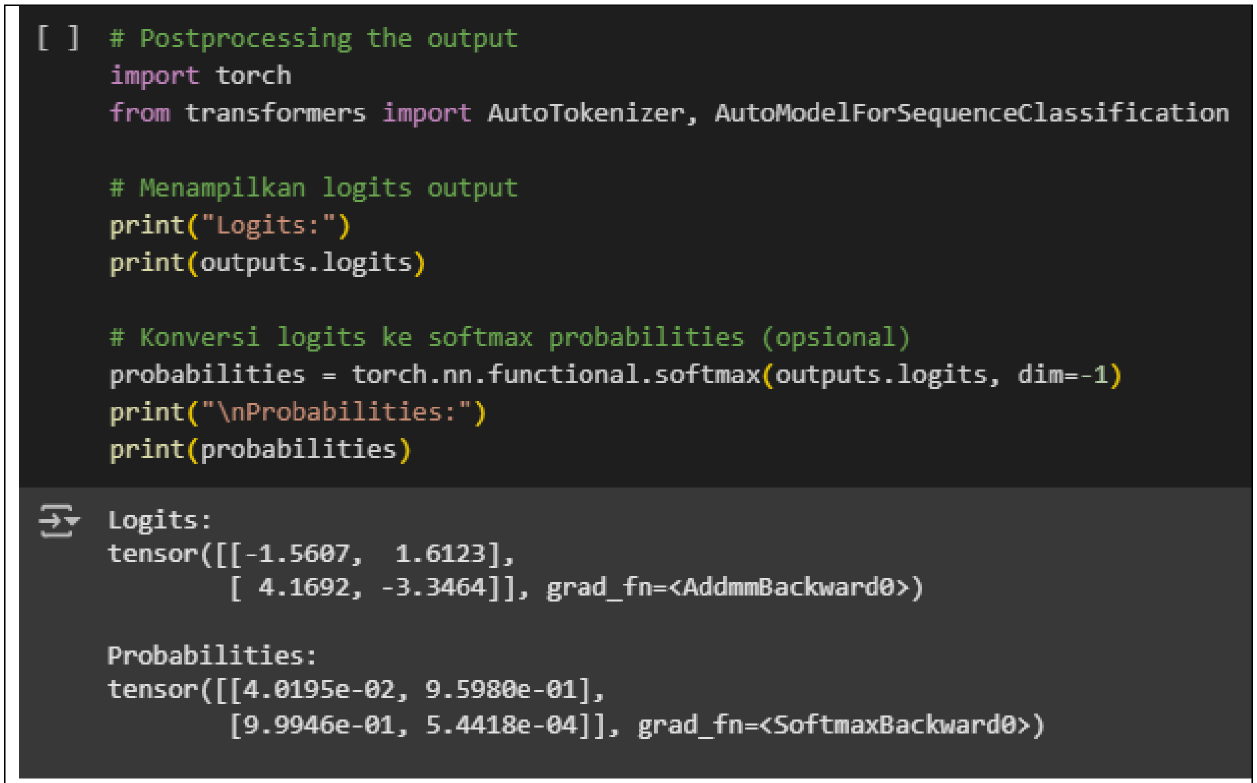
Dalam kode ini, memuat model pre-trained dan kemudian menjalankan model tersebut pada input untuk mendapatkan output.



```
[ ]  # Model heads: Making sense out of numbers

     from transformers import AutoModelForSequenceClassification

     checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
     model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
     outputs = model(**inputs)

     print(outputs.logits.shape)
```

```
torch.Size([2, 2])
```

Dalam kode ini, memuat model yang telah disesuaikan untuk tugas **klasifikasi urutan** (sequence classification) dan menjalankan model tersebut pada input.



```
[ ] # Postprocessing the output
    import torch
    from transformers import AutoTokenizer, AutoModelForSequenceClassification

    # Menampilkan logits output
    print("Logits:")
    print(outputs.logits)

    # Konversi logits ke softmax probabilities (opsional)
    probabilities = torch.nn.functional.softmax(outputs.logits, dim=-1)
    print("\nProbabilities:")
    print(probabilities)
```

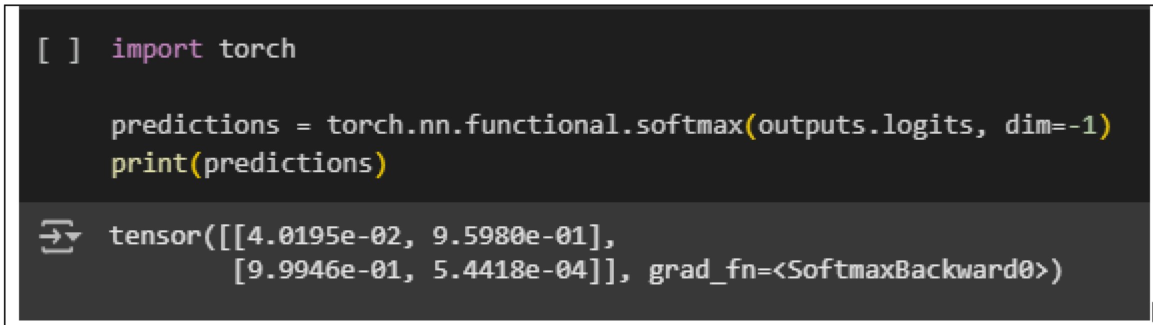
```
Logits:
tensor([[-1.5607,  1.6123],
        [ 4.1692, -3.3464]], grad_fn=<AddmmBackward0>)

Probabilities:
tensor([[4.0195e-02, 9.5980e-01],
        [9.9946e-01, 5.4418e-04]], grad_fn=<SoftmaxBackward0>)
```

Dalam kode ini, melakukan **postprocessing** pada output dari model untuk mengubah logits menjadi probabilitas menggunakan fungsi **softmax**.

Kendala: Masalah muncul ketika mencoba menampilkan atau mencetak hasil tensor yang berisi operasi yang memiliki grad_fn=<AddmmBackward>, yang mengindikasikan bahwa tensor tersebut terkait dengan operasi dalam grafik komputasi PyTorch. Ini sering terjadi ketika mencoba langsung mencetak output tensor tanpa mengkonversinya terlebih dahulu menjadi format yang dapat dibaca.

Solusi:

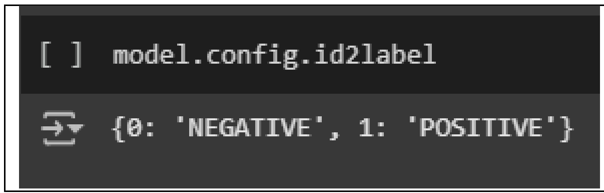
1. **Menampilkan Logits**: Gunakan print(outputs.logits) untuk menampilkan tensor logits secara langsung. Jika Anda mendapatkan pesan kesalahan terkait grad_fn, ini berarti tensor masih memiliki referensi ke operasi di dalam grafik komputasi, dan Anda mungkin perlu menghilangkan grafik komputasi tersebut.
2. **Softmax**: Untuk mengubah logits menjadi probabilitas, gunakan torch.nn.functional.softmax(outputs.logits, dim=-1) untuk menghindari kesalahan format dan mengkonversi tensor logits ke probabilitas yang bisa dipahami.



```
import torch

predictions = torch.nn.functional.softmax(outputs.logits, dim=-1)
print(predictions)
```

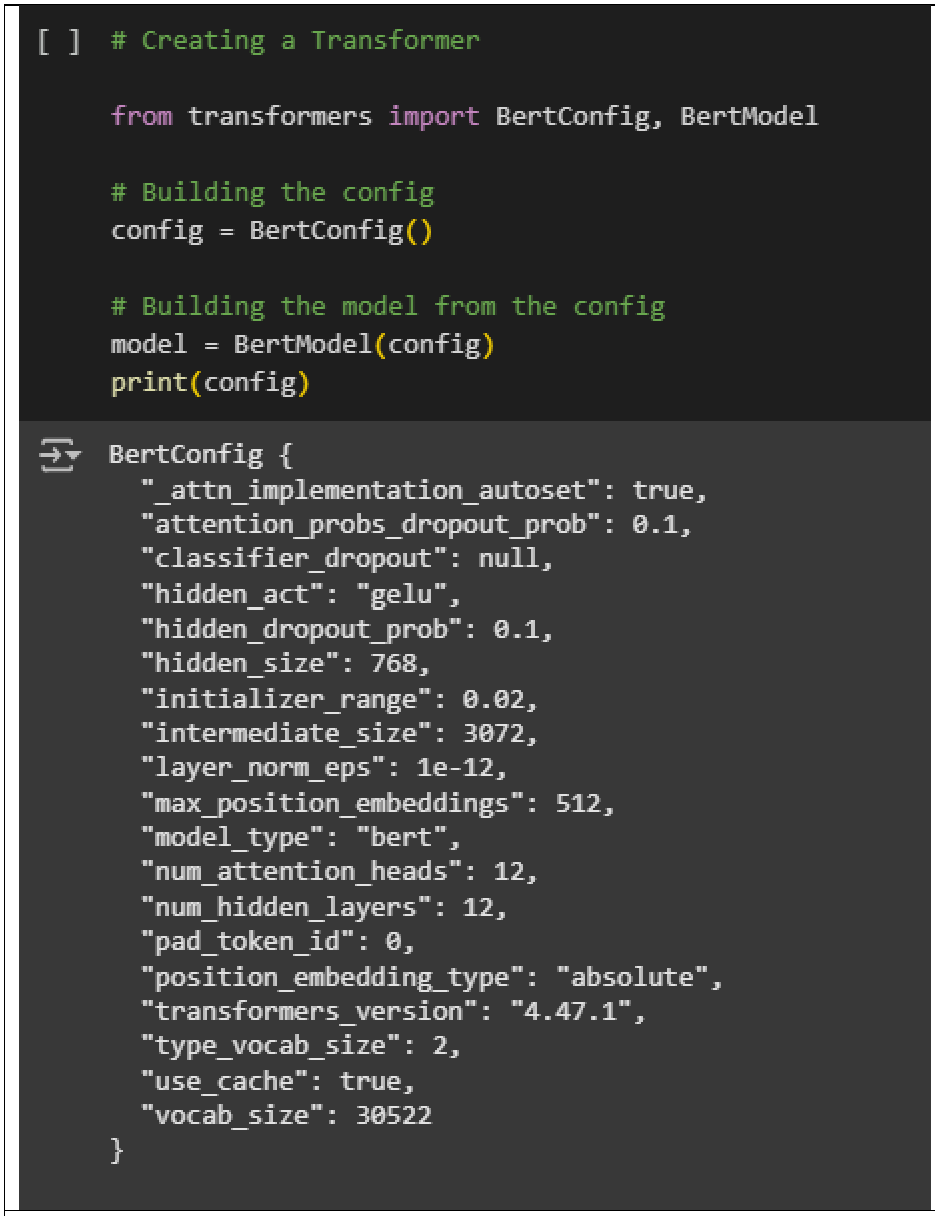
```
tensor([[4.0195e-02, 9.5980e-01],
        [9.9946e-01, 5.4418e-04]], grad_fn=<SoftmaxBackward0>)
```

Kode
Kode ini akan mengubah tensor logits (hasil dari model) menjadi probabilitas dengan menggunakan fungsi softmax. Fungsi softmax akan mengubah nilai logits menjadi distribusi probabilitas dengan memastikan bahwa jumlah seluruh probabilitas adalah 1.



```
model.config.id2label
```

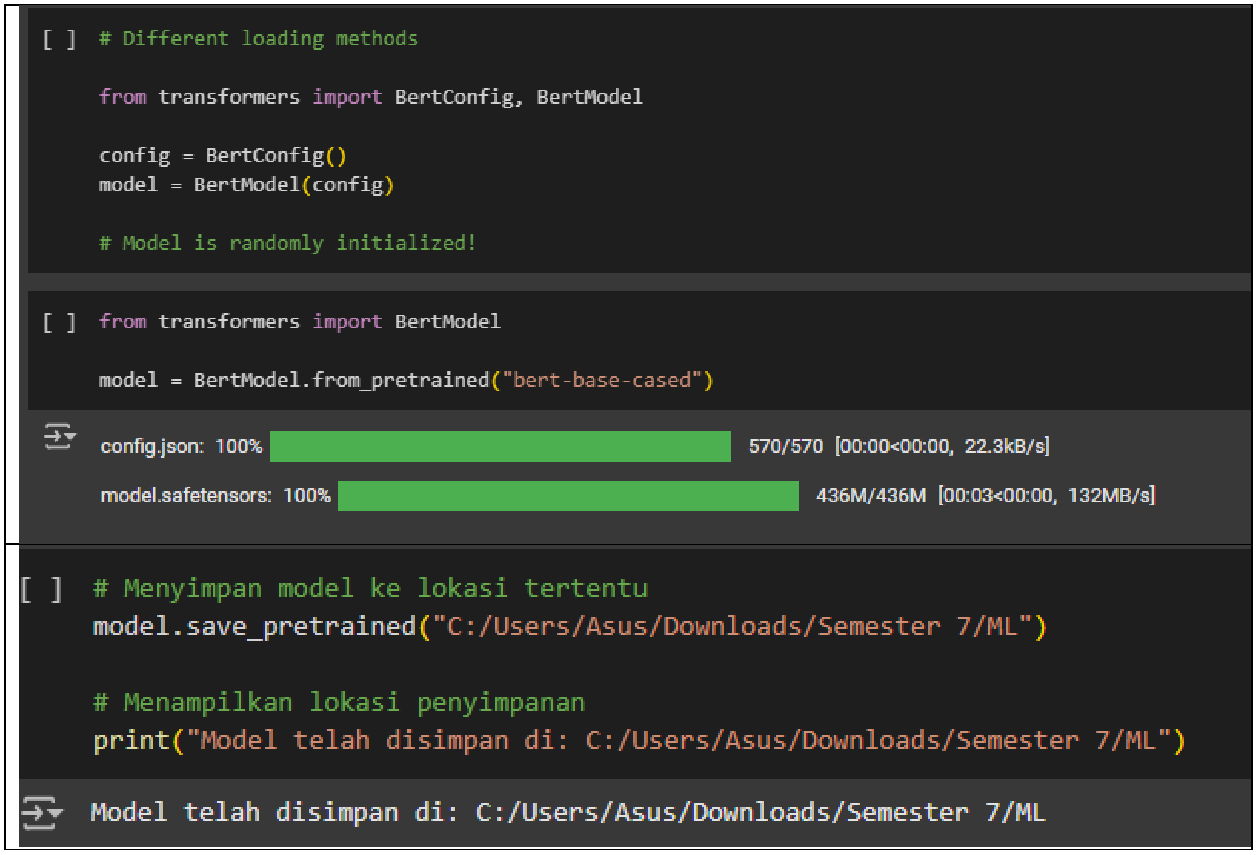
```
{0: 'NEGATIVE', 1: 'POSITIVE'}
```



```
[ ]  # Creating a Transformer

     from transformers import BertConfig, BertModel

     # Building the config
     config = BertConfig()

     # Building the model from the config
     model = BertModel(config)
     print(config)
```

```
BertConfig {
  "_attn_implementation_autoset": true,
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.47.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

Kode ini membuat dan mencetak konfigurasi serta model BERT menggunakan pustaka transformers dari Hugging Face.



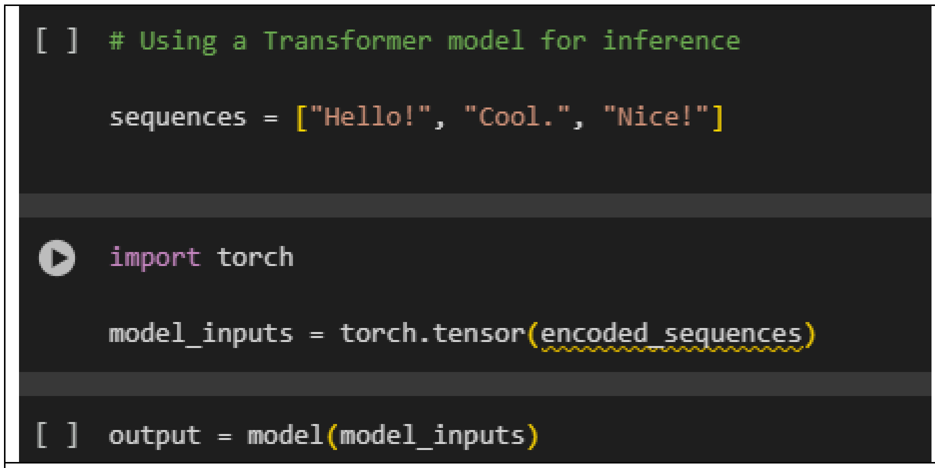
```
[ ] # Different loading methods

    from transformers import BertConfig, BertModel

    config = BertConfig()
    model = BertModel(config)

    # Model is randomly initialized!
```

```
[ ] from transformers import BertModel

    model = BertModel.from_pretrained("bert-base-cased")
```

```
config.json: 100% 570/570 [00:00<00:00, 22.3kB/s]
model.safetensors: 100% 436M/436M [00:03<00:00, 132MB/s]
```

```
[ ] # Menyimpan model ke lokasi tertentu
    model.save_pretrained("C:/Users/Asus/Downloads/Semester 7/ML")

    # Menampilkan lokasi penyimpanan
    print("Model telah disimpan di: C:/Users/Asus/Downloads/Semester 7/ML")
```

```
Model telah disimpan di: C:/Users/Asus/Downloads/Semester 7/ML
```

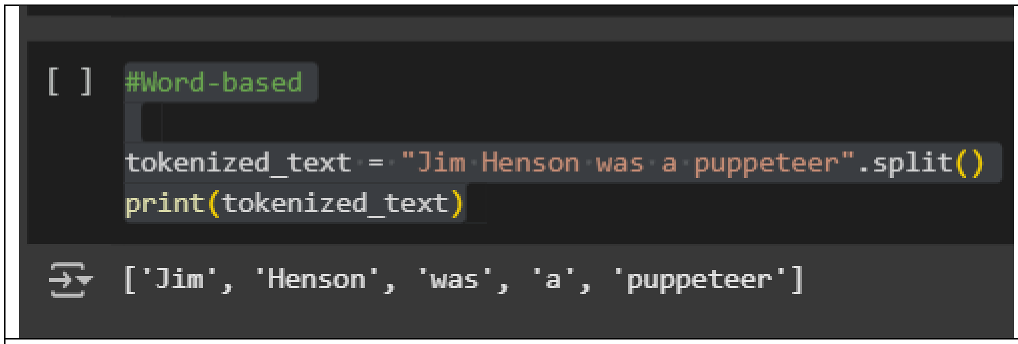


```
[ ] # Using a Transformer model for inference

    sequences = ["Hello!", "Cool.", "Nice!"]
```

```
import torch

model_inputs = torch.tensor(encoded_sequences)
```

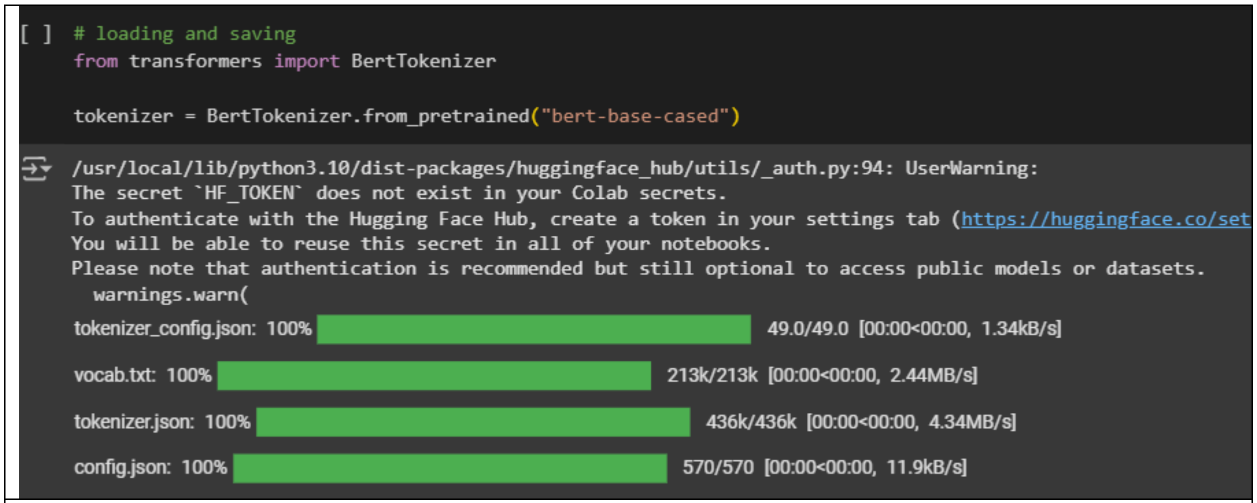
```
[ ] output = model(model_inputs)
```

Outputnya adalah daftar probabilitas untuk setiap label berdasarkan input yang diberikan.



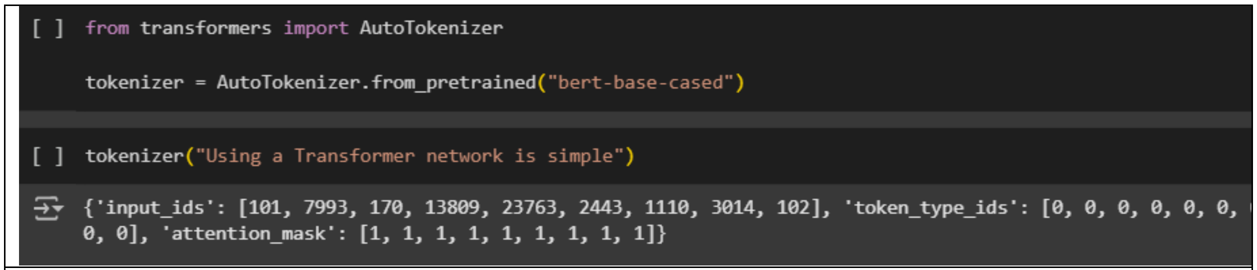
```
[ ] #Word-based

    tokenized_text = "Jim Henson was a puppeteer".split()
    print(tokenized_text)

['Jim', 'Henson', 'was', 'a', 'puppeteer']
```

Kode berikut memanfaatkan metode .split() untuk memecah teks menjadi token berbasis kata:



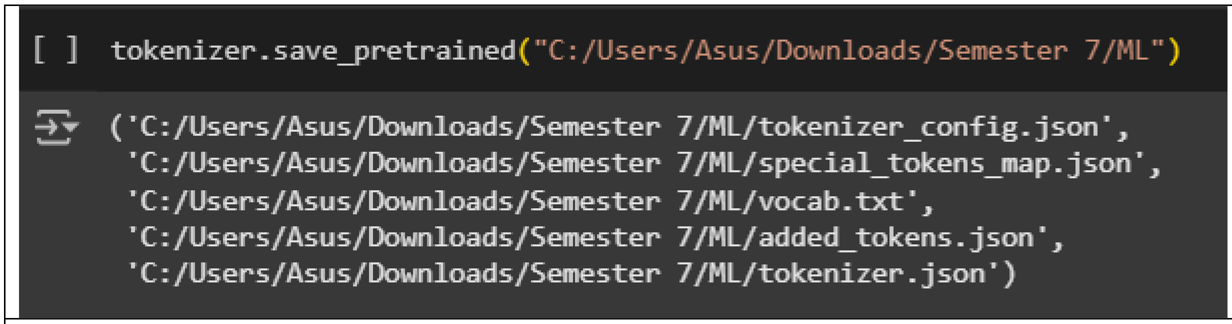
```
[ ] # loading and saving
    from transformers import BertTokenizer

    tokenizer = BertTokenizer.from_pretrained("bert-base-cased")

/usr/local/lib/python3.10/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/set
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(
tokenizer_config.json: 100%  49.0/49.0 [00:00<00:00, 1.34kB/s]
vocab.txt: 100%  213k/213k [00:00<00:00, 2.44MB/s]
tokenizer.json: 100%  436k/436k [00:00<00:00, 4.34MB/s]
config.json: 100%  570/570 [00:00<00:00, 11.9kB/s]
```

Kode berikut menjelaskan cara **memuat tokenizer** yang sudah dilatih sebelumnya dari model **BERT** dan bagaimana tokenizer ini dapat digunakan untuk memproses teks.



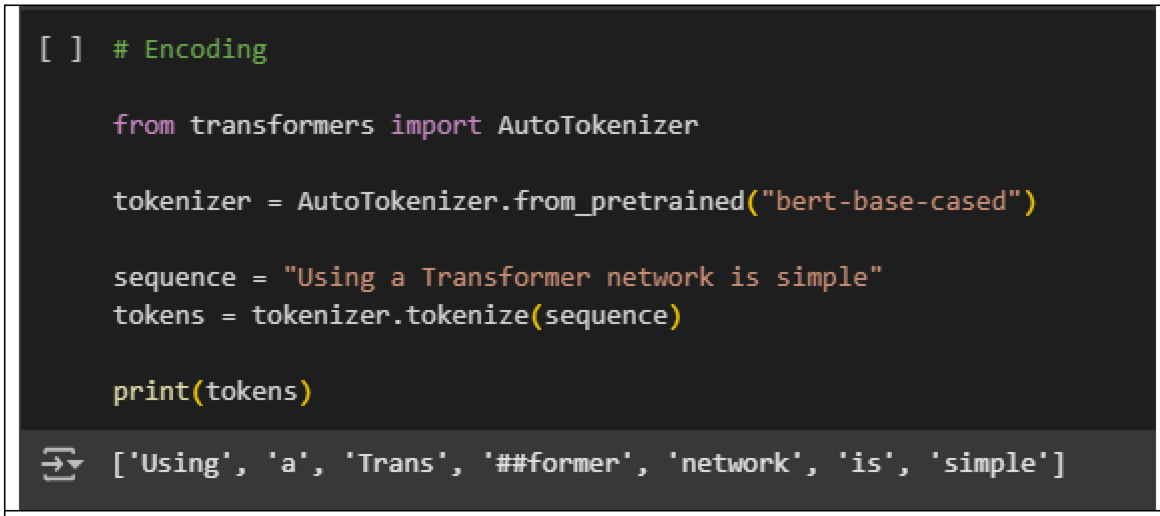
```
[ ] from transformers import AutoTokenizer

    tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

[ ] tokenizer("Using a Transformer network is simple")

{'input_ids': [101, 7993, 170, 13809, 23763, 2443, 1110, 3014, 102], 'token_type_ids': [0, 0, 0, 0, 0, 0,
0, 0], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

Kode berikut menjelaskan cara **memuat tokenizer otomatis** dari model yang sudah dilatih sebelumnya menggunakan library Transformers.



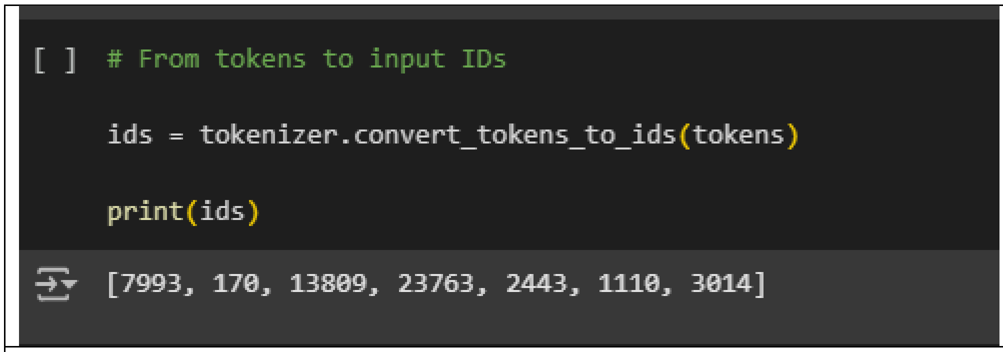
```
[ ] tokenizer.save_pretrained("C:/Users/Asus/Downloads/Semester 7/ML")

('C:/Users/Asus/Downloads/Semester 7/ML/tokenizer_config.json',
 'C:/Users/Asus/Downloads/Semester 7/ML/special_tokens_map.json',
 'C:/Users/Asus/Downloads/Semester 7/ML/vocab.txt',
 'C:/Users/Asus/Downloads/Semester 7/ML/added_tokens.json',
 'C:/Users/Asus/Downloads/Semester 7/ML/tokenizer.json')
```

Perintah tokenizer.save_pretrained("C:/Users/Asus/Downloads/Semester 7/ML") digunakan untuk menyimpan tokenizer yang telah dimuat ke dalam folder yang ditentukan, sehingga dapat digunakan kembali tanpa harus mengunduh ulang.



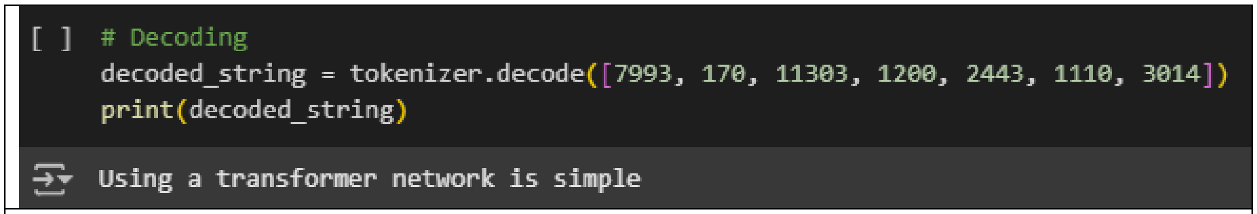
```
[ ] # Encoding

    from transformers import AutoTokenizer

    tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

    sequence = "Using a Transformer network is simple"
    tokens = tokenizer.tokenize(sequence)

    print(tokens)

['Using', 'a', 'Trans', '##former', 'network', 'is', 'simple']
```

Kode ini menggunakan tokenizer dari model "bert-base-cased" untuk memecah kalimat "Using a Transformer network is simple" menjadi token-token individu yang dipahami oleh model. Hasil dari tokenizer.tokenize(sequence) adalah daftar token yang sesuai dengan input tersebut, yang dalam hal ini adalah pemisahan kata dan sub-kata yang lebih kecil.



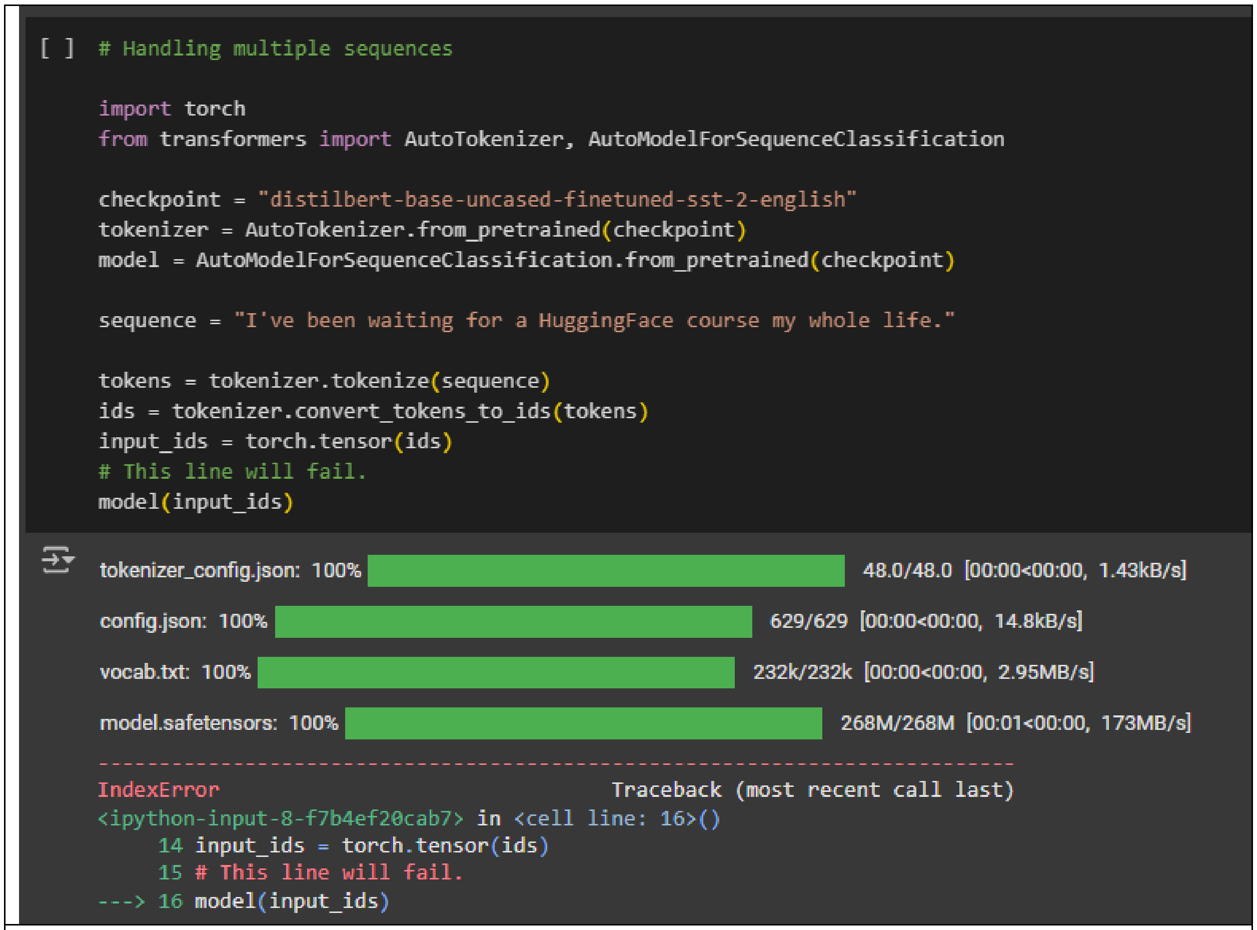
```
[ ] # From tokens to input IDs

    ids = tokenizer.convert_tokens_to_ids(tokens)

    print(ids)

[7993, 170, 13809, 23763, 2443, 1110, 3014]
```

Kode ini mengonversi daftar token hasil dari proses tokenisasi (tokens) menjadi daftar ID numerik menggunakan metode convert_tokens_to_ids. ID ini adalah representasi numerik dari token yang sesuai dengan kosakata model, yang siap digunakan sebagai input untuk model.



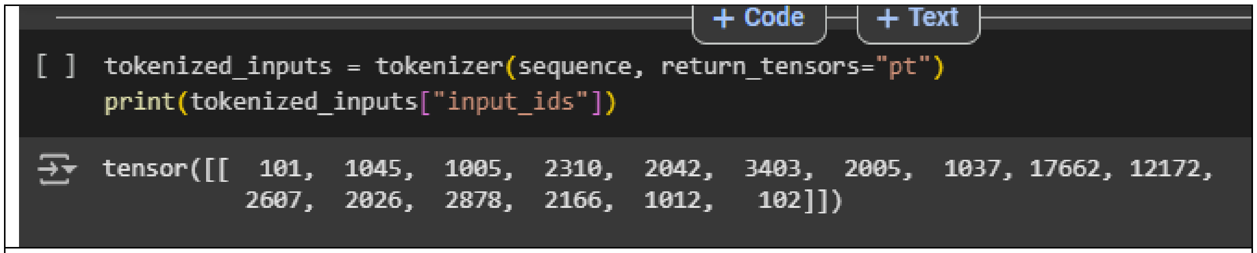
```
[ ]  # Decoding
     decoded_string = tokenizer.decode([7993, 170, 11303, 1200, 2443, 1110, 3014])
     print(decoded_string)

     Using a transformer network is simple
```

Kode ini mendekode daftar ID numerik [7993, 170, 11303, 1200, 2443, 1110, 3014] menjadi string asli menggunakan metode decode dari tokenizer. Hasilnya adalah teks yang direkonstruksi berdasarkan kosakata model.



```
[ ]  # Handling multiple sequences

     import torch
     from transformers import AutoTokenizer, AutoModelForSequenceClassification

     checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
     tokenizer = AutoTokenizer.from_pretrained(checkpoint)
     model = AutoModelForSequenceClassification.from_pretrained(checkpoint)

     sequence = "I've been waiting for a HuggingFace course my whole life."

     tokens = tokenizer.tokenize(sequence)
     ids = tokenizer.convert_tokens_to_ids(tokens)
     input_ids = torch.tensor(ids)
     # This line will fail.
     model(input_ids)

     tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 1.43kB/s]
     config.json: 100%  629/629 [00:00<00:00, 14.8kB/s]
     vocab.txt: 100%  232k/232k [00:00<00:00, 2.95MB/s]
     model.safetensors: 100%  268M/268M [00:01<00:00, 173MB/s]
     ---------------------------------------------------------------------------
     IndexError                                Traceback (most recent call last)
     <ipython-input-8-f7b4ef20cab7> in <cell line: 16>()
          14 input_ids = torch.tensor(ids)
          15 # This line will fail.
     ---> 16 model(input_ids)
```

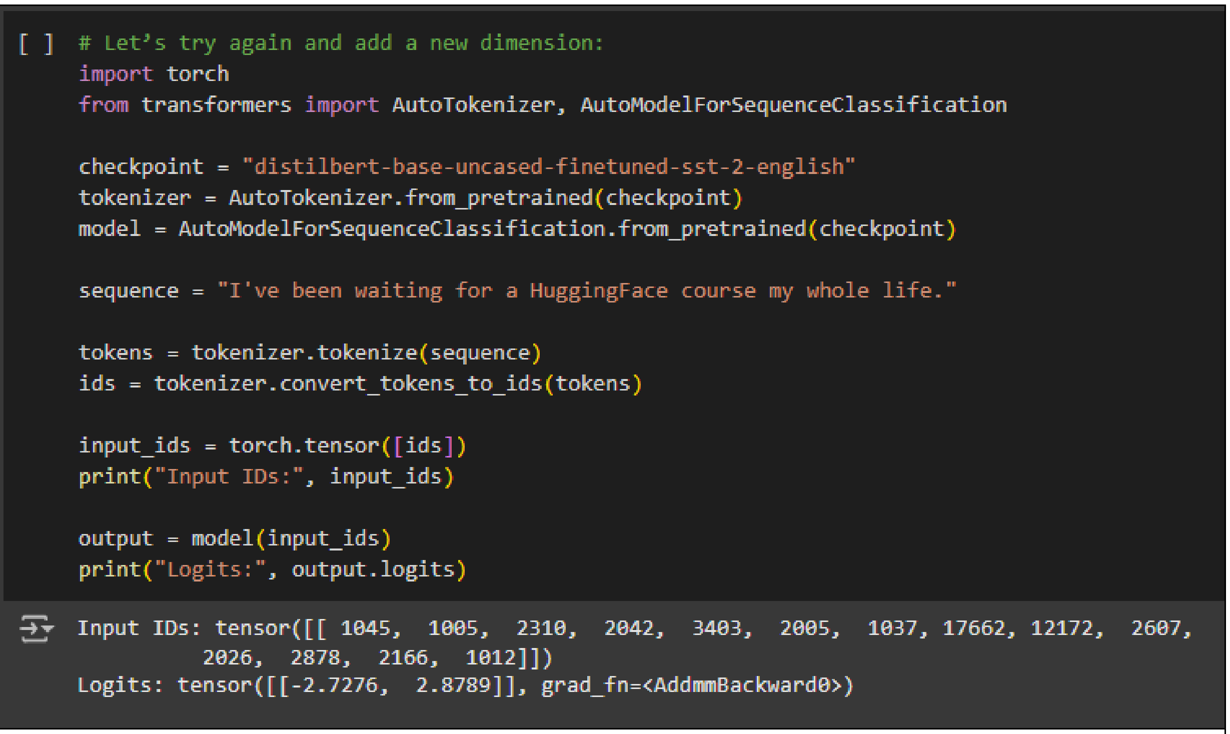
Model gagal memproses input karena input yang diberikan adalah **sebuah urutan tunggal** (single sequence), sedangkan model dari 🤗 Transformers secara default mengharapkan **beberapa urutan** (batch of sentences). Tokenizer secara otomatis menambahkan dimensi tambahan untuk memenuhi format input yang diharapkan oleh model, tetapi ketika kita mencoba mengimitasi proses ini secara manual, kita lupa menambahkan dimensi tambahan.



```
tokenized_inputs = tokenizer(sequence, return_tensors="pt")
print(tokenized_inputs["input_ids"])
```

```
tensor([[  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,
          2607,  2026,  2878,  2166,  1012,   102]])
```

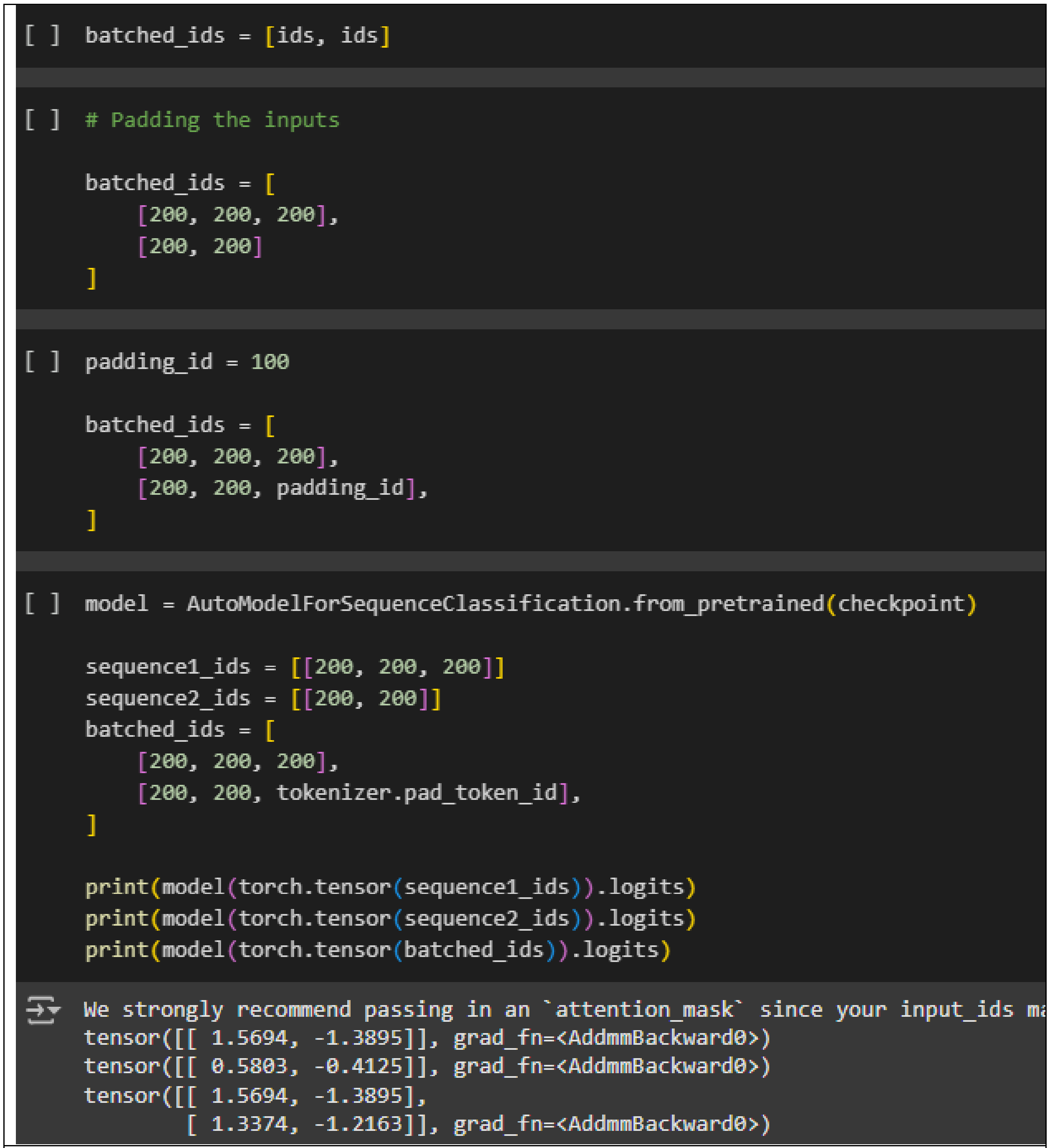
Kode berikut ini menggunakan tokenizer untuk secara otomatis memproses teks dan menghasilkan ID input dalam bentuk tensor PyTorch (return_tensors="pt"), termasuk menambahkan dimensi batch yang diperlukan.



```
# Let’s try again and add a new dimension:
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification

checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(checkpoint)

sequence = "I've been waiting for a HuggingFace course my whole life."

tokens = tokenizer.tokenize(sequence)
ids = tokenizer.convert_tokens_to_ids(tokens)

input_ids = torch.tensor([ids])
print("Input IDs:", input_ids)

output = model(input_ids)
print("Logits:", output.logits)
```

```
Input IDs: tensor([[ 1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,  2607,
          2026,  2878,  2166,  1012]])
Logits: tensor([[-2.7276,  2.8789]], grad_fn=<AddmmBackward0>)
```

Mengonversi urutan teks menjadi ID token dan kemudian mengemasnya ke dalam tensor PyTorch dengan menambahkan dimensi batch (menggunakan torch.tensor([ids])), yang diperlukan oleh model untuk memproses inputnya.



```
[ ]  batched_ids = [ids, ids]
```

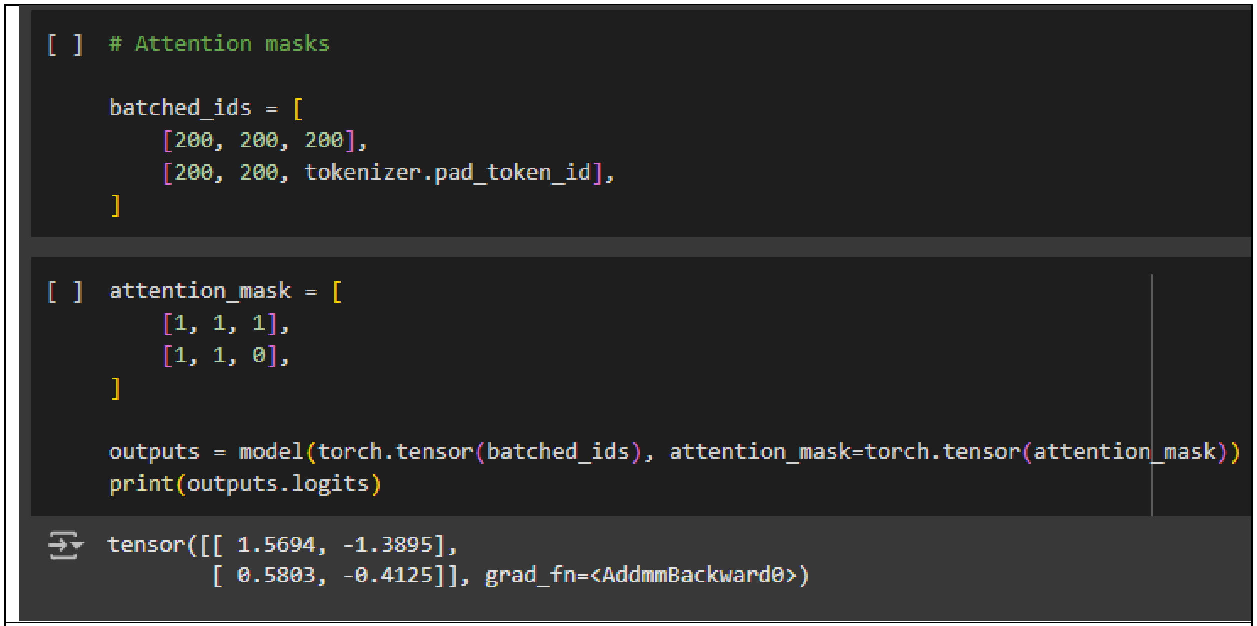
```
[ ]  # Padding the inputs

     batched_ids = [
         [200, 200, 200],
         [200, 200]
     ]
```

```
[ ]  padding_id = 100

     batched_ids = [
         [200, 200, 200],
         [200, 200, padding_id],
     ]
```

```
[ ]  model = AutoModelForSequenceClassification.from_pretrained(checkpoint)

     sequence1_ids = [[200, 200, 200]]
     sequence2_ids = [[200, 200]]
     batched_ids = [
         [200, 200, 200],
         [200, 200, tokenizer.pad_token_id],
     ]

     print(model(torch.tensor(sequence1_ids)).logits)
     print(model(torch.tensor(sequence2_ids)).logits)
     print(model(torch.tensor(batched_ids)).logits)
```

```
We strongly recommend passing in an `attention_mask` since your input_ids ma
tensor([[ 1.5694, -1.3895]], grad_fn=<AddmmBackward0>)
tensor([[ 0.5803, -0.4125]], grad_fn=<AddmmBackward0>)
tensor([[ 1.5694, -1.3895],
        [ 1.3374, -1.2163]], grad_fn=<AddmmBackward0>)
```

Memberikan beberapa jenis input ke model secara berurutan untuk melihat bagaimana model menangani tensor input yang berbeda

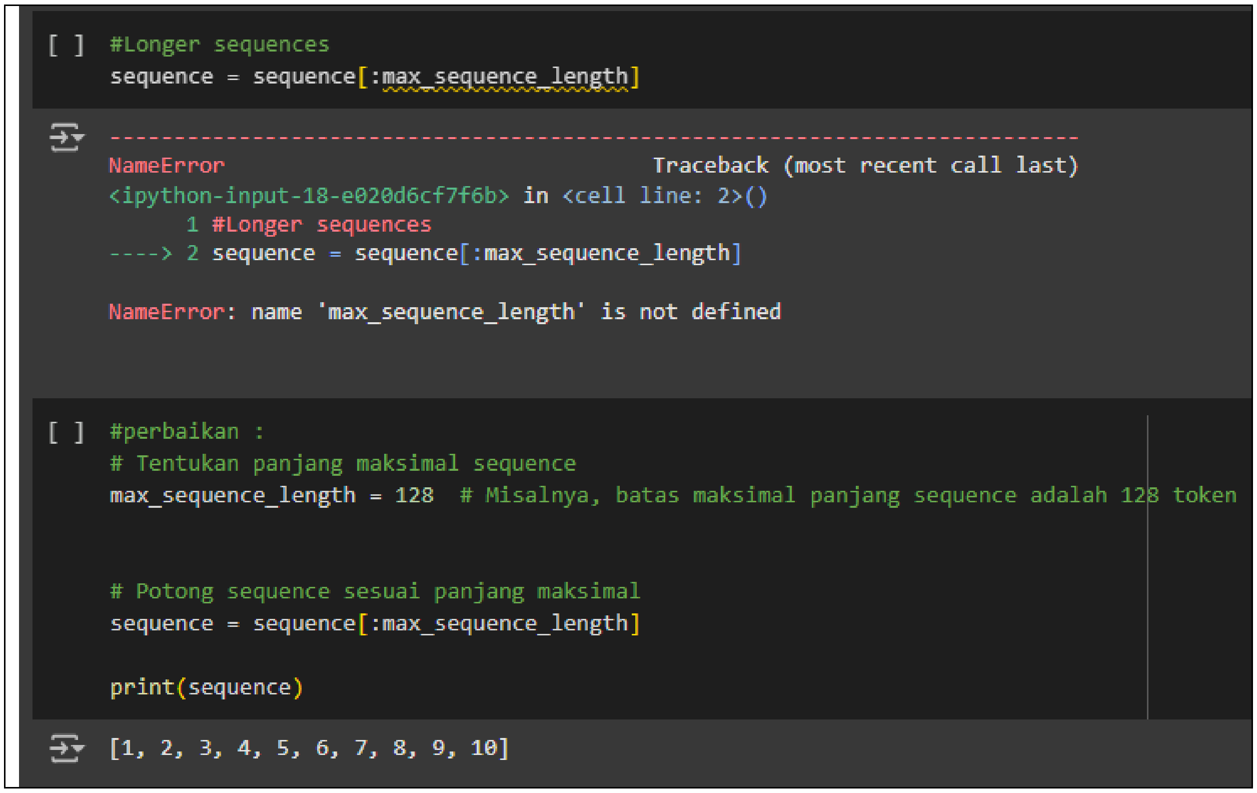


```
[ ] # Attention masks

    batched_ids = [
        [200, 200, 200],
        [200, 200, tokenizer.pad_token_id],
    ]
```

```
[ ] attention_mask = [
        [1, 1, 1],
        [1, 1, 0],
    ]

    outputs = model(torch.tensor(batched_ids), attention_mask=torch.tensor(attention_mask))
    print(outputs.logits)
```

```
tensor([[ 1.5694, -1.3895],
        [ 0.5803, -0.4125]], grad_fn=<AddmmBackward0>)
```

Pada kode ini, attention_mask digunakan untuk memberi tahu model token mana yang harus diperhatikan (1) dan token mana yang harus diabaikan (0), sehingga model hanya memproses token yang relevan dan mengabaikan padding saat menghitung logits.

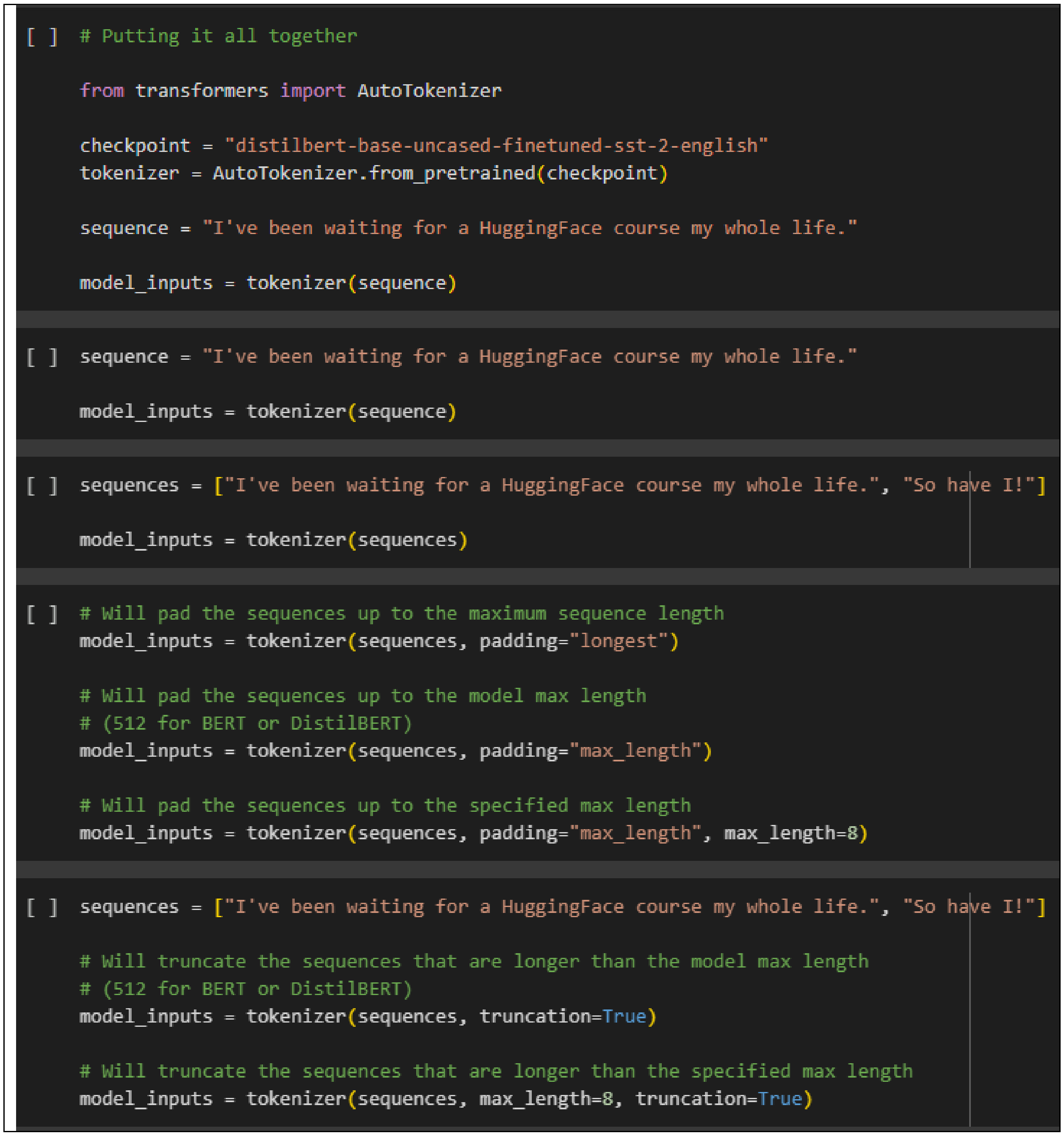


```
[ ] #Longer sequences
    sequence = sequence[:max_sequence_length]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-18-e020d6cf7f6b> in <cell line: 2>()
      1 #Longer sequences
----> 2 sequence = sequence[:max_sequence_length]

NameError: name 'max_sequence_length' is not defined
```

```
[ ] #perbaikan :
    # Tentukan panjang maksimal sequence
    max_sequence_length = 128  # Misalnya, batas maksimal panjang sequence adalah 128 token


    # Potong sequence sesuai panjang maksimal
    sequence = sequence[:max_sequence_length]

    print(sequence)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```



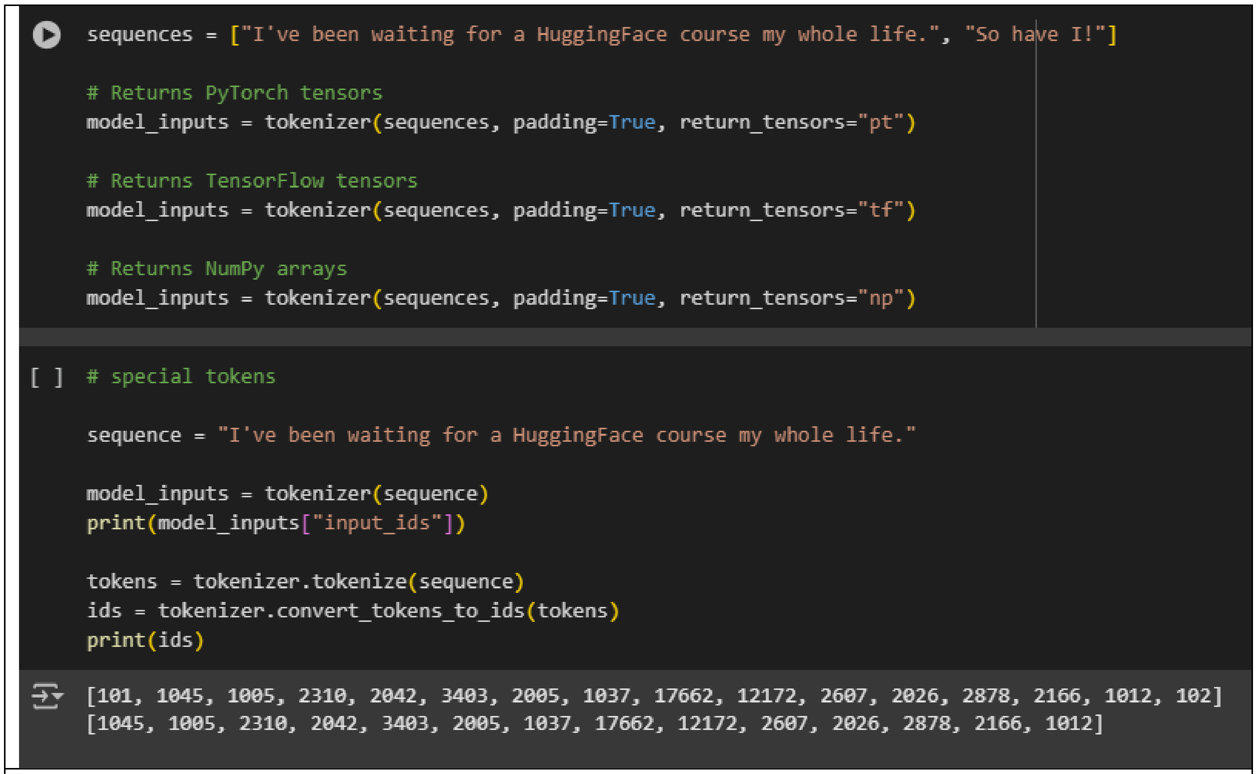
```
[ ]  # Putting it all together

     from transformers import AutoTokenizer

     checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
     tokenizer = AutoTokenizer.from_pretrained(checkpoint)

     sequence = "I've been waiting for a HuggingFace course my whole life."

     model_inputs = tokenizer(sequence)
```

```
[ ]  sequence = "I've been waiting for a HuggingFace course my whole life."

     model_inputs = tokenizer(sequence)
```

```
[ ]  sequences = ["I've been waiting for a HuggingFace course my whole life.", "So have I!"]

     model_inputs = tokenizer(sequences)
```

```
[ ]  # Will pad the sequences up to the maximum sequence length
     model_inputs = tokenizer(sequences, padding="longest")

     # Will pad the sequences up to the model max length
     # (512 for BERT or DistilBERT)
     model_inputs = tokenizer(sequences, padding="max_length")

     # Will pad the sequences up to the specified max length
     model_inputs = tokenizer(sequences, padding="max_length", max_length=8)
```

```
[ ]  sequences = ["I've been waiting for a HuggingFace course my whole life.", "So have I!"]

     # Will truncate the sequences that are longer than the model max length
     # (512 for BERT or DistilBERT)
     model_inputs = tokenizer(sequences, truncation=True)

     # Will truncate the sequences that are longer than the specified max length
     model_inputs = tokenizer(sequences, max_length=8, truncation=True)
```

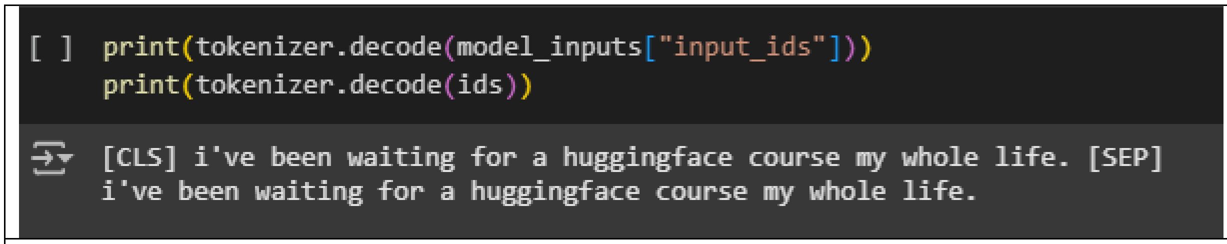


```
sequences = ["I've been waiting for a HuggingFace course my whole life.", "So have I!"]

# Returns PyTorch tensors
model_inputs = tokenizer(sequences, padding=True, return_tensors="pt")

# Returns TensorFlow tensors
model_inputs = tokenizer(sequences, padding=True, return_tensors="tf")

# Returns NumPy arrays
model_inputs = tokenizer(sequences, padding=True, return_tensors="np")
```

```
# special tokens

sequence = "I've been waiting for a HuggingFace course my whole life."

model_inputs = tokenizer(sequence)
print(model_inputs["input_ids"])

tokens = tokenizer.tokenize(sequence)
ids = tokenizer.convert_tokens_to_ids(tokens)
print(ids)
```

```
[101, 1045, 1005, 2310, 2042, 3403, 2005, 1037, 17662, 12172, 2607, 2026, 2878, 2166, 1012, 102]
[1045, 1005, 2310, 2042, 3403, 2005, 1037, 17662, 12172, 2607, 2026, 2878, 2166, 1012]
```

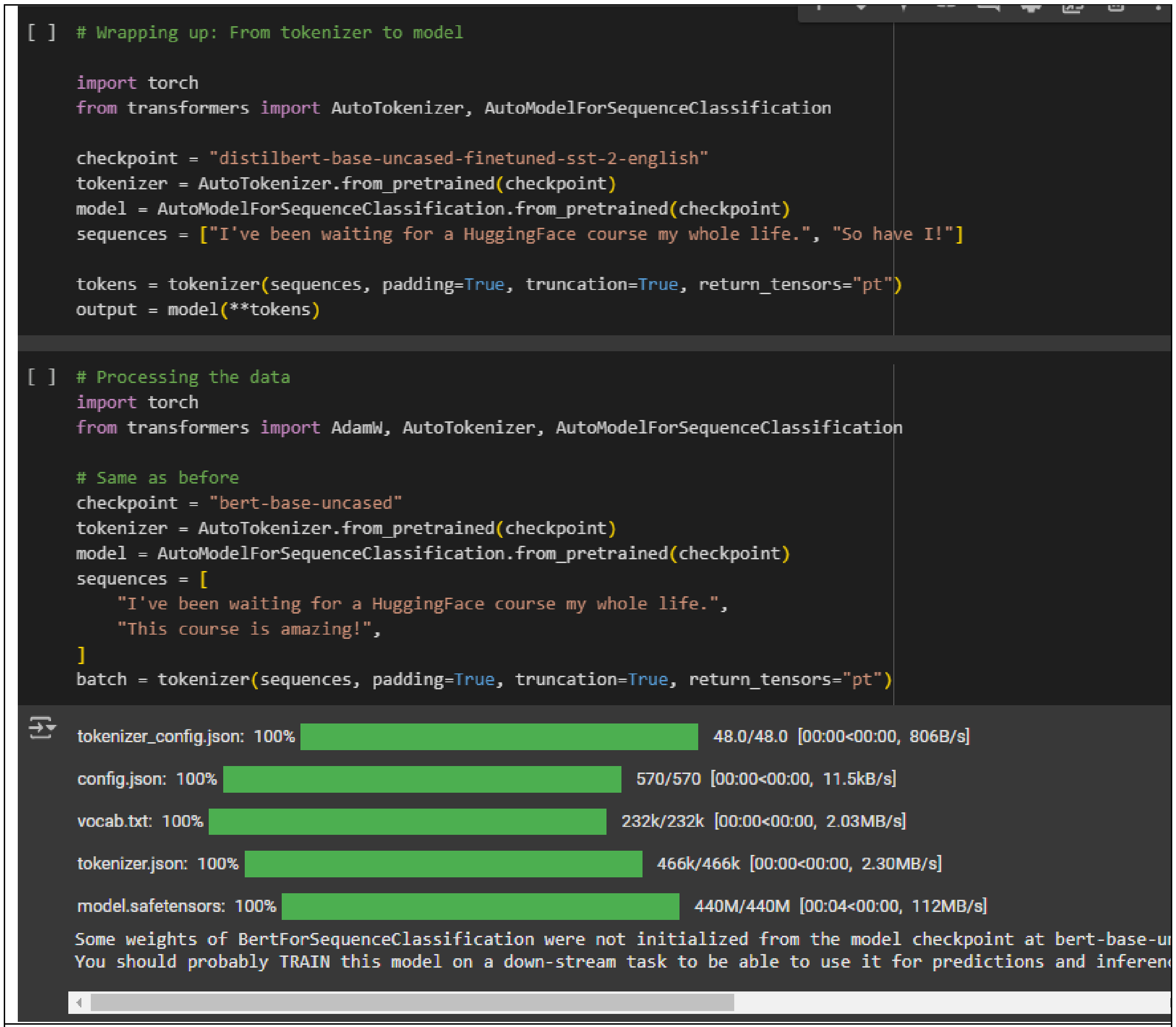
Pada kode ini, tokenizer(sequence) mengubah teks menjadi token dan input ID yang sesuai dengan model, termasuk token khusus seperti [CLS] dan [SEP] jika diperlukan oleh model. Sedangkan tokenizer.tokenize(sequence) memecah teks menjadi token-token individual dan tokenizer.convert_tokens_to_ids(tokens) mengonversi token-token tersebut menjadi ID yang bisa diproses oleh model.



```
print(tokenizer.decode(model_inputs["input_ids"]))
print(tokenizer.decode(ids))
```

```
[CLS] i've been waiting for a huggingface course my whole life. [SEP]
i've been waiting for a huggingface course my whole life.
```

Pada kode ini, tokenizer.decode(model_inputs["input_ids"]) mengonversi ID input yang dihasilkan oleh tokenizer kembali ke bentuk teks asli, termasuk token khusus yang digunakan oleh model. Sedangkan tokenizer.decode(ids) mengonversi ID yang dihasilkan dari tokenisasi manual (tanpa mempertimbangkan token khusus) kembali menjadi teks, yang mungkin tidak mencakup beberapa token yang diperlukan untuk pemrosesan model.

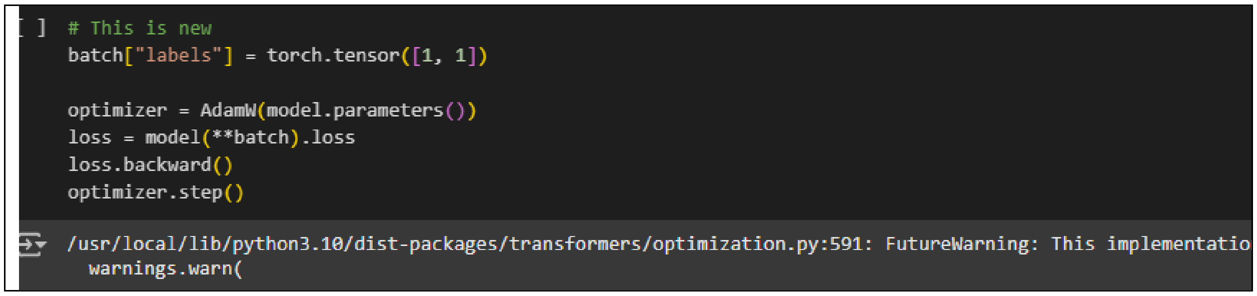


```
[ ] # Wrapping up: From tokenizer to model

    import torch
    from transformers import AutoTokenizer, AutoModelForSequenceClassification

    checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
    tokenizer = AutoTokenizer.from_pretrained(checkpoint)
    model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
    sequences = ["I've been waiting for a HuggingFace course my whole life.", "So have I!"]

    tokens = tokenizer(sequences, padding=True, truncation=True, return_tensors="pt")
    output = model(**tokens)
```

```
[ ] # Processing the data
    import torch
    from transformers import AdamW, AutoTokenizer, AutoModelForSequenceClassification

    # Same as before
    checkpoint = "bert-base-uncased"
    tokenizer = AutoTokenizer.from_pretrained(checkpoint)
    model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
    sequences = [
        "I've been waiting for a HuggingFace course my whole life.",
        "This course is amazing!",
    ]
    batch = tokenizer(sequences, padding=True, truncation=True, return_tensors="pt")
```

```
tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 806B/s]
config.json: 100% 570/570 [00:00<00:00, 11.5kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 2.03MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 2.30MB/s]
model.safetensors: 100% 440M/440M [00:04<00:00, 112MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-u
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inferen
```

Kode ini mempersiapkan data untuk diproses menggunakan model BERT. Dengan menggunakan AutoTokenizer.from_pretrained(checkpoint), tokenisasi dilakukan pada beberapa kalimat dalam sequences, menambahkan padding dan pemotongan agar ukuran input seragam. return_tensors="pt" menghasilkan output dalam format tensor PyTorch, yang siap untuk diberikan ke model BERT (AutoModelForSequenceClassification.from_pretrained(checkpoint)) untuk klasifikasi urutan.



```
[ ] # This is new
    batch["labels"] = torch.tensor([1, 1])

    optimizer = AdamW(model.parameters())
    loss = model(**batch).loss
    loss.backward()
    optimizer.step()
```

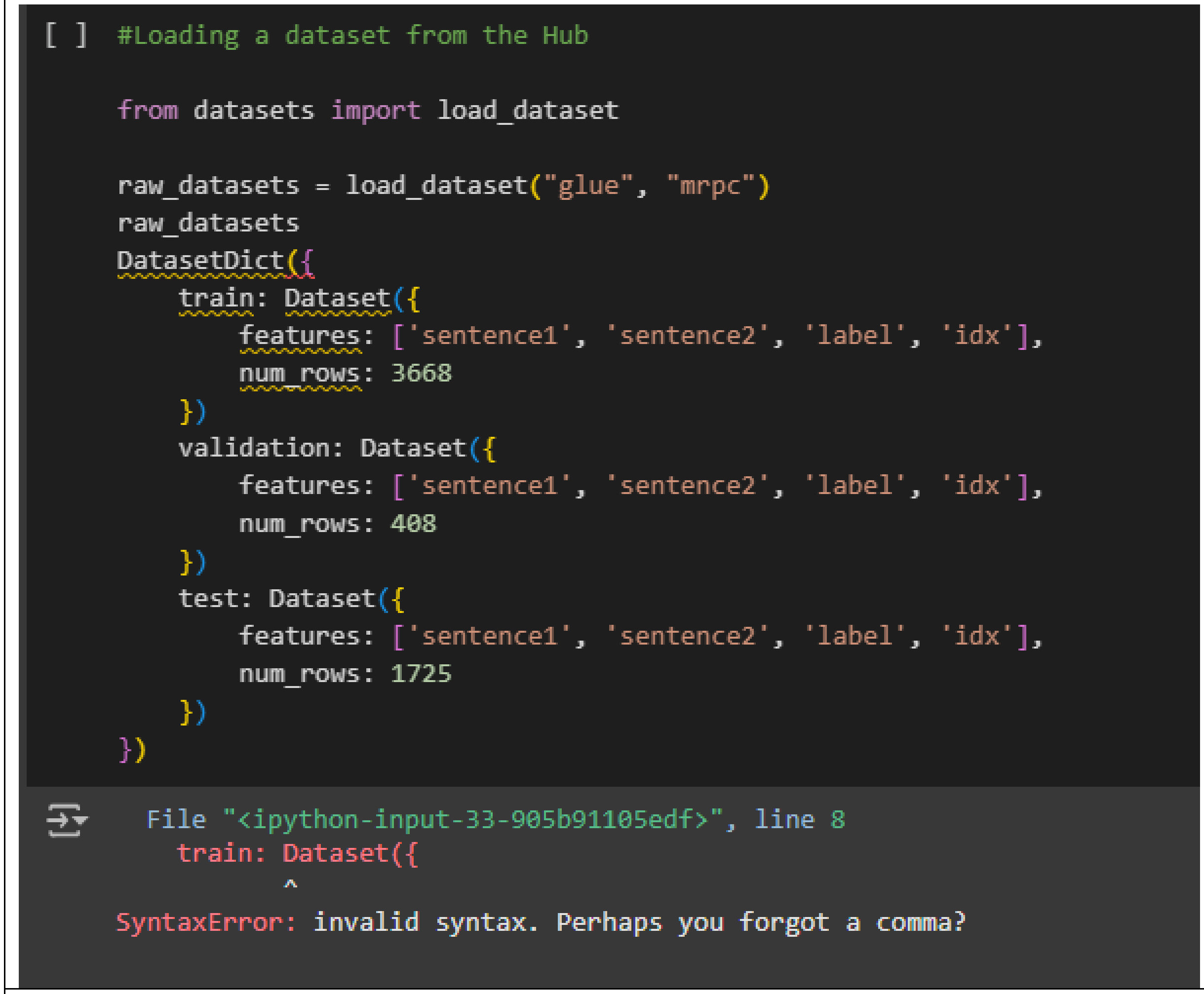
```
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:591: FutureWarning: This implementatio
  warnings.warn(
```

Kode ini menunjukkan bagaimana melakukan pelatihan dengan model BERT. Setelah menambahkan label ke dalam batch (dalam hal ini, dua label yang bernilai 1), optimizer `AdamW` digunakan untuk memperbarui parameter model. Langkah-langkahnya adalah sebagai berikut:

1. Menghitung loss dengan memberikan batch ke model.
2. Melakukan backward pass untuk menghitung gradien (`loss.backward()`).
3. Menggunakan optimizer untuk memperbarui parameter model berdasarkan gradien yang dihitung (`optimizer.step()`).

Proses ini merupakan bagian dari training loop untuk mengoptimalkan model.

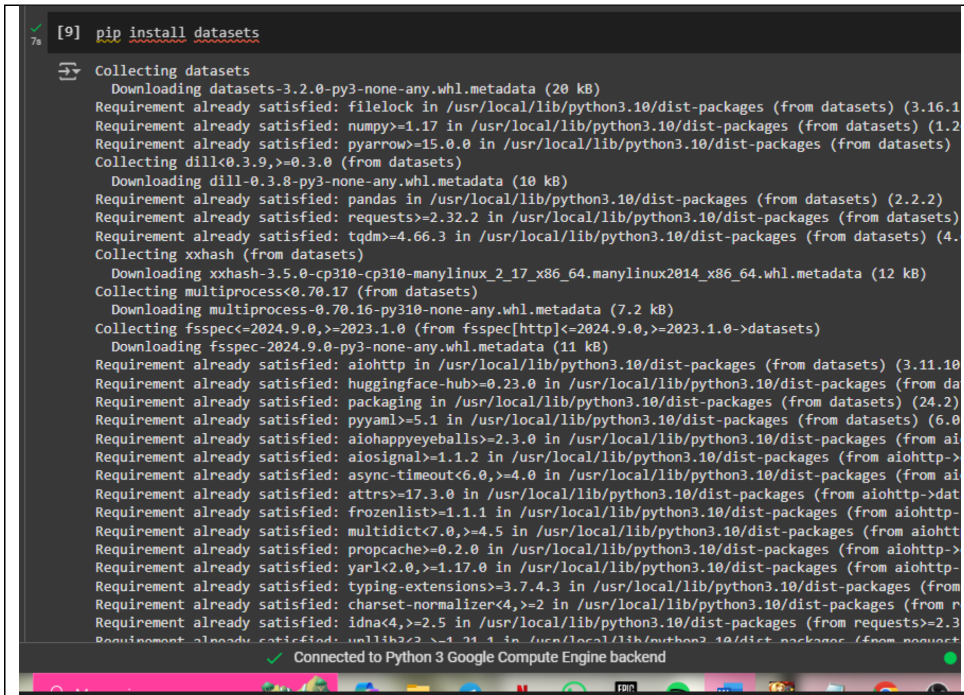
Masalah:



```
[ ]  #Loading a dataset from the Hub

     from datasets import load_dataset

     raw_datasets = load_dataset("glue", "mrpc")
     raw_datasets
     DatasetDict({
         train: Dataset({
             features: ['sentence1', 'sentence2', 'label', 'idx'],
             num_rows: 3668
         })
         validation: Dataset({
             features: ['sentence1', 'sentence2', 'label', 'idx'],
             num_rows: 408
         })
         test: Dataset({
             features: ['sentence1', 'sentence2', 'label', 'idx'],
             num_rows: 1725
         })
     })
```

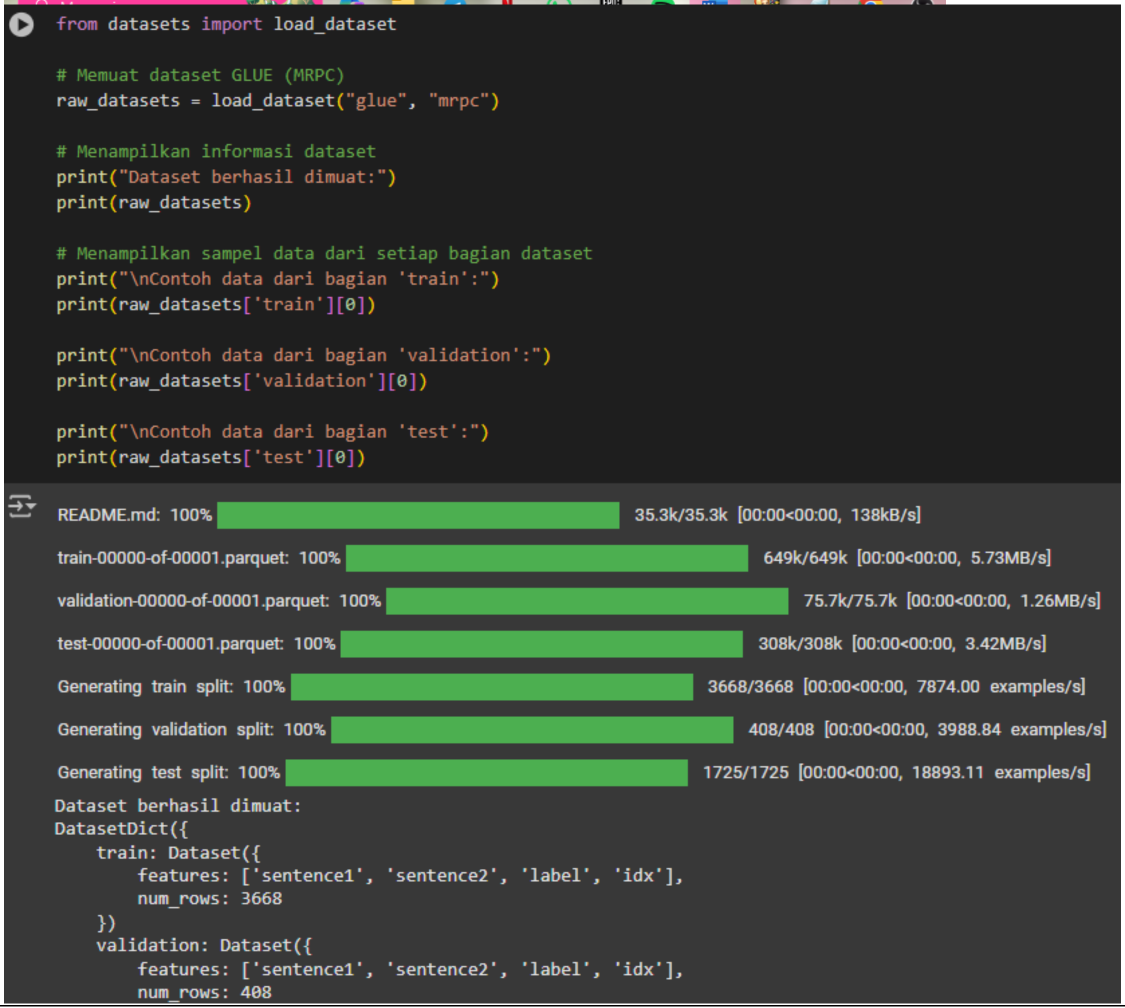
```
  File "<ipython-input-33-905b91105edf>", line 8
    train: Dataset({
         ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

Solusi:



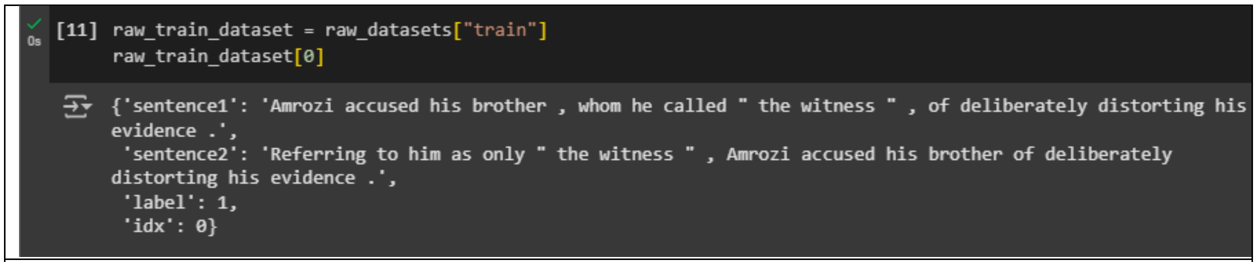
```
[9] pip install datasets

Collecting datasets
  Downloading datasets-3.2.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.2
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets)
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/python3.10/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packages (from datasets)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.9.0,>=2023.1.0 (from fsspec[http]<=2024.9.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.11.10
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from da
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from ai
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from ai
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->dat
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp-
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohtt
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp-
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from r
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from request
```

Connected to Python 3 Google Compute Engine backend



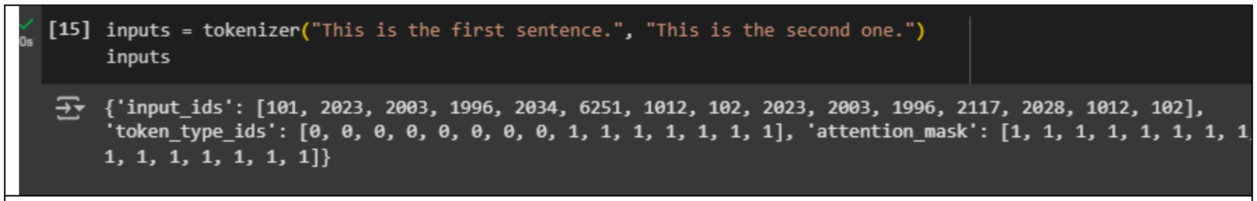
```python
from datasets import load_dataset

# Memuat dataset GLUE (MRPC)
raw_datasets = load_dataset("glue", "mrpc")

# Menampilkan informasi dataset
print("Dataset berhasil dimuat:")
print(raw_datasets)

# Menampilkan sampel data dari setiap bagian dataset
print("\nContoh data dari bagian 'train':")
print(raw_datasets['train'][0])

print("\nContoh data dari bagian 'validation':")
print(raw_datasets['validation'][0])

print("\nContoh data dari bagian 'test':")
print(raw_datasets['test'][0])
```

```
README.md: 100% 35.3k/35.3k [00:00<00:00, 138kB/s]
train-00000-of-00001.parquet: 100% 649k/649k [00:00<00:00, 5.73MB/s]
validation-00000-of-00001.parquet: 100% 75.7k/75.7k [00:00<00:00, 1.26MB/s]
test-00000-of-00001.parquet: 100% 308k/308k [00:00<00:00, 3.42MB/s]
Generating train split: 100% 3668/3668 [00:00<00:00, 7874.00 examples/s]
Generating validation split: 100% 408/408 [00:00<00:00, 3988.84 examples/s]
Generating test split: 100% 1725/1725 [00:00<00:00, 18893.11 examples/s]
Dataset berhasil dimuat:
DatasetDict({
    train: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 3668
    })
    validation: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 408
```



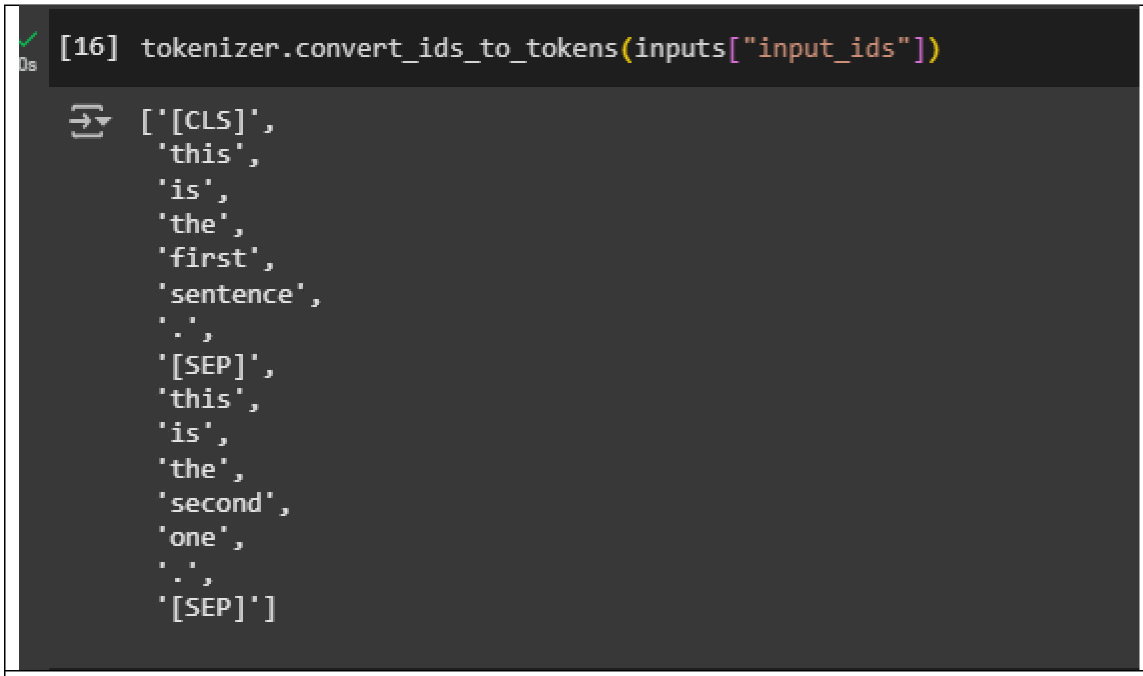
```
[11] raw_train_dataset = raw_datasets["train"]
     raw_train_dataset[0]

{'sentence1': 'Amrozi accused his brother , whom he called " the witness " , of deliberately distorting his
evidence .',
 'sentence2': 'Referring to him as only " the witness " , Amrozi accused his brother of deliberately
distorting his evidence .',
 'label': 1,
 'idx': 0}
```

Kode tersebut mengakses dataset pelatihan (train dataset) dari sebuah objek raw_datasets dan menampilkan data pertama pada dataset tersebut. raw_datasets["train"] adalah bagian dataset yang berisi data pelatihan, dan raw_train_dataset[0] mengakses item pertama dari dataset pelatihan.
Jika raw_datasets adalah dictionary yang berisi dataset terstruktur, maka raw_train_dataset[0] kemungkinan akan mengembalikan data yang berisi teks dan label atau fitur lainnya yang ada dalam dataset pelatihan tersebut.



```
[15] inputs = tokenizer("This is the first sentence.", "This is the second one.")
     inputs

{'input_ids': [101, 2023, 2003, 1996, 2034, 6251, 1012, 102, 2023, 2003, 1996, 2117, 2028, 1012, 102],
'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1]}
```

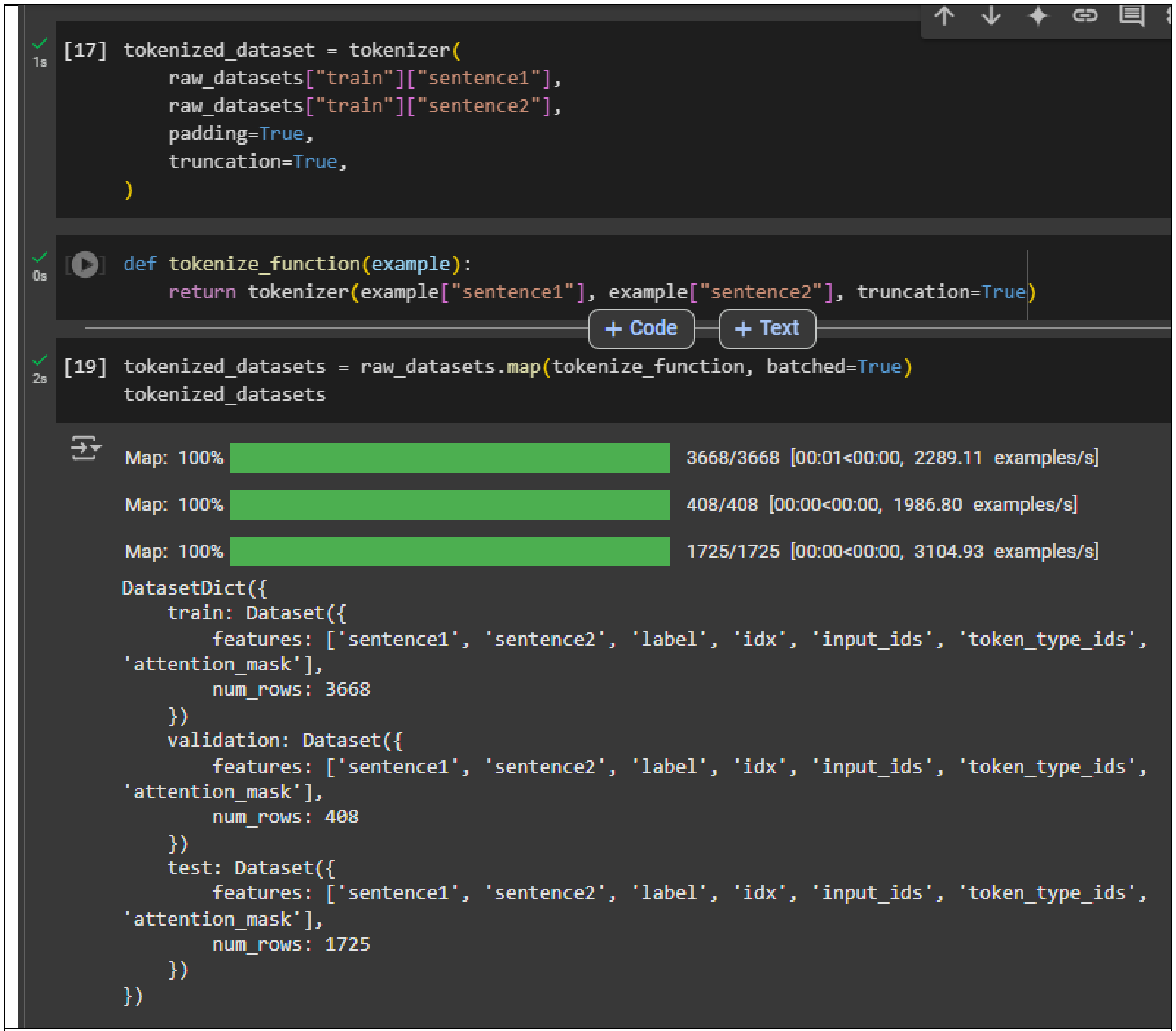
Kode tersebut menggunakan tokenizer untuk mengonversi dua kalimat ("This is the first sentence." dan "This is the second one.") menjadi format yang dapat diproses oleh model.



```
[16] tokenizer.convert_ids_to_tokens(inputs["input_ids"])

['[CLS]',
 'this',
 'is',
 'the',
 'first',
 'sentence',
 '.',
 '[SEP]',
 'this',
 'is',
 'the',
 'second',
 'one',
 '.',
 '[SEP]']
```

Fungsi tokenizer.convert_ids_to_tokens(inputs["input_ids"]) digunakan untuk mengonversi ID token kembali ke kata-kata atau token yang dapat dibaca manusia. inputs["input_ids"] berisi

ID token yang telah dihasilkan sebelumnya, dan fungsi ini akan mengembalikan daftar kata atau token yang sesuai dengan ID tersebut.



```
[17] tokenized_dataset = tokenizer(
         raw_datasets["train"]["sentence1"],
         raw_datasets["train"]["sentence2"],
         padding=True,
         truncation=True,
     )
```

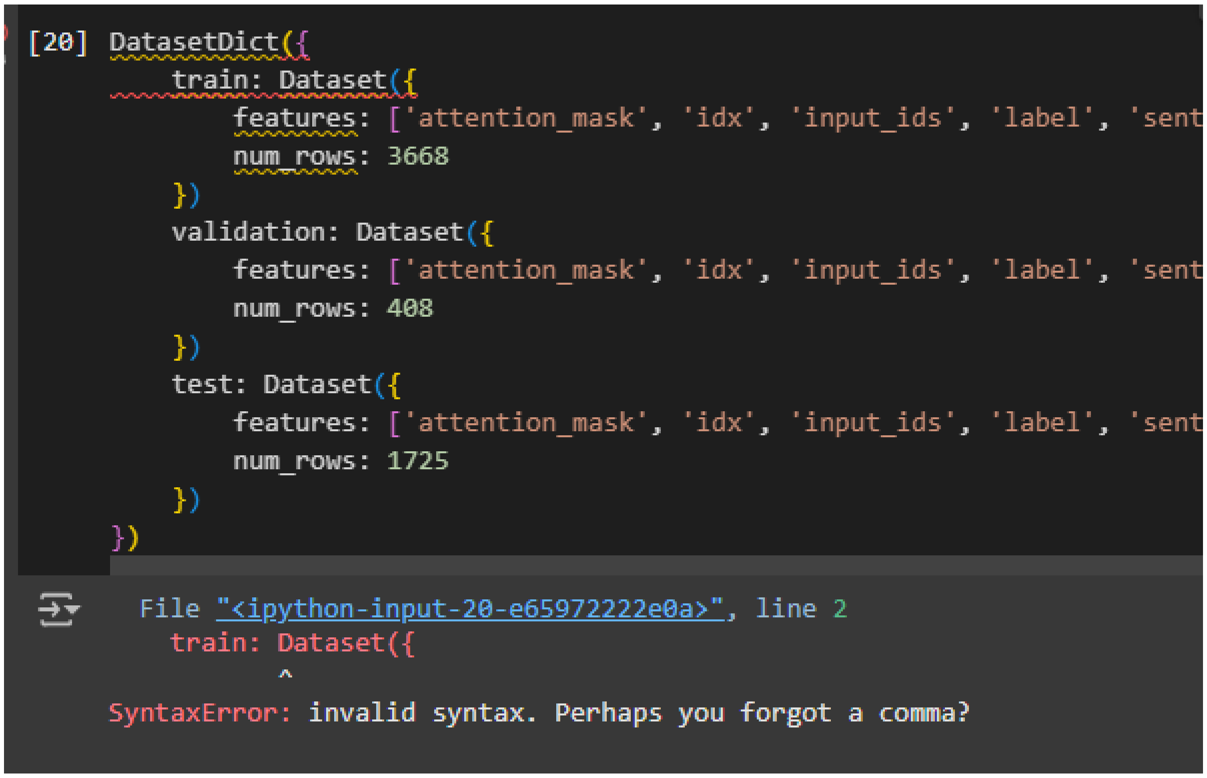
```
def tokenize_function(example):
    return tokenizer(example["sentence1"], example["sentence2"], truncation=True)
```

```
[19] tokenized_datasets = raw_datasets.map(tokenize_function, batched=True)
     tokenized_datasets
```

```
Map: 100%  3668/3668 [00:01<00:00, 2289.11 examples/s]
Map: 100%  408/408 [00:00<00:00, 1986.80 examples/s]
Map: 100%  1725/1725 [00:00<00:00, 3104.93 examples/s]
DatasetDict({
    train: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx', 'input_ids', 'token_type_ids',
'attention_mask'],
        num_rows: 3668
    })
    validation: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx', 'input_ids', 'token_type_ids',
'attention_mask'],
        num_rows: 408
    })
    test: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx', 'input_ids', 'token_type_ids',
'attention_mask'],
        num_rows: 1725
    })
})
```

tokenized_datasets adalah hasil dari pemetaan fungsi tokenisasi pada dataset mentah (raw_datasets) untuk mengubah data teks menjadi format yang dapat diproses oleh model pembelajaran mesin.

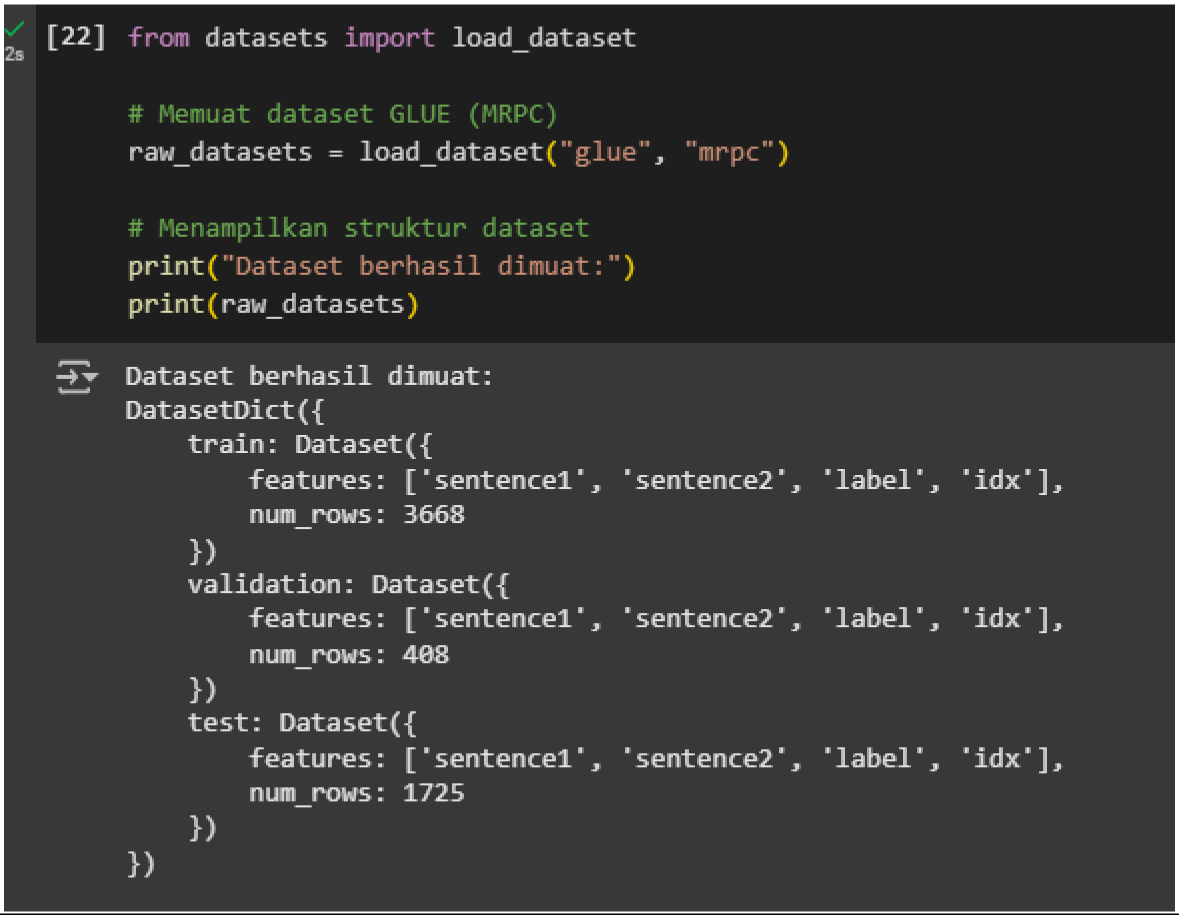
Kendala:



```
[20] DatasetDict({
        train: Dataset({
            features: ['attention_mask', 'idx', 'input_ids', 'label', 'sent
            num_rows: 3668
        })
        validation: Dataset({
            features: ['attention_mask', 'idx', 'input_ids', 'label', 'sent
            num_rows: 408
        })
        test: Dataset({
            features: ['attention_mask', 'idx', 'input_ids', 'label', 'sent
            num_rows: 1725
        })
    })

  File "<ipython-input-20-e65972222e0a>", line 2
    train: Dataset({
         ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

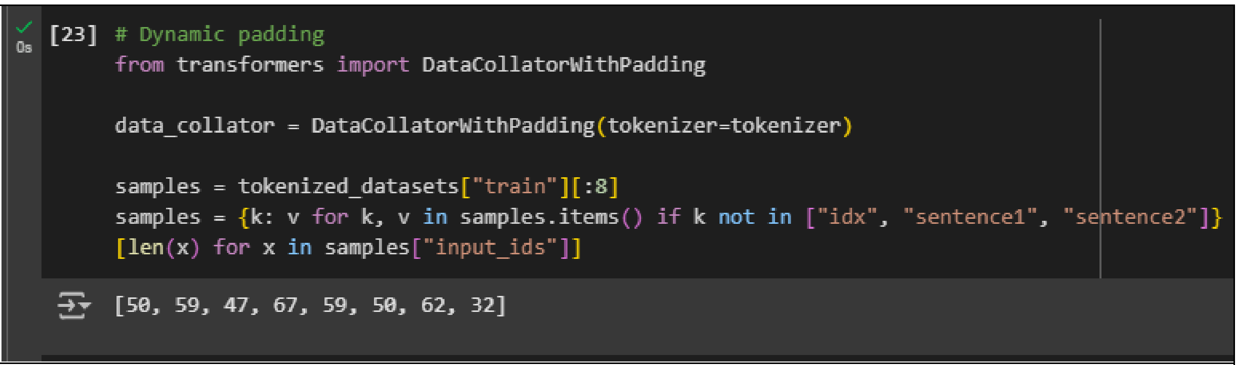
Error tersebut terjadi karena format DatasetDict yang Anda masukkan ke interpreter Python tidak valid secara sintaksis. Dalam Python, dictionary didefinisikan menggunakan pasangan kunci dan nilai yang dipisahkan oleh tanda titik dua (:), dan setiap pasangan dipisahkan oleh tanda koma (,). Anda juga tidak perlu menuliskan data dalam format literal Python seperti output log.

Solusi:



```
[22] from datasets import load_dataset

     # Memuat dataset GLUE (MRPC)
     raw_datasets = load_dataset("glue", "mrpc")

     # Menampilkan struktur dataset
     print("Dataset berhasil dimuat:")
     print(raw_datasets)
```

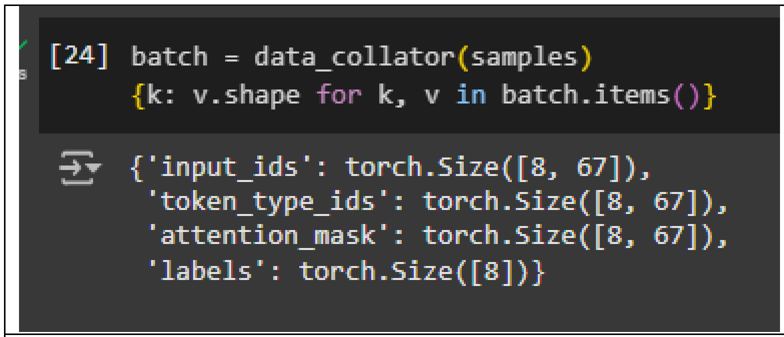
```
Dataset berhasil dimuat:
DatasetDict({
    train: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 3668
    })
    validation: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 408
    })
    test: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 1725
    })
})
```



```
[23] # Dynamic padding
     from transformers import DataCollatorWithPadding

     data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

     samples = tokenized_datasets["train"][:8]
     samples = {k: v for k, v in samples.items() if k not in ["idx", "sentence1", "sentence2"]}
     [len(x) for x in samples["input_ids"]]
```

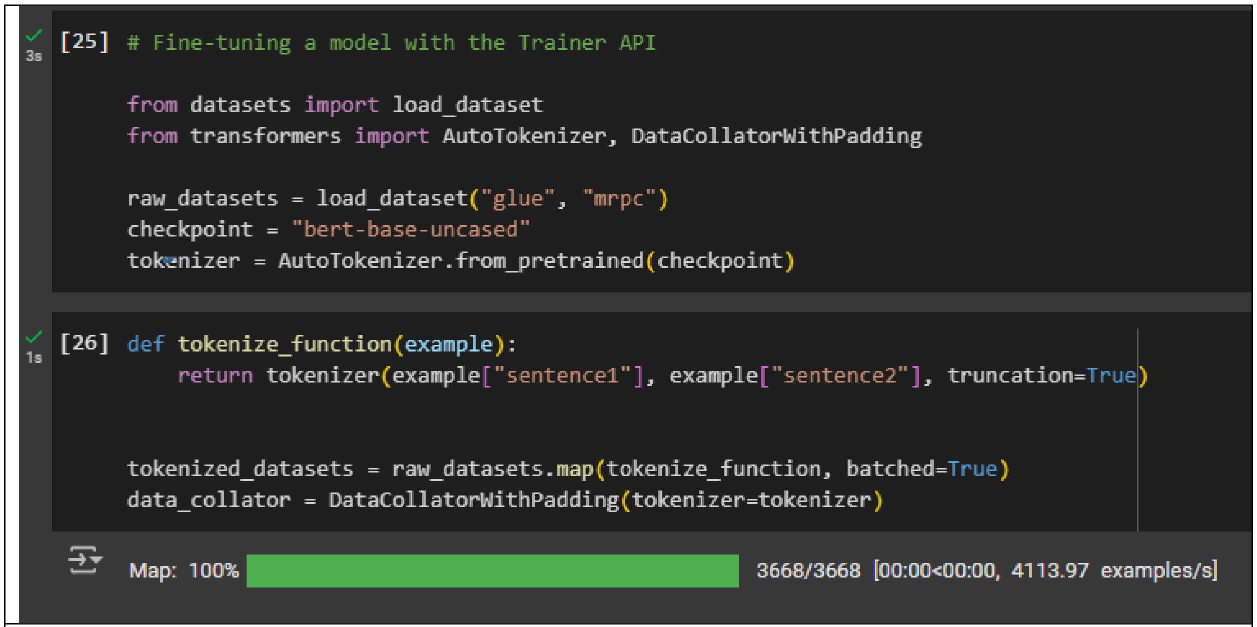
```
[50, 59, 47, 67, 59, 50, 62, 32]
```

Kode tersebut menggunakan DataCollatorWithPadding untuk menambahkan padding secara dinamis pada tokenisasi data, memastikan bahwa setiap sampel dalam batch memiliki panjang yang konsisten



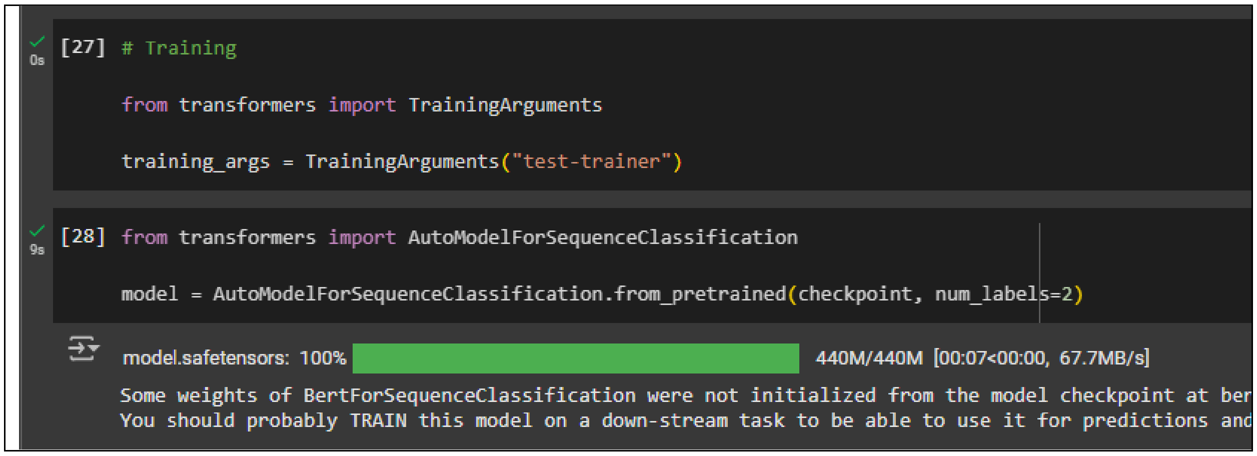
```
[24] batch = data_collator(samples)
     {k: v.shape for k, v in batch.items()}

{'input_ids': torch.Size([8, 67]),
 'token_type_ids': torch.Size([8, 67]),
 'attention_mask': torch.Size([8, 67]),
 'labels': torch.Size([8])}
```

Kode tersebut menggunakan data_collator untuk menggabungkan sampel menjadi satu batch dan kemudian memeriksa dimensi dari setiap item dalam batch tersebut.



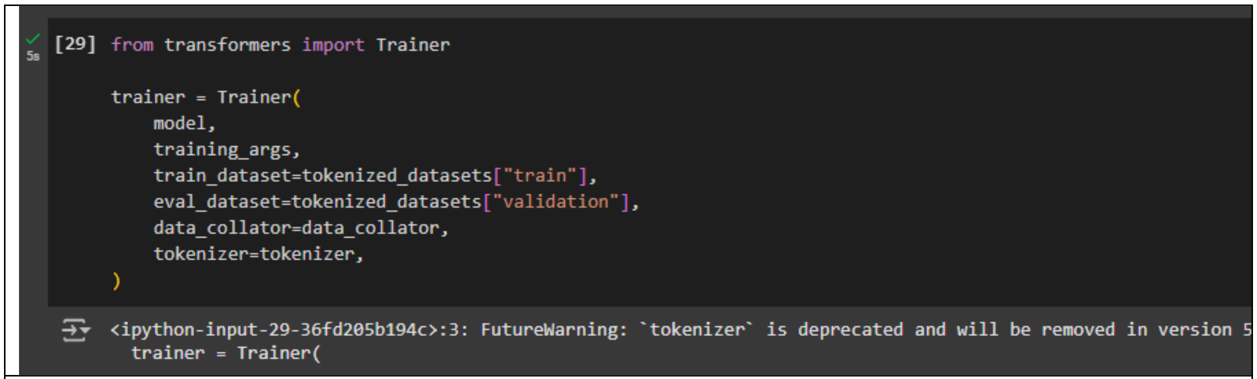
```
[25] # Fine-tuning a model with the Trainer API

     from datasets import load_dataset
     from transformers import AutoTokenizer, DataCollatorWithPadding

     raw_datasets = load_dataset("glue", "mrpc")
     checkpoint = "bert-base-uncased"
     tokenizer = AutoTokenizer.from_pretrained(checkpoint)

[26] def tokenize_function(example):
         return tokenizer(example["sentence1"], example["sentence2"], truncation=True)


     tokenized_datasets = raw_datasets.map(tokenize_function, batched=True)
     data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

Map: 100% 3668/3668 [00:00<00:00, 4113.97 examples/s]
```

memuat dan menyiapkan dataset GLUE MRPC untuk fine-tuning model menggunakan API Trainer dari Hugging Face.



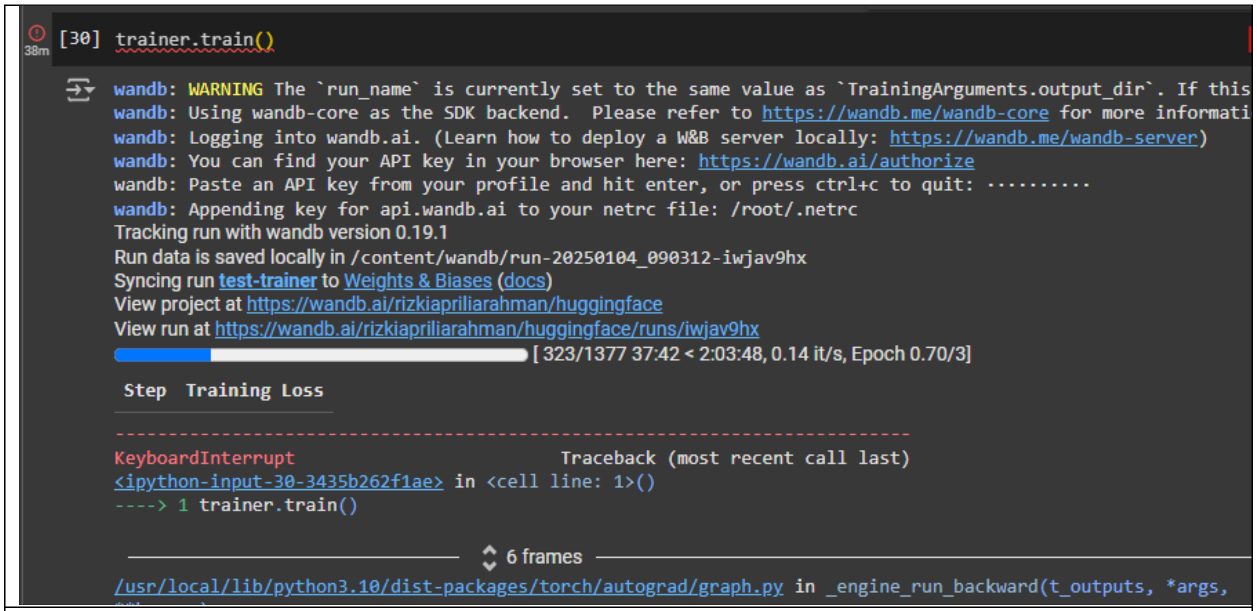
```
[27] # Training

     from transformers import TrainingArguments

     training_args = TrainingArguments("test-trainer")

[28] from transformers import AutoModelForSequenceClassification

     model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)

model.safetensors: 100% 440M/440M [00:07<00:00, 67.7MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at ber
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and
```

berfokus pada inisialisasi argumen pelatihan dan model untuk fine-tuning menggunakan API Trainer dari Hugging Face.



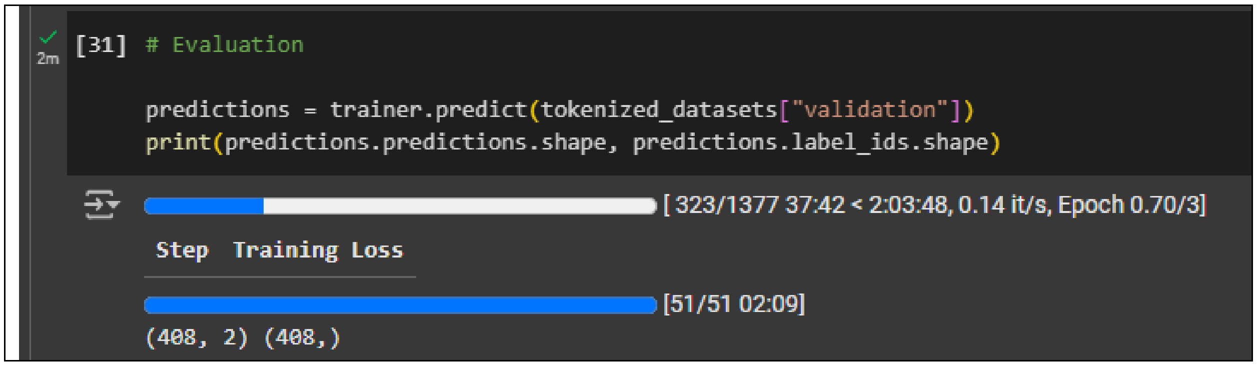
```
[29] from transformers import Trainer

     trainer = Trainer(
         model,
         training_args,
         train_dataset=tokenized_datasets["train"],
         eval_dataset=tokenized_datasets["validation"],
         data_collator=data_collator,
         tokenizer=tokenizer,
     )

<ipython-input-29-36fd205b194c>:3: FutureWarning: `tokenizer` is deprecated and will be removed in version 5
  trainer = Trainer(
```

Kode ini menyusun objek Trainer yang digunakan untuk melatih model dengan dataset yang telah diproses.



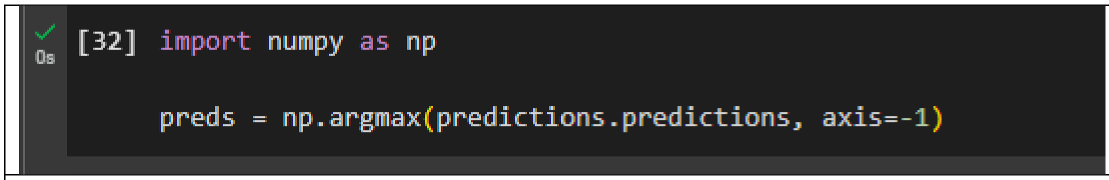
```
[30] trainer.train()

wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more informati
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.19.1
Run data is saved locally in /content/wandb/run-20250104_090312-iwjav9hx
Syncing run test-trainer to Weights & Biases (docs)
View project at https://wandb.ai/rizkiapriliarahman/huggingface
View run at https://wandb.ai/rizkiapriliarahman/huggingface/runs/iwjav9hx
[323/1377 37:42 < 2:03:48, 0.14 it/s, Epoch 0.70/3]

 Step  Training Loss

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-30-3435b262f1ae> in <cell line: 1>()
----> 1 trainer.train()

                              6 frames
/usr/local/lib/python3.10/dist-packages/torch/autograd/graph.py in _engine_run_backward(t_outputs, *args,
```

Kendala: Running sangat lama dan selalu error karena data sangat banyak.



```
[31] # Evaluation

     predictions = trainer.predict(tokenized_datasets["validation"])
     print(predictions.predictions.shape, predictions.label_ids.shape)

[323/1377 37:42 < 2:03:48, 0.14 it/s, Epoch 0.70/3]

 Step  Training Loss

[51/51 02:09]
(408, 2) (408,)
```

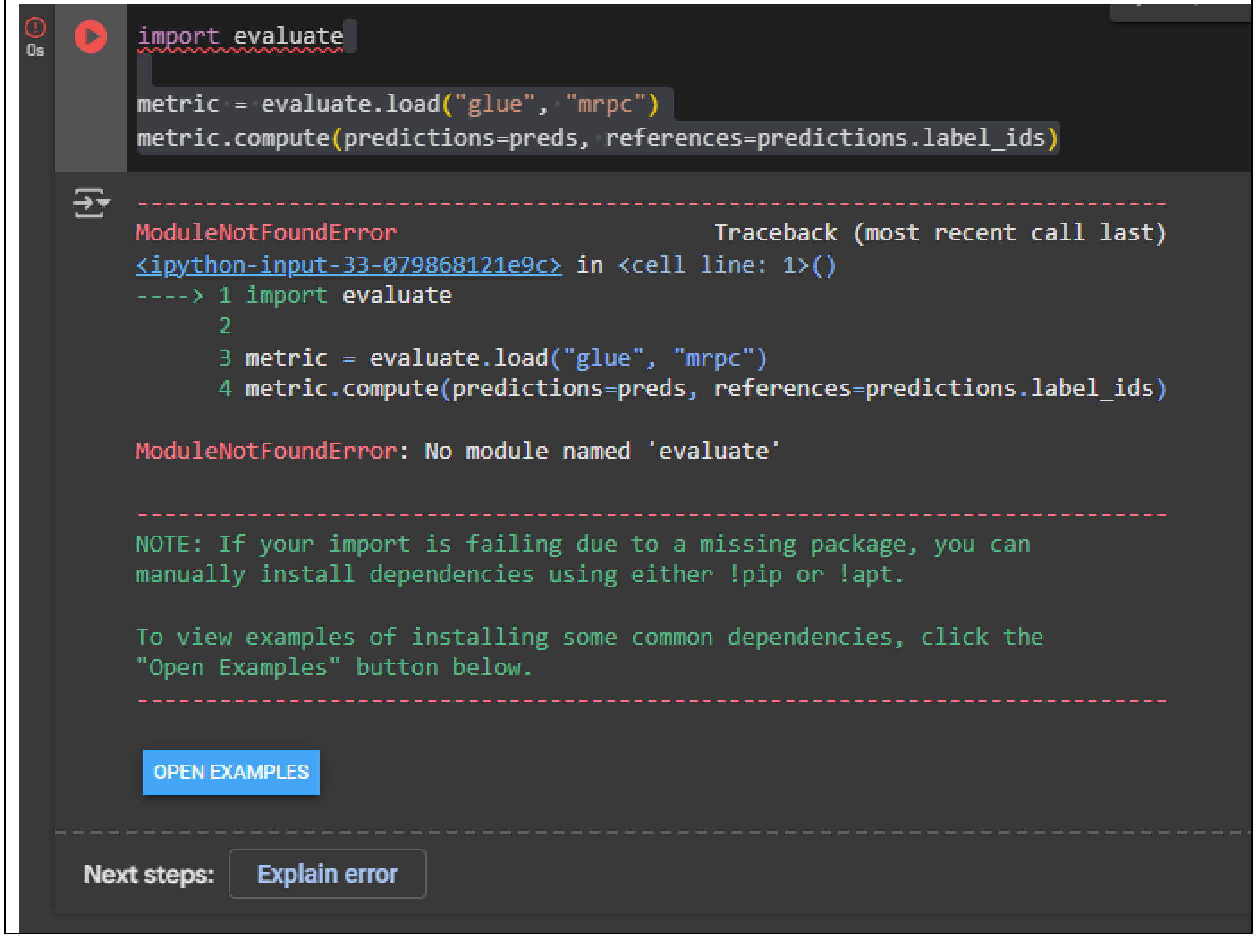
Kode ini digunakan untuk mengevaluasi model yang telah dilatih dengan menggunakan dataset validasi dan melihat hasil prediksi serta label yang terkait.



```
[32] import numpy as np

     preds = np.argmax(predictions.predictions, axis=-1)
```

Kode ini digunakan untuk mengonversi hasil prediksi model yang berupa probabilitas atau skor ke dalam bentuk label kelas yang dipilih (klasifikasi argmax).

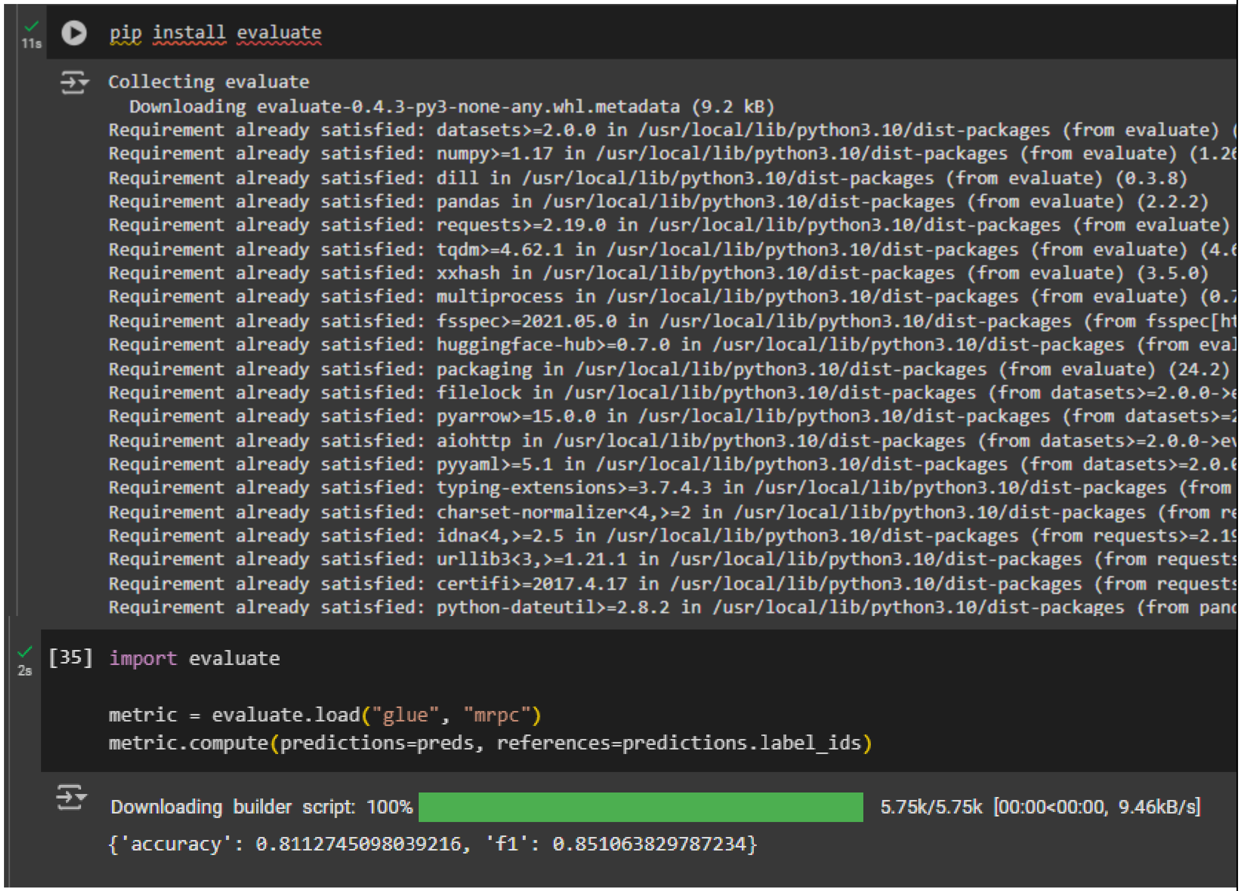
Kendala:



```
import evaluate

metric = evaluate.load("glue", "mrpc")
metric.compute(predictions=preds, references=predictions.label_ids)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-33-079868121e9c> in <cell line: 1>()
----> 1 import evaluate
      2 
      3 metric = evaluate.load("glue", "mrpc")
      4 metric.compute(predictions=preds, references=predictions.label_ids)

ModuleNotFoundError: No module named 'evaluate'

---------------------------------------------------------------------------
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.
---------------------------------------------------------------------------
```

Solusi:



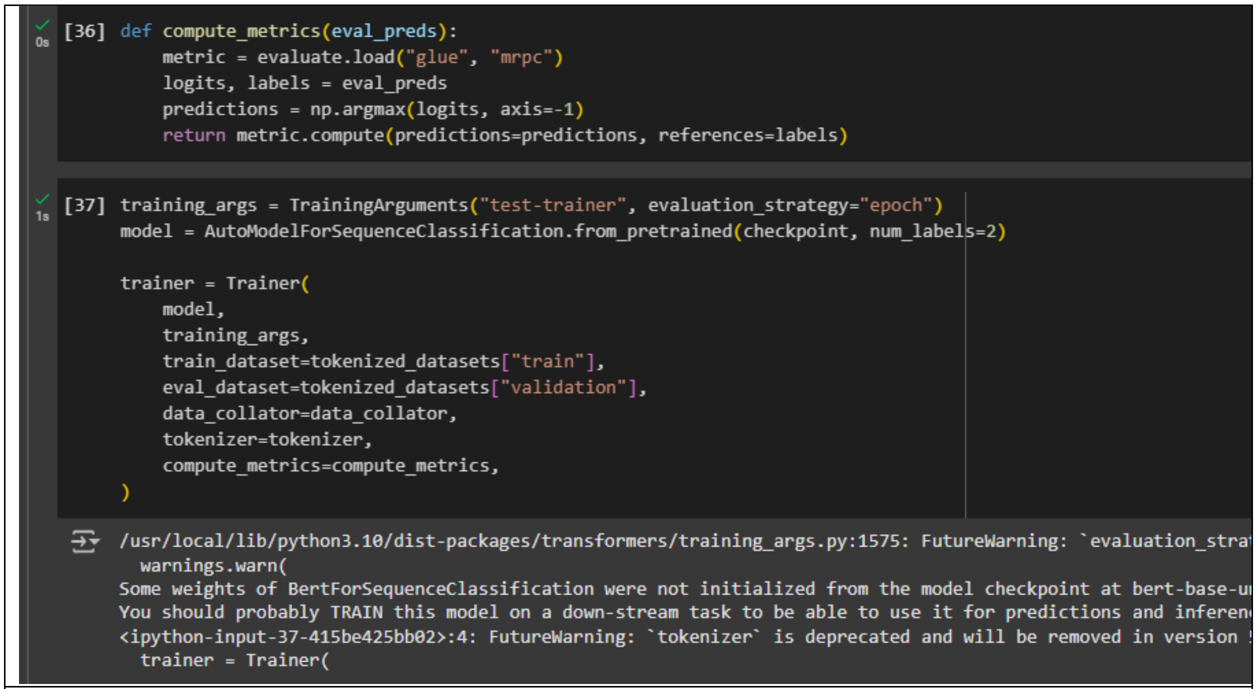
```
pip install evaluate
```

```
Collecting evaluate
  Downloading evaluate-0.4.3-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from evaluate) (1.26
Requirement already satisfied: dill in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from evaluate) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from evaluate)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from evaluate) (4.6
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from evaluate) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.7
Requirement already satisfied: fsspec>=2021.05.0 in /usr/local/lib/python3.10/dist-packages (from fsspec[ht
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from eval
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from evaluate) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->e
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets>=2
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->ev
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from re
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from panc
```

```
[35] import evaluate

     metric = evaluate.load("glue", "mrpc")
     metric.compute(predictions=preds, references=predictions.label_ids)
```

```
Downloading builder script: 100%  5.75k/5.75k [00:00<00:00, 9.46kB/s]
{'accuracy': 0.8112745098039216, 'f1': 0.851063829787234}
```

Kode ini digunakan untuk mengevaluasi kinerja model pada tugas MRPC menggunakan metrik dari GLUE (General Language Understanding Evaluation).

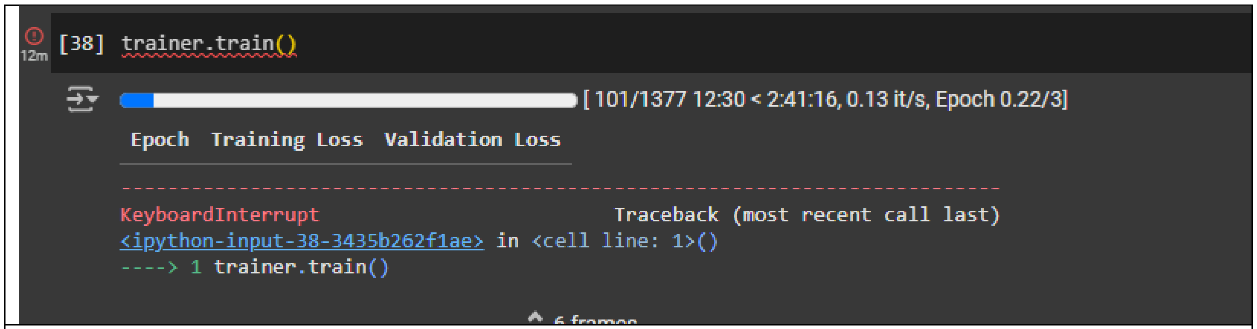


```
[36] def compute_metrics(eval_preds):
         metric = evaluate.load("glue", "mrpc")
         logits, labels = eval_preds
         predictions = np.argmax(logits, axis=-1)
         return metric.compute(predictions=predictions, references=labels)
```

```
[37] training_args = TrainingArguments("test-trainer", evaluation_strategy="epoch")
     model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)

     trainer = Trainer(
         model,
         training_args,
         train_dataset=tokenized_datasets["train"],
         eval_dataset=tokenized_datasets["validation"],
         data_collator=data_collator,
         tokenizer=tokenizer,
         compute_metrics=compute_metrics,
     )
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1575: FutureWarning: `evaluation_stra
  warnings.warn(
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-u
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inferen
<ipython-input-37-415be425bb02>:4: FutureWarning: `tokenizer` is deprecated and will be removed in version
  trainer = Trainer(
```

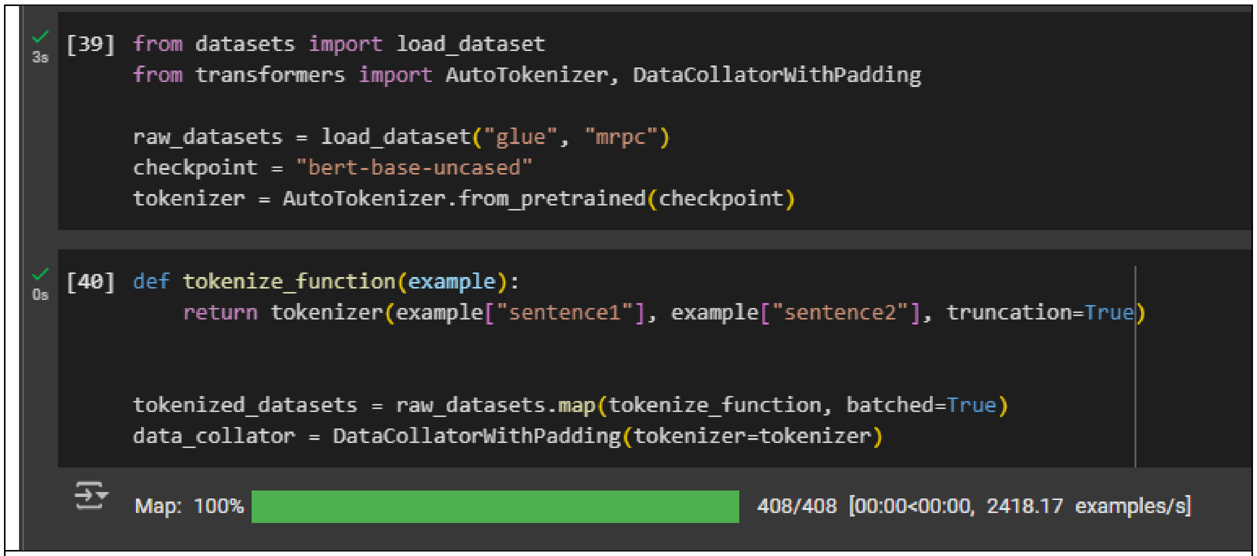
Kode ini menyusun kembali objek Trainer dengan menambahkan beberapa parameter untuk kontrol lebih lanjut atas proses pelatihan dan evaluasi model.



```
[38] trainer.train()
```

```
[101/1377 12:30 < 2:41:16, 0.13 it/s, Epoch 0.22/3]

 Epoch  Training Loss  Validation Loss

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-38-3435b262f1ae> in <cell line: 1>()
----> 1 trainer.train()
```

Kendala: Running sangat lama karena membuat data yang banyak.

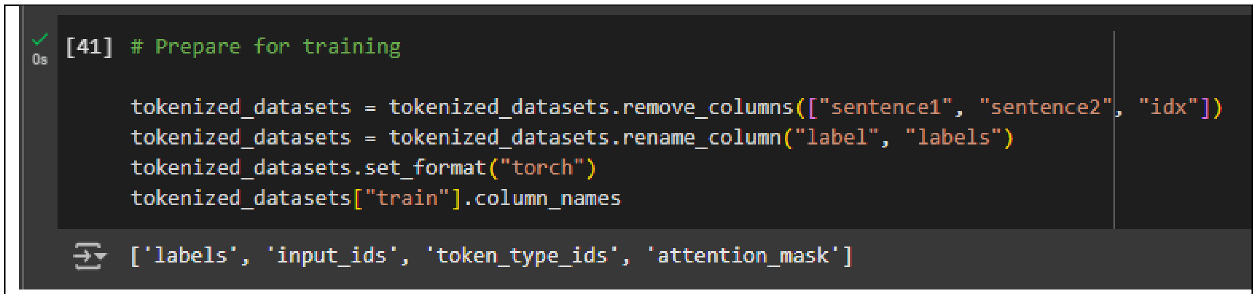


```
[39] from datasets import load_dataset
     from transformers import AutoTokenizer, DataCollatorWithPadding

     raw_datasets = load_dataset("glue", "mrpc")
     checkpoint = "bert-base-uncased"
     tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

```
[40] def tokenize_function(example):
         return tokenizer(example["sentence1"], example["sentence2"], truncation=True)


     tokenized_datasets = raw_datasets.map(tokenize_function, batched=True)
     data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

```
Map: 100%  408/408 [00:00<00:00, 2418.17 examples/s]
```

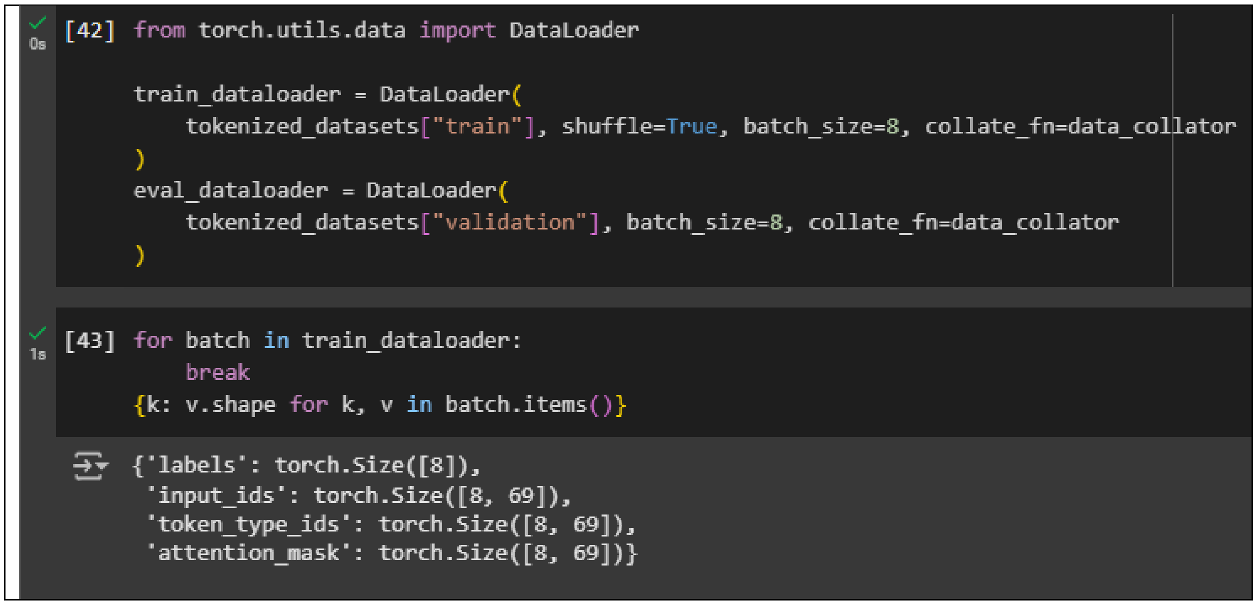
Kode ini digunakan untuk menyiapkan dataset dan tokenizer sebelum melatih model.



```
[41] # Prepare for training

     tokenized_datasets = tokenized_datasets.remove_columns(["sentence1", "sentence2", "idx"])
     tokenized_datasets = tokenized_datasets.rename_column("label", "labels")
     tokenized_datasets.set_format("torch")
     tokenized_datasets["train"].column_names
```

```
['labels', 'input_ids', 'token_type_ids', 'attention_mask']
```

Kode ini mempersiapkan dataset yang telah ditokenisasi agar dapat digunakan dalam pelatihan model.

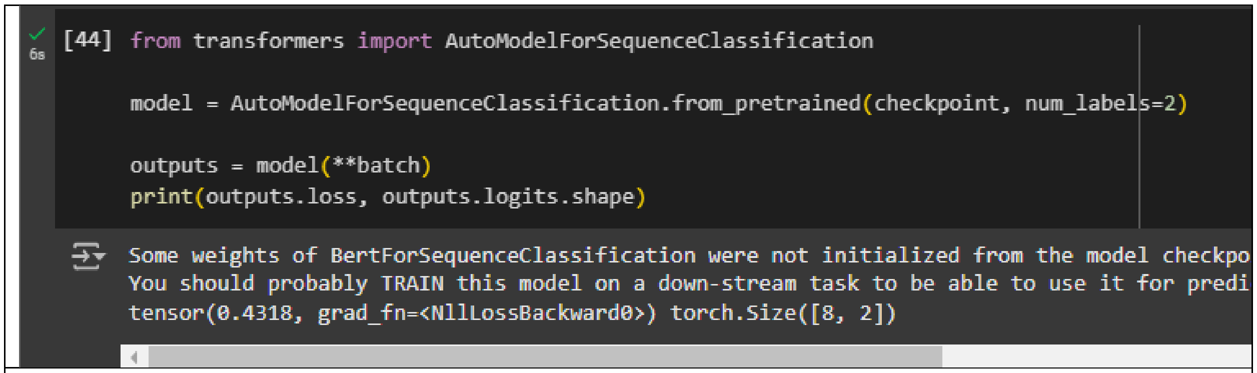


```
[42] from torch.utils.data import DataLoader

     train_dataloader = DataLoader(
         tokenized_datasets["train"], shuffle=True, batch_size=8, collate_fn=data_collator
     )
     eval_dataloader = DataLoader(
         tokenized_datasets["validation"], batch_size=8, collate_fn=data_collator
     )
```

```
[43] for batch in train_dataloader:
         break
     {k: v.shape for k, v in batch.items()}
```

```
{'labels': torch.Size([8]),
 'input_ids': torch.Size([8, 69]),
 'token_type_ids': torch.Size([8, 69]),
 'attention_mask': torch.Size([8, 69])}
```

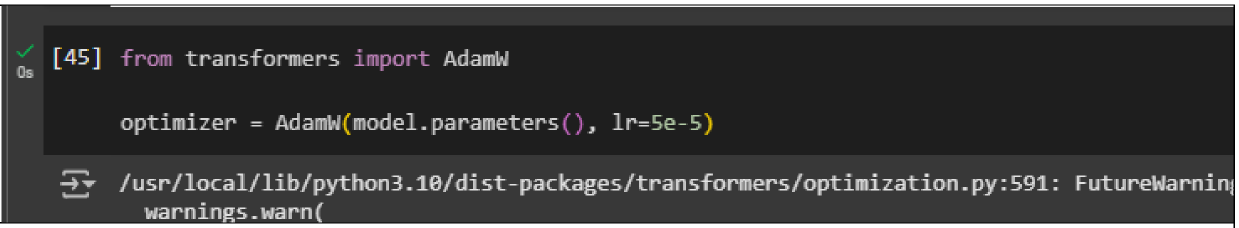
Kode ini digunakan untuk melihat bentuk (shape) dari data dalam batch pertama yang diambil dari train_dataloader.



```
[44] from transformers import AutoModelForSequenceClassification

     model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)

     outputs = model(**batch)
     print(outputs.loss, outputs.logits.shape)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpo
You should probably TRAIN this model on a down-stream task to be able to use it for predi
tensor(0.4318, grad_fn=<NllLossBackward0>) torch.Size([8, 2])
```

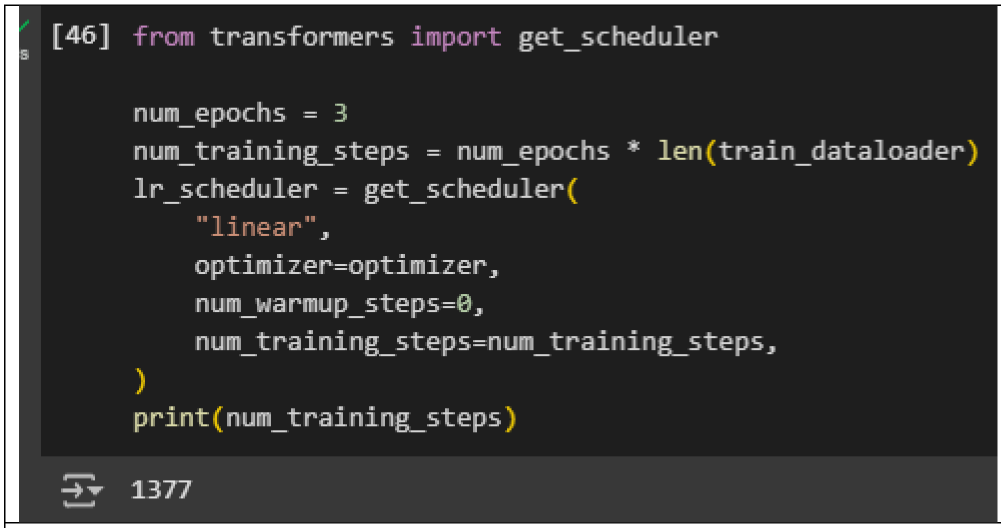
Kode ini digunakan untuk melakukan inferensi atau pelatihan pada model dengan batch data yang telah dipersiapkan.



```
[45] from transformers import AdamW

     optimizer = AdamW(model.parameters(), lr=5e-5)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:591: FutureWarnin
  warnings.warn(
```

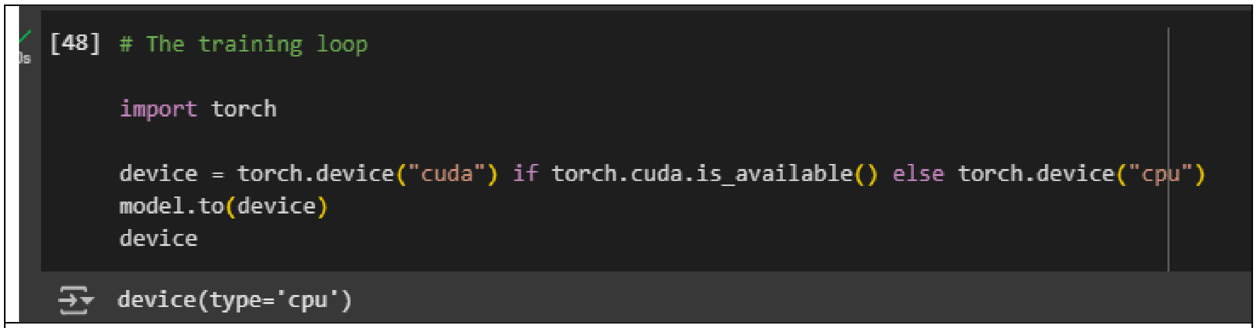
Kode ini digunakan untuk mengonfigurasi optimizer yang akan digunakan selama pelatihan model.



```
[46] from transformers import get_scheduler

     num_epochs = 3
     num_training_steps = num_epochs * len(train_dataloader)
     lr_scheduler = get_scheduler(
         "linear",
         optimizer=optimizer,
         num_warmup_steps=0,
         num_training_steps=num_training_steps,
     )
     print(num_training_steps)
```

```
1377
```

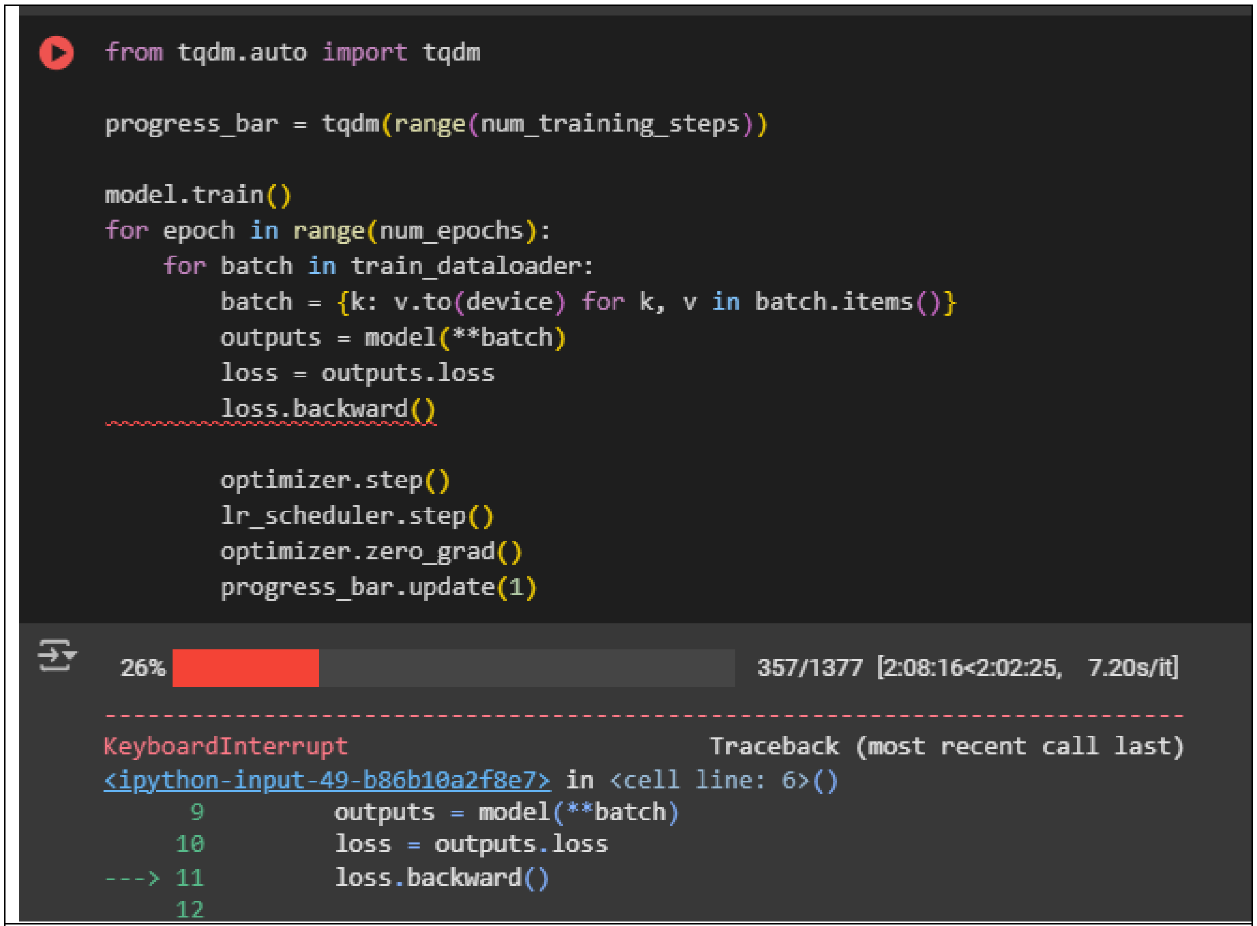
Kode ini digunakan untuk menyiapkan scheduler yang akan mengatur laju pembelajaran (learning rate) selama pelatihan.



```
[48] # The training loop

     import torch

     device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
     model.to(device)
     device
```

```
device(type='cpu')
```

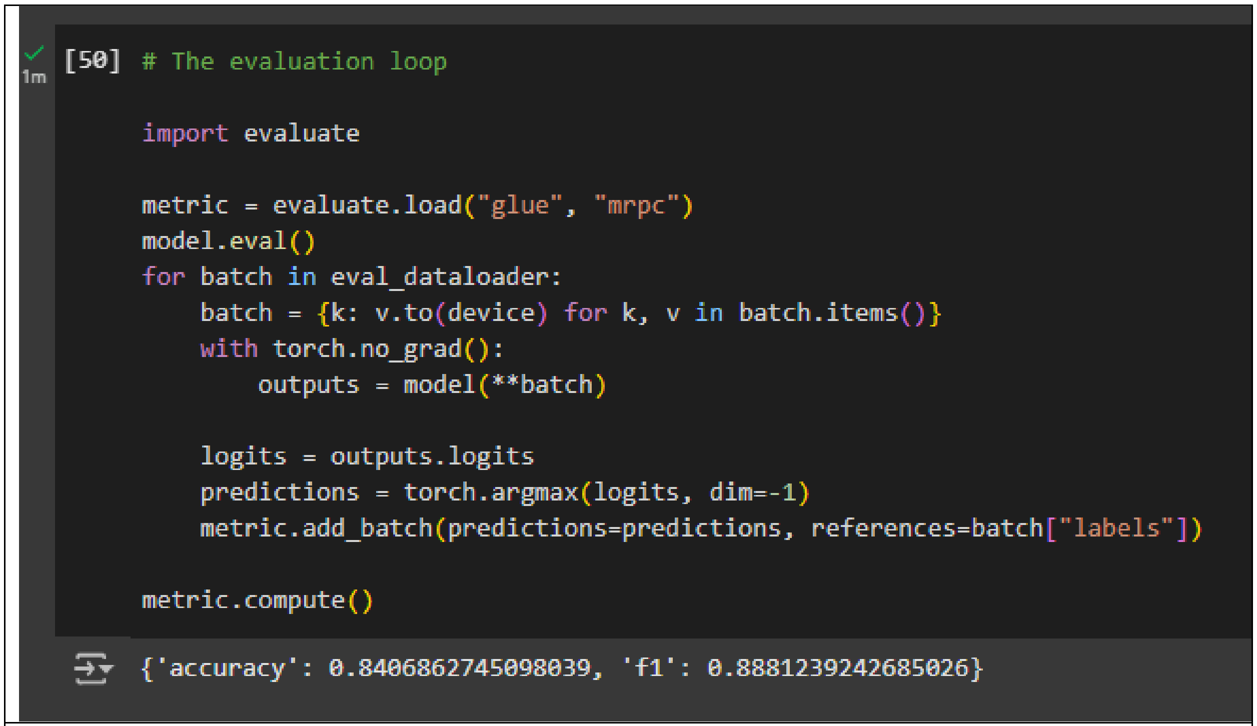
Kode ini digunakan untuk mempersiapkan perangkat (device) tempat model akan dilatih, apakah itu GPU (CUDA) atau CPU.



```
from tqdm.auto import tqdm

progress_bar = tqdm(range(num_training_steps))

model.train()
for epoch in range(num_epochs):
    for batch in train_dataloader:
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()

        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
        progress_bar.update(1)
```

```
26%                              357/1377 [2:08:16<2:02:25,  7.20s/it]
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-49-b86b10a2f8e7> in <cell line: 6>()
      9         outputs = model(**batch)
     10         loss = outputs.loss
---> 11         loss.backward()
     12
```

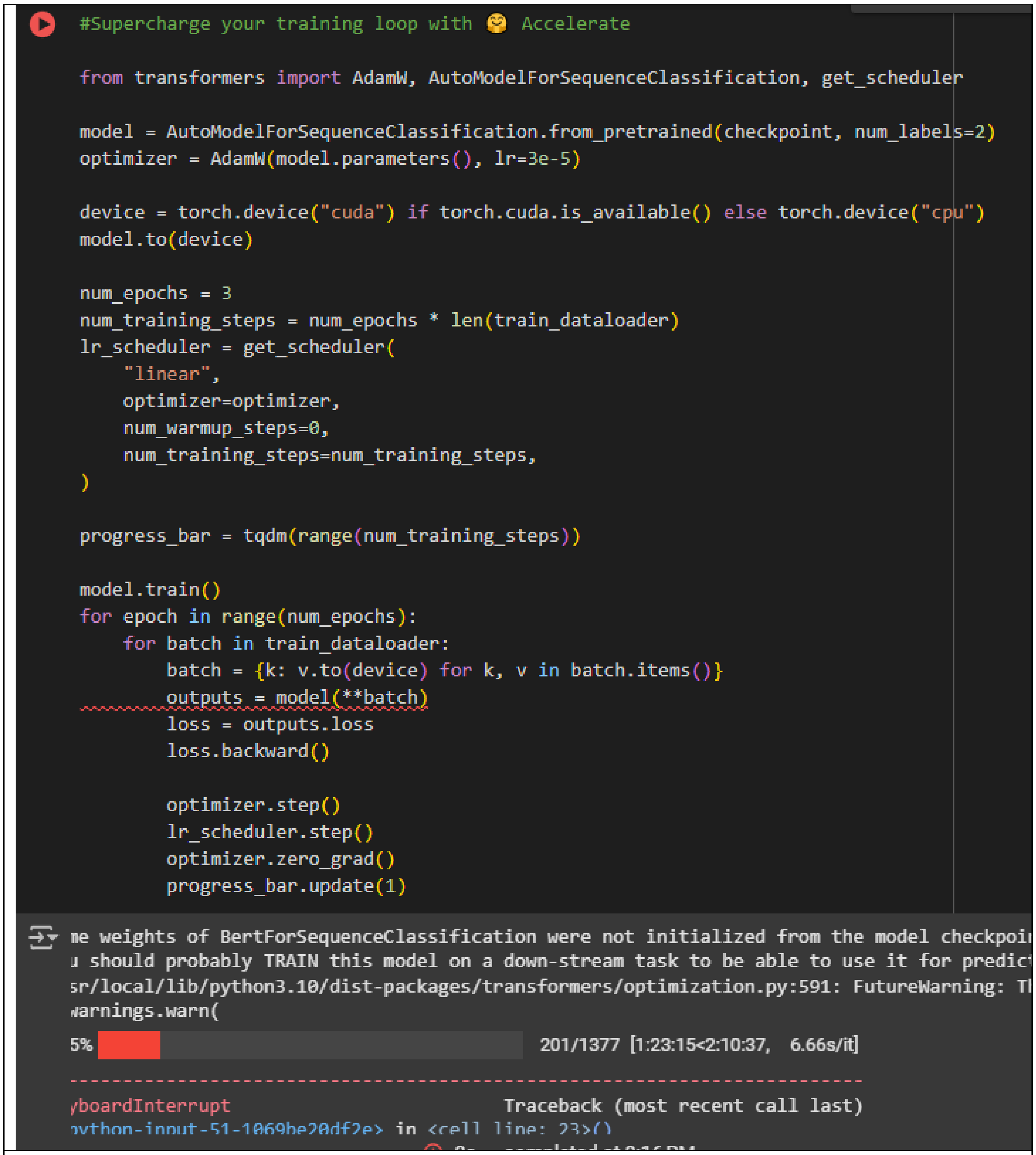
Kendala: Running sangat lama dan error dikarenakan data yg sangat banyak.



```
[50] # The evaluation loop

     import evaluate

     metric = evaluate.load("glue", "mrpc")
     model.eval()
     for batch in eval_dataloader:
         batch = {k: v.to(device) for k, v in batch.items()}
         with torch.no_grad():
             outputs = model(**batch)

         logits = outputs.logits
         predictions = torch.argmax(logits, dim=-1)
         metric.add_batch(predictions=predictions, references=batch["labels"])

     metric.compute()
```

```
{'accuracy': 0.8406862745098039, 'f1': 0.8881239242685026}
```

Kode ini digunakan untuk melakukan evaluasi model setelah pelatihan.

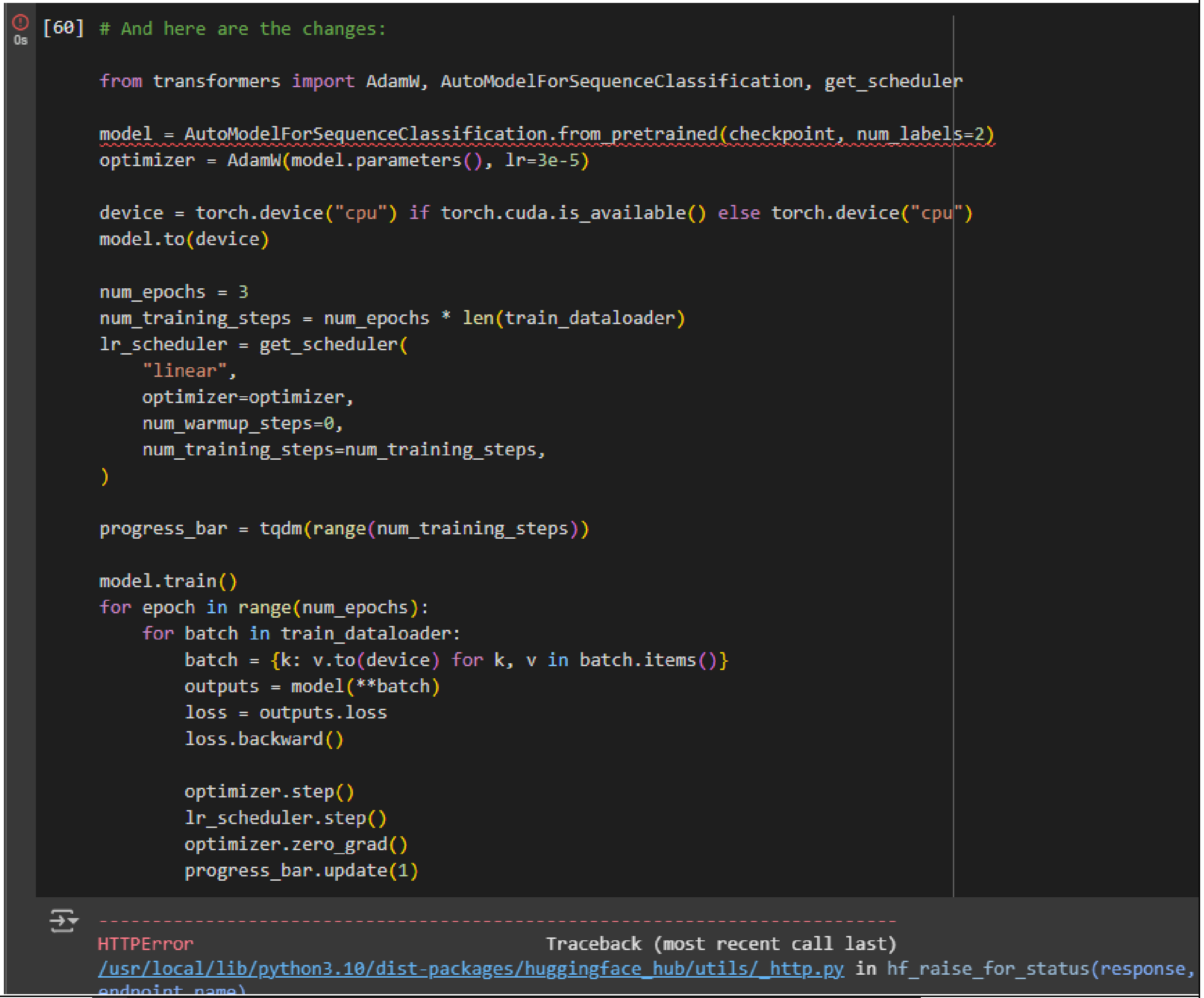


```
#Supercharge your training loop with 🤗 Accelerate

from transformers import AdamW, AutoModelForSequenceClassification, get_scheduler

model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)
optimizer = AdamW(model.parameters(), lr=3e-5)

device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
model.to(device)

num_epochs = 3
num_training_steps = num_epochs * len(train_dataloader)
lr_scheduler = get_scheduler(
    "linear",
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=num_training_steps,
)

progress_bar = tqdm(range(num_training_steps))

model.train()
for epoch in range(num_epochs):
    for batch in train_dataloader:
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()

        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
        progress_bar.update(1)
```

```
ne weights of BertForSequenceClassification were not initialized from the model checkpoi
u should probably TRAIN this model on a down-stream task to be able to use it for predic
sr/local/lib/python3.10/dist-packages/transformers/optimization.py:591: FutureWarning: Tl
warnings.warn(
5%  201/1377 [1:23:15<2:10:37,  6.66s/it]

---------------------------------------------------------------------------
yboardInterrupt                           Traceback (most recent call last)
ɔython-input-51-1069be20df2e> in <cell line: 23>()
```

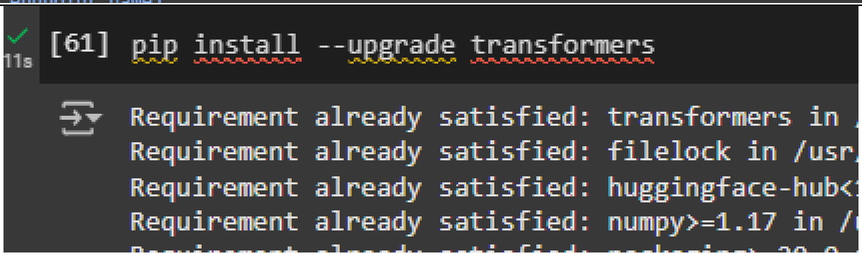
Kendala: Running sangat lama karena data yang sangt banyak.

Kendala:

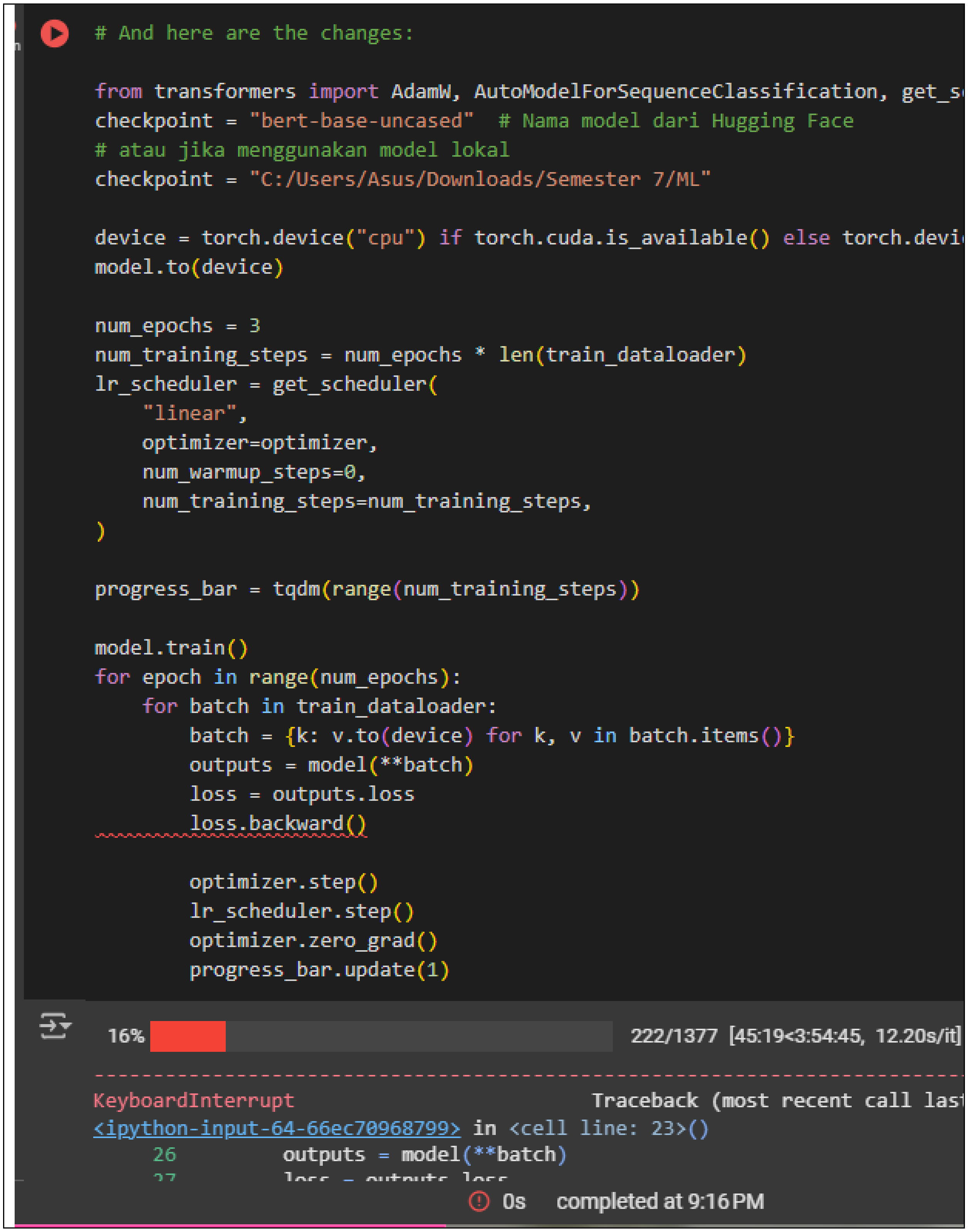


```
[60] # And here are the changes:

from transformers import AdamW, AutoModelForSequenceClassification, get_scheduler

model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)
optimizer = AdamW(model.parameters(), lr=3e-5)

device = torch.device("cpu") if torch.cuda.is_available() else torch.device("cpu")
model.to(device)

num_epochs = 3
num_training_steps = num_epochs * len(train_dataloader)
lr_scheduler = get_scheduler(
    "linear",
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=num_training_steps,
)

progress_bar = tqdm(range(num_training_steps))

model.train()
for epoch in range(num_epochs):
    for batch in train_dataloader:
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()

        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
        progress_bar.update(1)

---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_http.py in hf_raise_for_status(response,
endpoint_name)
```

Solusi:



```
[61] pip install --upgrade transformers

Requirement already satisfied: transformers in
Requirement already satisfied: filelock in /usr
Requirement already satisfied: huggingface-hub<
Requirement already satisfied: numpy>=1.17 in /
```

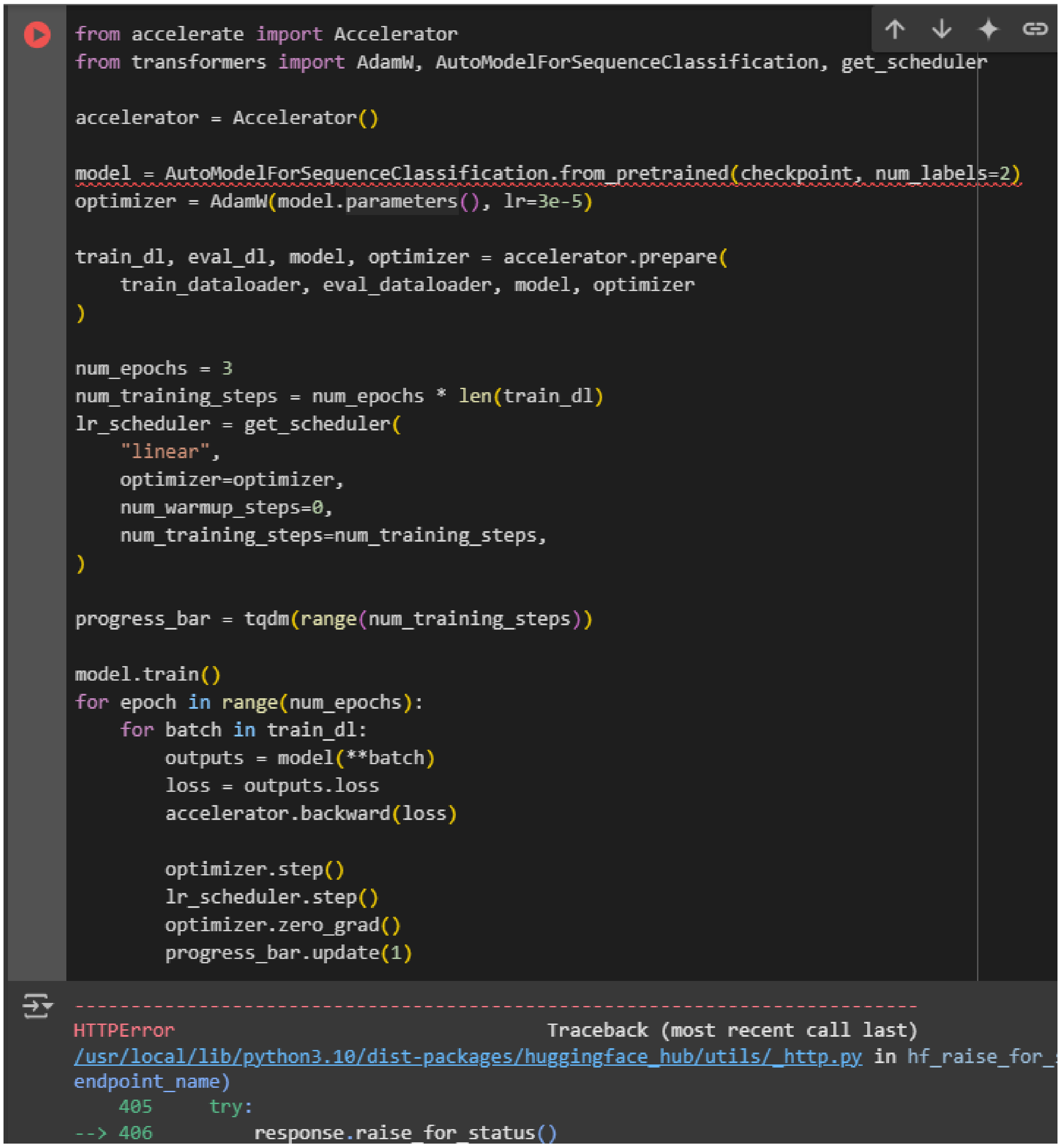


```
# And here are the changes:

from transformers import AdamW, AutoModelForSequenceClassification, get_s
checkpoint = "bert-base-uncased"  # Nama model dari Hugging Face
# atau jika menggunakan model lokal
checkpoint = "C:/Users/Asus/Downloads/Semester 7/ML"

device = torch.device("cpu") if torch.cuda.is_available() else torch.devi
model.to(device)

num_epochs = 3
num_training_steps = num_epochs * len(train_dataloader)
lr_scheduler = get_scheduler(
    "linear",
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=num_training_steps,
)

progress_bar = tqdm(range(num_training_steps))

model.train()
for epoch in range(num_epochs):
    for batch in train_dataloader:
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()

        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
        progress_bar.update(1)
```

```
16%  222/1377 [45:19<3:54:45, 12.20s/it]
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call las
<ipython-input-64-66ec70968799> in <cell line: 23>()
     26         outputs = model(**batch)
```

0s completed at 9:16 PM

Kendala:

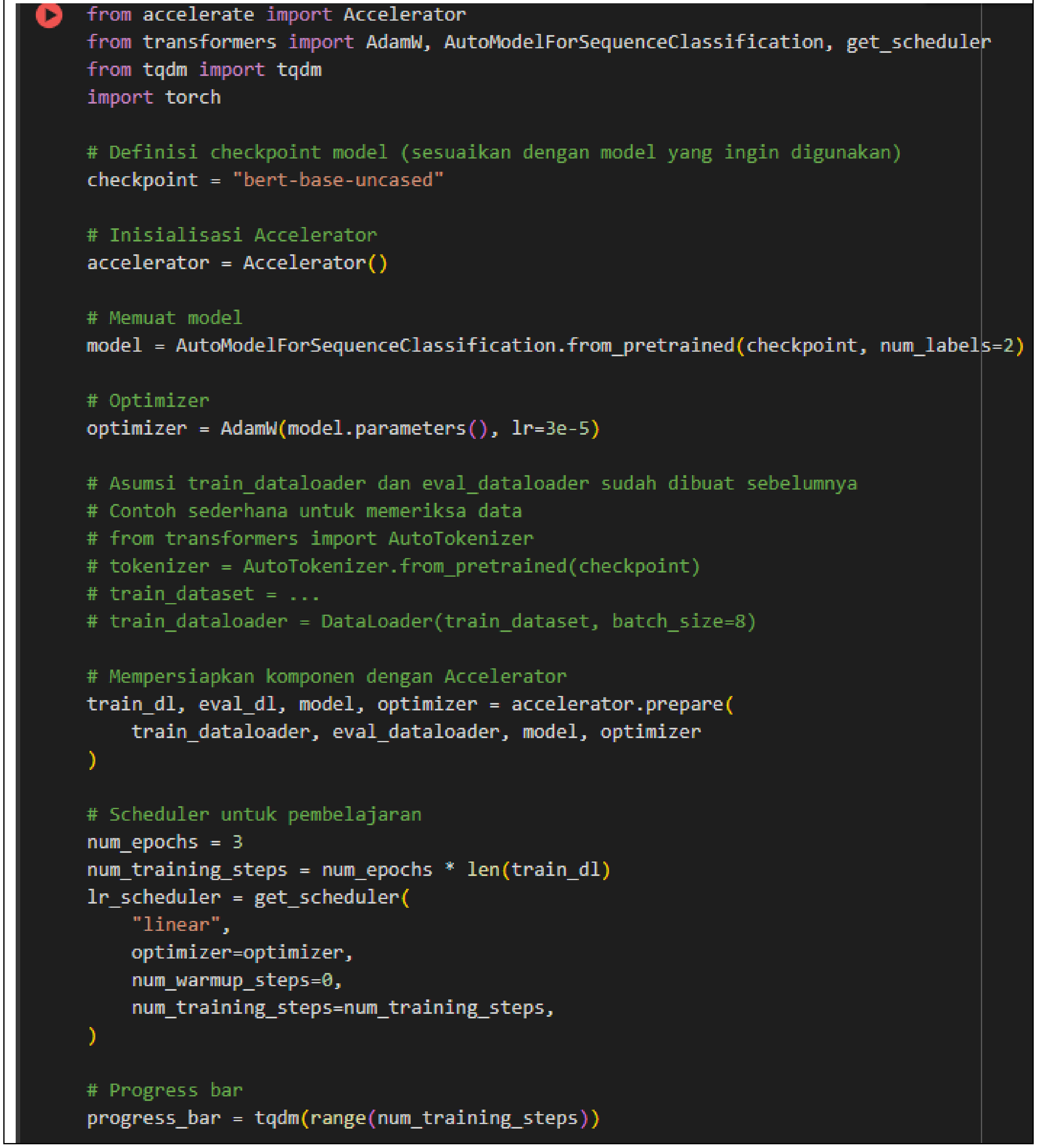


```
from accelerate import Accelerator
from transformers import AdamW, AutoModelForSequenceClassification, get_scheduler

accelerator = Accelerator()

model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)
optimizer = AdamW(model.parameters(), lr=3e-5)

train_dl, eval_dl, model, optimizer = accelerator.prepare(
    train_dataloader, eval_dataloader, model, optimizer
)

num_epochs = 3
num_training_steps = num_epochs * len(train_dl)
lr_scheduler = get_scheduler(
    "linear",
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=num_training_steps,
)

progress_bar = tqdm(range(num_training_steps))

model.train()
for epoch in range(num_epochs):
    for batch in train_dl:
        outputs = model(**batch)
        loss = outputs.loss
        accelerator.backward(loss)

        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
        progress_bar.update(1)
```

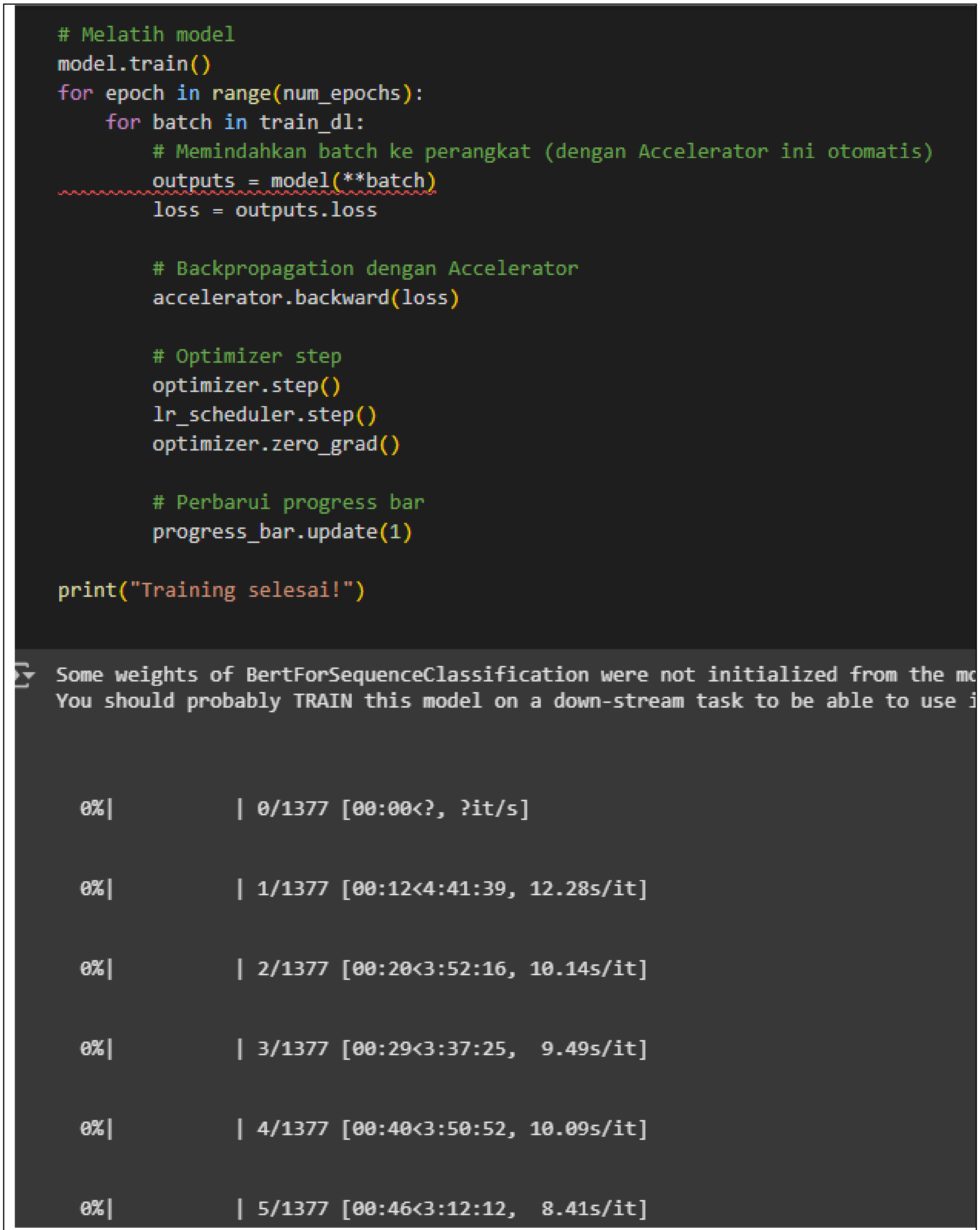
```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_http.py in hf_raise_for_
endpoint_name)
    405     try:
--> 406         response.raise_for_status()
```

Variabel checkpoint Tidak Didefinisikan Tambahkan definisi untuk variabel checkpoint dengan model yang valid. Data Tidak Didefinisikan train_dataloader dan eval_dataloader perlu didefinisikan sebelumnya. batch Tidak Dipindahkan ke Perangkat Perlu memindahkan batch ke perangkat (device) jika tidak menggunakan Accelerator untuk itu. tqdm Tidak Diimpor Tambahkan impor tqdm jika belum dilakukan.

Solusi:



```
from accelerate import Accelerator
from transformers import AdamW, AutoModelForSequenceClassification, get_scheduler
from tqdm import tqdm
import torch

# Definisi checkpoint model (sesuaikan dengan model yang ingin digunakan)
checkpoint = "bert-base-uncased"

# Inisialisasi Accelerator
accelerator = Accelerator()

# Memuat model
model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)

# Optimizer
optimizer = AdamW(model.parameters(), lr=3e-5)

# Asumsi train_dataloader dan eval_dataloader sudah dibuat sebelumnya
# Contoh sederhana untuk memeriksa data
# from transformers import AutoTokenizer
# tokenizer = AutoTokenizer.from_pretrained(checkpoint)
# train_dataset = ...
# train_dataloader = DataLoader(train_dataset, batch_size=8)

# Mempersiapkan komponen dengan Accelerator
train_dl, eval_dl, model, optimizer = accelerator.prepare(
    train_dataloader, eval_dataloader, model, optimizer
)

# Scheduler untuk pembelajaran
num_epochs = 3
num_training_steps = num_epochs * len(train_dl)
lr_scheduler = get_scheduler(
    "linear",
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=num_training_steps,
)

# Progress bar
progress_bar = tqdm(range(num_training_steps))
```



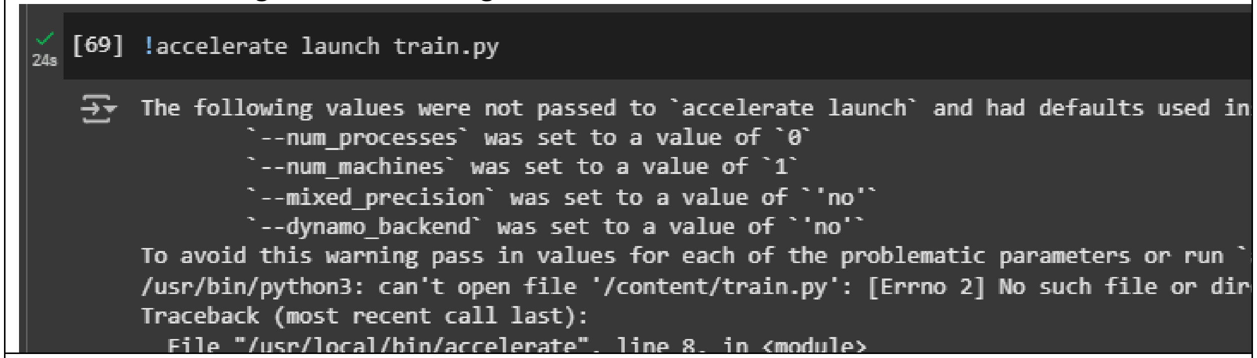
```
# Melatih model
model.train()
for epoch in range(num_epochs):
    for batch in train_dl:
        # Memindahkan batch ke perangkat (dengan Accelerator ini otomatis)
        outputs = model(**batch)
        loss = outputs.loss

        # Backpropagation dengan Accelerator
        accelerator.backward(loss)

        # Optimizer step
        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()

        # Perbarui progress bar
        progress_bar.update(1)

print("Training selesai!")
```

```
Some weights of BertForSequenceClassification were not initialized from the mo
You should probably TRAIN this model on a down-stream task to be able to use i


  0%|          | 0/1377 [00:00<?, ?it/s]

  0%|          | 1/1377 [00:12<4:41:39, 12.28s/it]

  0%|          | 2/1377 [00:20<3:52:16, 10.14s/it]

  0%|          | 3/1377 [00:29<3:37:25,  9.49s/it]

  0%|          | 4/1377 [00:40<3:50:52, 10.09s/it]

  0%|          | 5/1377 [00:46<3:12:12,  8.41s/it]
```

checkpoint: Pastikan model yang digunakan valid, seperti bert-base-uncased.
DataLoader: perlu memastikan bahwa train_dataloader dan eval_dataloader sudah didefinisikan dengan dataset yang benar.

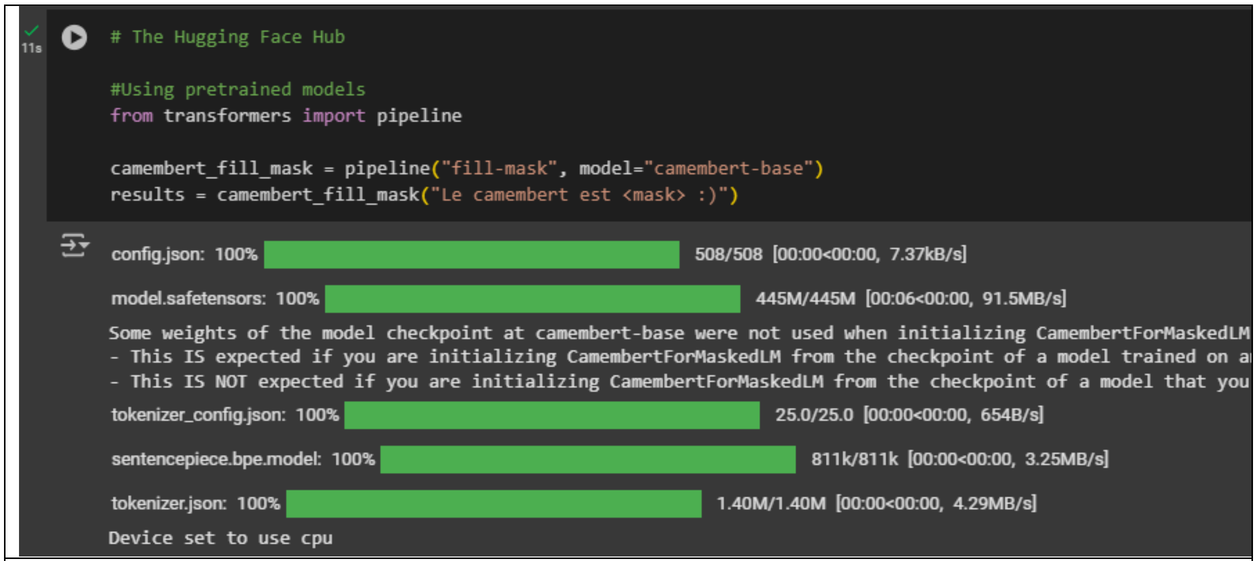
accelerator.backward(): Metode ini menggantikan loss.backward() untuk mendukung multi-GPU atau mixed precision.
Progress Bar: Progress bar diatur menggunakan tqdm untuk memantau langkah selama pelatihan.

menambahkan ! agar bisa di running bukan di terminal



```
[69] !accelerate launch train.py

The following values were not passed to `accelerate launch` and had defaults used in
        `--num_processes` was set to a value of `0`
        `--num_machines` was set to a value of `1`
        `--mixed_precision` was set to a value of `'no'`
        `--dynamo_backend` was set to a value of `'no'`
To avoid this warning pass in values for each of the problematic parameters or run `
/usr/bin/python3: can't open file '/content/train.py': [Errno 2] No such file or dir
Traceback (most recent call last):
  File "/usr/local/bin/accelerate", line 8, in <module>
```

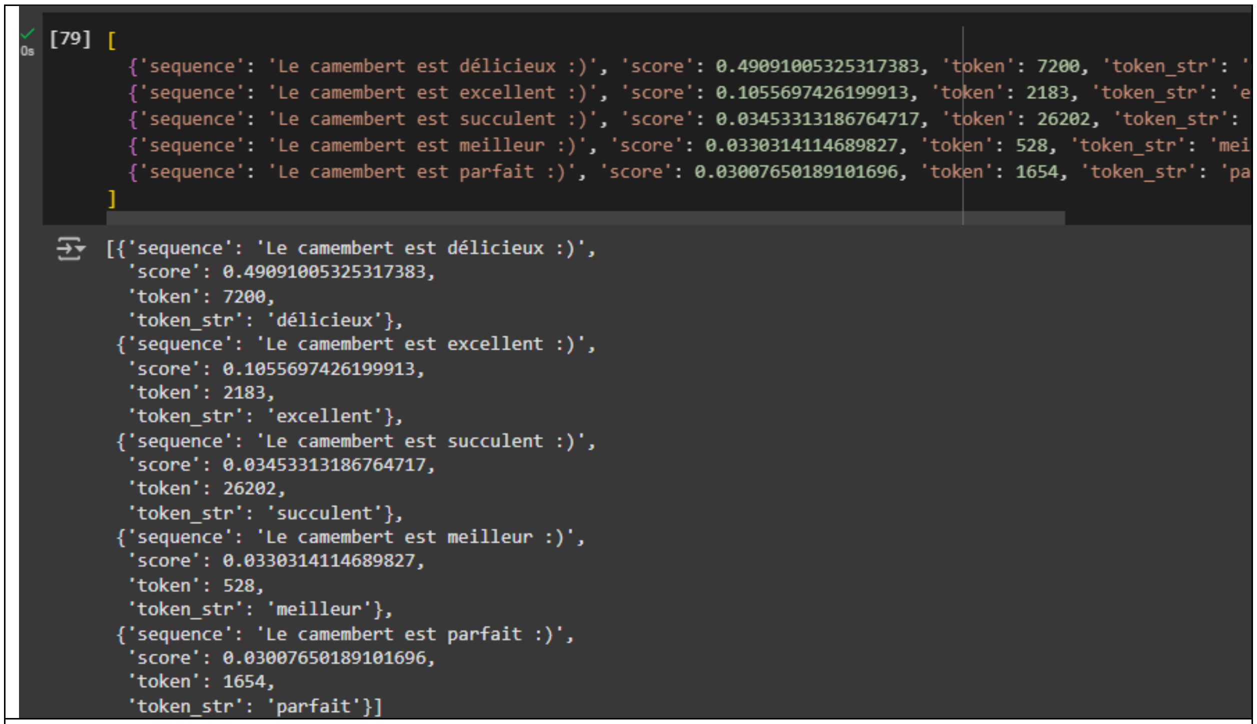
digunakan untuk meluncurkan pelatihan model menggunakan *Accelerate*, pustaka dari Hugging Face yang dirancang untuk menyederhanakan proses pelatihan model di berbagai perangkat (misalnya, satu atau lebih GPU, atau CPU) dengan konfigurasi yang lebih mudah dan lebih efisien.



```
# The Hugging Face Hub

#Using pretrained models
from transformers import pipeline

camembert_fill_mask = pipeline("fill-mask", model="camembert-base")
results = camembert_fill_mask("Le camembert est <mask> :)")
```

```
config.json: 100%  508/508 [00:00<00:00, 7.37kB/s]
model.safetensors: 100%  445M/445M [00:06<00:00, 91.5MB/s]
Some weights of the model checkpoint at camembert-base were not used when initializing CamembertForMaskedLM
- This IS expected if you are initializing CamembertForMaskedLM from the checkpoint of a model trained on a
- This IS NOT expected if you are initializing CamembertForMaskedLM from the checkpoint of a model that you
tokenizer_config.json: 100%  25.0/25.0 [00:00<00:00, 654B/s]
sentencepiece.bpe.model: 100%  811k/811k [00:00<00:00, 3.25MB/s]
tokenizer.json: 100%  1.40M/1.40M [00:00<00:00, 4.29MB/s]
Device set to use cpu
```

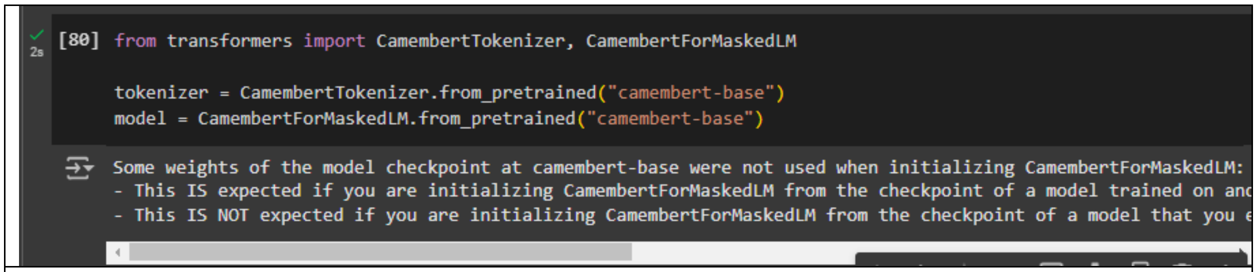
Kode ini menggunakan pipeline dari Hugging Face untuk melakukan tugas pengisian masker pada model bahasa *Camembert*.



```
[79] [
      {'sequence': 'Le camembert est délicieux :)', 'score': 0.49091005325317383, 'token': 7200, 'token_str': '
      {'sequence': 'Le camembert est excellent :)', 'score': 0.1055697426199913, 'token': 2183, 'token_str': 'e
      {'sequence': 'Le camembert est succulent :)', 'score': 0.03453313186764717, 'token': 26202, 'token_str':
      {'sequence': 'Le camembert est meilleur :)', 'score': 0.0330314114689827, 'token': 528, 'token_str': 'mei
      {'sequence': 'Le camembert est parfait :)', 'score': 0.03007650189101696, 'token': 1654, 'token_str': 'pa
     ]
```

```
[{'sequence': 'Le camembert est délicieux :)',
  'score': 0.49091005325317383,
  'token': 7200,
  'token_str': 'délicieux'},
 {'sequence': 'Le camembert est excellent :)',
  'score': 0.1055697426199913,
  'token': 2183,
  'token_str': 'excellent'},
 {'sequence': 'Le camembert est succulent :)',
  'score': 0.03453313186764717,
  'token': 26202,
  'token_str': 'succulent'},
 {'sequence': 'Le camembert est meilleur :)',
  'score': 0.0330314114689827,
  'token': 528,
  'token_str': 'meilleur'},
 {'sequence': 'Le camembert est parfait :)',
  'score': 0.03007650189101696,
  'token': 1654,
  'token_str': 'parfait'}]
```

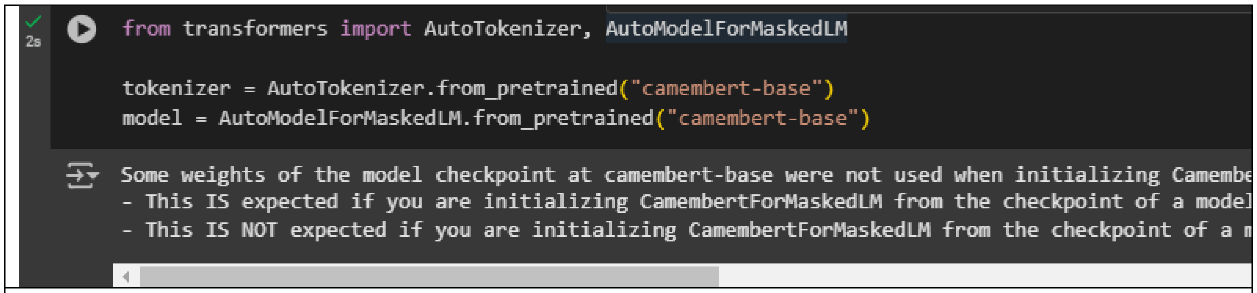
berisi beberapa prediksi dari model *Camembert* untuk mengisi masker dalam kalimat "Le camembert est <mask> :)"



```
[80] from transformers import CamembertTokenizer, CamembertForMaskedLM

     tokenizer = CamembertTokenizer.from_pretrained("camembert-base")
     model = CamembertForMaskedLM.from_pretrained("camembert-base")
```

```
Some weights of the model checkpoint at camembert-base were not used when initializing CamembertForMaskedLM:
- This IS expected if you are initializing CamembertForMaskedLM from the checkpoint of a model trained on anc
- This IS NOT expected if you are initializing CamembertForMaskedLM from the checkpoint of a model that you e
```

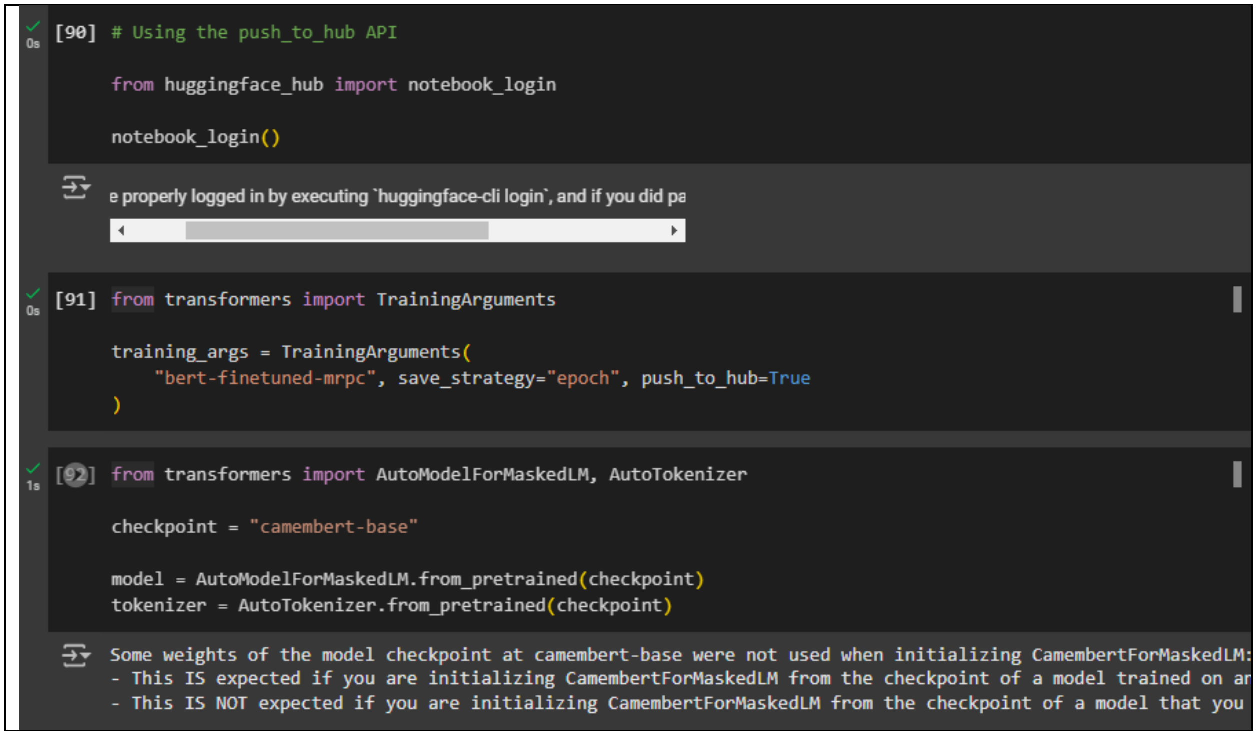
Kode ini memuat model dan tokenizer *Camembert* untuk tugas pengisian masker.



```
from transformers import AutoTokenizer, AutoModelForMaskedLM

tokenizer = AutoTokenizer.from_pretrained("camembert-base")
model = AutoModelForMaskedLM.from_pretrained("camembert-base")
```

```
Some weights of the model checkpoint at camembert-base were not used when initializing Camembe
- This IS expected if you are initializing CamembertForMaskedLM from the checkpoint of a model
- This IS NOT expected if you are initializing CamembertForMaskedLM from the checkpoint of a m
```

Kode ini menggunakan AutoTokenizer dan AutoModelForMaskedLM dari pustaka *transformers* untuk memuat model dan tokenizer *Camembert* dengan cara yang lebih generik, yang memungkinkan Anda menggunakan model dan tokenizer yang dapat dipertukarkan antara berbagai jenis model tanpa menyebutkan model secara spesifik.



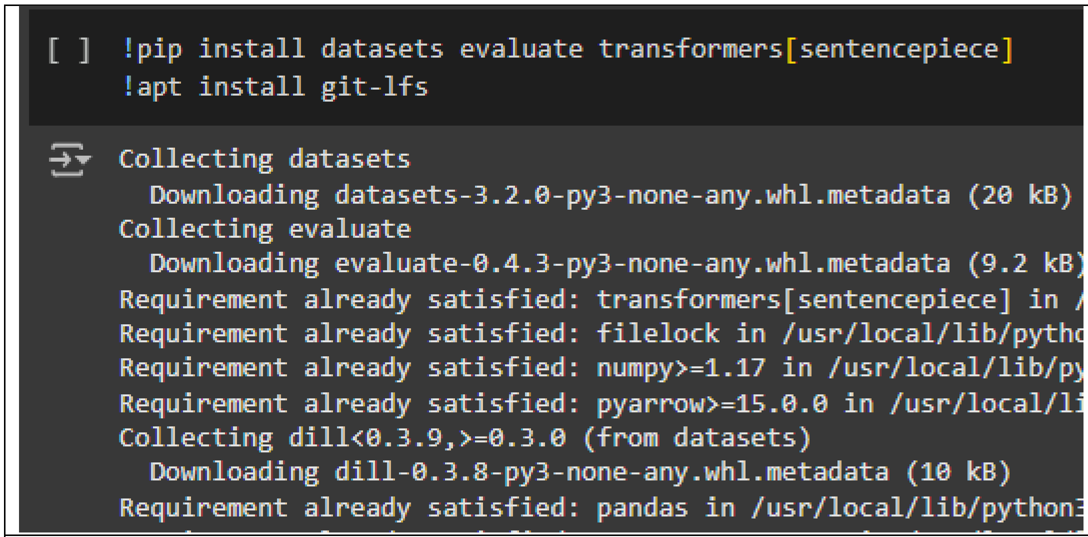
```
[90] # Using the push_to_hub API

     from huggingface_hub import notebook_login

     notebook_login()
```

```
e properly logged in by executing `huggingface-cli login`, and if you did pa
```

```
[91] from transformers import TrainingArguments

     training_args = TrainingArguments(
         "bert-finetuned-mrpc", save_strategy="epoch", push_to_hub=True
     )
```

```
[92] from transformers import AutoModelForMaskedLM, AutoTokenizer

     checkpoint = "camembert-base"

     model = AutoModelForMaskedLM.from_pretrained(checkpoint)
     tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

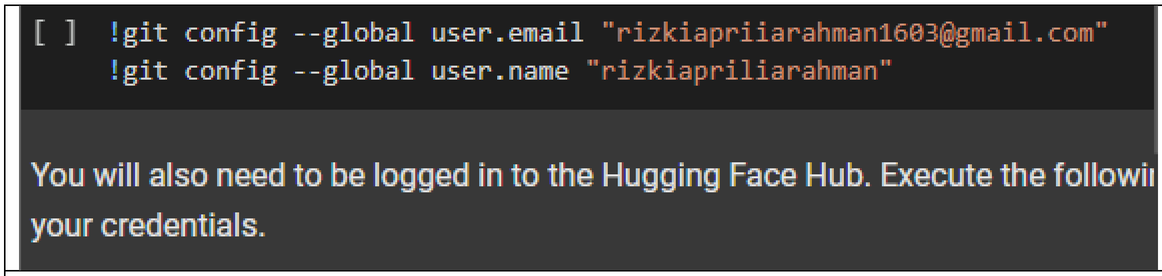
```
Some weights of the model checkpoint at camembert-base were not used when initializing CamembertForMaskedLM:
- This IS expected if you are initializing CamembertForMaskedLM from the checkpoint of a model trained on an
- This IS NOT expected if you are initializing CamembertForMaskedLM from the checkpoint of a model that you
```



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

```
Collecting datasets
  Downloading datasets-3.2.0-py3-none-any.whl.metadata (20 kB)
Collecting evaluate
  Downloading evaluate-0.4.3-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: transformers[sentencepiece] in /
Requirement already satisfied: filelock in /usr/local/lib/pytho
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/py
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/li
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
```

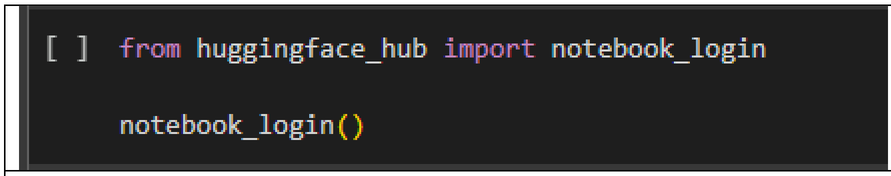
Dengan menjalankan kedua perintah tersebut, akan mendapatkan pustaka yang diperlukan untuk bekerja dengan dataset dan model dari Hugging Face, serta kemampuan untuk mengelola file besar melalui Git LFS.



```
[ ]  !git config --global user.email "rizkiapriiarahman1603@gmail.com"
     !git config --global user.name "rizkiapriliarahman"
```

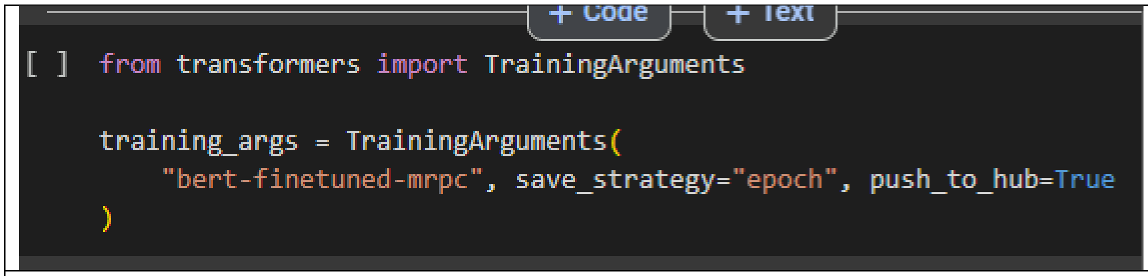
You will also need to be logged in to the Hugging Face Hub. Execute the followi
your credentials.

need to setup git. So, I adapt my email and name in the following cell.



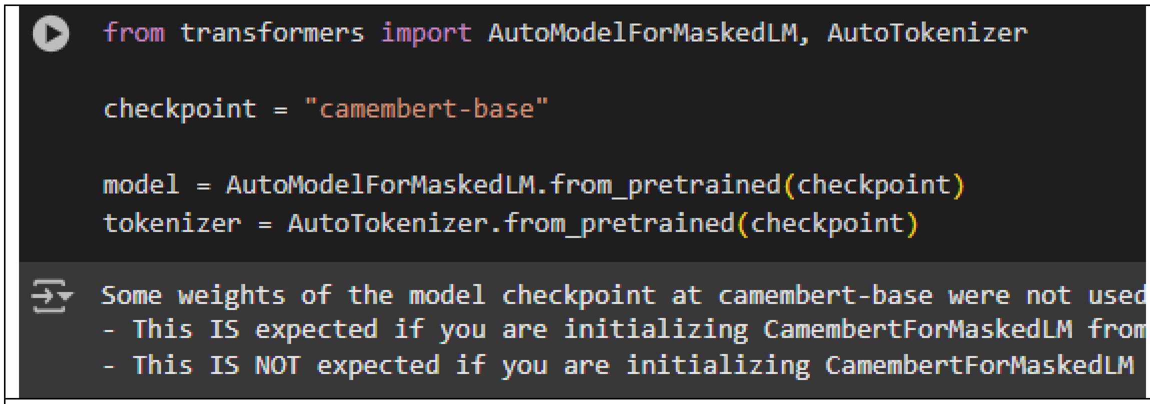
```
[ ]  from huggingface_hub import notebook_login

     notebook_login()
```

Perintah notebook_login() dari pustaka huggingface_hub digunakan untuk masuk ke akun Hugging Face Hub dalam lingkungan notebook.



```
[ ]  from transformers import TrainingArguments

     training_args = TrainingArguments(
         "bert-finetuned-mrpc", save_strategy="epoch", push_to_hub=True
     )
```

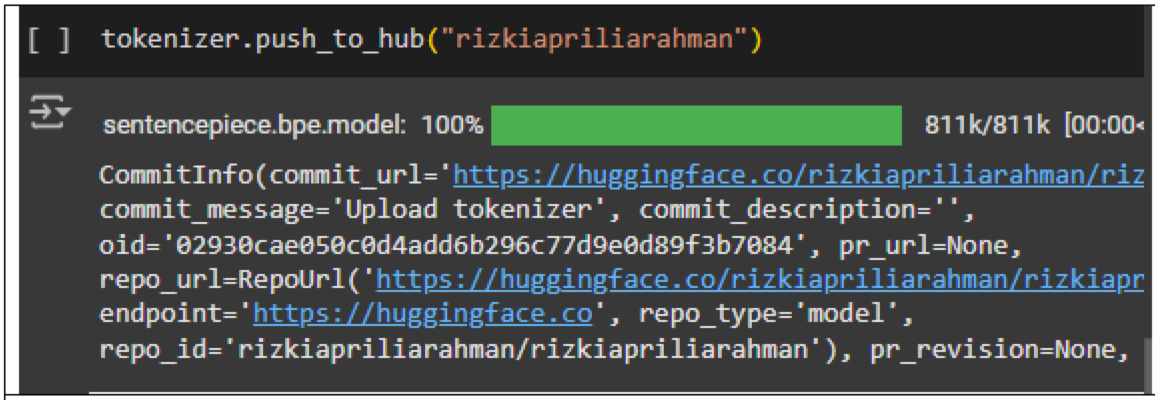
Perintah ini mengonfigurasi TrainingArguments untuk melatih model menggunakan *Transformers* library



```
from transformers import AutoModelForMaskedLM, AutoTokenizer

checkpoint = "camembert-base"

model = AutoModelForMaskedLM.from_pretrained(checkpoint)
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

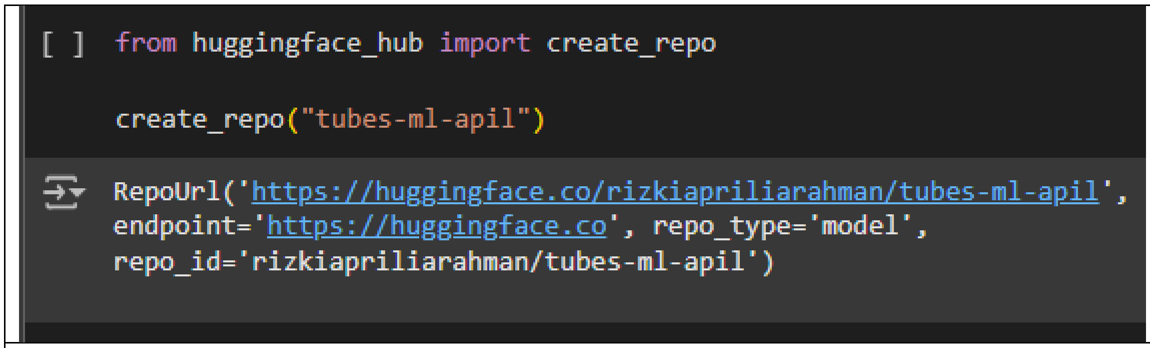
```
Some weights of the model checkpoint at camembert-base were not used
- This IS expected if you are initializing CamembertForMaskedLM from
- This IS NOT expected if you are initializing CamembertForMaskedLM
```

Kode ini digunakan untuk memuat model *Camembert* dan tokenizer-nya, yang dapat digunakan untuk tugas *Masked Language Modeling* (MLM),



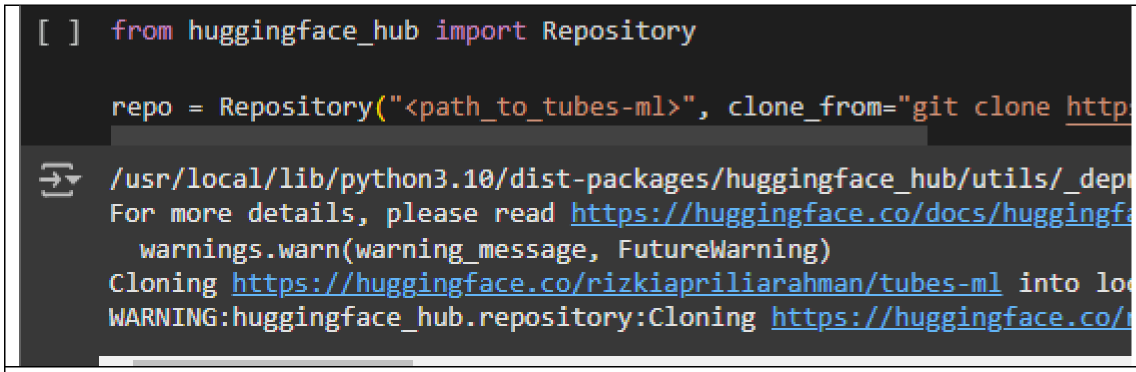
```
[ ]  tokenizer.push_to_hub("rizkiapriliarahman")

sentencepiece.bpe.model: 100%  811k/811k [00:00
CommitInfo(commit_url='https://huggingface.co/rizkiapriliarahman/riz
commit_message='Upload tokenizer', commit_description='',
oid='02930cae050c0d4add6b296c77d9e0d89f3b7084', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/rizkiapriliarahman/rizkiapr
endpoint='https://huggingface.co', repo_type='model',
repo_id='rizkiapriliarahman/rizkiapriliarahman'), pr_revision=None,
```

Perintah tokenizer.push_to_hub("rizkiapriliarahman") digunakan untuk mengunggah tokenizer ke Hugging Face Hub dengan nama repositori **rizkiapriliarahman**. Setelah berhasil diunggah, tokenizer akan tersedia di Hugging Face Hub dan dapat diakses atau digunakan oleh siapa saja dengan repositori tersebut.



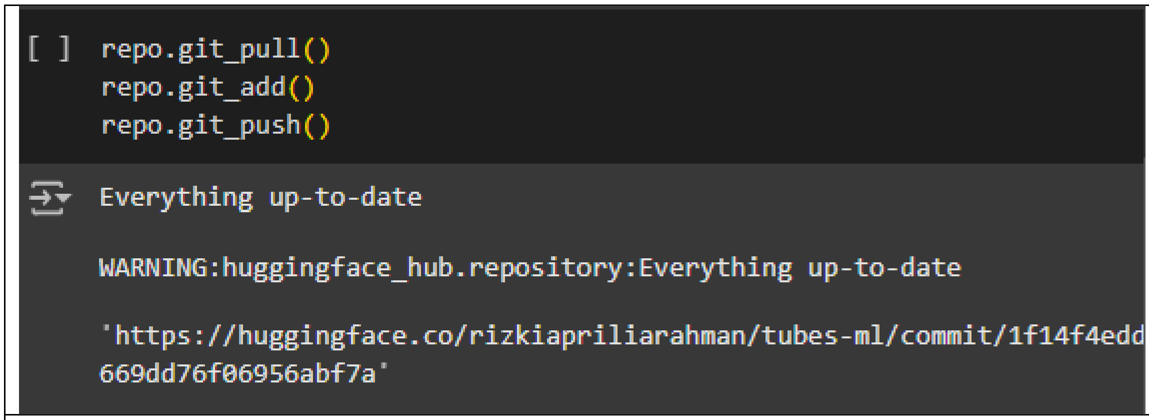
```
[ ]  from huggingface_hub import create_repo

     create_repo("tubes-ml-apil")

RepoUrl('https://huggingface.co/rizkiapriliarahman/tubes-ml-apil',
endpoint='https://huggingface.co', repo_type='model',
repo_id='rizkiapriliarahman/tubes-ml-apil')
```

Perintah create_repo("tubes-ml-apil") digunakan untuk membuat repositori baru di Hugging Face Hub dengan nama **tubes-ml-apil**.



```
[ ]  from huggingface_hub import Repository

     repo = Repository("<path_to_tubes-ml>", clone_from="git clone http

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_dep
For more details, please read https://huggingface.co/docs/huggingf
  warnings.warn(warning_message, FutureWarning)
Cloning https://huggingface.co/rizkiapriliarahman/tubes-ml into lo
WARNING:huggingface_hub.repository:Cloning https://huggingface.co/r
```

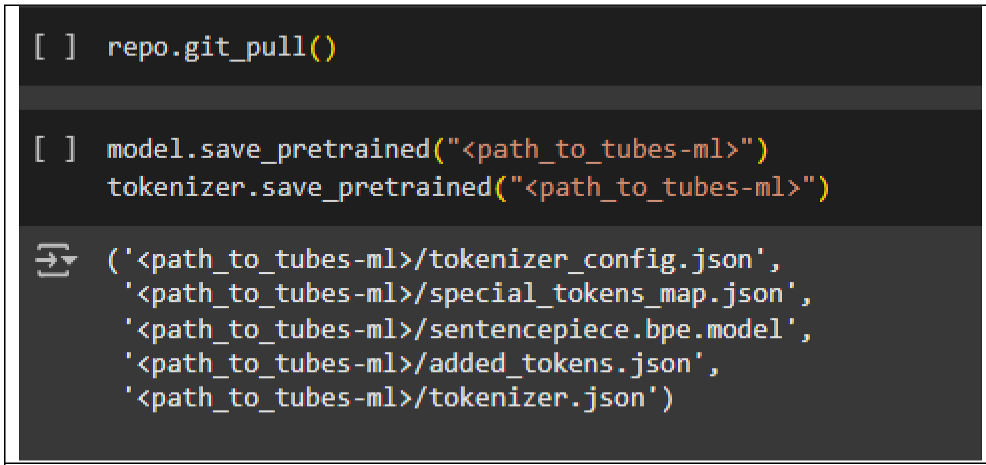
Setelah perintah ini dijalankan, repositori dari Hugging Face Hub akan dikloning ke folder lokal yang ditentukan, dan dapat mulai bekerja dengan file di dalamnya, serta mengunggah perubahan atau file baru ke repositori menggunakan push_to_hub.



```
[ ] repo.git_pull()
    repo.git_add()
    repo.git_push()
```

```
Everything up-to-date

WARNING:huggingface_hub.repository:Everything up-to-date

'https://huggingface.co/rizkiapriliarahman/tubes-ml/commit/1f14f4edd
669dd76f06956abf7a'
```

Perintah repo.git_pull(), repo.git_add(), dan repo.git_push() digunakan untuk melakukan operasi Git pada repositori lokal yang telah di kloning dari Hugging Face Hub.

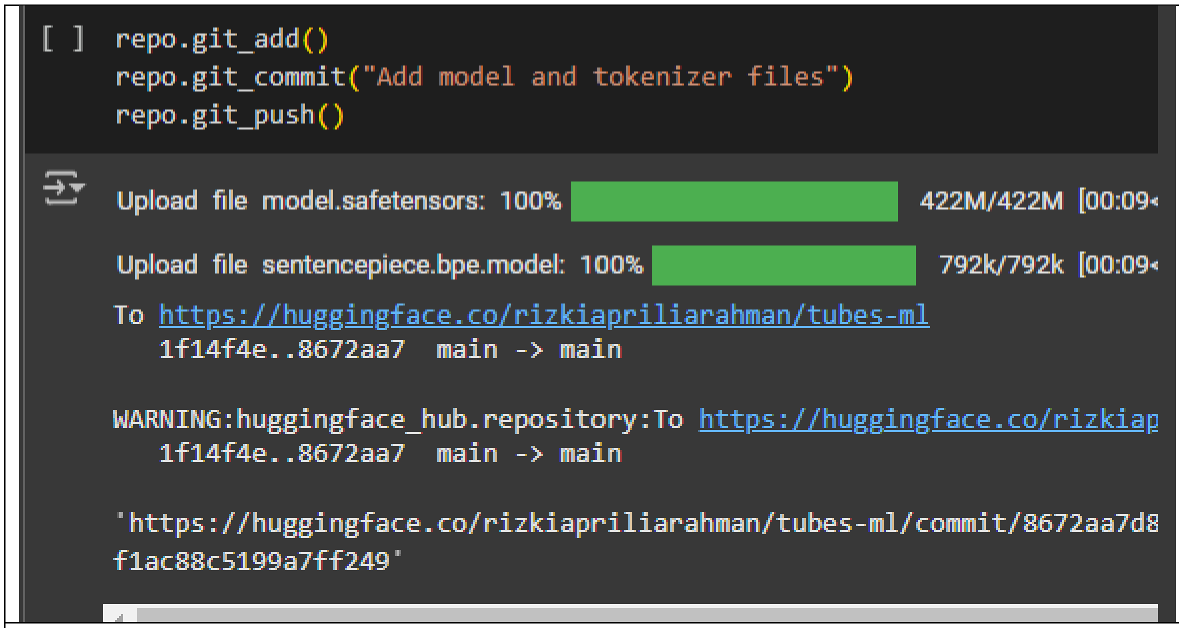


```
[ ] repo.git_pull()
```

```
[ ] model.save_pretrained("<path_to_tubes-ml>")
    tokenizer.save_pretrained("<path_to_tubes-ml>")
```

```
('<path_to_tubes-ml>/tokenizer_config.json',
 '<path_to_tubes-ml>/special_tokens_map.json',
 '<path_to_tubes-ml>/sentencepiece.bpe.model',
 '<path_to_tubes-ml>/added_tokens.json',
 '<path_to_tubes-ml>/tokenizer.json')
```

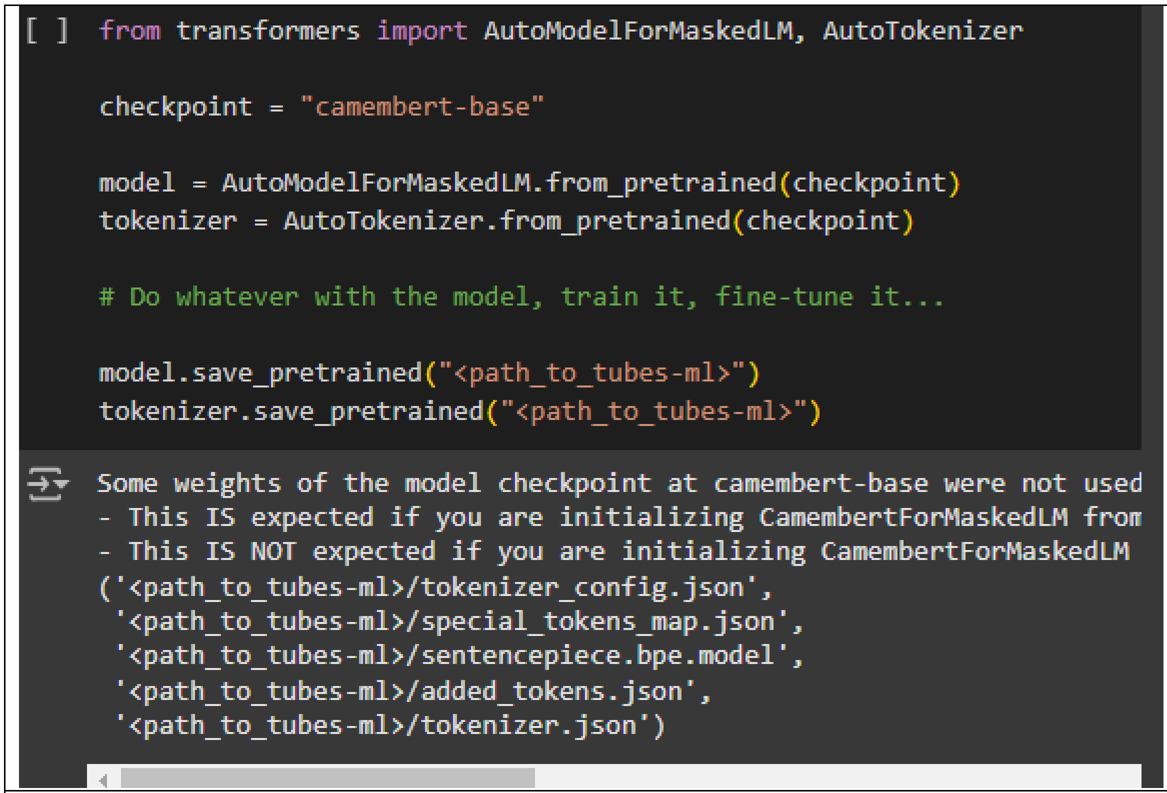
Perintah model.save_pretrained("<path_to_tubes-ml>") dan tokenizer.save_pretrained("<path_to_tubes-ml>") digunakan untuk menyimpan model dan tokenizer ke folder lokal. disini perlu mengganti "<path_to_tubes-ml>" dengan path lokal yang diinginkan di system.



```
[ ] repo.git_add()
    repo.git_commit("Add model and tokenizer files")
    repo.git_push()
```

```
Upload file model.safetensors: 100%                422M/422M [00:09
Upload file sentencepiece.bpe.model: 100%          792k/792k [00:09
To https://huggingface.co/rizkiapriliarahman/tubes-ml
   1f14f4e..8672aa7  main -> main

WARNING:huggingface_hub.repository:To https://huggingface.co/rizkiap
   1f14f4e..8672aa7  main -> main

'https://huggingface.co/rizkiapriliarahman/tubes-ml/commit/8672aa7d8
f1ac88c5199a7ff249'
```

Perintah repo.git_add(), repo.git_commit(), dan repo.git_push() digunakan untuk mengelola perubahan pada repositori Git dan mendorong perubahan ke repositori Hugging Face Hub.



```
[ ] from transformers import AutoModelForMaskedLM, AutoTokenizer

    checkpoint = "camembert-base"

    model = AutoModelForMaskedLM.from_pretrained(checkpoint)
    tokenizer = AutoTokenizer.from_pretrained(checkpoint)

    # Do whatever with the model, train it, fine-tune it...

    model.save_pretrained("<path_to_tubes-ml>")
    tokenizer.save_pretrained("<path_to_tubes-ml>")
```

```
Some weights of the model checkpoint at camembert-base were not used
- This IS expected if you are initializing CamembertForMaskedLM from
- This IS NOT expected if you are initializing CamembertForMaskedLM
('<path_to_tubes-ml>/tokenizer_config.json',
 '<path_to_tubes-ml>/special_tokens_map.json',
 '<path_to_tubes-ml>/sentencepiece.bpe.model',
 '<path_to_tubes-ml>/added_tokens.json',
 '<path_to_tubes-ml>/tokenizer.json')
```

Perintah yang ini adalah cara untuk memuat model dan tokenizer dari model camembert-base, melakukan proses (seperti pelatihan atau fine-tuning) pada model tersebut, dan kemudian menyimpan model dan tokenizer ke folder lokal.