

Intro Hugging Face and Python

Dirk Wulff & Zak Hussain



MAX PLANCK INSTITUTE
FOR HUMAN DEVELOPMENT



OUR SOFTWARE STACK



+



+



OUR SOFTWARE STACK



+



+



WHY IS HUGGING FACE?



WHY IS HUGGING FACE?



Traditional language modelling pipeline:

WHY IS HUGGING FACE?



Traditional language modelling pipeline:

1. Find out the model architecture

WHY IS HUGGING FACE?



Traditional language modelling pipeline:

1. Find out the model architecture
2. Implement the model architecture in code with deep learning frameworks (e.g PyTorch/Tensorflow).

1. DEEP LEARNING LIBRARIES CAN BE DIFFICULT

1. DEEP LEARNING LIBRARIES CAN BE DIFFICULT

I  KING HATE TENSORFLOW #53549

✓ Closed

ghost opened this issue 9 hours ago · 2 comments

WHY HUGGING FACE?

Traditional language modelling pipeline:

1. Find out the model architecture
2. Implement the model architecture in code with deep learning libraries (e.g PyTorch/Tensorflow).

WHY HUGGING FACE?

Traditional language modelling pipeline:

1. Find out the model architecture
2. Implement the model architecture in code with deep learning libraries (e.g PyTorch/Tensorflow).
3. Load the pretrained weights (if available) from a server.
4. Process the inputs (using the correct tokenizer for the model)
5. Implement data loaders
6. Define a loss function
7. Stick a task-specific “head” on the model

HUGGING FACE PIPELINES

Import pipeline

```
from transformers import pipeline
```

Initialise
pipeline

```
pipe = pipeline('text-generation', model='gpt2')
```

Load model
input

```
prompt = """
```

```
Once upon a time in a land far far away, there was a young prince named  
John. He was known for his bravery and courage. One day, he decided to go on  
an adventure to explore the unknown lands.
```


```
"""
```

Feed input
the model

```
output = pipe(prompt, max_length=100)
```

```
print(output)
```

HUGGING FACE

 **Hugging Face**

Main Tasks Libraries Languages Licenses Other

Tasks

Text Generation

Any-to-Any

Image-Text-to-Text

Image-to-Text

Image-to-Image

Text-to-Image

Text-to-Video

Text-to-Speech

+ 42

Parameters

< 1B

6B

12B

32B

128B

> 500B

Libraries

PyTorch

TensorFlow

JAX

Transformers

Diffusers

sentence-transformers

Safetensors

ONNX

GGUF

Transformers.js

MLX

+ 41

Apps

vLLM

TGI

llama.cpp

MLX LM

LM Studio

Ollama

Jan

+ 13

Inference Providers

Groq

Novita

Nebius AI

Cerebras

SambaNova

Nscale

fal

Hyperbolic


+ 11


Models 2,256,270


Full-text search


Inference Available


T1 Sort: Trending


 **Tongyi-MAI/Z-Image-Turbo**
Text-to-Image · Updated about 23 hours ago · \pm 111k · \star · \heartsuit 1.91k


 **deepseek-ai/DeepSeek-Math-V2**
Text Generation · \pm 685B · Updated 6 days ago · \pm 7.21k · \star · \heartsuit 616


 **black-forest-labs/FLUX.2-dev**
Image-to-Image · Updated 6 days ago · \pm 185k · \star · \heartsuit 850


 **tencent/HunyuanOCR**
Image-Text-to-Text · \pm 1.0B · Updated 1 day ago · \pm 226k · \star · \heartsuit 614


 **microsoft/Fara-7B**
Image-Text-to-Text · \pm 8B · Updated 2 days ago · \pm 25.3k · \star · \heartsuit 388


 **alibaba-pai/Z-Image-Turbo-Fun-Controlnet-Union**
Updated 1 day ago · \heartsuit 177


 **PrimeIntellect/INTELLECT-3**
Text Generation · \pm 107B · Updated 6 days ago · \pm 2.75k · \star · \heartsuit 170


 **AIDC-AI/Ovis-Image-7B**
Text-to-Image · Updated about 8 hours ago · \pm 1.26k · \star · \heartsuit 139


 **stepfun-ai/Step-Audio-R1**
Audio-Text-to-Text · \pm 33B · Updated 1 day ago · \pm 315 · \star · \heartsuit 110


 **Supertone/supertonic**
Text-to-Speech · Updated 11 days ago · \pm 14.4k · \star · \heartsuit 383


 **salakash/SamKash-Tolstoy**
Text Generation · Updated 6 days ago · \pm 7.88k · \star · \heartsuit 469


 **uncloth/One2One-80B-A3B-TextInstruct-GGUF**


 **deepseek-ai/DeepSeek-V3.2**
Text Generation · \pm 685B · Updated 2 days ago · \pm 5.01k · \star · \heartsuit 626


 **deepseek-ai/DeepSeek-V3.2-Special**
Text Generation · \pm 685B · Updated 2 days ago · \pm 1.87k · \star · \heartsuit 439


 **Comfy-Org/z_image_turbo**
Updated 6 days ago · \pm 1.5M · \star · \heartsuit 339


 **nvidia/Orchestrator-8B**
Text Generation · \pm 8B · Updated 1 day ago · \pm 1.04k · \star · \heartsuit 269


 **facebook/sam3**
Mask Generation · \pm 0.9B · Updated 13 days ago · \pm 327k · \star · \heartsuit 858


 **apple/starflow**
Updated 1 day ago · \star · \heartsuit 172


 **jayn7/Z-Image-Turbo-GGUF**
Text-to-Image · \pm 6B · Updated 5 days ago · \pm 70k · \star · \heartsuit 154

 **mistralai/Mistral-large-3-675B-Instruct-2512**
Updated about 19 hours ago · \pm 112 · \star · \heartsuit 113

 **tencent/HunyuanVideo-1.5**
Text-to-Video · Updated 1 day ago · \pm 2.63k · \star · \heartsuit 791

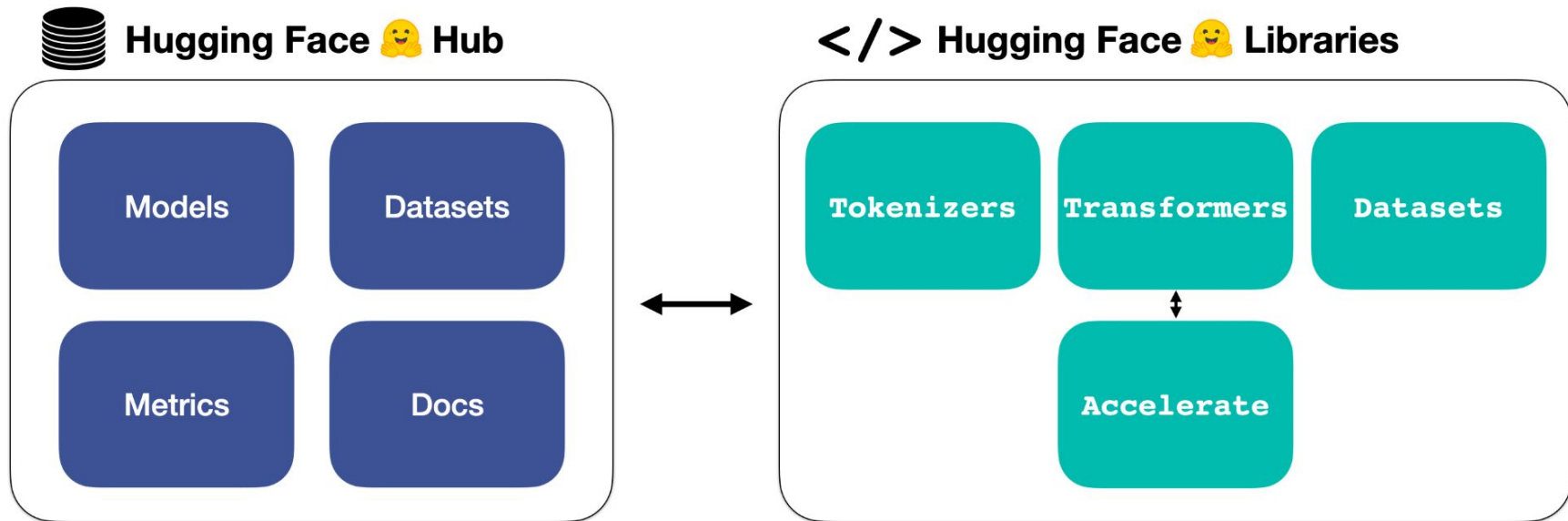
 **TSB/Z-Image-Turbo-FP8**
Text-to-Image · Updated 6 days ago · \pm 110k · \star · \heartsuit 91

 **arcee-ai/Trinity-Mini**
Text Generation · \pm 26B · Updated 1 day ago · \pm 240 · \star · \heartsuit 87

 **mistralai/Mistral-7B-Instruct-2512**

Wulff & Hussain | GSERM Ljubljana 2026

THE HUGGING FACE ECOSYSTEM



HUGGING FACE DOCUMENTATION

Documentations

 Search across all docs

• Hub

Host Git-based models, datasets and Spaces on the Hugging Face Hub.

• Hub Python Library

Client library for the HF Hub: manage repositories from your Python runtime.

• Inference API

Use more than 50k models through our public inference API, with scalability built-in.

• Transformers

State-of-the-art ML for Pytorch, TensorFlow, and JAX.

• Datasets

Access and share datasets for computer vision, audio, and NLP tasks.

• Huggingface.js

A collection of JS libraries to interact with Hugging Face, with TS types included.

• Inference Endpoints

Easily deploy your model to production on dedicated, fully managed infrastructure.

• Diffusers

State-of-the-art diffusion models for image and audio generation in PyTorch.

• Gradio

Build machine learning demos and other web apps, in just a few lines of Python.

• Transformers.js

Community library to run pretrained models from Transformers in your browser.

• PEFT

Parameter efficient finetuning methods for large models

OUR SOFTWARE STACK



+



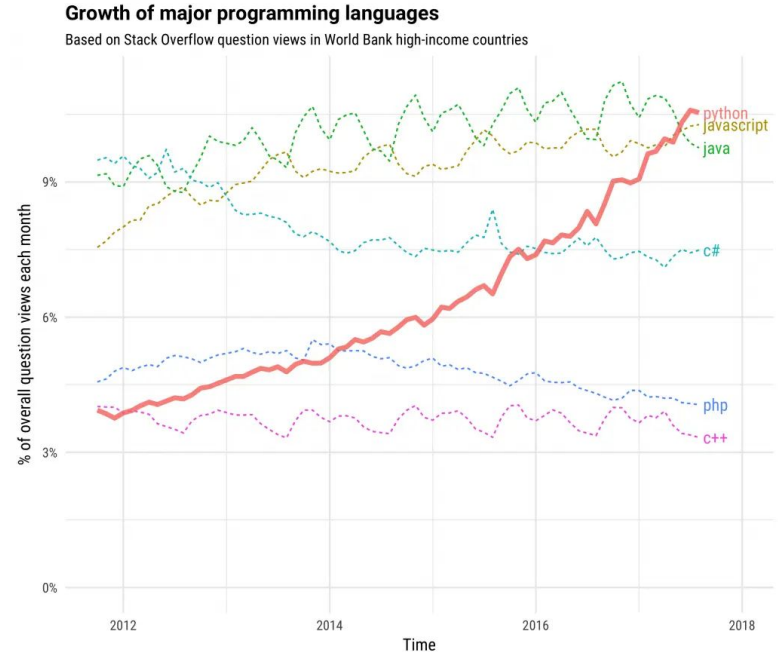
+





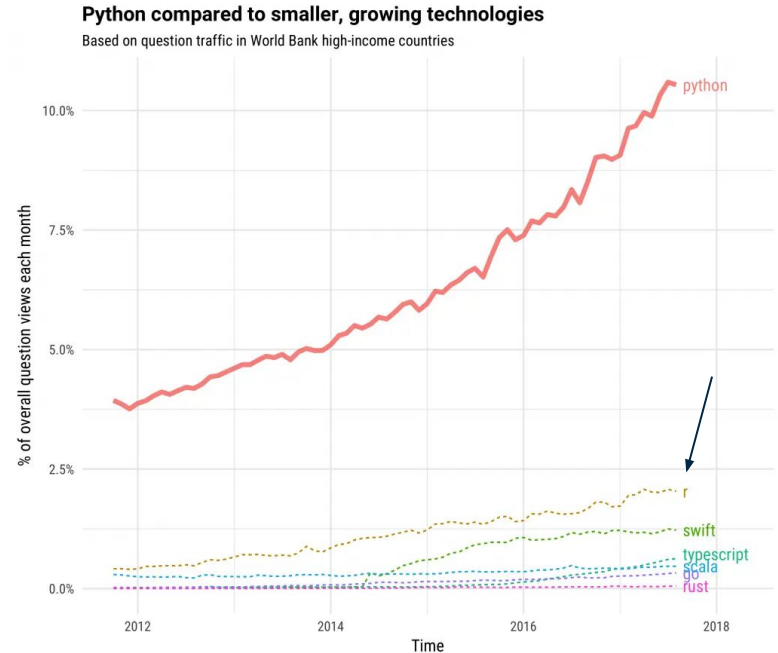


- A **high-level** programming language known for its simplicity and readability.
- Used in various domains such as web development, data analysis, **artificial intelligence**, and scientific computing.





- A **high-level** programming language known for its simplicity and readability.
- Used in various domains such as web development, data analysis, **artificial intelligence**, and scientific computing.



OUR SOFTWARE STACK



+




+



PYTHON  + Google Colab 


PYTHON + Google Colab

 day_1.py nb ☆


File Edit View Insert Runtime Tools Help [All changes saved](#)


+ Code + Text


RAM
Disk


 [6]

```
import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```

 {x}


 **Feature Extraction**

 We will begin by extracting features (numerical representations) from the text data using the `sentence-transformers` package. We will use the following three sentences, stored as a list of strings, as input to the model:

 [7]

```
sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```


We will use the `all-MiniLM-L6-v2` model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.



```
# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

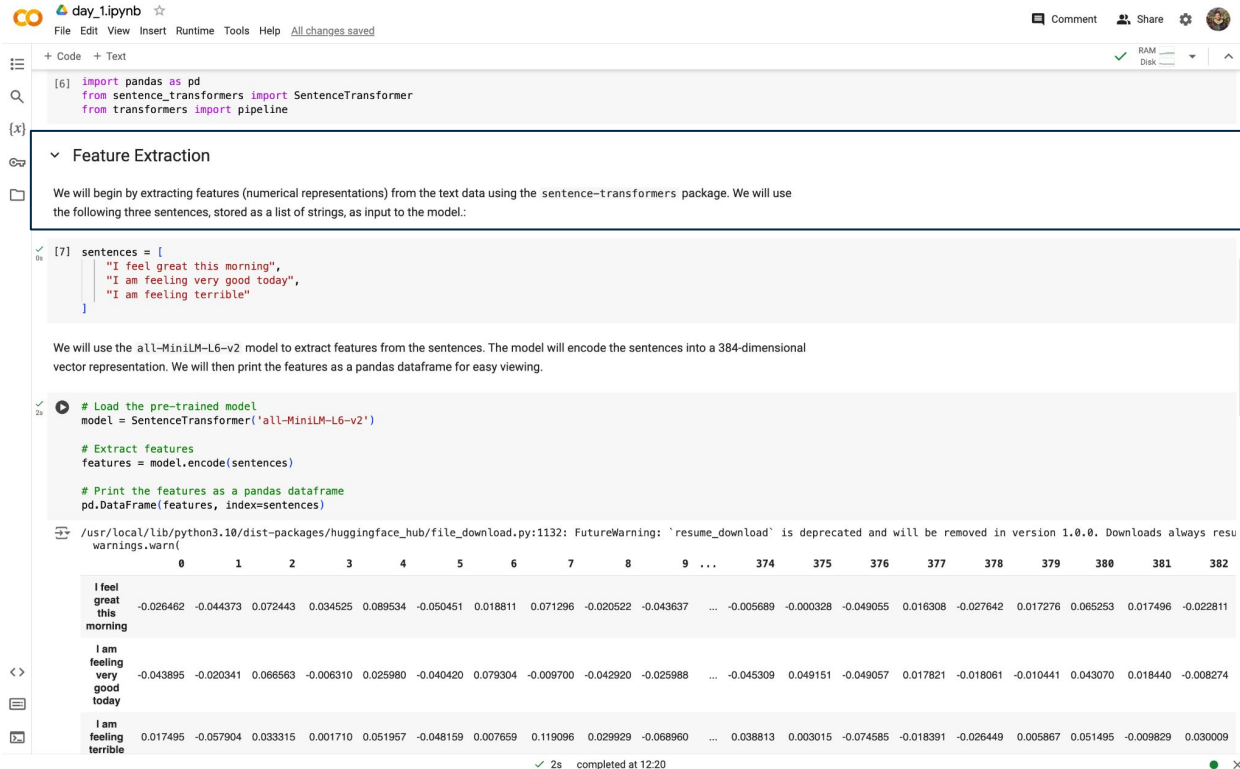
 `/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: 'resume_download' is deprecated and will be removed in version 1.0.0. Downloads always resu`
`warnings.warn(`

	0	1	2	3	4	5	6	7	8	9	...	374	375	376	377	378	379	380	381	382
I feel great this morning	-0.026462	-0.044373	0.072443	0.034525	0.089534	-0.050451	0.018811	0.071296	-0.020522	-0.043637	...	-0.005689	-0.000328	-0.049055	0.016308	-0.027642	0.017276	0.065253	0.017496	-0.022811
I am feeling very good today	-0.043895	-0.020341	0.068563	-0.006310	0.025980	-0.040420	0.079304	-0.009700	-0.042920	-0.025988	...	-0.045309	0.049151	-0.049057	0.017821	-0.018061	-0.010441	0.043070	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057904	0.033315	0.001710	0.051957	-0.048159	0.007659	0.119096	0.029929	-0.068960	...	0.038813	0.003015	-0.074585	-0.018391	-0.026449	0.005867	0.051495	-0.009829	0.030009

2s completed at 12:20

PYTHON + Google Colab

1. Markdown



The screenshot shows a Google Colab notebook titled "day_1.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with icons for adding code or text cells, a search icon, and a RAM/Disk usage indicator showing 0% usage.

The notebook contains two cells:

- Markdown Cell:** Titled "Feature Extraction", it contains the text: "We will begin by extracting features (numerical representations) from the text data using the sentence-transformers package. We will use the following three sentences, stored as a list of strings, as input to the model:"
- Code Cell:** Contains Python code to load a pre-trained model, extract features from three sentences, and print them as a pandas dataframe. The code is as follows:

```
[7] sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]

# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

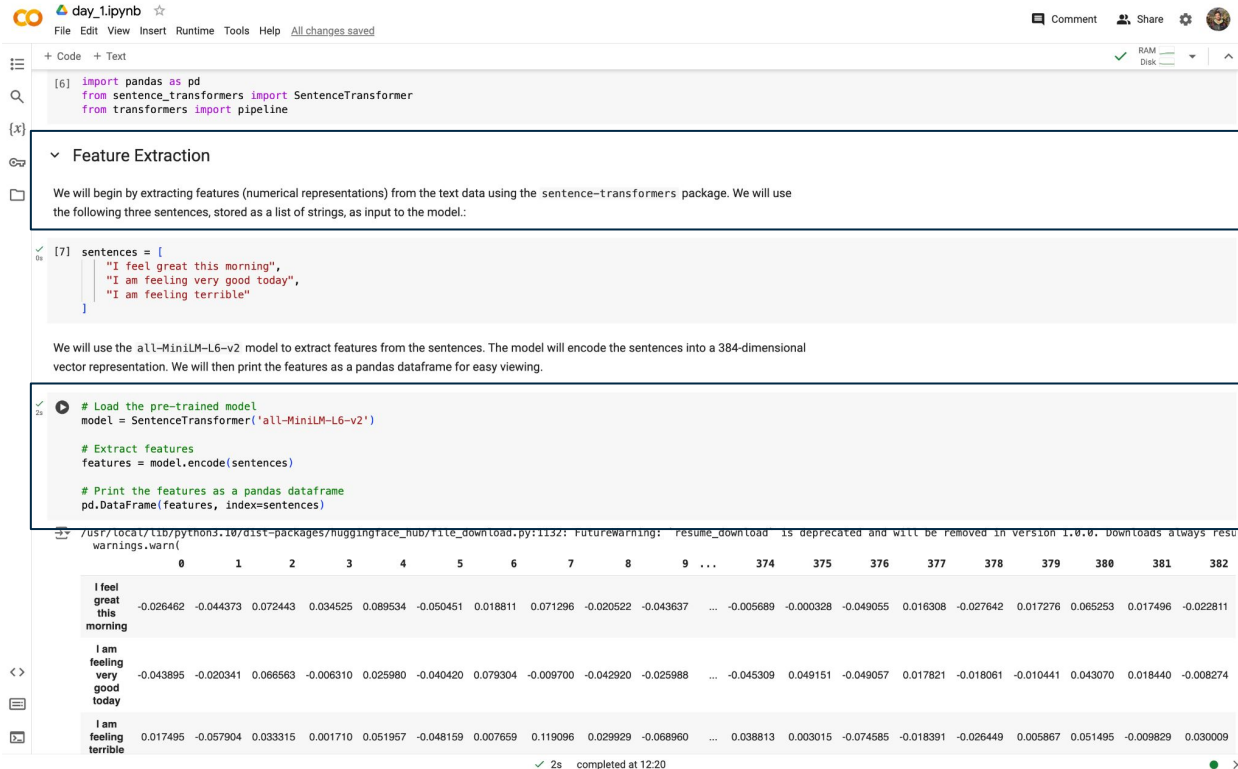
# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

Below the code cell, the output of the pandas dataframe is displayed. It shows a table with three rows (sentences) and 382 columns (features). The first row is "I feel great this morning", the second is "I am feeling very good today", and the third is "I am feeling terrible". The values are numerical representations ranging from approximately -0.049 to 0.065.

	0	1	2	3	4	5	6	7	8	9	...	374	375	376	377	378	379	380	381	382
I feel great this morning	-0.026462	-0.044373	0.072443	0.034525	0.089534	-0.050451	0.018811	0.071296	-0.020522	-0.043637	...	-0.005689	-0.000328	-0.049055	0.016308	-0.027642	0.017276	0.065253	0.017496	-0.022811
I am feeling very good today	-0.043895	-0.020341	0.066563	-0.006310	0.025980	-0.040420	0.079304	-0.009700	-0.042920	-0.025988	...	-0.045309	0.049151	-0.049057	0.017821	-0.018061	-0.010441	0.043070	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057904	0.033315	0.001710	0.051957	-0.048159	0.007659	0.119096	0.029929	-0.068960	...	0.038813	0.003015	-0.074585	-0.018391	-0.026449	0.005867	0.051495	-0.009829	0.030009

PYTHON + Google Colab

1. Markdown



The screenshot shows a Google Colab notebook titled "day_1.ipynb". The notebook contains the following content:

```
[6] import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```

Feature Extraction

We will begin by extracting features (numerical representations) from the text data using the sentence-transformers package. We will use the following three sentences, stored as a list of strings, as input to the model:

```
[7] sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```

We will use the all-MiniLM-L6-v2 model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.

```
26 # Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

Warning: FutureWarning: resume_download is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. To avoid this warning, you can set the variable resume_download=False.

	0	1	2	3	4	5	6	7	8	9	...	374	375	376	377	378	379	380	381	382
I feel great this morning	-0.026462	-0.044373	0.072443	0.034525	0.089534	-0.050451	0.018811	0.071296	-0.020522	-0.043637	...	-0.005689	-0.000328	-0.049055	0.016308	-0.027642	0.017276	0.065253	0.017496	-0.022811
I am feeling very good today	-0.043895	-0.020341	0.066563	-0.006310	0.025980	-0.040420	0.079304	-0.009700	-0.042920	-0.025988	...	-0.045309	0.049151	-0.049057	0.017821	-0.018061	-0.010441	0.043070	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057904	0.033315	0.001710	0.051957	-0.048159	0.007659	0.119096	0.029929	-0.068960	...	0.038813	0.003015	-0.074585	-0.018391	-0.026449	0.005867	0.051495	-0.009829	0.030009

2s completed at 12:20

2. Code

PYTHON + Google Colab

1. Markdown

Feature Extraction

We will begin by extracting features (numerical representations) from the text data using the `sentence-transformers` package. We will use the following three sentences, stored as a list of strings, as input to the model:

```
[7] sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```

We will use the `all-MiniLM-L6-v2` model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.

2. Code

```
# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

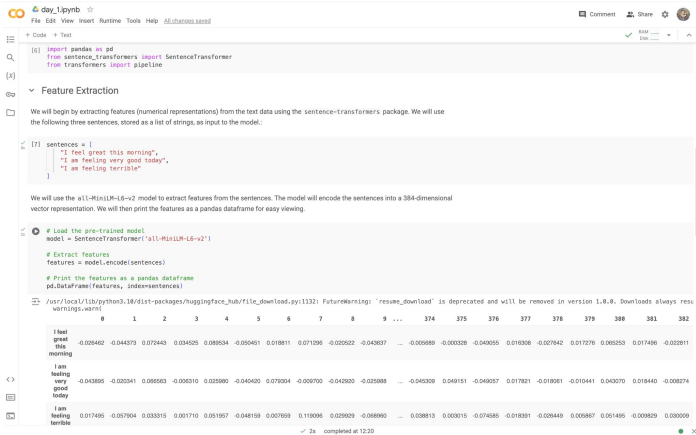
3. Printouts

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: resume_download is deprecated and will be removed in version 1.0.0. Downloads always resum...
```

	0	1	2	3	4	5	6	7	8	9	...	374	375	376	377	378	379	380	381	382
I feel great this morning	-0.026462	-0.044373	0.072443	0.034525	0.089534	-0.050451	0.018811	0.071296	-0.020522	-0.043637	...	-0.005689	-0.000328	-0.049055	0.016308	-0.027642	0.017276	0.065253	0.017496	-0.022811
I am feeling very good today	-0.043895	-0.020341	0.066563	-0.006310	0.025980	-0.040420	0.079304	-0.009700	-0.042920	-0.025988	...	-0.045309	0.049151	-0.049057	0.017821	-0.018061	-0.010441	0.043070	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057904	0.033315	0.001710	0.051957	-0.048159	0.007659	0.119096	0.029929	-0.068960	...	0.038813	0.003015	-0.074585	-0.018391	-0.026449	0.005867	0.051495	-0.009829	0.030009

✓ 2s completed at 12:20

PYTHON + Google Colab



```
day_1pythb
File Edit View Insert Runtime Tools Help 88.0x600x1800
+ Code + Text
[5] import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline

Feature Extraction

We will begin by extracting features (numerical representations) from the text data using the sentence-transformers package. We will use the following three sentences, stored as a list of strings, as input to the model.

[7] sentences = [
    "I feel great this morning",
    "I as feeling very good today",
    "I as feeling terrible"
]

We will use the all-MiniLM-L6-v2 model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.

# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

`Warning: 'resume_download' is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. To avoid this message, please use `force_download=True`.`

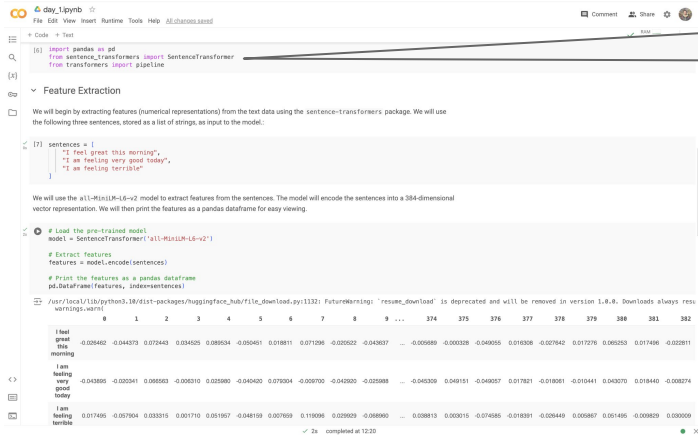
	0	1	2	3	4	5	6	7	8	...	374	375	376	377	378	379	380	381	382	
I feel great this morning	-0.026462	-0.044373	0.072943	0.034026	0.089534	-0.050461	0.018811	0.071296	-0.020522	-0.043037	...	-0.005889	-0.000328	-0.049005	0.016308	-0.027642	0.017276	0.060253	0.017406	-0.020811
I am feeling very good today	-0.043886	-0.020541	0.080903	-0.006210	0.025890	-0.040403	0.077604	-0.008700	-0.040262	-0.023588	...	-0.040330	0.068151	-0.048057	0.017821	-0.018061	-0.010441	0.040270	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057604	0.033315	0.001710	0.051867	-0.048169	0.007769	0.119306	0.023929	-0.088900	...	0.008813	0.003015	-0.074585	-0.018391	-0.026410	0.005867	-0.051405	-0.006829	0.030008

2s completed at 12:20

PYTHON + Google Colab

package imports

```
import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```



```
6 day_1.pytnb
File Edit View Insert Runtime Tools Help 88.48000.00000
+ Code + Text
[5] import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline

Feature Extraction
We will begin by extracting features (numerical representations) from the text data using the sentence-transformers package. We will use the following three sentences, stored as a list of strings, as input to the model.

[7] sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]

We will use the all-MiniLM-L6-v2 model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.

# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

FutureWarning: 'resume_download' is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. To avoid this warning, you can pass the argument 'resume_download=False'.

	0	1	2	3	4	5	6	7	8	9	...	374	375	376	377	378	379	380	381	382
I feel great this morning	-0.026462	-0.044373	0.072943	0.034026	0.089504	-0.050461	0.018811	0.071296	-0.020522	-0.043037	...	-0.005889	-0.000328	-0.049005	0.016306	-0.027642	0.017276	0.060253	0.017406	-0.020811
I am feeling very good today	-0.043886	-0.020541	0.080903	-0.006210	0.025890	-0.040403	0.076004	-0.008700	-0.040262	-0.020588	...	-0.046309	0.068151	-0.048057	0.017821	-0.018061	-0.010441	0.040270	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057604	0.033315	0.001710	0.051857	-0.048159	0.007659	0.119006	0.022929	-0.048900	...	0.008813	0.003015	-0.074585	-0.018391	-0.026440	0.005867	-0.051405	-0.008629	0.030009

2s completed at 12:20

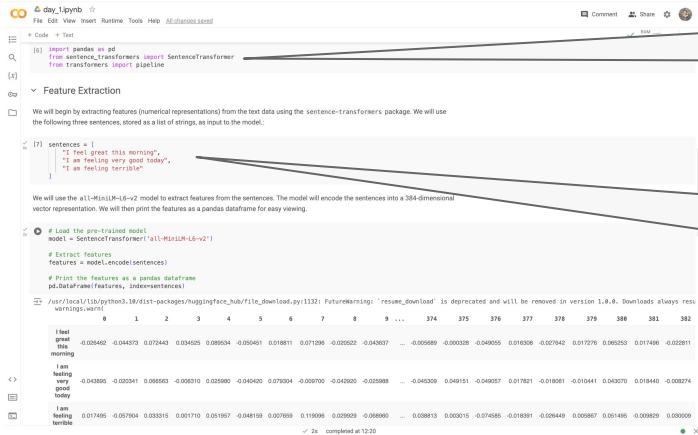
PYTHON + Google Colab

package imports

```
import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```

variable assignment, lists, strings

```
sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```



```
6 day_1.pytnb
File Edit View Insert Runtime Tools Help 88.48x100x1000

+ Code + Text

[5] import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline

Feature Extraction

We will begin by extracting features (numerical representations) from the text data using the sentence-transformers package. We will use the following three sentences, stored as a list of strings, as input to the model.

[7] sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]

We will use the all-MiniLM-L6-v2 model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.

# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

	0	1	2	3	4	5	6	7	8	...	374	375	376	377	378	379	380	381	382	
I feel great this morning	-0.026462	-0.044373	0.072943	0.034326	0.089534	-0.050461	0.018811	0.071296	-0.022522	-0.043037	...	-0.005889	-0.000328	-0.049005	0.016306	-0.027642	0.012726	0.060253	0.017406	-0.020811
I am feeling very good today	-0.043886	-0.020341	0.088953	-0.006210	0.025890	-0.040430	0.076204	-0.028700	-0.042620	-0.023588	...	-0.043339	0.068151	-0.048037	0.017821	-0.018061	-0.010441	0.040270	0.018440	-0.008274
I am feeling terrible	0.017495	-0.057604	0.033315	0.061710	0.051867	-0.048159	0.007659	0.119306	0.022929	-0.088900	...	0.028813	0.003015	-0.074585	-0.018331	-0.026440	0.005867	0.051405	-0.008629	0.030008

2x completed at 12:20

PYTHON + Google Colab

package imports

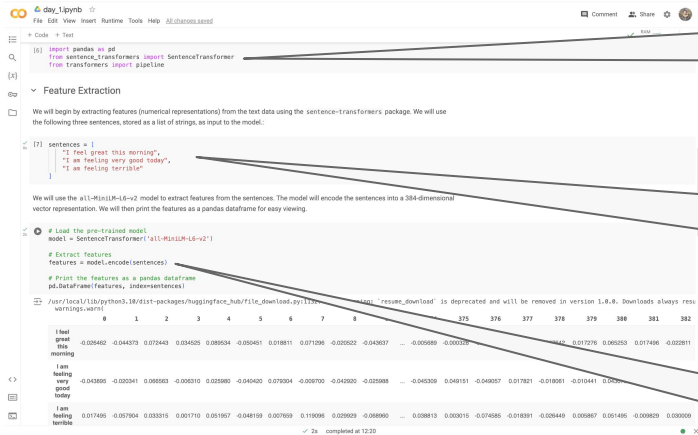
```
import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```

variable assignment, lists, strings

```
sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```

dot notation, methods, attributes

```
# Extract features
features = model.encode(sentences)
```



The screenshot shows a Google Colab notebook with the following content:

```
File Edit View Insert Runtime Tools Help
+ Code + Text
[1] import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```

Feature Extraction

We will begin by extracting features (numerical representations) from the text data using the sentence-transformers package. We will use the following three sentences, stored as a list of strings, as input to the model:

```
[7] sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```

We will use the all-MiniLM-L6-v2 model to extract features from the sentences. The model will encode the sentences into a 384-dimensional vector representation. We will then print the features as a pandas dataframe for easy viewing.

```
# Load the pre-trained model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Extract features
features = model.encode(sentences)

# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

The output shows a pandas DataFrame with 3 rows and 384 columns. The first row corresponds to "I feel great this morning", the second to "I am feeling very good today", and the third to "I am feeling terrible". The values are numerical representations of the sentences.

PYTHON + Google Colab

package imports

```
import pandas as pd
from sentence_transformers import SentenceTransformer
from transformers import pipeline
```

variable assignment, lists, strings

```
sentences = [
    "I feel great this morning",
    "I am feeling very good today",
    "I am feeling terrible"
]
```

dot notation, methods, attributes

```
# Extract features
features = model.encode(sentences)
```

printing

```
# Print the features as a pandas dataframe
pd.DataFrame(features, index=sentences)
```

	0	1	2	3	4	5	6
I feel great this morning	-0.026462	-0.044373	0.072443	0.034525	0.089534	-0.050451	0.018811
I am feeling very good today	-0.043895	-0.020341	0.066563	-0.006310	0.025980	-0.040420	0.079304
I am feeling terrible	0.017495	-0.079034	0.033303	0.048159	0.007659	0.119006	0.022929

OUR SOFTWARE STACK



+



+



Time to install our stack...



MAX PLANCK INSTITUTE
FOR HUMAN DEVELOPMENT



LLM4BeSci_Ljubljana2026 Public

Pin Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file t Add file <> Code

Zak-Hussain slide upload note a18ea0b · 9 hours ago 26 Commits		
day_1	use latest day_1.ipynb	last week
day_2	typo	10 hours ago
day_3	typo	9 hours ago
day_4	typo	9 hours ago
day_5	update with newer justaism paper version	9 hours ago
.gitignore	Initial commit	last week
LICENSE.txt	add LICENSE.txt	last week
README.md	slide upload note	9 hours ago
cover_gserm.png	Add files via upload	last week
notes.txt	add notes.txt	last week

README License

LLM4BeSci at GSERM, Ljubljana 2026



About

The course introduces the use of open large language models (LLMs) from the Hugging Face ecosystem for research in the behavioral and social sciences.

- Readme
- View license
- Activity
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 2

- Zak-Hussain Zak Hussain
- dwulff Dirk Wulff

Languages