

Hugging Face Models

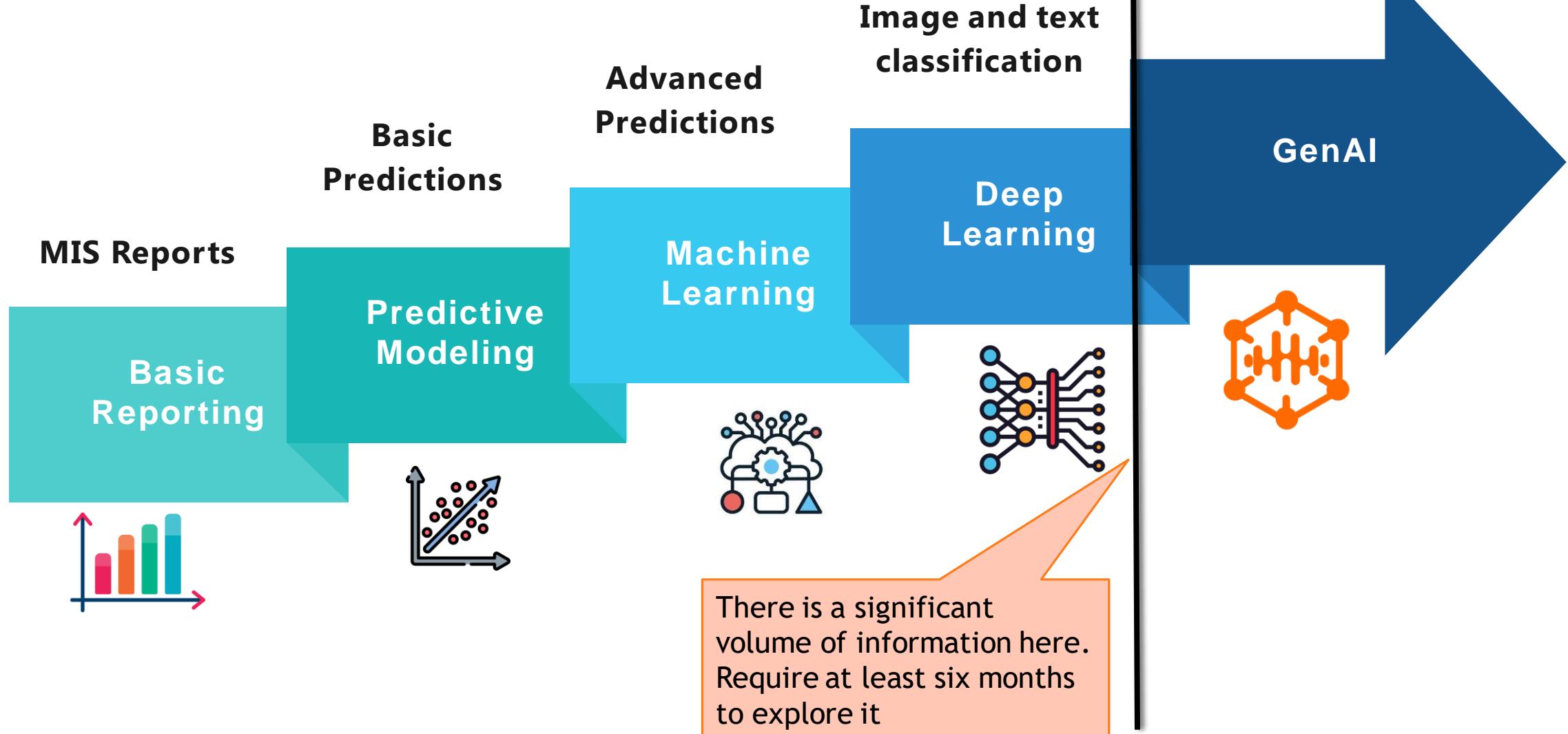
Venkata Reddy AI Classes

<https://www.youtube.com/@VenkataReddyAIClasses/playlists>

Contents

- Open source LLMs
- Model Hub on Hugging face
- Model from Hugging face
- pipeline() function
- Document Classification model
- Q&A Model
- NER Model
- Saving the Model on HuggingFace hub
- Final app - Bank Customers Complaints categorization App

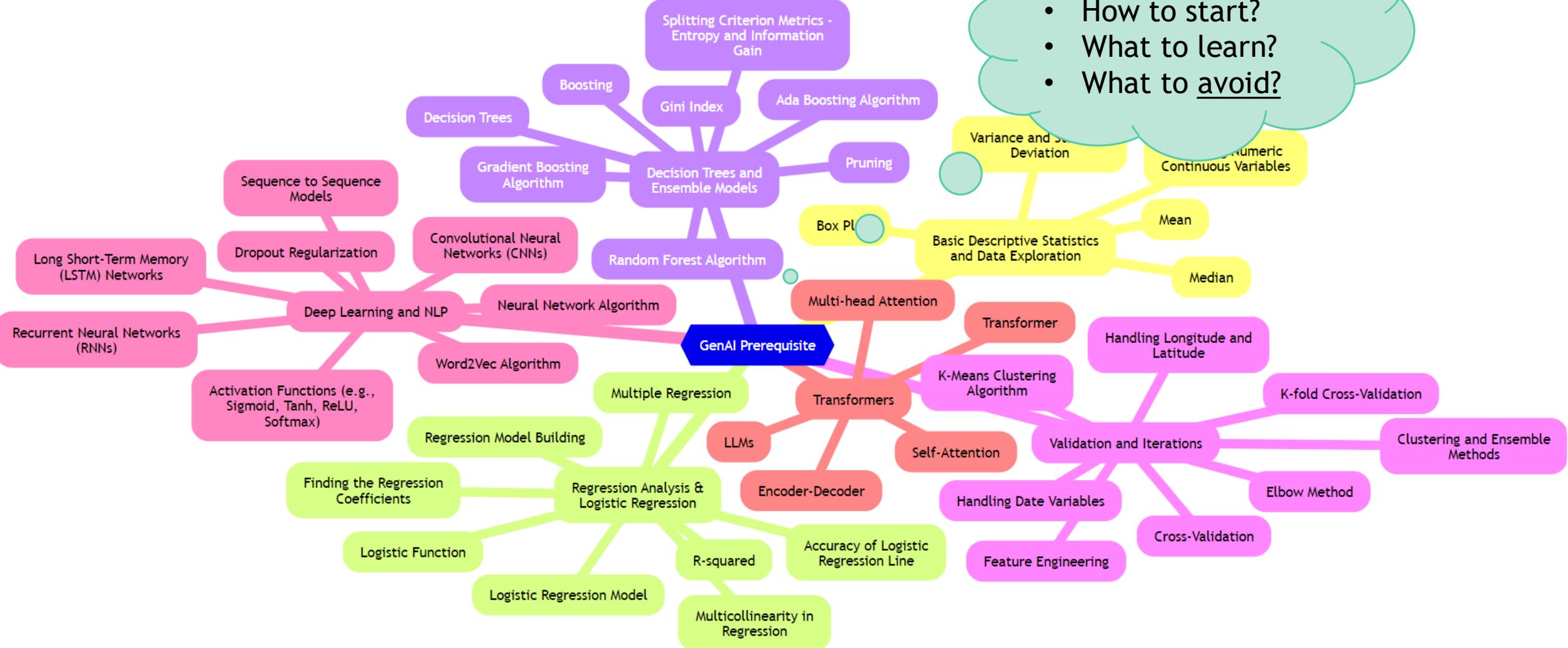
GenAI Pre-requisites



GenAI Pre-requisites



GenAI Pre-requisites

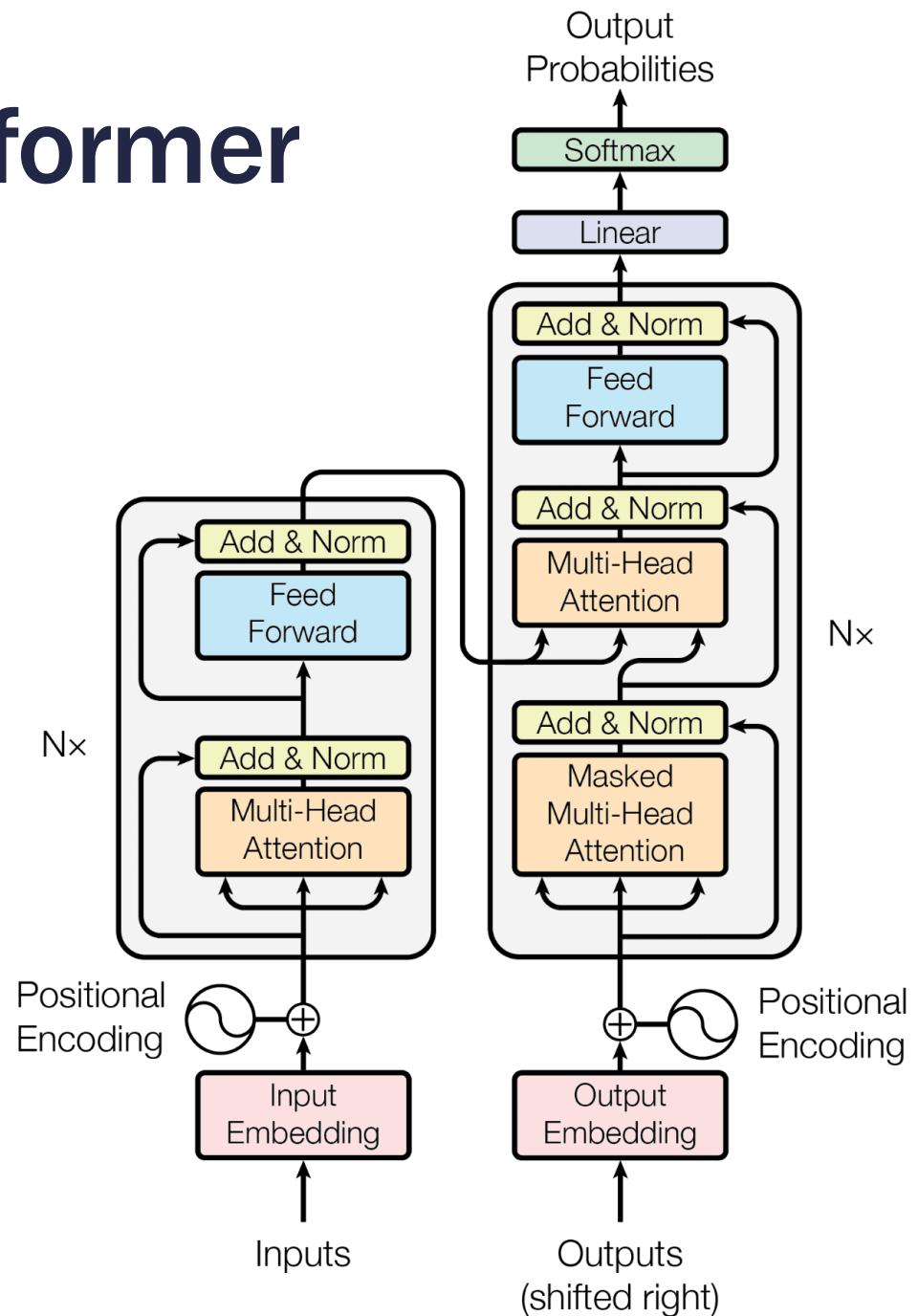


GenAI Pre-requisites



Almost every LLM is a Transformer

- Transformer is an advanced Deep Neural Network model
- Transformers revolutionize NLP, forming the basis for modern LLMs.
- Designed to process and understand sequential data like text.
- Capture relationships between words regardless of their position.
- Attention mechanism allows models to focus on all input parts simultaneously.



What are the opensource LLMs?

- What are the opensource LLMs?
- How to access them using APIs?
- Where to find them?
- Where can we find the documentation ?
- Is there any repository where I can find all the LLMs ? - Yes
- **Hugging Face**

Hugging Face

- If you are working with language models, you have to be comfortable with Hugging Face hub
- Hugging face is the repository for several pre-trained ML, NLP, Vision based models and datasets.
- You can access various models and datasets.
- You can also upload your own model into hugging face platform

Repository of all ML and AI models

- Designed for sharing, collaborating, and managing machine learning models, datasets, etc.
- Trying to democratize machine learning, especially in NLP and computer vision. Make the models available for everyone
- Contains good documentation and example codes
- **You must learn** how to use Hugging Face models before jumping on to LLMs



Hugging Face

Search models, datasets, users...

Models

Datasets

Spaces

Posts

Docs

Pricing

NEW Create Assistants in HuggingChat



The AI community building the future.

The platform where the machine learning community collaborates on models, datasets, and applications.

Tasks Libraries Datasets Languages Licenses Other

Filter Tasks by name

Multimodal

Text-to-Image Image-to-Text
Text-to-Video Visual Question Answering
Document Question Answering Graph Machine Learning

Computer Vision

Depth Estimation Image Classification
Object Detection Image Segmentation
Image-to-Image Unconditional Image Generation
Video Classification Zero-Shot Image Classification

Natural Language Processing

Text Classification Token Classification
Table Question Answering Question Answering
Zero-Shot Classification Translation
Summarization Conversational
Text Generation Text2Text Generation
Sentence Similarity

Audio

Text-to-Speech Automatic Speech Recognition
Audio-to-Audio Audio Classification

Models 469,541

meta-llama/Llama-2-
Text Generation • Updated

stabilityai/stable-diffusion
Updated 6 days ago • 2.01k

openchat/openchat
Text Generation • Updated 2 days ago

11lyasviel/ControlNet-v1.0
Updated Apr 26 • 1.87k

cerspense/zeroscope_v2_X
Updated 3 days ago • 2.66k • 33

meta-llama/Llama-2-13b
Text Generation • Updated 4 days ago

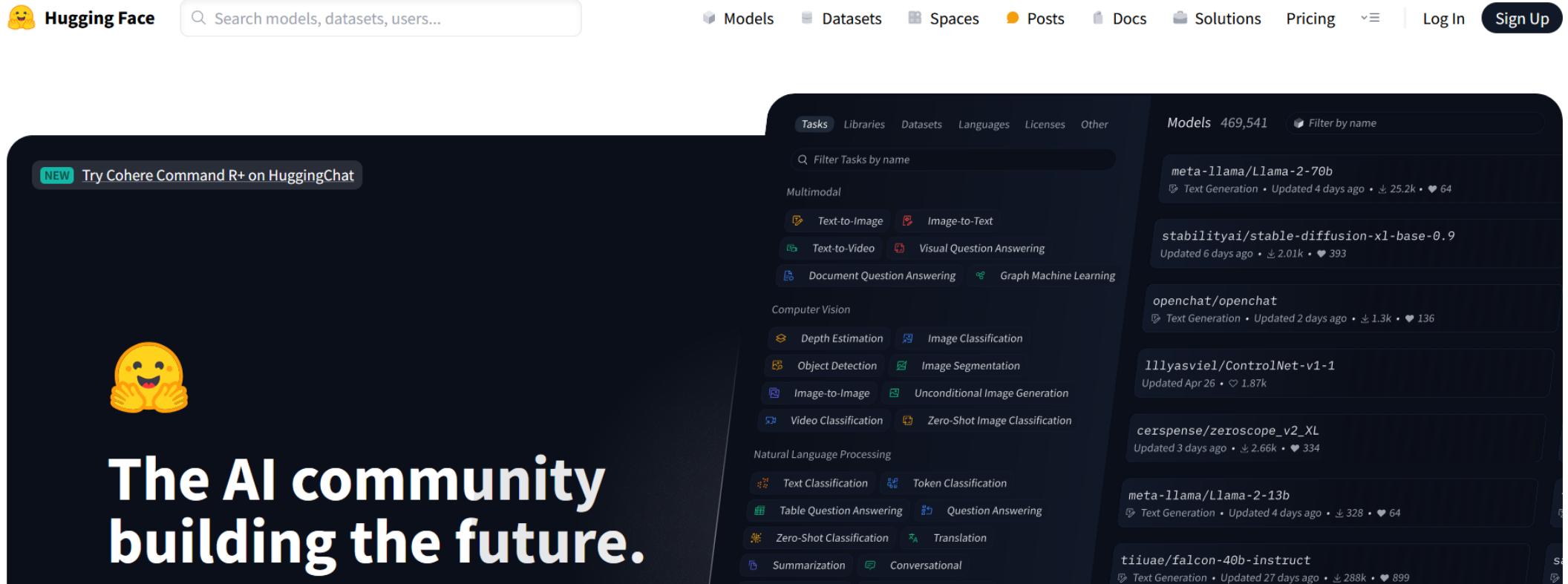
tiiuae/falcon-40b-instruct
Text Generation • Updated 27 days ago

WizardLM/WizardCoder-15B-V1
Text Generation • Updated 3 days ago

CompVis/stable-diffusion-v1-4
Text-to-Image • Updated about 17 hours ago

Sign-up

Signup and create a login-id



The screenshot shows the Hugging Face homepage and a modal window for sign-up.

Header:

- Hugging Face logo
- Search bar: Search models, datasets, users...
- Navigation menu: Models, Datasets, Spaces, Posts, Docs, Solutions, Pricing, Log In, Sign Up

Modal Window (Top Right):

Sign up and create a login-id

Homepage Content:

- A yellow emoji of a smiling face with hands clasped.
- Text: Try Cohere Command R+ on HuggingChat
- Section: The AI community building the future.

Left Sidebar (Dark Mode):

- Tasks tab (selected)
- Libraries, Datasets, Languages, Licenses, Other
- Filter Tasks by name
- Multimodal tasks: Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering, Document Question Answering, Graph Machine Learning
- Computer Vision tasks: Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, Zero-Shot Image Classification
- Natural Language Processing tasks: Text Classification, Token Classification, Table Question Answering, Question Answering, Zero-Shot Classification, Translation, Summarization, Conversational

Right Panel (Dark Mode):

Models 469,541 Filter by name

- meta-llama/Llama-2-70b (Text Generation, Updated 4 days ago, 25.2k, 64)
- stabilityai/stable-diffusion-xl-base-0.9 (Text Generation, Updated 6 days ago, 2.01k, 393)
- openchat/openchat (Text Generation, Updated 2 days ago, 1.3k, 136)
- llyasviel/ControlNet-v1-1 (Updated Apr 26, 1.87k)
- cerspense/zeroscope_v2_XL (Updated 3 days ago, 2.66k, 334)
- meta-llama/Llama-2-13b (Text Generation, Updated 4 days ago, 328, 64)
- tiiuae/falcon-40b-instruct (Text Generation, Updated 27 days ago, 288k, 899)

Model Hub contains a huge collection of pre-trained models



Hugging Face

Models 547,770 new Full-text search

Tasks Libraries Datasets Languages Licenses Other

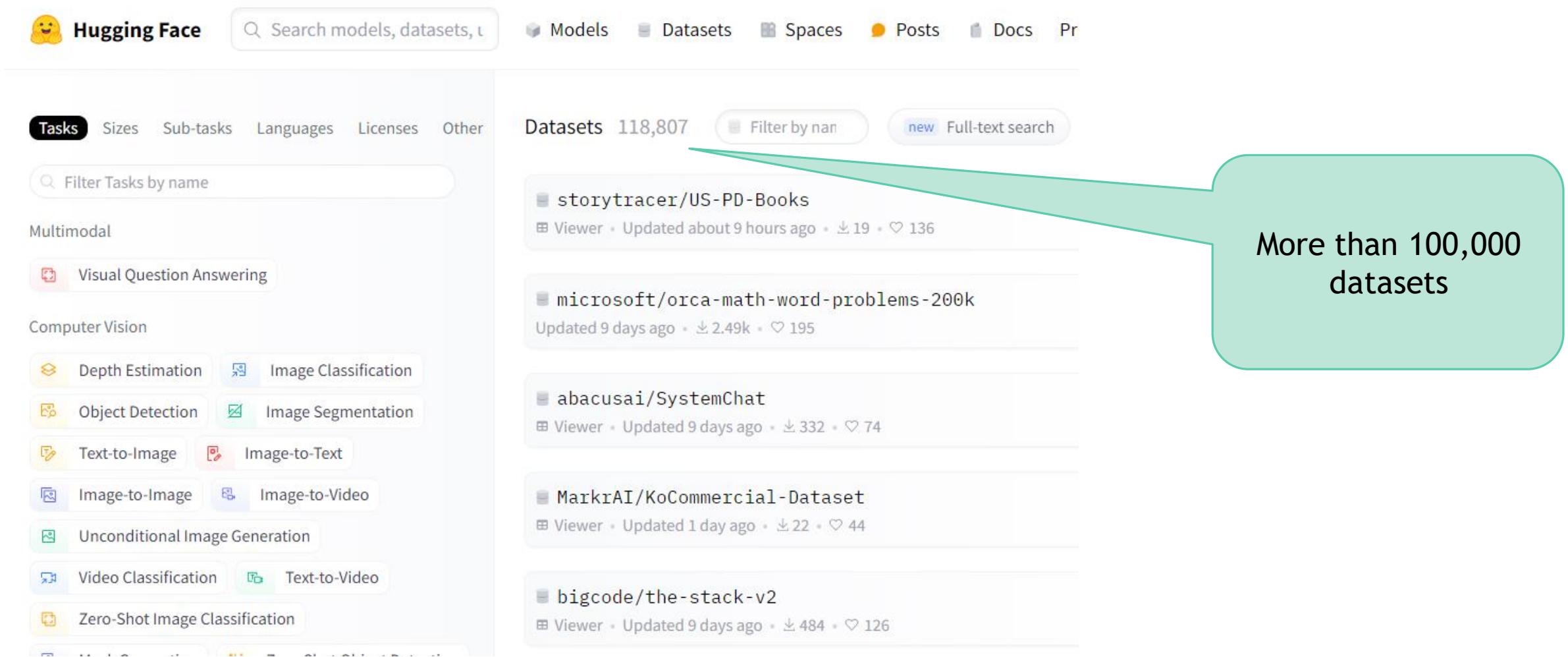
Multimodal
 Image-Text-to-Text Visual Question Answering Document Question Answering

Computer Vision
 Depth Estimation Image Classification
 Object Detection Image Segmentation
 Text-to-Image Image-to-Text
 Image-to-Image Image-to-Video
 Unconditional Image Generation
 Video Classification Text-to-Video
 Zero-Shot Image Classification

More than 500,000 pretrained models

- CohereForAI/c4ai-command-r-v01**
Text Generation • Updated about 9 hours ago • ↓ 1.14k • 421
- google/gemma-7b**
Text Generation • Updated 14 days ago • ↓ 243k • 2.23k
- NousResearch/Genstruct-7B**
Text Generation • Updated 6 days ago • ↓ 1.15k • 236
- ByteDance/SDXL-Lightning**
Text-to-Image • Updated 10 days ago • ↓ 454k • 1.23k
- 01-ai/Yi-9B**
Text Generation • Updated 5 days ago • ↓ 2.62k • 147
- stabilityai/TripoSR**
Image-to-3D • Updated 8 days ago • ↓ 20.1k • 216

A huge collection of datasets also



The screenshot shows the Hugging Face dataset catalog interface. At the top, there's a navigation bar with a yellow emoji icon, the text "Hugging Face", a search bar ("Search models, datasets, ..."), and links for "Models", "Datasets", "Spaces", "Posts", "Docs", and "Pr". Below the search bar, there's a dropdown menu with "Tasks" selected, followed by "Sizes", "Sub-tasks", "Languages", "Licenses", and "Other". A search input field says "Filter Tasks by name" with a placeholder "Multimodal". Under "Multimodal", there are categories: "Visual Question Answering" (selected), "Computer Vision", "Depth Estimation", "Image Classification", "Object Detection", "Image Segmentation", "Text-to-Image", "Image-to-Text", "Image-to-Image", "Image-to-Video", "Unconditional Image Generation", "Video Classification", "Text-to-Video", and "Zero-Shot Image Classification". On the right, the main content area displays "Datasets 118,807". It includes a "Filter by name" button and a "Full-text search" button. A large green callout bubble points to the dataset count with the text "More than 100,000 datasets". Below the count, several dataset cards are listed:

- storytracer/US-PD-Books: Updated about 9 hours ago, 19 views, 136 likes
- microsoft/orca-math-word-problems-200k: Updated 9 days ago, 2.49k views, 195 likes
- abacusai/SystemChat: Updated 9 days ago, 332 views, 74 likes
- MarkrAI/KoCommercial-Dataset: Updated 1 day ago, 22 views, 44 likes
- bigcode/the-stack-v2: Updated 9 days ago, 484 views, 126 likes

Model Hub on Hugging Face

Models 2,735

 meta

Full-text search

 [meta-llama/Meta-Llama-3-8B](#)

 Text Generation • Updated May 13 • ↓ 1.12M • ❤ 4.84k

 [meta-llama/Meta-Llama-3-8B-Instruct](#)

 Text Generation • Updated 23 days ago • ↓ 2.71M • ❤ 2.72k

 [meta-llama/Meta-Llama-3-70B-Instruct](#)

 Text Generation • Updated 23 days ago • ↓ 497k • ❤ 1.19k

 [eastwind/meta-chameleon-7b](#)

Updated 2 days ago • ❤ 23

 [meta-llama/Llama-2-7b-chat-hf](#)

 Text Generation • Updated Apr 17 • ↓ 1.03M • ❤ 3.63k

 [meta-llama/Llama-2-7b-hf](#)

Models 651

 openai

Full-text search

 [openai/whisper-large-v3](#)

 Automatic Speech Recognition • Updated 11 days ago • ↓ 3.4M • ❤ 2.7

 [openai-community/gpt2](#)

 Text Generation • Updated Feb 19 • ↓ 7.56M • ❤ 2.03k

 [openai/clip-vit-large-patch14](#)

 Zero-Shot Image Classification • Updated Sep 15, 2023 • ↓ 37.3M • ❤ 1

 [openai/clip-vit-base-patch32](#)

 Zero-Shot Image Classification • Updated Feb 29 • ↓ 17.4M • ❤ 400

 [openai/clip-vit-large-patch14-336](#)

 Zero-Shot Image Classification • Updated Oct 4, 2022 • ↓ 6.74M • ❤ 14

 [openai/whisper-large-v2](#)

Model Hub on Hugging Face

Models 2,244

google

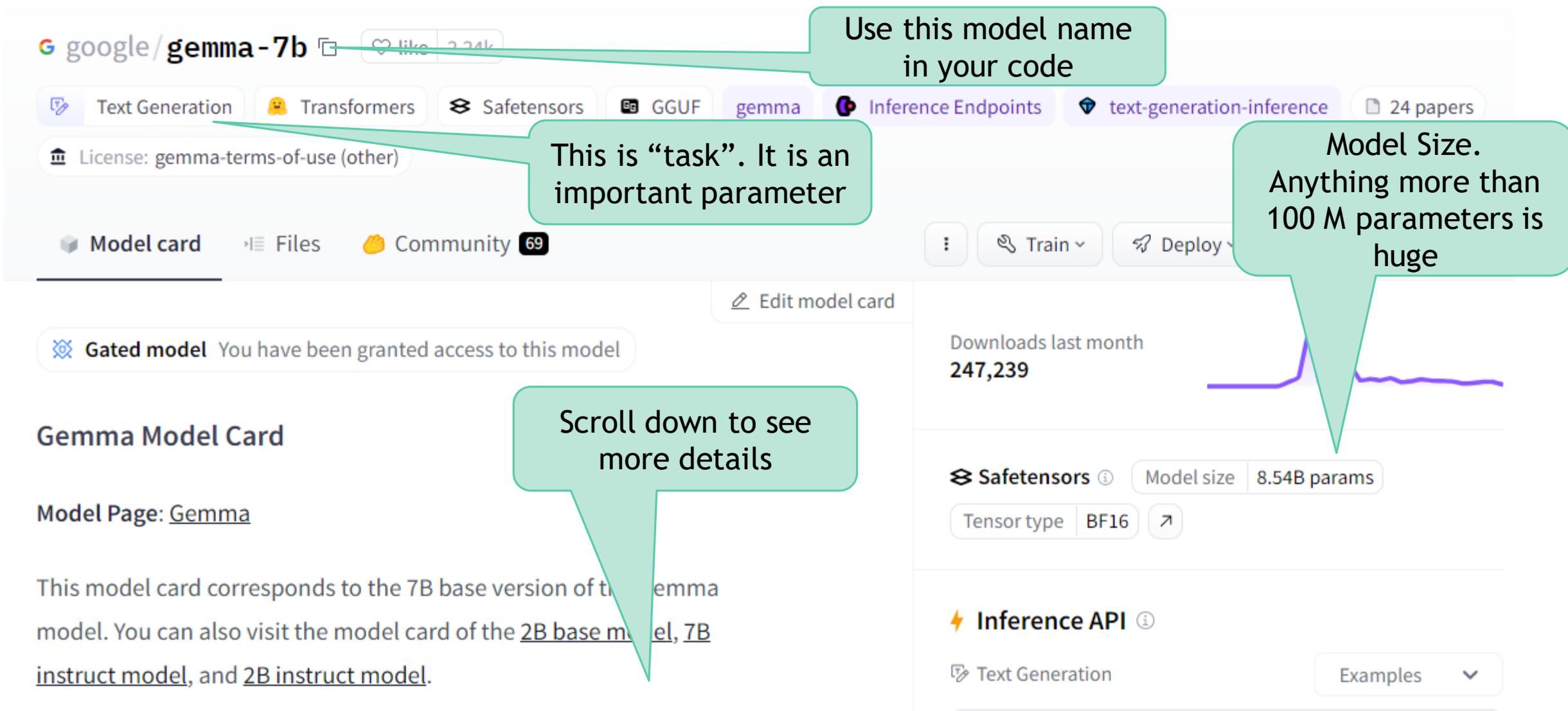
- google/bert/bert-base-uncased**
Fill-Mask • Updated Feb 19 • ↓ 67.4M • ❤ 1.62k
- google/gemma-7b**
Text Generation • Updated May 1 • ↓ 193k • ❤ 2.96k
- google/recurrentgemma-9b**
Text Generation • Updated 10 days ago • ↓ 436 • ❤ 39
- google/flan-t5-base**
Text2Text Generation • Updated Jul 17, 2023 • ↓ 1.54M • ❤ 722
- google/gemma-2b**
Text Generation • Updated Apr 16 • ↓ 381k • ❤ 767
- google/gemma-2b-bit**

Models 702

microsoft

- microsoft/Florence-2-large**
Image-to-Text • Updated 2 days ago • ↓ 3.22k • ❤ 282
- microsoft/Florence-2-large-ft**
Image-to-Text • Updated 2 days ago • ↓ 1.73k • ❤ 125
- microsoft/Florence-2-base**
Image-to-Text • Updated 2 days ago • ↓ 738 • ❤ 50
- microsoft/Florence-2-base-ft**
Image-to-Text • Updated 2 days ago • ↓ 1.5k • ❤ 41
- microsoft/Phi-3-vision-128k-instruct**
Text Generation • Updated 10 days ago • ↓ 235k • ❤ 741
- microsoft/Phi-3-mini-4k-instruct**

Sample Model from Hugging face



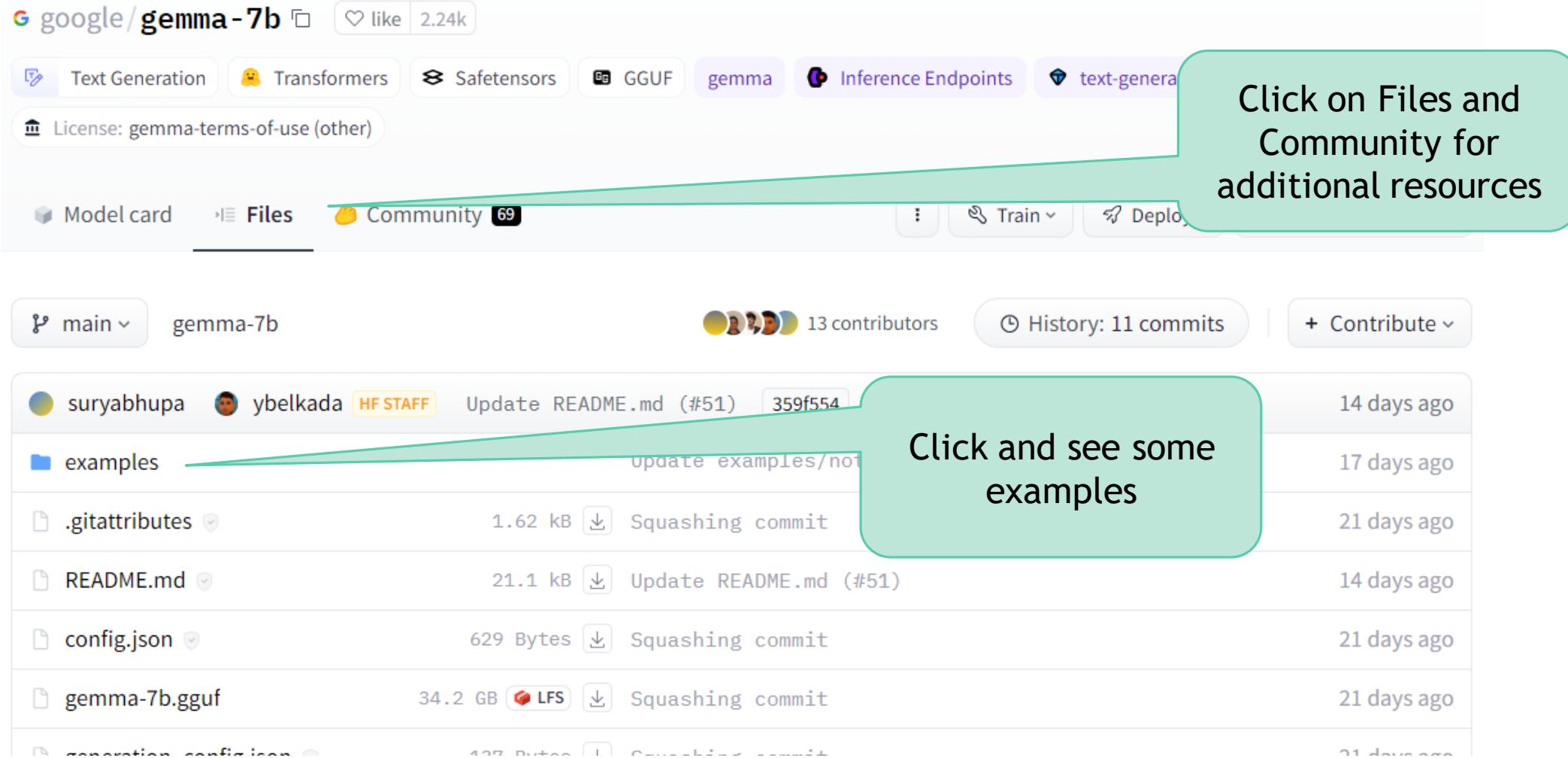
This screenshot shows the Hugging Face model card for the `gemma-7b` model. The card includes sections for tasks, model card, files, community, and a gated model access notice.

- Tasks:** Text Generation, Transformers, Safetensors, GGUF, gemma, Inference Endpoints, text-generation-inference, 24 papers.
- Model card:** A link to the detailed model card.
- Files:** A link to the file section.
- Community:** A link to the community section with 69 posts.
- Gated model:** A notice stating "You have been granted access to this model".
- Gemma Model Card:** A link to the Gemma Model Card.
- Model Page:** A link to the Gemma model page.
- Description:** Text explaining the model corresponds to the 7B base version of the Gemma model, with links to 2B base model, 7B instruct model, and 2B instruct model.
- Downloads last month:** 247,239
- Safetensors:** Model size: 8.54B params.
- Tensor type:** BF16.
- Inference API:** An orange button for interacting with the model.
- Text Generation:** A link to text generation examples.
- Examples:** A dropdown menu for more options.

Annotations on the screenshot:

- A callout bubble points to the `gemma` tag in the tasks section with the text: "Use this model name in your code".
- A callout bubble points to the "Task" tab with the text: "This is ‘task’. It is an important parameter".
- A callout bubble points to the "Downloads last month" chart with the text: "Model Size. Anything more than 100 M parameters is huge".
- A callout bubble points to the "Scroll down to see more details" text with the text: "Scroll down to see more details".

Model from Hugging face



The screenshot shows the Hugging Face model card for the "gemma-7b" model. At the top, there's a navigation bar with links for Text Generation, Transformers, Safetensors, GGUF, gemma, Inference Endpoints, and text-generation. Below that is a license section for "gemma-terms-of-use (other)". The main navigation tabs are Model card, Files (which is selected), and Community (with 69 contributions). To the right of the tabs are buttons for Train and Deploy.

A callout bubble points to the Community tab with the text: "Click on Files and Community for additional resources".

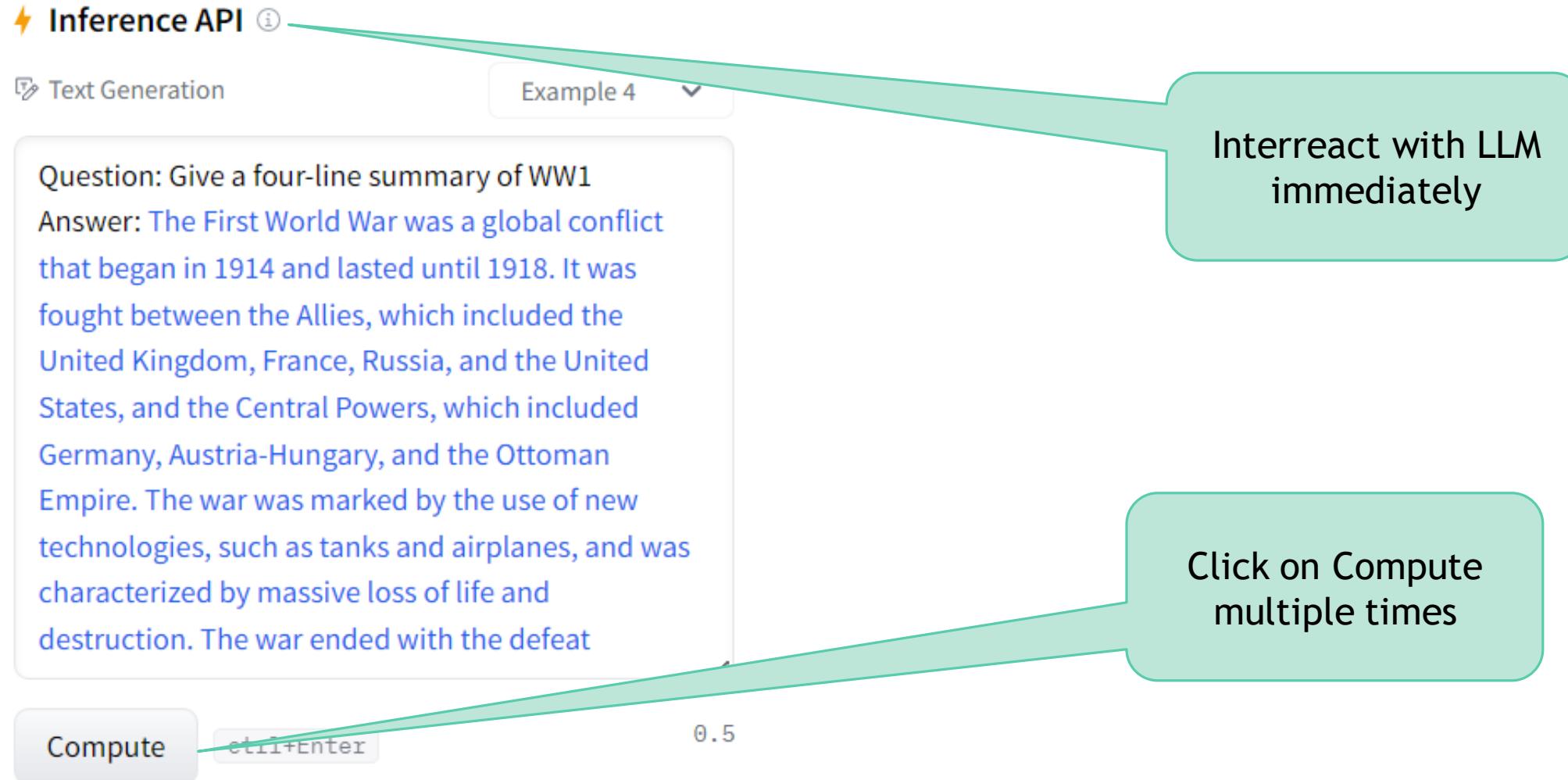
Below the tabs, there's a dropdown for the branch "main", the repository name "gemma-7b", information about 13 contributors, a history of 11 commits, and a "Contribute" button.

The file list shows the following entries:

File	Size	Description	Last Commit
examples	359f554	update examples/noi	14 days ago
.gitattributes	1.62 kB	Squashing commit	17 days ago
README.md	21.1 kB	Update README.md (#51)	21 days ago
config.json	629 Bytes	Squashing commit	14 days ago
gemma-7b.gguf	34.2 GB	LFS Squashing commit	21 days ago
generation-config.json	407 Bytes	Squashing commit	21 days ago

A callout bubble points to the "examples" entry with the text: "Click and see some examples".

Model from Hugging face



The screenshot shows the Hugging Face Inference API interface. At the top left is a yellow lightning bolt icon labeled "Inference API". To its right are "Text Generation" and "Example 4" dropdown menus. A large text box contains a question about WWI and its detailed answer. Below the text box are two buttons: "Compute" and "ctrl+Enter". A green callout bubble points to the "Compute" button with the text "Click on Compute multiple times". Another green callout bubble points to the "ctrl+Enter" button with the text "Interreact with LLM immediately".

⚡ Inference API ⓘ

Text Generation Example 4

Question: Give a four-line summary of WW1

Answer: The First World War was a global conflict that began in 1914 and lasted until 1918. It was fought between the Allies, which included the United Kingdom, France, Russia, and the United States, and the Central Powers, which included Germany, Austria-Hungary, and the Ottoman Empire. The war was marked by the use of new technologies, such as tanks and airplanes, and was characterized by massive loss of life and destruction. The war ended with the defeat

Compute ctrl+Enter

0.5

Interreact with LLM immediately

Click on Compute multiple times

Model from Hugging face

Running the model on a CPU

```
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("google/gemini-1.0")
model = AutoModelForCausalLM.from_pretrained("google/gemini-1.0")

input_text = "Write me a poem about Machine Learning."
input_ids = tokenizer(input_text, return_tensors="pt").input_ids

outputs = model.generate(**input_ids)
print(tokenizer.decode(outputs[0]))
```

Running the model on a single / multi GPU

```
# pip install accelerate
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("google/gemini-1.0")
model = AutoModelForCausalLM.from_pretrained("google/gemini-1.0")

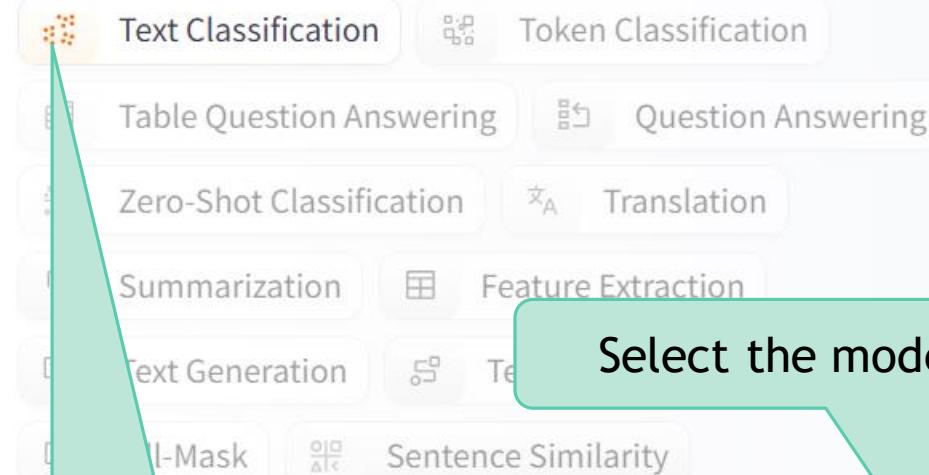
input_text = "Write me a poem about Machine Learning."
input_ids = tokenizer(input_text, return_tensors="pt").input_ids

outputs = model.generate(**input_ids)
print(tokenizer.decode(outputs[0]))
```

Code samples
provided as part of
documentation

Text Classification - sentiment analysis

Natural Language Processing



Models 64,133

Filter by name

Full-text search

↑↓ Sort: Trending

-  RLHFlow/ArmoRM-Llama3-8B-v0.1
Text Classification • Updated 2 days ago • ↓ 6.38k • ❤ 69
-  BAAI/bge-reranker-v2-m3
Text Classification • Updated Mar 19 • ↓ 1.11M • ❤ 127
-  mrm8488/distilroberta-finetuned-financial-news-sentiment-an...
Text Classification • Updated Jan 21 • ↓ 4.56M • ❤ 269
-  HuggingFaceFW/fineweb-edu-classifier
Text Classification • Updated 16 days ago • ↓ 1.37M • ❤ 62
-  j-hartmann/emotion-english-distilroberta-base
Text Classification • Updated Jan 2, 2023 • ↓ 809k • ❤ 313
-  cardiffnlp/twitter-roberta-base-sentiment-latest
Text Classification • Updated May 28, 2023 • ↓ 11.2M • ❤ 452

distilbert model for sentiment analysis

distilbert/distilbert-base-uncased-finetuned-sst-2-english

like 415

Text Classification Transformers PyTorch TensorFlow Rust ONNX Safetensors sst2 glue English doi:10.57967/hf/0181 distilbert

Eval Results Inference Endpoints arxiv:1910.01108 License: apache-2.0

Model card Files and versions Community 29 Edit model card

DistilBERT base uncased finetuned SST-2

Downloads last month 12,564,166

Safetensors Model size 67M params Tensor type F32

Inference API

Text Classification Examples

This product is too good

Compute

Computation time on cpu: 0.336 s

POSITIVE 0.999

Sentiment

Model Details

Model Description: This model is a fine-tuned version of the DistilBERT base model. It has been trained on the Stanford Sentiment Treebank (SST-2) dataset, which contains 16000 labeled sentences. The model has been trained to predict the sentiment of a given sentence, either positive or negative.

deepset/roberta-base-squad2

deepset/roberta-base-squad2 like 582

Question Answering Transformers PyTorch TensorFlow JAX Rust Safetensors squad_v2 English roberta Eval Results

Inference Endpoints License: cc-by-4.0

Model card Files and versions Community 17 Edit model card

roberta-base for QA

This is the [roberta-base](#) model, fine-tuned using the [SQuAD2.0](#) dataset. It's been trained on question-answer pairs, including unanswerable questions, for the task of Question Answering.

Downloads last month 1,012,062

Safetensors Model size 124M params Tensor type F32 · I64

Inference API

Question Answering Examples

Which name is also used to describe the Amazon rainforest Compute

Context

The Amazon rainforest (Portuguese: Floresta Amazônica or Amazônia; Spanish: Selva Amazónica, Amazonía or usually Amazonia; French: Forêt amazonienne; Dutch: Amazoneregenwoud), also known in English as Amazonia or the Amazon Jungle, is a moist broadleaf forest that covers most of the Amazon basin of South America. This basin encompasses 7,000,000 square kilometres (2,700,000 sq mi), of which 5,500,000 square kilometres (2,100,000

Hugging face pipeline()

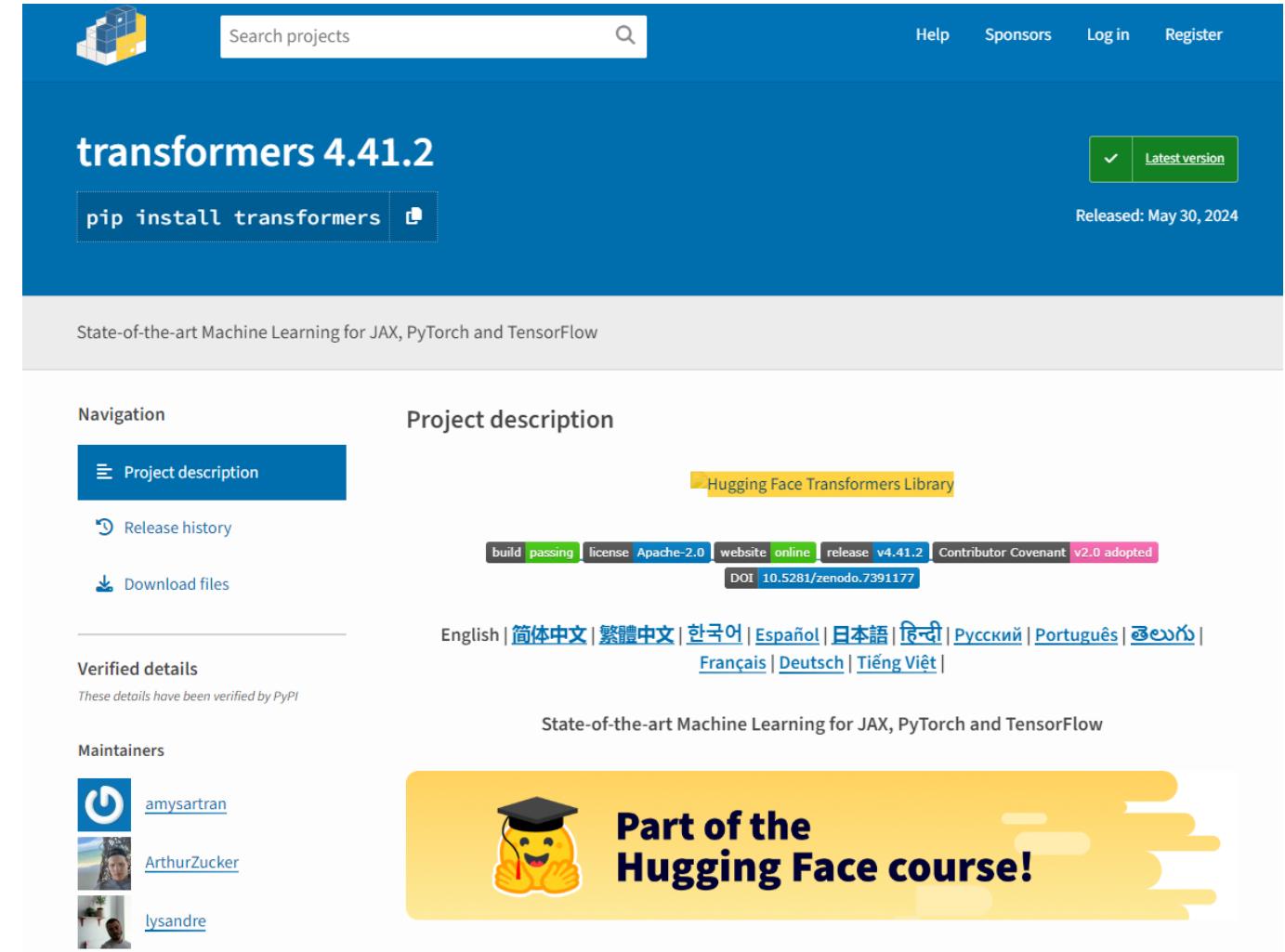
To get started quickly

```
-  
  
from transformers import AutoTokenizer, AutoModelForCausalLM  
  
tokenizer = AutoTokenizer.from_pretrained("google/gemini-1.0")  
model = AutoModelForCausalLM.from_pretrained("google/gemini-1.0")  
  
input_text = "Write me a poem about Machine Learning."  
input_ids = tokenizer(input_text, return_tensors="pt")  
  
outputs = model.generate(**input_ids)  
print(tokenizer.decode(outputs[0]))
```

- This is the original code, but there is an easy way to call this model
- We will use `pipeline()` function

About the transformers package

- Transformers is a community of projects and the Hugging Face Hub.
- It supports developers, researchers, students, professors, and engineers.
- Enables users to build their dream projects with ease.
- Provides thousands of pre-trained models for various tasks.
- Supports different modalities like text, vision, and audio.



The screenshot shows the official GitHub repository page for the `transformers` package. The header features the project name "transformers 4.41.2", a "pip install transformers" button, and a "Latest version" button. Below the header, a banner states "State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow". The page is divided into sections: "Navigation" (Project description, Release history, Download files), "Project description" (Hugging Face Transformers Library, build passing, license Apache-2.0, website online, release v4.41.2, Contributor Covenant v2.0 adopted, DOI 10.5281/zendodo.7391177), and "Verified details" (These details have been verified by PyPI). The "Maintainers" section lists three individuals: amysartran, ArthurZucker, and lysandre, each with a small profile picture. A yellow banner at the bottom right says "Part of the Hugging Face course!" with a graduation cap emoji.

About the pipeline() function

- The pipeline package in the transformers library simplifies the process of using pre-trained NLP models
- It helps us by abstracting away complex steps like tokenization, model loading, and post-processing.
- Almost similar code can be used for multiple pre-trained models.
- The pipeline() function takes care of the necessary steps, including tokenizing input text, passing it through the model, and returning the results in a user-friendly format.

```
from transformers import pipeline
```

Sentiment Analysis Model

```
senti_model = pipeline(task="sentiment-analysis")
```

- There are multiple models for sentiment analysis
- We have not specified any model name
- The default model will be considered

No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision af0f99b (<https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english>).

Using a pipeline without specifying a model name and revision in production is not recommended.

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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/settings/tokens>), set it as secret in your Google Colab and restart your session.

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(
```

- Hugging face provides an API key. We have not mentioned it here.
- It still works for some models

Sentiment Analysis Model - Result

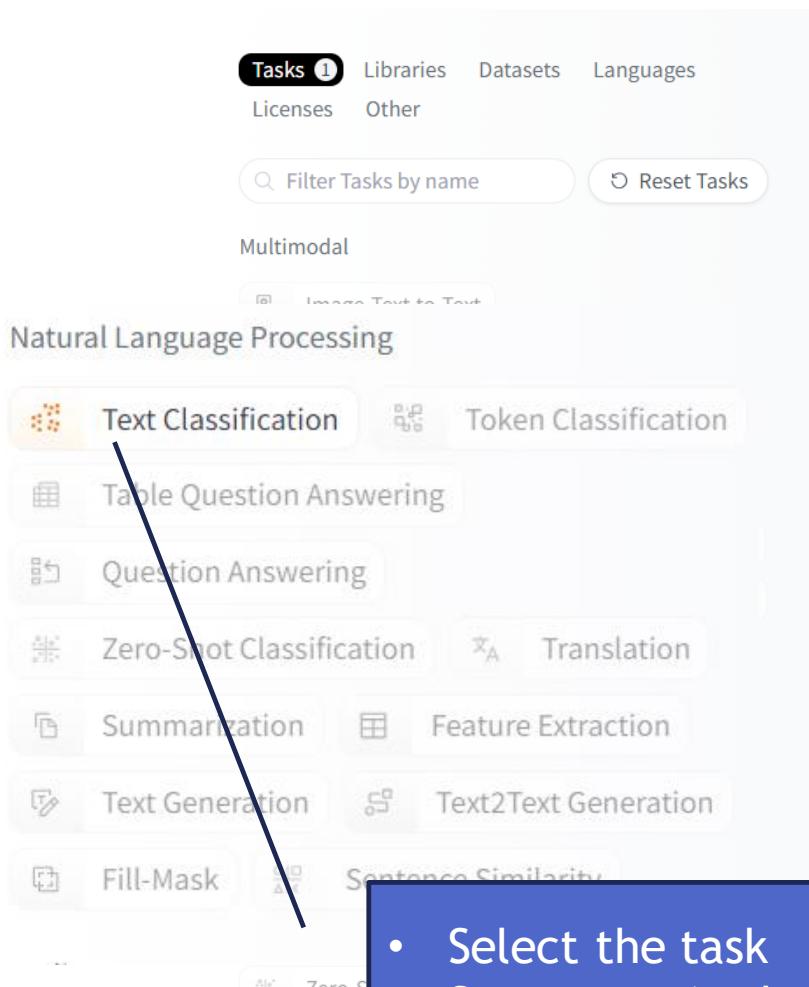
```
senti_model("This movie is damn good. I loved it")
```

```
[{'label': 'POSITIVE', 'score': 0.9998770952224731}]
```

```
senti_model("This is a bad phone. The screen and battery are of poor quality.")
```

```
[{'label': 'NEGATIVE', 'score': 0.9998168349266052}]
```

Choose your model



Tasks 1 Libraries Datasets Languages
Licenses Other

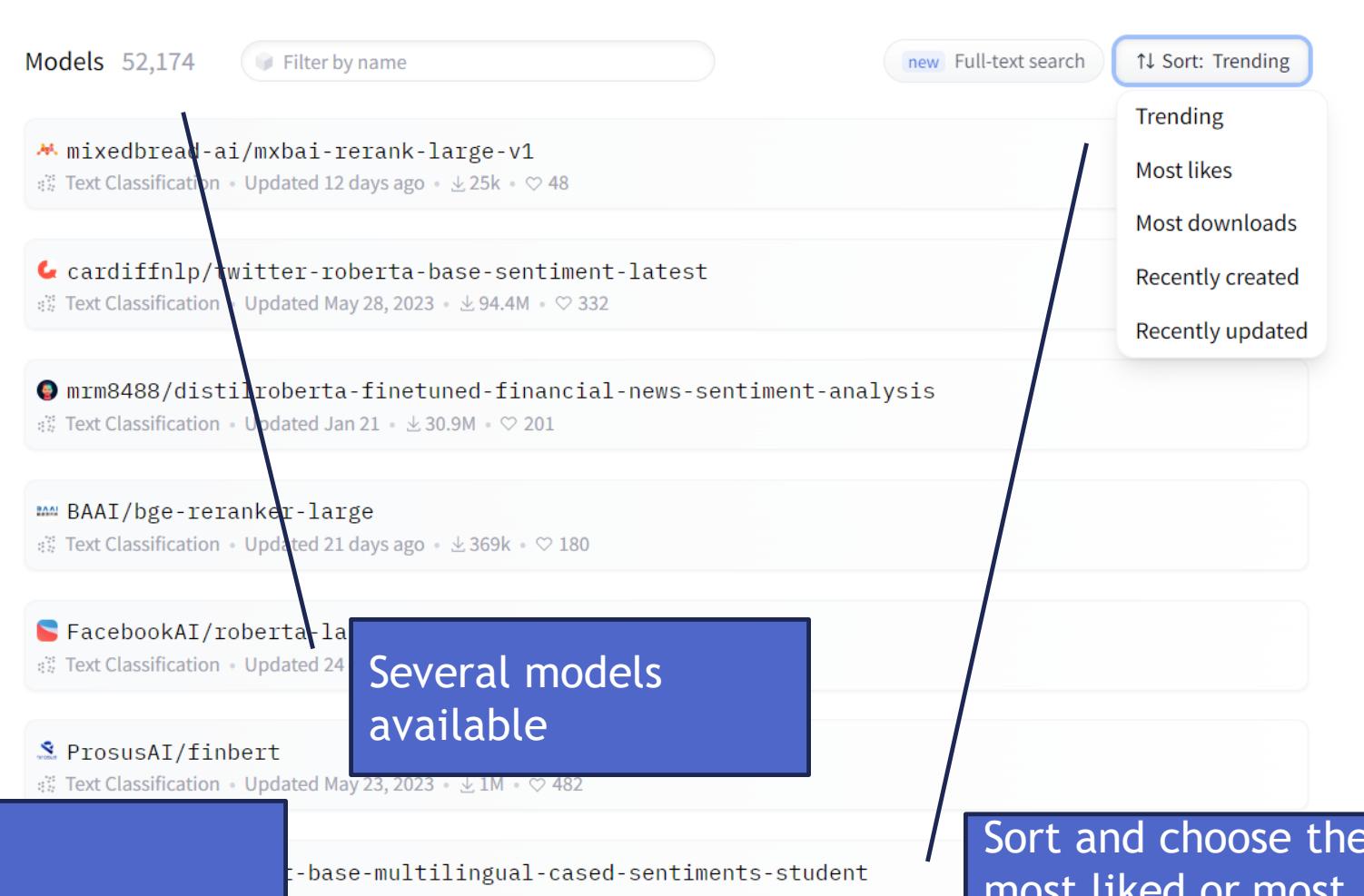
Filter Tasks by name Reset Tasks

Multimodal

Image Text-to-Text

Natural Language Processing

- Text Classification**
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Feature Extraction
- Text Generation
- Text2Text Generation
- Fill-Mask
- Sentence Similarity
- Zero-S



Models 52,174 Filter by name

↑ Sort: Trending

Trending
Most likes
Most downloads
Recently created
Recently updated

mixedbread-ai/mxbai-rerank-large-v1
Text Classification • Updated 12 days ago • 25k • 48

cardiffnlp/twitter-roberta-base-sentiment-latest
Text Classification • Updated May 28, 2023 • 94.4M • 332

mrm8488/distilroberta-finetuned-financial-news-sentiment-analysis
Text Classification • Updated Jan 21 • 30.9M • 201

BAAI/bge-reranker-large
Text Classification • Updated 21 days ago • 369k • 180

FacebookAI/roberta-large
Text Classification • Updated 24 days ago • 1.1B • 482

ProsusAI/finbert
Text Classification • Updated May 23, 2023 • 1M • 482

t-base-multilingual-cased-sentiments-student
Text Classification • Updated Jun 24, 2023 • 10.8M • 156

- Select the task
- Sentiment Analysis comes under text classification

Several models available

Sort and choose the most liked or most downloaded or any other model of your choice

Choose your model

[cardiffnlp/twitter-roberta-base-sentiment-latest](#)

like 332

Text Classification Transformers PyTorch TensorFlow tweet_eval English roberta Inference Endpoints arxiv:2202.03829

Model card Files and versions Community 19 Edit model card

Downloads last month 94,437,085

Inference API Examples

Text Classification

Copy the model name

Computation time on cpu: cached

Label	Time (ms)
negative	0.724
neutral	0.229
positive	0.048

JSON Output Maximize

Twitter-roBERTa-base for Sentiment Analysis - UPDATED (2022)

This is a RoBERTa-base model trained on ~124M tweets from January 2018 to December 2021, and finetuned for sentiment analysis with the TweetEval benchmark. The original Twitter-based RoBERTa model can be found [here](#) and the original reference paper is [TweetEval](#). This model is suitable for English.

- Reference Paper: [TimeLMs paper](#).
- Git Repo: [TimeLMs official repository](#).

Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive

This sentiment analysis model has been integrated into [TweetNLP](#). You can access the demo [here](#).

Choose your model

Senti_model_2("Over heating issue don't by this product camera was good")

```
[{'label': 'neutral', 'score': 0.4210317134857178}]
```

Senti model 2 ("Waste of money")

1 ★ Horrible

Waste of money

Avinash Kumar Certified Buyer, Gurugram 6 days ago

```
[{'label': 'negative', 'score': 0.7434294819831848}]
```

```
Senti_model_2("Nice product under 24k .... overall good")
```

5 ★ Super!

Nice product under 24k overall good

Flipkart Customer Certified Buyer, Habra 3 days ago

Zero shot prediction

- Zero-shot prediction means asking the model to understand and carry out a task without having been given any specific examples of how to do it during its training.
- It depends on the LLM's broad understanding of language and concepts to make educated guesses.
- **Example:** If an LLM is given a task to classify customer reviews into categories like "service", "quality", "price", without ever being trained on these specific categories, it uses its general knowledge to infer which category a review belongs to based on the content of the review.

Using this model on your dataset

```
import pandas as pd
user_review_data=pd.read_csv("https://raw.githubusercontent.com/abhishekkrishna1993/Text-Sentiment-Analysis/master/reviews.csv")
user_review_data=user_review_data.sample(50)
user_review_data[ "Review"]
```

```
23      I have yet to run this new battery below two b...
917          Leopard Print is wonderfully wild!.
243      No additional ear gels provided, and no instru...
1049     My side Greek salad with the Greek dressing wa...
235          One of my favorite purchases ever.
1245     -Drinks took close to 30 minutes to come out a...
348      This little device has transformed my organiz...
542      Perhaps my phone is defective, but people cann...
595      What possesed me to get this junk, I have no i...
1708     Service is quick and even "to go" orders are j...
304      Everything worked on the first try.The device ...
325          good item, low price.
489      This results in the phone being either stuck a...
949          I had to purchase a different case.
87          The construction of the headphones is poor.
```

Actual Reviews Data

Sentiment Analysis on a Data frame

```
user_review_data["Predicted_Sentiment"] = user_review_data["Review"].apply(lambda x: Senti_model_2(x)[0]["label"])
user_review_data
```

	Id	Review	Sentiment	Predicted_Sentiment
0	1	So there is no way for me to plug it in here i...	0	negative
1	2	Good case, Excellent value.	1	positive
2	3	Great for the jawbone.	1	positive
3	4	Tied to charger for conversations lasting more...	0	negative
4	5	The mic is great.	1	positive
...
1995	1996	I think food should have flavor and texture an...	0	negative
1996	1997	Appetite instantly gone.	0	negative

✓ 1m 5s completed at 10:42 AM

Prediction takes a little longer if we load the model on CPU

Load the model on GPU

```
Senti_model_2_gpu = pipeline(task="sentiment-analysis",
                             model="cardiffnlp/twitter-roberta-base-sentiment-latest",
                             device="cuda")
```

	<u>ID</u>	<u>Review</u>	<u>Sentiment</u>	<u>Predicted_Sentiment</u>
0	1	So there is no way for me to plug it in here i...	0	negative
1	2	Good case, Excellent value.	1	positive
2	3	Great for the jawbone.	1	positive
3	4	Tied to charger for conversations lasting more...	0	negative
4	5	The mic is great.	1	positive
...
1995	1996	I think food should have flavor and texture an...	0	negative
1996	1997	Appetite instantly gone.	0	negative
1997	1998	Overall I was not impressed and would not go b...	0	negative

device="cuda" for GPU

Takes less time to complete it

Other models on Hugging Face

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Text-to-Image
- Image-to-Text
- Image-to-Image
- Image-to-Video
- Unconditional Image Generation
- Video Classification
- Text-to-Video
- Zero-Shot Image Classification
- Mask Generation
- Zero-Shot Object Detection
- Text-to-3D
- Image-to-3D
- Image Feature Extraction

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Feature Extraction
- Text Generation
- Text2Text Generation
- Fill-Mask
- Sentence Similarity

Audio

- Text-to-Speech
- Text-to-Audio
- Automatic Speech Recognition
- Audio-to-Audio
- Audio Classification
- Voice Activity Detection

Tabular

- Tabular Classification
- Tabular Regression

Reinforcement Learning

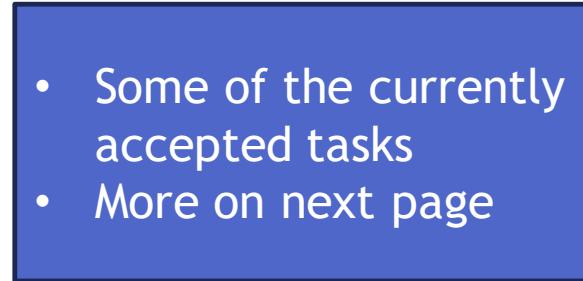
- Reinforcement Learning
- Robotics

Other

- Graph Machine Learning

Models for various tasks.

- ♪ "audio-classification": Returns a **AudioClassificationPipeline**.
- 🎙️ □ "automatic-speech-recognition": Returns a **AutomaticSpeechRecognitionPipeline**.
- 💬 "conversational": Returns a **ConversationalPipeline**.
- 🖌️ "depth-estimation": Returns a **DepthEstimationPipeline**.
- 📄 "document-question-answering": Returns a **DocumentQuestionAnsweringPipeline**.
- 🔎 "feature-extraction": Returns a **FeatureExtractionPipeline**.
- 🧩 "fill-mask": Returns a **FillMaskPipeline**.
- 🖼️ □ "image-classification": Returns a **ImageClassificationPipeline**.
- 🖼️ □ → 🔎 "image-feature-extraction": Returns an **ImageFeatureExtractionPipeline**.
- 🖼️ □ □ "image-segmentation": Returns a **ImageSegmentationPipeline**.
- 🖼️ □ → 🖼️ □ "image-to-image": Returns a **ImageToImagePipeline**.
- 🖼️ □ → 📄 "image-to-text": Returns a **ImageToTextPipeline**.
- 🧩 "mask-generation": Returns a **MaskGenerationPipeline**.
- 🚶♂️ □ "object-detection": Returns a **ObjectDetectionPipeline**.

- 
- Some of the currently accepted tasks
 - More on next page

- **"question-answering"**: Returns a **QuestionAnsweringPipeline**.
- **"summarization"**: Returns a **SummarizationPipeline**.
- **"table-question-answering"**: Returns a **TableQuestionAnsweringPipeline**.
- **"text2text-generation"**: Returns a **Text2TextGenerationPipeline**.
- **"text-classification"** (alias **"sentiment-analysis"**): Returns a **TextClassificationPipeline**.
- **"text-generation"**: Returns a **TextGenerationPipeline**.
- **"text-to-audio"** (alias **"text-to-speech"**): Returns a **TextToAudioPipeline**.
- **"token-classification"** (alias **"ner"**): Returns a **TokenClassificationPipeline**.
- **"translation"**: Returns a **TranslationPipeline**.
- **"translation_xx_to_yy"**: Returns a **TranslationPipeline**.
- **"video-classification"**: Returns a **VideoClassificationPipeline**.
- **"visual-question-answering"**: Returns a **VisualQuestionAnsweringPipeline**.
- **"zero-shot-classification"**: Returns a **ZeroShotClassificationPipeline**.
- **"zero-shot-image-classification"**: Returns a **ZeroShotImageClassificationPipeline**.
- **"zero-shot-audio-classification"**: Returns a **ZeroShotAudioClassificationPipeline**.
- **"zero-shot-object-detection"**: Returns a **ZeroShotObjectDetectionPipeline**.

We are interested in
these NLP based
tasks

Python 3 Google Compute Engine backend (GPU)

Showing resources from 10:18 AM to 12:01 PM



System RAM
6.8 / 51.0 GB



GPU RAM
5.8 / 15.0 GB



Disk
28.0 / 201.2 GB



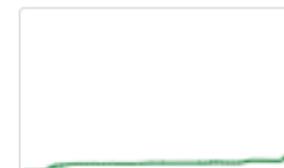
Clear cache in GPU

```
import torch
torch.cuda.empty_cache()
```

Python 3 Google Compute Engine backend (GPU)

Showing resources from 10:18 AM to 12:02 PM

System RAM
6.9 / 51.0 GB



GPU RAM
1.1 / 15.0 GB



Disk
28.0 / 201.2 GB



Language Translation Model

```
translator_model = pipeline(task="translation_en_to_fr",
                            model="google-t5/t5-small",
                            device="cuda")
translator_model("Good bye")
```

```
[{'translation_text': 'Bon droit'}]]
```

Q&A Model

```
qa_model = pipeline(task="question-answering",
                     model="deepset/roberta-base-squad2",
                     device="cuda")
```

Our model

```
#Importing computer_scientists.txt document from github
!wget https://raw.githubusercontent.com/venkatareddykonasani/DV/main/computer_scientists.txt
document=open("computer_scientists.txt").read()
```

Input Document

Q&A based on a document

```
qa_model({'question':'What did Yann LeCun contribute?',  
| | | | | 'context':document})
```

```
{'score': 0.24149088561534882,  
'start': 330,  
'end': 377,  
'answer': 'Revolutionized AI for image and text processing'}
```

```
qa_model({'question':'Who is the father of deep learning?',  
| | | | | 'context':document})
```

```
{'score': 0.7754606008529663,  
'start': 1421,  
'end': 1444,  
'answer': 'Geoffrey Everest Hinton'}
```

NER Model

```
ner_model = pipeline(task="ner",
                      model="dslim/bert-base-NER",
                      device="cuda",
                      aggregation_strategy="simple")
#aggregation_strategy ="Simple" ; simplifies the output and makes it easy to read
```

NER Model

```
sample_doc="""
Hello,
    I, John Smith, a member of the Tech Innovators team, would like to schedule a meeting with you,
    Mary Johnson, from the Quantum Solutions group, on Tuesday, February 8th, 2024, at 10:00 AM.
    We can meet at your office in San Francisco or, if more convenient, at the Cafe Bella in New York City.
    Please let me know if this date and time work for you.
"""
```

NER output

```
entities = ner_model(sample_doc)
print(entities)
```

```
[{'entity_group': 'PER', 'score': 0.9994373, 'word': 'John Smith', 'start': 13, 'end': 23}, {'entity_group': 'ORG', 'score': 0.9968871, 'word': 'Tech Innovators', 'start': 41, 'end': 56}, {'entity_group': 'PER', 'score': 0.99906284, 'word': 'Mary Johnson', 'start': 108, 'end': 120}, {'entity_group': 'ORG', 'score': 0.9988469, 'word': 'Quantum Solutions', 'start': 131, 'end': 148}, {'entity_group': 'LOC', 'score': 0.9993695, 'word': 'San Francisco', 'start': 233, 'end': 246}, {'entity_group': 'ORG', 'score': 0.63540447, 'word': 'Cafe Bella', 'start': 278, 'end': 288}, {'entity_group': 'LOC', 'score': 0.9994677, 'word': 'New York City', 'start': 292, 'end': 305}]
```

NER Model

```
# Convert the above output into a dataframe and print it with the entity name
NER_result = pd.DataFrame(entities, columns=["word", "entity_group"])

# Print the DataFrame
print(NER_result)
```

	word	entity_group
0	John Smith	PER
1	Tech Innovators	ORG
2	Mary Johnson	PER
3	Quantum Solutions	ORG
4	San Francisco	LOC
5	Cafe Bella	ORG
6	New York City	LOC

Output stored in a data frame

Text Summarization Model

```
summarizer_model = pipeline(task="summarization",
                             model="google/pegasus-xsum",
                             device="cuda")
```

```
Book_essay = """
The 7 Habits of Highly Effective People" is a timeless self-help book by Stephen R. Covey. His philosophy centers on the idea that true success is achieved by aligning with principles like integrity, respect, and a sense of purpose. The next three habits delve into the concept of interdependence, emphasizing the importance of building relationships and working together. The seventh habit, "Sharpen the Saw," encourages continuous self-renewal and personal growth through physical, mental, spiritual, and social development. Throughout the book, Covey provides practical advice and real-life examples to illustrate these principles.
"""
```

```
print(summarizer.model(Book essay, max_length=120, min_length=30))
```

```
[{'summary_text': '"The 7 Habits of Highly Effective People" is a timeless self-help book by Stephen R. Covey that offers holistic approach to personal and professional effectiveness.'}]
```

Text Generation Model

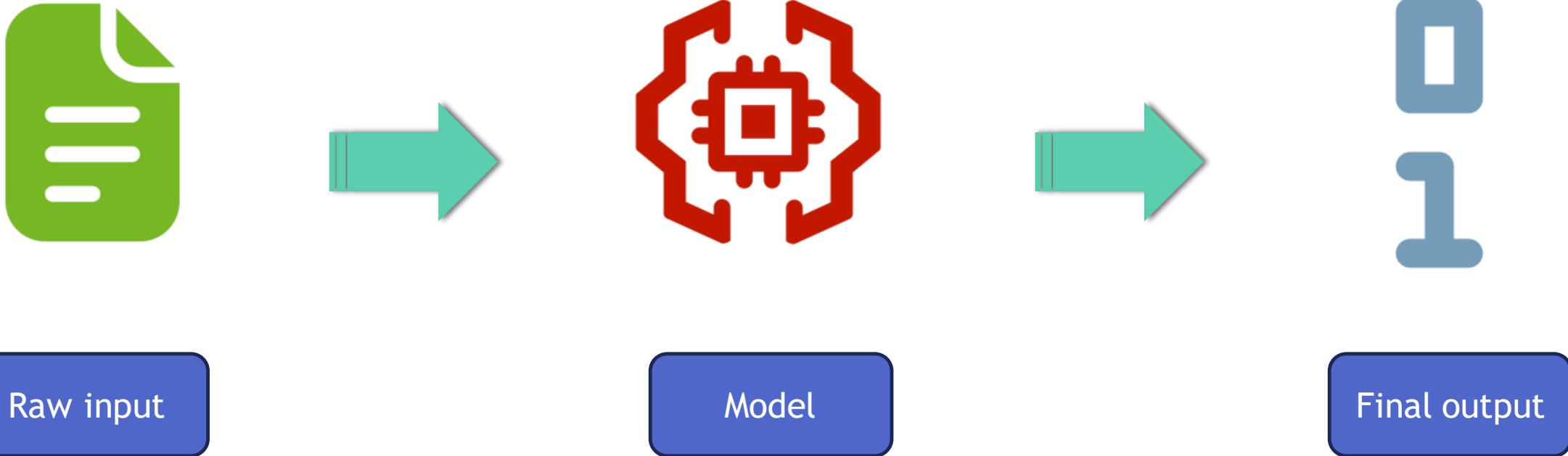
```
text_generator_model = pipeline(task="text-generation",
                                 model="gpt2",
                                 device="cuda")
```

```
# Generate text starting with the given prompt
text_result = text_generator_model("The best way to start a presentation is")
print(text_result)
```

Setting `pad token id` to `eos token id`:50256 for open-end generation.

```
[{'generated_text': 'The best way to start a presentation is to find the most exciting thing about any subject imaginable--how exciting a subject it is, and the value it creates," says Mark Bivens, a researcher as well as a research associate at the Massachusetts Institute'}]
```

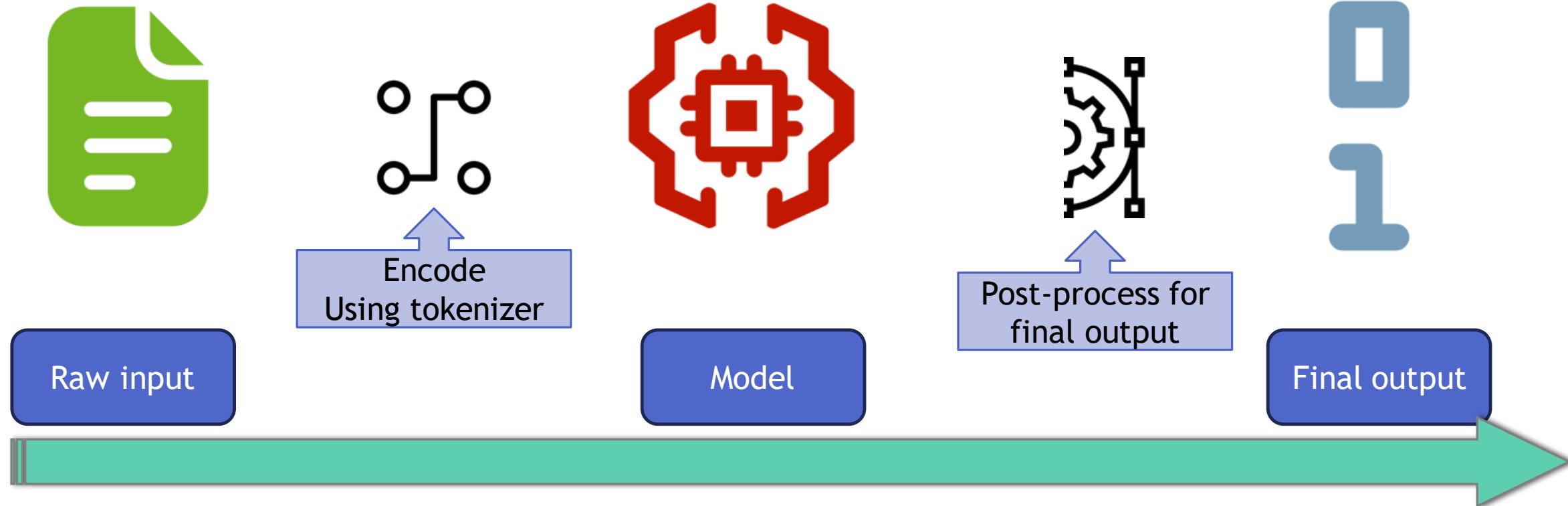
pipeline() function



```
Senti_model_2 = pipeline(task="sentiment-analysis",
                         model="cardiffnlp/twitter-roberta-base-sentiment-latest")
```

```
Senti_model_2("Over heating issue don't by this product camera was good")
```

Without pipeline() function



- We have to convert raw input text into encoded values using a tokenizer
- Get the output from the model. The output will be in the form of tensors. We need to process it to display the original classes

Without pipeline() function

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")

model = AutoModelForSequenceClassification.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")
```

- Every model has its own tokenizer function.
- Auto tokenizer and Auto model functions work for almost all the models

Without pipeline() function

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")

model = AutoModelForSequenceClassification.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")
```

```
import numpy as np
raw_text = "This is a great book"
encoded_input = tokenizer(raw_text, return_tensors='pt')
output = model(**encoded_input)
logits = output.logits.detach().numpy()
y_pred = np.argmax(logits)
y_pred
```

2

- Model output will be in the form of tensors
- This post processing step will extract the logits from the output
- We use those logits to decide our final predicted class

- Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive

Finetuning the models on our custom data

Transfer learning from Hugging Face Models

- So far, we've been using the model just as it was originally trained on.
- The model might NOT have learned from data that's exactly similar to ours before.
- Now, let's learn how to adjust an LLM model that was already trained, to work better with our data.

What happens if we don't fine tune?

- The LLM might find it hard to figure out and sort the categories in our special data.
- We can use the LLM's ability to think and solve problems, but we shouldn't depend only on its Zero-shot prediction(first-time guesses) .
- For instance, in tasks like text classification, LLMs usually look for positive or negative sentiment.
- But what if all we have are bad complaints, yet they're about different kinds or departments?
- How do we help the LLM understand that our data isn't like the usual sentiment analysis data it sees?
- By finetuning and showing some examples to the existing LLM

The Problem Statement and Dataset



- <https://www.consumerfinance.gov/>
- The Consumer Financial Protection Bureau (CFPB) acts as a mediator between financial institutions and consumers, facilitating dispute resolution when complaints arise.
- To improve efficiency and accuracy in handling customer complaints, they would like to automatically classify and route complaints to the appropriate teams based on their content and associated financial products.
- <https://www.kaggle.com/datasets/adhamelkomy/bank-customer-complaint-analysis/data>

About the dataset

```
!wget https://github.com/venkatareddykonasani/Datasets/raw/master/complaints_v2.zip
!unzip -o complaints_v2.zip
complaints_data = pd.read_csv("/content/complaints_v2.csv")
complaints_data.head()
```

product	text	label
credit_card	purchase order day shipping amount receive pro...	1
credit_card	forwarded message date tue subject please inve...	1
retail_banking	forwarded message cc sent friday pdt subject f...	1
credit_reporting	payment history missing credit report special...	0
credit_reporting	payment history missing credit report made mis...	0

- Bank customers complaints data
- Customer send email complaints for various products
- we are broadly classifying everything as “credit reporting” and “other complaints “

Lets try Zero Shot Prediction without finetuning

```
from transformers import pipeline
distilbert_model = pipeline(task="text-classification",
                             model="distilbert-base-uncased",
                             device="cuda",
                             )
```

- 
- The distil-bert model is lightweight
 - We can finetune it in our system
 - Remaining models are heavy and we may take a lot of execution time.

Prediction

- Take a sample dataset 100 -1000
- Consider only the initial 350 words in a complaint

```
# Sampling 1000 random entries from the complaints data for analysis
sample_data = complaints_data.sample(1000, random_state=42)

# Trimming each text entry to the first 350 words to reduce processing time and memory usage
sample_data["text"] = sample_data["text"].apply(lambda x: " ".join(x.split()[:350]))

# Applying the DistilBERT model to predict the category of each complaint in the sample
sample_data["bert_predicted"] = sample_data["text"].apply(lambda x: distilbert_model(x)[0]["label"])

# Extracting the numerical part of the predicted category (assuming the label is something like 'LABEL_0')
sample_data["bert_predicted_num"] = sample_data["bert_predicted"].apply(lambda x: x[-1])
```

Prediction for each row

Accuracy of Zero Shot Prediction

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(sample_data["label"], sample_data["bert_predicted_num"])
print(cm)
accuracy=cm.diagonal().sum()/cm.sum()
print(accuracy)
```

```
[[ 0 544]
 [ 1 455]]
0.455
```

Almost all the records are predicted
as class-1

Lets finetune on our data

- Steps in finetuning

1. Import your dataset and pre-process the data and convert it into hugging face friendly datasets
2. Tokenize and pad the dataset
3. Define the model and number of labels
4. Define the training arguments like model output directory
5. Finetuning step - Train the model by supplying the model and tokenizer
6. Ultimately utilizing the refined model for making predictions

Install and Import the packages

```
# Quietly update and install the accelerate package
!pip -q install accelerate -U
# Quietly install the transformers library along with its PyTorch dependencies
!pip -q install transformers[torch]
# Quietly install the Hugging Face datasets library
!pip -q install datasets

# Importing necessary libraries and classes for model training and data handling
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, Trainer, TrainingArguments
from transformers import Trainer, TrainingArguments # Trainer and TrainingArguments are imported twice, which
from datasets import load_dataset, DatasetDict, ClassLabel, Dataset
import pandas as pd # For data manipulation and analysis
from sklearn.model_selection import train_test_split # For splitting data into training and testing sets
import torch # PyTorch library for deep learning applications
```

Hugging face friendly datasets

```
# Convert the pandas DataFrame `sample_data` to a Hugging Face `Dataset`
Sample_data = Dataset.from_pandas(sample_data)

# Split the dataset into training and testing sets
# Using an 80-20 split for training and testing, respectively
train_test_split = Sample_data.train_test_split(test_size=0.2) # 80% training, 20% testing

# Creating a DatasetDict to organize the train and test splits
dataset = DatasetDict({
    'train': train_test_split['train'],
    'test': train_test_split['test']
})
```

Tokenize and pad your data

```
# Load the tokenizer for the 'distilbert-base-uncased' model
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

# Setting up padding token to be the same as the EOS (end of sequence) token
# This aligns padding behavior with DistilBERT's expected inputs
tokenizer.pad_token = tokenizer.eos_token
tokenizer.pad_token_id = tokenizer.eos_token_id

# Adding a special pad token ('[PAD]') to the tokenizer
# This step might not be necessary since pad_token is already set to eos_token, but it explicitly ensures '[PA
tokenizer.add_special_tokens({'pad_token': '[PAD]'})

# Define a function to tokenize the input texts
def tokenize_function(examples):
    # Tokenizes the examples text, ensures they are padded to a maximum length of 512 tokens, and truncated if
    return tokenizer(examples["text"], padding="max_length", truncation=True, max_length=512)

# Apply the tokenize function to all examples in the dataset
# The 'batched=True' option processes the texts in batches, making this operation faster
tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

```
# Initialize the DistilBERT model for sequence classification with the 'distilbert-base-uncased' pre-trained model
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased',
                                                               num_labels=2, # Adjust this based on the number of classes
                                                               pad_token_id=tokenizer.eos_token_id) # Set the pad token id
```

DistilBertForSequenceClassification(
 (distilbert): DistilBertModel(
 (embeddings): Embeddings(
 (word_embeddings): Embedding(30522, 768)
 (position_embeddings): Embedding(512, 768)
 (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
 (dropout): Dropout(p=0.1, inplace=False)
)
 (transformer): Transformer(
 (layer): ModuleList(
 (0-5): 6 x TransformerBlock(
 (attention): MultiHeadSelfAttention(
 (dropout): Dropout(p=0.1, inplace=False)
 (q_lin): Linear(in_features=768, out_features=768, bias=True)
 (k_lin): Linear(in_features=768, out_features=768, bias=True)
 (v_lin): Linear(in_features=768, out_features=768, bias=True)
 (out_lin): Linear(in_features=768, out_features=768, bias=True)
)
 (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
 (ffn): FFN(
 (dropout): Dropout(p=0.1, inplace=False)
 (lin1): Linear(in_features=768, out_features=3072, bias=True)
 (lin2): Linear(in_features=3072, out_features=768, bias=True)
 (activation): GELUActivation()
)
 (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
)
)
)
 (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
 (classifier): Linear(in_features=768, out_features=2, bias=True)



Model size
67M params

Define the model and the training arguments

```
# Initialize the DistilBERT model for sequence classification with the 'distilbert-base-uncased' pre-trained model
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased',
                                                               num_labels=2, # Adjust this based on the number of classes
                                                               pad_token_id=tokenizer.eos_token_id) # Set the padding token ID

# Set up the training arguments specifying the directory for saving results, the number of training epochs, and other parameters
training_args = TrainingArguments(
    output_dir='./results_bert_custom', # Directory where the training results will be saved
    num_train_epochs=1, # Number of epochs to train for
    logging_dir='./logs_bert_custom', # Directory where the training logs will be saved
    evaluation_strategy="epoch", # Evaluate the model at the end of each epoch
)
```

Train the model

```
# Initialize the Trainer with the model, training arguments, and the train and evaluation datasets
trainer = Trainer(
    model=model, # The pre-initialized DistilBERT model
    args=training_args, # Training arguments specifying the training setup
    train_dataset=tokenized_datasets['train'], # The tokenized training dataset
    eval_dataset=tokenized_datasets['test'], # The tokenized evaluation dataset
)
# Start the training process
trainer.train()
```

This is the Fine-tuning of the BERT model to align with our updated dataset by modifying the weight

This training takes a lot of time

Save the model

```
# Define the directory where you want to save your model and tokenizer
model_dir = "./distilbert_finetuned"

# Save the model
model.save_pretrained(model_dir)

# Save the tokenizer
tokenizer.save_pretrained(model_dir)

#Save the model with
trainer.save_model('Distilbert_CustomModel_10K')
```

Make the predictions using the finetuned model

```
# Define a function for making predictions with the finetuned model
def make_prediction(text):
    # Prepare the input text for the model
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
    # Move the inputs to the GPU
    inputs = inputs.to(torch.device("cuda:0"))
    # Get model predictions
    outputs = model(**inputs)
    # Extract the predicted class (the one with the highest score) from the logits
    predictions = outputs.logits.argmax(-1)
    # Move the predictions back to CPU and convert to numpy for easy handling
    predictions = predictions.detach().cpu().numpy()
    # Return the predicted class
    return predictions

# Apply the prediction function to the 'text' column of the sample_data DataFrame
# Note: If running this line results in performance issues or out-of-memory errors, consider applying predicti
sample_data["finetuned_predicted"] = sample_data["text"].apply(lambda x: make_prediction(str(x))[0])
```

If this doesn't work, use my pre-trained model

Loading a pre-built model and making prediction

```
#Code to download the distilbert model
!gdown --id 1785J3ir19RaZP3ebbFvWUX88PMaBouro -O distilbert_finetuned_V1.zip
!unzip -o -j distilbert_finetuned_V1.zip -d distilbert_finetuned_V1

model_v1 = DistilBertForSequenceClassification.from_pretrained('/content/distilbert_finetuned_V1')
model_v1.to("cuda:0")
```

Accuracy with the new finetuned model

```
from sklearn.metrics import confusion_matrix
# Create the confusion matrix
cm1 = confusion_matrix(sample_data["label"], sample_data["finetuned_predicted"])
print(cm1)
accuracy1=cm1.diagonal().sum()/cm1.sum()
print(accuracy1)
```

```
[[479  65]
 [ 81 375]]
0.854
```

Much better than the previous model

Saving the Model on Hugging Face hub

```
from huggingface_hub import notebook_login  
notebook_login()  
#To get Auth token: Profile >> Settings >>Access Token
```



Access token

Copy a token from [your Hugging Face tokens page](#) and paste it below.

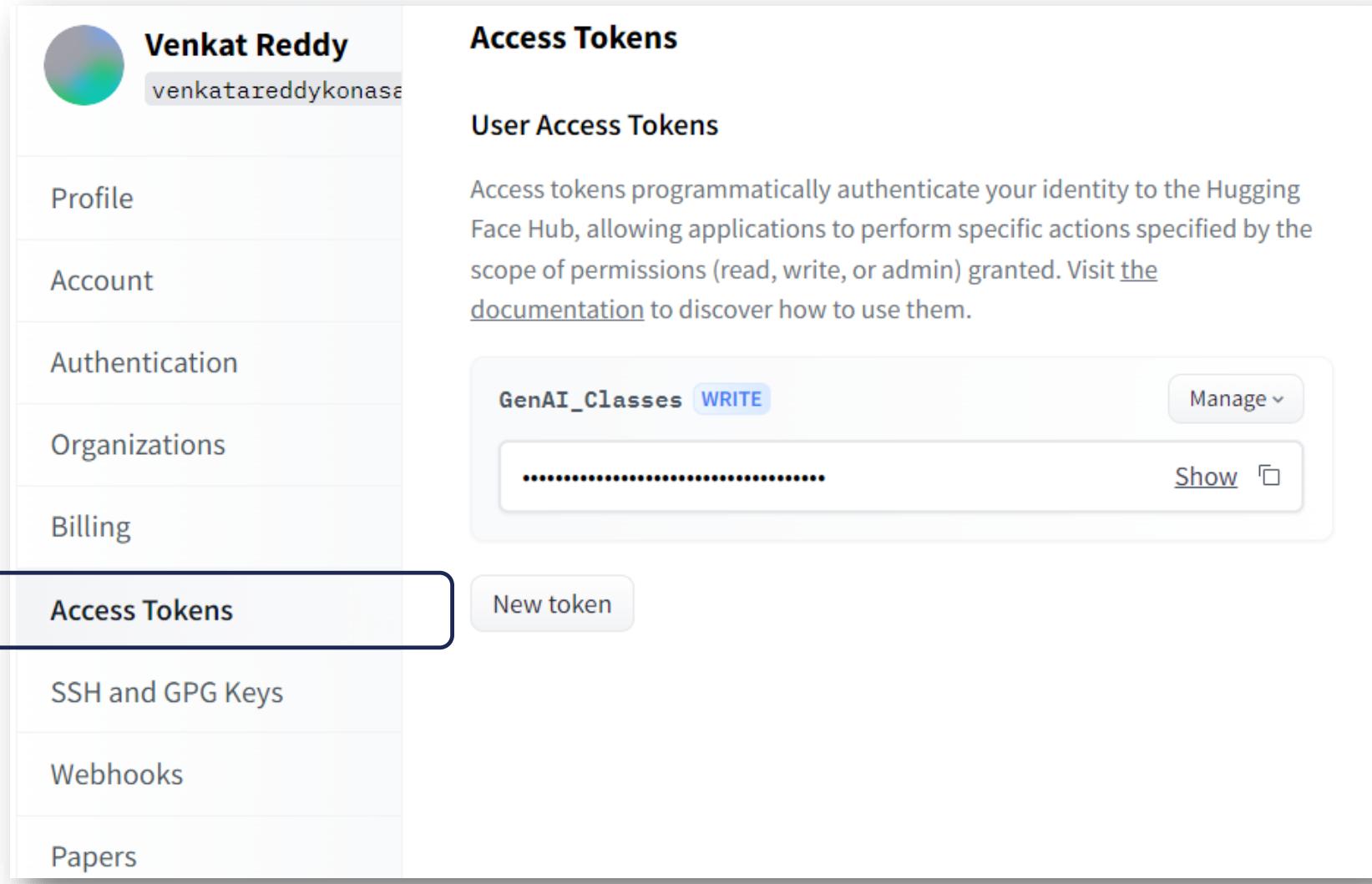
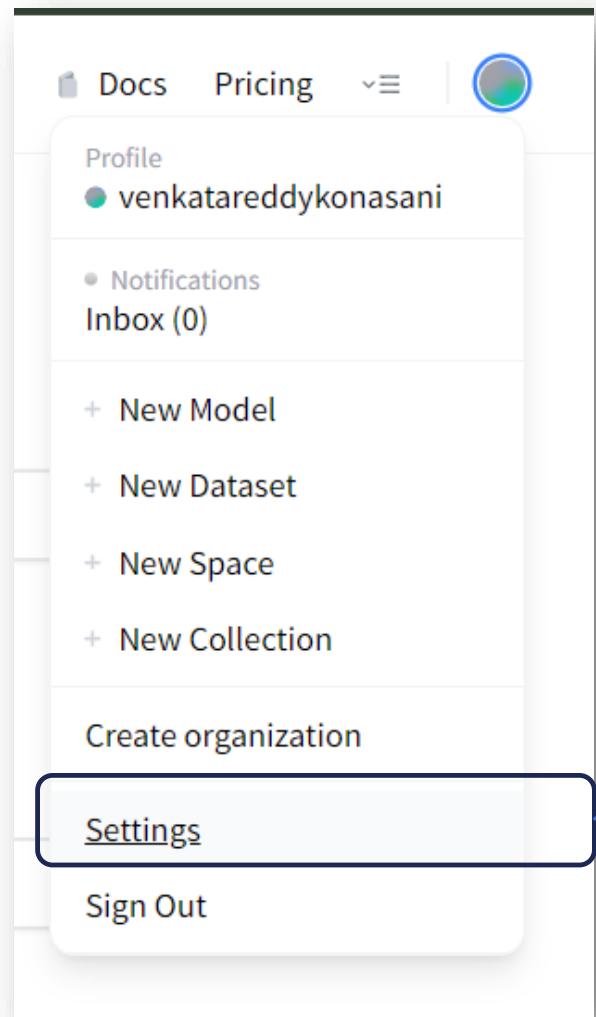
Immediately click login after copying your token or it might be stored in plain text in
this notebook file.

Token:

Add token as git credential?

Login

Get your API key here



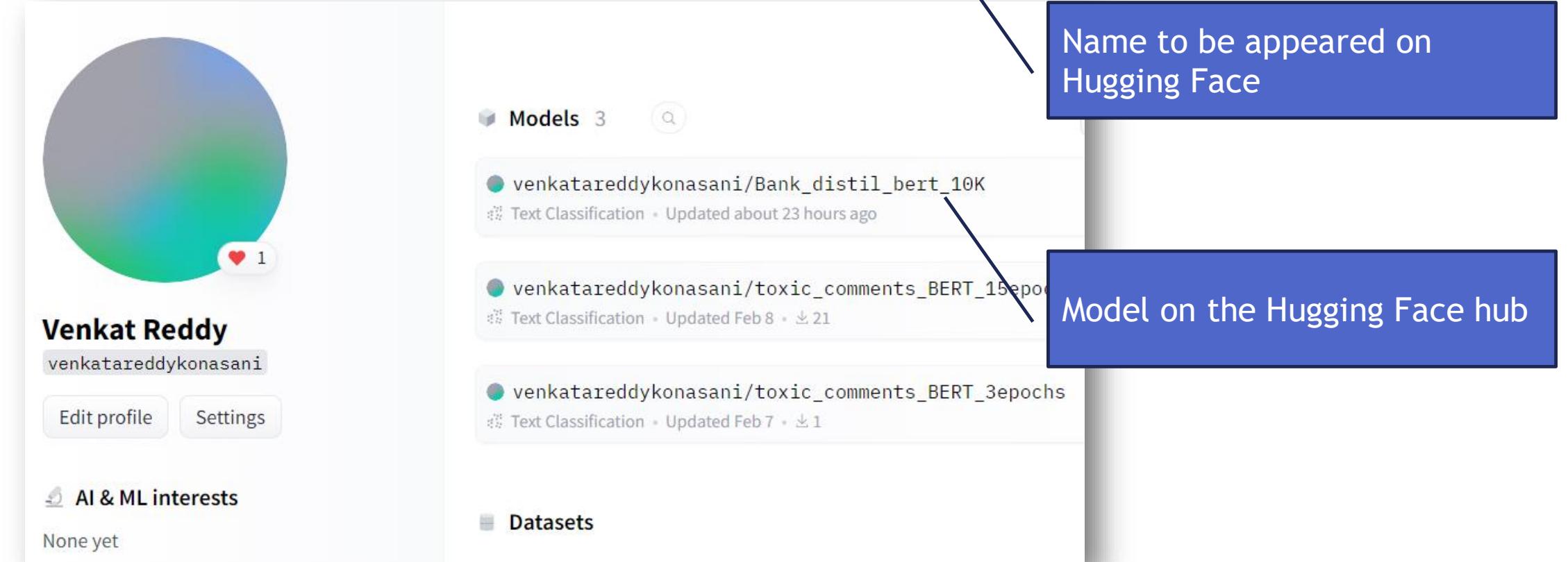
The screenshot shows the 'Access Tokens' section of the DV Analytics interface. It includes the following components:

- A profile card for Venkat Reddy (venkatareddykonasani).
- A sidebar with links: Profile, Account, Authentication, Organizations, Billing, Access Tokens (which is the current page, highlighted with a red box and an arrow), SSH and GPG Keys, Webhooks, and Papers.
- A main content area titled 'User Access Tokens' with the following text:

Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit [the documentation](#) to discover how to use them.
- A token card for 'GenAI_Classes' with a 'WRITE' scope. It shows a redacted token value and buttons for 'Manage' and 'Show'.
- A 'New token' button.

Push the model to HuggingFace hub

```
model.push_to_hub("venkatareddykonasani/Bank_distil_bert_10K")
```



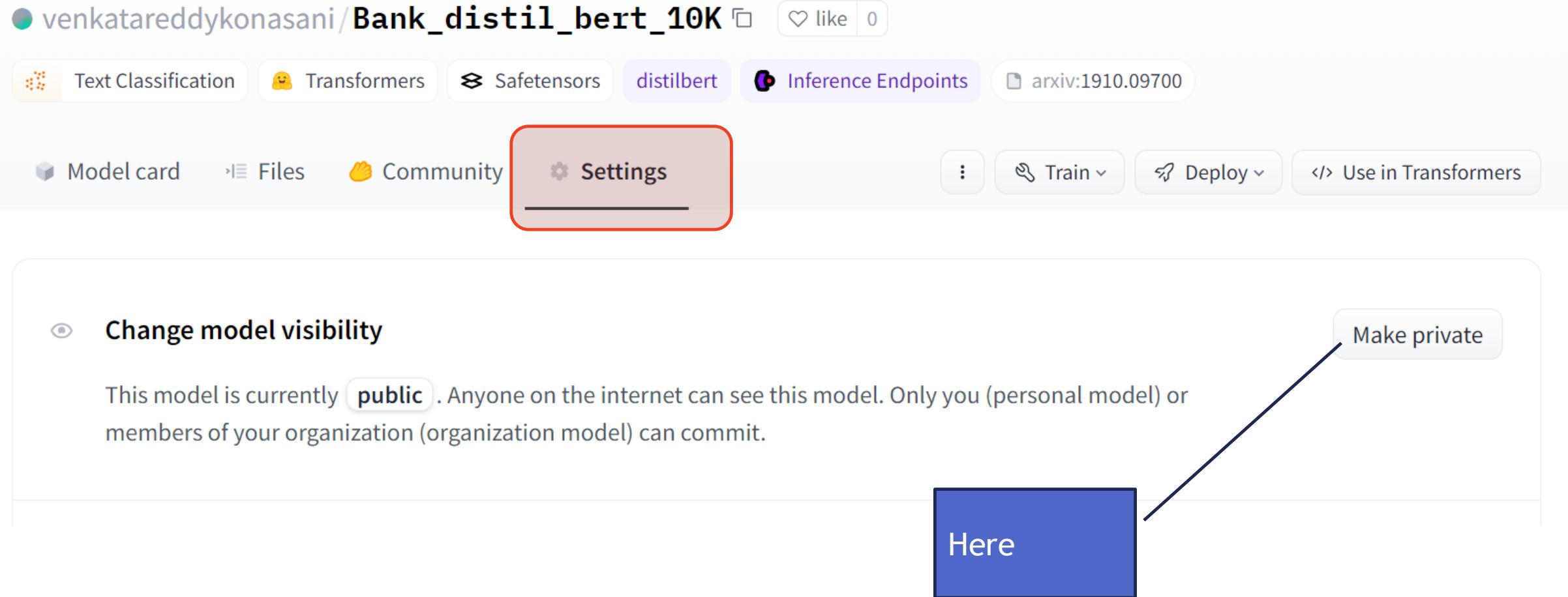
The screenshot shows a user profile for "Venkat Reddy" (venkatareddykonasani). The profile includes a circular profile picture, a "Venkat Reddy" name card with a "1" badge, and two buttons: "Edit profile" and "Settings". Below the profile is an "AI & ML interests" section with the note "None yet". On the right, there's a sidebar with sections for "Models" (containing three items) and "Datasets".

Name to be appeared on Hugging Face

Model on the Hugging Face hub

Model Name	Type	Last Updated
venkatareddykonasani/Bank_distil_bert_10K	Text Classification	About 23 hours ago
venkatareddykonasani/toxic_comments_BERT_15epochs	Text Classification	Feb 8
venkatareddykonasani/toxic_comments_BERT_3epochs	Text Classification	Feb 7

Make the model private



The screenshot shows a model card for "Bank_distil_bert_10K" by venkatareddykonasani. The "Settings" tab is highlighted with a red box. In the "Change model visibility" section, it says the model is currently "public". A blue callout box with the word "Here" points to the "Make private" button, which is also highlighted with a blue box.

venkatareddykonasani/Bank_distil_bert_10K □ like 0

Text Classification Transformers Safetensors distilbert Inference Endpoints arxiv:1910.09700

Model card Files Community Settings Train Deploy Use in Transformers

Change model visibility

This model is currently **public**. Anyone on the internet can see this model. Only you (personal model) or members of your organization (organization model) can commit.

Make private

Here

Loading the model from Hugging Face

```
model=DistilBertForSequenceClassification.from_pretrained("venkatareddykonasani/Bank_distil_bert_10K")
```

```
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
```

Creating a WebApp

Model Deployment on Streamlit

```
%%writefile requirements.txt
streamlit
numpy
pandas
torch
transformers
huggingface_hub
```

Model Deployment on Streamlit

```
import streamlit as st
import numpy as np
import pandas as pd
import torch
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, Trainer,
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
model = DistilBertForSequenceClassification.from_pretrained('venkatareddykonasani/Bank_dist:

st.title("Bank Complaints Categorization")
st.write("Sample Complaints are given below")
Sample_Complaints = [
    {"Sentence": "Credit Report - payment history missing credit report made mistake put ac
    {"Sentence": "Retail Related - forwarded message cc sent friday pdt subject final legal
]
st.table(Sample_Complaints)
user_input = st.text_input("Enter a complaint:")
button=st.button("Classify")
```

Model Deployment on Streamlit

```
d={  
    0: "Credit reporting",  
    1: "Mortgage and Others"  
}  
  
if user_input and button:  
    inputs=tokenizer(user_input, return_tensors="pt")  
    outputs=model(**inputs)  
    predictions=outputs.logits.argmax(-1)  
    predictions=predictions.detach().cpu().numpy()  
    print(predictions)  
    st.write("Prediction :" , d[predictions[0]])
```

Final app

The screenshot shows a web browser window with the title bar "app · Streamlit". The URL in the address bar is "https://grumpy-clouds-suffer.localt". The main content area displays the title "Bank Complaints Categorization" in large bold letters. Below it, a subtitle says "Sample Complaints are given below". A table follows, showing two rows of complaints:

	Sentence
0	Credit Report - payment history missing credit report made mistake put account forbearance without authorization
1	Retail Related - forwarded message cc sent friday pdt subject final legal payment well fargo well fargo clearly wrong need look actually opened account see court hearing several different government agency

Below the table, there is a text input field with the placeholder "Enter a complaint:" containing the text "My credit score is shown totally wrong. Extra loans are shown on my credit report that I have not taken". A "Classify" button is located below the input field. At the bottom, the prediction is displayed as "Prediction : Credit reporting".