Assessing the Vulnerabilities of the Open-Source Artificial Intelligence (AI) Landscape: A Large-Scale Analysis of the Hugging Face Platform

Data Science and Artificial Intelligence Lab at the Kelley School of Business

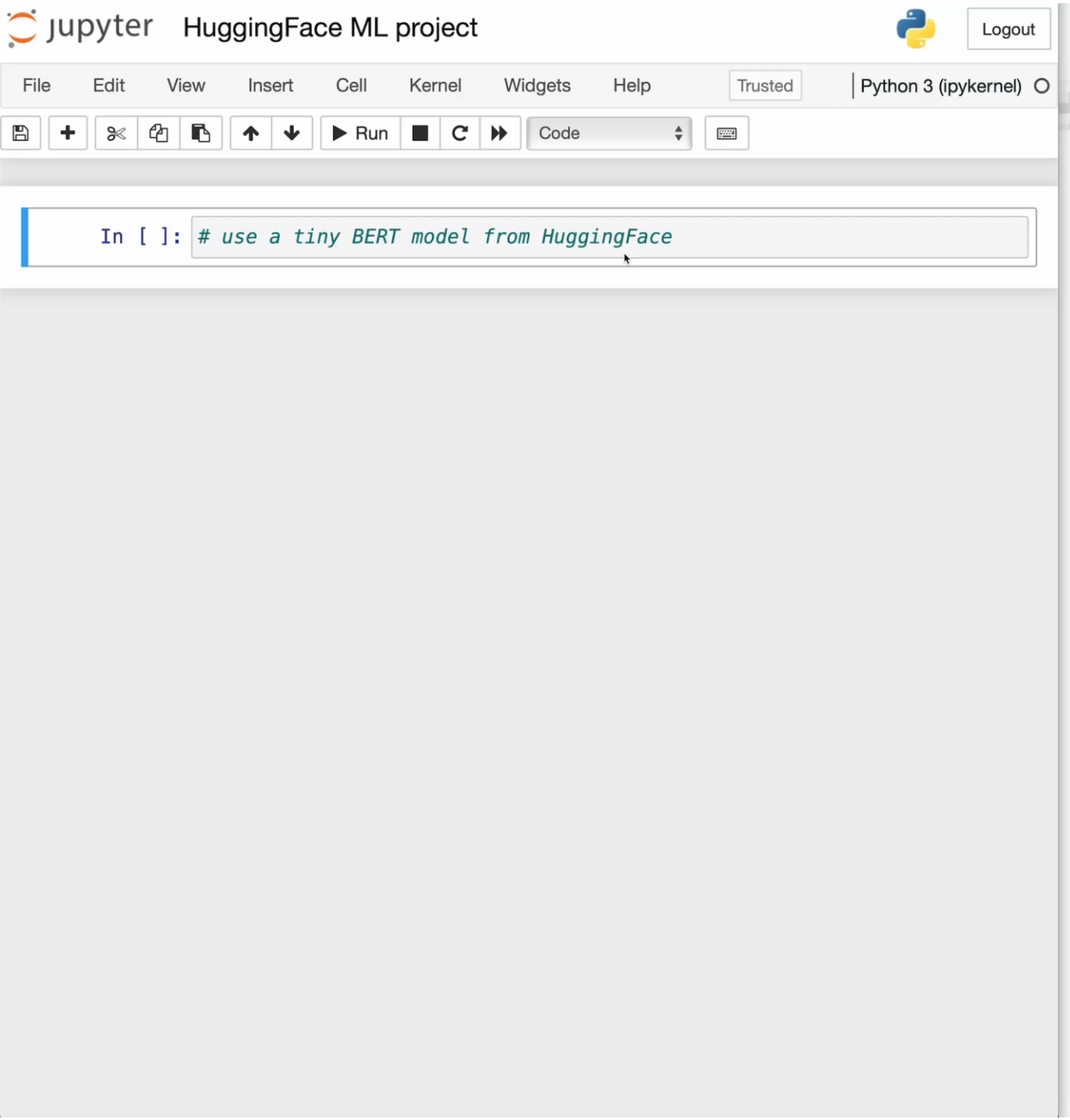
Adhishree Kathikar, Aishwarya Nair, Ben Lazarine, Agrim Sachdeva, Sagar Samtani, Hyrum Anderson (Robust Intelligence)

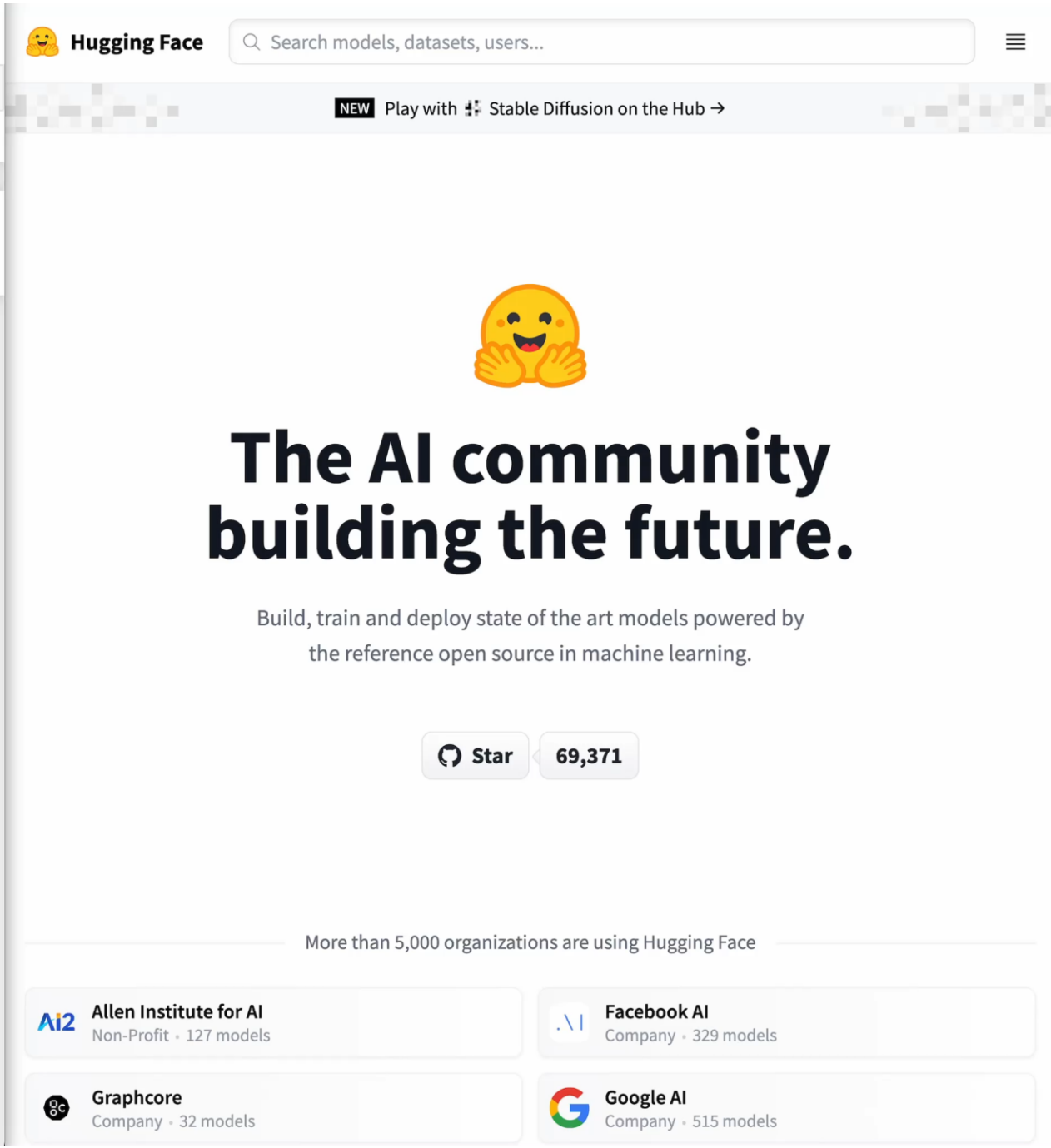
ROBUST INTELLIGENCE



What is the importance of Open-Source AI?


While Hugging Face democratizes access to AI models, these models may contain unknown security vulnerabilities


jupyter
HuggingFace ML project
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
In [ ]:
use a tiny BERT model from HuggingFace


Hugging Face
Search models, datasets, users...
NEW
Play with Stable Diffusion on the Hub →
The AI community building the future.
Build, train and deploy state of the art models powered by the reference open source in machine learning.
Star
69,371
More than 5,000 organizations are using Hugging Face
Allen Institute for AI
Non-Profit • 127 models
Facebook AI
Company • 329 models
Graphcore
Company • 32 models
Google AI
Company • 515 models

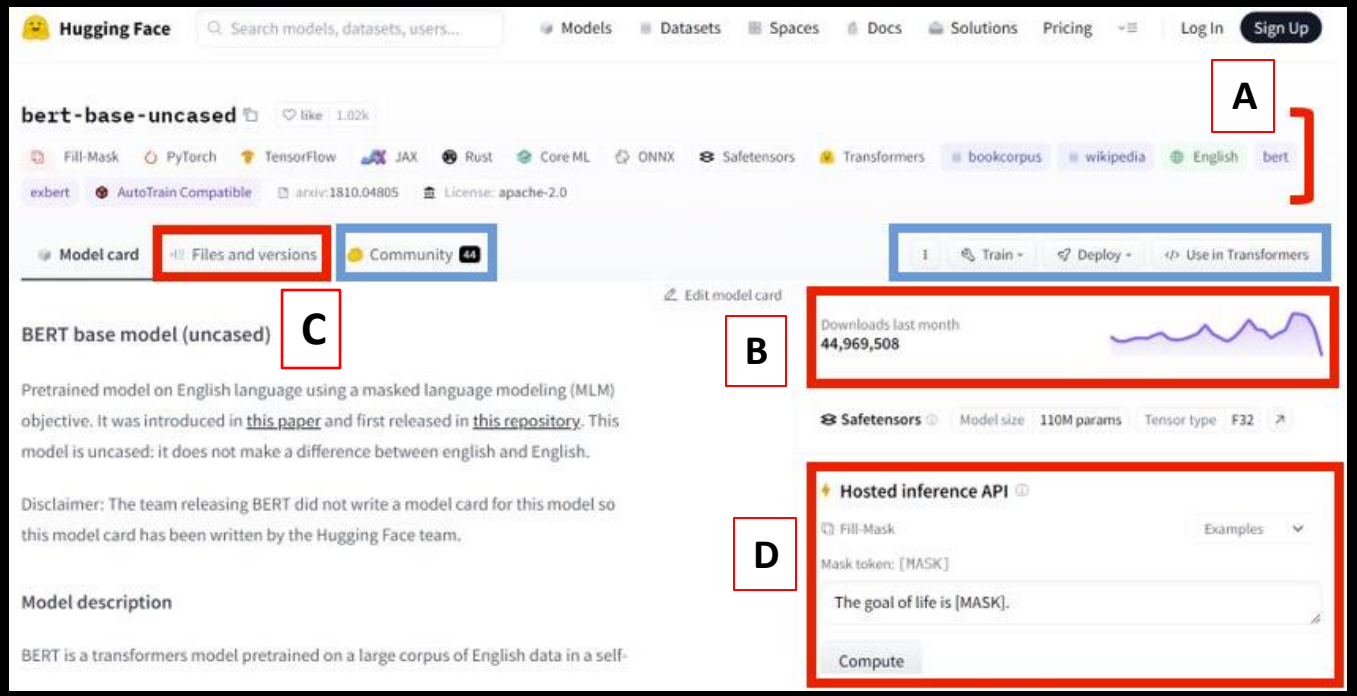
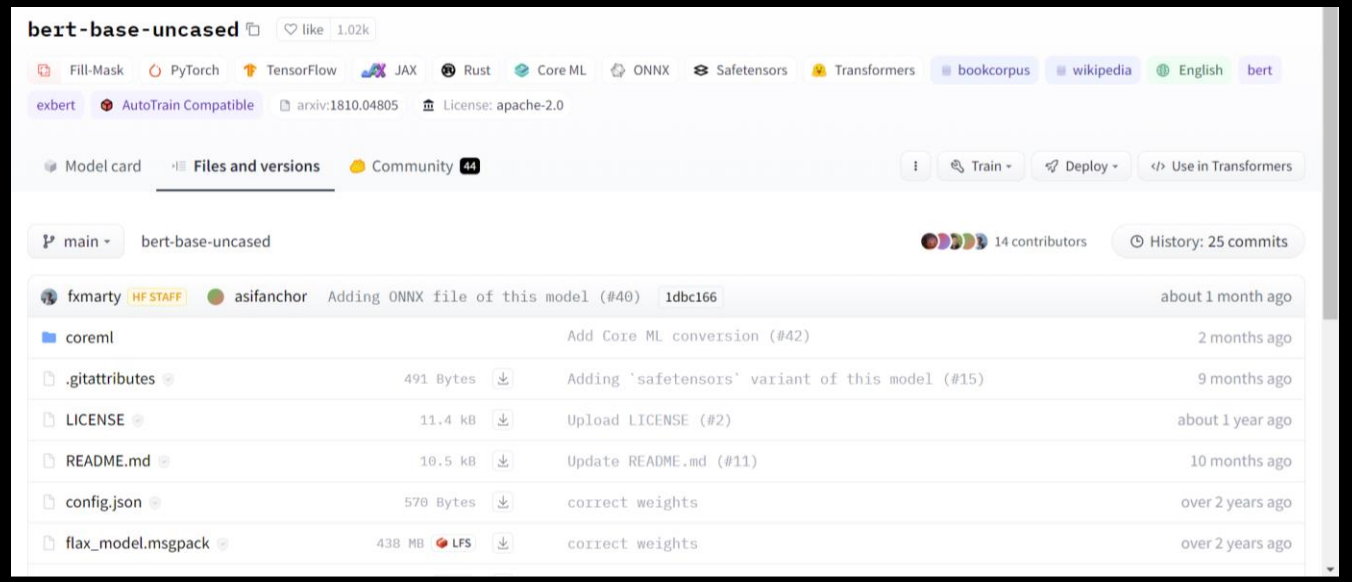


Figure 1: Hugging Face model cards Include:
(a) the basic details of each model including the model name and the model filters,
(b) the number of times the repository has been downloaded in a month,
(c) access model's files and versions,
(d) Hosted Inference API

Figure 2: Files and Versions Section of Model



Why GitHub?

Hugging Face does not provide source code on their website, making it difficult for developers and users to clone and edit models

Hugging Face provides resources through GitHub repositories for developing models before posting onto the Hugging Face platform

Because of GitHub's popularity, **Hugging Face reaches a wider audience**, leading to more code on Hugging Face's platform

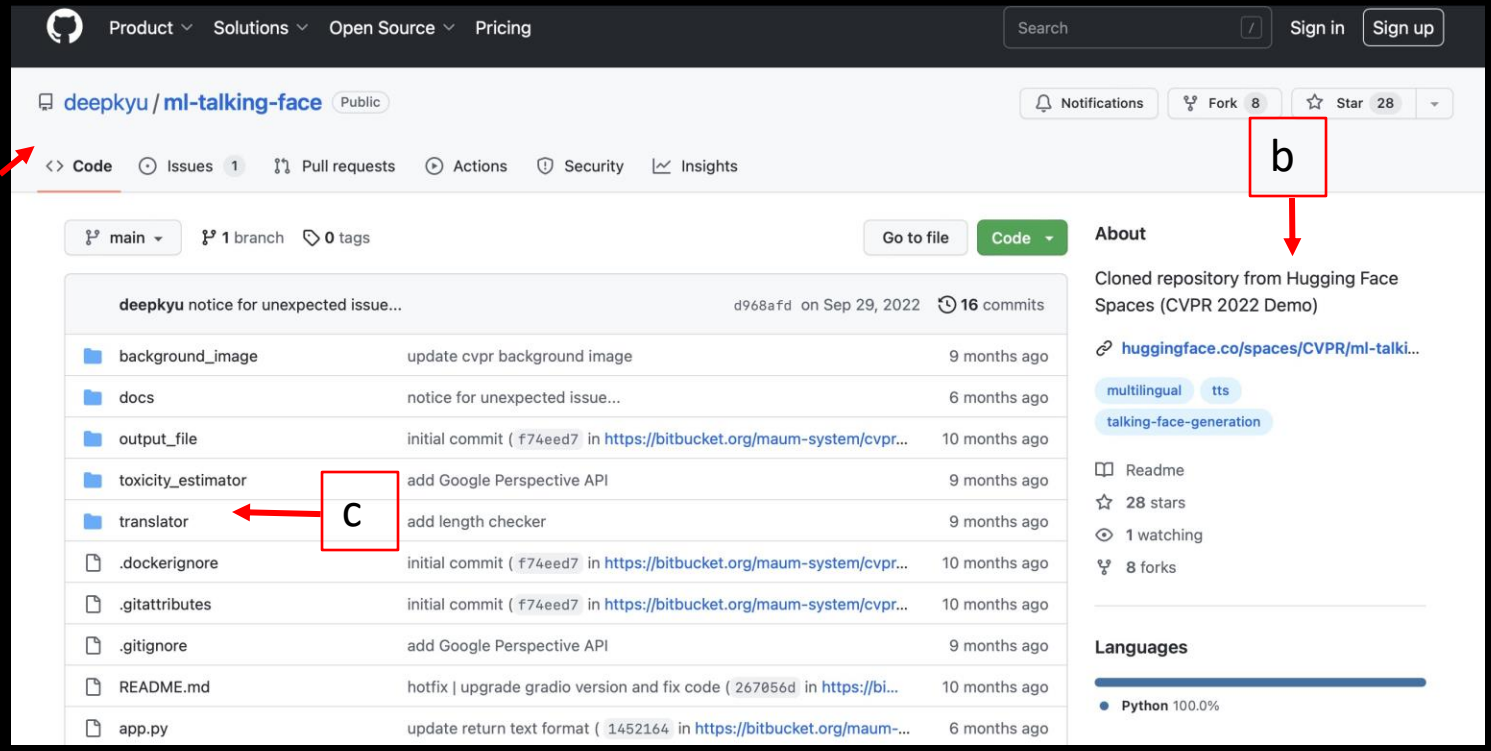
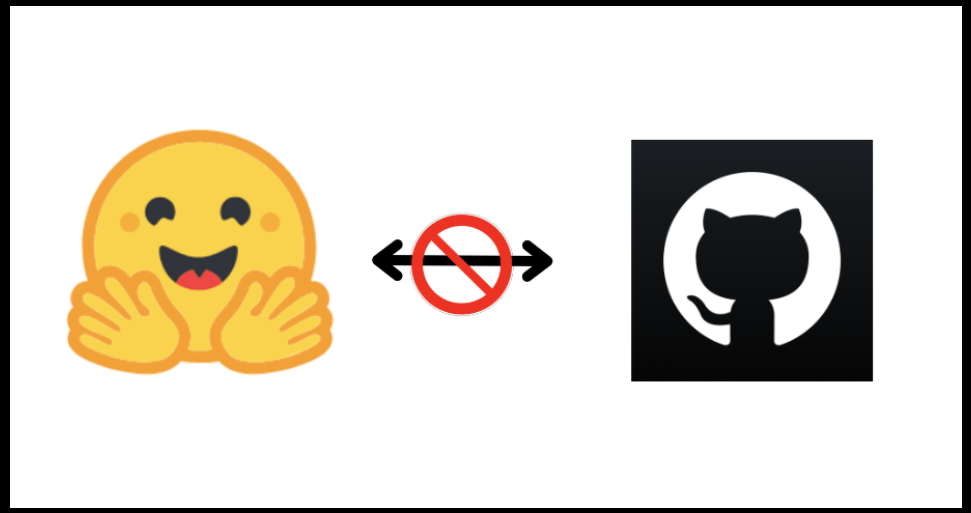


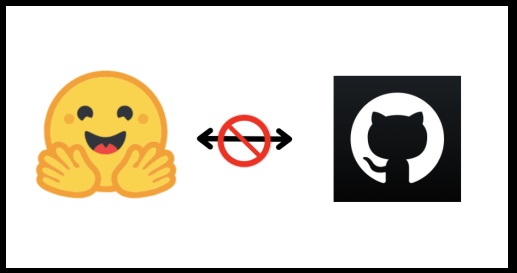
Figure 4: An Example of Hugging Face repository in GitHub
(a) The user's name, research aspect & name,
(b) Details about this repository,
(c) The relative files that the user has made

GitHub and Hugging Face Linkages

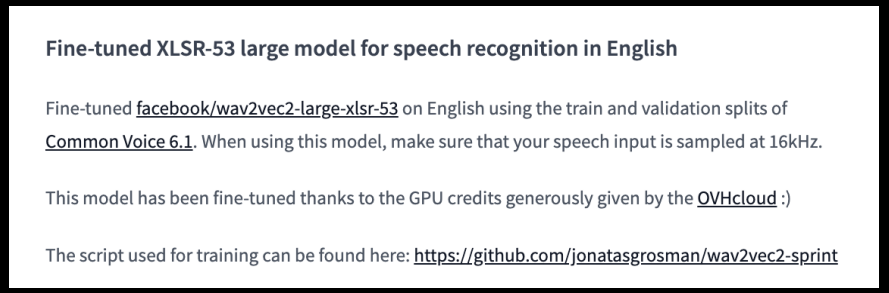


- Aren’t **proper** linkages between models on Hugging Face and their underlying GitHub repositories
- Difficult to **identify vulnerabilities** in the AI models and their code.
- Scale of models on Hugging Face necessitates automated approaches to detect vulnerabilities.

Vulnerability Linkages



- Vulnerabilities between Hugging Face and GitHub could manifest themselves through three types of linkages:
  1. Hugging Face model cards or linkages identified through model card analysis
  2. GitHub readmes or linkages identified by examining mentions of Hugging Face in GitHub documentation readme files
  3. Hugging Face API



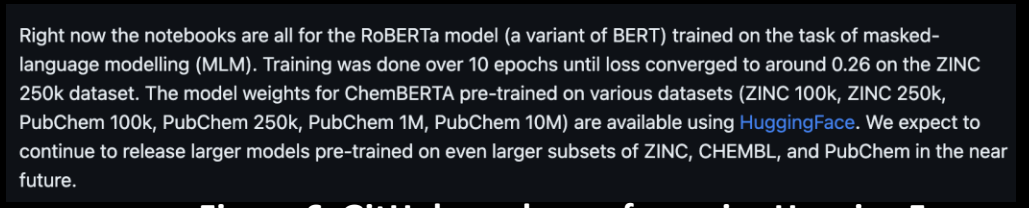
Fine-tuned XLSR-53 large model for speech recognition in English

Fine-tuned facebook/wav2vec2-large-xlsr-53 on English using the train and validation splits of Common Voice 6.1. When using this model, make sure that your speech input is sampled at 16kHz.

This model has been fine-tuned thanks to the GPU credits generously given by the OVHcloud :)

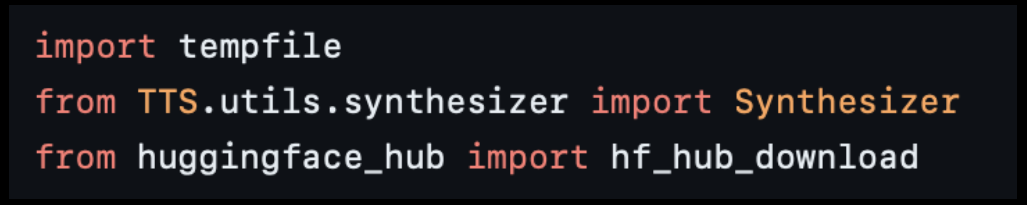
The script used for training can be found here: https://github.com/jonatasgrosman/wav2vec2-sprint

Figure 5: Hugging Face model card referencing GitHub



Right now the notebooks are all for the RoBERTa model (a variant of BERT) trained on the task of masked-language modelling (MLM). Training was done over 10 epochs until loss converged to around 0.26 on the ZINC 250k dataset. The model weights for ChemBERTA pre-trained on various datasets (ZINC 100k, ZINC 250k, PubChem 100k, PubChem 250k, PubChem 1M, PubChem 10M) are available using HuggingFace. We expect to continue to release larger models pre-trained on even larger subsets of ZINC, CHEMBL, and PubChem in the near future.

Figure 6: GitHub readme referencing Hugging Face datasets and models



```
import tempfile
from TTS.utils.synthesizer import Synthesizer
from huggingface_hub import hf_hub_download
```

Figure 7: GitHub source code calling Hugging Face API

Static

- Scans model source code to identify code-based vulnerabilities
- Identifies vulnerabilities such as secrets, insecurities, attacks, and AI-specific vulnerabilities
- e.g. Bandit, FlawFinder, Semgrep

VS

Dynamic

- Scans the compiled pre-trained models themselves to identify model vulnerabilities
- Due to their ability to scan the models themselves, dynamic scanners often provide a richer vulnerability scan
- e.g. Counterfit

Vulnerability Scanners Used

Semgrep

Scans for: secrets, insecurities, attacks, and AI-specific vulnerabilities
However, Semgrep cannot scan cross-functional files and application components.
Some Semgrep rulesets can be used to scan ML models (e.g. trailofbits).

Bandit

Scans for: some secrets, insecurities, and attacks in Python code
However, Bandit does not scan for any AI-specific vulnerabilities.

Flawfinder

Scans for: weak cryptography, file permission, insecure function, and insecure input vulnerabilities
Specifically scans C/C++ code
However, Flawfinder does not scan for any AI-specific vulnerabilities.

Flawfinder

Research Objectives & Questions

Our research **objectives** are to:

1. Collect a large scale of models from the Hugging Face platform
2. Identify linkages between Hugging Face models and their underlying GitHub repositories
3. Perform an automated vulnerability assessment

Through our research we plan to answer the following **questions**:

What Hugging Face models have an underlying GitHub repository with the model's source code?

How can static and/or dynamic vulnerability assessment scanners be leveraged to identify vulnerabilities within the GitHub repositories linked to Hugging Face models?

Proposed Research Framework

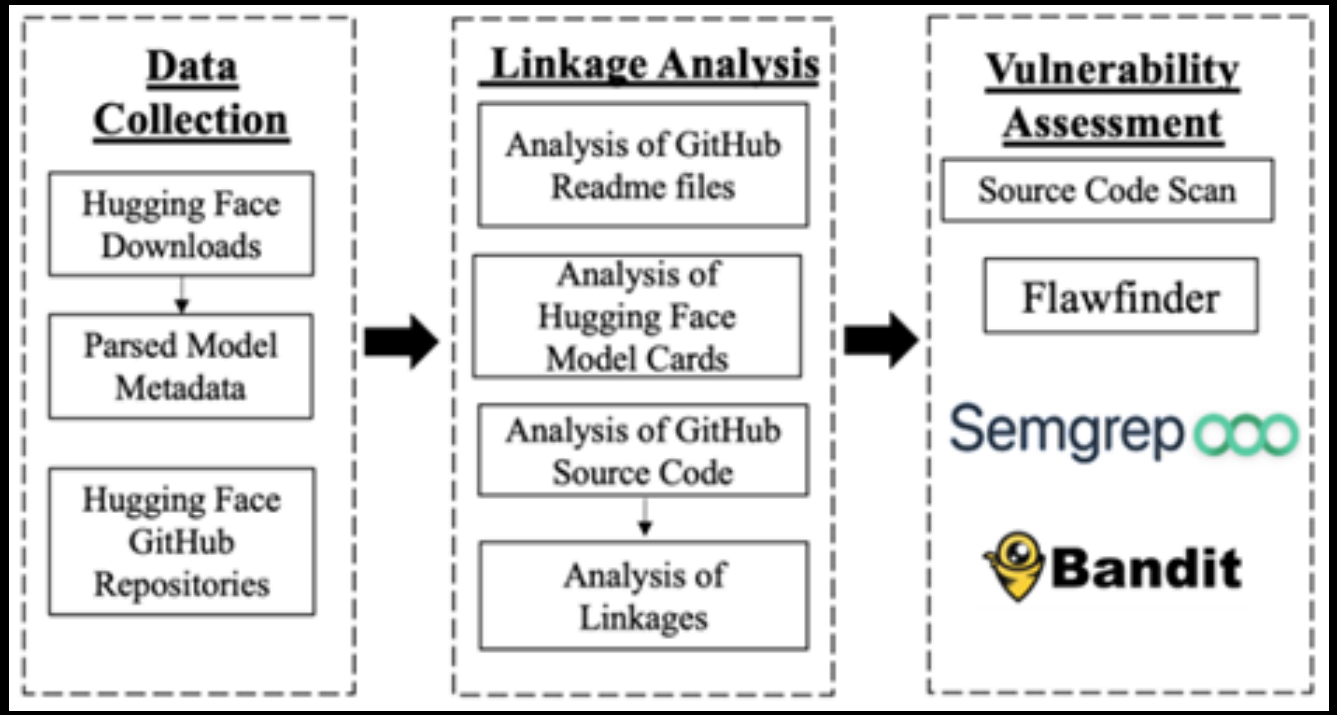


Figure 8: Proposed Research Framework

The aim of our collection process is to **collect pretrained models** on Hugging Face for subsequent vulnerability assessment and analysis.

Our data collection consists of:

- **Model Cards**: Downloaded all associated model cards from Hugging Face **before 02/08** (110,000 model cards)
- **Pre-Trained Models**: Collect all models **before 02/08** from Hugging Face using the Hub Client API (110,000 pre-trained models)
- **GitHub Hugging Face Repositories**: Collect root, forked, and searched repositories from Hugging Face, which we will explain further in our presentation (29,000+ GitHub repositories)

Hugging Face Category Breakdown

- Hugging Face separates their models based on tasks that fall under 6 large categories
- The majority of models are NLPs
- Understanding this breakdown helps us determine the various vulnerabilities in each category

Categories	Description	Tasks Associated	# of Models	Top 5 Datasets Used	# Datasets
NLP	Category is focused on text actions and understanding context in sentences.	Translation, Fill-Mask, Token Classification	56,863	Glue, squad, common_voice, Wikipedia, mozilla-foundation/common_voice_11_0	390
RL	This category is usually utilized in the video games where the AI model continuously gets better at the game the more it plays.	Reinforcement Learning, Robotics	11,009	0 datasets were used here.	0
Audio	Category is focused on audio and determining voice activity.	Automatic Speech Recognition, Audio Classification,	7,357	Common_voice, Wikipedia, mozilla-foundation/common_voice_	21
Multimodal	Uses the above modes of image classification, NLP, and audio in unison.	Feature Extraction, Text-to-Image	5,229	Glue, squad, Wikipedia, imagenet-1k, bookcorpus	31
Computer Vision	focused on images and videos determine depth and can classify images into different categories.	Image Classification, Image Segmentation, Image Classification	3,220	mozilla-foundation/common_voice_7_0, imagenet-1k,	11
Tabular	This category provides provide statistical analysis from tables.	Tabular Regression, Tabular Classification	167	Gustavosta/Stable-Diffusion-Prompts	1

Table 1: Breakdown of Hugging Face model category

Linkage Analysis Results

Linkage	Total # of Linkages	Total % of Collection	Category	# of Linkage Per Category	% of Linkage Per Category
HF Model Cards	9,562	9%	Multimodal	636	12%
			NLP	5,629	10%
			Audio	843	11%
			Tabular	15	9%
			RL	2,039	19%
			CV	400	12%
GitHub Readme files	5,192	18%	Searched and Root	739	67%
			Forked	4,453	16%

Table 2: Linkages between GitHub and Hugging Face

- **Hugging Face model cards** which mentioned GitHub repositories: 9,562
  - 9% of our collection of 110,000 models
- **GitHub readmes** that mentioned Hugging Face models: 5,192
  - 18% of our GitHub repository collection had a linkage to Hugging Face
  - 67% of search and root repositories link to Hugging Face
  - 16% of forked repositories link to Hugging Face
- **Linkages of Hugging Face API** are part of our future steps

GitHub Repository Collection and Vulnerability Assessment

- Our scan consists of **29,168** repositories identified across three categories :
- 111 **root** repositories
  - Repositories posted by HuggingFace on GitHub
  - Contain foundational and supplementary repositories, datasets, and toolkits
    - Only foundational (5) and supplementary (40) repositories store source code
- 28,067 **fork** repositories
  - Forked from root repositories
- 990 **searched** repositories
  - Found with the Keyword search: "huggingface" via the GitHub API

- We categorize vulnerabilities based on vulnerability severities and Common Weakness Enumerations (CWEs)
  - Help determine how developers should prioritize identified vulnerabilities and determine what kinds of vulnerabilities are found

Type of Repository	Vulnerability Severity			Total Vulnerabilities
	High	Medium	Low	
Root	689	1,096	130	1,915
Forked	537,815	472,304	4,824,765	5,834,884
Searched	5,987	7,683	66,229	79,899
Totals	544,491	481,083	4,891,124	5,916,698

Table 3: Vulnerability Severities of GitHub Searched, Fork, and Root Repositories

Root: 6.79% low-severity; 57.23% medium-severity; 35.98% high-severity
Fork: 82.69% low-severity; 8.09% medium-severity; 9.22% high-severity
Searched: 82.89% low-severity; 9.62% medium-severity; 7.49% high-severity

Root repositories have a smaller percentage of vulnerabilities classified as low-severity, while both searched and fork have the greatest percentages of low-severity vulnerabilities

Low-severity vulnerabilities may have developed and persisted in the development of new repositories

Repository Type	Vulnerability	Occurrences	Distinct Repositories	Vulnerability Definition	Top Vulnerable Repository	Vulnerability Occurrences
Root (Foundational & Supplemental)	CWE-502	734	10	Warning against using pickle and recommends serializing to avoid arbitrary code	transformers	516
	CWE-676	208	10	PyTorch memory is not automatically pinned to restrict access from uncertified personnel	transformers	174
	CWE-319	126	2	Sensitive data (e.g., passwords) is sent in plain text which is easily accessed	transformers	126
Fork	CWE-502	661,154	241,207	Warning against using pickle and recommends serializing to avoid arbitrary code	zhangxiangxiao/tokenizers	26
	CWE-676	252,634	131,244	PyTorch memory is not pinned automatically to restrict access from uncertified personnel	zh-plus/accelerate	14
	CWE-532	36,024	21,954	Sensitive information (e.g., passwords) used in debugging code	wise-east/transfer-learning-cov-ai	6
Searched	CWE-703	53,786	9,156	The external function does not account for/handle exceptional conditions that may occur	aws/sagemaker-python-sdk	311
	CWE-502	4,419	2,094	Warning against using pickle and recommends serializing to avoid arbitrary code	huggingface/tokenizers	26
	CWE-259	3,083	2,291	Sensitive information (e.g., passwords) embedded into source code or files	D-Yifan/AgileLightning	17

Table 4: Main CWEs Identified for Different Repository Types

Top 3 Vulnerabilities Detected Across All Repositories Scanned:
CWE-502 = Count: 667,977
CWE-676 = Count: 254,604
CWE-703 = Count: 53,786

Repository Type	Vulnerability	Occurrences	Distinct Repositories	Vulnerability Definition	Top Vulnerable Repository	Vulnerability Occurrences
Root (Foundational & Supplemental)	CWE-502	734	10	Warning against using pickle and recommends serializing to avoid arbitrary code	transformers	516
	CWE-676	208	10	PyTorch memory is not automatically pinned to restrict access from uncertified personnel	transformers	174
	CWE-319	126	2	Sensitive data (e.g., passwords) is sent in plain text which is easily accessed	transformers	126
Fork	CWE-502	661,154	241,207	Warning against using pickle and recommends serializing to avoid arbitrary code	zhangxiangxiao/tokenizers	26
	CWE-676	252,634	131,244	PyTorch memory is not pinned automatically to restrict access from uncertified personnel	zh-plus/accelerate	14
	CWE-532	36,024	21,954	Sensitive information (e.g., passwords) used in debugging code	wise-east/transfer-learning-cov-ai	6
Searched	CWE-703	53,786	9,156	The external function does not account for/handle exceptional conditions that may occur	aws/sagemaker-python-sdk	311
	CWE-502	4,419	2,094	Warning against using pickle and recommends serializing to avoid arbitrary code	huggingface/tokenizers	26
	CWE-259	3,083	2,291	Sensitive information (e.g., passwords) embedded into source code or files	D-Yifan/AgileLightning	17

Table 4: Main CWEs Identified for Different Repository Types

Based on our results, **502** and **676** were the **top** two CWEs identified in **root** and **forked** repositories.

- CWE-502 detects the use of pickling within the code
  - Possible arbitrary code during unpickling
  - The forked repo with highest CWE-502 vulnerabilities is forked from tokenizers, which had the second highest number of occurrences (164) of CWE-502
- CWE-676 is identified if the code does not automatically pin PyTorch memory to secure the memory's access from uncertified users
  - Can cause attackers to access information within these repositories
  - Accelerate library can train Transformers models
    - Possible vulnerability propagation

Repository Type	Vulnerability	Occurrences	Distinct Repositories	Vulnerability Definition	Top Vulnerable Repository	Vulnerability Occurrences
Root (Foundational & Supplemental)	CWE-502	734	10	Warning against using pickle and recommends serializing to avoid arbitrary code	transformers	516
	CWE-676	208	10	PyTorch memory is not automatically pinned to restrict access from uncertified personnel	transformers	174
	CWE-319	126	2	Sensitive data (e.g., passwords) is sent in plain text which is easily accessed	transformers	126
Fork	CWE-502	661,154	241,207	Warning against using pickle and recommends serializing to avoid arbitrary code	zhangxiangxiao/tokenizers	26
	CWE-676	252,634	131,244	PyTorch memory is not pinned automatically to restrict access from uncertified personnel	zh-plus/accelerate	14
	CWE-532	36,024	21,954	Sensitive information (e.g., passwords) used in debugging code	wise-east/transfer-learning-cov-ai	6
Searched	CWE-703	53,786	9,156	The external function does not account for/handle exceptional conditions that may occur	aws/sagemaker-python-sdk	311
	CWE-502	4,419	2,094	Warning against using pickle and recommends serializing to avoid arbitrary code	huggingface/tokenizers	26
	CWE-259	3,083	2,291	Sensitive information (e.g., passwords) embedded into source code or files	D-Yifan/AgileLightning	17

Table 4: Main CWEs Identified for Different Repository Types

CWE-**703** is only found in **searched** repositories.

- We can determine that this CWE was developed by individual repositories and not inherited from root repositories
- CWE-703 is identified if the code does not handle exceptional conditions

Main Takeaways

- Collected 110,000 Hugging Face models and parsed the associated model cards to understand the main categories of models on Hugging Face
- Identified linkages between GitHub repositories and Hugging Face models
- Scanned linked GitHub repositories for vulnerabilities
- Discovered that while a majority of the vulnerabilities detected in the root repositories were high-severity, the majority of vulnerabilities in the forked and searched repositories were low-severity
- Identified the common vulnerability types (CWE-502, CWE-676, CWE-703)

AI Risk Database

Report Vulnerability

Hyrum Anderson

0:00

Explore AI Supply Chain Risk with the AI Risk Database

AI Risk Database is a tool for discovering and reporting the risks associated with public machine learning models. The database is specifically designed for organizations that rely on AI for their operations, providing them with a comprehensive and up-to-date overview of the risks and vulnerabilities associated with publicly available models.

Our database is continuously updated with the latest models, file reputation, and model vulnerabilities to ensure that you have the most accurate and up-to-date information at your fingertips.

Search by model name or URL...

Report a Vulnerability

Tell us about an AI vulnerability that you've discovered.

Top Risk Reports

Top Public Models

Future Directions

- Incrementally collect models on the Hugging Face platform to get a deeper understanding of the overall Hugging Face landscape
- Further breakdown categories of linkages between GitHub and Hugging Face (Hugging Face API model calls, GitHub repositories for training Hugging Face models, etc.)
- Start case study and analyze how the connections between Hugging Face models and GitHub repositories can propagate AI vulnerabilities through both platforms.
- Integrating our open-source vulnerability assessment capabilities with MITRE's AI Risk Database

Questions?

Adhishree Kathikar
akathika@iu.edu

Aishwarya Nair
aishnair@iu.edu

Sagar Samtani
ssamtani@iu.edu