10-2024

What Do We Know About Hugging Face? A Systematic Literature Review and Quantitative Validation of Qualitative Claims

Jason Jones
Purdue University

Wenxin Jiang
Purdue University

Nicholas Synovic
nsynovic@luc.edu

George K. Thiruvathukal
Loyola University Chicago, gkt@cs.luc.edu

James C. Davis
Purdue University, davisjam@purdue.edu

Author Manuscript

This is a pre-publication author manuscript of the final, published article.

Recommended Citation

Jones, Jiang, Synovic, Thiruvathukal, and Davis. What do we know about Hugging Face? A systematic literature review and quantitative validation of qualitative claims. Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) 2024.

What do we know about Hugging Face? A systematic literature review and quantitative validation of qualitative claims

Jason Jones
jone2078@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Wenxin Jiang
jiang784@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Nicholas Synovic
nsynovic@luc.edu
Loyola University Chicago
Chicago, Illinois, USA

George K. Thiruvathukal
gthiruv@luc.edu
Loyola University Chicago
Chicago, Illinois, USA

James C. Davis
davisjam@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Abstract

Background: Software Package Registries (SPRs) are an integral part of the software supply chain. These collaborative platforms unite contributors, users, and packages, and they streamline package management. Much engineering work focuses on synthesizing packages from SPRs into a downstream project. Prior work has thoroughly characterized the SPRs associated with traditional software, such as NPM (JavaScript) and PyPI (Python). Pre-Trained Model (PTM) Registries are an emerging class of SPR of increasing importance, because they support the deep learning supply chain. *Aims*: A growing body of empirical research has examined PTM registries from various angles, such as vulnerabilities, reuse processes, and evolution. However, no existing research synthesizes them to provide a systematic understanding of the current knowledge. Furthermore, much of the existing research includes unsupported qualitative claims and lacks sufficient quantitative analysis. Our research aims to fill these gaps by providing a thorough knowledge synthesis and use it to inform further quantitative analysis.

Methods: To consolidate existing knowledge on PTM reuse, we first conduct a systematic literature review (SLR). We then observe that some of the claims are qualitative and lack quantitative evidence. We identify quantifiable metrics assoiated with those claims, and measure in order to substantiate these claims. *Results*: From our SLR, we identify 12 claims about PTM reuse on the HuggingFace platform, 4 of which lack quantitative validation. We successfully test 3 of these claims through a quantitative analysis, and directly compare one with traditional software. Our findings corroborate qualitative claims with quantitative measurements. Our two most notable findings are: (1) PTMs have a significantly higher turnover rate than traditional software, indicating a dynamic and rapidly evolving reuse environment within the PTM ecosystem; and (2) There is a strong correlation between documentation quality and PTM popularity. *Conclusions*: Our findings validate several qualitative research claims with concrete metrics, confirming prior qualitative and case study research. Our measures show further dynamics of PTM reuse, motivating further research infrastructure and new kinds of measurements.

1 Introduction

As the size and cost of developing deep learning (DL) models from scratch continue to rise, engineers are increasingly turning to adapt open-source Pre-trained Models (PTMs) as a cost-effective alternative [31]. PTM registries facilitate the reuse of open-source models by providing packages that include pre-trained weights, configuration, and documentation [29]. Hugging Face has become a prominent PTM registry, comparable in popularity to traditional software registries like NPM and PyPI [29].

Prior research has empirically compared PTM registries to traditional software package registries such as NPM and PyPI, tackling diverse issues such as carbon emissions, model selection, and vulnerabilities [14, 29, 33]. Despite these efforts, no work has yet synthesized the existing knowledge of PTM registries. Furthermore, not all topics about PTM registries such as Hugging Face have been studied from both qualitative and quantitative perspectives. This lack of a comprehensive approach has led to a lack of quantitative validation, leaving numerous qualitative insights about PTM registries under-verified.

Our work aims to bridge the knowledge gap concerning the synthesis of extensive research on PTM reuse facilitated by Hugging Face, and provide further measurements to validate the prior qualitative and quantitative claims. An overview is shown in Figure 1. Our work consists of two parts: First, we conduct a systematic literature review to transform qualitative insights about PTM package reuse on Hugging Face into quantitative metrics. Second, we evaluate the robustness of these insights through quantitative validation. This approach enhances our understanding of package reuse dynamics on Hugging Face compared to traditional SPRs.

Our findings indicate that most of the claims from prior qualitative results are correct. Of the 3 prior claims we quantitatively evaluated, 2 were supported by our results, and 1 was supported by 1 measurement and not supported by the other. Our measurement of library usage indicated a preference for using the `Transformers` library when creating descendents of models, with 80% of descendents of models choosing to utilize `Transformers`. Our measurement of package turnover indicates that HuggingFace has a significantly higher turnover rate than traditional package registries. Our measurement of model popularity and descendent count found a correlation between the two, further supporting claims that popularity is a driver of model selection. Finally, we found a strong correlation between documentation quality and PTM popularity, with the top 1000 models outperforming the bottom 1000 in documentation.

Our contributions are:

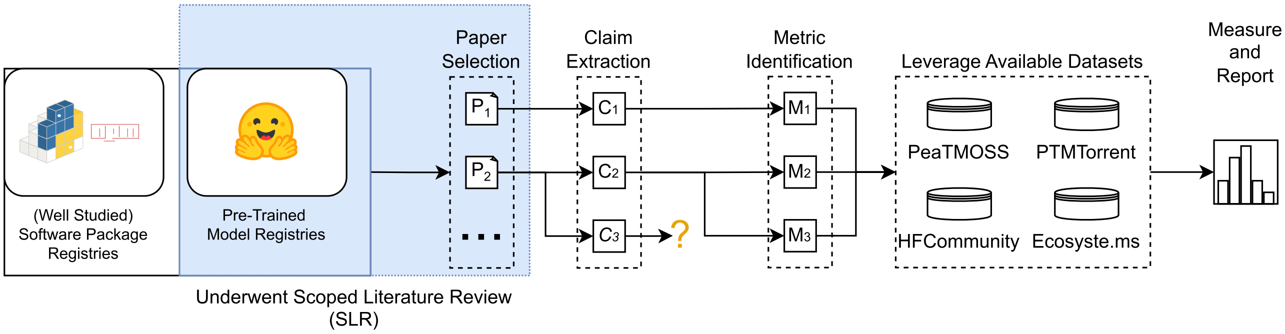

Figure 1: Overview of this work's context and approach. Much is known, both qualitatively and quantitatively, about package reuse processes in traditional software package registries such as NPM and PyPI. Meanwhile, empirical data about the reuse of pre-trained deep neural network models (PTMs) is emerging. This work provides the first systematic literature review on PTM reuse, focused on extracting the claims present in the prior work (RQ1) and providing quantitative evaluation of the un-quantified and under-quantified claims (RQ2).

- We conduct a systematic literature review on PTM reuse in the Hugging Face registry, and extract a list of qualitative claims from prior work. (§4)
- We map the qualitative claims to quantitative measurements. We use these measurements to validate the prior findings, via comparison of our quantitative measurements of Hugging Face to representative traditional SPRs. (§5)
- Our work provides recommendations for future work on developing tools to analyze and keeping datasets updated to support further investigations on the PTM supply chain. (§7)

Significance for Software Engineering: Prior work has made both quantitative and qualitative claims about software engineers' reuse of PTMs. We systematically synthesized this knowledge through a literature review and developed quantifiable metrics to corroborate the qualitative claims. Our findings confirm several qualitative results about PTM reuse, and also quantify the dynamic reuse environment within the PTM ecosystem. Our results will inform research infrastructure and the development of new metrics to guide and refine PTM development and reuse.

2 Background and Related Work

In this section, we first explain how software package registries facilitate the reuse of software (§2.1). We then discuss the rise of Deep Neural Networks (DNNs) and their presence in software engineering reuse as PTM packages (§2.2). We also detail how software package registries have been measured in previous works, and the limited effort towards quantitatively evaluating PTM registries (§2.3). Our work aims to advance the state of the art when it comes to quantitative evaluation of PTM registries.

2.1 Software Package Registries

Figure 2 depicts the software supply chain. Software Package Registries (SPRs) serve as collaborative hubs that connect package contributors, reusers, and the packages themselves, facilitating software reuse. These registries are important for engineers because they provide comprehensive information that significantly enhances downstream software development.

SPRs act as platforms for creators to upload and share their software packages, featuring version control to allow users to access previous package versions as needed [1, 4]. These platforms promote package discoverability and perceived quality through user engagement tools like comments, likes, and ratings. Additionally, collaboration is fostered by discussion threads and version management, enhancing user interaction and visibility. Registries also provide comprehensive metadata, improving package visibility in search results and aiding in package selection. Tools for package downloading, bundling, and managing versions and updates streamline the reuse process, simplifying package lifecycle management.

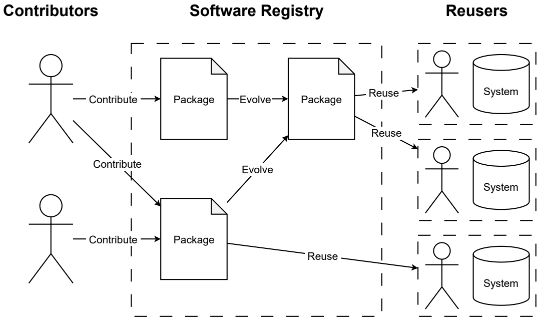

Figure 2: The Software Supply Chain. Engineers contribute Packages to model registries, Packages evolve through internal development and external dependence on other Packages, and are used by downstream Reusers who incorporate them into Systems.

Recently, PTM package registries (*e.g.,* Hugging Face [57], PyTorch Hub [5], ONNX Model Zoo [2]) emerged to support efficient development of AI systems [15, 31]. PTM packages include traditional components (*e.g.,* documentation, dependencies). However, they also include additional DNN-specific components, such as pre-trained weights, training dataset, and model architecture [29]. Figure 3 depicts the structural similarity in package reuse between traditional and PTM packages.

2.2 Deep Neural Networks and Pre-Trained Model Packages

Deep Neural Networks (DNNs), comprising numerous hidden layers, have gained popularity as a cutting-edge solution for complex problems in various fields, such as image recognition in autonomous

Table 1: Example metrics used to characterize traditional software package registries. The metrics in this table are used as guidance when developing metrics to measure the claims in Table 3.

Metric	Description & Implication	Package Registries	Example Works
User Reach	The amount of ecosystem controlled by a fraction of its maintainers.	NPM, PyPI, Cargo, Elm, CRAN	[7, 9, 10, 24, 36, 63, 64]
Dependency Degree	The number of dependencies among software packages.	NPM, PyPi, Cargo, CPAN, CRAN, NuGet, Packageist, RubyGems, Maven	[9, 10, 16, 17, 24, 35, 36, 44, 49, 55, 58, 60–62, 64]
Popularity	The frequency of use for top packages, indicating the concentration of usage.	NPM, PyPI, OpenStack	[18, 55, 62]
Technical Lag	The delay in adopting new updates.	NPM, PyPI, Maven, Cargo	[16, 44, 49, 55, 60, 61]

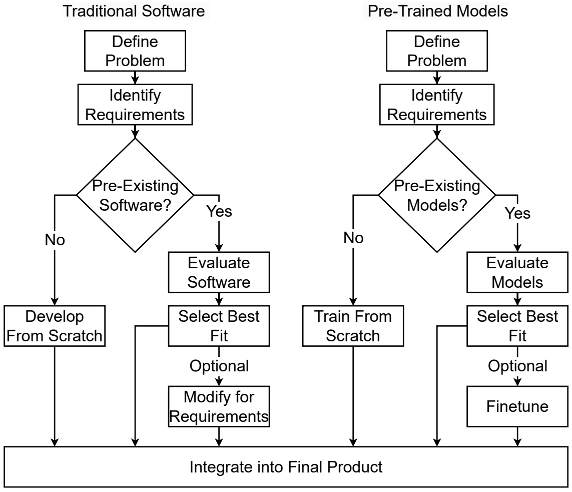

Figure 3: Reuse processes of traditional software and of PTMs, as reported by Jiang *et al.* [29]. Reuse processes are similar, suggesting that SPR measurements and trends may be similar.

vehicles [19] and AI voice assistant systems [41]. Developing and training these models from scratch requires significant time and resources [25, 42, 45]. For example, a `Llama-2-70B` model needs 1720K GPU hours to train from scratch [52]. To overcome this, engineers increasingly opt for using Pre-Trained Models (PTMs), which allows them to forego the lengthy and resource-intensive initial training phase. By leveraging PTMs, they can focus on a much shorter training period to fine-tune the models for specific tasks [15, 27, 29].

Within the context of PTMs, the term "software reuse" refers to the reuse of models along with their training configurations and weights, *i.e.,* reuse of the PTM packages. Figure 3 shows the comparison of reuse process between PTM and traditional packages. Reusing DNNs as PTM packages mirrors traditional practices of software package reuse, where existing software components are integrated and adapted rather than built from scratch [21, 31]. This practice can enhance efficiency and reducing development time [23]. This adaptation involves not only the models but also their pre-trained states, which can be adjusted for specific applications, thereby constituting a combination of model reuse and customization rather than traditional software package reuse alone.

2.3 Measurements of Software Package Registries

Research on software package registries has traditionally focused on quantifying various aspects [16, 20, 61]. However, PTM registry research has primarily focused on qualitative and small-scale quantitative measurements.

2.3.1 General Comparisons: The measurement of software package registries is an established research area within traditional software. Table 1 shows the metrics used in prior work *e.g.,* user reach [64], license [20], and technical lag [16]. These studies provide critical insights for software engineers, aiding in effective package selection and reuse, and for researchers, deepening understanding of software supply chains and registry dynamics.

This research approach has been adapted to PTM registries recently [28, 29]. Early studies have explored how PTMs are reused and adapted, examining both qualitative and quantitative aspects. For instance, research has started to analyze contribution patterns and the reuse dynamics specific to PTM registries, such as those found in TensorFlow Hub and Hugging Face [15, 29, 37]. However, there are still gaps in understanding the evolution and reuse patterns in the PTM registries [29, 32]. Our work bridges this gap by providing additional quantitative measurements and comparing our results to traditional SPRs.

2.3.2 To Explain or Quantify Phenomena: Despite these efforts, there remains a gap in research that connects qualitative observations with quantitative measurements across these registries.

Recent studies within traditional SPRs typically start with qualitative observations that are later supported by quantitative data. Issues such as package obsolescence and the spread of vulnerabilities, initially observed quantitatively, have been rigorously quantified in ecosystems like NPM [6, 40]. These investigations form a crucial basis for comprehending how features of software registries impact the practices of software development.

In the research domain of PTM registries, similar investigations are needed. For example, Jiang *et al.* provide a comprehensive analysis on the risks while using PTMs, and the PTM reuse process [29, 31]. As a follow-up work, they also collected both qualitative and quantitative data on PTM naming practices [28]. This line of research is crucial for understanding PTM package registries. However, their qualitative insights have not been substantiated by quantitative analysis. We address this gap by conducting a systematic literature review and deriving quantifiable metrics to assess the claims from previous studies.

3 Knowledge Gap and Research Questions

To summarize the knowledge gap, we lack a cohesive understanding on PTM package reuse that synthesizes prior research on Hugging Face. Additionally, there is a lack of some quantitative measurements, which reduce our understanding of the registry.

To address the gap of knowledge synthesis, we ask:

RQ1 What claims about package reuse on Hugging Face are made by prior research?

By gathering and analyzing these claims, we aim to convert the qualitative data into a set of quantifiable metrics. This effort will not only enrich our understanding of Hugging Face as a PTM platform but also enhance the comparability of data across different software ecosystems. Specifically, this analysis enables future researchers to establish metrics that facilitate the comparison of Hugging Face and traditional software registries similarities and differences in package reuse practices.

Answering RQ1 also prepares us for further empirical inquiry:

RQ2 Do the qualitative claims about package reuse (PTM) on Hugging Face hold up when quantified?

3.1 Overview of Methodology

Figure 1 shows the overview of our work's context and approach. We answered these questions through three steps:

(1) We conducted a systematic literature review (SLR) on PTM reuse, aiming to compile a comprehensive list of claims about package reuse.
(2) We synthesized these claims into quantifiable measurements, performed the measurements, and then compared them to the previous findings.
(3) Where possible, we compared these to those from traditional software to see differences between the reuse patterns in different registries.

The detailed method per RQ is in the corresponding sections.

4 RQ1: What claims about package reuse on Hugging Face are made by prior research?

Finding 1. The Systematic Literature Review (SLR) identified 12 quantifiable claims about package reuse on Hugging Face.
Finding 2. The claims are distributed among five quantifiable categories of methods: small- and large-scale quantitative measurements, qualitative surveys, interviews, and case studies.
Finding 3. After this classification, four quantitatively unevaluated claims remained, within the categories of *design trends*, *documentation and understanding*, and *selection considerations*. This finding shows a gap that we address in RQ2.

To enhance our understanding of the existing claims related to PTM reuse within package registries, we followed empirical standards and conducted a systematic literature review [11, 34, 46, 47]. Our initial step involved a pilot study to define the scope of our review (§4.1.1). A systematic literature review entails five steps: identification of research, study selection, study quality assessment, data extraction, and data analysis [34, 38]. Figure 4 illustrates the results of each step. We detail them next.

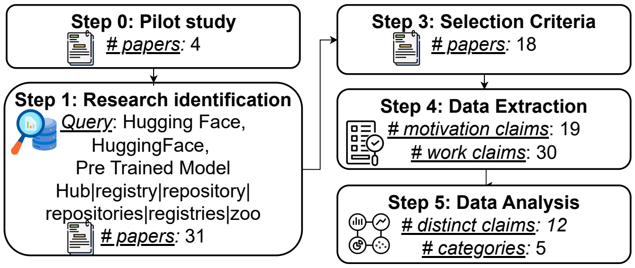

Figure 4: Systematic literature review process. Step 1 details the search query used. The subsequent filtering and distinct claims, along with the summarized categories, are discussed in §4.2.

4.1 Methods

4.1.1 Pilot Study We define the scope of our review by conducting a pilot study. In the pilot study, we first search for papers about "pre-trained model reuse" using Google Scholar and looked at the first three results, which were [21, 29, 31]. The papers indicated that Hugging Face is the only "open" model registry [31] and is the most popular model registry. Additionally, Hugging Face hosts the largest number of PTM packages, and provides useful tools to facilitate PTM reuse. We then decided to scope down our study on Hugging Face model registry specifically to represent the PTM supply chain, as indicated by Jiang *et al.* [29, 32].

4.1.2 Search Strategy and Query The goal of our search is to identify papers that are relevant to the categories of PTM reuse, the PTM supply chain, or the PTM ecosystem. Informed by our pilot study (§4.1.1), the final search query we used is indicated in Figure 4. These search queries gave us 45 papers. We then removed duplicate entries, reducing the number of papers to 31.

To verify the efficacy of our search queries, we employed papers from the pilot study as benchmarks to assess each query's retrieval effectiveness. We ensured that all papers identified in the pilot study were also retrieved by our final search query. This step was essential to confirm the robustness of our search strategy, ensuring it was capable of capturing the most relevant studies. Such a comprehensive approach allowed for an exhaustive review of the literature concerning PTM reuse within its operational ecosystem.

4.1.3 Selection Criteria The goal of our selection criteria is to identify the most relevant and rigorously supported research that specifically addresses the context and impact of PTM reuse. During our study, we applied two types of criteria: *inclusive* and *exclusive*.

Inclusion Criteria We applied the following *inclusion criterion*: a paper is included if it describes the reuse of models within a specific PTM registry. This criterion excluded papers that apply PTMs to specific tasks, reducing our set from 31 papers to 20.

Exclusion Criteria Our *exclusion criterion* was that we excluded non-primary sources. We exclude works whose claims are not substantiated directly through qualitative or quantitative methods. This reduced our set from 20 papers to 18.

4.1.4 Data Extraction Once we identified the most relevant and rigorously supported research that specifically addresses the context and impact of PTM reuse, the next step is to extract "claims" from these papers that provide evidence of qualitative or quantitative methods that could inform our quantitative study. A "claim" in this context refers to a statement or assertion made in a research paper, which is supported by evidence.

The data extraction involve four steps. *First*, two co-authors went through a paper from the pilot study together to get an agreement of the data extraction process. The goal of this process was to identify and extract all claims that might be tangentially related to PTM reuse. *Second*, they individually extracted the claims from the papers. This involved reading each paper to identify the key claims, with an emphasis placed on the abstract, introduction, and, if available, finding boxes of each paper. Exact quotations from the papers were extracted. *Third*, the two co-authors met and presented their extracted claims from each paper, with an explanation of why it was chosen, and a discussion of the relevance of the claim if it was not immediately obvious. A total of 256 claims were discussed in this step. *Finally*, for each paper, they discussed which claims were the most descriptive of *PTM reuse* and discarded the rest. This selection step resulted in a total of 49 claims.

4.2 Analysis and Results

We categorized these claims into two categories: (1) "*Motivation claims*" and (2) "*Work claims*". The detailed definitions and examples of each claim are shown in Table 2. This classification process yielded 19 *motivation claims* and 30 *work claims*. One of our goals (RQ2) was to substantiate prior findings with further quantitative measurements, so we chose to focus more deeply on the work claims. Within this subset, we identified overlapping claims and consolidated them, resulting in a refined set of 12 distinct claims. These consolidated claims are detailed in Table 3.

The primary aim of our SLR is to summarize the existing claims about package reuse on Hugging Face and to extract quantifiable measurements from these claims. From the consolidated set of 12 claims, we categorized the basis of each claim into one of five methods: small- and large-scale quantitative measurements, qualitative surveys, interviews, and case studies. *Small-scale measurements* involved less than 10% of a population, whereas *large-scale measurements* encompassed more than 10%. After this classification, five quantitatively unevaluated claims remained: one concerning *design trends*, two regarding *selection considerations*, and two about *documentation and understanding*.

The categorization result and extracted themes are detailed in Table 3. Our results provide a thorough knowledge synthesis which are then used to answer RQ2 (§5).

5 RQ2: Do the qualitative claims about package reuse (PTM) on Hugging Face hold up when quantified?

Finding 4. Our study shows that the Transformers library is preferred in over 80% of PTM descendants, surpassing PyTorch. The rise of the SafeTensors library underscores a shift toward prioritizing security in PTM development.

Finding 5. Figure 6 reveals that Hugging Face has a significantly higher package turnover rate than traditional software registries indicative of a fast-paced, innovation-driven PTM ecosystem.
Finding 6. There is a correlation between model popularity and descendant count, indicating that while popular models have more descendants, other factors influence model selection decisions.
Finding 7. There is a strong correlation between documentation quality and PTM popularity, with the top 1000 models significantly outperforming the bottom 1000 in documentation, highlighting its importance in model selection.

To answer this question, we first derive quantifiable metrics from the claims we extracted from our SLR (§4) on Hugging Face (§5.1). Then we present the available datasets and the specific data we used from each (§5.2). Subsequently, we present our methods and results for measurement on metrics for each claims (§5.3–§5.5).

5.1 Metrics Developed from Claims

In this section, we explain how we developed quantifiable metrics from the claims identified in our SLR (§4). The metrics were specifically designed to quantify the hypothesized results inferred from these claims. Drawing on traditional software engineering practices, we adopted metrics that have been previously used to evaluate similar claims in other contexts if available.

Particular attention was paid to metrics that are widely recognized and have been frequently cited in the literature, as well as those that have been implemented across various traditional software registries. This approach ensures that our metrics are grounded in established methodologies and are robust enough to provide meaningful insights.

Table 4 shows the relationship between the claims extracted and the corresponding metrics, along with the expected measurements. This mapping both validaties the claims and contextualizes their implications within the context of package reuse on Hugging Face.

5.2 Available Datasets

The section presents the PTM package datasets (§5.2.1) and traditional software package dataset (§5.2.2) we used in our work.

5.2.1 PTM Datasets In the PTM literature, there are four datasets available publicly: HF Model Metadata [53], PTMTorrent [30], HF-Community [8], and PeaTMOSS [32].

(1) **HF Model Metadata** provides a snapshot of 10,406 HuggingFace model metadata as of 11/2022, including details of model label, README and length of each README file [53].

Table 2: The detailed definition and examples of each claim category we extracted from the literature review.

Claim Category	Definition	Examples
Motivation claims	Articulates the rationale behind a paper's problem statement, illustrating why the work is significant and worthy of investigation.	"*The reuse of pre-trained models introduces large costs and additional problems of checking whether arbitrary pre-trained models are suitable for the task-specific reuse or not.*" [21] "*With the commodification of AI, and NLP in particular...[we] need easy to use, no-code tools for understanding AI artifacts.*" [43]
Work claims	Refers to the assertions a paper makes based on its collected data and analyses.	"*Hugging Face's popularity has exponentially increased over time, which is evident from the upward trends in the number of new models, likes, commits, unique authors, and discussions aggregated monthly.*" [12] "*Engineers follow specific naming practices and encounter challenges that are specific to PTM naming.*" [28]

Table 3: Consolidated themes and claims collected from our systematic literature review. Small-Scale measurements refer to measurements made on a selection of a population less than 10% of the overall size, Large-scale measurements are measurements made on a selection of a population that is more than 10%, Survey refers to a survey, Interviews refer to interviews, Case study refers to either an examination of a specific case or the creation of a model to verify the claim. † : The claim basis is not a large-scale quantification.

Themes	Claims	Claim Basis	Works
Trends in Design	A small group of contributors owns popular models.	Large-Scale Measurement	[12]
	The Transformers library increases the accessibility of PTM creation and downstream reuse.	Large-Scale Measurement, Case Study	[12, 32, 33, 57]
	† *(1) The Transformers library improves the process of PTM evolution.*	**Case Study**	[57]
	Forking repositories introduces low severity vulnerabilities.	Large-Scale Measurement	[33]
Selection Considerations	† *(4) DL-specific attributes such as model architecture, performance, reproducibility, and portability affect PTM selection and reuse.*	**Interviews**	[29]
	† *(2) The traditional attribute of Popularity affects PTM selection and reuse more than Maintenance and Quality.*	**Interviews**	[29]
Repository Lifecycle and Maintenance	Models receive perfective maintenance over time, with high-maintenance models tending to be more popular, larger, and better documented.	Large-Scale Measurement	[12]
Documentation and Understanding	Model properties are under-documented across Hugging Face.	Small-Scale and Large-Scale Measurement, Survey	[21, 33, 39, 51]
	† *(3) Documentation quality impacts model selection.*	**Survey, Case Study**	[13, 39, 51]
	Naming models is inconsistent and can inadequately represent model architectures.	Large-Scale Measurement, Survey	[28]
	Dataset documentation is correlated with dataset popularity.	Small-Scale Measurement	[59]
Downstream Usage	Hugging Face is an exponentially growing platform.	Large-Scale Measurement	[12]

(2) **PTMTorrent** encompasses a snapshot of five model hubs, totaling 15,913 PTM packages as of 08/2023, all formatted in a uniform data schema to facilitate cross-hub mining [30].
(3) **HFCommunity** is an offline relational database constructed from the data at the Hugging Face Hub. It allows for queries on the repositories hosted within the Hugging Face platform [8].
(4) **PeaTMOSS** offers comprehensive metadata on 281,638 PTM packages as of 10/2023, including 281,276 from Hugging Face and 362 from PyTorch Hub, along with details on 28,575 GitHub projects that use PTMs as dependencies and 44,337 links from these GitHub repositories back to the PTMs they depend on [32].

In this work, we primarily used the PeaTMOSS dataset, as the accessibility of metadata allowed for easier measurements. We also used the HF Model Metadata and PTMTorrent datasets for longitudinal trends in the turnover metric (§5.4), and an April 2024 recent snapshot of the most popular models from the Hugging Face Hub API for the same reason. The detailed data used for each measurement are presented in their respective sections.

5.2.2 Traditional package datasets To directly compare measurements between the PTM registry and traditional Software Package Registries (SPRs), we utilized the Ecosyste.ms software package dataset for traditional packages, following the approach outlined in prior work [48]. This dataset provides a set of free and open resources for those working to sustain and secure open source software. `Ecosyste.ms` publishes open data and APIs that map software interdependencies, as well as providing data on the usage, creation, and potential impact of packages.

In this work, we used the version of the dataset from October 2023, as this is the closest in time to when the PeaTMOSS dataset was created. This includes the usage of data from two software package registries: PyPI and NPM.

5.3 C_1: The Transformers library increases the accessibility of PTM creation and downstream reuse.

5.3.1 Method We present the metric we developed for claim 1. **Metric 1: Preservation rate of libraries to descendents:** Some models support multiple libraries. If the claim holds, then descendants of those models should make use of (and thus support) the more reuse-friendly libraries. We consider each library L in turn to assess how frequently its descendants continue to use it. The ingredients of our measure are: the library L being assessed, the set of base models B that use library L and at least one other library, the set D_b of direct descendants of base model $b \in B$, and a function $S(d, L)$ that returns 1 if descendant $d \in D_b$ supports library L, otherwise 0. We can then calculate the *preservation rate* P_L of a library L as:

$$P_L = \frac{\sum_{b \in B} \sum_{d \in D_b} S(d, L)}{\sum_{b \in B} |D_b|} \tag{1}$$

The numerator is the total descendants supporting library L, calculated as: $\sum_{b \in B} \sum_{d \in D_b} S(d, L)$. The denominator is the total count of direct descendants across all base models in set B: $\sum_{b \in B} |D_b|$.

This equation will give the percentage of direct descendants that support library L. The raw count of descendants supporting the library can be found using the numerator.

5.3.2 Results **Metric 1: Preservation rate of libraries to descendents:** As depicted in Figure 5, the Transformers library has the highest survival rate from a parent model to its descendant among all libraries used. Specifically, over 80% of descendant models continue to employ the Transformers library, establishing it as the preferred choice for generating descendant models. Transformers is also the most popular library (has the largest population of models that use it). Contrarily, despite previous studies suggesting high prevalence, PyTorch was not favored among PTM descendants, a departure from claims within our systematic literature review that both Transformers and PyTorch are dominant on Hugging Face [12].

Instead, our findings highlight the Transformers and SafeTensors libraries as the most prevalent, suggesting a community shift towards prioritizing security, as SafeTensors is designed to replace the commonly used Python pickle library with a more secure container for deploying models. We attribute this discrepancy to prior studies' broader approach, which did not differentiate between model types and analyzed only a single snapshot of Hugging Face, thus lacking the detailed analysis presented here. Our results suggest that PTM synthesizers are willing to compromise on functionality and portability to ensure the distribution of more secure PTM packages.

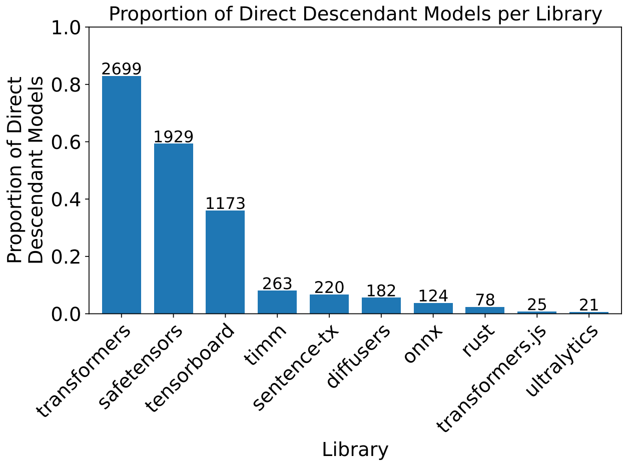

Figure 5: The usage proportion of the top-10 libraries that PTMs utilizing at least two different libraries use on Hugging Face. For PTM packages that leverage support at least two libraries, most packages support the transformers library, followed by the Hugging Face promoted SafeTensors library. Most other libraries have little usage in comparison. In contrast to previous work [12], PyTorch is not one of the most popular library to be supported when a PTM package supports more than one library.

5.4 C_2: Popularity Affects PTM Selection and Reuse More than Other Trad'l. Attributes

Our claim interpretation is that popular models are more likely to be used, so that the "rich get richer". We examined this claim with two metrics. First, we measure the stability (*i.e.,* non-turnover) of the top PTM packages over time, expecting it to be low (popular packages are used directly). Second, we measure the correlation between popularity and the number of descendants of top PTM packages, expecting it to be positive (popular packages are fine-tuned).

5.4.1 Methods We present the metrics we developed for claim 2. **Metric 2: Turnover of Top PTMs:** Drawing on prior work characterizing the stability of top packages over time [18], we measured the *top-K turnover* for each registry. Let $S_{current}$ be the set of K most popular packages in the current snapshot, and S_{last} be the set of top K packages in the last snapshot. We also consider $S_{history}$, the set of packages in any snapshot before S_{last}. We then distinguish three categories for packages in $S_{current}$:

(1) **Remained:** Packages that were in both S_{last} and $S_{current}$.

$$\text{Remained} = S_{last} \cap S_{current} \tag{2}$$

(2) **Newcomers:** Packages in $S_{current}$ in no previous snapshot.

$$\text{Newcomers} = S_{current} \setminus (S_{history} \cup S_{last}) \tag{3}$$

(3) **Returning:** Packages that were once in the top 1000 ($S_{history}$), were not in S_{last}, but are in $S_{current}$ again.

$$\text{Returning} = (S_{history} \setminus S_{last}) \cap S_{current} \tag{4}$$

Table 4: This table displays the relationships between Qualitative Claims, the Metric(s) used to evaluate them, and the Hypothesized Results. Where applicable, a reference is made to the traditional software prior work and metric that informed our metric. Note that not every metric contains a reference to a traditional software prior work that uses this metric as no analog exists. For those metrics, we utilize the Goal-Question-Metric process to develop metric(s) associated with the claim. Since the goal of these measurements is to substantiate existing claims from prior work, we include a hypothesis about what the quantitative measurement will be if the claim is true.

Qualitative Claim	Metric	Hypothesized Result
C_1: The Transformers library improves the process of PTM evolution.	Preservation rate of Libraries to descendents.	The preservation of Transformers as a library to its descendents will be greater than that of other libaries.
C_2: The traditional attribute of Popularity affects PTM selection and reuse more than Maintenance and Quality.	Turnover of top Packages over time [18].	Models with high popularity remain popular over time ("rich get richer").
	Descendent amount of models	Models with high popularity have a larger number of descendent models.
C_3: Documentation quality impacts model selection.	Popularity of PTMs based on their documentation quality.	Models with more information are more discoverable and therefore more popular [60].
C_4: DL-specific attributes such as model architecture, performance, reproducibility, and portability affect PTM selection and reuse.	Popularity of PTMs based on their Attributes [60].	Popular architectures, high performance, description of where the PTM came from, ease of use, and size all impact model popularity.

For the measurement, we defined popularity by the number of downloads, and examined the top-1000 packages. We obtained snapshots across four dates for both traditional software and PTMs. Data came from several HuggingFace snapshots (datasets: HF Model Metadata–June 2022, PTMTorrent–May 2023, PeaTMOSS–Oct. 2023). We took a current snapshot of the top 1000 PTMs directly from Hugging Face (April 2024). We used the Ecosyste.ms dataset (NPM, PyPI) for a comparison with traditional software package registries.

Metric 3: Number of Descendents of Top Packages: Our second measure of the impact of popularity on reuse was the number of descendent models. In this case, we defined descendent models as a downstream model that is fine-tuned and references the original model as a base model. We compare the number of descendent models with the popularity of model and determine the strength of correlation between them — the claim implies it should be positive.

For the measurement, we again defined popularity by the number of downloads. The descendent-base relation is available from the PeaTMOSS dataset, for the 15,000 most popular PTMs on Hugging Face. Given that these models account for ~99% of the downloads in the snapshot, we believe this is representative.

We initially planned to compare our findings with traditional software, similar to Metric 2. However, identifying a direct counterpart to PTM descendants in traditional software proved challenging. We considered using GitHub forks and GitHub or registry dependencies as analogs, but each presented unique implications that complicated a direct comparison with PTM descendants.

5.4.2 Results **Metric 2: Turnover of Traditional Software and PTM registries:** Figure 6 shows the results for Hugging Face. The Hugging Face data does not match our interpretation of the claim. About half of the top-1K Hugging Face PTMs turned over in each snapshot. This high turnover rate suggests that packages on Hugging Face have a shorter lifespan, indicating a dynamic PTM environment where the requirements and preferred models frequently change [23]. Further investigation could analyze the lifecycle of these PTMs to determine whether they are newer models briefly appearing or established ones losing prominence, as well as identifying the traits of PTMs or maintainers who consistently stay within the top rankings. This result is consistent with the claim, but implies that popularity is likely driven by performance (a latent variable not present in the claim).

Figure 6 also shows NPM for comparison. The results for PyPI were similar. Traditional registries show stability with their most popular packages. Quantitatively, 2535 distinct packages were in the HuggingFace top-1K, while only 1127 were there for NPM.

The rapid turnover on Hugging Face compared to the stability observed in traditional software packages indicates that PTM needs are continually evolving, unlike the more established needs in traditional software domains. This evolution often leads to older PTMs being supplanted by newer models that better meet current requirements or offer superior performance, highlighting a market driven by innovation and rapid adaptation. This trend suggests that PTM users are quick to adopt new advancements, reflecting the field's fast-paced development and the shifting demands across its various application domains.

Metric 3: Number of Descendents of Top Packages: Figure 7 shows the results for this metric. We observe a weak positive correlation between popularity (downloads) and the number of descendent models. Popular models tend to have more descendants, but the correlation was weaker than expected. Notably, some highly popular models had relatively few descendants, while others with similar popularity levels had many. This suggests that while popularity influences the likelihood of a model being chosen for further development, it is not the sole factor. Further analysis, including a

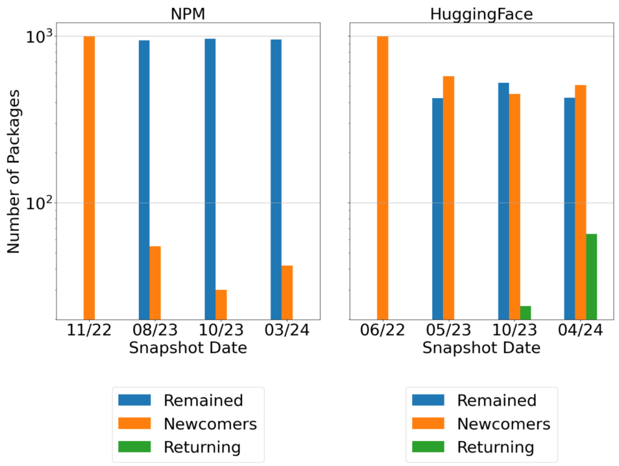

Figure 6: The turnover of the top-1000 Packages on Hugging Face, NPM. Note that in traditional software package registries such as NPM, the level of turnover is low. Hugging Face has a much larger amount of packages re-enter the Top 1000 as shown by the larger green bars in the second and fourth snapshot compared to NPM, with few returners as shown by the relative lack of green bars.

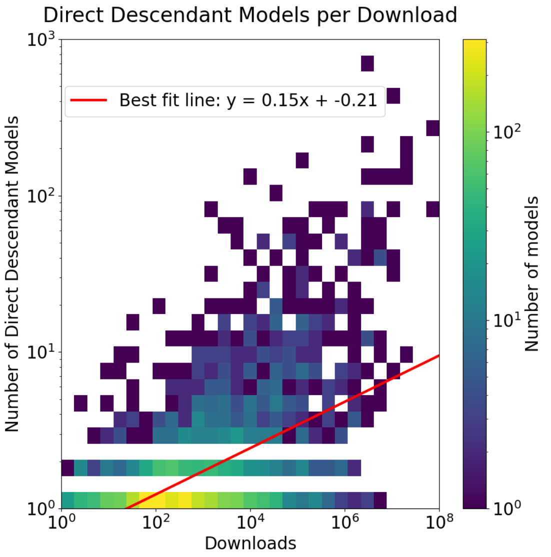

Figure 7: A heatmap showing the correlation between Downloads and Number of Direct Descendent Models. Note that this is a log-log plot. The red-line displays the best-fit relationship between downloads and descendant count, with a slope of 0.15**. This positive correlation suggests that models with higher download counts generally have a larger number of direct descendants, indicating that popular models tend to be reused more frequently in derivative work.**

breakdown by model task and domain, is necessary to understand if variations in descendant counts are influenced by the specific popularity within less prominent domains or tasks. This analysis could expose the decision-making processes engineers use when selecting models to fine-tune and develop further.

5.5 C_3: Docs Quality Impacts Model Selection

Our interpretation of Claim 3 is that PTMs with better documentation will be more popular.

5.5.1 Method **Metric 4: Documentation Quality:** Prior work has examined documentation quality in many ways [22, 50]. We used those ideas to develop our measure of quality. The primary documentation for PTMs is called a "model card", which is similar to the README of a GitHub repository or the landing page of an NPM or PyPI package. We considered two factors: (1) the completeness of the model card; and (2) the availability of metadata.

To measure completeness, we identified five typical sections found in highly popular PTMs such as Google's *Bet base uncased*: *Model Description*, *Limitations*, *How to Use*, *Training*, and *Evaluation*. We scored model cards on an integer scale from 0 to 5 — to receive a score of 5, a PTM's card needed all of these sections. To assess whether each section was present, we queried OpenAI's ChatGPT-4 with the prompt shown in Listing 1.

Listing 1: ChatGPT-4 prompt for evaluating model cards.

```
You will receive a model card and are expected to analyze
      it for the following details:
1. Model description: A description of the model itself
2. Limitations: Any limitations of the model
3. How to use: Instructions on how to use the model
    downstream
4. Training: Details of the training process or data
5. Evaluation: Reports on the model's performance
    evaluation

Please respond with a JSON object indicating whether each
      of these points is present with true/false.
Here is the model card to evaluate:
```

To measure the availability of metadata, we referenced the PeaTMOSS dataset, which extracted over 20 distinct pieces of metadata if they were present in a PTM's model card and associated configuration files. We scored PTMs on an integer scale from 0 to the maximum, *i.e.,* the number of distinct pieces of metadata considered in the PeaTMOSS database schema. A PTM scoring perfectly in this category would possess all available metadata according to PeatMOSS.

The PeaTMOSS dataset comprises extracted metadata from all models on Hugging Face and includes snapshots of the top 15,000 most popular models, each with over 50 monthly downloads. These models were further analyzed using an LLM, which extracted additional metadata from the model cards and the `config.json` files. Consequently, our analysis focused on these top 15,000 models to assess whether documentation quality influences model selection.

For this measure, we considered popularity in three ways: Downloads and Likes, according to Hugging Face, and Downstream Dependents, based on the mapping offered by the PeaTMOSS dataset.

To test our interpretation of the metric, we selected the top 1000 and bottom 1000 models from PeaTMOSS's set of 15,000 PTMs. We evaluated their documentation quality, summing an overall documentation score as a (0,1) metric that normalized and then weighted the two components equally. Then we compared the distributions using a box-and-whisker plot and statistical tests.

5.5.2 Results **Metric 4: Documentation Quality:** We evaluated the impact of documentation quality on the popularity of PTMs. Two representative box and whisker plots are shown in Figure 8.

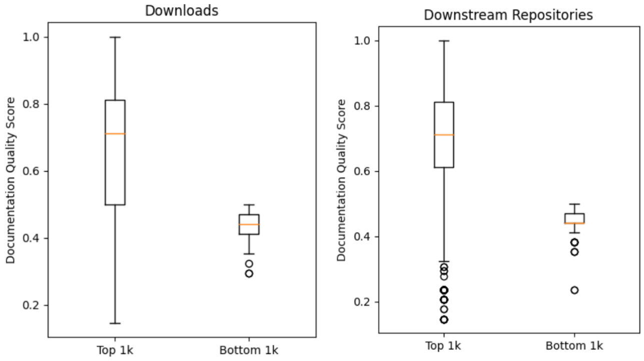

Figure 8: This figure shows the impact of documentation quality on model popularity using two popularity metrics: downloads and downstream reuse. The left box plot compares the documentation quality of the top 1,000 and bottom 1,000 models based on the number of downloads. The right box plot makes a similar comparison based on the number of downstream repositories. In both metrics, the top models demonstrate higher documentation quality scores than the bottom models, highlighting that models with better documentation are more popular and are reused more frequently.

Specifically, the top 1000 most popular models consistently exhibited significantly better documentation than the bottom 1000 models ($p < 0.01$. This finding supports the claim that documentation quality significantly influences model selection. If the relation is causative (documentation → popularity), then research can focus on developing tools and methods to enhance the documentation quality of models, supporting better usability and adoption.

6 Threats to Validity

We discuss three types of threats to validity [56], while considering the criticisms of Verdecchia *et al.* [54].

Construct Threats are potential limitations of how we operationalized concepts. In the systematic literature review (RQ1), we manually extracted claims from papers, which might introduce potential bias to our results. As a mitigation, to improve objectivity two authors worked together on the process and the filtering of claims. In the validation of non-quantified claims, we proposed metrics that seemed suitable based on our judgment. As a mitigation, where possible we used multiple measures and leveraged previously-defined metrics.

Internal threats are those that affect cause-effect relationships. We emphasize that our approach for RQ2 is of the form: "If claim C is true, then measurement M should show us that...". In each case the measurement produced the expected result. However, this result is correlative, not causative — there may be a latent variable in each case, or the causative relationship may be reversed. For example, in Figure 8 we found that better documentation correlates with greater popularity. It may be that the latent variable is performance, such that models become popular because they have good performance, and they accrue documentation because they are popular. When qualitative and quantitative claims agree, as is the case in this study, we learn both "Why?" and "How much?".

External threats may impact generalizability. We recognize both immediate and longitudinal threats in this regard. Immediately, we were interested in studying PTM reuse, but we only examined one registry for PTMs: the Hugging Face platform. While this is by far the most popular and feature-rich platform, other platforms exist, such as PyTorch Hub (less popular), PapersWithCode (fewer features), and GitHub (not PTM-specific). In terms of longitudinally, keep in mind that Hugging Face is itself relatively young — created in 2016 and only seeing major use beginning in 2020 — so developer practices may not have stabilized. Lastly, we consider that the technologies and platforms that support PTMs are rapidly evolving, so current claims (whether qualitative or quantitative) may change over time and require ongoing reassessment. As indicated in the discussion, this property suggests an opportunity for further research, but it also means that our findings may be unstable.

7 Future Work

In Table 4, we identified one claim that we could not operationalize for measurement: C_4, that different deep learning-specific attributes would affect PTM selection and reuse. Although the PeaTMOSS dataset does include some of these measures in a structured format, the claim is sufficiently broad that quantifying it was beyond the scope of this study. Future work could explore the multi-variable relationship implied by this claim.

Our study provides the first systematic literature review of current knowledge about PTM reuse. Another opportunity is a systematic *comparison* of the software package registries associated with traditional software packages, and the software package registries associated with PTMs. Jiang *et al.* observed similarities and differences in the reuse processes [29] — how and to what extent can we measure the differences? As discussed in §5.4.1 with respect to an analogue for PTM descendants, finding the limits of comparison (*e.g.,* appropriate measurements) between traditional vs. PTM software package registries is an open challenge.

The rapid development of PTM technologies presents an ongoing challenge for empirical research on PTM reuse. The Hugging Face platform continues to grow exponentially [12], the state of the art performance of models at all sizes continues to advance [3], and the tooling available to adapt and deploy these models continues to improve [26]. While datasets and tooling such as HFCommunity and PeaTMOSS support studies like ours, they also present some limitations. Innovation is needed to help empirical software engineering researchers keep up with the scale of activity and volume of data that we are seeing in the context of PTM reuse. Given Hugging Face's dynamic growth, even data that is a few months old may not accurately reflect the current state of the platform, suggesting a need for regular snapshotting, which imposes significant storage requirements (beyond the already-substantial requirements of >50 TB). This highlights the need for tools that can provide real-time, incrementally-updated data to keep pace with rapid changes, ensuring that analyses remain relevant and reflective of the present situation in PTM reuse.

8 Conclusion

Pre-trained models are the motive force of the new generation of software engineering. Understanding engineers' PTM reuse practices is crucial to optimizing and securing the process. Our systematic literature review and quantification of claims has illuminated significant aspects of PTM reuse. We also shed light on unique dynamics within the PTM reuse landscape as compared to traditional software package registries. Specifically, we observed a shorter lifespan of packages in PTM registries compared to traditional SPRs. Our findings underscore the need for research infrastructure and novel tools to support PTM reuse, adapting to the much higher turnover of popular PTM packages. We must ensure that PTM registries can meet the evolving demands of the software engineering community.

9 Data Availability

An anonymous artifact containing the results of the systematic literature review (RQ1) as well as our software and results for quantification of claims (RQ2) is available at https://github.com/anonsub1234/ptm-quantify-esem-2024.

10 Research Ethics

No human subjects were involved in the conduct of this project. We are aware of no other ethical concerns.

Acknowledgments

We acknowledge support from NSF awards OAC-2107020 and OAC-2107230.

References

[1] [n. d.]. npm: a package manager for JavaScript. https://www.npmjs.com/.

[2] [n. d.]. ONNX Model Zoo: a collection of pre-trained, state-of-the-art models in the ONNX format. https://github.com/onnx/models.

[3] [n. d.]. Papers with Code. https://paperswithcode.com

[4] [n. d.]. PyPI: the Python Package Index. https://pypi.org/.

[5] [n. d.]. PyTorch Hub: a pre-trained model repository designed for research reproducibility. https://pytorch.org/hub/.

[6] Rabe Abdalkareem, Olivier Nourry, Sultan Wehaibi, Suhaib Mujahid, and Emad Shihab. 2017. Why do developers use trivial packages? an empirical case study on npm. In *Proceedings of the 2017 11th joint meeting on foundations of software engineering*. 385–395.

[7] Iftekhar Ahmed, Darren Forrest, and Carlos Jensen. 2017. A case study of motivations for corporate contribution to FOSS. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (Raleigh, NC, 2017-10-01). IEEE, 223–231. https://doi.org/10.1109/VLHCC.2017.8103471

[8] Adem Ait, Javier Luis Cánovas Izquierdo, and Jordi Cabot. 2023. HFCommunity: A tool to analyze the hugging face hub community. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 728–732.

[9] Oliver A Blanthorn, Colin M Caine, and Eva M Navarro-López. 2019. Evolution of communities of software: using tensor decompositions to compare software ecosystems. *Applied Network Science* 4, 1 (2019), 120.

[10] Ethan Bommarito and Michael James Bommarito. 2019. An Empirical Analysis of the Python Package Index (PyPI). https://doi.org/10.2139/ssrn.3426281

[11] Michael Borenstien, Larry Hedges, Julian Higgins, and Hannah Rothstein. 2009. Introduction to meta-analysis. *West Sussex, United Kingdon: John Wiley & Sons* (2009).

[12] Joel Castaño, Silverio Martínez-Fernández, Xavier Franch, and Justus Bogner. 2023. Analyzing the evolution and maintenance of ml models on hugging face. *arXiv preprint arXiv:2311.13380* (2023).

[13] Joel Castaño, Silverio Martínez-Fernández, and Xavier Franch. 2024. Lessons Learned from Mining the Hugging Face Repository. arXiv:2402.07323 [cs.SE]

[14] Joel Castaño, Silverio Martínez-Fernández, Xavier Franch, and Justus Bogner. 2023. Exploring the Carbon Footprint of Hugging Face's ML Models: A Repository Mining Study. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12. https://doi.org/10.1109/ESEM56168.2023.10304801

[15] James C Davis, Purvish Jajal, Wenxin Jiang, Taylor R Schorlemmer, Nicholas Synovic, and George K Thiruvathukal. 2023. Reusing deep learning models: Challenges and directions in software engineering. In *2023 IEEE John Vincent Atanasoff International Symposium on Modern Computing (JVA)*. IEEE, 17–30.

[16] Alexandre Decan, Tom Mens, and Eleni Constantinou. 2018. On the evolution of technical lag in the npm package dependency network. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 404–414.

[17] Alexandre Decan, Tom Mens, and Philippe Grosjean. 2019. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical Software Engineering* 24, 1 (2019), 381–416.

[18] Tapajit Dey and Audris Mockus. 2018. Are software dependency supply chain metrics useful in predicting change of popularity of npm packages?. In *Proceedings of the 14th international conference on predictive models and data analytics in software engineering*. 66–69.

[19] Joshua Garcia, Yang Feng, Junjie Shen, Sumaya Almanee, Yuan Xia, Chen, and Qi Alfred. 2020. A comprehensive study of autonomous vehicle bugs. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*. 385–396.

[20] Daniel M. German, Massimiliano Di Penta, and Julius Davies. 2010. Understanding and Auditing the Licensing of Open Source Software Distributions. In *2010 IEEE 18th International Conference on Program Comprehension* (2010-06). 84–93. https://doi.org/10.1109/ICPC.2010.48 ISSN: 1092-8138.

[21] Lina Gong, Jingxuan Zhang, Mingqiang Wei, Haoxiang Zhang, and Zhiqiu Huang. 2023. What Is the Intended Usage Context of This Model? An Exploratory Study of Pre-Trained Models on Various Model Repositories. *ACM Transactions on Software Engineering and Methodology* 32, 3 (2023), 69:1–69:57. https://doi.org/10.1145/3569934

[22] Lina Gong, Jingxuan Zhang, Mingqiang Wei, Haoxiang Zhang, and Zhiqiu Huang. 2023. What is the intended usage context of this model? An exploratory study of pre-trained models on various model repositories. *ACM Transactions on Software Engineering and Methodology* 32, 3 (2023), 1–57.

[23] Nikhil Krishna Gopalakrishna, Dharun Anandayuvaraj, Annan Detti, Forrest Lee Bland, Sazzadur Rahaman, and James C. Davis. 2023. "If security is required": engineering and security practices for machine learning-based IoT devices. In *Proceedings of the 4th International Workshop on Software Engineering Research and Practice for the IoT* (Pittsburgh Pennsylvania, 2022-05-19). ACM, 1–8. https://doi.org/10.1145/3528227.3528565

[24] Yacong Gu, Lingyun Ying, Yingyuan Pu, Xiao Hu, Huajun Chai, Ruimin Wang, Xing Gao, and Haixin Duan. 2023. Investigating package related security threats in software registries. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1578–1595.

[25] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556* (2022).

[26] Purvish Jajal, Wenxin Jiang, Arav Tewari, Joseph Woo, George K Thiruvathukal, and James C Davis. 2023. Analysis of failures and risks in deep learning model converters: A case study in the onnx ecosystem. *arXiv preprint arXiv:2303.17708* (2023).

[27] Wenxin Jiang, Vishnu Banna, Naveen Vivek, Abhinav Goel, Nicholas Synovic, George K Thiruvathukal, and James C Davis. 2023. Challenges and practices of deep learning model reengineering: A case study on computer vision. *arXiv preprint arXiv:2303.07476* (2023).

[28] Wenxin Jiang, Chingwo Cheung, Mingyu Kim, Heesoo Kim, George K. Thiruvathukal, and James C. Davis. 2024. Naming Practices of Pre-Trained Models in Hugging Face. arXiv:2310.01642 [cs.SE]

[29] Wenxin Jiang, Nicholas Synovic, Matt Hyatt, Taylor R. Schorlemmer, Rohan Sethi, Yung-Hsiang Lu, George K. Thiruvathukal, and James C. Davis. 2023. An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 2463–2475. https://doi.org/10.1109/ICSE48619.2023.00206

[30] Wenxin Jiang, Nicholas Synovic, Purvish Jajal, Taylor R. Schorlemmer, Arav Tewari, Bhavesh Pareek, George K. Thiruvathukal, and James C. Davis. 2023. PTMTorrent: A Dataset for Mining Open-source Pre-trained Model Packages. , 57-61 pages. https://doi.org/10.1109/MSR59073.2023.00021

[31] Wenxin Jiang, Nicholas Synovic, Rohan Sethi, Aryan Indarapu, Matt Hyatt, Taylor R Schorlemmer, George K Thiruvathukal, and James C Davis. 2022. An empirical study of artifacts and security risks in the pre-trained model supply chain. In *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. 105–114.

[32] Wenxin Jiang, Jerin Yasmin, Jason Jones, Nicholas Synovic, Jiashen Kuo, Nathaniel Bielanski, Yuan Tian, George K Thiruvathukal, and James C Davis. 2024. PeaTMOSS: A Dataset and Initial Analysis of Pre-Trained Models in Open-Source Software.

[33] Adhishree Kathikar, Aishwarya Nair, Ben Lazarine, Agrim Sachdeva, and Sagar Samtani. 2023. Assessing the Vulnerabilities of the Open-Source Artificial Intelligence (AI) Landscape: A Large-Scale Analysis of the Hugging Face Platform. In *2023 IEEE International Conference on Intelligence and Security Informatics (ISI)* (2023-10). 1–6. https://doi.org/10.1109/ISI58743.2023.10297271

[34] Staffs Keele et al. 2007. Guidelines for performing systematic literature reviews in software engineering.

[35] Raula Gaikovina Kula, Coen De Roover, Daniel German, Takashi Ishio, and Katsuro Inoue. 2014. Visualizing the evolution of systems and their library dependencies. In *2014 Second IEEE Working Conference on Software Visualization*. IEEE, 127–136.

[36] Chengwei Liu, Sen Chen, Lingling Fan, Bihuan Chen, Yang Liu, and Xin Peng. 2022. Demystifying the vulnerability propagation and its evolution via dependency trees in the npm ecosystem. In *Proceedings of the 44th International Conference on Software Engineering*. 672–684.

[37] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. 2017. Modelhub: Deep learning lifecycle management. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 1393–1394.

[38] David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G Altman, Prisma Group, et al. 2010. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *International journal of surgery* 8, 5 (2010), 336–341.

[39] Diego Montes, Pongpatapee Peerapatanapokin, Jeff Schultz, Chengjun Guo, Wenxin Jiang, and James C Davis. 2022. Discrepancies among pre-trained deep neural networks: a new threat to model zoo reliability. , 1605–1609 pages.

[40] Suhaib Mujahid, Rabe Abdalkareem, and Emad Shihab. 2023. What are the characteristics of highly-selected packages? A case study on the npm ecosystem. *Journal of Systems and Software* 198 (2023), 111588.

[41] Farzaneh Nasirian, Mohsen Ahmadian, and One-Ki Daniel Lee. 2017. AI-based voice assistant systems: Evaluating from the interaction and trust perspectives. (2017).

[42] D Patterson. 2022. Good news about the carbon footprint of machine learning training. *Google AI Blog* (2022).

[43] Aleksandra Piktus, Odunayo Ogundepo, Christopher Akiki, Akintunde Oladipo, Xinyu Zhang, Hailey Schoelkopf, Stella Biderman, Martin Potthast, and Jimmy Lin. 2023. GAIA search: Hugging face and pyserini interoperability for nlp training data exploration. *arXiv preprint arXiv:2306.01481* (2023).

[44] Donald Pinckney, Federico Cassano, Arjun Guha, and Jonathan Bell. 2023. A large scale analysis of semantic versioning in npm. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 485–497.

[45] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (2021).

[46] Paul Ralph, Nauman bin Ali, Sebastian Baltes, Domenico Bianculli, Jessica Diaz, Yvonne Dittrich, Neil Ernst, Michael Felderer, Robert Feldt, Antonio Filieri, et al. 2020. Empirical standards for software engineering research. *arXiv preprint arXiv:2010.03525* (2020).

[47] Paul Ralph and Sebastian Baltes. 2022. Paving the way for mature secondary research: the seven types of literature review. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1632–1636.

[48] Taylor R. Schorlemmer, Kelechi G. Kalu, Luke Chigges, Kyung Myung Ko, Eman Abdul-Muhd Abu Isghair, Saurabh Baghi, Santiago Torres-Arias, and James C. Davis. 2024. Signing in Four Public Software Package Registries: Quantity, Quality, and Influencing Factors. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

[49] Jacob Stringer, Amjed Tahir, Kelly Blincoe, and Jens Dietrich. 2020. Technical lag of dependencies in major package managers. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 228–237.

[50] Henry Tang and Sarah Nadi. 2023. Evaluating Software Documentation Quality. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 67–78. https://doi.org/10.1109/MSR59073.2023.00023

[51] Mina Taraghi, Gianolli Dorcelus, Armstrong Foundjem, Florian Tambon, and Foutse Khomh. 2024. Deep Learning Model Reuse in the HuggingFace Community: Challenges, Benefit and Trends.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[53] Daniel Van Strien. 2022. Hugging Face Model Metadata Dataset. https://huggingface.co/datasets/davanstrien/hf_model_metadata.

[54] Roberto Verdecchia, Emelie Engström, Patricia Lago, Per Runeson, and Qunying Song. 2023. Threats to validity in software engineering research: A critical reflection. *Information and Software Technology* 164 (2023), 107329.

[55] Erik Wittern, Philippe Suter, and Shriram Rajagopalan. 2016. A Look at the Dynamics of the JavaScript Package Ecosystem. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)* (Austin Texas). ACM, 351–361. https://doi.org/10.1145/2901739.2901743

[56] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[57] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/arXiv.1910.03771 arXiv:1910.03771 [cs]

[58] Yulun Wu, Zeliang Yu, Ming Wen, Qiang Li, Deqing Zou, and Hai Jin. 2023. Understanding the threats of upstream vulnerabilities to downstream projects in the maven ecosystem. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1046–1058.

[59] Xinyu Yang, Weixin Liang, and James Zou. 2024. Navigating Dataset Documentations in AI: A Large-Scale Analysis of Dataset Cards on Hugging Face. https://doi.org/10.48550/arXiv.2401.13822 arXiv:2401.13822 [cs]

[60] Ahmed Zerouali, Eleni Constantinou, Tom Mens, Gregorio Robles, and Jesús González-Barahona. 2018. An empirical analysis of technical lag in npm package dependencies. In *International conference on software reuse*. Springer, 95–110.

[61] Ahmed Zerouali, Tom Mens, Jesus Gonzalez-Barahona, Alexandre Decan, Eleni Constantinou, and Gregorio Robles. 2019. A formal framework for measuring technical lag in component repositories—and its application to npm. *Journal of Software: Evolution and Process* 31, 8 (2019), e2157.

[62] Ahmed Zerouali, Tom Mens, Gregorio Robles, and Jesus M Gonzalez-Barahona. 2019. On the diversity of software package popularity metrics: An empirical study of npm. In *2019 IEEE 26th international conference on software analysis, Evolution and Reengineering (SANER)*. IEEE, 589–593.

[63] Yuxia Zhang, Minghui Zhou, Klaas-Jan Stol, Jianyu Wu, and Zhi Jin. 2020. How do companies collaborate in open source ecosystems? an empirical study of openstack. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1196–1208.

[64] Markus Zimmermann, Cristian-Alexandru Staicu, Cam Tenny, and Michael Pradel. 2019. Small world with high risks: A study of security threats in the npm ecosystem. In *28th USENIX Security Symposium (USENIX Security 19)*. 995–1010.