Automated categorization of pre-trained models for software engineering: A case study with a Hugging Face dataset

Claudio Di Sipio
claudio.disipio@univaq.it
University of L'Aquila
L'Aquila, Italy

Riccardo Rubei
riccardo.rubei@univaq.it
University of L'Aquila
L'Aquila, Italy

Juri Di Rocco
juri.dirocco@univaq.it
University of L'Aquila
L'Aquila, Italy

Davide Di Ruscio
davide.diruscio@univaq.it
University of L'Aquila
L'Aquila, Italy

Phuong T. Nguyen
phuong.nguyen@univaq.it
University of L'Aquila
L'Aquila, Italy

ABSTRACT

Software engineering (SE) activities have been revolutionized by the advent of pre-trained models (PTMs), defined as large machine learning (ML) models that can be fine-tuned to perform specific SE tasks. However, users with limited expertise may need help to select the appropriate model for their current task. To tackle the issue, the Hugging Face (HF) platform simplifies the use of PTMs by collecting, storing, and curating several models. Nevertheless, the platform currently lacks a comprehensive categorization of PTMs designed specifically for SE, i.e., the existing tags are more suited to generic ML categories.

This paper introduces an approach to bridge the gap by enabling the automatic classification of PTMs for SE tasks. First, we utilize a public dump of HF to extract PTMs information, including model documentation and associated tags. Then, we employ a semi-automated method to identify SE tasks and their corresponding PTMs from existing literature. The approach involves creating an initial mapping between HF tags and specific SE tasks, using a similarity-based strategy to identify PTMs with relevant tags. The evaluation shows that model cards are informative enough to classify PTMs considering the pipeline tag. Moreover, we provide a mapping between SE tasks and stored PTMs by relying ons model names.

KEYWORDS

Pre-trained models, Hugging Face, Model classification

ACM Reference Format:
Claudio Di Sipio, Riccardo Rubei, Juri Di Rocco, Davide Di Ruscio, and Phuong T. Nguyen. 2024. Automated categorization of pre-trained models for software engineering: A case study with a Hugging Face dataset. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3661167.3661215

EASE 2024, June 18–21, 2024, Salerno, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1701-7/24/06
https://doi.org/10.1145/3661167.3661215

1 INTRODUCTION

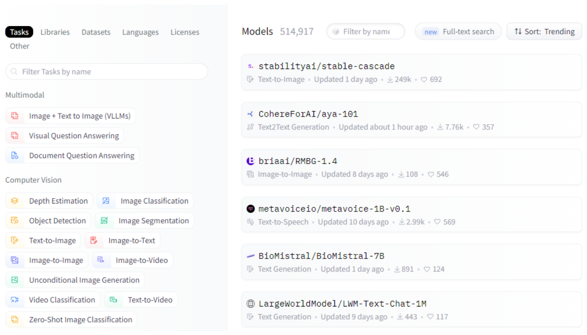
The emergence of foundation models (FMs), e.g., large language models (LLMs) [15] or pre-trained models (PTMs) [13] has significantly transformed traditional software engineering (SE) practices. Interestingly, PTMs are playing a crucial momentum while being used as a promising technology to support a wide range of SE tasks, including domain analysis, source code development, or testing. Although recent studies have shown that PTMs can outperform traditional approaches in terms of accuracy [10, 26, 28], selecting the appropriate model for a given task remains a challenge for non-expert users. To alleviate the burden of choice, the developers can rely on *model repositories* defined as dedicated open-source software repositories that store, collect, and maintain FMs [12]. Furthermore, most of those community-based platforms offer dedicated capabilities to search and filter stored models, e.g., tagging system or model documentation. Despite this, a proper mapping between the generic machine learning tasks and specific SE tasks is still missing.

In this paper, we present a semi-automated approach that centers around constructing a mapping between SE tasks and pre-trained models (PTMs) stored on the Hugging Face (HF) model repository,[1] recognized for housing the largest number of PTMs compared to other platforms [12]. Additionally, the HF community project [1] represents an initiative that regularly updates the platform's dump, thereby facilitating empirical analyses of the stored PTMs. In particular, we aim at answering the following research questions:

- **RQ$_1$**: *How effectively can state-of-the-art classifiers categorize pre-trained models based on the information provided in their model cards?*
- **RQ$_2$**: *How can the suggested mapping contribute to assisting developers in the selection of an appropriate pre-trained model?*

To address RQ_1, we evaluate the informativeness of the model card accompanying each stored pre-trained model (PTM) in facilitating traditional classifiers. Specifically, we employ two state-of-the-art text multi-label classifiers, namely the Complement Naive Bayesian (CNB) network [23] and the C-Support Vector classifier (SVC) [4]. To answer RQ_2, we establish a taxonomy of SE tasks, building upon existing work through a lightweight systematic study that considers the available literature. Utilizing the data, we formulate an initial version of the mapping using

[1] https://huggingface.co/



Models 514,917

stabilityai/stable-cascade
Text-to-Image • Updated 1 day ago • 249k • 692

CohereForAI/aya-101
Text2Text Generation • Updated about 1 hour ago • 7.76k • 357

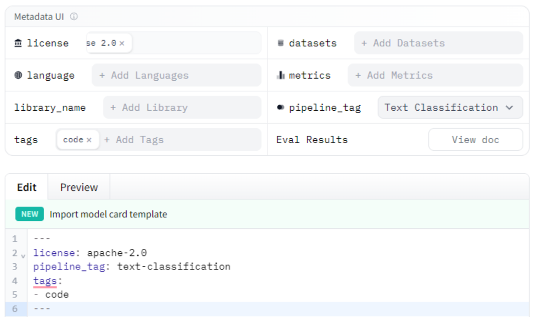
briaai/RMBG-1.4
Image-to-Image • Updated 8 days ago • 108 • 546

metavoiceio/metavoice-1B-v0.1
Text-to-Speech • Updated 10 days ago • 2.99k • 569

BioMistral/BioMistral-7B
Text Generation • Updated 1 day ago • 891 • 124

LargeWorldModel/LWM-Text-Chat-1M
Text Generation • Updated 9 days ago • 443 • 117

(a) Selection mechanism based on pipeline tags



license	se 2.0	datasets	
language		metrics	
library_name		pipeline_tag	Text Classification
tags	code	Eval Results	

```
---
license: apache-2.0
pipeline_tag: text-classification
tags:
- code
---
```

(b) Model card configuration

Figure 1: The Hugging Face PTM reuse-oriented capabilities.

a similarity-based algorithm. Our preliminary findings indicate that the provided model cards are sufficiently informative for automatic classification, relying on the associated tags. However, it is noteworthy that more than half of the PTMs lack corresponding cards, leaving a substantial number of models on Hugging Face uncovered. Furthermore, we present an initial iteration of the mapping between SE tasks and the generic PTM category. This mapping can be refined to encompass additional SE tasks or cater to different PTM repositories.

We suppose that devising a set of more advanced techniques to recommend the proper PTM given a specific SE task is crucial to support the creation of SE task-dedicated AI-based agents framework [11, 14]. Thus, this paper can be seen as the very first stepping stone to enable dedicated recommender systems for software engineering (RSSEs) [5, 24] that are able to automatically select the proper models given the SE task. It is our firm belief that devising advanced techniques for recommending the most suitable pre-trained model (PTM) based on a specific SE task is imperative to sustain the establishment of frameworks consisting of AI-based agents that are tailored for specific SE tasks [11, 14]. Our paper serves as a foundational step, paving the way for the creation of dedicated RSSEs [5, 24] endowing the capability to autonomously identify and recommend appropriate models tailored to given SE tasks. The main contributions of this paper are as follows:

- An initial mapping between generic PTM tags and SE tasks by adopting a semi-automatic screening of existing literature;
- An empirical evaluation conducted using the textual documentation available on Hugging Face to train state-of-the-art classifiers;
- A publicly available replication package to foster further studies in this domain [7].

2 MOTIVATION AND BACKGROUND

2.1 Overview of the PTM reuse workflow

As part of the *decision-making PTM reuse* process outlined by Jiang et al. [17], developers undertake distinct steps to search for and identify PTMs suitable for supporting specific tasks. Initially, developers engage in a *reusability assessment*, requiring a comprehensive analysis of the requirements to select the most appropriate PTM. Then, during the *model selection* phase, developers identify a set of candidate PTMs. Following this, developers may evaluate whether the chosen PTM is suitable for deployment through the *downstream evaluation*, which typically involves a fine-tuning phase specific to the task at hand. Finally, the *model deployment* phase encompasses testing the finalized version of the PTM in a real development environment.

Even though this process is facilitated by model repositories and other open-source platforms such as GitHub, the presented workflow still considers generic categories (e.g., object detection, image to text, text generation, and image classification) that are not directly linked to SE tasks. In this respect, we propose a taxonomy as a starting point to improve the first two phases of the above mentioned process, i.e., *reusability assessment* and *model Selection*, for SE tasks.

2.2 The Hugging Face model repository

To allow for their practical usage in SE tasks, PTMs need to be stored, maintained, and documented. In this respect, developers can share their PTMs on model repositories. Among the others, Hugging Face offers the largest number of PTMs and the corresponding documentation. Furthermore, the *HFCommunity* project [1] allows the analysis of metadata and source code contained in Hugging Face.

Figure 1 shows the model searching capabilities provided by Hugging Face. In particular, an interested developer can browse the hub by directly using the search bar. However, this process is time-consuming given the large number of PTMs stored on the platform. To mitigate this, HF provides a tagging system that can automatically filter the models by category as depicted in Figure 1a. For instance, the user can obtain the list of models that perform `text-classification`.

To further improve the visibility, PTM owners can upload the relevant information in the *model card*, i.e., a README-like document that provides the information required to configure and run the PTM at hand [19]. Figure 1b represents the creation process of a model card using the dedicated interface. In addition, the user who wants to publish a model can select two types of tags, i.e., `pipeline tag` and `user-based tags`. The former represents the community-based tagging system that expresses generic tasks, e.g., image classification, text-generation. Users manually specify the latter, like the GitHub topics mechanism.[2]

Unfortunately, many models stored in HF are not documented, or present missing information and discrepancies [3, 17]. Moreover, non-expert developers may need help to elicit the proper PTM according to the SE task of interest. Thus, we see an urgent need to provide developers with a rigorous mapping between HF tags and SE-related activities. In the scope of this paper, we focus on mapping pipeline tags to SE tasks since they are maintained by HF, thus avoiding erroneous labeling by users.

3 PROPOSED APPROACH

Figure 2 shows the proposed mapping approach: the data filtering phase is split into two main phases, i.e., *HF data gathering* ① and *SE task filtering* ②. A *mapping phase* ③ is subsequently performed to correlate identified SE tasks with candidate pre-trained models stored on Hugging Face. The subsequent sections elaborate on these three core phases.

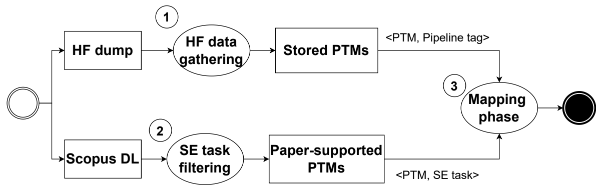


Figure 2: The proposed mapping approach.

3.1 HF data gathering

To gather the necessary data, we utilized the latest available dump from the HF community website,[3] specifically from October 2023. We deployed this dump locally into a MySQL database to expedite the data retrieval process. Within the scope of this paper, our focus is on extracting textual documentation, as represented by the model card, and the *pipeline tags*. Additionally, we collected information for each pre-trained model, including the number of likes and downloads, facilitating further qualitative analysis discussed in Section 4. For these purposes, we employed the dedicated Python library[4] to interact with the Hugging Face dump.

Listing 1: SQL query.

```
SELECT model_id,card_data,pipeline_tag,likes,downloads FROM
      model,repository WHERE model.model_id = repository.id;
```

The SQL query used for this interaction is provided in Listing 1. This process resulted in a CSV file containing 381,240 PTMs along with the aforementioned metadata.

3.2 SE task filtering

We followed the steps shown in Figure 3 to collect SE tasks that are supported by PTMs. In particular, the process starts from the initial work by Gong et al. [12] identifying 13 SE tasks supported by PTMs from 37 papers. Though it is recent work, we expanded this initial mapping since PTM-related technologies are fast-evolving [9]. To this end, we executed a query on the Scopus digital library[5] with the following set of keywords: *(i)* pre-trained model* OR PTM* OR large language model* OR LLM OR transformer*; *(ii)* AUT support* OR recommend* OR task* OR automat*; *(iii)* requirement* OR develop* OR source code. All such keywords are combined by using the AND operator. We considered papers published over the last five years in top-tier SE venues, e.g., ICSE, ASE, and TSE. Through the executed query, we retrieved 80 papers and 46 tasks for which PTMs were employed. Afterwards, we refined the result by *(i)* merging the papers and SE tasks that were in common with those defined by Gong et al., and *(ii)* applying a set of inclusion and exclusion criteria to the titles and abstracts. In particular, we included all the approaches that reuse PTMs to cover different SE tasks, e.g., code completion, bug fixing, or vulnerability assessment. In contrast, foundational papers on PTMs and works that combine PTMs with other techniques were filtered out from the final list. In addition, we excluded empirical studies that investigate qualitative aspects of PTMs, e.g., carbon emission or advanced fine-tuning strategies.

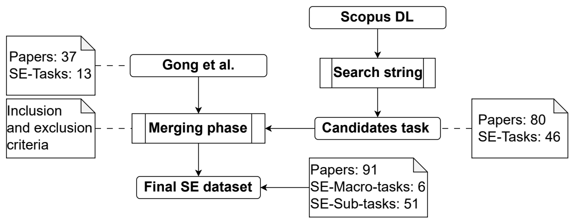


Figure 3: The performed process to elicit SE tasks.

We eventually identified 91 papers and 51 SE tasks. Since some of the collected papers address a very specific task, e.g., *Smart contract generation* or *Software traceability*, we refined those categories by manually grouping SE tasks into 6 macro tasks[6] as reported in Section 4, e.g., *Code-related tasks* or *Documentation support.*

3.3 Mapping phase

The proposed mapping algorithm is represented in Algorithm 1. While a comprehensive mapping between each identified SE task and every individual pre-trained model stored on Hugging Face is out of the scope of this paper, we present a generic approach that holds applicability to model repositories resembling HF.

Algorithm 1 Pseudocode for the mapping phase

```
1: Input: $\langle ptm : PTM, t : SETask, HF\rangle$
2: Output: $MAPPING$
3: $MAPPING \leftarrow \{\}$
4: for $ptm_i \in HF$ do
5:     if MATCH($ptm_i, ptm$) then
6:         $MAPPING.add\ (\langle ptm_i.pipelinetag, t\rangle)$
7:     end if
8: end for
9: function MATCH($PTM, PTM^*$)
10:     $\alpha \leftarrow$ SIM($PTM, PTM^*$)
11:     return $\alpha > T$
12: end function
```

[2]https://github.com/topics
[3]https://som-research.github.io/HFCommunity/download.html
[4]https://pypi.org/project/mysql-connector-python/
[5]https://www.scopus.com/
[6]Interested readers can refer to the complete list of macro and sub-tasks in the online appendix https://tinyurl.com/2p8y25cr

Table 1: Filtering process to create D_f.

	#PTMs	#pipeline tags
PTMs in the initial dump	381,240	40
PTMs with missing data	241,091	-
PTMs with support≤ α and downloads≤ β	4,234	21
D_f	**135,915**	**19**

As shown in Algorithm 1, there are three main inputs: a pre-trained model denoted as *ptm*, a corresponding SE task labeled as *t*, and the entire repository *HF* (Line 1). For each pre-trained model ptm_i stored in the specified repository (Line 4), the algorithm analyzes its name and calculates the similarity with the name of the input `ptm` (Line 5). If the similarity surpasses the designated threshold `T` (Line 11), it signifies that ptm_i is a relevant pre-trained model to support the SE task *t*. Consequently, the algorithm establishes a mapping between the pipeline tag of ptm_i and the SE task *t* provided as input (Line 6).

Intuitively, given the PTM name that is applied on a given SE task, we tried to find a list of stored PTMs using the similarity between the given model name and different versions available on the dump. In this paper, we employed the Levenshtein distance metric [21] as similarity function with T=0.8 as threshold.

Listing 2: Example of the similar models retrieved for RoBERTa

```
('sloberta', 'fill-mask'),
('roberta-go', 'fill-mask')
('me-roberta', 'fill-mask')
('am-roberta', 'fill-mask')
('numroberta', 'fill-mask')
```

Listing 2 shows an explanatory result, which is obtained by considering RoBERTa [18] as ptm model and Code related-task as SE task. By running the query on the HF dump, we got that the similar PTMs are labeled as `fill-mask`. Thus, we can establish a mapping between the pipeline tag `fill-mask` with the `Code-related SE` task.

It is worth noting that the PTMs belonging to the same family can have different pipeline tags. Similarly, the same model can be employed for different SE tasks. We try to mitigate these issues by *(i)* filtering HF data by using quality filters; and *(ii)* grouping similar SE tasks into macro tasks. The results of the mapping are discussed in Section 4.

4 PRELIMINARY EVALUATION

This section presents the evaluation conducted to answer the aforementioned research questions, and highlights the threats to validity of our findings.

4.1 Addressing RQ_1

To address this question, we initiated the process by refining the initial dump in Section 3.1, through the application of well-established preprocessing steps commonly utilized in the realm of automatic categorization for heightened overall accuracy [6, 16].

The steps involve filtering and shaping the final dataset, denoted as D_f (see Table 1). Initially, we excluded PTMs lacking essential data, such as pipeline tags or the model card. Subsequently, a further set of PTMs and tags were eliminated, employing two distinct thresholds: support=α and downloads=β, where α is the median value and β is the mean value of the corresponding parameters (the attributes "support" and "downloads" refer to the number of models for a specific category, and the number of downloads for a given PTM, respectively).

Table 2: D$_f$ Results.

	Precision		Recall		F1 Score	
Fold	**CNB**	**SVC**	**CNB**	**SVC**	**CNB**	**SVC**
1	0.892	**0.940**	0.890	**0.938**	0.887	**0.938**
2	0.888	**0.935**	0.886	**0.933**	0.882	**0.932**
3	0.893	**0.936**	0.889	**0.933**	0.886	**0.933**
4	0.891	**0.939**	0.889	**0.936**	0.886	**0.936**
5	0.892	**0.936**	0.889	**0.933**	0.886	**0.933**
6	0.892	**0.936**	0.891	**0.934**	0.888	**0.934**
7	0.887	**0.934**	0.883	**0.931**	0.880	**0.931**
8	0.894	**0.942**	0.891	**0.940**	0.887	**0.940**
9	0.890	**0.939**	0.887	**0.936**	0.883	**0.936**
10	0.887	**0.938**	0.884	**0.935**	0.881	**0.935**
Average	0.891	**0.938**	0.888	**0.935**	0.885	**0.935**

Afterward, we performed the cross-fold validation [22] of two state-of-the-art binary classifiers, i.e., CNB and SVC models, and widely-adopted accuracy metrics, i.e., *Precision*, *Recall*, and *F1-score* [2]. Table 2 shows the outcomes of the performed evaluation by considering the PTM stored on HF as the input and the `pipeline tag` as the predict label.

It is worth noting that the two employed models obtain decent performance, with the SVC model outperforming the CNB network by 10% on average in terms of the considered accuracy metrics. However, we report that 241,091 PTMs in the dump have no model cards as shown in Table 1.

Answer to RQ$_1$

Our findings confirm that model cards can be used to enable traditional classifiers based on textual content, even though most of the developers do not upload this information on the platform.

4.2 Addressing RQ_2

The outcome of the SE task filtering process described in Section 3.2 is reported in Figure 4. In particular, we grouped similar SE tasks into six different *macro-tasks* as follows:

- *M1-Code-related tasks:* This category includes PTMs that have been used for several code-related tasks, from suggesting libraries to code summarization.
- *M2-Program repair:* Compared to the previous category, we consider in this macro-task PTMs that have been applied to specific testing and program repair tasks e.g., bug fixing or vulnerability detection in the code.
- *M3-Documentation support:* We classify in this category PTMs that help developers generate textual documentation for different SE artifacts, i.e., code, commits or bug reports.

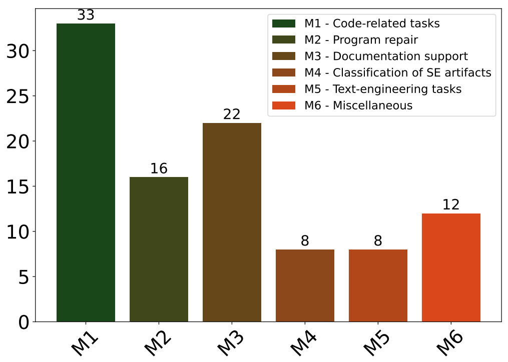


Figure 4: The identified SE macro tasks.

- *M4-Classification of SE artifacts:* This category identifies PTMs that perform automatic classification of specific artifacts, e.g., SO posts, commits, and issue reports.
- *M5-Text-engineering tasks:* Unlike documentation-related tasks, we labeled PTMs that generate a textual description of specific SE artifacts, e.g., titles for GitHub issues, Stack Overflow posts, or pull request descriptions.
- *M6-Miscellaneous:* This category includes PTMs that cannot be categorized in one of the previous macro-tasks. For instance, PTMs that support sentiment analysis in SE fall in this category.

As expected, most of the PTMs are used to address *code-related* tasks, i.e., 33 papers employed them for code generation, code summarization, or code2code translation. On the contrary, the application to more generic tasks such as *text-engineering* or *classification of SE artifacts* is limited (we collected only 8 supporting papers for these two categories). Afterward, we run the algorithm shown in Section 3.3 for three well-adopted PTMs, i.e., `BERT`, `RoBERTa`, and `T5`. To this end, we first retrieved a list of similar PTMs stored on HF to find the most frequent `pipeline tags` used on the dump. We eventually searched the PTM name in the abstract of the retrieved papers to get the corresponding SE macro-tasks and sub-tasks. The results are reported in Table 3.

Table 3: Explanatory mapping using our approach.

PTM	Pipeline tag	Macro SE tasks
BERT	text-classification	M1,M2,M3,M4,M5,M6
RoBERTa	fill-mask	M1, M3, M4
T5	text2text-generation	M1, M2, M3, M5

Notably, `BERT` has been used to cover the whole set of SE macro tasks even though it may be not suitable for generation tasks. Remarkably, we observe that `T5` is employed to address both code-related and documentation tasks, indicating the adaptability of text-to-text generation strategies even in the context of source code. The trend reveals a significant surge in the utilization of PTMs across diverse SE tasks, extending beyond the realm of code-related activities. In this context, our approach is envisioned as an initial solution to aid developers in selecting a specific PTM tailored to the current SE task.

Answer to RQ$_2$

Although we tested only three PTMs, the proposed mapping could be adopted to assist developers in selecting specific models that are used in practice.

4.3 Threats to validity

- *Internal validity* concerns the employed dump to perform the automatic classification given the model card, wherein the data may be incomplete or unbalanced. To mitigate this, we applied standard filtering techniques on the original dataset (Table 1). Moreover, there is a possibility of the mapping algorithm retrieving erroneous matches, where the HF model name may differ from the actual one. To mitigate this, we set a higher threshold similarity for the Levenshtein function.
- Threats to *external validity* are related to the generalizability of our approach considering *(i)* additional papers; and *(ii)* other model repositories beyond Hugging Face. Concerning the first aspect, we expanded the initial taxonomy by querying the Scopus digital library to exclusively collect peer-reviewed papers. By the second aspect, our proposed approach is adaptable to any repository that exposes a curated list of tags and textual documentation for each stored model.

5 RELATED WORK

Several approaches have been proposed to predict and recommend GitHub topics. Di Sipio et al. [8] proposed a multi-label stochastic classifier that predicts featured topics given a README file. The same authors extended their approach by employing a collaborative filtering engine to increase the topic coverage [6]. SED-KG [16] combines the BERT model with a topic model based on semantic relationships and knowledge graphs (KG) to enhance state-of-the-art models. GitRanking [25] exploits a semi-automated technique to create a ranked taxonomy of GitHub projects in terms of SE technologies. ZestXML [27] is an approach based on extreme multi-labeling learning conceived to outperform traditional approaches. Di Rocco et al. [6] conceived a hybrid recommender system that employs a multi-label stochastic classifier and collaborative filtering algorithm to predict featured topics given a README file. SED-KG [16] combines the BERT model with a topic model based on semantic relationships and knowledge graphs (KG) to enhance state-of-the-art models. Similarly, GHTRec [29] employs BERT to predict an initial set of topics that have been used to retrieve most similar repositories given the input one. GitRanking [25] exploits a semi-automated technique to create a ranked taxonomy of GitHub projects in terms of SE technologies. ZestXML [27] is an approach based on extreme multi-labeling learning conceived to outperform traditional approaches.

While several studies highlight the limitation of existing PTM repositories [12, 17, 20], to the best of our knowledge there has been no prior approach to automatically classify PTMs. In this context, we can redefine the challenge of categorizing generic open-source software within the framework of PTM repositories, such as Hugging Face. Additionally, we introduce an initial semi-automated approach to establish a mapping between generic categories and SE-specific tasks.

6 CONCLUSION AND FUTURE WORK

Aiming to map generic PTM categories with specific SE tasks, this paper envisions a semi-automated strategy based on a similarity-based algorithm. We first collected metadata from Hugging Face to foster automatic categorization of the stored PTMs. Then, we built an SE taxonomy by grouping tasks that exploit PTMs into macro and sub-tasks. The mined data has been used to *(i)* automatically categorize PTMs by exploited data available in the corresponding model cards and *(ii)* map generic PTMs categories to corresponding SE tasks. Our findings demonstrate that our methodology can help developers browse Hugging Face even though further experiments are needed to confirm our initial results.

For future work, we aim to improve the mapping algorithm by using well-known strategies to build software taxonomy e.g., active sampling algorithm. Furthermore, we can apply our approach to additional public repositories of PTMs. Last but not least, we can envision a set of recommender systems that rely on our approach to assist developers in deploying the chosen PTMs, thus fostering the creation of an AI-based multi-agent framework tailored for SE tasks.

ACKNOWLEDGMENTS

This work has been partially supported by the EMELIOT national research project, which has been funded by the MUR under the PRIN 2020 program grant n. 2020W3A5FY. The work has been also partially supported by the European Union–NextGenerationEU through the Italian Ministry of University and Research, Projects PRIN 2022 PNRR *"FRINGE: context-aware FaiRness engineerING in complex software systEms"* grant n. P2022553SL. We acknowledge the Italian "PRIN 2022" project TRex-SE: *"Trustworthy Recommenders for Software Engineers,"* grant n. 2022LKJWHC.

REFERENCES

[1] Adem Ait, Javier Luis Cánovas Izquierdo, and Jordi Cabot. 2023. HFCommunity: A Tool to Analyze the Hugging Face Hub Community. In *Procs. of SANER 2023.* 728–732. https://doi.org/10.1109/SANER56733.2023.00080 ISSN: 2640-7574.

[2] Michael Buckland and Fredric Gey. 1994. The relationship between recall and precision. *Journal of the American society for information science* 45, 1 (1994), 12–19. Publisher: Wiley Online Library.

[3] Joel Castaño, Silverio Martínez-Fernández, Xavier Franch, and Justus Bogner. 2023. Analyzing the Evolution and Maintenance of ML Models on Hugging Face. https://doi.org/10.48550/arXiv.2311.13380 arXiv:2311.13380 [cs].

[4] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology* 2, 3 (April 2011), 1–27. https://doi.org/10.1145/1961189.1961199

[5] Juri Di Rocco, Davide Di Ruscio, Claudio Di Sipio, Phuong T. Nguyen, and Riccardo Rubei. 2021. Development of recommendation systems for software engineering: the CROSSMINER experience. *Empirical Software Engineering* 26, 4 (July 2021), 69. https://doi.org/10.1007/s10664-021-09963-7

[6] Juri Di Rocco, Davide Di Ruscio, Claudio Di Sipio, Phuong T. Nguyen, and Riccardo Rubei. 2023. HybridRec: A recommender system for tagging GitHub repositories. *Applied Intelligence* 53, 8 (April 2023), 9708–9730. https://doi.org/10.1007/s10489-022-03864-y

[7] Claudio Di Sipio, Riccardo Rubei, Juri Di Rocco, Davide Di Ruscio, and Phuong T. Nguyen. 2024. *Replication Package: Automated categorization of pre-trained models for software engineering: A case study with a Hugging Face dataset.* https://github.com/MDEGroup/EASE2024-HF-ReplicationPackage

[8] Claudio Di Sipio, Riccardo Rubei, Davide Di Ruscio, and Phuong T. Nguyen. 2020. A Multinomial Naïve Bayesian (MNB) Network to Automatically Recommend Topics for GitHub Repositories. In *Procs. of the Evaluation and Assessment in Software Engineering.* ACM, Trondheim Norway, 71–80. https://doi.org/10.1145/3383219.3383227

[9] Malinda Dilhara, Ameya Ketkar, and Danny Dig. 2021. Understanding Software-2.0: A Study of Machine Learning Library Usage and Evolution. *ACM Trans. Softw. Eng. Methodol.* 30, 4, Article 55 (jul 2021), 42 pages. https://doi.org/10.1145/3453478

[10] Zishuo Ding, Heng Li, Weiyi Shang, and Tse-Hsun Peter Chen. 2022. Can pre-trained code embeddings improve model performance? Revisiting the use of code embeddings in software engineering tasks. *Empirical Software Engineering* 27, 3 (March 2022), 63. https://doi.org/10.1007/s10664-022-10118-5

[11] Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. 2023. Self-collaboration Code Generation via ChatGPT. arXiv:2304.07590 [cs.SE]

[12] 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 Trans. on Software Engineering and Methodology* 32, 3 (May 2023), 69:1–69:57. https://doi.org/10.1145/3569934

[13] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, et al. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250. https://doi.org/10.1016/j.aiopen.2021.08.002

[14] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, et al. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv:2308.00352 [cs.AI]

[15] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, et al. 2023. Large Language Models for Software Engineering: A Systematic Literature Review. https://doi.org/10.48550/arXiv.2308.10620 arXiv:2308.10620 [cs].

[16] Maliheh Izadi, Mahtab Nejati, and Abbas Heydarnoori. 2023. Semantically-enhanced topic recommendation systems for software projects. *Empirical Software Engineering* 28, 2 (Feb. 2023), 50. https://doi.org/10.1007/s10664-022-10272-w

[17] Wenxin Jiang, Nicholas Synovic, Matt Hyatt, Taylor R. Schorlemmer, Rohan Sethi, et al. 2023. An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry. In *Procs. of ICSE 2023.* IEEE Press, Melbourne, Victoria, Australia, 2463–2475. https://doi.org/10.1109/ICSE48619.2023.00206

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL]

[19] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, et al. 2019. Model Cards for Model Reporting. In *Procs. of the Conf. on Fairness, Accountability, and Transparency* (Atlanta, GA, USA) *(FAT*'19).* ACM, 220–229. https://doi.org/10.1145/3287560.3287596

[20] Diego Montes, Pongpatapee Peerapatanapokin, Jeff Schultz, Chengjun Guo, Wenxin Jiang, et al. 2022. Discrepancies among pre-trained deep neural networks: a new threat to model zoo reliability. In *Procs. of ESEC/FSE 2022.* ACM, 1605–1609. https://doi.org/10.1145/3540250.3560881

[21] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *Comput. Surveys* 33, 1 (2001), 31–88. https://doi.org/10.1145/375360.375365

[22] Payam Refaeilzadeh, Lei Tang, and Huan Liu. 2009. *Cross-Validation.* Springer US, Boston, MA, 532–538. https://doi.org/10.1007/978-0-387-39940-9_565

[23] Jason D M Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. 2003. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. (2003).

[24] Martin P. Robillard, Walid Maalej, Robert J. Walker, and Thomas Zimmermann (Eds.). 2014. *Recommendation Systems in Software Engineering.* Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-45135-5

[25] Cezar Sas, Andrea Capiluppi, Claudio Di Sipio, Juri Di Rocco, and Davide Di Ruscio. 2023. GitRanking: A ranking of GitHub topics for software classification using active sampling. *Software: Practice and Experience* 53, 10 (Oct. 2023), 1982–2006. https://doi.org/10.1002/spe.3238

[26] Rosalia Tufano, Simone Masiero, Antonio Mastropaolo, Luca Pascarella, Denys Poshyvanyk, et al. 2022. Using pre-trained models to boost code review automation. In *Procs. of the 44th Int. Conf. on Software Engineering (ICSE '22).* ACM, 2291–2302. https://doi.org/10.1145/3510003.3510621

[27] Ratnadira Widyasari, Zhipeng Zhao, Thanh Le Cong, Hong Jin Kang, and David Lo. 2023. Topic Recommendation for GitHub Repositories: How Far Can Extreme Multi-Label Learning Go?. In *2023 IEEE Int. Conf. on Software Analysis, Evolution and Reengineering (SANER).* IEEE, Taipa, Macao, 167–178. https://doi.org/10.1109/SANER56733.2023.00025

[28] Jialu Zhang, Todd Mytkowicz, Mike Kaufman, Ruzica Piskac, and Shuvendu K. Lahiri. 2022. Using pre-trained language models to resolve textual and semantic merge conflicts (experience paper). In *Procs. of ISSTA 2022.* ACM, 77–88. https://doi.org/10.1145/3533767.3534396

[29] Yuqi Zhou, Jiawei Wu, and Yanchun Sun. 2021. GHTRec: A Personalized Service to Recommend GitHub Trending Repositories for Developers. In *IEEE Int. Conf. on Web Services.* IEEE, 314–323. https://doi.org/10.1109/ICWS53863.2021.00049