# Pre-Trained Model Naming Conventions

SHIVA SAI VUMMAJI, Purdue University, U.S.

**ACM Reference Format:**
Shiva Sai Vummaji. 2024. Pre-Trained Model Naming Conventions. 1, 1 (May 2024), 4 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 ABSTRACT

### 1.1 Overview

Pre-Trained models (PTMs) are neural networks that have been trained on large datasets. They are becoming increasingly popular among researchers in today's world due to their ability to be fine-tuned for specific applications.

### 1.2 Current Knowledge

When researchers publish PTMs, they are reused by other users. However, model cards (README) and metadata for many models are insufficient or missing information, which limits how users can reuse PTMs. As a result, many users rely on model names for information about a PTM's capabilities and uses.

### 1.3 Gap

However, there isn't a standard naming scheme currently, which results in many discrepancies in PTM naming. Specifically, when researchers do not choose appropriate model names, the reuse of PTMs becomes more difficult.

### 1.4 Approach

We analyzed naming patterns for various models from Hugging Face in order to establish a proper naming convention to foster easy communication among engineers on the platform.

### 1.5 Results

We labeled a dataset to find potential outliers and analyzed model architectures for these outliers to establish a more precise taxonomy. We also tried creating pipelines to automatically detect mislabeled models and inconsistencies in cluster groups.

## 2 INTRODUCTION

Pre-Trained Models are gaining popularity among engineers as they can be fine-tuned for specific tasks and improve performance. Specifically, they widely being used in image recognition, natural language processing (text summarizing, language translation, chatbots), and speech recognition. Due to the widespread reuse of PTMs, choosing appropriate model names is crucial since it provides users with knowledge about architecture, model size, and capabilities, which facilitates the reuse process.

We analyzed a dataset of "cluster" models, or in other words, models that were clustered based on a specific pattern. The dataset included various architectures and a set of cluster groups for each architecture (ex. 0, 1, etc). For instance, BERT (architecture) had 3 cluster groups (0, 1, and 2), and each cluster group contained a set of models.


Fig. 1. Bert Models in cluster group 0 from the clusters file.

Given a dataset of several models on open and closed model hubs (Hugging Face was open and Onnx Model Garden/TensorFlow were closed), we identified models that followed improper naming conventions based on our cluster dataset. Based on the results, we identified 10 mislabeled models, which were further analyzed to provide a more detailed taxonomy.

In the cluster dataset, we created a Python pipeline to automatically identify models that were "uninformative," meaning that they do not contain architecture's name in the model name. For instance, consider the model *l3cube-pune/MarathiSentiment*. This model is under Albert architecture; however, it does not contain Albert anywhere in the model name. Additionally, we attempted to create a pipeline that identifies any inconsistencies in a specific cluster

Author's Contact Information: Shiva Sai Vummaji, svummaji@purdue.edu, Purdue University, West Lafayette, Indiana, U.S..


ACM XXXX-XXXX/2024/5-ART
https://doi.org/XXXXXXX.XXXXXXX

group. For instance, if a cluster group contained a model that did not belong there (instead, if it matched patterns in another cluster group), the pipeline would be able to identify it and provide the cluster group that is a better match.

## 3 BACKGROUND

Previously, we created an automated pipeline that could identify predefined naming defects. This involved parsing through the clusters dataset and identifying models that do not contain architecture's name in the model name (example mentioned in the introduction). The dataset was structured with key, value pairs, where the architectures (represented as strings) are keys and the "clusters" (represented as dictionaries) are values, and each dictionary contains a string ("0", "1", etc), which represents the cluster group, and values (the model names, represented as arrays of strings).

We created the pipeline by looping through each architecture's cluster groups, saving the specific architecture name, and checking whether the model name contains the architecture's name. However, although this method works for a majority of the models, there were cases where it was unsuccessful. For instance, consider the model *uer/roberta-large-wwm-chinese-cluecorpussmall*. This model has Bert architecture; however the model name contains Roberta. According to our previous method, this model would not be detected as a defect since Ro**bert**a contains Bert in its name.

Additionally, our first task last semester was to classify a dataset of models from Hugging Face and Onnx Model Garden/TensorFlow. Specifically, we identified all potential models that were incorrectly labeled or followed improper naming conventions. We compared this dataset with our cluster dataset to identify any inconsistencies within architectures, authors, fine-tuning, mislabeling, or underlabeling. Our results indicated that among the models from Hugging Face, there were 10 outliers.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | https://huggingface.co/damlab/GO-language | damlab/GO-language | Bert | Yes | Mislabel | Model Card | Yes |
| 9 | https://huggingface.co/uer/chinese_roberta_L-12_H-256 | uer/chinese_roberta_L-12_H-256 | Bert | Yes | Underlabel | Paper | Yes |
| 11 | https://huggingface.co/bs-la/bloomz-7b1-4b-xp3ru | bs-la/bloomz-7b1-4b-xp3ru | Bloom | Yes | Finetuned, di | config | |
| 71 | https://huggingface.co/facebook/nllb-200-3.3B | facebook/nllb-200-3.3B | M2M100 | Yes | Mislabel | Model card | Yes |
| 72 | https://huggingface.co/uer/t5-v1_1-base-chinese-cluecorpussmall | uer/t5-v1_1-base-chinese-cluecorpussmall | MT5 | Yes | Underlabel | Model card | Yes |
| 101 | https://huggingface.co/Intel/xlnet-base-cased-mrpc-int8-static | Intel/xlnet-base-cased-mrpc-int8-static | XLNet | Yes | Underlabel | Model card | |
| 113 | https://huggingface.co/hajime9652/xlnet-japanese | hajime9652/xlnet-japanese | XLNetLMHead | Yes | Added additi | Model card | |
| 119 | https://huggingface.co/hkunlp/instructor-large | hkunlp/instructor-large | T5Encoder | Yes | No Cluster, S | Model name | |
| 120 | https://huggingface.co/hkunlp/instructor-xl | hkunlp/instructor-xl | T5Encoder | Yes | No Cluster, S | Model name | |
| 121 | https://huggingface.co/hkunlp/instructor-base | hkunlp/instructor-base | T5Encoder | Yes | No Cluster, S | Model name | |

Fig. 2. 10 outliers with information about model name, link to Hugging Face page, model architecture, and why it was an outlier.

## 4 RELATED WORK

This area of PTM naming is a relatively new field with little research. However, there do exist certain model registries that suggest specific naming conventions contributors should follow. In particular, OpenMMLab advises contributors to publish model names that include 5 parts: algorithm information, module information, pretrain information, training information, and data information, where each part is separated by an underscore. The website also mentions that algorithm and pretrain information can be optional. Figure 3 provides an example of a model name that follows the specific naming convention described.

This naming convention can be very beneficial to many engineers since it provides important information about a model's capability that is needed to facilitate reuse. However, it is important to note

```
beit_beit-base-p16_8xb256-amp-coslr-300e_in1k
```

- `beit` : The algorithm information
- `beit-base` : The module information, since the backbone is a modified ViT from BEiT, the backbone name is also `beit` .
- `8xb256-amp-coslr-300e` : The training information.
  - `8xb256` : Use 8 GPUs and the batch size on each GPU is 256.
  - `amp` : Use automatic-mixed-precision training.
  - `coslr` : Use cosine annealing learning rate scheduler.
  - `300e` : To train 300 epochs.
- `in1k` : Dataset information. The model is trained from ImageNet-1k dataset and the input size is `224x224` .

Fig. 3. An example model name from OpenMMLab.

that the page does not support this model naming convention with any research or data.

Additionally, Wenxin's paper exploring naming practices of PTMs in Hugging Face provides a technique to effectively detect naming anomalies using only architectural information: DNN Architecture Assessment (DARA). Their results suggest that DARA has a 92% accuracy in being able to detect naming anomalies using architectural information.

Studies show that naming is a very important part of software (non AI/ML) engineering, as users understand code through identifier names and comments. Specifically, an article focusing on identifier names in systems and software engineering mentions that when functions do not have any comments, engineers rely heavily on identifier names.

## 5 PROBLEM STATEMENT

After identifying the 10 outliers from the internal outliers (Hugging Face) dataset, we wanted to establish a more precise taxonomy, so we compared the outlier models with other models from the architecture's cluster group.

Additionally, we wanted to improve our pipeline that detects "uninformative" identifiers, models that do not include their architecture name in the model name, by addressing scenarios that were unsuccessful (example in background).

Our third goal was to create a pipeline that automatically identifies inconsistencies within cluster groups. Our pipeline would identify if a specific model does not belong in a cluster group (if it matches more closely with another cluster group) and provide what cluster group it should belong to (matches more closely with) instead.

## 6 METHODOLOGY

### 6.1 Taxonomy for Outlier Models

In order to identify and establish clear differences between the 10 outlier models and cluster models, we compared each outlier model with the most downloaded models from each cluster group for the specific architecture. We created a Python script that identifies the most downloaded model for each cluster group for a specific architecture. For instance, consider the first outlier model *damlab/GO-language*, which is a BERT model. We visualized this model's architecture using Netron. Then, we compared this outlier model's visualization

with that of the most downloaded models from each cluster group for BERT architecture.

### 6.2 Improve Pipeline for detecting "uninformative" models

Since our initial pipeline to find "uninformative" identifiers was not fully successful, we wanted to improve it so that it can address failed cases. Our previous pipeline was not successful due to incorrect identifiers: if the model name contains Roberta but has a BERT architecture. The initial pipeine would not be able to detect this incorrect identifier because Roberta has BERT in its name. Our method to accomplish this goal was to create multiple cases and count the number of incorrect identifiers exist.

### 6.3 Pipeline to identify naming inconsistencies within cluster groups

In order to automatically identify inconsistencies within cluster groups, we first created a Python pipeline to identify semantic components: **architecture, size, version**, and compare discrepancies. However, we realized that this would not work as intended since we cannot prioritize one component over another (ex. architecture over version) and there are many ways that people can be inconsistent (wide variety of model name discrepancies). As a result, we decided to use a Large Language Model (LLM) to accomplish our goal. We provided a prompt that contains definition, task specifications, and an example. Figure 4 illustrates how this pipeline would work.

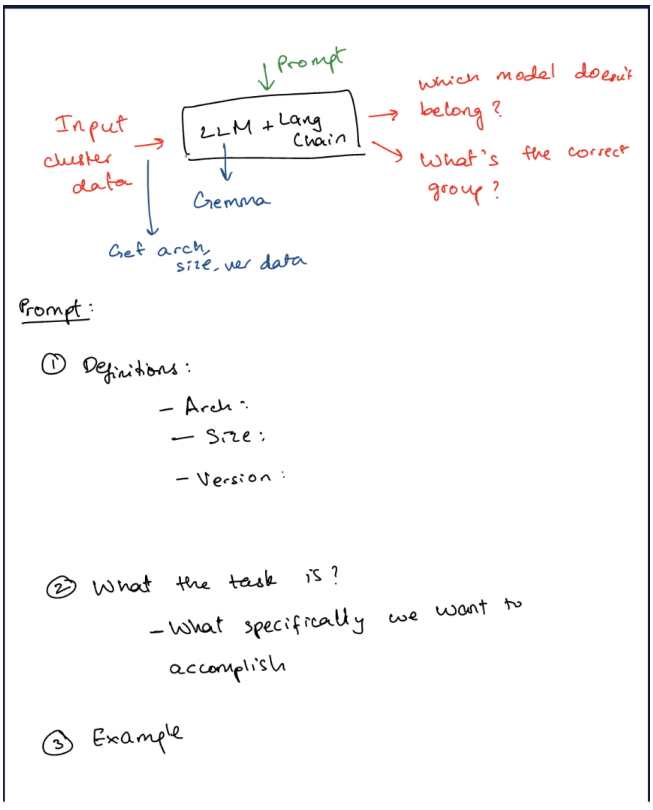


Fig. 4. Flowchart showing how the pipeline would work.

## 7 RESULTS AND ANALYSIS

### 7.1 Taxonomy for Outlier Models

We were able to compare the first few outlier models. Our results indicate that in some comparisons, there are a different number of blocks between the outlier models and the cluster models. Additionally, some outlier models also had different output parameters than the cluster model they were compared with. However, it is important to note that we were unable to visualize and compare the last few models, as we ran into many memory and compilation errors.

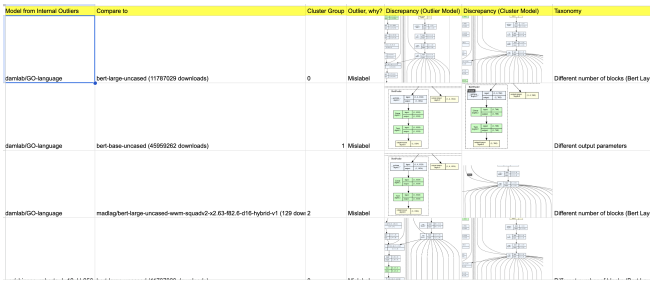


Fig. 5. This table visualizes and compares an outlier model with models from the cluster dataset.

### 7.2 Pipeline for detecting "uninformative" models

We were able to successfully improve our initial pipeline by addressing some incorrect identifiers. However, this pipeline still needs improvement because 28.63% incorrect models. We can further classify these incorrect identifiers into more specific categories.

```
(ptm_env) (base) shivasaivummaji@Alex-s-Z
Number of models in clusters: 7653
Number of mislabeled models: 2191
Percentage of mislabeled models: 28.63%
Number of architectures: 82
Number of uninformable models: 2041
Number of case 2 wrong identifiers: 150
Number of case 1 wrong identifiers: 976
(ptm_env) (base) shivasaivummaji@Alex-s-Z
```

Fig. 6. Result of the new pipeline for uninformative identifiers.

Additionally, it is important to note that our algorithm might not work as intended since we expected that less than 10% of the models would be mislabeled identifiers.

### 7.3 Pipeline to identify inconsistencies within cluster groups

Our pipeline used Gemma, Google's new LLM, to accomplish our goal. However, this pipeline was very inaccurate and was unable to successfully identify inconsistencies. Specifically, we provided this pipeline with the same cluster groups and inconsistency that was given as the example in our prompt. However, our pipeline

was unable to successfully accomplish our goal, even though we provided the answer in the example part of the prompt.

## 8 FUTURE WORK

We discuss three directions of future work.

**To improve taxonomy** Although we were able to compare outlier models with the most downloaded models from each cluster group, we can further improve this by comparing them with the top three models from each cluster group. For some cluster groups, the number of downloads for the top three models was very close. Additionally, we were not able to complete comparing all 10 outlier models. As a result, successfully completing the comparison of the remaining outliers can either support the current classifications or provide more variability and distinction between the outlier and cluster models.

**To detect uninformative identifiers** Although we were able to improve our initial pipeline by distinguishing incorrect and uninformative identifiers, we need to determine whether our algorithm functions as expected by testing and analyzing further. Additionally, we can also create specific categories for the incorrect identifiers. In particular, instead of classifying them all as incorrect identifiers, we can analyze patterns and differences between models and classify them accordingly.

**To identify inconsistencies within cluster groups** In order to improve our current pipeline, we can build a Retrieval-Augmented Generation (RAG) Model using Gemma (the LLM we used) and Langchain. By doing so, we will not only improve the accuracy of our pipeline, but also facilitate and enhance the process of obtaining semantic patterns (architecture, size, version).

## 9 CONCLUSIONS

Pre-Trained Models (PTMs) are widely being exchanged and reused by many engineers due to their ability to be fine-tuned for specific tasks. While PTMs offer many advantages in applications and performance, the absence of a standardized naming convention presents discrepancies, making reuse very difficult. Our study reveals the importance of having appropriate PTM names, as they can provide crucial information regarding a model's capabilities. We labeled outliers from a given dataset of Hugging Face models by comparing them with models in our cluster dataset. This provided insights into the architectural, and naming differences between outlier and cluster models. We then further classified how these outliers differ visually from cluster models. Additionally, we created a pipeline that would automatically identify models that do not have their architecture's name in the model name (uninformative identifier). Although we were able to improve this pipeline by addressing incorrect identifiers, we can further classify these incorrect identifiers into their own specific categories. We also attempted to create a pipeline that would leverage Gemma (LLM) and Langchain to automatically identify any inconsistencies or defects in cluster groups. However, the accuracy for this model was inadequate. This can be improved by building a RAG model and using it to improve accuracy and obtain semantic components (architecture, model size, version).

## 10 STATEMENT OF WORK

I was the only person on my team, so I did all of this.

## ACKNOWLEDGMENTS

To Wenxin and Professor Davis for their support throughout the last 2 semesters.

## REFERENCES

(1) OpenMMLab, "Convention in MMPretrain," in *MM Pretrain*, (n.d.) https://mmpretrain.readthedocs.io/en/latest/advanced_guides/convention.html
(2) W. Jiang, C. Cheung, M. Kim, H. Kim, G. K. Thiruvathukal, J. C. Davis, "Naming Practices of Pre-Trained Models in Hugging Face," *arXiv preprint arXiv: 2310.01642*, 2024.
(3) D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "Effective identifier names for comprehension and memory," *Innovations in Systems and Software Engineering*, vol. 3, no. 4, pp. 303–318, Nov. 2007. [Online]. Available: http://link.springer.com/10.1007/s11334-007-0031-2