



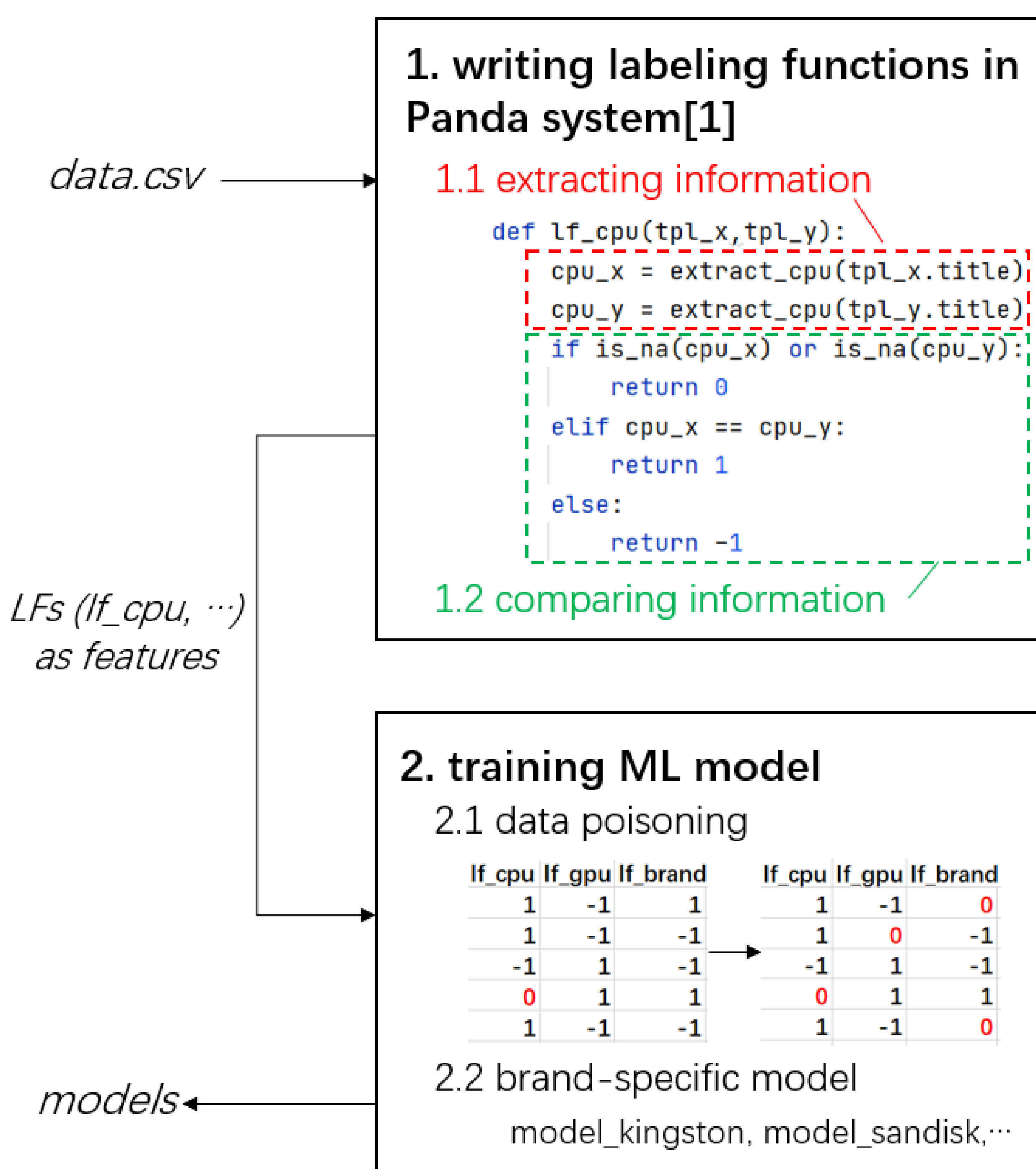
ACM SIGMOD PROGRAMMING CONTEST 2021

Team: panda (Runner up) Members: Renzhi Wu, Peng Li, Xu Chu (advisor)
Contact: {renzhiwu@, pengli@, xu.chu@cc.}gatech.edu

Task Overview

Task: Entity resolution on three different datasets.
For each dataset D_i , we are provided with
 X_i : a subset of the instances in D_i
 Y_i : matching/non-matching labels for pairs in $X_i \times X_i$
Metadata (e.g., how many instances it contains)
Solutions are evaluated over $Z_i = D_i \setminus X_i$
Performance metric: Averaged F1 score on the three datasets.

Solution Overview



Our solution consists of two major components:

- 1.** We write labeling functions (LFs) to provide many versions of weak labels based on different heuristics. This step requires huge human effort. We use the **Panda** system[1], an IDE for developing labeling functions, to reduce the pain.
- 2.** We use the output of the LFs as features and Y_i as labels to train ML models.

[1] R. Wu, P. Sakala, P. Li, X. Chu, and Y. He. Demonstration of panda: A weakly supervised entity matching system. Proc. VLDB Endow., 14(10):1952–1955, 2021.

Writing labeling functions in Panda

Labeling function: A labeling function typically has two components: **extracting** relevant information and **comparing** the extracted information.

For example, labeling function `lf_cpu` first extracts `cpu` types for two tuples (`tpl_x`, `tpl_y`) then compares the extracted `cpu` types to emit a weak label (1 for match, -1 for unmatched and 0 for failure of `cpu` extraction).

IDE for writing LFs: Writing a LF involves a lot of data exploration, data wrangling and debugging. We write LFs in **Panda**[1], an IDE with intuitive GUI providing various supports for writing LFs. For example, it automatically identifies informative tuple pairs and supports LF quality debugging.

Training ML model

Data poisoning: Information extraction is developed based on the training set and can easily fail on the unseen test set. This causes the **distribution** of the features of the training set and test set to be **different**.

For example, one may write a regular expression to extract the `cpu` type (e.g. `i5-530`), which, however, may fail to extract the `cpu` type with unexpected format (e.g. `intel core 7th gen 7700T`) in the unseen test set.

In the training set, we randomly inject zeros to the feature matrix (this is equivalent to injecting missing values to the extracted information) to simulate the failure of information extraction. This adapts the training distribution to the test distribution.

Brand-specific model: Different brands can have very different characteristics especially for difficult datasets (Altosight). On the Altosight dataset, we train a model for each brand (sandisk, Kingston, ...).

Performance

F1 scores on the three datasets:
Notebook: 0.972 NotebookLarge: 0.992 Altosight: 0.907
Averaged F1 score is 0.957.