Semantic Mapping of FWH Building Data

P23: Yujin Kim, ykim68 (https://github.com/ykim68ncstate/ncsu-engr-ALDA-F25-Project-P23)

1 Introduction

Labeling sensor data from building management systems into a standardized format is a time-consuming and error-prone task. Different buildings and systems often use inconsistent point names, making it difficult to integrate data for analytics, simulation, or digital twin applications.

Brick is an open-source schema that provides a unified, machine-readable framework for representing the relationships among building entities such as equipment, sensors, and control points. By defining consistent classes (e.g., Air Temperature Sensor, Zone, AHU) and relationships (feeds, isPointOf, etc.), Brick enables interoperability and automation across heterogeneous building systems. However, applying the Brick schema to existing datasets still requires significant manual labeling and expert knowledge, particularly when point names are inconsistent or incomplete.

The goal of this project is to develop an auto-labeling AI model that can automatically map building operation data to Brick ontology tags. This task addresses a key challenge in semantic modeling—automatically interpreting diverse and inconsistent point names or data streams into standardized representations.

By leveraging time-series features (such as mean, variance, range, and periodicity) extracted from raw building operation data, the project aims to train a model that predicts possible Brick tags (e.g., temperature, flow, status) without relying on manually standardized naming conventions.

2 Method

A complete data processing pipeline was designed and implemented to transform raw building operation data into machine-readable features suitable for AI training. The overall workflow consists of four major stages: data pre-processing, feature extraction, candidate generation, and model training.

2.1 Data pre-processing

The raw data was collected from a building's automation system and stored in an alternating column formant (timestamp, value, timestamp, value, ...).

A Python script was developed to automatically convert the data into a long-form table with unified timestamps. The script detects whether each point is ratio-type (continuous, e.g., temperature) or binary-type (on/off, e.g., status) and resample them at 15 minute intervals for consistency.

2.2 Feature extraction

For each point, statistical and temporal characteristics were calculated, including:

- Mean, standard deviation, and range
- Zero value ratio (for detecting flow or binary behavior)
- Daily and weekly periodicity (via autocorrelation analysis)

These features describe how each signal behaves over time, enabling the model to distinguish between different physical quantities without relying on textual names.

2.3 Candidate generation

A heuristic "candidate tag generator" was implemented to assign the top three probable tags for each point based on its feature profile For example, a signal with values between 60F and 90F showing strong daily patterns is likely a temperature sensor, while a binary signal toggling between 0 and 1 is likely a status or command. This automatic labeling produces a candidate table used both for inspection and model input.

2.4 Manual labeling and Gold tags

To establish a minimal training set, approximately 20-50 representative points were manually labeled with their correct tags (e.g., "temperature, sensor", "status"). These gold labels serve as ground truth for training and evaluating the baseline model.

3 Experiment Setup

3.1 Dataset

The experiment used operational data collected from the Fitts-Woolard Hall (FWH) building at North Carolina State University. The dataset consists of time-series signals from the building's automation system, primarily focusing on Air Handling Units (AHUs) and Variable Air Volume (VAV) boxes.

Each data point represents a physical or control variable measured at 15-minute intervals. The dataset includes both continuous (ratio-type) signals, such as temperature and flow rate, and binary (on/off) signals, such as equipment status and control modes. The data was exported in an alternating time–value format, which required transformation before analysis.

After preprocessing, the dataset contained approximately N points (each representing one sensor or control variable) with several months of historical data. Among them, around 20–50 representative points were manually annotated with Brick-like tags to create a small but reliable ground-truth set for model evaluation.

3.2 Feature construction

Each signal was processed through the preprocessing pipeline introduced in Section 2. For every point, statistical and temporal features were extracted to describe its dynamic behavior. These include:

- Statistical features: mean, standard deviation, range, coefficient of variation
- Temporal features: daily and weekly periodicity scores (autocorrelation-based)
- Behavioral features: zero ratio, switch count, on-time ratio for binary points

This feature set provided a compact numeric representation of each signal's operational characteristics, enabling the model to distinguish different sensor or control types purely based on behavior.

3.3 Gold tag labeling

To train and validate the model, a small labeled subset was created by manually assigning Brick-style tags to representative points. For example:

- "AHU-4 SaTmp" - temperature, sensor
- "AHU-4 HrwSts" - status
- "AHU-4 EconMd" - mode, status

These gold tags represent the true semantic type of each signal and serve as the ground-truth labels for training. All labels were stored in a CSV file (tag_candidate_sheet_labeled.csv) along with the model's heuristic tag predictions for comparison.

3.4 Model configuration

A multi-label classification model was implemented using the XGBoost framework, where each possible tag (temperature, flow, pressure, status, command, mode, etc.) is treated as an independent binary classification target under a One-vs-Rest scheme.

- Model type: XGBoost (gradient boosting trees)
- Parameters: n_estimators = 400, max_depth = 6, learning_rate = 0.05, subsample = 0.8, colsample_bytree = 0.8
- Evaluation metric: Macro F1-score (average over all tags)

At this point, the focus has been placed on building and validating the end-to-end learning pipeline. Only a simple cross-validation (CV) was conducted to confirm that the prototype model can process features correctly and produce meaningful tag predictions.

3.5 Evaluation procedure

Each fold in cross-validation used a stratified sampling approach to maintain a balanced distribution of tags across training and testing splits. Predictions were evaluated based on the overlap between predicted and true tags for each point. A tag was considered correctly predicted if its probability exceeded a threshold of 0.5.

The final Macro F1-score was computed by averaging F1 values across all tags and folds, providing an overall measure of how well the model generalizes to new, unlabeled points.

4 Results

The prototype multi-label classification model was trained and evaluated using the manually labeled subset of building operation data. The dataset included seven representative points from the AHU system, covering both continuous (temperature-related) and binary (status or mode) signals. Each point was associated with multiple tag candidates automatically generated during pre-processing, and corresponding gold tags were used as ground truth for model evaluation.

4.1 Quantitative results

Using 2-fold cross-validation, the prototype XGBoost model achieved a Macro F1-score of 0.40. This result indicates that the model was able to correctly identify several tag patterns, particularly distinguishing temperature-type signals from status-type binary signals. Compared to the initial baseline (Macro F1 = 0.22), this improvement demonstrates that the refined pre-processing pipeline, corrected unit normalization, and manually verified labels significantly enhanced model learning.

4.2 Qualitative results

Table 1 illustrates examples of the top-3 tag candidates automatically generated for each point, compared with manually assigned gold tags.

Table 1: Top-3 tag candidates automatically inferred by the model compared to gold labels.

Point ID	tag1	tag1_score	tag2	tag2_score	tag3	tag3_score	Gold Tags
AHU-4 AvgCcoilTmp	temperature	0.898	sensor	0.500	–	–	temperature
AHU-4 AvgMaTmp	temperature	0.887	sensor	0.500	–	–	temperature
AHU-4 ChwEnTmp	temperature	0.800	sensor	0.500	–	–	temperature
AHU-4 EconMd	status	0.650	command	0.552	mode	0.520	status
AHU-4 HrwClgMd	status	0.670	command	0.570	mode	0.536	status
AHU-4 HrwSts	status	0.650	command	0.552	mode	0.520	status
AHU-4 SaTmp	temperature	0.898	sensor	0.500	–	–	temperature

The model's predictions generally follow the human-verified tags (Gold Tags), demonstrating consistency between automated inference and manual annotation. This result validates the preprocessing and feature extraction pipeline designed to separate continuous variables from binary mode signals.

However, since the current dataset contains only a small number of samples, the model's ability to capture more fine-grained distinctions remains limited.

5 Conclusion

A complete framework for automated tag inference from building data was developed, capable of transforming raw time-series signals into semantically meaningful categories. The preliminary model results confirm the technical feasibility of this approach and establish the foundation for more scalable and intelligent metadata generation in the future.

In the next phase, the model will be refined through expanded labeling and feature enhancement, with the goal of developing a practical tool that supports large-scale semantic modeling across multiple buildings.

A Appendix / Next Tasks

A.1 Expand and diversify the labeled dataset

I will increase the number of manually verified samples to cover a broader range of building systems and sensor types. This expansion will provide a more balanced training dataset and enable the model to learn finer distinctions, such as differentiating between supply and return air temperature sensors.

A.2 Enhance model robustness and validation methods

The XGBoost-based prototype will be refined through hyperparameter tuning and feature optimization to improve generalization. In addition to cross-validation, I plan to apply multiple evaluation methods, including confusion matrices and per-class precision–recall analysis, to more clearly identify which tags are well recognized and which remain ambiguous. These evaluations will help quantify model stability and guide further improvements.

A.3 Implement query-based verification with the Brick schema

After tag prediction, I will perform query-based validation to check whether each auto-labeled point aligns correctly with the Brick ontology structure. For instance, predicted tags such as temperature sensor or status will be verified through SPARQL queries against the Brick schema to confirm valid class inheritance and relationships. This step will ensure semantic consistency and compliance with Brick standards.