Metadata for Energy Disaggregation

Jack Kelly and William Knottenbelt
Department of Computing, Imperial College London, UK.
e-mail: jack.kelly@imperial.ac.uk

***Abstract*—Energy disaggregation is the process of estimating the energy consumed by individual electrical appliances given only a time series of the whole-home power demand. Energy disaggregation researchers require datasets of the power demand from individual appliances and the whole-home power demand. Multiple such datasets have been released over the last few years but provide metadata in a disparate array of formats including CSV files and plain-text README files. At best, the lack of a standard metadata schema makes it unnecessarily time-consuming to write software to process multiple datasets and, at worse, the lack of a standard means that crucial information is simply absent from some datasets. We propose a metadata schema for representing appliances, meters, buildings, datasets, prior knowledge about appliances and appliance models. The schema is relational and provides a simple but powerful inheritance mechanism.**

I. Introduction

In the quest to design and implement a high performance energy disaggregation system, researchers require several types of data:

The primary requirement is for datasets which record the power demand of whole homes as well as the 'ground truth' power demand of individual appliances within the home. In 2011, researchers at MIT released the first public dataset for energy disaggregation research [4]. Since then, over ten more datasets have been released and, in March 2014, a project called Wiki Energy[1] launched to share datasets online. These datasets have been well received but, because each dataset uses a different file format, it is time consuming to import multiple datasets. This is an issue because an important criteria for evaluating any machine learning algorithm is how well it generalises across multiple datasets. A further challenge with existing datasets is that machine-readable metadata is often minimal and uses a schema and vocabulary unique to that dataset. At best, the lack of a standard metadata schema makes it time-consuming to write software to process multiple datasets. At worse, some datasets simply lack sufficient metadata to allow the data to be properly interpreted. For example, the mains wiring connecting meters to each other and to appliances forms a tree (with the whole-house meter at the root and appliances at the leaves). In some datasets, this tree structure has more than two levels (i.e. when an appliance turns on or off, the resulting change in power consumption is sensed by more than two meters) yet the metadata rarely specifies the wiring tree.

A secondary requirement is for data describing the behaviour of appliances (e.g. a probability distribution describing the typical times per day that each appliance is used). This prior knowledge can be used to fine-tune the estimates produced by a disaggregation system. Such data is currently available in research papers and industry reports but not in a machine-readable form.

Finally, consumers are unlikely to put effort into training a disaggregation tool. As such, if open-source disaggregation solutions such as NILMTK [1] are to be viable as consumer-facing disaggregation solutions then researchers must distribute pre-trained models for each appliance. If these models adhere to a standard metadata schema then multiple software systems can exchange models.

Against this background, we propose the first draft of a hierarchical metadata schema for energy disaggregation. Specifically, our schema models electricity meters, appliances (including prior knowledge such as probability distributions describing typical times of use and parameters describing inferred models of appliances), buildings and datasets.

Although we have made every effort to ensure that our proposed schema and controlled vocabularies capture the information present in all the datasets we are aware of, our schema can undoubtedly be improved and so the schema is presented as an open-source project[2] (under a permissive Apache 2.0 license) to which contributions are most welcome!

In this paper we first outline related work; then we describe the design of our metadata schema; then look at the implementation of a schema validator and finally we discuss conclusions. A longer version of this paper is available at [2].

II. Related work

The general principal of using metadata to describe research datasets is not new. For example, the not-for-profit organisation DataCite[3] (established in 2009) publishes the DataCite Metadata Schema for describing research datasets.

In late-2013, the Nature Publishing Group launched a journal called 'Scientific Data'[4] for describing datasets. The machine-readable description of each dataset is captured using the ISA_Tab metadata specification[5] which specifies a hierarchical schema consisting of the 'investigation' (the project context), the 'study' (a unit of research) and the 'assay' (an analytical measurement).

Biology researchers have embraced the need for metadata schemas and controlled vocabularies as demonstrated by, for example, The Open Biological and Biomedical Ontologies database[6] which aims to enable the creation of a suite of

[1] http://wiki-energy.org/

[2] https://github.com/nilmtk/nilm_metadata

[3] http://www.datacite.org

[4] http://www.nature.com/scientificdata

[5] http://isa-tools.org

[6] http://www.obofoundry.org

interoperable reference ontologies for the biomedical domain.

Few metadata projects are specifically for energy datasets. One notable project is Project Haystack[7] which is an open source initiative to develop taxonomies and tagging conventions for building equipment and operational data. Amongst other achievements, Haystack defines a language for describing electricity meters, including which parameters each meter records and the relationships between meters and between meters and loads. But Haystack is primarily targeted at large commercial buildings rather than domestic buildings, and does not define a controlled vocabulary for appliance names, let alone more granular detail about appliances.

The UK Energy Research Council's Energy Data Centre provides a simple schema[8] based on the Dublin Core Metadata Initiative[9] (DCMI).

The Power Consumption Database[10] is a community project which aims to collect a database of appliance power consumption information.

To summarise the related work: there are many metadata projects for describing research datasets in general but only a small number of metadata projects for describing energy datasets. To the best of our knowledge, there are no existing metadata schemas specifically for describing objects relevant to energy disaggregation. Existing datasets for energy disaggregation do provide some metadata (e.g. a text file mapping appliance names to recording channels) but this metadata does not use a controlled vocabulary and often provides scant details.

III. Design

The NILM Metadata schema models several objects relevant to energy disaggregation: electricity meters, appliances, prior knowledge about appliances, appliance models, buildings and datasets. The schema specifies property names for each object, the type for each value and controlled vocabularies (e.g. for appliance names and categories).

A UML Class diagram showing the relationship between classes is shown in Figure 1 and a brief example metadata instance is shown in Figure 2.

In the sections below, we describe our 'dataset' and 'building' schemas; the distinction between *meters* and *appliances*; the representation of electricity meters; the representation of the mains wiring; the inheritance mechanism for appliances; categorisation; the containment mechanism that allows an appliance to contain other appliances; prior knowledge; and finally our representation of learnt models.

A. Dataset

NILM Metadata places the primary objects of interest into a tree shaped hierarchy (Figure 1). At the root is a dataset object. This contains buildings, each of which contains electricity meters, many of which measure the power demand of appliances.

[7]http://project-haystack.org
[8]http://ukedc.rl.ac.uk/format.html
[9]http://dublincore.org
[10]http://www.tpcdb.com

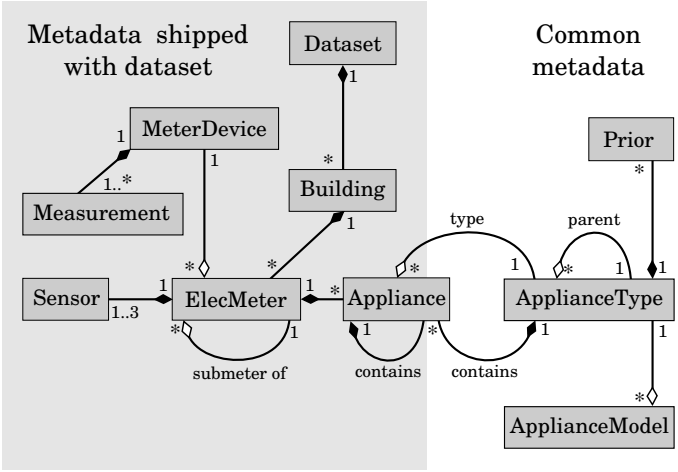


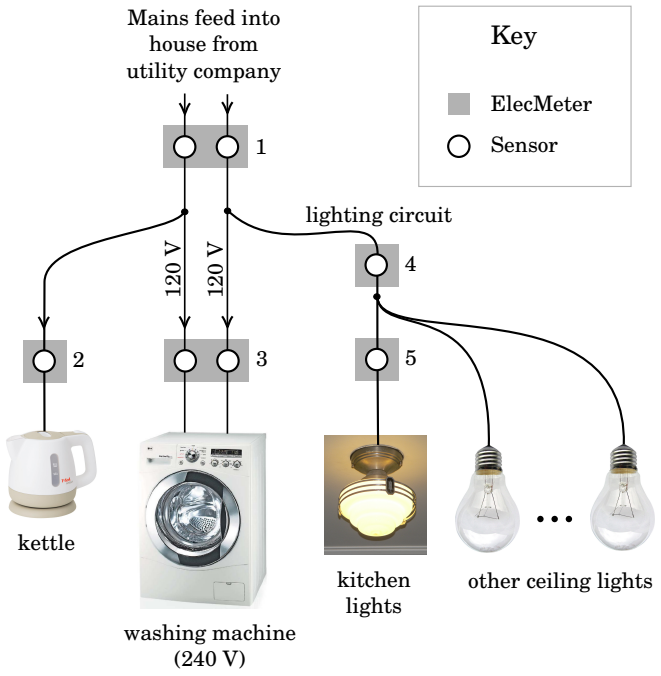
Fig. 1. UML Class Diagram showing the relationships between classes. A dark black diamond indicates a 'composition' relationship whilst a hollow diamond indicates an 'aggregation'. For example, the relationship between 'Dataset' and 'Building' is read as '*each Dataset contains any number of Buildings and each Building belongs to exactly one Dataset*'. We use hollow diamonds to mean that objects of one class refer to objects in another class. For example, each Appliance object refers to exactly one ApplianceType. Instances of the classes in the shaded area on the left are intended to be shipped with each dataset whilst objects of the classes on the right are common to all datasets and are stored within the NILM Metadata project. Some ApplianceTypes contain Appliances, hence the box representing the Appliance class slightly protrudes into the 'common metadata' area on the right.

B. Building

Buildings are identified by an integer property `instance` (unique within the dataset). Each building may have a list of rooms (using a controlled vocabulary for room names), and some properties shared with dataset: `temporal_coverage, geo_location, timezone`. These properties default to the values set in the parent dataset but can be overridden per building. Each building contains a `elec_meters` property which stores a list of ElecMeter objects.

C. Meters are distinct from appliances

A tempting simplification would be to assume a one-to-one relationship between electricity meters and appliances. But we often observe one-to-*many* relationships (e.g. multiple appliances plugged into a multi-way mains adapter which, in turn, is connected to a single meter) and we occasionally observe many-to-one relationships (e.g. in the US and Canada many large domestic appliances like washing machines draw a total of 240 volts from two 120 volt 'split-phase' supplies found in a typical house and some datasets use two meters per 240 volt appliance). We frequently observe situations where some appliances are not submetered. To handle the case where a single appliance receives more than one power supply (e.g. split-phase or three-phase power), we allow each ElecMeter object to contain between one and three Sensor objects. Each Sensor models the physical sensors recording data in the field. To handle the case where a single meter is connected to multiple appliances, each ElecMeter can contain any number of Appliance objects. The meter may also specify a `dominant_appliance` property to specify if a single appliance is on more often than other appliances on that meter.




```yaml
# building1.yaml
instance: 1
elec_meters:
- instance: 1
  site_meter: true,
  sensors: [{data_location: 1a.csv}, {data_location: 1b.csv}]
  device_model: Whole-House Meter Model FOO

- instance: 2
  sensors: [{data_location: 2.csv}]
  submeter_of: 1
  appliances: [{type: kettle}]
  device_model: Individual Appliance Monitor Model BAR

- instance: 3
  submeter_of: 1
  appliances: [{type: washing machine}]
  sensors: [{data_location: 3a.csv}, {data_location: 3b.csv}]
  device_model: Individual Appliance Monitor Model BAR

- instance: 4
  sensors: [{data_location: 4.csv}]
  submeter_of: 1
  appliances: [{type: light, multiple: true}]
  device_model: Circuit Monitor Model BAZ

- instance: 5
  sensors: [{data_location: 5.csv}]
  submeter_of: 4
  appliances: [{type: light, room: kitchen}]
  device_model: Circuit Monitor Model BAZ
```

Fig. 2. The illustration on the left shows a cartoon mains wiring diagram for a domestic building. Black lines indicate mains wires. This home has a split-phase mains supply (common in North America, for example). The washing machine draws power across both splits. All other appliances draw power from a single split. The text on the right shows a minimalistic description (using the NILM Metadata schema) of the wiring diagram on the left.

D. ElecMeters and MeterDevices

Each MeterDevice object records properties which apply to a specific model of electricity meter. For example, the `sample_period` in seconds, the `measurements` recorded by the meter (e.g. voltage, reactive power, active energy etc), the meter `manufacturer` and `model`.

ElecMeter objects represent each physical meter installed in a building. Each ElecMeter references exactly one `device_model`. ElecMeter is also the place where any pre-processing carried out on the data can be described (for example, have gaps been filled? Or unrealistic values been removed?)

E. Mains wiring

Each building in a typical dataset will have one meter which records the aggregate, whole-building mains power demand. Downstream of this meter might be meters which measure entire circuits within the building (e.g. the lighting circuit). Finally, there are often meters which measure individual appliances. An example wiring diagram is shown in Figure 2.

As such, the mains wiring connecting meters with each other can be described as a tree. Each ElecMeter can specify either a `submeter_of` property (the numeric ID of the upstream meter) or a `site_meter` property (a boolean flag which is set to `true` if this meter measures the whole-building aggregate). In large, commercial installations, a meter in one building may be downstream of a meter in another building. This case can be handled by specifying the numeric ID of the other building using the `upstream_meter_in_building` property.

F. Appliance and ApplianceType

With each dataset, we specify a set of Appliance objects. Each Appliance object represents an appliance *instance* in the dataset. Each Appliance object in a dataset has a `type` property which refers to an ApplianceType object. ApplianceType objects are not shipped with the dataset; instead they are stored within NILM Metadata and embody the controlled vocabulary of appliance names and all the prior knowledge about appliance types (e.g. the categories each appliance type falls within, probability distributions describing the power demand for the appliance etc).

G. Inheritance for ApplianceTypes

Electrical appliances can be described as a hierarchical tree of objects. For example, a ‘wine cooler’ can be considered a specialisation of a ‘fridge’ and, as such, inherits properties from fridges.

Inheritance is a well-established technique in software engineering for maximising code re-use. NILM Metadata implements a simple but powerful form of inheritance known as prototype-based inheritance. In NILM Metadata, each ApplianceType object has a ‘parent’ from which it inherits properties. These properties can be modified by the child and the child can specify properties not specified by the parent. The inheritance tree can be any depth.

H. Appliance categorisation

When analysing domestic power consumption, we often want to group appliances into certain categories. For example, we might want to ask ‘what is the total energy consumption for all consumer electronics?’.

Domestic appliances are traditionally classified as one of 'wet', 'cold', 'consumer electronics', 'ICT', 'cooking', 'lighting' or 'heating'. An alternative classification is a simple binary classification of 'large appliances' (e.g. dish washer) versus 'small appliances' (e.g. a radio). A more finely-grained classification based on the electrical properties of appliances was proposed by Tsagarakis *et al* [5]. For example, the taxonomy proposed by Tsagarakis *et al.* splits lighting into general incandescent lamps, fluorescent lamps and light-emitting diode (LED) sources. An appliance can have multiple classifications from this taxonomy. Yet another taxonomy for domestic appliances is the Google product taxonomy[11] (used on Google Shopping). This taxonomy is a tree which we represent as list of classifications.

NILM Metadata currently supports all four taxonomies listed above and it would be trivial to add more.

I. Appliances can contain other appliances

Some appliances can be modelled as a container of other objects. For example, a washing machine can be modelled as a drum motor and a water heating element (and a few other components). Appliance (and ApplianceType) objects in NILM Metadata have a `components` property which stores an array of appliance objects. Containment is recursive and can be of any depth.

Of course, *all* appliances can be decomposed into components. Do we model each individual resistor and transistor? No; the end-goal is to model appliances only in sufficient detail to allow an energy disaggregation system to identify the whole appliance given prior knowledge of the components. As such, we only describe individual components if their electrical behaviour is observable from a typical mains electricity meter. It is also important that components be truly separate entities from an electrical perspective.

The container appliance inherits categories from each of its components.

J. Prior knowledge

Prior knowledge can be exploited to improve disaggregation performance. Examples of prior knowledge include: the distribution of on-powers of an appliance; the typical time of use per day or per week; correlations with other appliances (e.g. the computer monitor is often on when the computer is on).

NILM Metadata specifies a 'prior' object which holds several properties, the two most important of which are `distribution_of_data` (the distribution of the data expressed as normalised frequencies per discrete bin (for continuous variables) or per category (for categorical variables)) and `model` (which describes a model fitted to describe the probability density function (for continuous variables) or the probability mass function (for discrete variables)).

K. Learnt models of appliances

End-users of domestic disaggregation software are unlikely to put any effort into training the system. This means that we must use either a supervised learning algorithm with pre-trained models or an unsupervised disaggregation algorithm (which must still have some form of prior appliance model to be able to provide human-readable names for each appliance). As such, we specify a simple 'appliance model'. This has properties such as `model_type` (a controlled vocabulary with terms such as 'HMM' for hidden Markov model), `training_data`, `date_prepared` etc. The model's parameters are stored in a `parameters` object.

[11] https://support.google.com/merchants/answer/160081

IV. Implementation

The syntactic elements of the schema are specified using JSON Schema Draft 4[12]. The code which implements the semantics of NILM Metadata and performs validation is written in Python. We make use of the `jsonschema`[13] package for validation and `PyYAML`[14] for loading YAML files. Metadata instances can be written in JSON or YAML.

V. Conclusions

We have proposed the first draft of a metadata schema for representing objects relevant to energy disaggregation. The schema adapts ideas from DCMI, Project Haystack, ISA_Tab and DataCite; and adds new elements relevant to energy disaggregation (only a few of these elements are required to be instantiated). We also propose a simple but powerful inheritance mechanism to minimise duplication of information and effort. The schema has successfully been used to capture metadata for the UK-DALE (Domestic Appliance-Level Electricity) Dataset [3].

Whilst NILM Metadata is fit for use now, there will inevitably be use-cases that we have neglected hence we warmly welcome contributions from the community! NILM Metadata is open-source to facilitate collaboration and is available at github.com/nilmtk/nilm_metadata

[12] http://json-schema.org/

[13] https://github.com/Julian/jsonschema

[14] http://pyyaml.org/wiki/PyYAML

References

[1] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. NILMTK: An Open Source Toolkit for Non-intrusive Load Monitoring. In *Fifth International Conference on Future Energy Systems (ACM e-Energy)*, Cambridge, UK, 2014. arXiv:1404.3878, doi:10.1145/2602044.2602051.

[2] Jack Kelly and William Knottenbelt. Metadata for Energy Disaggregation. In *The 2nd IEEE International Workshop on Consumer Devices and Systems (CDS 2014)*, Västerås, Sweden, July 2014. arXiv:1403.5946.

[3] Jack Kelly and William Knottenbelt. UK-DALE: A dataset recording UK Domestic Appliance-Level Electricity demand and whole-house demand. *ArXiv e-prints*, April 2014. arXiv:1404.0284.

[4] J. Z. Kolter, M. J. Johnson, J. Zico Kolter, and Matthew J. Johnson. REDD: A Public Data Set for Energy Disaggregation Research. In *ACM Special Interest Group on Knowledge Discovery and Data Mining, workshop on Data Mining Applications in Sustainability*, San Diego, CA, USA, 2011.

[5] G. Tsagarakis, A. J. Collin, and A. E. Kiprakis. A Statistical Survey of the UK Residential Sector Electrical Loads. *International Journal of Emerging Electric Power Systems*, 14:509–523, September 2013. arXiv:1306.0802, doi:10.1515/ijeeps-2013-0078.