Energy Disaggregation

Time Series Data Mining

Introduction

- Non-Intrusive Load Monitoring (NILM) deals with the disaggregation of individual appliances from the aggregate time series energy consumption data collected from a smart power meter.
- NILM aims to provide better monitoring of the appliances (loads) present in buildings, from households to tertiary buildings

Load DisAggregation

- Load disaggregation methods can be classified based on the intrusiveness of the training process and the nature of the classification algorithm (event-based vs. non-event based)
- As illustrated in Figure 1, an event-based algorithm tries to detect On/Off transitions whereas non event-based methods try to detect whether an appliance is On during the sampled duration.
- For this project, a non-event based NILM approach is taken.

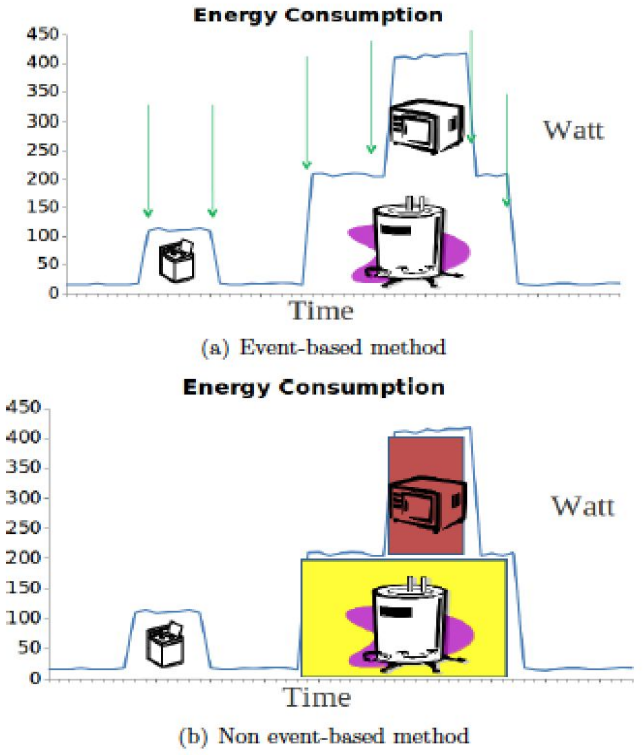
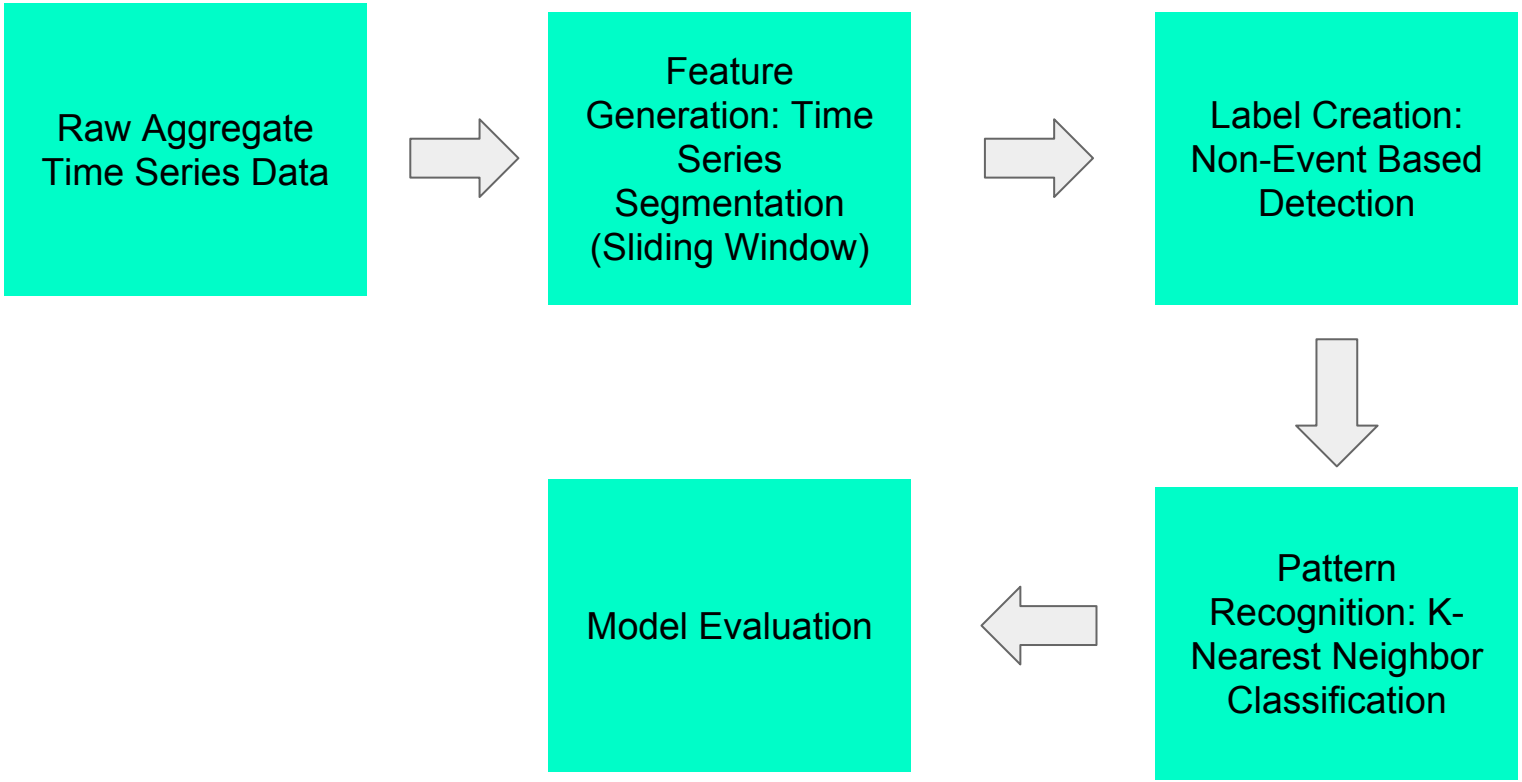


Figure 1: NILM methods

Our WorkFlow Pipeline



Workflow

- A simplified graphical workflow of this approach is illustrated below:

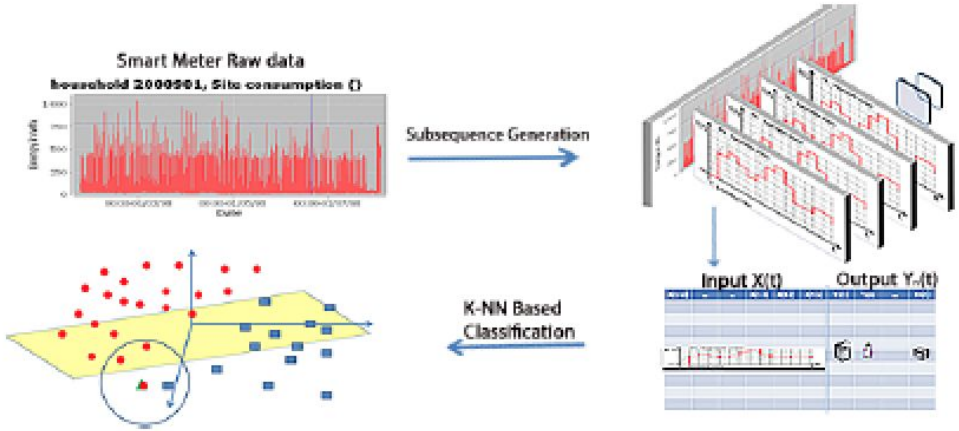


Figure 2. Data-flow Diagram for NILM

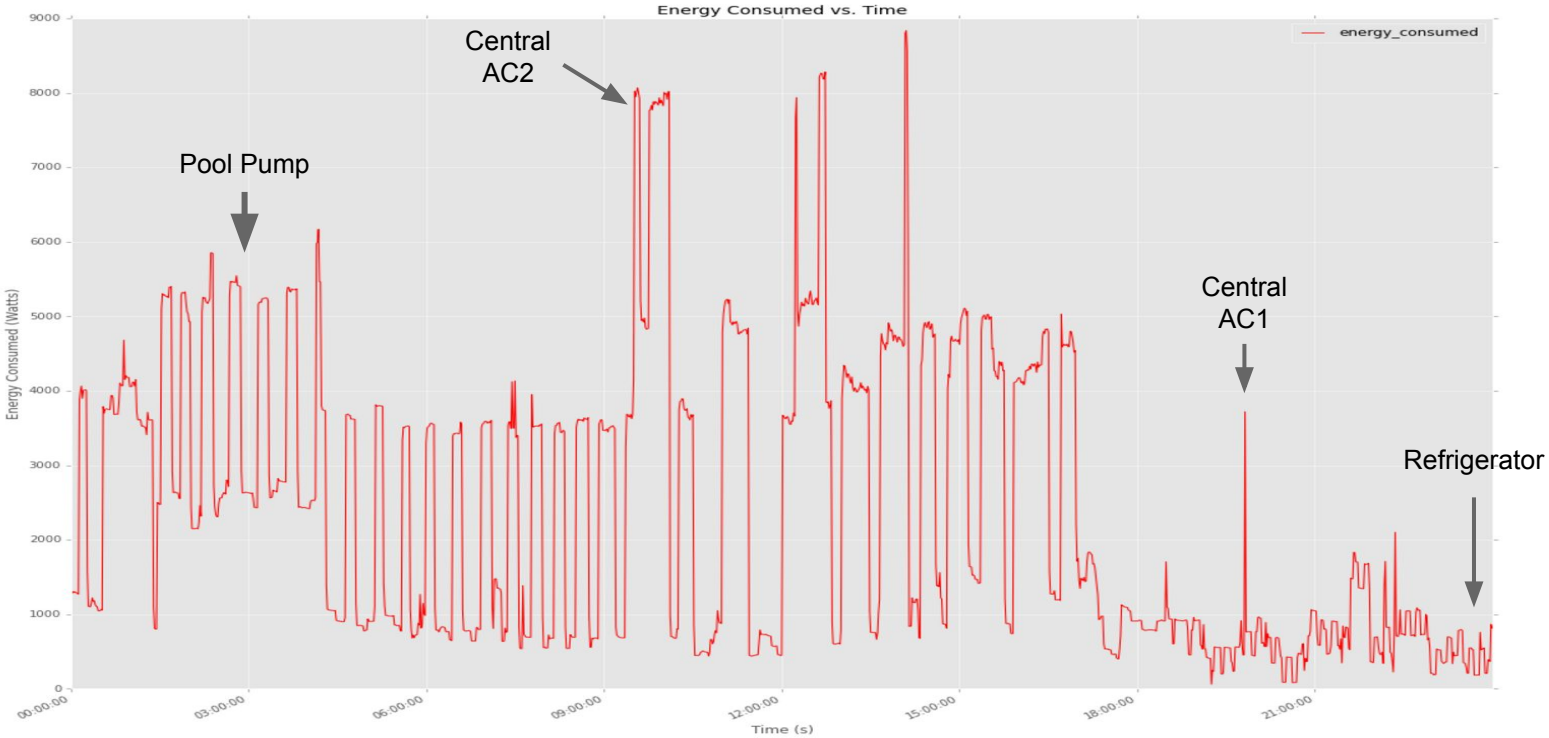
Data Processing

- We first downsampled the aggregate time series energy consumption data from 1s granularity to 1 min granularity to reduce the data size in general.
- A sampling rate of 1 minute allows our method to be directly implemented using current smart meters technologies, without any other energy boxes. It also reduces the privacy concerns as the daily user activity is less detectable. As well, only the high consuming events do leave a footprint (even if the event is very short in time, a high energy consumption application will be detected from the aggregated power source).

Time Series Segmentation

- The aggregate time series data is segmented into sub-sequences using temporal sliding window.
- The size of the sliding window is fixed experimentally to 30 units, the unit being the 1 minute sampling rate.
- Increasing the window size can increase the complexity of the algorithm (not always with noticeable change in the performance) and decreasing the window size can lower the performance of the classifier.

Label Creation: Non Event Based Detection



K Nearest NeighBor (KNN) Classifier

- The mechanism of Knn is straightforward. For any new data instance,the attributes of the new data is compared with all the previously seen instances in the training database, based on a similarity (distance) metric.
- The process of classification is to assign the new instance to the class of the majority of ***k*** neighboring instances.
- In order to classify time segments accurately, the use of the proper similarity metric is imperative.
- Empirically, the best results for this algorithm using DTW Euclidean distance have come when ***k = 1***.

Similarity Metric: Euclidean Distance

- Finding a good similarity measure between time series is a very non-trivial task.
- A naive choice for a similarity measure would be Euclidean distance.
- When applied to time series, the Euclidean distance metric produces pessimistic similarity measures when it encounters distortion in the time axis. As a result, it is not adept at taking into account the temporal relation between the data points.
- In order to counteract distortion in the time axis, we must use Dynamic Time Warping (DTW) as the similarity metric instead.

Similarity Metric: Dynamic Time Warping (DTW)

- Based on a dynamic programming algorithm, DTW finds the optimal non-linear alignment between two time series, in order to minimize a distance measures.
- DTW works the following way. Let $X = (x_1, x_2, \ldots, x_p)$ and $Y = (y_1, y_2, \ldots, y_q)$ be two time series of length p and q. We then construct a time warp path D define by:

$$D_k = \{d_1, d_2, \ldots, d_k\} \text{ with } \max(p, q) \leq k < p + q$$

- Each element of the warp path D_k, of length k, is the euclidean distance between an element from X and an element from Y, i.e. $d_k = (x_k - y_k)^2$.
- We want to find the path with the minimum Euclidean Distance $D^* = \min_d \sqrt{\sum_{k=1}^{K} d_k}$. The optimal path is found via dynamic programming using the following recursive function:

$$\gamma(i, j) = d(x_i, y_j) + \min\left(\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\right)$$

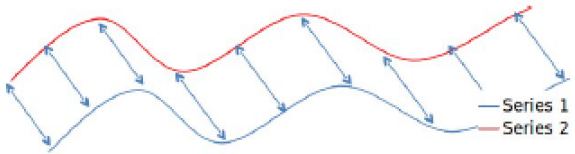


Figure 3. Two Time Series having similar value-based characteristics

DTW Complexity

- Dynamic time warping has a complexity of ***O(pq)***.
- If you are performing dynamic time warping multiple times on long time series data, this can be prohibitively expensive. To speed things, we enforced a locality constraint, which works under the assumption that it is unlikely for x_i and y_j to be matched if ***i*** and ***j*** are far apart.
- The threshold is determined by a window size ***w***. This way, only mappings within this window are considered which speeds up the inner loop of the dynamic time warping algorithm. For our implementation, we set ***w = 10.***
- Given that dynamic time warping is quadratic, this can be very computationally expensive. We can speed up classification using the LB Keogh lower bound. It is defined as $\mathrm{LBKeogh(X,Y)} = \sum_{i=1}^{n}(y_i - U_i)^2 I(y_i > U_i) + (y_i - L_i)^2 I(y_i < L_i)$ where U_i and L_i are the upper and lower bounds for time series ***X*** which are defined as $\mathrm{U}_i = \max(x_{i-r}, \ldots, x_{i+r})$ and $\mathrm{L}_i = \min(x_{i-r}, \ldots, x_{i+r})$ for a reach ***r*** and ***I(·)*** is the indicator function. For our implementation, we set ***r = 5.***
- The LB Keogh lower bound method is linear whereas dynamic time warping is quadratic in complexity which make it very advantageous for searching over large sets of time series.
- Furthermore, since $\mathrm{LBKeogh(X,Y)} \leq DTW(X,Y)$, we can eliminate time series that cannot possibly be more similar that the current most similar time series. Thus, we avoid making many unnecessary dynamic time warping computations.

Model Evaluation

- We evaluated the model using a 10-fold time-series cross-validation analysis with the following metrics: Precision, Recall, and F1.
- Given the time constraint, we computed the performance of the 2-fold and 10-fold cases as our lower bound and upper bound performances of our model.

2-Fold Case

	precision	recall	f1-score	support
click to scroll output; double click to hide			0.92	1712
2	0.74	0.95	0.84	872
3	0.98	0.93	0.96	59
5	0.93	0.72	0.81	1814
avg / total	0.87	0.86	0.86	4457

10-Fold Case

	precision	recall	f1-score	support
1	0.98	1.00	0.99	50
2	0.95	1.00	0.97	552
3	0.98	0.99	0.98	2000
4	1.00	0.99	0.99	1810
5	0.43	0.27	0.33	45
avg / total	0.98	0.98	0.98	4457

Results

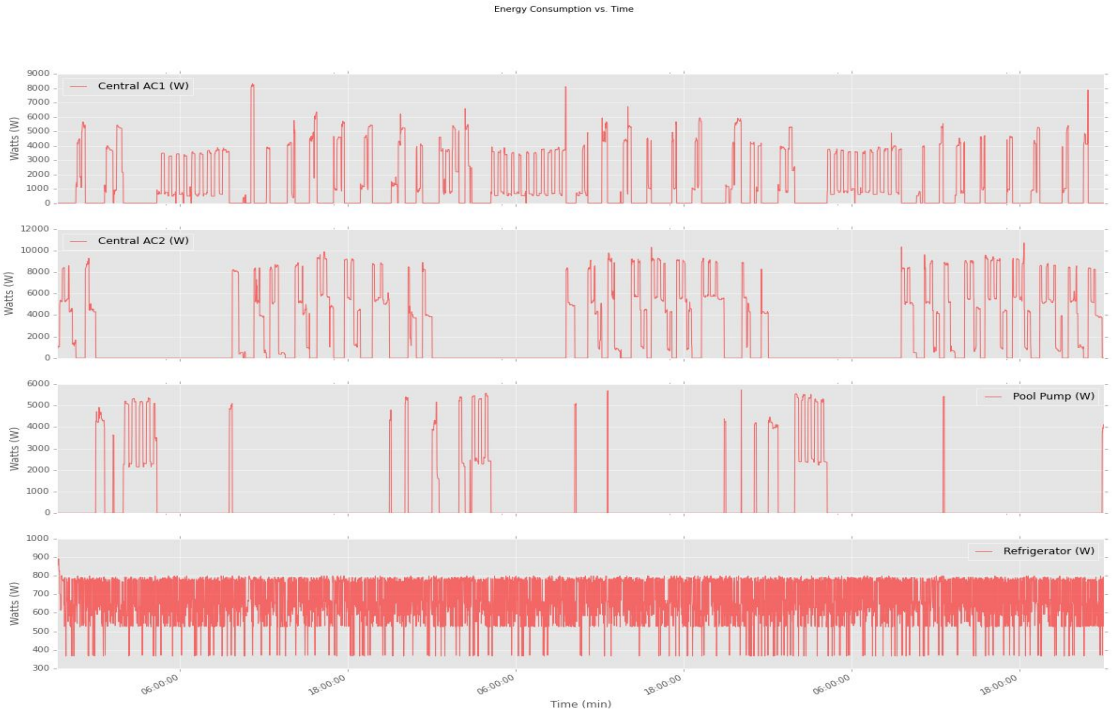


Figure 4. Energy Consumption Time Series for Individual Appliances on the dates of August 28th - 31st.

Further Work

- How do we approach the problem if we weren't given the appliance descriptions for the home?
  - After time series segmentation, we would apply clustering techniques (hierarchical clustering, DBSCAN, OPTICs). With any of these clustering algorithms, the number of clusters is set apriori and similar time series segments are clustered together. Each cluster would then represent a different appliance.
  - After clustering, we can apply our kNN algorithm using DWT to classify our time series segments and then resegment the time series energy consumption data for each individual appliance.
  - Besides this, we can approach the energy disaggregation problem with deep neural networks.