# Unsupervised Learning of Human Behaviours

**Sook-Ling Chua** and **Stephen Marsland** and **Hans W. Guesgen**
School of Engineering and Advanced Technology
Massey University
Palmerston North, New Zealand
{s.l.chua, s.r.marsland, h.w.guesgen}@massey.ac.nz

## Abstract

Behaviour recognition is the process of inferring the behaviour of an individual from a series of observations acquired from sensors such as in a smart home. The majority of existing behaviour recognition systems are based on supervised learning algorithms, which means that training them requires a preprocessed, annotated dataset. Unfortunately, annotating a dataset is a rather tedious process and one that is prone to error. In this paper we suggest a way to identify structure in the data based on text compression and the edit distance between words, without any prior labelling. We demonstrate that by using this method we can automatically identify patterns and segment the data into patterns that correspond to human behaviours. To evaluate the effectiveness of our proposed method, we use a dataset from a smart home and compare the labels produced by our approach with the labels assigned by a human to the activities in the dataset. We find that the results are promising and show significant improvement in the recognition accuracy over Self-Organising Maps (SOMs).

## Introduction

Behaviour recognition has a wide range of applications, including healthcare (e.g., monitoring the daily activities of elderly people and detecting anomalies in a home), security and surveillance (such as detecting unusual events or interaction in airports), industrial applications (e.g., analysing social patterns in organisations), and automation systems (e.g., automatic HVAC control). These have made behaviour recognition a topic of interest, which has led to a variety of solutions based on graphical models (Chua, Marsland, and Guesgen 2009; Hu and Yang 2008). In a smart home, the behaviours are likely to be the standard human behaviours of living, and the observations will depend upon the sensors that the house is equipped with. Since sensor observations from the home in some way represent the behaviours of the human inhabitants, we view the behaviour recognition problem as a task of finding a mapping from a stream of sensor information to a sequence of recognised activities performed by the inhabitant, with the aim of the home being to monitor their behaviour.



There are many ways to learn the mapping between sensor outputs and behaviours. One is to manually label the data, either directly when observing the activities performed by the inhabitant or having the inhabitant keeps a diary of what they do. The first often raises privacy concerns, while the latter is burdensome. Regardless of whether using direct or indirect observations, manual labelling is a time-consuming process; imagine someone has to go through several weeks of data labelling before the recognition system can be put into use. This is rather impractical, especially when implementing a home for the elderly. Considerations such as these have led to the approach of training a recognition system from a limited number of labeled examples, aided by the unlabelled data in a semi-supervised approach to learning. There have been works that attempt to transfer learned knowledge to a new physical domain (Zheng, Hu, and Yang 2009; Rashidi and Cook 2010), or use active learning to engage users to label classes that have the lowest confidence (Stikic, Van Laerhoven, and Schiele 2008); they do, however, rely on some partially labelled data.

An alternative approach is to automatically learn the mapping between sensor information and behaviours in an unsupervised manner. However, there are a number of challenges implicit in recognising human behaviours, such as the fact that the sensors present only a very partial picture of what the inhabitant is doing, are noisy, and that behaviours present in different ways at different times (e.g., the order of events can change, the particular events within a behaviour vary, etc.). One common method is to use standard machine learning approaches to unsupervised learning, such as the k-means algorithm and Self-Organising Map (SOM), with the aim of identifying clusters of similar patterns (Huynh and Schiele 2005; Nguyen, Moore, and McCowan 2007; Hein and Kirste 2008). Some works also progressed toward using available knowledge from the web to mine models of activities (such as 'making tea') associating with household objects (such as 'teapot', 'kettle', etc.) (Perkowitz et al. 2004; Wyatt, Philipose, and Choudhury 2005). The basic idea of these methods is to identify sensors that are related to particular activities. However, relating these clusters of sensor readings to the activities is difficult, as different people have their own ways of performing activities. In order to address this, the resulting output from clustering or the mined model is usually mapped to a supervised algorithm (such as the hidden Markov model or dynamic Bayesian network), which is then used to recognise behaviours.

In this paper we present an unsupervised learning approach based on compression and text analysis that can automatically cluster the unlabelled data and segment the data stream into behaviours without the need of training a supervised algorithm. The main reason why a set of activities form a behaviour is because they are repeated. For an example, making a beverage is a behaviour, since it might well be repeated several times a day, and showering is a behaviour because it is probably repeated daily. However, answering a phone call while having dinner is not a behaviour since it does not occur frequently. Based on this reasoning, it seems clear that we can identify behaviours from a set of sensors that are seen repeatedly in the data, which can be considered as 'redundant' in the representational sense and therefore detectable. It turns out that compression can be used to exploit the redundancy in the sensory stream without any prior human labelling.

To illustrate how compression can be achieved, we represent the sensor stream as a sequence of tokens (e.g. letters from the Roman alphabet), where a token could be the direct representation of the current sensor states being triggered (e.g. bathroom light is turned off, kitchen door is opened, microwave is turned on, etc.). Hence, a behaviour can be identified in the data stream as a repeated set of 'words', albeit with variations, during the inhabitant's daily activities. Any dictionary-based compression algorithm, such as Lempel-Ziv-Welch (LZW) (Welch 1984), could be used to exploit the repetitions by creating a codebook of potential patterns (i.e. 'words'), which is defined by a set of prototype vectors of clusters. However, patterns (e.g. 'AAAI') often do not repeat perfectly each time they are seen, such as the ordering of certain tokens being additionally present (e.g. 'AA*y*AI') or absent (e.g. 'AAI'), that the tokens could be in different order (e.g. 'AAIA'), or that there is minor variation in a token (e.g., 'ABAI'). We hence want to recognise variations in the patterns. Unfortunately, the LZW method does not generalise to variations of the input. To allow for variability, a lossy compression is more suited to our problem. We do this by extending the LZW method to perform lossy compression based on edit distance (Levenshtein 1966).

In this paper, we test our proposed method on a dataset collected from a real smart home and compare it to the Self-Organising Map (SOM) (Kohonen 1990). We also show that the proposed method can be used in a semi-supervised approach by providing labels to training data for a supervised algorithm. We evaluate the effectiveness of our proposed method with a baseline supervised method.

## Our Proposed Approach

Compression has been a topic of interest since the birth of information theory in the work of Shannon (1948), with the aim of reducing the size of data for storage and/or transmission. Compression exploits the repetition in the data by building a dictionary of codewords, and then replacing each incidence of the word with an index into the dictionary, with shorter indices being used for frequent words, and longer indices for less frequently used words. Provided that the indices are shorter than the words in the codebook, compression is achieved. Most compression algorithms require no prior knowledge about the input data stream and can deal with codewords of different lengths without problem.

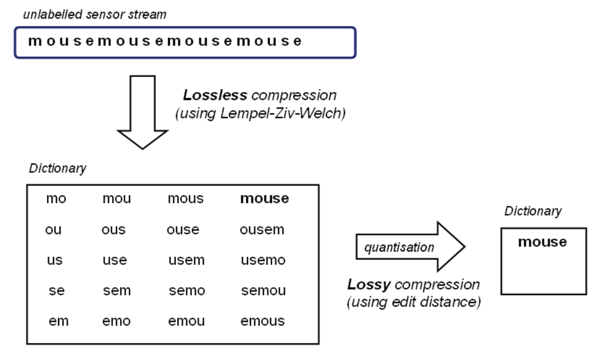


Figure 1: Lossless compression is first performed on the unlabelled sensor stream using the Lempel-Ziv-Welch (LZW) method, which creates a dictionary of phrases. There are indices associated to each substring in the dictionary, but they are omitted for clarity. The dictionary is then quantised using edit distance. The word '*mouse*' after quantisation represents a behaviour.

Figure 1 shows an overview of our approach. Standard lossless compression of the data stream is performed using the LZW algorithm (Welch 1984), which adaptively builds a codebook of commonly seen words. This will include a number of variations of each word, and we then edit this dictionary to produce individual prototype datapoints. Based on this reduced dictionary, lossy matching (i.e., allowing some relatively minor changes between the input and the dictionary words) is used to find the closest matching word in the dictionary.

### Identifying patterns in unlabelled data streams

The only input that we expect to see for our approach is the unannotated data stream. The LZW algorithm is used to parse this and to identify potential sequences that can be added to the dictionary. As an example, the second time the phrase 'mo' is seen in the token sequence 'mousemousemousemouse...', it will take the index of 'mo' found in the dictionary and extend the phrase by concatenating it with the next character from the sequence to form a new phrase ('mou'), which is later added to the dictionary. The search then continues from the token 'u'. The dictionary produced by LZW is typically large, since it contains everything that has been learnt during training, including all the substrings of each dictionary word (see Figure 1).

To identify patterns, we are only interested in the longest frequent words in the dictionary. To illustrate this, assuming for now, we use the word 'mouse' to represent the tea making behaviour, where token '*m*' could be a sensor event on the kitchen door, '*o*' that the tap was running, '*u*' that the kettle was switched on, '*s*' that the fridge was opened and '*e*' that the teapot was in use. Since LZW organises around a dictionary by concatenating a phrase found in the dictionary with the next character from the token sequence, this will result in the dictionary containing many similar phrases such as '*mo*', '*ou*', '*us*', '*mou*', '*ous*', etc. We want to identify the longest common 'words', arguing that they represent patterns; thus we want '*mouse*' to represent one complete tea making behaviour rather than '*mo*' and '*use*' separately.

Lossy compression can help to deal with variability and noise in the data, provided that the component that is lost is not important, or is the noisy part of the data. For this reason, we extend the LZW algorithm to perform lossy compression. The aim of dictionary reduction is to find a single prototype vector for typical data entries. We address this problem using the edit distance (Levenshtein 1966) (also known as Levenshtein edit distance), which measures the similarity between pairs of strings. The edit distance can be efficiently computed by dynamic programming and is commonly used for biological sequence analysis (Sokol, Benson, and Tojeira 2006) and spell checkers (Brill and Moore 2000). It works by computing the minimum number of actions required to transfer one string $p$ into another string $q$, where an action is a *substitution*, *deletion*, or *insertion* of a character into the string. For example, given $p =$'IAAI' and $q =$'AAAI', the edit distance is 1 since we only need to substitute the first letter 'I' in 'IAAI' with 'A'.

The algorithm for computing the edit distance uses a two-dimensional matrix (size $(|p|+1) \times (|q|+1)$, where $|p|$ is the word length of string $p$ and $|q|$ is the word length of string $q$) to keep track of the edit distance values. The algorithm begins by initialising for the first column to have value $[0, 1, 2, \ldots, |p|]$ and likewise for the first row to have the value $[0, 1, 2, \ldots |q|]$. The entry for each remaining cell in the matrix is computed using Equation 1:

$$\min \begin{cases} dist_{i-1,j-1} & + \quad \begin{cases} 0 & \text{if} \quad p[i] = q[j] \\ 1 & \text{otherwise} \end{cases} \\ dist_{i-1,j} & + \quad 1 \\ dist_{i,j-1} & + \quad 1 \end{cases} \tag{1}$$

where $dist_{i,j}$ is the element of the $i^{th}$ row and $j^{th}$ column of the $dist$ matrix.

To perform lossy compression, we have found experimentally that the most effective way to quantise the dictionary is to pick a phrase in the dictionary and find its 'neighbouring' phrases, i.e., those that are edit distance 1 away. The word with the highest frequency count and longest word length is selected as the potential pattern. The algorithm iterates until the pattern does not change. Algorithm 1 shows the steps of using the edit distance for lossy compression.

Once the prototype 'words' for the dictionary have been selected, the next task is to use these prototypes to identify words in the data stream, which is described next.

**Algorithm 1** Lossy Compression using Edit Distance

**Input:** LZW dictionary $D$
**Initialisation:** $m =$ length of $D$
  $P$ = get first phrase from $D$
  **while** not end of $m$ **do**
    **for** $l = 1$ to $m$ **do**
      $\omega \leftarrow$ Using Eq. 1, find phrases where $dist(P, D_l) = 1$
    **end for**
    **if** $\omega \neq 0$ **then**
      select $\tilde{\omega} \subseteq \omega$ where $\max$(freq count + word length)
      delete $\omega$ from $D$
      $P = \tilde{\omega}$
    **else**
      $P$ = get next phrase from $D$
    **end if**
  **end while**
  output quantised dictionary $D'$

## Recognising patterns in the unlabelled data stream

Given a dictionary, we need to parse the data stream to recognise dictionary exemplars and allowable variations on them. To formulate the problem, given the data stream $S = \{d_1, d_2, d_3, \ldots d_k\}$ and the quantised set of words $D = \{w_1, w_2, w_3, \ldots w_n\}$ in the dictionary, we are trying to find a match $w_r$; $r = 1, \ldots, n$ for some subset of $S$, and then make that subset maximal given $w_r$, so that the distance between $w$ and $d$ is minimal.

One of the challenges in segmentation is that the presentation of a behaviour will almost always vary between instances of the same behaviour. For this reason, we use the edit distance to identify the matches between the data stream and the set of words that correspond to behaviours in the quantised dictionary. Segmentation of the data stream can be summarised in a three-step procedure:

**1. Compute the matches between each $w_r$ and the data stream $S$ using edit distance.**
We compute the matches for each $w_r$ in the quantised dictionary and the data stream $S$ using edit distance. The distance values are stored in a two-dimensional matrix ($dist$). Here the value for the first row is initialised as 0, which is shown in Figure 2. This enables an approximate match for some subset of $S$.

**2. Select the maximal 'word' in $S$ with edit distance below some threshold $\epsilon$.**
A threshold $\epsilon$ is chosen to control how much variation is allowed between word samples. Based on experiments, the $\epsilon$ value that we used is half the word length of the word in the dictionary. Referring to the example in Figure 2, the $\epsilon$ value for the dictionary word 'mouse' is 2.5. Looking at the figure, the distance values in the last row for columns 4 and 12 are less than 2.5, which indicates a match.

**3. Perform backward-traversal**
We can distinguish two types of match: perfect matches (i.e. matches with a distance value of 0, such as the last row of column 12 in Figure 2) and matches with errors (i.e. those with a distance value greater than 0, but less than $\epsilon$). When a perfect match is found, we can determine the number of steps to move backwards through the word length. In the example, the word length for '*mouse*' is 5 and thus we can move backward 5 steps. However, if the edit distance is 1,

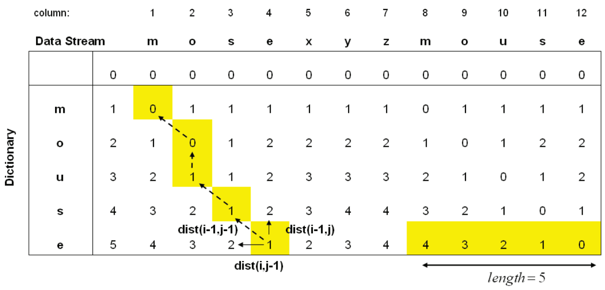


Figure 2: Illustration of how backward traversal is performed on the distance matrix to identify the starting point of the word boundary. When a perfect match is found i.e., when the distance is 0 (column 12), the number of steps to move backward is based on word length. When there is an error (column 4), the algorithm recursively traverses the distance matrix back and upward by finding the minimum distance (shown in dashed arrow). For details, see the text.

i.e. there is an error, then this approach is not sufficient, as it is hard to know if there is a missing or extra letter included (e.g. '*mose*') or a switch of a letter (e.g. '*moues*'). An example of this is shown in column 4 of Figure 2. In this case the starting point of word boundary can be identified by traversing the $dist$ matrix back and upward to find the minimum distance of $min(dist[i, j-1], [i-1, j-1], [i-1, j])$ and thus segment the data stream according to the 'words' in the quantised dictionary.

## Experimental Results

In this section, we describe our experiment setup and the datasets used. In the three experiments reported here, we used the annotation in the training set only to attach a recognisable label to the words in the quantised dictionary, and used the annotation of the test set as a ground truth. *Recognition accuracy* is the ratio of the total number of activities correctly identified by the algorithm over the total number of activities used for testing.

### The Smart Home Data

To demonstrate our system, we used a real smart home dataset from the MIT PlaceLab (Tapia, Intille, and Larson 2004). They collected data using a set of 77 state-change sensors that were installed in an apartment over a period of 16 days. The sensors were attached to objects within the home such as the washing machine, toaster, refrigerator, etc. The dataset was annotated by the subject herself, meaning that there is a ground truth segmentation of the dataset. To simplify the experiment, we examine 5 different behaviours (i.e. toileting/showering, grooming/dressing, preparing meal/snack/beverages, washing/putting away dishes and doing/putting away laundry). Based on these behaviours, there are a total of 310 activity examples and 1805 token observations.

We present three experiments using the MIT PlaceLab data. The objective of the first experiment is to test the proposed unsupervised approach based on compression and edit distance to identify words that are seen frequently in the sensor stream and then segment the data stream into words by finding the closest words in the dictionary using edit distance. We evaluate the performance of our method by comparing it with an unsupervised method based on SOM. The second experiment investigates a semi-supervised approach, i.e. using the output of compression to train a supervised method. This is done by training the hidden Markov model (HMM) based on the 'words' in the quantised dictionary. The third experiment trains a baseline supervised classifier using HMMs, with the aim to evaluate the effectiveness of the unsupervised and semi-supervised methods.

| Test Sets | Proposed Method: No. of Activity Examples | No. of Activities Correctly Identified | Unidentified Activities | Recognition Accuracy | SOM: Recognition Accuracy |
|---|---|---|---|---|---|
| 1st Set | 31 | 25 | 2 | 81% | 56% |
| 2nd Set | 54 | 41 | 7 | 76% | 57% |
| 3rd Set | 20 | 18 | 0 | 90% | 69% |
| 4th Set | 33 | 30 | 0 | 91% | 56% |
| 5th Set | 49 | 41 | 3 | 84% | 57% |
| 6th Set | 34 | 27 | 2 | 79% | 64% |
| 7th Set | 37 | 32 | 2 | 86% | 75% |
| 8th Set | 52 | 41 | 3 | 79% | 47% |
| Average | | | | **83%** | **60%** |

Table 1: A comparison results between our proposed method based on compression and edit distance, and the self-organising map (SOM)

We used a leave-two-out cross validation method for each evaluation in order to calculate the confusion matrix and measure the recognition accuracy. From the total of 16 days of data, we used 14 days for training and the remaining two days for testing. We repeated the process 8 times, and the final recognition accuracy is calculated by averaging the accuracies in each run.

### Experiment 1: Unsupervised Learning

In this experiment, we used the LZW algorithm to build a dictionary of substrings based on the tokens in the training set. We next performed a lossy compression using edit distance to quantise the dictionary. Once we had the quantised dictionary, our next task was to segment the data stream into behaviours. This was performed by parsing the tokens from the test set into the set of quantised words in the dictionary. A visualisation of the output of the system is shown in Figure 3. The tokens that have been segmented into behaviours are then validated against the ground truth annotations on the test set. The results are shown in Table 1.

Some behaviours were too rare and compression simply could not be achieved, which results in some behaviours (such as 'doing/putting away laundry' and 'washing/putting away dishes') not being identified when building the dictionary. These are shown as 'unidentified activities' in Table 1. However, it is still instructive to see if there are consistent reasons for these to occur. Instances of these behaviours vary from the usual norm and the behaviour is often interrupted as another event or noise from the sensors. This results in a high value in the edit distance, and our algorithm is unable to recognise the word.

Since the Self-Organising Map (SOM) is an unsupervised learning method that builds a codebook of prototype vectors, we used the SOM as a baseline to test how effective our proposed method is. The difference is that the SOM determines

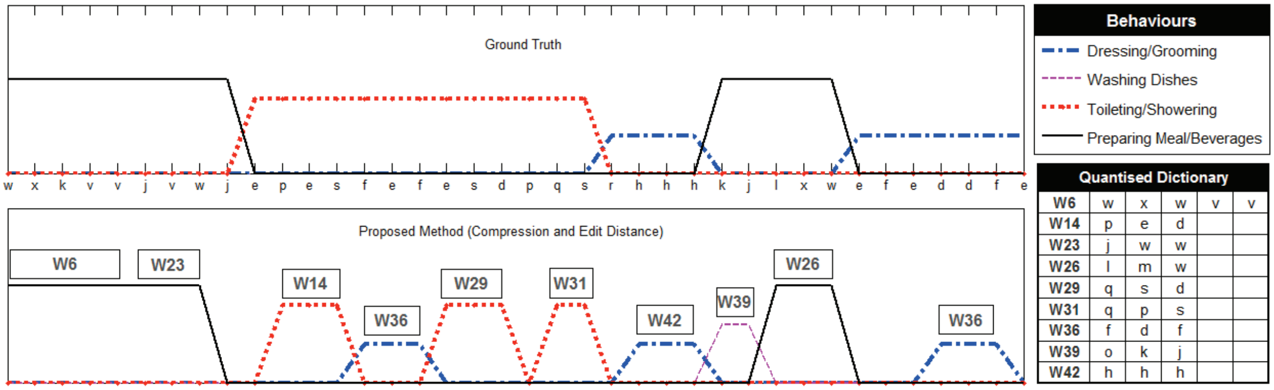


Figure 3: Visualisation of the output of the proposed method with ground truth. The lower case letters on the *x*-axis show the sensor readings, while the *y*-axis shows the potential behaviours (which correspond to the top-right of the figure). The notation 'W6' refers to one of the words in the quantised dictionary (shown on the bottom-right of the figure). The example is based on 5 activity examples in the 3rd test set.

the similarity between the input vector and codebook vectors by minimising the Euclidian distance, while our method minimises the edit distance between the input vector and the 'words' in the dictionary. The data from the sensor stream is presented to the SOM by taking the frequency count of each sensor activation using a window that slides over the data. The size of the window is determined by taking the average number of sensor observations that describe the behaviours. The training of the SOM was in batch mode and the learning rate was set to 0.05. Recognition accuracy in the SOM is calculated by determining the nodes that are the best match according to the target classes in the map after training. The results are presented in Table 1, which shows that our method has a higher recognition accuracy compared to the SOM. Our method did well because it can deal with codewords of different lengths, while in the SOM, the size of the input vector is chosen in advance. This means that the number of tokens that are presented is preset, here to 3.

The system that we are proposing will essentially report which behaviour is identified at each time. However, this might not always be sufficient, depending upon what the aim of the smart home is. For example, suppose that we wish to add other information, such as context (Guesgen and Marsland 2010), or to detect abnormal behaviour. Neither of these would be easy to do using the current method. An alternative is to use the unsupervised learning approach as a way to provide labels to training data for a supervised algorithm in a bootstrap approach to learning. This supervised algorithm can then be used to recognise behaviours from that point onwards. We will now demonstrate how we use the output of our algorithm to train a supervised classifier.

## Experiment 2: Semi-Supervised Learning

This experiment trains a supervised algorithm (i.e. the hidden Markov model (HMM)) based on the 'words' in the quantised dictionary. In our work, the observations are the tokens from the sensors and the hidden states are the events that caused the observations. For example, the token could be that the shower faucet is turned on and the possible state that caused this token is that somebody is showering. We train a set of HMMs, where each HMM represents one behaviour (e.g. we have one HMM to represent the 'toileting' behaviour, another to represent the 'doing laundry' behaviour, etc.), using the standard Expectation-Maximization (EM) algorithm (Rabiner 1989). We use the method described in (Chua, Marsland, and Guesgen 2009) to perform segmentation and behaviour recognition.

The general idea of the method is to slide an initial window of length 10 along the sensor stream and present the 10 observations in the window to the sets of trained HMMs for competition. A winning HMM, $\lambda$, is chosen based on the HMM that maximises the likelihood of the 10 observations $(O_1, O_2, \ldots, O_{10})$ in the window, (i.e. $\arg\max_\lambda P(O_1, O_2, \ldots, O_{10}|\lambda)$). Since it is unlikely that all of the sequences in the window belong to one behaviour, a re-segmentation is performed by using the forward algorithm (Rabiner 1989) to calculate the likelihood of each observation in the window according to the winning HMM. The results are shown in Table 2.

| Test Sets | No. of Activity Examples | Recognition Accuracy | |
|---|---|---|---|
| | | Semi-supervised Learning | Supervised Learning |
| 1st Set | 31 | 84% | 90% |
| 2nd Set | 54 | 87% | 91% |
| 3rd Set | 20 | 95% | 95% |
| 4th Set | 33 | 94% | 91% |
| 5th Set | 49 | 90% | 92% |
| 6th Set | 34 | 88% | 91% |
| 7th Set | 37 | 92% | 95% |
| 8th Set | 52 | 83% | 85% |
| Average | | **89%** | **91%** |

Table 2: A comparison results between semi-supervised and supervised learning methods. The semi-supervised method used the learned 'words' from the quantised dictionary to train a supervised method (i.e. HMM), while the supervised method trains the HMMs directly from the ground truth.

## Experiment 3: Supervised Learning

The aim of this experiment is to build a baseline classifier on the annotated data, where the sensors and activities are known *a priori*. This enables us to obtain a baseline recognition systems. We followed the method described in (Chua, Marsland, and Guesgen 2009). Since all HMMs are trained from the examples of the training datasets, there are no unidentified activities reported. The results of supervised learning are presented in Table 2.

From Tables 1 and 2, the results of the unsupervised method have recognition accuracy of 83%, which is comparable to the supervised method with an accuracy of 91%, considering that our method works on unannotated data streams. This means that the unsupervised method presented in this paper works effectively to identify behaviours from the unlabelled data stream. The ouput of our method can also be used to train a supervised classifier, achieving an accuracy of 89%.

## Conclusions

In this paper, we present a new approach based on compression and edit distance to exploit the redundancy in an unlabelled data stream, which we define to be behaviours. In order to allow variations in the behaviours, we extend the Lempel-Ziv-Welch method to perform lossy compression using edit distance, which is also used to segment the unlabelled data stream into behaviours that we have identified. The results are promising since the method does not need any prior human labelling, which is effective for a real implementation, where the smart home can be built up from nothing when the sensors are placed into a new environment, learning from the unlabelled sensor stream to perform behaviour recognition. Hence, generalisation is not an issue for our system. We also show how the output of the unsupervised method described in this paper can be used to train a supervised classifier in a bootstrap approach to learning. One challenge with unsupervised learning is that there is no guarantee that the identified behaviour will match precisely the labels that would have been assigned by a human. For example, if a person always has a shower before having breakfast, the algorithm may well decide that there is only one behaviour here.

The MIT PlaceLab dataset uses 77 state-change sensors to describe approximately 20 activities in the home although some sensors are not used. We are currently working on detecting informative sensors in the data stream. For that we need to determine the amount of information that we can get from each sensor to identify a behaviour. We also plan to extend our work by exploring other effective method for quantisation, such as the fuzzy clustering method.


## Acknowledgments

The financial support from Massey University is gratefully acknowledged. We also acknowledge the support of the other members of the MUSE group (http://muse.massey.ac.nz). We would like to thank MIT PlaceLab for providing access to their dataset.


## References


Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 286–293.

Chua, S.-L.; Marsland, S.; and Guesgen, H. W. 2009. Behaviour recognition from sensory streams in smart environments. In *Australasian Conference on Artificial Intelligence*, 666–675.

Guesgen, H. W., and Marsland, S. 2010. *Spatio-Temporal Reasoning and Context Awareness*. Berlin, Germany: Springer. 609–634.

Hein, A., and Kirste, T. 2008. Towards recognizing abstract activities: An unsupervised approach. In *Behaviour Monitoring and Interpretation*, 102–114.

Hu, D. H., and Yang, Q. 2008. Cigar: concurrent and interleaving goal and activity recognition. In *AAAI'08: Proceedings of the 23rd Conference on Artificial Intelligence*, 1363–1368.

Huynh, T., and Schiele, B. 2005. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on smart objects and ambient intelligence*, 159–163. New York, USA: ACM.

Kohonen, T. 1990. The self-organising map. *Proceedings of the IEEE* 78(9):1464–1480.

Levenshtein, V. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10:707–710.

Nguyen, A.; Moore, D.; and McCowan, I. 2007. Unsupervised clustering of free-living human activities using ambulatory accelerometry. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 4895–4898.

Perkowitz, M.; Philipose, M.; Fishkin, K.; and Patterson, D. J. 2004. Mining models of human activities from the web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 573–582. New York, NY, USA: ACM.

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Rashidi, P., and Cook, D. J. 2010. Activity recognition based on home to home transfer learning. In *AAAI'10: Proceedings of the 24th conference on Artificial intelligence*, 45–52. AAAI press.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27:379–423, 625–56.

Sokol, D.; Benson, G.; and Tojeira, J. 2006. Tandem repeats over the edit distance. *Bioinformatics* 23(2):e30–e35.

Stikic, M.; Van Laerhoven, K.; and Schiele, B. 2008. Exploring semi-supervised and active learning for activity recognition. In *ISWC'08: Proceedings of the 12th IEEE International Symposium on Wearable Computers*, 81–88. Washington, DC, USA: IEEE Computer Society.

Tapia, E. M.; Intille, S. S.; and Larson, K. 2004. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive*, 158–175.

Welch, T. A. 1984. A technique for high-performance data compression. *Computer* 17(6):8–19.

Wyatt, D.; Philipose, M.; and Choudhury, T. 2005. Unsupervised activity recognition using automatically mined common sense. In *AAAI'05: Proceedings of the 20th National Conference on Artificial Intelligence*, 21–27. AAAI Press.

Zheng, V. W.; Hu, D. H.; and Yang, Q. 2009. Cross-domain activity recognition. In *Ubicomp'09: Proceedings of the 11th International Conference on Ubiquitous Computing*, 61–70.