Unsupervised Learning of Patterns in Data Streams Using Compression and Edit Distance

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

Many unsupervised learning methods for recognising patterns in data streams are based on fixed length data sequences, which makes them unsuitable for applications where the data sequences are of variable length such as in speech recognition, behaviour recognition and text classification. In order to use these methods on variable length data sequences, a pre-processing step is required to manually segment the data and select the appropriate features, which is often not practical in real-world applications. In this paper we suggest an unsupervised learning method that handles variable length data sequences by identifying structure in the data stream using text compression and the edit distance between 'words'. We demonstrate that using this method we can automatically cluster unlabelled data in a data stream and perform segmentation. We evaluate the effectiveness of our proposed method using both fixed length and variable length benchmark datasets, comparing it to the Self-Organising Map in the first case. The results show a promising improvement over baseline recognition systems.

1 Introduction

While there are many methods of performing unsupervised learning (i.e., clustering and classification) described in the literature, very few of them can cope with data sequences of variable length. Essentially, most work by taking input vectors of known dimensionality and identifying prototypes or exemplars that best represent clusters within the data using a distance metric in the vector space of inputs; examples include the Self-Organising Map, Principal Component Analysis and its variants, and hierarchical clustering; see e.g. [Marsland, 2009; Bishop, 2006] for an overview. While useful for a great many applications, these solutions may not be practical in applications where the inputs are not of fixed dimensionality, so that data sequences are variable in length, such as speech recognition, behaviour recognition, text classification, etc. Even where data are of fixed length, they may be embedded into a data stream, which requires segmentation before the individual input vectors can be assembled. In this paper we address the problem of identifying and then learning patterns of variable length in a data stream without any prior knowledge about them. This includes both the segmentation of the data stream into suitable patterns and the identification of patterns when they are seen during testing.

The essential observation of our method is that within a data stream, patterns are repeated observations, albeit with the presence of noise. They can therefore be considered to be redundant in the representational sense. Since the purpose of compression is to remove redundancy in a data stream, by using a dictionary-based compression algorithm we are able to identify potential patterns. We represent data as a set of tokens; here, characters in the Roman alphabet. Depending upon the data, a token could be a representation of an instance of an attribute such as an instance of motion primitive (e.g. the arm is stretched), a pixel in an image, phonetic representation or a sensor reading. A pattern is therefore a repeated set of tokens, which we call a word. We then use the Lempel-Ziv algorithm [Jacob and Abraham, 1978] (although any dictionary-based compression algorithm could be used) to build a codebook of potential patterns, which is then processed to produce the prototype vectors of our clusters.

Lempel-Ziv is a lossless compression algorithm, which means that it does not generalise to variations of the input, which is part of the role of the distance metric in normal unsupervised learning methods. Potential variations that could appear within the data stream are that there could be extra tokens present (e.g. 'h*y*ello') or absent (e.g. 'ello'), that tokens could occur in different order (e.g. 'helol'), or that there is minor variations in a token (e.g., 'hfllo'). To allow for these kinds of variability, a lossy algorithm is more suited to our problem, or at least a lossy matching algorithm that allows variations of a word to be matched in the dictionary. We will present such a method based on the edit distance.

The method that is most similar to our approach is that of Cilibrasi and Vitányi [2005], who propose a similarity distance based on the length of compressed data files, the 'normalised compression distance (NCD)' and then use hierarchical clustering to identify clusters within the data based on the most dominant shared feature, which is computed from the lengths of the compressed data files.

We demonstrate our approach on two datasets. The first is Fisher's iconic iris dataset, which we use to demonstrate our method will work on fixed length data sets. This enables it to be compared to the Self-Organising Map (SOM), although the SOM 'knows' that the data is four-dimensional, which our algorithm does not. We chose the SOM because it is a well-known algorithm for unsupervised learning and, as Learning Vector Quantization (LVQ) [Kohonen, 1990; Somervuo and Kohonen, 1999], it can also be considered to be building a codebook of prototype vectors. The difference is that LVQ finds the similarity between the input vector and codebook vectors by minimising the Euclidian distance in a fixed dimensional space, while our approach attempts to minimise the edit distance in a space of potentially varying dimensionality. The second dataset is a set of readings from sensors installed at a smart home, with the aim being to identify and recognise behaviours from the sequence of sensor events.

2 Unsupervised Learning Using Compression

The idea of compression is to reduce the size of data with the aims of reducing data storage and/or transmission time. It has been a topic of interest for a very long time, but was put on a theoretical footing with the birth of information theory in the work of Shannon [1948]. A common method of performing compression is to exploit regularities in the data by building a dictionary of codewords, and then replacing each instance of a word with an index into the dictionary, with shorter indices being used for frequent words, and longer indices for words that are less frequently used. Provided that the indices are shorter than the words in the codebook, compression is achieved. One of the main attractions of most compression algorithms is that they require no prior knowledge about the input data, and can deal with codewords of different lengths without problem.

Figure 1 shows an overview of our approach. Standard lossless compression of the data stream is performed using the Lempel-Ziv-Welch (LZW) algorithm [Welch, 1984], which produces a dictionary of phrases that are seen most frequently in the data. 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.

2.1 Selecting patterns from an unlabelled data stream

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, we use the

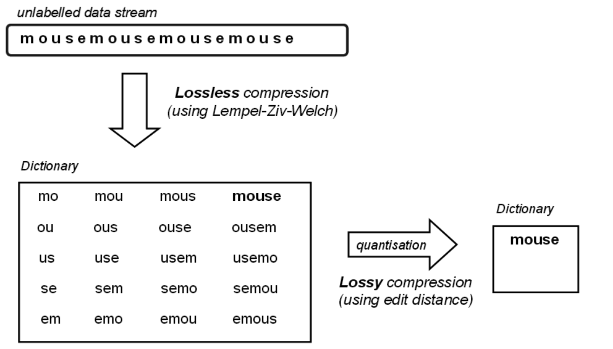


Figure 1: Lossless compression is first performed on the unlabelled data stream using the Lempel-Ziv-Welch (LZW) method, which creates a dictionary of phrases. There are codewords associated to each substring in the dictionary, but they are omitted for clarity. The dictionary is then quantised using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

problem of behaviour recognition in a smart home as a running example. 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 results the dictionary containing many similar phrases such as '*mo*', '*ou*', '*us*', '*mou*', '*ous*', etc. We want to identify the longest possible common 'words', arguing that they represent patterns; thus we want '*mouse*' to represent one complete tea making behaviour, rather than '*mo*' and '*use*' separately. Thus, the next step is to perform a reduction of the dictionary so that it contains only the prototype words.

Our approach to this problem is to consider a modification to the LZW algorithm that enables it to perform lossy compression, so that the dictionary after reduction is biased towards longest words, while still being common in the data stream. Lossy compression is normally applied to data where some loss of fidelity is acceptable in order to achieve a higher compression ratio, and thus is often used in video, sound or images. While there are a variety of ways that the loss can happen, such as the truncation of high frequency signals in sounds files, one common option is to use some form of quantisation, mapping the number of allowable values that the signal can take into some smaller set so that fewer bits are required to describe each one, with the hope that the lost component of the data is either not important, or is even simply the noise in the signal.

The aim of the dictionary reduction is to find a single prototype vector for typical data entries. In fact, we also wish to use the quantisation to ensure that allowable variations on the 'word' in the dictionary are recognised as being equivalent to the dictionary exemplar during use of the algorithm, which is a related problem. We have chosen to approach both of these problems by using the edit distance [Levenshtein, 1966] (also known as Levenshtein edit distance), which produces a measure of 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 *et al.*, 2006], spell checkers [Brill and Moore, 2000], and plagiarism detection [Zini *et al.*, 2006]. It works by computing the minimum number of actions required to transfer one string into the other, where an action is a *substitution*, *deletion*, or *insertion* of a character into the string. Thus, the edit distance $dist(p, q)$ to turn a string p with word length $|p|$ to string q with word length $|q|$ is defined as:

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

where $p[i]$ is the i^{th} character of the string p, $i = 1, \ldots, |p|$; and $q[j]$ is the j^{th} character of the string q, $j = 1, \ldots, |q|$.

For example, given $p =$'feast' and $q =$'fest', the edit distance is 1 since we only need to perform one delete action i.e., deleting the letter 'a' from 'feast'. The edit distance uses a two-dimensional matrix $((|p|+1) \times (|q|+1))$ to keep track of the edit distance values.

Based on this distance, we are now in a position to quantise the dictionary. We do this by picking a phrase from the dictionary at random, and finding its 'neighbouring' phrases, i.e., those that are edit distance 1 away. From this set, the word with the highest frequency count and longest word length is selected as the potential pattern, and the algorithm iterates until the pattern does not change. Algorithm 1 shows the steps of using edit distance for lossy compression. The average computational complexity of our method is $O(Tn^2)$, where T is the number of words, and n is the maximum length of a word.

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 $D'$
```

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. This is described next.

2.2 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 pattern will almost always vary between instances of the same behaviour. Using the same example of the word '*mouse*' to represent the tea making behaviour, two related words, '*moue*' (i.e., missing token, which literally means that the tea is made without the milk) and '*mosue*' (different orders of token activations, where somebody takes the milk out from the fridge before turning on the kettle), could well represent the same behaviour. For this reason, we use the edit distance to identify the matches between the token sequence and the set of words – that correspond to behaviours – in the quantised dictionary. Segmentation of the data stream can be summarised as a three-step procedure:

1. Compute the edit distance between each w_r and the data stream S.

We compute the edit distance for each w_r in the quantised dictionary and the data stream S. The distance values are stored in a two-dimensional matrix ($dist$).

2. Select the maximal 'word' in S with edit distance below some threshold ϵ.

A threshold ϵ is chosen to control how much variation is allowed between word samples. Based on experiments, the ϵ value that we used is half the word length of the word in dictionary. Referring to the example in Figure 2, the ϵ 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 identified two ways to determine a match when a low edit distance score was found. The first works when there is a perfect match, i.e., with a distance value of 0 (such as in the last row of column 12 in Figure 2). The second is for use when there are errors in the match (i.e., those with distance value greater than 0, but less than ϵ). When a perfect match is found, we can determine the number of steps to move backwards through the word length. In this example, the word length for '*mouse*' is 5 and thus we can move backward 5 steps. However, when there are errors in the match, 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 when the edit distance is 1. In this case the starting point of the word boundary can be identified by traversing the $dist$ matrix back and upward to find the minimum distance of $min(dist[i, j-1], [i-1, j-1], [i-1, j])$. By traversing back and upward i.e., following the shaded paths in Figure 2, we can identify the starting point of word boundary

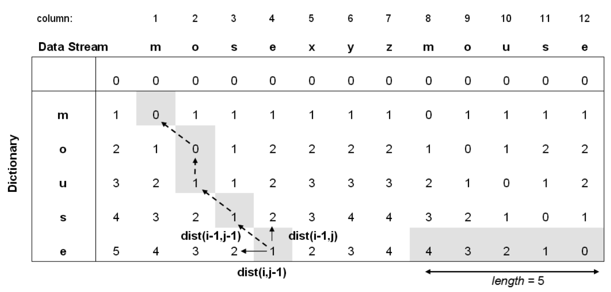


Figure 2: Illustration of how backward traversal is performed on the distance matrix to identify the starting point of the word boundary. When a perfect match is found i.e. when the distance is 0 (column 12), the number of steps to move backward is based on word length. When there is an error (column 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.

and thus segment the data stream according to the ‘words’ in the quantised dictionary.

3 Experimental Results

In this section, we describe our experiment setup and the datasets used. In fact, the datasets that we have used do have labels, and thus are suitable for supervised learning. We have only used this annotation as a ground truth for the test set. We report experiments on two datasets. The first has fixed length inputs and is well-known as a test dataset: the iris set. This enables us to compare the results with other unsupervised learning algorithms, here the SOM. The second dataset is the sensor readings for a smart home, and the behaviours that should be identified are of variable length. The main statistic that we are interested in is *recognition accuracy*, which is the ratio of the total number of activities correctly identified by the algorithm divided by the total number of activities used for testing.

3.1 The Iris Dataset

Fisher’s iris data [Fisher, 1936] is widely used for cluster analysis and is available through the University of California, Irvine (UCI) machine learning database [Frank and Asuncion, 2010]. The data has four attributes, which are measured on fifty iris specimens from each of three species: Setosa, Vericolor and Virginica. We partitioned the dataset into 5 equal partitions and used leave-one-out cross validation, training on 4 partitions and using the last one for testing. The recognition accuracy is calculated by averaging the accuracies in each run. Since the iris data are numeric, to use it with our compression approach, we transformed the data into a series of tokens using quantisation by treating each dimension separately and breaking it into ten equally spaced steps, with tokens ‘a’ to ‘j’. Thus, attribute instances whose values were in the range $[0, 0.1)$ were assigned token ‘a’, those in $[0.1, 0.2)$ were assigned token ‘b’ and so on.

In this experiment, we also used a SOM as a baseline to understand how effective our proposed method is. The training of the SOM was in batch mode and was trained for 100 epochs. Recognition accuracy for the SOM was calculated by determining the nodes that were the best match according to the target classes in the map. The results are presented in Table 1. Our algorithm did very well given that it did not know that the input data were four dimensional, which the SOM obviously did. However, we were interested in how much of the error was caused by the quantisation that was required to turn the data into tokens. We therefore re-transformed the tokens back into numbers, by replacing each token by the median of the values range for each tokens (e.g. token ‘a’ is transformed into 0.05, token ‘b’ into 0.15, etc.). The results in the final column of Table 1 show the results when the SOM is trained on these quantised data, which leads to a significant reduction in the accuracy of the SOM, and makes the accuracy of our method comparable to the baseline. The choice of how to quantise the data into tokens is an important one, providing a trade-off between compression rate and accuracy. One can use a smaller range of values for the transformation to achieve a higher compression rate or use a larger range of values to achieve a higher recognition accuracy.

3.2 The Smart Home Data

One aim of our method is that it works well where the inputs are of variable length, such as behaviour recognition and speech recognition, where conventional unsupervised learning algorithms do not do well. 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 the number of sensory inputs that define a behaviour is not fixed and the presentation of the behaviours almost always vary at different times (e.g., the order of events can change, the particular events within a behaviour vary, etc.).

To demonstrate our algorithm on this type of problem, we used a real smart home dataset from the MIT PlaceLab [Tapia *et al.*, 2004]. They collected data using a set of 77 state-change sensors that were installed in an apartment and collected data while a person lived in the apartment for a period of 16 days. The sensors were attached to objects within the home, such as the washing machine, toaster, and refrigerator. The dataset was annotated by the subject, meaning that there is a ground truth segmentation of the dataset, although there are notable omissions and inaccuracies in the annotations. To simplify the experiment, we examine 5 different behaviours (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 sensor observations.

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. Once the prototype ‘words’ for the dictionary have been identified, we parse the tokens from the test set to recognise dictionary examplars.

In this experiment we compare the unsupervised approach using compression with a supervised method based on Hidden Markov Models (HMM), where the sensors and activ-

Test Sets	Recognition Accuracy		
	Self-Organising Map (on original IRIS data)	**Our Proposed Method** (compression and edit distance)	**Self-Organising Map** (on transformed IRIS data)
1st Set	90%	90%	90%
2nd Set	97%	90%	90%
3rd Set	93%	87%	83%
4th Set	93%	93%	87%
5th Set	97%	93%	90%
Average	**94%**	**91%**	**88%**

Table 1: Iris data: A comparison results between the Self-Organising Map (SOM) method and our proposed method

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

Table 2: Results for the Smart Home dataset using unsupervised learning based on compression and edit distance on different training/testing splits.

Test Sets	No. of Activity Examples	No. of Activity Correctly Identified	Recognition Accuracy
1st Set	31	28	90%
2nd Set	54	49	91%
3rd Set	20	19	95%
4th Set	33	30	91%
5th Set	49	45	92%
6th Set	34	31	91%
7th Set	37	35	95%
8th Set	52	44	85%
Average			**91%**

Table 3: Results for the Smart Home dataset using supervised learning based on the hidden Markov models (HMM) on different training/testing splits

ities are known *a priori*. For the HMM, 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 is turned on and a possible state that arises from this token is that somebody is showering. We trained a set of HMMs, where each HMM represents one behaviour (e.g. we have one HMM to represent the 'toileting' behaviour, another HMM to represent 'doing laundry' behaviour, etc.). We trained the HMMs using the standard Expectation-Maximization (EM) algorithm [Rabiner, 1989] and used the method described in [Chua *et al.*, 2009] to perform segmentation and behaviour recognition. The central idea of the method is to slide an initial window of length 10 across the sensor stream and presenting the 10 observations in the window to the sets of trained HMMs for competition. A winning HMM, λ, 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. $\arg\max_\lambda P(O_1, O_2, \ldots, O_{10}|\lambda)$). 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.

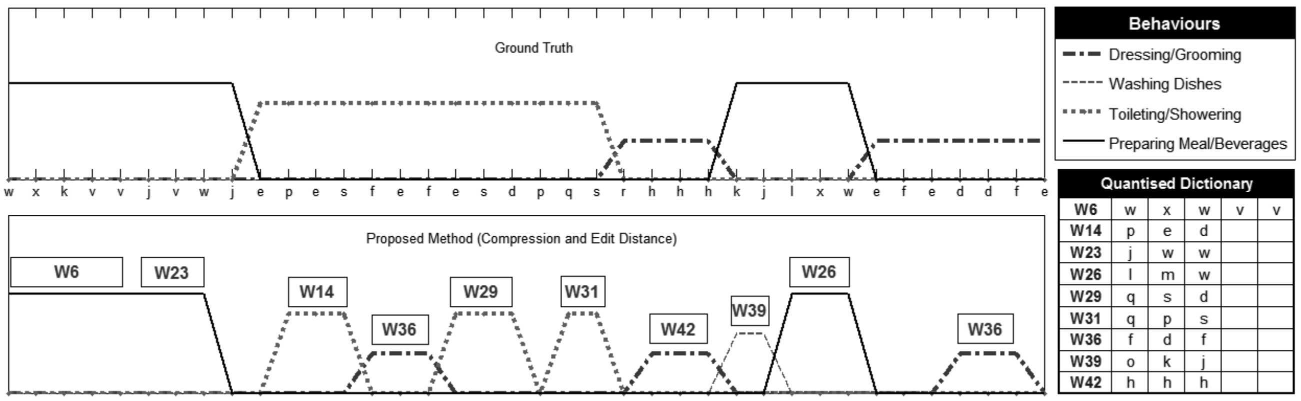
For each method, from the total of 16 days of data, we used 14 days for training and the remaining two days for testing. We used a leave-two-out cross validation method for each evaluation in order to calculate the confusion matrix and measure the recognition accuracy. We repeated the process 8 times, with the final recognition accuracy being calculated by averaging the accuracies in each run. The results for the compression-based unsupervised method are shown in Table 2 and Figure 3, while those for the supervised HMMs are shown in Table 3.

For the unsupervised learning approach, 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. This is shown as 'unidentified activities' in Table 2. However, it is still instructive to see if there are consistent reasons for these to occur. Instances in these behaviours vary from the usual norm and the behaviours are often interrupted by another event or noise from other sensors. This results in high values in the edit distance, meaning that our algorithm is unable to recognise the word. We expect that if there had been more examples of those words, they would have been identified successfully.

The results from Tables 2 and 3 show that the proposed unsupervised method, with a recognition accuracy of 83%, is comparable to the supervised method with 91% recognition accuracy. This means that the unsupervised method presented in this paper works effectively to identify behaviours from the unlabelled data stream. Our results are promising since firstly, our method does not need any prior annotations, which is likely to be the aim of any real smart home, which would want to learn from scratch without any built-in prior knowledge when the sensors are installed in the home, since there is such variation between houses, sensors, and the activities of people. Secondly, our method works effectively on data with variable length data sequences without requiring any pre-processing step for segmentation.

4 Conclusions

We have presented a new method based on compression and edit distance to recognise patterns from the unannotated data. We have demonstrated the applicability of using compression to exploit regularities in the data stream, which we define to be patterns, without any prior knowledge or human labelling. In order to allow for variations in the patterns, we perform lossy compression based on edit distance to quantise the dictionary produced by the Lempel-Ziv-Welch (LZW) method



W6	w	x	w	v	v
W14	p	e	d		
W23	j	w	w		
W26	l	m	w		
W29	q	s	d		
W31	q	p	s		
W36	f	d	f		
W39	o	k	j		
W42	h	h	h		

Figure 3: Visualisation of the output of the ground truth (top) and proposed method (bottom). The lower case letters on the x-axis show the sensor readings, while the y-axis shows the potential behaviours (which corresponds to the top-right of the figure). The notation 'W6' refers to 'Word #6' in the quantised dictionary (shown on the bottom-right of the figure). The example is based on 5 activity examples on the 3rd test set.

by using the edit distance, which is also then used to segment the data stream into patterns that we have identified. This is performed by identifying matches between the tokens in the data stream and the quantised words in the dictionary. We have demonstrated that our method works effectively both on data with fixed length data sequences (the Iris data) and variable length data sequences (the MIT PlaceLab data).

One option that we have not explored in this paper, but will in the future, is the use of the compression approach to identify the behaviours that should be learnt. Each behaviour can then be learnt by separate HMMs, so that we get the ability to use supervised learning methods on what is actually unlabelled data. We also plan to extend our work by exploring other methods of quantisation, such as fuzzy clustering.

5 Acknowledgments

We would like to thank MIT PlaceLab and UCI for providing access to their dataset. 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).

References

[Bishop, 2006] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

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

[Chua *et al.*, 2009] Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Behaviour recognition from sensory streams in smart environments. In *Australasian Conference on Artificial Intelligence*, AI '09, pages 666–675. Springer-Verlag, 2009.

[Cilibrasi and Vitányi, 2005] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[Fisher, 1936] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[Frank and Asuncion, 2010] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[Jacob and Abraham, 1978] Ziv Jacob and Lempel Abraham. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

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

[Levenshtein, 1966] Vladimir Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[Marsland, 2009] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. CRC Press, New Jersey, USA, 2009.

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

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

[Sokol *et al.*, 2006] Dina Sokol, Gary Benson, and Justin Tojeira. Tandem repeats over the edit distance. *Bioinformatics*, 23(2):e30–e35, 2006.

[Somervuo and Kohonen, 1999] Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10:151–159, October 1999.

[Tapia *et al.*, 2004] Emmanuel M. Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive*, pages 158–175, 2004.

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

[Zini *et al.*, 2006] Manuel Zini, Marco Fabbri, Massimo Moneglia, and Alessandro Panunzi. Plagiarism detection through multilevel text comparison. In *AXMEDIS '06: Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, pages 181–185, 2006.