Language Model Co-occurrence Linking for Interleaved Activity Discovery

Eoin Rogers, John D. Kelleher, and Robert J. Ross

Applied Intelligence Research Centre, Technological University Dublin
eoin.rogers@tudublin.ie
robert.ross@tudublin.ie
john.d.kelleher@tudublin.ie

Abstract. As ubiquitous computer and sensor systems become abundant, the potential for automatic identification and tracking of human behaviours becomes all the more evident. Annotating complex human behaviour datasets to achieve ground truth for supervised training can however be extremely labour-intensive, and error prone. One possible solution to this problem is activity discovery: the identification of activities in an unlabelled dataset by means of an unsupervised algorithm. This paper presents a novel approach to activity discovery that utilises deep learning based language production models to construct a hierarchical, tree-like structure over a sequential vector of sensor events. Our approach differs from previous work in that it explicitly aims to deal with interleaving (switching back and forth between between activities) in a principled manner, by utilising the long-term memory capabilities of a recurrent neural network cell. We present our approach and test it on a realistic dataset to evaluate its performance. Our results show the viability of the approach and that it shows promise for further investigation. We believe this is a useful direction to consider in accounting for the continually changing nature of behaviours.

1 Introduction

Activity discovery (AD) refers to the automated and unsupervised extraction of *activities* (recurrent patterns of behaviour) from a given dataset of sequential sensor events [6]. These sensor events could come from any one of a wide range of sensors installed in a real-world environment (such as a house or office), or could be events logged by a server or other computer monitoring a virtual environment. Activity discovery is part of the wider field of *activity recognition*, which refers to the use of machine learning in the identification and classification of activities. AD has a wide range of applications, from the automatic labelling of datasets for more general activity recognition [16], to more complex end-to-end systems which combine elements of activity discovery and recognition [9]. The wider problem of identifying usable sub-sequences in larger sequences that have semantic meaning also has wider applicability for areas like anomaly and crime detection [27] [10] and network intrusion detection [15].

One issue that poses a challenge for many existing activity discovery systems is *interleaving*, where two or more activities are being carried out by a single actor at the same time. Usually, this appears on the dataset as if the actor is switching back and forth between activities, much like how a compute processor switches back and forth between processes when the operating system performs a context switch. Equivalently, multiple individuals might be carrying out tasks in parallel, which would register on the sensors in a manner that closely resembles interleaving.

With the goal of improving the performance of such systems on interleaved data, we present here a novel system for activity discovery which is explicitly designed to account for the interleaved data of modern datasets. Like most activity discovery systems, we base our work on the intuition that activities appear in datasets as *sub-sequences* that repeat multiple times throughout the dataset. Unlike most other AD systems, however, we do not require that our activities consist of contiguous substrings of the dataset, but rather can be interrupted with sensor events which may be parts of other activities, or may not belong to other activities. This allows activities to be interleaved, and allows the model to explicitly model and *disentangle* the interleaving.

On top of this, we build activities that are hierarchical, tree-like structures. Thus, discovered activities can contain other activities as a subset. This models the hierarchical nature of real-world activities: a large, complex activity such as *washing* could consist of smaller sub-activities such as *using the shower* or *brushing teeth*. This allows our system to take sequential input data and convert it into a rich, complex structure, much like how a language parser can convert a sequential string of symbols into a parse tree, thus exposing non-obvious structure contained within the original sequence.

The remainder of this paper is structured as follows: in **Sect. 2** we review prior and related work in activity discovery and sequential pattern discovery more generally, some of which has been an important influence on the work presented here. In **Sect. 3** we briefly introduce the formal notation that we use in this paper, before introducing our activity discovery model in **Sect. 4**. We present our evaluation study design and the results of that study in Sections **Sect. 5** and **Sect. 6** respectively. Before concluding we provide a brief discussion in **Sect. 7**.

2 Prior work

A number of techniques have been proposed to tackle the activity discovery problem. A good general introduction to the field is Cook et al. [6], which introduces an activity discovery system that applies a beam search algorithm using an operator called *ExtendSequence* to discover activities in an unlabelled dataset. Like a number of other systems in the field, this algorithm utilises the *minimum description length* (MDL) principle [22][23], which proposes evaluating machine learning models by measuring the degree to which they *compress* their input

dataset. This is an important principle, and is one that we shall return to later in this paper.

Activity discovery can also be carried out by relatively simple systems that utilise topic models [16]. Here, the *latent Dirichlet allocation* (LDA) topic model [3] is used to model the relationship between sensor events and latent variables which are presumed to represent activities. The model is shown to have good performance, even on a complex dataset. More recently, other models based on statistical models have been proposed: for example, Fang et al. [11] proposes activity discovery by means of a hierarchical mixture model. [24] propose using activity discovery to build a model of normal behaviour patterns of a person in order to detect anomalous behaviours that may be of interest to medical professionals.

Related fields also provide an important source of ideas. Grammar induction is a concept from computational linguistics which refers to the derivation of grammar productions for a language given only a dataset. Some forms of grammar induction require labelled input, distinguishing positive and negative examples, but others require only positive examples. In the general case, grammar induction is not a tractable problem, regardless of whether the dataset is labelled or not [13], but tractable approximations have been demonstrated which solve the problem to a degree [8]. While the problems are related, grammar induction is by no means equivalent to activity discovery (and is in fact in many ways harder), but it does involve the induction of structure from a one-dimensional input vector. *Adios* [26] induces a grammar by loading a dataset into memory as a graph, with words represented as vertexes and sentences represented as directed edges between these. This representation allows for the identification of *equivalence classes* between words and phrases which share the same input and output edges, which can then be added to the graph as nonterminals. A variant of the Adios approach, which supplements the basic grammar induction algorithm with logical predicates to allow for more accurate induction in a limited linguistic domain is presented by [12].

Remaining on the theme of grammar induction, the *eGrids* grammar induction algorithm [21] bears a resemblance to the beam search-based system mentioned previously from [6]. This approach also uses an MDL-based objective function to guide the search. More recently, an interesting deep learning-based grammar induction model using convolutional networks to determine *syntactic distance* (the degree to which two neighbouring words or symbols belong to the same POS phrase) has been proposed by [25]; this approach does have some similarities to the approach we take later in this paper.

More generally, it should be noted that the activity discovery problem as presented by us can also be understood as a non-local variant of the tree structure induction algorithm *Sequitur* [20], which groups input symbols together even if they do not appear contiguously.

However, none of the approaches discussed above were explicitly developed with the goal of dealing with interleaving. Some solve the interleaving problem better than others, but the system we propose in this paper has the advantage of

explicitly disentangling interleaved activities from each other, identifying where one activity switches to another and when it switches back. Thus, it solves a problem not tackled by most existing activity discovery systems.

3 Activity Discovery Process Notation

In order to have a clean model description, it is necessary for us to briefly introduce the terminology that we will use later. Formally, an activity discovery system can be modelled as a 5-tuple (Σ, D, A, f, g), where:

- Σ is a set of *event types*;
- D is an ordered sequence of events, $D = \langle d_1, d_2, \ldots, d_L \rangle$ of length L, such that each $d_i \in D$ is drawn from the set Σ. We call this the *dataset*;
- A is a set of *activity types*;
- f is a mapping $f : D \to X^*$, which takes a sequence of events D as input, and returns a set of (possibly non-contiguous) sub-sequences of D as output; and
- g is a mapping $g : X \to A$, where $X \subset D^*$, which takes a sub-sequence produced by f as input, and returns an activity type $a \in A$ as output.

This definition can be made clearer with a concrete example. Supposing we have a dataset $D = \langle d_1, d_2, \ldots, d_N \rangle$. Each $d_i \in \Sigma$ is a sensor event drawn from Σ, our full set of sensor events. In an environment where sensors have been set up in a home, for instance, Σ could consist of events like *open front door*, *turn oven on*, *flush toilet* and similar domestic events. An *activity*, then, is simply a sub-sequence of D consisting of events that appear to the activity discovery system to be semantically related. For instance, we would expect that events like *turn oven on*, *open kitchen cupboard*, *open refrigerator* might occur in an activity together, since they tend to occur together temporally. It should be noted that D is not a set of sequences as might be the case in a supervised learning setting; D is a single large dataset from which we extract activities.

Multiple similar activities can then be clustered or lifted into one *type*. The activity discovery system might notice that an activity similar to the one mentioned in the previous paragraph seems to occur nightly, and may cluster them all into a single *making dinner* activity type. The concrete sub-sequences of D are referred to as the *instances* of the *making dinner* activity.

Note that we don't generally expect an activity discovery system to operate with human-like semantic knowledge or expectations in the basic case. Thus, it would not be expected to be able to name the new activity type as *making dinner*, only to identify that the instances involved can be sensibly clustered together. A commercial activity discovery system might well be supplemented with real-world knowledge, with the intention of biasing towards the sort of activities we would expect to find in the environment in which it operates. For instance, knowledge that events relating to a fridge or oven indicates activities relating to food preparation such as *making dinner* are taking place. In many ways this would stray over into being a form of activity recognition as well as

discovery. For this reason, we stick to a pure form of activity discovery without any real-world knowledge. We do still expect it to be able to discover *making dinner* as an activity, just not to be able to give it a label (*making dinner*) that would be semantically meaningful to a human observer.

4 Model

We now proceed to outline our approach to the activity discovery problem. At a bird's-eye view, one can conceptualise our model as being composed of the following four elements:

- A *neural language model* to analyse the input dataset, and build probability distributions over future events given past events;
- A *linking component* to link events into activity instances using the probability distributions;
- A *clustering component* to cluster activity instances into activity types in a principled way;
- A *hierarchy construction component* to remove discovered activities and replace them with new synthetic events, thus allowing the above steps to be repeated, and a hierarchy to be iteratively built up.

In the following we take each of these components and describe them in detail.

4.1 Association Estimation

All approaches to activity discovery depend upon the assumption that the activities present in a dataset will be composed of events, which are assumed to be associated with each other in some predictable way. For example, if we want an activity like *cooking dinner* to be found in a dataset, we would need to see that a set of events such as *turn on oven*, *open refrigerator*, *open cupboard* and the like occur close to one another in the dataset on a daily basis. Intuitively, our approach is based on the insight that if observing the *turn on oven* event allows me to predict that the *open refrigerator* event will soon follow, I can use this fact to infer the existence of the *cooking dinner* activity.

To detect these associations, and to estimate how strong they are, we turn to *language modelling*, a concept originating from the natural language processing (NLP) community. Given a sentence of words $W = \langle w_1, w_2, \ldots w_Q \rangle$, a language model estimates the probability of a word w_i given the previous n words, which can be written $p(w_i|w_{i-1}, w_{i-2}, w_{i-n})$. Equivalently, we can view this as assigning a probability to a given sentence or sentence fragment, since the probability of the entire sentence W must be:

$$p(W) = p(w_1)p(w_2|w_1)\ldots p(w_N|w_{N-1}, w_{N-2}, \ldots w_{N-n}) \tag{1}$$

A typical language model attempts to predict the *next* word of the sequence, but we need to take into account the possible presence of interleaving. In other

words, *open refrigerator* may occur a number of events *after* the *turn on oven* event. Keep in mind, therefore, that we build a probability distribution not over the next event, but rather over the next m events.

Traditionally the training of language models entails the collection of statistics from a dataset. Such systems have been used for many decades within the NLP community.

Rather than relying on statistical language models, we apply a so-called *neural language model* (NLM) [2][19] to the association estimation task. In recent years, neural language modelling systems have achieved parity with, and subsequently overtaken, their more classical (statistical) counterparts. These models build on general trends in *deep learning* [18] by applying large artificial neural network architectures and taking advantage of recent advances in hardware and training algorithms. Yoshua Bengio [2] proposed the first NLM system in 2003, with more recent systems adopting sophisticated additions such as recurrent models, attention [4], internal memory [28] and levels of representation other than individual words [19].

Our modelling approach builds on the LSTM network architecture [14]. This architecture is a form of recurrent neural network that is particularly good at encoding long range dependencies into a decision process. An LSTM unit consists of a single memory unit called a *cell*. Computational units called *gates* determine the contents of the cell and the behaviour of the unit by controlling the movement of data in and out of the cell. Typically, three gates are used: an *input gate*, which controls the extent to which the current input to the unit influences the cell, an *output gate*, which controls the extent to which the current cell contents influence the unit's output, and a *forget gate*, which controls the extent to which the current value of the cell will be retained for the next iteration of the LSTM unit. A given LSTM can be trained using labelled data and the backpropogation algorithm in the normal way. Moreover, rather than using a single LSTM network, our approach makes use of a collection of LSTM estimators. Specifically, we train m LSTM networks, one to predict the *next event*, one to predict the event immediately after it, one to predict the event after that one and so on. Each network net_j thus predicts the distribution:

$$p_j(d_{i+j}|d_i, d_{i-1}, \ldots, d_{i-n}) \tag{2}$$

For each $j \in \langle 1, 2, \ldots m \rangle$, where we refer to each j as a *lookahead offset*. This builds distributions over the next m events, motivated above for allowing us to detect interleaving. Since some events might be very common in the dataset, we actually use the difference between the probability distribution in Eq. 2 and the distribution already computed for the previous position of the sliding window $i-1$. In a slight abuse of notation, therefore, we actually use p_j in Eq. 2 to refer to *relative probability*, defined as the following:

$$p_j(d_{i+j}|d_i, d_{i-1}, \ldots, d_{i-n}) - p_j(d_{(i-1)+j}|d_{i-1}, d_{i-2}, \ldots, d_{(i-1)-n)}) \tag{3}$$

The equation above gives us the relative probability distribution for the $i+j$th slot in the dataset, where $1 \leq j \leq m$. This process is illustrated in Fig. 1. In Fig. 1(a), we see a short dataset of events A, B, C, D, E, F, G, H. A sliding window of length 2 ($n = 2$) is placed over events C and D of the dataset, and a lookahead window of length 3 ($m = 3$) is placed over events E to G in the dataset. The sliding window is fed as input into each of the m networks (there would be three in this case, one for each element of the lookahead window). Thus each network is trained to output a probability distribution over the event it expects to observe at a particular offset into the lookahead window.

To illustrate the example further, the network net_1 might predict a 20% higher relative probability that event E would be observed at an offset of 1 compared to when the sliding window ends at event C instead of D. net_2 might predict an 80% higher relative probability that event F would be observed at an offset of 2, and net_3 might predict a 40% higher relative probability of event G being observed at an offset of 3. Note that each network outputs a probability distribution over the entire set of events Σ: we only show the probabilities of the event types that actually occur in Fig. 1(b). Thus, the output from this stage can be viewed as P, an $n \times m \times |\Sigma|$ matrix, where each P_{ijk} is the relative probability that the $i+j$th event in the dataset is predicted to be the kth event type in the set of events Σ.

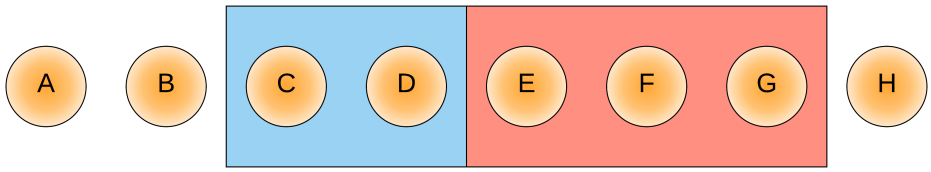
4.2 Linking events into activities

Given a sliding window across the dataset, the association estimator will provide likelihoods of particular events occurring at given offsets into the look-ahead window – these are encoded as the association matrix P. Based on that information we next establish links between strongly associated events.

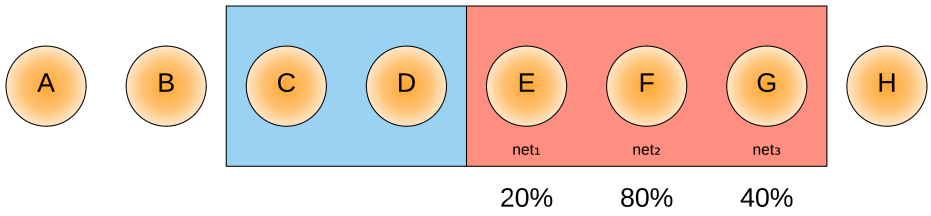
There are a number of ways in which the linking process could be achieved. For the experiments presented in this paper, we begin by iterating over each $i \in \langle 1, 2, \ldots, n \rangle$ and each $j \in \langle 1, 2, \ldots, m \rangle$. For each (i, j) pair, we can view P_{ij} as the relative probability distribution that each $d \in \Sigma$ will be the jth event *after* the ith event. In the case of Fig. 1(b), we see that net_2 has assigned a high relative probability to the event type F occurring at an offset of two after D. By contrast, net_1 and net_3 have assigned much lower probabilities to their corresponding values. Thus, we would expect that this means that events D and F are part of the activity, which would justify connecting them via a *link*, as shown in Fig. 1(c).

This is essentially a greedy linking strategy as we are guaranteed that the strongest links only for a given symbol are created. This has advantages over alternatives such as a thresholding-based model in that no threshold parameter needs to be derived from the dataset.

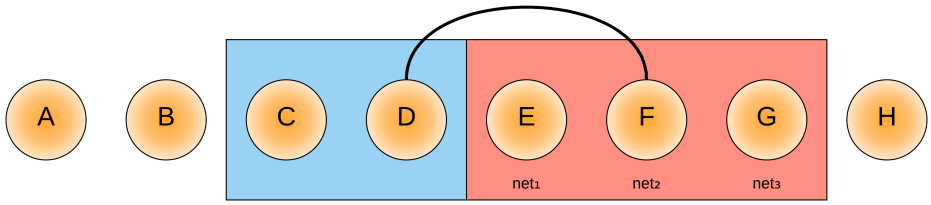
This is almost correct, but naive implementations of the above algorithm for linking result in most links being made with the event of an offset of one into the lookahead window, i.e. the first events in the lookahead window. This is obviously not ideal from the perspective of identifying activities that are interleaved. If we train a system like that described above with a window and lookahead length



(a) Activities A and B are contained within a sliding window; activities C, D and E are in the lookahead window



(b) net_1 assigns a 20% relative probability to offset (j) 1 being equal to event type E, net_2 assigns an 80% relative probability of offset 2 being equal to event type F, and net_3 a 40% relative probability of offset 3 being equal to event type G



(c) A single link connecting activities D and F

(d) The new dataset after the link has been abstracted into one event

Fig. 1. Using probability distributions to construct links between events

of 20 (so $n = m = 20$), and evaluate the accuracy of the resulting twenty networks, we observe something interesting. The first network, net_1, is about 98.8% accurate. The second network's accuracy has dropped down to about 97.9%, which is a trend that continues throughout the lookahead window. By net_{20}, the accuracy is down to about 94%. This makes intuitive sense, since predicting the *next* event will generally be easier than predicting the event that will occur three events from now. Analogously, predicting the next word in a sentence is easier than predicting the word that will come a number of words after. We thus feel that we are justified in modifying the algorithm to explicitly take distance into account, so longer offset networks get a small boost in their probabilities to offset the inherent higher difficulties in what is expected in them.

Thus, we multiply each relative probability by a *correcting factor* that is equal to 1 for an offset of 1, some value larger than one for offsets greater than 1, and which increases linearly. We call the parameter which controls the degree with which the factor increases x. Since this value is no longer a valid probability we refer to it as a *score*. Thus, for formula for computing the score for offset j and window position i is as follows, which is a modification of the probability in Eq. 2:

$$score(d_i, d_{i+j}) = p_j(d_{i+j}|d_i, d_{i-1}, \ldots, d_{i-n}) \times (1 + \frac{j}{x}) \tag{4}$$

This link will *only* be built if D and F do not link more strongly with some other event in the dataset. For example, if D was predicted with a relative probability of 90% when the sliding window ended at event B, we would have built the link between B and D instead.

4.3 Clustering activity instances into activity types

We now need to match all the links we have found with links of the same type. We call this step *clustering*. In the case of the example presented in Fig. 1, we would need to cluster all other links between event types D and F (or equivalently between F and D) together. Note that this differs from clustering in the usual sense of the word, since we are trying to find exact matches between link types, not semantic similarity as would be done in a clustering algorithm such as k-means clustering.

At this point, we also apply a threshold factor, which we call y, to remove spurious links. Link types that do not appear at least y times in the entire dataset are removed.

4.4 Building a hierarchy

The final step in a single iteration of the model is to build a new dataset, where each discovered link of two activities is removed, and replaced with a *new event*, with each activity type giving rise to a new event type. The outcome of this process applied to the small dataset we have used as an example in this section is shown as Fig. 1(d). From here, we can train a new set of m LSTM networks, and

repeat the process again. At the end, a tree-like structure will result, showing a hierarchy of (possibly overlapping) activities contained within each other. This is inspired from the way the Sequitur algorithm [20] constructs tree structures from sub-sequences that occur multiple times in a sequence, generalised to allow for non-contiguous sub-sequences. Ideally, the process would be run until a sufficiently high level of abstraction (where the tree-like structures correspond to activities) has been reached. In practice, the process can be stopped early if only a partial result is needed. The new event could be placed into the position formerly occupied by either event D or event F. The choice shouldn't affect the evaluation metrics we are using, so the choice of which position is somewhat arbitrary. In our case, we place the new event in event D's position.

5 Experiment Design

For our experiments, we utilised the Kyoto 3 dataset gathered by the CASAS project [7]. This dataset consists of readings from a range of sensors installed in a small apartment. The dataset was gathered by asking a number of participants to perform *activities of daily living* (ADLs) in a natural manner in the apartment. Most of the sensor readings are either binary (they have a simple on/off state), or can only enter one of a handful of states. This means they can be easily converted to the sequence of events format our system expects by creating event types of the form $SensorName\_SensorState$. For example, one of the sensors are refered to as $M17$ in the dataset, and can take the state ON, so $M17\_ON$ becomes an event type in the dataset. For the few sensor types that did have continuous values, we used the *Jenks natural breaks algorithm* [17] to discretise the data. Our system does not take temporal distance into account, so it cannot, for instance, see large gaps between events. This makes the systems task substantially harder, but it allows us to put our system through much more challenging testing than most AD practitioners settle for.

Evaluating activity discovery systems can be a challenge for a number of reasons. Human annotators may not come to an agreement with each other over the start and end points of activities, which makes working from a gold-standard ground truth quite difficult. For example, when does the activity of $Making\_Dinner$ start? When a person enters the kitchen? When they turn on the oven? In many cases, a ground truth may not even be available (although that isn't an issue for the Kyoto dataset). The output from an AD system may be on a different level of abstraction from the ground truth: for example the system may discover an activity that could be called something like *chop_vegetables*, but the ground truth instead has an activity called *make_dinner*, which *chop_vegetables* would be a constituent of. A good overview of evaluation for activity discovery can be found in [5].

Since we do have access to a ground truth in this experiment, it makes sense to use it, although we must keep the above issues in mind. Because of the abstraction issue mentioned before, we argue that both raw accuracy and F-measures are inappropriate for evaluating this system. Instead, we compare each

new event type from the topmost (i.e. most abstract layer) of the hierarchy using the *precision* metric, i.e. the true positives divided by the sum of the true and false positives. Each event type is then matched with the ground truth activity with which it achieves the highest associated precision.

Our system is implemented using Python/TensorFlow [1] running on an Nvidia graphics card. We trained the hierarchy for 5 layers: each layer took roughly an hour to train and cluster. Our LSTMs were two layers deep, with a width of 150 LSTM cells per layer. We used a sliding window length $n = 20$, a lookahead window length $m = 10$, a score factor $x = 400$ and an event type threshold $y = 3$.

We have already mentioned minimum description length (MDL) in Sect. 2 of this paper. This is the second metric that we propose for our system. MDL draws a parallel between machine learning on the one hand, and data compression on the other. If person A wishes to transmit a dataset to person B while minimising the bandwidth, they could do so by encoding the dataset directly according to some optimal encoding scheme and send it. By contrast, person A could also train a machine learning model and send this, since person B could use it to re-create the dataset. Of course, no model will be perfect, so we must also send *data that would be required to correct the model's mistakes*. Since the mistakes will hopefully be small, these corrective measures should not consume many bits of bandwidth. In its purest form, MDL proposes computing a *score* for a machine learning system, which is the length of the machine learning model (in bits), plus the length of all the corrective values (also in bits). The smaller this value, the better the model is taken to be.

However, a pure MDL approach won't work in our case, since the output from the system is actually a compressed form of the input (so the original input can't be recovered from it). For this reason, [5] suggest simply using the *compression ratio* as a metric. In other words, since our model is compressing the input directly, evaluate how well it carries out this compression.

6 Results

We now present the results of the experiment described in Sect. 5.

Our system discovers in excess of 500 event types, reproducing the full result of this evaluation here would not be possible. Nonetheless, we present an extract from these results as Table 1. The results are reasonably good: a little over half of the events discovered correlate to a precision of at least 50%, meaning that the results show a correlation (but not a perfect overlap) between the ground truth and the discovered output. Considering both the differing levels of abstraction, and the large amount of interleaving present in the Kyoto 3 dataset, we feel that this is an acceptable initial result. The large number of events suggests that in the future, more needs to be done to combine the discovered event types; see Sect. 7 for more details.

As mentioned earlier, another important evaluation metric is the compression rate. Our system compresses the original input dataset to about 68% of its

Event name	Precision
new_event_10	0.2857142857142857
new_event_11	0.6666666666666666
new_event_12	0.6666666666666666
new_event_13	0.3333333333333333
new_event_14	0.42857142857142855
new_event_160	0.75
new_event_161	0.6666666666666666
new_event_162	0.3333333333333333
new_event_163	0.6666666666666666
new_event_231	0.6666666666666666
new_event_232	0.6666666666666666
new_event_233	0.6666666666666666
new_event_301	0.7142857142857143
new_event_302	0.6666666666666666
new_event_303	0.3333333333333333
new_event_304	0.25
new_event_305	0.3333333333333333

Table 1. Some event types discovered and associated precision values

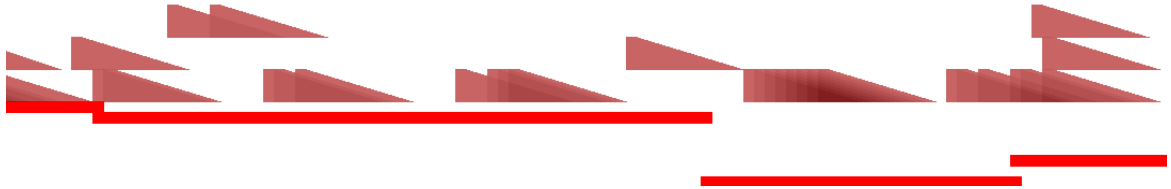
original input size. This is a good result, albeit one that we hope to improve upon in the future.

Finally, we have produced a visualisation of our system's output to allow us to see that a hierarchy is being built up, and visualising it as a tree-like structure. Because of the length of the input dataset, this is again far too big to show in this paper. However, we present some extracts from it as Fig. 2. The bars at the bottom of the image are the original ground truth, each row represents a certain activity type, and the bar will be present along the row when the activity is active, but not otherwise. The triangles above it represent the discovered events, with the wide bottoms at the bottom of the triangles compressing into the narrower tops. In some places, the hierarchy is quite deep, as visible in Fig. 2(a). Visually, it is noticeable that clusters tend to form around activities. Our previous evaluation methods had no way of picking up on this phenomenon, which we will discuss further in Sect. 7. The events sometimes cross activity boundaries, but these incursions are small. This could be evidence that the human annotator of this dataset and the system are seeing similar activities, but can't agree when they start or end as discussed above.

7 Discussion

Originally, we hypothesised building an NLM that would not predict a distribution over the next event, but rather over the *next m events*. In other words, for each $d \in \Sigma$, it would output $p_{i:i+m}(d|d_{i-1}, d_{i-2}, \ldots, d_{i-n})$, the probability that d would be one of the next m events observed.

(a) A deep hierarchy of events discovered by the system.



(b) Although events sometimes cross activity boundaries, the incursions are always small, indicating they could still be part of the activity

Fig. 2. A visualisation of the system output, where the red bars at the bottom correspond to ground truth activities, and the triangles correspond to discovered events that can be understood as *compressing* the original dataset.

The approach shows promise, as shown by our experiment results. Deep learning for activity discovery is in its infancy, so we do not claim that we can outperform other AD techniques, but this is a starting point.

There are a number of ways that we intend to build on this work in the future. As noted in the previous section, visualising the output shows clusters of new events forming around activities. This seems to suggest that the method finds activities, but these aren't being seen or enlarged by subsequent layers of the hierarchy. This could be an artifact of the dataset, or could be evidence that we need to change how we change the LSTM probability distributions into links. It could also turn out that we need to find some way to cluster the discovered event types. This could be done based on temporal proximity, for instance. Clustering rare event types into more common ones might also make learning higher layers easier. We *did* attempt to use a more complex clustering method in the past, but this turned out to perform poorly at discriminating between events from different activities. We aim to investigate and correct this issue as future work.

8 Conclusion and future work

This paper presented a novel approach to activity discovery (AD), and tested it on a real-world dataset. We have shown that the approach is viable, and appears to show promise for the task of activity discovery. The system has been evaluated on a number of distinct evaluation metrics, which show it to be robust and suggest that the measured performance is not merely a result of picking a favourable evaluation metric. Finally, we have outlined a number of changes we plan to make in the future to further improve the system, and make it capable of discovering activities even in very complex activities.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. Journal of machine learning research **3**(Feb), 1137–1155 (2003)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
4. Chan, W., Jaitly, N., Le, Q.V., Vinyals, O., Shazeer, N.M.: Speech recognition with attention-based recurrent neural networks (Oct 24 2017), uS Patent 9,799,327
5. Cook, D.J., Krishnan, N.C.: Activity learning: discovering, recognizing, and predicting human behavior from sensor data. John Wiley & Sons (2015)

6. Cook, D.J., Krishnan, N.C., Rashidi, P.: Activity discovery and activity recognition: A new partnership. IEEE transactions on cybernetics **43**(3), 820–828 (2013)
7. Cook, D.J., Schmitter-Edgecombe, M.: Assessing the quality of activities in a smart environment. Methods of information in medicine **48**(05), 480–485 (2009)
8. Cramer, B.: Limitations of current grammar induction algorithms. In: Proceedings of the 45th annual meeting of the ACL: student research workshop. pp. 43–48. Association for Computational Linguistics (2007)
9. Domingo, C., See, S., Legaspi, R.: Unsupervised habitual activity detection in accelerometer data. In: Mechatronics and Machine Vision in Practice 3, pp. 253–272. Springer (2018)
10. Emonet, R., Varadarajan, J., Odobez, J.M.: Multi-camera open space human activity discovery for anomaly detection. In: 2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). pp. 218–223. IEEE (2011)
11. Fang, L., Ye, J., Dobson, S.: Discovery and recognition of emerging human activities using a hierarchical mixture of directional statistical models. IEEE Transactions on Knowledge and Data Engineering (2019)
12. Gaspers, J., Cimiano, P., Griffiths, S.S., Wrede, B.: An unsupervised algorithm for the induction of constructions. In: 2011 IEEE International Conference on Development and Learning (ICDL). vol. 2, pp. 1–6. IEEE (2011)
13. Gold, E.: Language identification in the limit. i nformation andcontrol, 10: 447-474, 1967.[11] s. Jain. An infinite class of functions identifiable using minimal programs in all Kolmogorov numberings. I nternational J ournalofFoundationsofComputer Science **6**(1), 89–94 (1967)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
15. Huang, J., Kalbarczyk, Z., Nicol, D.M.: Knowledge discovery from big data for intrusion detection using lda. In: 2014 IEEE International Congress on Big Data. pp. 760–761. IEEE (2014)
16. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: UbiComp. vol. 8, pp. 10–19 (2008)
17. Jenks, G.F.: The data model concept in statistical mapping. International yearbook of cartography **7**, 186–190 (1967)
18. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
19. Merity, S., Keskar, N.S., Socher, R.: An analysis of neural language modeling at multiple scales. arXiv preprint arXiv:1803.08240 (2018)
20. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research **7**, 67–82 (1997)
21. Petasis, G., Paliouras, G., Karkaletsis, V., Halatsis, C., Spyropoulos, C.D.: e-grids: Computationally efficient gramatical inference from positive examples. Grammars **7**, 69–110 (2004)
22. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465–471 (1978)
23. Rissanen, J.: Stochastic complexity in statistical inquiry. World Scientific (1989)
24. Saives, J., Pianon, C., Faraut, G.: Activity discovery and detection of behavioral deviations of an inhabitant from binary sensors. IEEE Transactions on Automation Science and Engineering **12**(4), 1211–1224 (2015)
25. Shen, Y., Lin, Z., Huang, C.W., Courville, A.: Neural language modeling by jointly learning syntax and lexicon. arXiv preprint arXiv:1711.02013 (2017)
26. Solan, Z., Horn, D., Ruppin, E., Edelman, S.: Unsupervised learning of natural languages. Proceedings of the National Academy of Sciences **102**(33), 11629–11634 (2005)

27. Xu, D., Wu, X., Song, D., Li, N., Chen, Y.L.: Hierarchical activity discovery within spatio-temporal context for video anomaly detection. In: 2013 IEEE International Conference on Image Processing. pp. 3597–3601. IEEE (2013)
28. Yogatama, D., Miao, Y., Melis, G., Ling, W., Kuncoro, A., Dyer, C., Blunsom, P.: Memory architectures in recurrent neural network language models (2018)