Learning event representation: As sparse as possible, but not sparser

Tuan Do and James Pustejovsky
Department of Computer Science
Brandeis University
Waltham, MA 02453 USA
{tuandn, jamesp}@brandeis.edu

Abstract

Selecting an optimal event representation is essential for event classification in real world context. In this paper, we investigate the application of qualitative spatial reasoning (QSR) framework for classification of human-object interaction in three dimensional space, in comparison with the use of quantitative feature extraction approach for the same purpose. In specific, we modify QSRLib (Gatsoulis et al. 2016), a library that allows computation of Qualitative Spatial Relations and Calculi, and employ it for feature extraction, before inputting features into our neural network models. Using an experimental setup involving motion captures of human-object interaction as three dimensional inputs, we observe that the use of qualitative spatial features significantly improves the performance of our machine learning algorithm against our baseline, while quantitative features of similar kinds fail to deliver similar improvement. We also observe that sequential representation of QSR features yields the best classification performance. A side product of our learning method is a simple approach to qualitative representation of 3-d activities as composition of 2-d actions that can be visualized and learned using 2-dimensional QSR.

Introduction

The study of events has long been a focus of many disciplines, including philosophy, cognitive psychology, linguistics, computer science, and AI. Tulving (Tulving 1983) postulated a separate cognitive process for event recognition called *episodic memory*. In natural language, events have been studied from many different approaches, from formal logic and AI (Allen 1984), and frames (Fillmore 1975), to computational linguistics (Pustejovsky et al. 2011). In computer science, event representation has been learnt in a classification manner (Shahroudy et al. 2016) or represented as composition of primitive actions (Veeraraghavan, Papanikolopoulos, and Schrater 2007).

In this paper, our focus lies on a smaller and more restricted set of human activities involving human and object interaction. We use a fine-grained event capture and annotation framework (Do and Pustejovsky 2017) as our basic setup for event classification. This framework, which takes into account what is called *extra-verbal factors*(Pustejovsky 1995) in treating the classification of events (considering the difference between actions such as "jump on" and "jump over"). This allows us to investigate effects of QSR on our event classification framework.

In particular, we used an event capture and annotation tool called ECAT (Do, Krishnaswamy, and Pustejovsky 2016), which employs Microsoft Kinect® to capture sessions of performers interacting with two types of objects, a cube (which can be slid on a flat surface) and a cylinder (which can be rolled). Objects are tracked using markers fixed to their sides facing the camera. They are then projected into three dimensional space using Depth of Field (DoF). Performers are tracked using the Kinect®API, which provides three dimensional inputs of their joint points (e.g., wrist, palm, shoulder). Sessions are first sliced, and each slice is annotated with a textual description with our event language. The event descriptions are in turn parsed into tuples of semantic roles.

We then transformed the raw data into a spectrum of feature types. The first type is quantitative features reflecting positions of humans and objects projected on 3-dimensional and 2-dimensional feature spaces. The second type is built on top of the first type, producing qualitative spatial (QS) representation for each image frame. The third type is QS representation for each whole event duration, summarized on top of second-type features. In short, we compared our ML methods on 7 different kinds of features.

Depending on the form of extracted features (sequence vs non-sequence), two different machine learning methods are implemented. For frame-sequence learning, we used Long-short term memory (LSTM); for whole-event learning, we used Multilayer perceptron model (MLP). Their similar neural network structures allow us to compare between these two methods fairly. In addition, we put a layer of constraints by Conditional Random Field (CRF) before outputs.

The main contributions of our study are twofolds. Firstly, we proposed a framework for event recording, annotation and classification, achieving high accuracy using qualitative spatial reasoning for feature extraction. Secondly, by analyzing different levels of feature representation, from dense and continuous to sequential and discretized features to summary event-level features, we found out the most economical and effective way for classification of human-object interaction.

Related works

Monolithic human activities such as *running*, *sitting*, *eating*, *playing sport* have been investigated in a fair amount of research work, such as of (Shahroudy et al. 2016) and (Dubba et al. 2015). These human activities have significantly different motion signatures and durations. Sometimes algorithms learned to distinguish these events are actually learned on the distinction of background or color histogram. Recently, some studies have started to propose dataset with more complex activities, especially involving human-object interation, such as cooking activities (Rohrbach et al. 2012), taking medicine (Koppula and Saxena 2016), or human-human interaction, such as hand shaking (Ryoo and Aggarwal 2010). More recently, (Li and Fritz 2016) investigates the possibility of predicting partial activities using a hierarchy label space. These studies have gradually focused on a more fine-grained treatment of event classification.

To facilitate event classification, events are required to be presented in learnable format. There come the difficulties of event representation, namely the difficulty in defining their temporal and spatial expansions and also the difficulty in selecting an observational perspective. For example, it is generally hard to demarcate an event duration from other events building up to it or are its consequences. *He kicks the ball* might or might not involve the person running up to the ball, or include the ball flying to the goal. Similarly, to pick out a set of things to include in an event representation is not trivial, even for event described in text. For example, in *He fried an egg*, should we include the frying pan into the event representation or not? The point of view (POV) is also normally under-specified, even though there could be infinite ways to interpret an event depending on the rendering location. That leads to many different approaches in representation of events for classification in computer science and machine learning. Events can be represented atomically, i.e., entire events are predicted in a classification manner (Shahroudy et al. 2016), or as combinations of more primitive actions (Veeraraghavan, Papanikolopoulos, and Schrater 2007), i.e., complex event types are learned based on recognition of combined primitive actions. For the former type of event representation, there are quantitative approaches based on low-level pixel features such as in (Le et al. 2011) and qualitative approaches such as induction from relational states among event participants (Dubba et al. 2015). For the latter approach, systems such as (Hoogs and Perera 2008), use state transition graphical models such as Dynamic Bayesian Networks (DBN).

Event classification using qualitative spatial methods have been discussed in a fair amount of work. (Suchan, Bhatt, and Santos 2013) used Regional Connected Calculus (RCC5) adjusted for depth field with data also recorded by Kinect®sensor. This work classified events related to people moving or sitting or passing each other. (Dubba et al. 2015) provided an interesting framework which give summary explanation for a sequence of observations by alternating between inductive and abductive commonsense reasoning. This work was applied for two activity types, one being activities happened at an airport as boundary boxes of aircraft and trucks are tracked, one of human interaction with object and with each other.

QSRLib and extension

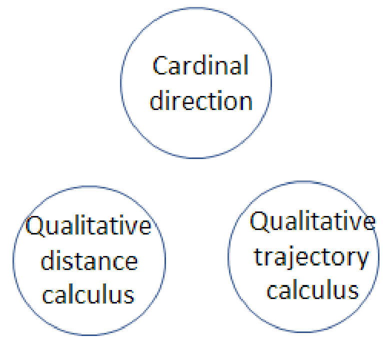


Figure 1: Three feature types that we implemented 3-dimensional extension to QSRLIb

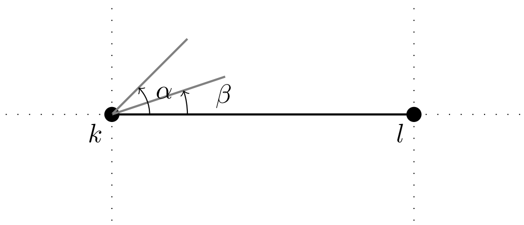
We used the following feature types in QSRLib:

- Cardinal direction(Andrew, Mark, and White 1991) (QSRLIb *cardir*) measures compass relations between two objects into canonical directions such as North, North East etc.

- Moving or static (QSRLIb *mos*) measures whether a point is moving or not.

- Qualitative Distance Calculus (QSRLib *argd*) discretizes the distance between two moving points. There is a significant portion of literature supporting use of discretization for feature embedding. (Yang and Webb 2009) shows that "discretization is equivalent to using the true probability density function". More recently, (Jiang et al.) has used this method for classification of GPS trajectory. They studied three different approaches for ways of discretization, including *equal-width binning*, Recursive Minimal Entropy Partitioning (RMEP) (Dougherty et al. 1995) and fuzzy discretization(Roy and Pal 2003). Their findings is that *equal-width binning* approach is both simple and effective, so we used this approach with interval length of 1/20 meter in embedded space. We didn't try other interval lengths, leaving that for future experiments.

- Qualitative Trajectory Calculus (Double Cross) (QSRLib *qtccs*): QCC_C is a representation of motions between two objects by considering them as two moving point objects (MPOs) (Delafontaine, Cohn, and Van de Weghe 2011). The type C21 of QCC_C (implemented in QSRLib) considers whether two points are moving toward each other or whether they are moving to the left or to the right of each other. Apparently, this is the kind of spatial semantics that are needed to learn the prepositions we used in this experiment. The following diagram explains this:



QCC_C produces a tuple of 4 slots (A, B, C, D) whereas each could be given either -, + or 0, depending on the angle α. For example, C is $+$ if $\alpha > 0 \wedge \alpha < 180$, $-$ if $\alpha > 180 \wedge \alpha < 360$ and 0 otherwise. QSRLib also allows specification of a *quantisation factor* θ, which dictates whether the movement of a point is significant in compare to the distance between k and l.

Modification of Qualitative trajectory calculus Double Cross (QCC_C) implementation in QSRLib:

QCC_C is however not a good calculus to be used in event classification without any modification, because approximating objects as MPOs leads to loss of representation. Exact modeling of object 3-d volumes are feasible but cumbersome.

Our solution for this involves an *angle quantisation factor* $\beta > 0$ which should be relatively small. When $|\alpha| < \beta$ or $|\alpha - \pi| < \beta$, we set the the value of C to 0. Similarly for slot A, when $|\alpha - \frac{\pi}{2}| < \beta$ or $|\alpha - \frac{3\pi}{2}| < \beta$, we also set the value to be 0.

3-dimensional extension We extended QSRLib for the following feature types:

- Cardinal direction (3D): We followed the 3D grid approach to partition the space into 3x3x3=27 voxels (Sabharwal and Leopold 2014). The center voxel is the Minimum Bounding Hyper-rectangle (MBHR) of a reference object. In this framework, taken two objects A (reference object) and B (target object), cardinal direction from A to B is calculated firstly by generating the MBHR of A (it could be approximated), then setting up the 3D grid for A, then finding which voxels intersect with B. We actually used a much simpler alternative that replaces B with its centroid, so that only one direction is resulted.
- Qualitative Distance Calculus (3D): This is basically the same as for two dimensional.
- Qualitative Trajectory Calculus (3D): It is noted that there is no features analogous to lateral slots (C, D) for 3-dimension. (Mavridis et al. 2015) provides an alternative for lateral relations, using Frenet-Serret frame (FS frame). We don't give the details of calculation here, as we provide our implementation. In a nutshell, the calculation steps are as followings:
  - For each point P and Q, calculate the tangent vector, the binormal vector and the normal vector. For continuous data domain, this requires calculating second derivative of each point's moving curve. For discrete domain, this translates to taking three data points into account for each calculation, of the current and two previous time steps. These three vectors create a FS frame for each moving point.
  - Assuming that two FS frames F_P and F_Q are calculated (special values are assigned in the case of degeneration), we need to find a transformation matrix from F_P to F_Q. This rotation matrix is in turn decomposed into three values of yaw, pitch and roll angles. These three values, together with two feature values (A, B) in QCC_C make a tuple of 5 values in our QCC_{3D}.

Feature extraction

Figure 2 shows our downstream feature extraction methods.

Our motivation for creating downstream features is of the following basic intuition.

- Object Model: State-by-state characterization of an object as it changes or moves through time.
- Action Model: State-by-state characterization of an actors motion through time. When action involves multiple objects, this also includes effect of objects on each other.
- Event Model: Composition of the object model with the action model.

From recognition point of view, object model is translated to inherent motion of objects whereas action model is translated to inter-object relative motion.

$$M_E = M_R * \prod_{i \in [1,2]} M_{Oi} * M_{O1O2} * \prod_{i \in [1,2]} M_{ROi}$$

whereas R stands for human body rig, O_i stands for objects. This method factorizes the model representation into $n*(n+3)/2$ terms where n is the number of objects. Apparently it is not a very economical way of representation when there are large number of objects in the scene ($O(n^2)$), but in reality, either a number of objects are relatively static to each other, or we in fact need to consider smaller number of objects to allow possible description (think of Lunar eclipse occurs when the Moon passes directly behind the Earth into Earth's shadow, aligning with the sun and Earth, where we in fact do not take into account other planets movement).

Preprocessing

The raw data come from two sensors on Kinect®: RGB camera and time of flight (ToF) depth sensor. In turn these produce three streams of vision input: a stream for RGB, a stream for depth field and a stream for tracked human body rigs. These data streams have different rates and resolutions.

Tracking object: We used Glyph detection algorithm (Kirillov 2016) with some adjustment to detect Glyph markers stuck on the objects (Glyph markers are black and white checked square). Markers put on different objects are different to simplify the tracking process. For frames where tracking is lost, marker's 2-d position is interpolated. 2 dimensional data is projected into 3-dimensional by using depth field. Body rig joint points are already tracked by Kinect®'s SDK.

Normalizing rate: Different streams of data were regenerated with the same rate by re-sampling with interpolation. We used the rate of 24fps, which is the same as the RGB stream.

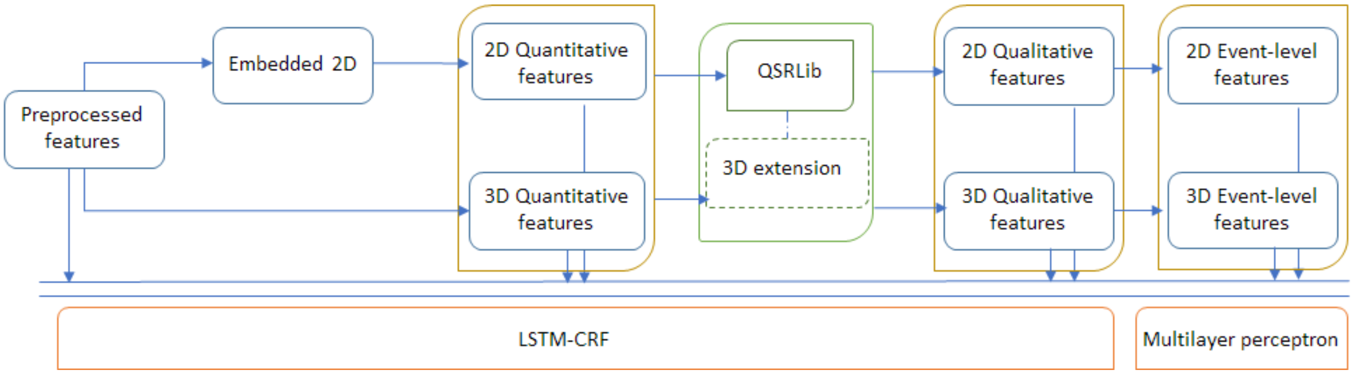


Figure 2: Downstream feature extraction methods used in this study. Our focus is on the performance gain from quantitative features to qualitative features through the use of QSRLib and its extension for 3-dimensional data

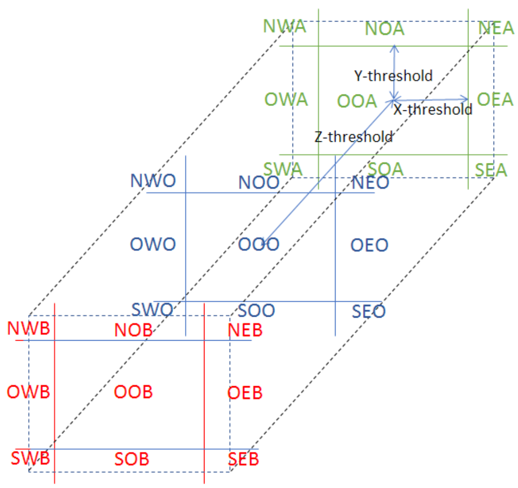


Figure 3: 3D Grid around a point. N, S, W, E, B, A stand for North, South, West, East, Below, Above.

Quantitative features

3D features are generated by the following methods:

- Relative motions between different objects are approximated by calculating the distance vectors among these entities.
- To model human body rig, vectors among the following points are calculated: middle point between the shoulders, left hand tip and right hand tip.
- To model objects, vectors between two diagonally opposing points are used.

Embedded 2D features

For each factor model, we used Principle Component Analysis (PCA) to project points considered into 2-dimensional planes, with the hope that the kept dimensions will keep the maximum variation, while provide an efficient way to visualize and reason about the data. The set of features for each frame is analogous to the 3-D case, but with all data points from each factor model projected through PCA.

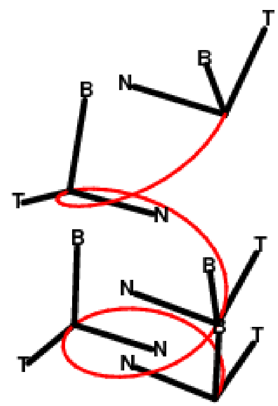


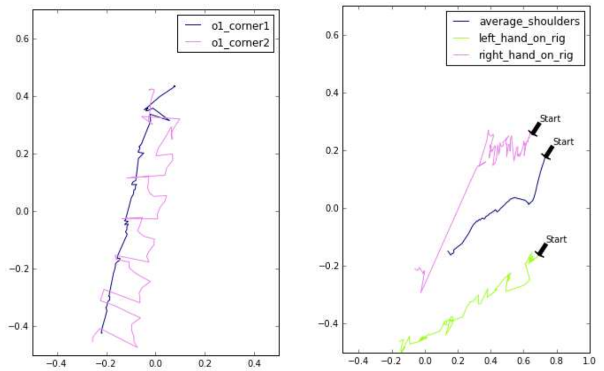
Figure 4: Frenet-Serret frame for 3-D qualitative trajectory calculus

Frame-level qualitative features

The set of qualitative features are *downstream* features from the set of quantitative features. We employed 4 feature types as listed before. Visualizing trajectories of objects in embedded spaces gave us insight the quality of our extracted features. For example, we observed that 2D qualitative trajectory calculus has a strong explanatory power in distinction of inherent motion of objects. For example, Figure 5a shows typical trajectories of two corner points of a rolling object. Direction between these points shows periodical change that is easily captured as change of cardinal directions in the feature space.

Event-level qualitative features

A *downstream* set of event features would require a method to summarize the change of frame-level features across the event duration. There are two ways to do that, one is by specifying a set of primitive actions, such as in (Suchan, Bhatt, and Santos 2013). This approach requires a hierarchical *fuzzy decomposition* of event into subevents. For our problem, we use a simpler and *feature-based only* representation. In fact, we just use the frame-level features of the first frame and the last frame, plus the different vector between



(a) Corners of a rolling object (b) Hands in relative to body

Figure 5: Samples of trajectories in embedded spaces

these two.

Learning algorithms

Multilayer perceptron learning (MLP)

For event-level features, we use a simple multilayer perceptron model for comparison with our sequential neural network models. Multilayer perceptron (MLP) is one of the earliest neural network model, employing a feedforward infrastructure with backpropagation update. Here we use a rectified MLP (a stack of layers in which each has a linear layer combined with a Rectified Linear Unit (Glorot, Bordes, and Bengio 2011)). Dropout is also applied to reduce overfitting.

Long-short term memory (LSTM)

LSTM is a flavor of deep Recursive neural network (RNN) that has generally solved the problem of "vanishing gradients" in traditional RNN learning (Hochreiter and Schmidhuber 1997; Schmidhuber 2015) and has found their application in a wide range of problems involving sequential learning, such as hand written recognition, speech recognition, gesture recognition, etc.

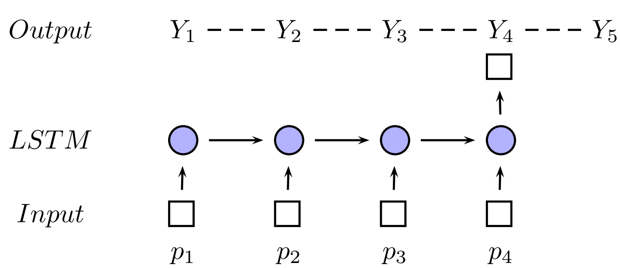


Figure 6: LSTM model with possible constraints of outputs with CRF. CRF layer is represented as dashed links among predicted labels.

We will not describe in details here LSTM model, as it has become a standard sequential learning method. Interested readers are recommended to take a look at either our implementation [1] or a popular page on RNN and LSTM [2]. Briefly, however, the model passes each feature vector through a linear layer before feeding each sequence into an LSTM. Each label Y_i requires a separate LSTM cell, X_i. Output of each LSTM cell is a term $t, t \in t_s, t_v, t_o, t_p, t_l$ corresponding to 5 semantic slots in the tuple. We will combine these values with our CRF weights, discussed in LSTM-CRF.

Conditional Random Field (CRF)

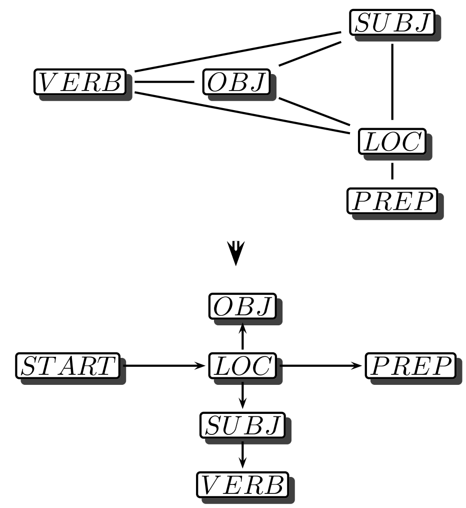


Figure 7: Reformation from general **CRF** (left) to **Tree-CRF** (right)

CRF has been used extensively to learn structured output as it allows specification of constraints of output labels(Sutton and McCallum 2006). In this model we wish to constrain the outputs so that: one object (Performer or the other objects) is not allowed to fill two different syntactic slots; when there is no verb, all the other slots should be None; locative and preposition are dependent, because if locative is None, preposition must also be None and vice versa. The edges between nodes on the left side of Figure 7 show the dependencies on output labels that we wish to model.

However, training and classifying using a full CRF model would be more difficult, especially when implemented with a neural network architecture. We modified the model into a tree-CRF structure (right side of Figure 7) to make the model learnable using dynamic programming. The complexity of the algorithm reduced from $O(n^5)$ to $O(n^2) * 5)$ where n is the size of our vocabulary. The learning problem is thereby changed to learning the weights along the edges on the tree-CRF, for example, $P\_locative\_preposition$ (together with parameters of LSTM). The directionality of edges is the forward direction of the message passing algorithm used for learning (and in reverse, for testing using the backward direction).

Use of CRF as a constraint layer Naturally, as we have learned 5 models (either 5 MLP or 5 LSTM) to predict 5 values in the output, we want to regularize the output combination by CRF. Moreover, LSTM-CRF is a natural extension

[1] https://github.com/tuandnvn/ecat_learning

[2] http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/

of LSTM applied for constrained outputs. For instance it is used for named entities recognition task to model constraints on BIO labels(Huang, Xu, and Yu 2015).

To put CRF learning on top of MLP or LSTM, we modify the term t (the term before softmax) produced by outputs of MLP or LSTM as followings.

$$
\begin{aligned}
t(l,s,o,p,v) &= t_l + t_s + t_o + t_p + t_v \quad \textbf{Original target} \\
t(l,s,o,p,v) &= t_l + t_s + t_o + t_p + t_v \quad \textbf{Modified target} \\
&\quad + P_{start\_l} + P_{ls} + P_{lo} + P_{lp} + P_{sv}
\end{aligned}
$$

where l, s, o, p, v stand for Locative, Subject, Object, Preposition and Verb respectively.

In training, $softmax$ is calculated for a predicted label combination, namely (l', s', o', p', v') as below. We can calculate the log of sum using message passing over the tree nodes of the CRF tree. We use cross entropy between predicted distribution and correct output as the $cost$ in training.

$$
\begin{aligned}
softmax &= exp[t(l', s', o', p', v') - \\
&\quad log[\sum_l \sum_s \sum_o \sum_p \sum_v exp(t(l,s,o,p,v))]] \\
&= exp[t(l', s', o', p', v') - \\
&\quad log[\sum_l exp(t_l + P_{start\_l})[\sum_s exp(t_s + P_{ls}) \\
&\quad \sum_v exp(t_v + P_{sv})][\sum_o exp(t_o + P_{lo})][\sum_p exp(t_p + P_{lp})]]
\end{aligned}
$$

In evaluation, a similar message passing algorithm is used, but instead of log_sum, we use max to calculate the probabilities and $argmax$ to keep track of the best combination.

Experimental setup

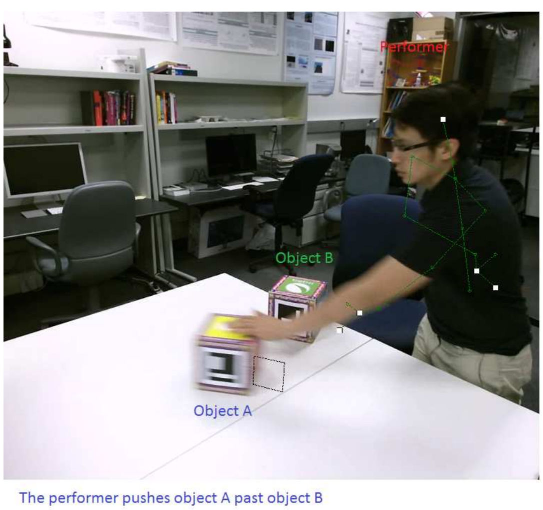


Figure 8: Event capture with fine-grained annotation

To demonstrate our model's capability to learn the *spatio-temporal* dynamics of object interactions in events, we use a collection of four action types: *push*, *pull*, *slide*, and *roll*, along with three different *spatial* prepositions used for space configurations between objects, namely *toward* (when the trajectory of a moving object is straightly lined up with a destination static object and makes it closer to that target), *away from* (makes it further from that object) and *past* (moving object getting closer to static object then further again).

Afterwards, for each session, we sliced the events into short segments of 20 frames. Two annotators were assigned to watch and annotate them (segments can be played back). To speed up annotation, only event types related to original captured types are shown for selection. For instance, if the event type of the captured session is "The performer pushes A toward B", other available event types are "The performer pushes A", "A slides toward B" or "None".

We tried different combinations of hyperparameters for our MLP and LSTM models. Two methods are used to combat over-fitting: (i) dropout (Hinton et al. 2012) (for LSTM this a dropout wrapper on LSTM cell, for MLP this is a dropout wrapper on each layer) and (ii) gradient clipping by a global norm. Following is the list of hyperparameters we used for tuning:

- Number of LSTM layers: [1, 2]
- Size of hidden layer: [200, 400]
- Learning rate: [0.05, 0.1, 0.2, 0.5]
- Dropout rate: [0.5, 0.6, 0.8]
- Learning rate decay: [0.94, 0.95, 0.96]

The network is trained with mini-batch gradient descent optimization for 200 epochs (LSTM) or 500 epochs (MLP) on the Tensorflow library.

Results

Captured sessions are split for 5-fold cross validation, i.e., 24 sessions for training and 6 for testing on each fold. We use the LSTM-CRF model with raw input data as the baseline. A prediction is correct if all slots are correct. Performance is reported in the following tables:

	Model	Precision
Frame level	3D-Raw-LSTM-CRF	43%
	3D-Quant-LSTM-CRF	44%
	3D-Qual-LSTM-CRF	52%
	2D-Quant-LSTM-CRF	48%
	2D-Qual-LSTM-CRF	**60%**
Event level	3D-Event-Qual-LSTM-CRF	20%
	2D-Event-Qual-LSTM-CRF	23%

Table 1: Evaluation

Label	Precision
Subject	93%
Object	90%
Locative	80%
Verb	83%
Preposition	82%

Table 2: Label precision breakdown for Frame-Qual-LSTM-CRF

We can observe a significant improvement of classification using 2-dimensional frame-level qualitative features. Frame-level quantitative features did improve over our baseline, but the improvement is not as impressive. Moreover, summarizing frame-level features to create event-level features create a lossy representation that is not be able to learned efficiently.

We want to give some discussion and explanation for our results. Firstly, as pointed out by (Yang and Webb 2009) or (Jiang et al.), qualitative representation is a method of discretization, which makes data sparser, therefore easier to learn. Especially when taking the difference between features of two adjacent frames, as qualitative feature strongly distinguishes between 0 and 1, effect of sequential change is more pronounced.

Moreover, we observed that the best performance for **Qual-LSTM-CRF** is achieved by configuring two layers of LSTM with 400 nodes on the hidden layers, while for other models, the number of layers doesn't affect the performance significantly. Different from a feed-forward neural network such as Convolutional Neural Network (CNN), which can learn more abstract and useful features when it get deeper, LSTM needs some help from representation of features to reap benefit from going deeper.

We also learned that a simple summary representation for events is not effective. If we also takes into account features in intermediate frames, the representation comes back to a sequential one. There might be a better way to do so, such as summarizing for each feature separately, or using the inductive/abductive method but we would leave that for future investigation.

Conclusion and Future Directions

In this study we have explored a wide range of feature extraction methods to learn what is the best way to represent event for classification. Even though our event domain is quite artificial, it represents complex human-object interaction, and therefore we believe that our conclusion could be generalized to other domains of similar complexity, such as of human-robot interaction.

We acknowledge our shortcoming in the investigation of event-level qualitative features. More complex method, such as one of (Dubba et al. 2015) could be used in similar manner, and we will leave that for future direction.

One more dimension that we wanted to explore but we haven't is that of the embedding method from 3-dimensional to 2-dimensional data. Methods other than PCA could be used, such as multidimensional scaling. We will also leave that for future discussion.

Acknowledgements

This work is supported by a contract with the US Defense Advanced Research Projects Agency (DARPA), Contract W911NF-15-C-0238. Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We would like to thank Nikhil Krishnaswamy and Keigh Rim for their discussion and input on this topic. All errors and mistakes are, of course, the responsibilities of the authors..

References

Allen, J. 1984. Towards a general theory of action and time. *Arificial Intelligence* 23:123–154.

Andrew, U.; Mark, D.; and White, D. 1991. Qualitative spatial reasoning about cardinal directions. In *Proc. of the 7th Austrian Conf. on Artificial Intelligence. Baltimore: Morgan Kaufmann*, 157–167.

Delafontaine, M.; Cohn, A. G.; and Van de Weghe, N. 2011. Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications* 38(5):5187–5196.

Do, T., and Pustejovsky, J. 2017. Fine-grained event learning of human-object interaction with lstm-crf.

Do, T.; Krishnaswamy, N.; and Pustejovsky, J. 2016. Ecat: Event capture annotation tool. *Proceedings of ISA-12: International Workshop on Semantic Annotation*.

Dougherty, J.; Kohavi, R.; Sahami, M.; et al. 1995. Supervised and unsupervised discretization of continuous features. In *Machine learning: proceedings of the twelfth international conference*, volume 12, 194–202.

Dubba, K. S. R.; Cohn, A. G.; Hogg, D. C.; Bhatt, M.; and Dylla, F. 2015. Learning relational event models from video. *Journal of Artificial Intelligence Research* 53:41–90.

Fillmore, C. 1975. *Santa Cruz Lectures on Deixis*. Indiana University Linguistics Club. Bloomington, IN.

Gatsoulis, Y.; Alomari, M.; Burbridge, C.; Dondrup, C.; Duckworth, P.; Lightbody, P.; Hanheide, M.; Hawes, N.; and Cohn, A. 2016. Qsrlib: a software library for online acquisition of qualitative spatial relations from video. In *Workshop on Qualitative Reasoning (QR16), at IJCAI-16*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Aistats*, volume 15, 275.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hoogs, A., and Perera, A. A. 2008. Video activity recognition in the real world. In *AAAI*, 1551–1554.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jiang, X.; N de Souza, E.; Liu, X.; Haji Soleimani, B.; Wang, X.; L. Silver, D.; and Matwin, S. Partition-wise recurrent neural networks for point-based ais trajectory classification.

Kirillov, A. 2016. From glyph recognition to augmented reality. `https://www.codeproject.com/articles/258856/from-glyph-recognition-to-augmented-reality`. Accessed: 2016-03-15.

Koppula, H. S., and Saxena, A. 2016. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence* 38(1):14–29.

Le, Q. V.; Zou, W. Y.; Yeung, S. Y.; and Ng, A. Y. 2011. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 3361–3368. IEEE.

Li, W., and Fritz, M. 2016. Recognition of ongoing complex activities by sequence prediction over a hierarchical label space. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, 1–9. IEEE.

Mavridis, N.; Bellotto, N.; Iliopoulos, K.; and Van de Weghe, N. 2015. Qtc 3d: Extending the qualitative trajectory calculus to three dimensions. *Information Sciences* 322:20–30.

Pustejovsky, J.; Lee, K.; Bunt, H.; and Romary, L. 2011. Iso-timeml: A standard for annotating temporal information in language. In *LREC*.

Pustejovsky, J. 1995. *The Generative Lexicon*. Cambridge, MA: MIT Press.

Rohrbach, M.; Amin, S.; Andriluka, M.; and Schiele, B. 2012. A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 1194–1201. IEEE.

Roy, A., and Pal, S. K. 2003. Fuzzy discretization of feature space for a rough set classifier. *Pattern Recognition Letters* 24(6):895–902.

Ryoo, M. S., and Aggarwal, J. 2010. Ut-interaction dataset, icpr contest on semantic description of human activities (sdha). In *IEEE International Conference on Pattern Recognition Workshops*, volume 2, 4.

Sabharwal, C. L., and Leopold, J. L. 2014. Modeling cardinal direction relations in 3d for qualitative spatial reasoning. In *Mining Intelligence and Knowledge Exploration*. Springer. 199–214.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

Shahroudy, A.; Liu, J.; Ng, T.-T.; and Wang, G. 2016. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. *arXiv preprint arXiv:1604.02808*.

Suchan, J.; Bhatt, M.; and Santos, P. E. 2013. Perceptual narratives of space and motion for activity interpretation. In *27th International Workshop on Qualitative Reasoning*, 32.

Sutton, C., and McCallum, A. 2006. *An introduction to conditional random fields for relational learning*, volume 2. Introduction to statistical relational learning. MIT Press.

Tulving, E. 1983. *Elements of Episodic Memory*. Oxford University Press.

Veeraraghavan, H.; Papanikolopoulos, N.; and Schrater, P. 2007. Learning dynamic event descriptions in image sequences. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–6. IEEE.

Yang, Y., and Webb, G. I. 2009. Discretization for naive-bayes learning: managing discretization bias and variance. *Machine learning* 74(1):39–74.