Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

Entity Embedding-Based Anomaly Detection for Heterogeneous Categorical Events

Ting Chen,[1*] **Lu-An Tang,**[2] **Yizhou Sun,**[1] **Zhengzhang Chen,**[2] **Kai Zhang**[2]
[1]Northeastern University, [2]NEC Labs America
{tingchen, yzsun}@ccs.neu.edu, {ltang, zchen, kzhang}@nec-labs.com

Abstract

Anomaly detection plays an important role in modern data-driven security applications, such as detecting suspicious access to a socket from a process. In many cases, such events can be described as a collection of categorical values that are considered as entities of different types, which we call heterogeneous categorical events. Due to the lack of intrinsic distance measures among entities, and the exponentially large event space, most existing work relies heavily on heuristics to calculate abnormal scores for events. Different from previous work, we propose a principled and unified probabilistic model APE (Anomaly detection via Probabilistic pairwise interaction and Entity embedding) that directly models the likelihood of events. In this model, we embed entities into a common latent space using their observed co-occurrence in different events. More specifically, we first model the compatibility of each pair of entities according to their embeddings. Then we utilize the weighted pairwise interactions of different entity types to define the event probability. Using Noise-Contrastive Estimation with "context-dependent" noise distribution, our model can be learned efficiently regardless of the large event space. Experimental results on real enterprise surveillance data show that our methods can accurately detect abnormal events compared to other state-of-the-art abnormal detection techniques.

1 Introduction

With increasing amount of data collected from everywhere, such as computer systems, transaction activities, social networks, it becomes more and more important for people to understand the underlying regularity of the data, and to spot the unexpected or abnormal instances [Chandola *et al.*, 2009]. Centered around this goal, anomaly detection plays a very important role in many security related applications, such as securing enterprise network by detecting abnormal connectivities, and so on.

However, the problem has not been satisfyingly addressed yet. Many traditional anomaly detection methods focus on either numerical data or supervised settings [Chandola *et al.*, 2009]. When it comes to unsupervised anomaly detection in heterogeneous categorical events data, i.e., events containing a collection of categorical values that are considered as entities of different types, there is less existing work [Das and Schneider, 2007; Das *et al.*, 2008; Tong *et al.*, 2008; Akoglu *et al.*, 2012].

The heterogeneous categorical event data are ubiquitous, such as events of process interactions in computer systems, where each data point is an event that involves heterogeneous types of attributes/entities: time, user, source process, destination process, and so on. In order to detect abnormal events that deviate from the regular patterns, a common approach is to build a model that can capture the underlying factors/regularities of data. However, events with multiple heterogeneous entities are difficult to model in a systematic and unified framework due to two major challenges: (1) the lack of intrinsic distance measures among entities and events, and (2) the exponentially large event space.

Consider that in real computer systems, given two users with ids of 1 and 10, we almost know nothing about their distance/similarity without other information. In addition to the lack of intrinsic distance measure, the exponentially large event space is also an issue. For example, a heterogeneous categorical event, in real systems, can involve more than ten types of entities. If each entity type has more than one hundred possible choices of entities the overall event space will be as large as 100^{10}, which is prohibitively large and makes it challenging to model regularities.

Due to these two difficulties, most existing work relies heavily on heuristics to quantify the normal/abnormal scores for events [Das and Schneider, 2007; Das *et al.*, 2008; Tong *et al.*, 2008; Akoglu *et al.*, 2012]. However, a more systematic and accurate model is in demand as the vastly emerging of big complicated data in important applications.

To tackle the aforementioned challenges, we propose a probabilistic model that directly models the event likelihood. We first embed entities into a common latent space where distance among entities can be naturally defined. Then to access the compatibility of entities in the event, we quantify

[*]Part of the work is done during first author's internship at NEC Labs America.

their pairwise interactions by the dot product of the embedding vectors. Finally the weighted sum of interactions is used to define the probability of the event.

Compared to traditional methods, the proposed method has several advantages: (1) by modeling the likelihood of event based on entity embeddings, the proposed model can produce normal/abnormal score in a principled and unified framework; (2) by modeling weighted pairwise interaction instead of all possible interactions, the model is less susceptible to over-fitting, and can provide better interpretability; and (3) the proposed model can be learned efficiently by Noise-Contrastive Estimation with "context-dependent" noise distribution regardless of large event space. Empirical studies on real-world enterprise surveillance data show that by applying our method we can detect unknown abnormal events accurately.

2 Problem Statement

Here we introduce some notations and define the problem.

Heterogeneous Categorical Event. A heterogeneous categorical event $e = (a_1, \cdots, a_m)$ is a record contains m different categorical attributes, and the i-th attribute value a_i denotes an entity from the type A_i. In the computer process interaction network, an event is a record involving entities of types such as the user, time[1], source/destination process and folder. In the following, we will call it event for short.

By treating the categorical attributes of an event as entities/nodes, we can also view categorical events as a heterogeneous network of multiple node types [Sun and Han, 2012]. In the computer process interaction example, the network schema is shown in Figure 1, where event acts as a super node connecting other nodes of different types.

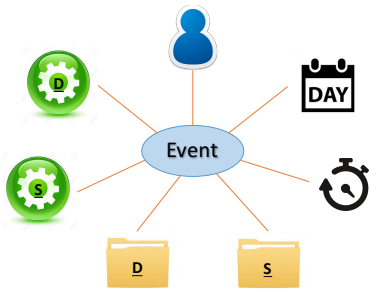


Figure 1: The heterogeneous network view of categorical events. Node types include event, user, day, hour, source/destination process and folder.

Problem: abnormal event detection. Given a set of training events $D = \{e_1, \cdots, e_n\}$, by assuming that most events in D are normal, the problem is to learn a model M, so that when a new event e_{n+1} comes, the model M can accurately predict whether the event is abnormal or not.

[1]Although time is continuous value, it can be chunked into segments of different granularities, such as day and hour, which then can be viewed as entities.

3 The Proposed Model

In this section, we introduce the motivation and technical details about the proposed model.

3.1 Motivations

We directly model the event likelihood as it indicates how likely an event should occur according to the data. An event with unusual low likelihood is naturally abnormal. To achieve this, we need to deal with the two main challenges as mentioned before: (1) the lack of intrinsic distance measures among entities and events, and (2) the exponentially large event space.

To overcome the lack of intrinsic distance measures among entities, we embed entities into a common latent space where their semantic can be preserved. More specifically, each entity, such as a user, or a process in computer systems, is represented as a d-dimensional vector and will be automatically learned from the data. In the embedding space, the distance of entities can be naturally computed by distance/similarity measures in the space, such as Euclidean distances, vector dot product, and so on. Compared with other distance/similarity metrics defined on sets, such as Jaccard similarity, the embedding method is more flexible and it has nice property such as transitivity [Zhang *et al.*, 2015].

To alleviate the large event space issue and enable efficient model learning, we come up with two strategies: (1) at the model level, instead of modeling all possible interactions among entities, we only consider pairwise interaction that reflects the strength of co-occurrences of entities [Rendle, 2010]; and (2) at the learning level, we propose using noise-contrastive estimation [Gutmann and Hyvärinen, 2010] with "context-dependent" noise distribution.

The pairwise interaction is intuitive/interpretable, efficient to compute, and less susceptible to over-fitting. Consider the following anomaly example we may encounter in real scenarios:

- A maintenance program is usually triggered at midnight, but suddenly it is trigged during the day.
- A user usually connect to servers with low privilege, but suddenly it tries to access some server with high privilege.

In these examples, abnormal behaviors occur as a result of the unusual pairwise interaction among entities (process and time in the first example, and user and machine in the second example).

3.2 The probabilistic model for event

We model the probability of a single event $e = \{a_1, \cdots, a_m\}$ in event space Ω using the following parametric form:

$$P_\theta(e) = \frac{\exp\Big(S_\theta(e)\Big)}{\sum_{e' \in \Omega} \exp\Big(S_\theta(e')\Big)} \quad (1)$$

Where θ is the set of parameters, $S_\theta(e)$ is the scoring function for a given event e that quantifies its normality. We instantiate

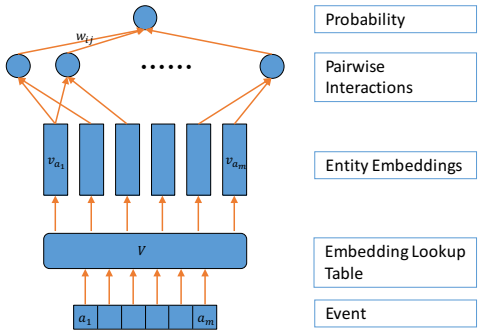


Figure 2: The framework of proposed method.

the scoring function by pairwise interactions among embedded entities:

$$S_\theta(e) = \sum_{i,j:1\le i<j\le m} w_{ij}(v_{a_i} \cdot v_{a_j}) \tag{2}$$

Where v_{a_i} is the embedding vector for entity a_i, and the dot product between a pair of entity embeddings v_{a_i} and v_{a_j} encodes the compatibility of two entities co-occur in a single event. w_{ij} is the weight for pairwise interaction between entity types A_i and A_j, and it is non-negative constrained, i.e. $\forall i, j, w_{ij} \ge 0$. Different pairs of entity types can have different importances, interaction among some pairs of entity types are very regular and important, e.g. user and machine, while others may be less regular and important, e.g. day and hour. Using w_{ij}, the model can automatically learn the importances of different pairwise interactions. Finally $\theta = \{w, v\}$ denotes all parameters used in the model.

Our model APE, which is abbreviated for Anomaly detection via Probabilistic pairwise interaction and Entity embedding, is summarized in Figure 2.

The learning problem is to optimize the following maximum likelihood objective over events in the training data D:

$$\arg\max_\theta \sum_{e\in D} \log P_\theta(e) \tag{3}$$

To solve the optimization problem, the major challenge is that the denominator in Eq. 1 sums over all possible event configurations, which is prohibitively large ($O(\exp m)$). To address this challenging issue, we propose using Noise Contrastive Estimation.

3.3 Learning via noise-contrastive estimation

Noise-Contrastive Estimation (NCE) has been introduced in [Gutmann and Hyvärinen, 2010] for density estimation, and applied to estimate language model [Mnih and Teh, 2012], and word embedding [Mnih and Kavukcuoglu, 2013; Mikolov *et al.*, 2013a; 2013b]. The basic idea of NCE is to reduce the problem of density estimation to binary classification, which is to discriminate samples from data distribution $P_d(e)$ and some artificial known noise distribution $P_n(e)$ (the selection of P_n will be explained later). In another word, the samples fed to the APE model can come from real training data set or being generated artificially, and the model is trained to classify them a posteriori.

Assuming generated noise/negative samples are k times more frequent than observed data samples, the posterior probability of an event e came from data distribution is $P(D = 1|e,\theta)$[2] $= P_\theta(e)/(P_\theta(e) + kP_n(e))$. To fit the objective in Eq. 3, we maximize the expectation of $\log P(D|e,\theta)$ under the mixture of data and noise/negative samples [Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012]. This leads to the following new objective function:

$$\begin{aligned} J(\theta) =& E_{e\sim P_d}\left[\log \frac{P_\theta(e)}{P_\theta(e) + kP_n(e)}\right] + \\ & kE_{e\sim P_n}\left[\log \frac{kP_n(e)}{P_\theta(e) + kP_n(e)}\right] \end{aligned} \tag{4}$$

However, in this new objective function, the model distribution $P_\theta(e)$ is still too expensive to evaluate. NCE sidesteps this difficulty by avoiding explicit normalization and treating normalization constant as a parameter. This leads to $P_\theta(e) = P_{\theta^0}(e)\exp(c)$, where $\theta = \{\theta^0, c\}$, and c is the original log-partition function as a single parameter, and is learned to normalize the whole distribution. Now we can re-write the event probability function in Eq. 1 as follows:

$$P_\theta(e) = \exp\left(\sum_{i,j:1\le i<j\le m} w_{ij}(v_{a_i} \cdot v_{a_j}) + c\right) \tag{5}$$

To optimize the objective E.q. 4 given the training data D, we replace P_d with $\tilde{P}_d$ (the empirical data distribution), and since the APE model is differentiable, stochastic gradient descent is used: for each observed training event e, first sample k noise/negative samples $\{e'\}$ according to the known noise distribution P_n, and then update parameters according to the gradients of the following objective function (which is derived from Eq. 4 on given $e, \{e'\}$ samples):

$$\begin{aligned} & \log\sigma\Big(\log P_\theta(e) - \log kP_n(e)\Big) + \\ & \sum_{e'} \log\sigma\Big(-\log P_\theta(e') + \log kP_n(e')\Big) \end{aligned} \tag{6}$$

Here $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function.

The complexity of our algorithm is $O(Nkm^2d)$, where N is the number of total observed events it is trained on, k is number of negative examples drawn for each observed event, m is the number of entity type, and d is the embedding dimension. The complexity indicates that the APE model can be learned efficiently regardless of the $O(\exp m)$ large event space.

[2] Since we want to fit the model distribution to data distribution, we use P_θ in place of P_d.

3.4 "Context-dependent" noise distribution

To apply NCE, as shown in Eq. 6, we need to draw negative samples from some known noise distribution P_n. Intuitively, the noise distribution should be close to the data distribution, otherwise the discriminating task would be too easy and the model cannot learn much structure from the data [Gutmann and Hyvärinen, 2010]. Note that, different from previous work (such as language modeling or word embedding [Mnih and Teh, 2012; Mikolov *et al.*, 2013a]) that utilizes NCE, where each negative sample only involves one word/entity. Each event, in our case, involves multiple entities of different types.

One straight-forward choice of noise distribution is *"context-independent"* noise distribution, where a negative event is drawn independently and does not depend on the observed event. One can sample a negative event according to some factorized distribution on event space, i.e. $P_n^{factorized}(e) = p_{A_1}(a_1) \cdots p_{A_i}(a_i)$. Here $p_{A_i}(a_i)$ is the probability of choosing entity a_i of the type A_i, which can be specified uniformly or computed by counting unigram in data. In this work we stick to unigram as it is reported better [Mnih and Teh, 2012; Mikolov *et al.*, 2013a].

Although the "context-independent" noise distribution is easy to evaluate. Due to the large event space, this noise distribution would be very different from data distribution, which will lead to poor model learning.

Here we propose a new *"context-dependent"* noise distribution where negative sampling is dependent on its context (i.e. the observed event). The procedure is, for each observed event e, we first uniformly sample an entity type A_i, and then sample a new entity $a_i' \sim p_{A_i}(a_i')$ to replace a_i and form a new negative sample e'. As we only modify one entity in the observed event, the noise distribution will be close to data distribution, thus can lead to better model learning. However, by utilizing the new "context-dependent" noise generation, it becomes very hard to compute the exact noise probability $P_n(e)$. Therefore, we use an approximation instead as follows.

For a given observed "context" event e, we define the "context-dependent" noise distribution for sampled event e' as $P_n^c(e'|e)$. Since e' is sampled by randomly replacing one of the entity a_i with a_i' of the same A_i type, the conditional probability $P_n^c(e'|e) = P_{A_i}(a_i')/m$ (here we assume A_i is chosen uniformly). Considering the large event space, it is unlikely that event e' is generated from observed events other than e, so we can approximate the noise distribution with $P_n(e') \approx P_n^c(e'|e)P_d(e)$. Furthermore, as $P_d(e)$ is usually small for most events, we simply set it to some constant l, which leads to the final noise distribution term (which is used in E.q. 6):

$$\log kP_n(e') \approx \log P_{A_i}(a_i') + z,$$

where $z = \log kl/m$ is a constant term. Although we do not know the exact value of z, we let $z = 0$ when plugging the approximated $\log kP_n(e')$ into Eq. 6. We find that ignoring z will only lead to a constant shift of learned parameter c. Since c is just the global normalization term, it will not affect the relative normal/abnormal scores of different events.

Table 1: Entity types in data sets.

Data	Types of entity and their arities
P2P	day (7), hour (24), uid (361), src proc (778), dst proc (1752), src folder (255), dst folder (415)
P2I	day (7), hour (24), src ip (59), dst ip (184), dst port (283), proc (91), proc folder (70), uid (162), connect type (3)

To compute $P_n(e)$ for an observed event e, since we do not know which entity is replaced as in the negative event case, we will use the expectation as follows:

$$\log kP_n(e) \approx \sum_i \frac{1}{m} \log P_{A_i}(a_i) + z.$$

And again the z will be ignored when plugging into Eq. 6.

4 Experiments

In this section, we evaluate the proposed method using real surveillance data collected in an enterprise system during a two-week period.

4.1 Data Sets

One of the main application scenarios of anomaly detection is to detect abnormal activity in surveillance data collected from computer systems. Hence, in our experiments, a two-week period of activity data of an enterprise computer system is used. The collected surveillance data include two types of events, which are viewed as two separate data sets.

P2P. Process to process event data set. Each event of this type contains the system activity of a process interacting with another process, the time and user id of the event are also recorded. P2P events are among the most important system activities since modern operating systems are based on processes.

P2I. Process to Internet Socket event data set. Each event of this type contains the system activity of a process sending or receiving Internet connections to/from other machine at destination ports, the time and user id of the event are recorded as well. We only consider the P2I events among the enterprise system since we focus on inside enterprise activities.

The entity types and their number of entities for both data sets are summarized in Table 1.

We do not have the ground-truth labels for collected events, however, it is assumed that majority of events are normal. In order to evaluate anomaly detection task, similar to [Das and Schneider, 2007; Das *et al.*, 2008; Akoglu *et al.*, 2012], we create some artificial anomalies, and ask the algorithms to detect them. The artificial anomaly events are generated as follows: for each event in the test data, we select c of its entities (we consider $c = \{1, 2, 3\}$ in following experiments), randomly replace them with other entities of the same type, and make sure the new generated events do not occur in both training and test data sets, so that they can be considered more abnormal than observed events.

We split the two-week data into two of one-weeks. The

Table 2: Statistics of the collected two-week events.

Data	# week 1	# week 2	# week 2 new
P2P	95,434	107,619	53,478 (49.69%)
P2I	1,316,357	1,330,376	498,029 (37.44%)

events in the first week are used as training set[3], and *new* events that only appeared in the second week are used as test sets. The statistics of observed events are summarized in Table 2.

4.2 Comparing methods and settings

We compare the following state-of-the-art methods for abnormal event detection.

Condition. This method is proposed in [Das and Schneider, 2007]. For each test event, it computes the conditional scores for all pairs of dependent and mutually exclusive subsets having up to k attributes, and combine the scores with a heuristic algorithm. The conditional score is calculated based on statistics of events in the training set, and reflect dependencies between two given attribute sets of an event.

CompreX. This method is proposed in [Akoglu *et al.*, 2012]. It utilizes a compression technique to encode training data and learns a set of code tables that summarize patterns. When a new event comes, it first encodes it using existing code tables, and then the number of bits used in encoding is treated as abnormal score for the event.

APE. This is the proposed method. Noted that we use the negative of its likelihood output as the abnormal score.

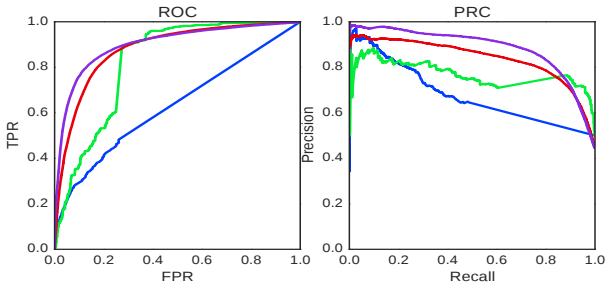
APE (no weight). This method is the same as APE, except that instead of learning w_{ij}, we simply set $\forall i, j, w_{ij} = 1$, i.e. it is APE without automatic weights learning on pairwise interactions. All types of interactions are weighted equally.

For the (hyper-)parameter settings, we use part of the training data as validation set to tune (hyper-)parameters. For Condition, we set $k = 1, \alpha = 1, \beta = 0.5$. For CompreX, we adopt their implementation, and since it is parameter free, we do not need to tune any parameters. For both APE and APE (no weight), the following setting is used: the embedding is randomly initialized, and dimension is set to 10; for each observed training event, we draw 3 negative samples for each of the entity type, which accounts for a total of $3m$ negative samples per training instance; we also use a mini-batch of size 128 for speed up stochastic gradient descent, and 5-10 epochs are general enough for convergence.

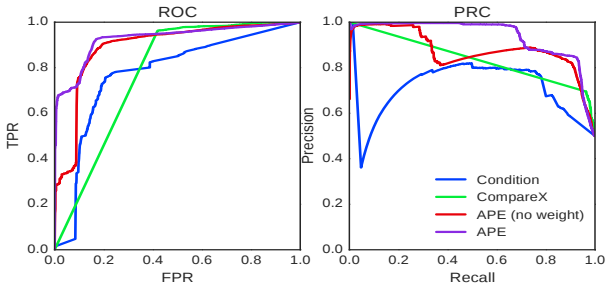
4.3 Evaluation Metrics

Since all methods listed above produce abnormal scores instead of binary labels, and there is no fixed threshold, thus metrics for binary labels such as accuracy are not suitable for measuring the performance. Similar to [Das and Schneider, 2007; Akoglu *et al.*, 2012], we adopt ROC curves (Receiver Operating Characteristic curves) and PRC (Precision Recall curves) for evaluation. Both of these two curves reflect the quality of predicted scores according to their true labels at different threshold levels. A detailed discussion about the two metrics can be found in [Davis and Goadrich, 2006]. To get a quantitative measurements, the AUC (area under curve) of both ROC and PRC are utilized.

[3]With randomly selected portion as validation set for selection of hyper-parameters.



(a) P2P abnormal event detection.



(b) P2I abnormal event detection.

Figure 3: Receiver operating characteristic curves and precision recall curves for abnormal event detections.

4.4 Results for abnormal event detection

Table 3 shows the AUC of ROC and PRC of different methods on P2P and P2I data sets. Note the last two rows in Table 3 are mean scores averaged over three sampled smaller test sets, due to the slowness of CompreX at test time (which can takes hundreds of hours to finish on the half million sized P2I events). Figure 3 shows both ROC curves and PR curves for all methods using test set with entity replacement $c = 1$ (for $c = 2, 3$, results are similar thus not shown).

From the results we can see, on different c number of entity replacement, our method consistently outperforms both Condition and CompreX significantly. When comparing APE with APE (no weight), we see that by considering weights and learning them automatically, the detection results can be further improved.

The learned weight matrix W for P2P and P2I events can be found in Figure 4 and 5, respectively. The matrix is upper-triangulated since the pairwise interaction is symmetric and model only among different type of entities. From the weights, we can see the importance of different types of interactions in the data sets. For example, in P2P events, the weight for interaction between day and hour is insignificant; while the weight for interaction between source process and destination process is large, indicating they are highly dependent and capture the regularity of P2P events.

Table 4 shows some detected abnormal events (we only

Table 3: Performance of abnormal event detection. Values left to slash are AUC of ROC, and ones on the right are average precision. The last two rows (* marked) are averaged over 3 smaller (1%) test samples due to long runtime of CompreX.

	P2P			P2I		
Models	c=1	c=2	c=3	c=1	c=2	c=3
Condition	0.6296 / 0.6777	0.6795 / 0.7321	0.7137 / 0.7672	0.7733 / 0.7127	0.8300 / 0.7688	0.8699 / 0.8165
APE (no weight)	0.8797 / 0.8404	0.9377 / 0.9072	0.9688 / 0.9449	0.8912 / 0.8784	0.9412 / 0.9398	0.9665 / 0.9671
APE	**0.8995 / 0.8845**	**0.9540 / 0.9378**	**0.9779 / 0.9639**	**0.9267 / 0.9383**	**0.9669 / 0.9717**	**0.9838 / 0.9861**
CompreX*	0.8230 / 0.7683	0.8208 / 0.7566	0.8390 / 0.7978	0.7749 / 0.8391	0.7834 / 0.8525	0.7832 / 0.8497
APE*	**0.9003 / 0.8892**	**0.9589 / 0.9394**	**0.9732 / 0.9616**	**0.9291 / 0.9411**	**0.9656 / 0.9729**	**0.9829 / 0.9854**

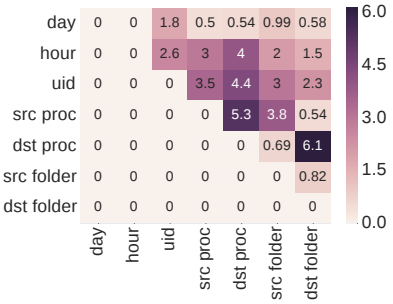


Figure 4: Pairwise weights learned for P2P events.

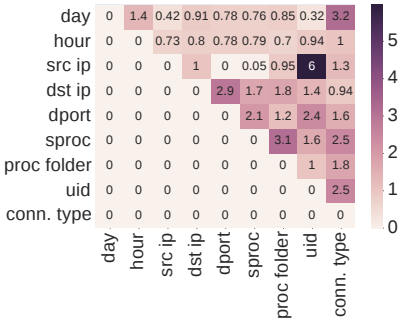


Figure 5: Pairwise weights learned for P2I events.

Table 4: Detected abnormal events examples.

Data	Abnormal event
P2P	..., src proc: bash, src folder: /home/, ...
P2P	..., uid: 9 (some main user), hour: 1, ...
P2I	..., proc: ssh, dst port: 80, ...

highlight the pairs of entities that have the particular low comparability score). In the first event, the interaction between process bash and its folder is irregular and results in small likelihood; in the second event, the abnormality is caused by a main user (who usually active during the work hour) involved in the event on 1 a.m.; in the third example, the process ssh connects to an unexpected port 80 and thus raising the alarm.

4.5 Results for different noise distributions

Table 5 shows performances under different choices of noise distribution. Results shown are collected from test set with

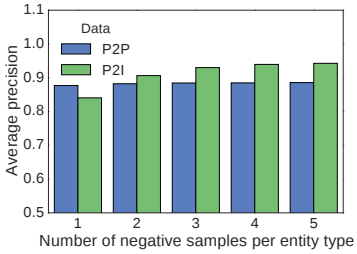


Figure 6: Performance versus number of negative samples drawn per entity type.

$c = 1$ (for $c = 2, 3$, the results are similar thus not shown), and using the same number of training events.

First we compare the "context-independent" noise distribution (first row) and the proposed "context-dependent" noise distribution (third row), clearly the "context-dependent" one performs significantly better. This confirms that by using the proposed "context-dependent" noise distribution, the APE model can learn much more effectively given the same amount of resources.

We also compare the importance of the approximated noise probability term $\log kP_n(e)$ in Eq. 6. Simply ignore the term by setting it to zero (second row) (as similarity used in [Mikolov *et al.*, 2013a; 2013b]) results in much worse performances compared to our proposed approximated one.

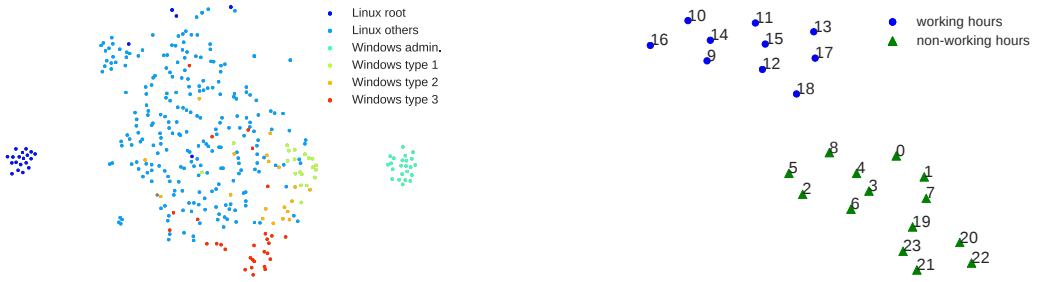
Table 5: Average precision under different choice of noise distributions.

Noise distribution	P2P	P2I
Context-independent	0.8463	0.7534
Context-dependent, $\log kP_n(e) = 0$	0.8176	0.7868
Context-dependent, $\log kP_n(e) =$ appx	0.8845	0.9383

Figure 6 shows the detection performance versus the number of negative samples drawn per entity type. As we can see, it only requires a reasonable number of negative samples to learn well, though adding more negative samples may marginally improve performances.

4.6 A case study for entity embedding

In order to see if the learned embedding is meaningful, we use t-sne [Van der Maaten and Hinton, 2008] to find 2d coordinates of the original entity embeddings. Figure 7a shows the embedding of users in P2P data. We color each user



(a) User embeddings. (b) Hour embeddings.

Figure 7: 2d visualization of some entity embeddings.

according to the user type. We find that, in the embedding space, similar types of users are clustered together, indicating they play the same role [Chen *et al.*, 2016]; and in particular, root users are grouped together and far away from other types of users, reflecting that root users behave very different from other users. Figure 7b shows the embedding of hours in P2I data. Although not knowing a priori, the APE model clearly learns the separations of working hours and non-working hours.

Knowing the types of users and differences among hours can be important for detecting abnormal events. The entity embedding learned by the APE model suggests it can distinguish the semantics/similarities of different entities, thus can help better detect anomalies.

5 Related Work

5.1 Anomaly Detection

There are many literatures for anomaly detection, a good summary of the anomaly detection methods can be found in [Chandola *et al.*, 2009]. However, most of those work focuses on either numerical data type or supervised settings.

As for unsupervised categorical anomaly detection, recent work includes [Das and Schneider, 2007; Das *et al.*, 2008; Akoglu *et al.*, 2012]. Most of these methods try to model the regular patterns behind data, and produce abnormal score of data according to some heuristics, such as the compression bits for an event [Akoglu *et al.*, 2012].

There is some work on applying graph mining methods for anomaly detection in graph [Tong *et al.*, 2008; Akoglu *et al.*, 2014]. However, our setting is different in the sense that, as shown in Section 2, when treating categorical events as a network, it is a heterogeneous network [Sun and Han, 2013].

There is also some work on anomaly detection for heterogeneous data [Ren *et al.*, 2009; Das *et al.*, 2010], However, most of them are not suitable for event data due to the lack of distance measure among data points. For example, [Das *et al.*, 2010] uses LCS to measure distance between two sequences, but will not work for two events.

5.2 Embedding Methods

Embedding methods are widely studied in graph/network setting [Belkin and Niyogi, 2001; Tang *et al.*, 2015]. And more recently, there is some work [Bengio *et al.*, 2003; Mikolov *et al.*, 2013a; 2013b] on natural language processing, which tries to embed words into some high dimensional space.

Our work also explores the embedding methods, however, there are some fundamental differences between our method and other embedding methods. Firstly, many of those embedding methods aim to embed pairwise interactions, but they only consider one type of entities. For pairwise interaction of different types of entities, we provide a weighted scheme for distinguishing their importance. Secondly, existing embedding methods cannot be directly applied to predicting abnormal score.

There is some work [Agovic *et al.*, 2009] applying graph embedding methods for anomaly detection in numerical data where the distance among data points are easy to compute. However, as far as we know, embedding methods have not explored in anomaly detection applications on categorical event data.

6 Conclusions

In this paper, we tackle a challenging problem of anomaly detection on heterogeneous categorical event data. Different from previous work that heavily relies on heuristics, we propose a principled and unified model APE that directly learns the likelihood of events. The model is instantiated by weighted pairwise interactions among entities that are quantified based on entity embeddings. Using Noise-Contrastive Estimation with "context-dependent" noise distribution, our model can be learned efficiently regardless of the exponentially large event space. Experimental results on real enterprise surveillance data show that our method can accurately detect abnormal events compared to other state-of-the-art abnormal detection techniques.

As for the future work, it is interesting to consider the temporal correlations among multiple events instead of treating them independently, as many intrusions/attacks can involve a series of events.

Acknowledgement

We would like to thank Zhichun Li, Haifeng Chen, Guofei Jiang, and Jiawei Han for some helpful discussions. This work is partially supported by NSF CAREER #1453800, Northeastern TIER 1, and Yahoo! ACE Award.

References

[Agovic *et al.*, 2009] Amrudin Agovic, Arindam Banerjee, Auroop Ganguly, and Vladimir Protopopescu. Anomaly detection using manifold embedding and its applications in transportation corridors. *Intelligent Data Analysis*, 13(3):435–455, 2009.

[Akoglu *et al.*, 2012] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. Fast and reliable anomaly detection in categorical data. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 415–424. ACM, 2012.

[Akoglu *et al.*, 2014] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.

[Belkin and Niyogi, 2001] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14, pages 585–591, 2001.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Chandola *et al.*, 2009] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys*, 41(3):15, 2009.

[Chen *et al.*, 2016] Ting Chen, Lu-An Tang, Yizhou Sun, Zhengzhang Chen, Haifeng Chen, and Guofei Jiang. Integrating community and role detection in information networks. *SIAM International Conference on Data Mining*, 2016.

[Das and Schneider, 2007] Kaustav Das and Jeff Schneider. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 220–229. ACM, 2007.

[Das *et al.*, 2008] Kaustav Das, Jeff Schneider, and Daniel B Neill. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–176. ACM, 2008.

[Das *et al.*, 2010] Santanu Das, Bryan L Matthews, Ashok N Srivastava, and Nikunj C Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 47–56. ACM, 2010.

[Davis and Goadrich, 2006] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[Gutmann and Hyvärinen, 2010] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Mnih and Kavukcuoglu, 2013] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

[Mnih and Teh, 2012] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[Ren *et al.*, 2009] Jiadong Ren, Qunhui Wu, Jia Zhang, and Changzhen Hu. Efficient outlier detection algorithm for heterogeneous data streams. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pages 259–264. IEEE, 2009.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *2010 IEEE 10th International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[Sun and Han, 2012] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012.

[Sun and Han, 2013] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 14(2):20–28, 2013.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. 2015.

[Tong *et al.*, 2008] Hanghang Tong, Yasushi Sakurai, Tina Eliassi-Rad, and Christos Faloutsos. Fast mining of complex time-stamped events. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 759–768. ACM, 2008.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[Zhang *et al.*, 2015] Kai Zhang, Qiaojun Wang, Zhengzhang Chen, Ivan Marsic, Vipin Kumar, Guofei Jiang, and Jie Zhang. From categorical to numerical: Multiple transitive distance learning and embedding. *SIAM International Conference on Data Mining*, 2015.