BAnDIT: Business Process Anomaly Detection in Transactions

Nico Rudolf[1], Kristof Böhmer[1], and Maria Leitner[1,2][0000−0003−1371−5446]

[1] University of Vienna, Faculty of Computer Science, Research Group Workflow Systems and Technology, Währinger Straße 29, Vienna, Austria
kristof.boehmer@univie.ac.at
[2] University of Regensburg, Faculty of Informatics and Data Science, Chair of Artificial Intelligence in IT Security, Regensburg, Germany
maria.leitner@ur.de

Abstract. Business process anomaly detection enables the prevention of misuse and failures. Existing approaches focus on detecting anomalies in control, temporal, and resource behavior of individual instances, neglecting the communication of multiple instances in choreographies. Consequently, anomaly detection capabilities are limited. This study presents a novel neural network-based approach to detect anomalies in distributed business processes. Unlike existing methods, our solution considers message data exchanged during process transactions. Allowing the generation of detection profiles incorporating the relationship between multiple instances, related services, and exchanged data to detect point and contextual anomalies during process runtime. To validate the proposed solution, it is demonstrated with a prototype implementation and validated with a use case from the ecommerce domain. Future work aims to further improve the deep learning approach, to enhance detection performance.

Keywords: Anomaly detection · Business processes · Service-oriented systems · Deep learning · Security

1 Introduction

Process anomaly detection is a crucial methodology that enables the identification of anomalous behavior within business processes. Anomalies can signify various issues such as fraud, misuse, and errors with the potential to result in process failure. In today's process-driven organizations, where the reliability and robustness of business process executions form the backbone of their operations, the need for effective anomaly detection becomes apparent [21, 12]. Existing approaches to anomaly detection often analyze each process instance's control, temporal, and resource behavior independently [3, 11, 19], neglecting the communication between multiple instances. This limitation restricts their applicability in use cases, where processes collaborate and communicate extensively.

Consider, for example, a straightforward online shopping scenario, like buying shoes, involving collaboration across delivery, procurement, warehouse, and accounting processes (cf. Figure 1). Malicious actions can be concealed by orchestrating attacks over multiple process instances. For instance, an attacker could alter the communicated article count to the delivery process, while leaving the accounting process unchanged, allowing receipt of more articles than paid for. Conventional anomaly detection methods concentrating on individual instances or control flow would miss such attacks, leaving process communication vulnerabilities unnoticed. The approach for solving this issue, lies in analyzing communication and data flow in multi-instance choreographies. Examining inter-process transactions and their data exchanges uncovers this behavior.

Due to communication's dynamic nature in distributed systems, concept drift in process choreography is frequent. These deviations shouldn't be treated as anomalies but as an evolving norm. This paper addresses these challenges via an innovative deep learning anomaly detection approach. It analyzes message flow and content to spot point and contextual anomalies during live process executions. Point anomalies signify unexpected behavior within a single process event, such as an unfamiliar article request. Contextual anomalies, stem from manipulated context misaligning with the subsequent process choreography, like an attacker altering delivery item counts.

The approach combines two neural networks for unsupervised anomaly detection. With these, a two-step approach offers two settings: one involving a classifier network for message transformation and the other utilizing an autoencoder network. The proposed solution is evaluated concerning its applicability and performance in an experimental setting. For evaluation, we used two data sets and describe the synthetic one from micro-service online shop system simulation. The rest of the paper is structured as follows: Section 2 revisits preliminary background on service-oriented business processes. Section 3 describes the two-step profile-based anomaly detection approach. Section 4 valides the approach with a use case from the e-commerce domain. Section 5 summarizes relevant related works and Section 6 concludes the paper.

2 Preliminary

Many scenarios where processes are collaborating and communicating in a choreography are given by service-oriented systems. Each service s provides a certain set of operations $\mathbb{O}_s$. An operation o fulfills a specific task in the system and is available to other services in the system via an endpoint u. In the example of an online shop system (cf. Figure 1), there could be one service responsible for managing the inventory of the shop. One operation this service may provide is requesting the availability of products. Formally this behavior is defined as:

Definition 1. *(Service, Operation, Endpoint) Let $\mathbb{S}$ be the finite set of services within a system. A service may provide one or multiple operations. Let $\mathbb{O}$ be the finite set of all possible operations, then $\mathbb{O}_s$ are all operations for the service $s \in \mathbb{S}$. Let $\mathbb{U}$ be the finite set of all endpoints, then $\mathbb{U}_s$ is the set of all endpoints for service $s \in \mathbb{S}$. There exists a bijective function $\alpha : \mathbb{U} \to \mathbb{O}$, mapping each endpoint to a specific operation, such that $o = \alpha(u)$ (cf. [22]).*

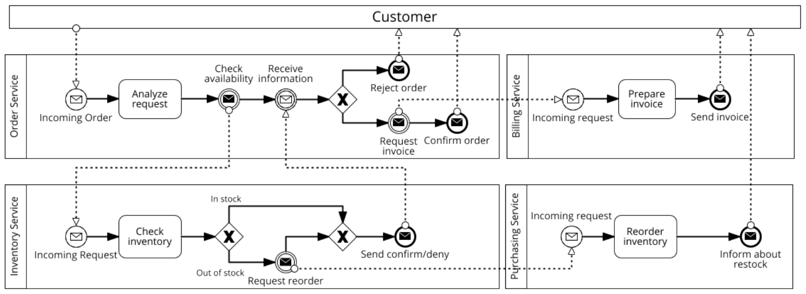

Fig. 1: Illustration of an online shop system designed as a service-oriented system.

Each operation transfers information in the form of a message m. If we look at the example of an online shop, a message for the operation that is requesting the availability of products could contain products, their quantity and other relevant information. Formally, message m is defined as:

Definition 2. *(Message) Let $\mathbb{T}$ be the finite set of all possible terms included in a message, then a message is defined as a tuple of terms, like $m_o = (t_0, t_1, ..., t_n), n \in \mathbb{N}$ and $m_o \in \mathbb{M}$ with $\mathbb{M}$ as the set of all possible messages. The message of an operation can be empty, then $m_o = \emptyset$. For a message there exists a surjective function, mapping to an operation such that $\beta : \mathbb{M} \to \mathbb{O}$.*

Some of the steps in the process instances illustrated in Figure 1 require transactions (i.e. communication) with other services. This particular type of process step triggers an invocation of an operation o of another service in the system. We define this class of process steps to be **message events** e, as together with the invocation of an operation, they are transmitting a message to another service. The formal definition of a message event goes as follows:

Definition 3. *(Message Event) Let $\mathbb{E}$ be the finite set of all possible message events. A message event e corresponds to the invocation of an operation $o \in \mathbb{O}$. As there exists a bijective function $\alpha : \mathbb{U} \to \mathbb{O}$ each message event can be characterized by an endpoint $u \in \mathbb{U}$. Defined by the function β a message event also has a message $m \in \mathbb{M}$ corresponding to a particular message event. As a shorthand, the messages of a particular message event is denoted as $m(e)$.*

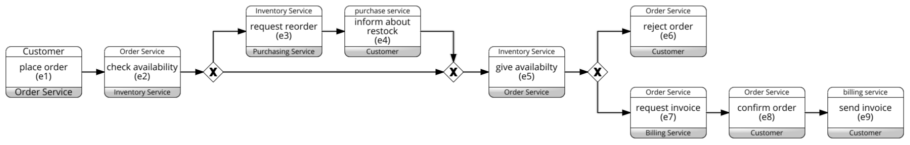

Fig. 2: Resulting process choreography of the online shop system.

Focusing solely on message events, the online shop example generates a process choreography, cf. Figure 2. These message events are collected into a *message event log L*, defined similarly to [1].
In order to provide more fine-grained monitoring of the system, we introduce the concept of a profile p, which is responsible for monitoring only a specified part of the system, i.e. area of interest. When it comes to drawing the boundaries regarding which services are included in the profile, technologies from distributed tracing can be utilized as proposed in [17]. A pattern r can be used on the endpoints of all operations $\mathbb{U}_s$ provided by a particular service. All message events with an endpoint matching the defined pattern are considered by the profile, others are ignored. This leads to the following formal definition:

Definition 4. *(Profile, Compound, Pattern) A profile p is defined to monitor an area of interest within that system. That is a certain compound of services K such that $K \subseteq \mathbb{S}$. Let $\mathbb{R}$ be the set of specified regular expression patterns, then the compound of services for the pattern r is K_r such that two different services $s_1 \in K_r$ and $s_2 \in K_r$ both provide only operations with an endpoint matching that pattern $\delta(\mathbb{U}_{s_1}, r) = \delta(\mathbb{U}_{s_2}, r)$, with δ being a function evaluating the pattern over an endpoint.*

Based on historic process cases a model will be trained to abstract a compounds normal behavior. Given a novel process execution with the head of a trace $hd^k(\hat{c})$ to message event at step k, the model will generate the next probable message event $\hat{c}(k+1)$. In case the predicted message event deviates significantly from the observation of $\hat{c}(k+1)$ the event will be detected as an anomaly.
Types of anomalies can be grouped into two major categories: (1) Sequence anomalies and (2) content anomalies. Note, that these types are not directly corresponding to the distinction between contextual and point anomalies. Content anomalies can be contextual as well as point anomalies, whereas sequence anomalies are always contextual. Sequence anomalies involve skipping, adding and swapping message events. Content anomalies modify message content by skipping, adding or swapping terms.

3 Two-step Profile-based Anomaly Detection

3.1 Two-step Deep Learning Approach

The proposed deep-learning approach for anomaly detection in service-oriented business processes consists of two separate neural networks. The first network is responsible for encoding individual message events in a trace, into an intermediate state. The second network is a recurrent neural network (RNN) that is trained to predict the expected sequence of encoded message events.
There are two settings explored for evaluating traces. In the first setting (encoder-setting), an autoencoder network is used for encoding individual message events. This network is called *message event encoder*. It projects message events into a lower-dimensional latent space. The encoder network processes the input message event and encodes it into a fixed-length vector, which is then passed to the

decoder. The decoder generates the output based on this fixed-length vector. The autoencoder is trained to minimize the reconstruction loss, which measures the difference between the input and predicted vectors. After training, only the encoder network is used to encode a message event for every step in the sequence into a latent space vector. The sequence RNN takes the encoded message events until a particular step in the sequence, to predict the next probable message events latent space vector. Using this prediction, the distance to the observed message event can be computed by the loss function.
In the second setting (classifier-setting), the message event encoder is replaced by the *message event classifier*. This classifier network maps message events to a specific type. The input and output sequences are of potentially different lengths. The sequence RNN predicts the next probable message event given a starting sequence. Similar to the encoder-setting, the model is trained to minimize the distance between the predicted and observed message events.
Each of the settings focuses on a particular aspect of the evaluation of process executions. The classifier-setting gives more attention to the control flow of the process execution, while the encoder-setting focuses on the content of exchanged messages. That is, because the reconstruct from latent space can be compared to an observed message event, to spot specific deviations in the message content.
The challenge of emerging concept drift in the monitored system can be addressed by retraining the model iteratively, to gradually adapt to changes in the process over time. This can be done by training the pre-trained network with new data, that reflects the new normal behavior of the process.
The autoencoder is built as a sequence-to-sequence network [24]. All networks use gated recurrent units (GRUs) as recurrent units. These show similar performance to its close alternative the LSTM, while being relatively more efficient [7]. The decoder network of the message event encoder involves the use of an attention mechanism, allowing the network to focus on different parts of the sequence [2]. Models are optimized using negative log-likelihood (NLL) as a loss function for classification problems and cosine embedding loss for latent space comparison.

3.2 Profile Development

The trained neural networks model the system's normal behavior, creating a profile for the process choreography of the involved instances. Focusing on specific interaction parts allows finer monitoring of the whole system. Applying profiles to individual actors is particularly used in fraud and intrusion detection, which involves multiple participants [6]. This concept is uncommon in anomaly detection for business processes [11, 14, 19, 21].
Looking at the example of an attacker altering the number of ordered items, the critical section of the entire system can be limited to the ordering and billing service. Therefore, billing and accounting service build a separate compound of services $K_{b,a}$. For this, a pattern $r_{b,a}$ has to be defined such that only relevant operations match that pattern, resulting in $K_{b,a} = (s_{billing}, s_{accounting})$. For monitoring, this results in a reduced process choreography, only involving

message events emitted from the operations included by pattern $r_{b,a}$, hence reducing the complexity of process traces.
When training the neural network model for the service compound $K_{b,a}$, the reduced set of message events also results in a reduction of dimensionality for the input of the model. That is, as we can expect the set of all possible terms in a message $\mathbb{T}_{b,a}$ to be a proper subset of $\mathbb{T}$ like $\mathbb{T}_{b,a} \subsetneq \mathbb{T}$. Likewise, the number of possible types of message events can be expected to be smaller for the reduced compound of services $K_{b,a}$ like $k_{b,a} < k$, which influences the input and output dimension of the message event classifier.

3.3 Threshold Heuristic and Decision Boundary

Distinguishing between normal and anomalous message events and process executions relies on a decision boundary typically established using an anomaly score. As in [18], this score can be the reconstruction loss, measuring the prediction-observation distance. In the encoder-setting, a threshold heuristic sets this boundary. Introducing a dynamic threshold T involves using a flexible function for precise value assignment. Such adaptability is crucial due to varying anomaly score distributions across data sets, influenced by factors like message length or content diversity. This approach alignes with [11, 18, 19].
For anomaly score values we can assume a skewed normal distribution, as normal samples can be expected to occur more frequently [6]. Most anomaly scores will thus be relatively low, starting at zero as a lower boundary. Samples differing more than a chosen multiplicative of the standard deviation from the mean are then considered anomalous. A common factor is choosing three times the standard deviation [6]. Hence, the threshold can be computed as $T = \mu + 3\sigma$.
In the classifier-setting, or for point anomaly detection, where sequences are formed using probability distributions at each step, an alternative decision boundary approach is utilized. This involves examining the top n outputs of the probability distribution for each step, accumulating their probabilities to surpass a threshold $p_T \in [0; 1]$, specifically $p_T > \sum_{i=0}^{n} p_i$. These outputs are then treated as candidates. If the observed output isn't among the candidates, an anomaly is signaled. Through experimentation, a threshold of 0.9 yielded the best results after evaluating various parameter values. This dynamic candidate sampling method aligns with [9].

4 Evaluation

The evaluation combines a) real-life logs from the financial domain; b) artificial logs from a publicly available distributed online shop implementation[3]; and c) artificially injected anomalies. Enabling to assess detection performance even though publicly available execution logs hardly cover data flow. Therefore, a baseline algorithm is introduced to provide a comparison point.

[3] https://github.com/nico-ru/BAnDIT

4.1 Data sets

FINANCE: The data set was conducted over a time frame of several months from individual financial institutions. Anomalous data requests and process cases are produced with the consultation of expert knowledge. Point anomalies of each type are induced in three data sets from financial institutions. Data set one **(point-1)** consists of a total of 3868 samples. Data set two **(point-2)** consists of a total of 2570 samples. And data set three **(point-3)** consists of a total of 5716 samples. Contextual anomalies were induced in the data set **(sequence-1)**, which consists of a total of 101 samples (one sample constitutes of one day of requests). The use case is discussed in [23].
MICRO: The *MICRO* data set is generated by the prototype implementation of a micro-service online shop system. The **(micro-1)** data set is used for inducing point and contextual anomalies. It is generated by 1000 executions of the implemented process choreography. In total, 4979 message events are produced by this simulation. The message content is generated in a JSON serialization and transformed into a flat dotlist format to better fit the message definition, cf. Section 2.
Anomalies are inserted randomized in both data sets with a ratio of 1-2% for point anomaly detection and 5-10% for contextual anomaly detection. The rates chosen for this evaluation align with the assumption of anomalies being rare as opposed to normal samples, cf. [4, 18].

4.2 Baseline Solution

To contextualize the results from the proposed anomaly detection, a baseline algorithm is introduced to address the same anomaly detection task. For point anomaly detection, skip-grams are generated from message content. These skip-grams exclude one term at a time. The probabilities for each term's absence is calculated, by counting all possible missing terms. Message occurrence probability is computed by multiplying term probabilities for every term in the message. Contextual anomaly detection employs the same technique, but with skip-grams of activities across cases instead of terms. The skip-gram and probability algorithms are available online[3]. In both cases, anomalies arise when a sequence's probability falls outside one standard deviation from the data set's probability distribution. Context anomaly results solely consider activity order, not message event content. This approach for the baseline solution draws inspiration from [3, 15, 16].

4.3 Evaluation Results

As detecting anomalies poses a binary classification problem, we use commonly accepted metrics for classification to report the performance of the proposed solution [10]. In particular, these are precision, recall, and F-scores. This aligns with [5], where all metrics are described. We consider the detection of all present anomalies as the most relevant, as any anomaly can cause serious errors. The results reported in Table 1 show the evaluation metrics for the individual data sets as an average over all anomaly types. For training the neural networks the data was split into train (75%), validation (5%) and test (20%) data.

Table 1: Evaluation results for the proposed solution as well as baseline solution.

	Point Anomalies				Context Anomalies (classifier-setting)		Context Anomalies (encoder-setting)
	point-1	point-2	point-3	micro-1	sequence-1	micro-1	sequence-1
Precision	1.00	0.75	0.83	**0.17**	0.83	0.85	**0.22**
Recall	0.95	0.90	0.98	0.80	0.91	0.85	0.54
F-score	0.97	0.82	0.90	0.28	0.87	0.85	0.31
F2-score	**0.96**	**0.87**	**0.94**	**0.36**	**0.91**	**0.85**	0.42
Baseline Results							
Precision	0.60	0.06	0.57	0.01	0.15	0.87	–
Recall	0.92	0.60	0.85	0.17	0.23	0.72	–
F-score	0.75	0.11	0.69	0.02	0.18	0.79	–
F2-score	0.84	0.21	0.78	0.04	0.21	0.75	–

Interpretation Results show that the proposed model effectively learns the overall distribution of message events and successfully detects rare anomalies. While precision scores of the proposed solution may be sub-optimal in some cases, this can be attributed to the low number of real anomalous samples, making a single false positive significantly impact precision. Overall, the proposed solution outperforms the baseline, particularly with notable improvements in the F-scores, as observed in the point-2 data set.

Regarding contextual anomaly detection in the classifier-setting, similar performance is achieved compared to point anomaly detection. Performance variations can be attributed to variances in the accuracy of the message event classifier and the complexity of the monitored process.

Compared to related work on anomaly detection using deep neural networks [11, 18], the proposed approach demonstrates comparable performance. However, direct comparison is possible only in a limited way due to the absence of message content evaluation in related research.

Detecting anomalies in the sequence of encoded message events poses the most challenging problem. Consequently, the results for contextual anomaly detection in the encoder-setting are less satisfying. Accurately predicting the next encoded message event proves difficult for the sequence RNN. This limitation may stem from a sub-optimal choice of loss function, as RNNs are generally capable of learning sequential patterns. It is likely that quantifying the distance between encoded message events is the main issue. Various loss functions were tested, with the cosine embedding loss yielding the best results.

5 Related Work

The general problem of anomaly detection is a well-explored topic; For an extensive discussion of this domain, we refer to a survey by Chandola et al. [6]. We and existing work make the assumption that anomalous samples occur far less frequently than normal ones, enabling training on uncleaned data. With this we align with other publications [6, 8, 18].

In the area of business processes, anomaly detection approaches have been proposed in the control flow of processes [3–5, 11, 18], in the temporal behavior of process executions [14, 21], or also in activity attributes of processes [11, 19]. While the detection performance of some of the proposed solutions is quite sophisticated, none of them provides the possibility of detecting anomalies in the exchanged message data of multiple process instances. Merely [11, 19] are considering the data-flow of single process instances, by feeding activity attributes into the model abstracting the normal behavior of the process. Both methods are extensible towards a higher number of activity attributes, however, the highly versatile and complex message data can not be represented by either of the solutions. Predictive monitoring techniques are highly related to anomaly detection, as they also involve building a model of the normal behavior of a process [13, 20]. However, the proposed solutions do not consider distributed process choreographies nor incorporate message data.

6 Conclusion

This paper introduces a novel method for identifying anomalies in distributed process instances engaged in a process choreography. The focus is on analyzing message data exchanged between these instances. The approach involves defining message events, creating process logs, and using a neural network-based process model. This model employs a two-step process to detect both point anomalies and contextual anomalies. By abstracting service compounds into profiles, the approach allows for detailed monitoring. The method's performance was evaluated on a real financial data and a synthetic data set of an e-commerce shop system. In the encoder-setting, where the complexity was considerably higher than in the decoder-setting and the loss function potentially unfitting, the results for contextual anomalies were less convincing. Overall, results showed effective abstraction of communication between process instances and adaptability to dynamic changes. For future work, we aim to refine the approach to enhance contextual anomaly detection and explore other anomaly detection techniques for process choreography message data. These efforts promise to advance anomaly detection methods and provide insights into analyzing message data in process choreography.

References

1. van der Aalst, W.: Process Mining: Data Science in Action, chap. Chapter 5: Getting the Data. Springer Berlin, Heidelberg, 2 edn. (2016)

2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2014)
3. Böhmer, K., Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: OTM 2016 Conferences. pp. 80–98. Springer (2016)
4. Böhmer, K., Rinderle-Ma, S.: Multi instance anomaly detection in business process executions. In: BPM 2017. pp. 77–93. Springer (2017)
5. Böhmer, K., Rinderle-Ma, S.: Association rules for anomaly detection and root cause analysis in process executions. In: CAiSE 2018. pp. 3–18. Springer (2018)
6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3) (Jul 2009)
7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)
8. Eskin, E.: Anomaly detection over noisy data using learned probability distributions (2000)
9. Holtzman, A., Buys, J., Forbes, M., Choi, Y.: The curious case of neural text degeneration. CoRR **abs/1904.09751** (2019), http://arxiv.org/abs/1904.09751
10. Hossin, M., Sulaiman, M.N.: A review on evaluation metrics for data classification evaluations. IJDKP **5**(2), 1 (2015)
11. Huo, S.e.a.: Graph autoencoders for business process anomaly detection. In: 19th International Conference, BPM 2021. pp. 417–433. Springer (2021)
12. Leitner, M., Rinderle-Ma, S.: A systematic review on security in process-aware information systems – constitution, challenges, and future directions. Information and Software Technology **56**(3), 273–293 (2014). https://doi.org/https://doi.org/10.1016/j.infsof.2013.12.004
13. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: CAiSE 2014. pp. 457–472. Springer (2014)
14. Mavroudopoulos, I., Gounaris, A.: Detecting temporal anomalies in business processes using distance-based methods. In: Discovery Science. pp. 615–629. Springer (2020)
15. Meng, W., Liu, Y., Zhang, S., Pei, D., Dong, H., Song, L., Luo, X.: Device-agnostic log anomaly classification with partial labels. In: IWQoS 2018. pp. 1–6 (2018)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
17. Nedelkoski, S., Cardoso, J.S., Kao, O.: Anomaly detection and classification using distributed tracing and deep learning. In: CCGRID 2019. pp. 241–250. IEEE (2019)
18. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Machine Learning **107**(11), 1875–1893 (apr 2018)
19. Nolle, T., Seeliger, A., Mühlhäuser, M.: Binet: multivariate business process anomaly detection using deep learning. In: BPM 2018. pp. 271–287. Springer (2018)
20. Pauwels, S., Calders, T.: Incremental predictive process monitoring: The next activity case. In: BPM 2021. pp. 123–140. Springer (2021)
21. Rogge-Solti, A., Kasneci, G.: Temporal anomaly detection in business processes. In: BPM 2014. pp. 234–249. Springer (2014)
22. Rud, D., Schmietendorf, A., Dumke, R.: R.: Product metrics for service-oriented infrastructures. In: IWSM/MetriKon 2006 (01 2006)
23. Rudolf, N.: Profile-based Anomaly Detection in Service Oriented Business Processes. masterthesis, University of Vienna (2023)
24. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. CoRR **abs/1409.3215** (2014), http://arxiv.org/abs/1409.3215