Latent representations of transaction network graphs in continuous vector spaces as features for money laundering detection

Dominik Wagner[1]

Abstract: This paper explores the construction of network graphs from a large bank transaction dataset and draws from findings in language modeling and unsupervised learning to transform these graphs into multidimensional vector representations. Such latent representations encode relationships and community structures within the transaction network. Three classifiers with varying complexity are trained on these latent representations to detect suspicious behavior with respect to money laundering. The specific challenges accompanying highly imbalanced classes are discussed as well and two strategies to overcome these challenges are compared.

Keywords: feature learning; graph embeddings; latent representations; DeepWalk; anti-money laundering; language modeling; class imbalances; SMOTE; machine learning; support vector machine; naive bayes; multilayer perceptron

1 Introduction

The process of money laundering can be divided into three stages: placement, layering and integration [Su01, p. 145]. During the placement stage, illegally obtained funds (mostly cash) are channeled into the financial system. The layering stage refers to the removal of traces and the distribution of those illegally obtained funds. Numerous transactions are carried out in this stage, often involving offshore-accounts and complex financial instruments. In the integration stage, the funds re-enter the legitimate economy, e. g. through the purchase of real estate or luxury assets. The three stages of the money laundering process show that a typical money laundering operation includes multiple transactions, which are transferred by a group of entities trough a multitude of channels. Thereby, secret collusion takes place between the entities involved. The Financial Action Task Force (on Money Laundering) also names networks as one of the key characteristics of professional money laundering organizations. Networks are described as a collection of associates or contacts working together to facilitate money laundering schemes [Fi18, p. 13]. As money laundering involves groups of collaborating individuals, signs for money laundering may only be apparent when the collective behavior of these groups is examined.

[1] Technische Hochschule Nürnberg Georg Simon Ohm, Fakultät Informatik, Keßlerplatz 12, 90489 Nürnberg, Deutschland wagnerdo49883@th-nuernberg.de

2 Related Work

Considering the behavior of communities in a network to facilitate money laundering and fraud detection has produced promising results in several studies. Most of the related work, however, focuses on feature extraction with graph mining methods and metrics from Social Network Analysis (SNA). Furthermore, not all approaches are tested on real-world data.

For instance, Michalak and Korczak [MK11] propose a model for graph structure learning that can be trained on a previously annotated transaction graph and can be matched against a graph without annotated transactions. The model used in their method is parametrized using fuzzy numbers, which represent parameters of transactions. The authors perform their experiments with artificially-generated data.

Savage et al. [Sa17] generate network graphs from data provided by the Australian Transaction Reports and Analysis Centre (AUSTRAC). The authors apply a combination of network analysis and supervised learning (Support Vector Machine and Random Forest) to identify suspicious behavior indicative of money laundering activity. The study focuses on identifying small sets of interacting parties whose collective behavior is suspicious. Their model considers a range of demographic, network specific, transaction specific and time-dependent features derived from the extracted communities.

In [EH07], the authors develop three algorithms to uncover anomalies in graph structured data, particularly for fraud. The algorithms focus on different types of changes in the graph such as vertex or edge insertions. The algorithms are validated on both synthetic and real-world data. In [DF15] methods from SNA, such as centrality measures are used to investigate money laundering cases. Their algorithms are verified with random graph data.

However, to the author's knowledge, no bank transaction network graphs have been transformed with the DeepWalk algorithm [PAS14] so far. This work contributes by carrying out such transformations (Sect. 5) and by testing the effectiveness of the approach with three different supervised classification methods (Sect. 6). Most closely related is the work by Weber et al. [We18] who apply scalable Graph Convolutional Networks (GCN) [CMX18] to predict the degree of suspicion of a given target vertex in a transaction network and to identify other potential bad actors in the network via direct or indirect connections to vertices known to be suspicious. The graph structure is generated by a simulator based on transaction distributions and dynamics observed in real data.

3 Problem Definition

The data used in this work is a subset of real account transactions at a German retail bank. The majority of those entities are individual persons and a small portion are legal persons. The dataset contains 241 exemplary entities, which have been flagged as suspicious regarding money laundering. The suspicion refers to Suspicious Activity Reports (SARs) made by the bank. SARs are filed to the Financial Intelligence Unit after compliance officers have confirmed the suspicion in a multi-step process.

The data can be represented as an undirected graph $G(V, E)$ where V is a set of vertices and E is a set of edges between the vertices with $E \subseteq \{\{u, v\} \mid u, v \in V\}$. A vertex v_i constitutes either a bank client or a country described by a two-letter country code, as defined in ISO 3166-1 alpha-2 [In19]. V is further divided into two sets of vertices. The vertices V_M for which the correct labels are known and the vertices V_N whose labels have to be determined. The task is to label the vertices $Y_i \in V_N$ with one of the two labels $\mathcal{L} = \{L_0, L_1\}$ where L_0 represents unsuspicious entities and L_1 represents suspicious entities with respect to money laundering. y_i is the label of vertex Y_i. The classification task is discussed in Sect. 6.

The information about the dependence of the examples embedded in the structure of G is utilized to classify the vertices V_N. The goal in that regard is to build multidimensional representations $X_E \in \mathbb{R}^{|V| \times k}$ where k is the number of latent dimensions. The process of creating these multidimensional representations is discussed in Sect. 5.

4 Graph Construction

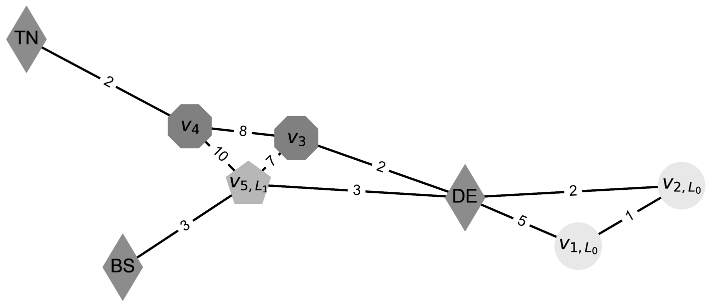

Fig. 1: Transaction graph example

Fig. 1 shows an example of a partially labeled transaction network graph $G(V, E)$ with $|V| = 8$ and $|E| = 10$. The vertices v_1 and v_2 are labeled as unsuspicious (L_0), while the vertex v_5 is labeled as suspicious (L_1) regarding money laundering. The vertices DE, BS, TN represent financial institutions in Germany (DE), The Bahamas (BS) and Tunisia (TN). The vertices v_3 and v_4 are not labeled yet. The edge attributes represent the number of transactions between the two vertices. The graph structure shows that the already suspicious vertex v_5 transacts with a country known to have strategic anti-money laundering deficiencies (BS) [Fi19]. The unlabeled entities v_3, v_4 interact primarily with v_5 and with each other. The vertex v_4 also transacts with a high risk country (TN). From these observations one could conclude that the vertices v_3 and v_4 should also be labeled as suspicious ($y_3 = L_1$ and $y_4 = L_1$).

In order to establish a transaction network graph $G(V, E)$ as illustrated in Fig. 1, the two counterparts u and v of each transaction are examined. In case the account is an internal account, i. e. it refers to an entity that is also available in the dataset, the unique client identifier is added as a vertex to the network, provided that the identifier in question does not already exist in the network. In case the account is an external account, the target or origin country of the transaction is extracted from the SWIFT Code. SWIFT (Society for Worldwide Interbank Financial Telecommunication) Codes are a standard format of Business Identifier Codes defined in ISO 9362 [SW19]. In this way, external accounts are aggregated based on the country they are located in. Each edge $e \in E$ represents at least one transaction between two connecting vertices u and v. The final graph contains $|V| = 1,128,921$ vertices and $|E| = 1,489,209$ edges.

5 Latent Representations

The graph structure is transformed into latent multidimensional representations by means of the DeepWalk algorithm [PAS14]. DeepWalk is an unsupervised learning algorithm that learns a latent representation of each vertex in a graph with the Skip-gram language model [Mi13a; Mi13b]. The graph is explored through a sequence of random walks. These random walk sequences are equivalent to sentences in language modeling. The learned vector representations encode relationships and community structures and can be used for various purposes such as classification, clustering or similarity search.

The Skip-gram model is built from a neural network with one hidden layer. Applied within the DeepWalk framework, the model receives a vertex v_i and generates the probability for all other vertices that a given vertex appears in the neighborhood or context of the input vertex v_i. The context is defined by the window c.

Learning a latent representation means learning a mapping function from vertex co-occurrences: $\Phi : v \in V \longmapsto \mathbb{R}^{|V| \times k}$. The mapping Φ is the latent representation of each vertex $v \in V$. The mapping Φ is represented by a $|V| \times k$ matrix, which is the target matrix X_E. The objective of DeepWalk is to find latent representations that are useful for predicting the surrounding vertices in a random walk sequence. The optimization problem is formalized as follows:

$$\min_{\Phi} -logPr(\{v_{i-c}, \dots, v_{i+c}\} \setminus v_i \mid \Phi(v_i)) \tag{1}$$

The Skip-gram model is used to update the representations according to the objective function Eq. (1). Skip-gram approximates the conditional probability in Eq. (1) by:

$$Pr(\{v_{i-c}, \dots, v_{i+c}\} \setminus v_i \mid \Phi(v_i)) = \prod_{\substack{j=i-c \\ j \neq i}}^{i+c} Pr(v_j \mid \Phi(v_i)) \tag{2}$$

5.1 Implementation

DeepWalk receives the following parameters: window size c, embedding size k, number of random walks per vertex γ and walk length t. It returns a matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times k}$. The algorithm iterates over all possible collocations of a given random walk sequence R_{v_i} that appear within the window c. R_{v_i} is a random walk sequence rooted at vertex v_i with length t. A window of length $2c + 1$ is slid over the random walk R_{v_i}, mapping the central vertex in the window v_f to its representation ($\Phi(v_f) \in \mathbb{R}^k$). Given the representation of v_f, the task is to maximize the probability of its neighbors in the random walk sequence. Such a posterior distribution is approximated using hierarchical softmax [MB05]. Hierarchical softmax is a computationally efficient approximation of the full softmax, where the graph vertices are assigned to the leaves of a balanced binary tree, turning the problem into maximizing the probability of a specific path in the tree hierarchy. The path to a neighboring vertex v_n of the central vertex v_f is identified by a sequence of tree nodes $(a_0, a_1, \ldots, a_{\lceil log|V| \rceil}$ where a_0 is the root of the tree and $a_{\lceil log|V| \rceil} = v_n)$. The conditional probability is then approximated by:

$$Pr(v_n \mid \Phi(v_f)) = \prod_{j=1}^{\lceil log|V| \rceil} Pr(a_j \mid \Phi(v_f)) \tag{3}$$

$p(a_j \mid \Phi(v_f))$ is now calculated by a binary classifier assigned to the parent of the node a_j:

$$Pr(a_j \mid \Phi(v_f) = 1/(1 + e^{-\Phi(v_f) \cdot \Psi(a_j)}) \tag{4}$$

$\Psi(a_j) \in \mathbb{R}^k$ is the vertex representation at the parent node of a_j. Stochastic gradient descent (SGD) is used to optimize the model parameter set ($\theta = \{\Phi, \Psi\}$). DeepWalk learns a latent

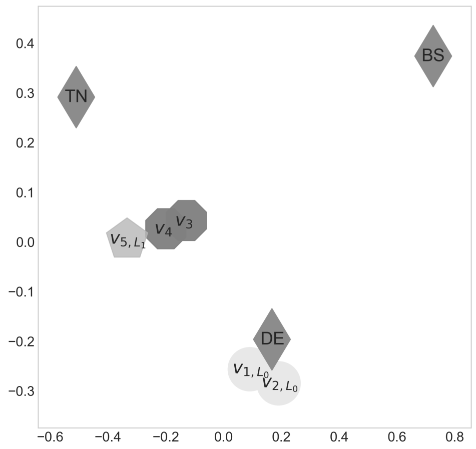

Fig. 2: Latent representation of the example graph

representation of transactions in $\mathbb{R}^k$. In Fig. 2 the method is used on the example graph from

Fig. 1 to generate a latent representation in $\mathbb{R}^2$. The community structure in the input graph corresponds well with the embedding. The unlabeled vertices v_3 and v_4 are located close to v_5, which has been already labeled as suspicious. The vertices labeled as unsuspicious v_1 and v_2 are grouped close to financial institutions in Germany (DE) and are relatively far away from v_3, v_4 and v_5. With standard classification methods applied on the latent representations, v_3 and v_4 would be more likely classified as suspicious (L_1).

The DeepWalk algorithm is trained on the full bank transaction dataset with $\gamma = 1000$ walks per vertex, walk length $t = 8$, window size $c = 2$, learning rate of the neural network $\alpha = 10^{-2}$ and $k \in \mathbb{R}^{20}$ hidden units (latent dimensions).

6 Classification

The learned representations are used to classify entities into the categories unsuspicious (L_0) and suspicious (L_1). Under the assumption that "suspicious" representations are sufficiently different from "unsuspicious" ones, supervised learning algorithms are used for this task. The class L_1 refers to Suspicious Activity Reports (SARs) made by the bank, while the class L_0 refers to hitherto inconspicuous parties.

The latent representations serve as inputs for three widely used classifiers with varying complexity: Naive Bayes (NB), Support Vector Machine (SVM) and Multilayer Perceptron (MLP). The standard variant of Naive Bayes, in which the likelihood of the features is assumed to be Gaussian, is applied here. SVMs are trained with both a linear and a Radial Basis Function (RBF) kernel. The optimal SVM hyperparameters are determined by evaluating the area under the ROC curve [Fa06; Sw01] in a threefold cross-validation on the training set, using the grid search technique. The kernel parameter γ for the RBF kernel is selected from the set $\gamma \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\} \subset \mathbb{R}_{>0}$. The linear SVM employs, the ℓ^2 penalty on a squared hinge loss. The penalty parameter of the error term C is selected from the set $C \in \{10^k \mid k = -2, -1, \ldots, 5\} \subset \mathbb{R}_{>0}$ for the linear and the RBF kernel. Several

Hyperparameter	Value
Optimization	Stochastic Gradient Descent
Learning Rate	$\alpha = 10^{-4}$
Learning Rate Decay	$d = 10^{-6}$
Momentum	Nesterov ($m = 0.9$)
Dropout	$p = 0.2$
Activation Function	leakyReLU $f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.3x, & \text{else} \end{cases}$

Tab. 1: MLP hyperparameters

architectures and hyperparameters have been considered to train the MLP. The best results were achieved with three hidden layers and 150 hidden units in each layer combined with the hyperparameters shown in Tab. 1. The output layer consists of a single neuron and employs a sigmoid activation function. The training samples are processed in mini-batches of 32 samples. The training procedure runs for 20 epochs, i. e. the number of complete passes through the training dataset is 20.

6.1 Class Imbalances and Synthetic Over-sampling

The dataset at hand is highly imbalanced, i. e. the classification categories are not at all equally represented. Vertex representations were obtainable for 223 of the 241 entities flagged as suspicious. For the remaining 18 entities, the data provided in the subset was insufficient to generate embeddings, i. e. no transactions were available for these entities. On the other hand, more than one million vertex representations encoding regular client behavior were generated.

To overcome this problem, two strategies are considered and compared afterwards. First, the majority class instances (i. e. unsuspicious entities) are under-sampled and the minority class instances are randomly duplicated until the desired share of suspicious samples to total samples is reached. Second, the majority class instances (i. e. unsuspicious entities) are under-sampled and the minority class instances are over-sampled with synthetic examples by means of the SMOTE algorithm [Bo11]. Both strategies use subsets from the 1,128,921 available latent representations. The processed training data consist of 12,000 "unsuspicious" records and 12,000 "suspicious" records. The test data consist of 3,000 "unsuspicious" and 45 "suspicious" records.

The second approach to overcome the class imbalance problem is inspired by the work of Xu et al. [Xu15], who use the SMOTE algorithm as an over-sampling method to produce balanced training sets from word embeddings. This work proceeds as follows: First, 12,000 samples are randomly drawn from the majority class ("unsuspicious"). Second, the SMOTE algorithm is used, to generate samples for the minority class ("suspicious"), until the training set is fully balanced. The algorithm attempts to over-sample the minority class by taking each minority class sample and generates synthetic examples from the k minority class nearest neighbors. The neighbor parameter is set to $k = 5$ here. The training pipeline has five

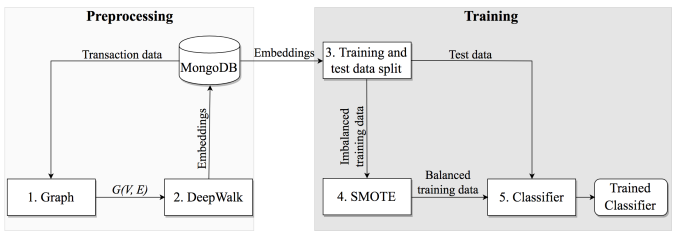

Fig. 3: Preprocessing and training pipeline with SMOTE

main steps, which are illustrated in Fig. 3. The first step constructs the transaction graph as discussed in Sect. 4. The second step performs the DeepWalk algorithm according to Sect. 5

and stores the latent representations in a database. The first two steps are preprocessing components, which have to be executed at least once before the actual training procedure can begin. The training procedure starts with the third step, which splits the vertex embeddings retrieved from the database into a training set and a test set. The fourth step applies the SMOTE algorithm on the minority class samples of the training data. The final step trains a classifier on the data.

Principal component analysis (PCA) is applied for the purpose of visualizing the learned 20-dimensional embeddings in two-dimensional space [TB99]. Fig. 4 shows the PCA projection of vertex representations from the majority class “unsuspicious” and the minority class “suspicious” without over-sampling in the left diagram. In contrast, the PCA projection of vertex representations from the majority class “unsuspicious” and the minority class “suspicious” with synthetic over-sampling of the minority class is shown in the right diagram. The newly generated samples are denoted by rhombuses. The right diagram shows better formed clusters for the “suspicious” class compared to the left diagram. Furthermore, the newly generated samples in the right diagram are usually close to the original samples, indicating that over-sampling with the SMOTE algorithm generally maintains the same distribution as the original samples.

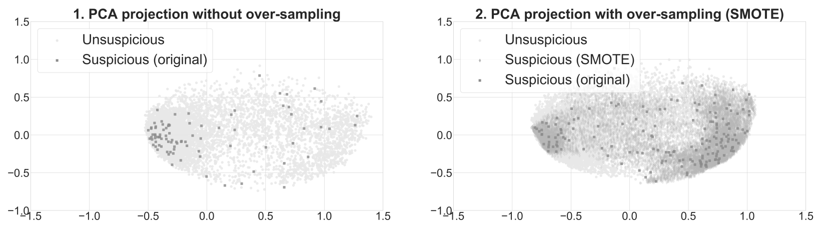

Fig. 4: PCA projection with and without over-sampling

6.2 Results

Each classifier is trained ten times on different subsets of the data. The samples from the majority class are randomly drawn prior to each new training procedure. The unprocessed minority class samples are shuffled and split into training data and test data prior to each new training procedure. Both over-sampling strategies are only applied on minority class samples of the training data. Hence, the 45 minority class samples used for performance evaluation are not synthetically generated or duplicated. Each input vector is scaled individually such that the ℓ^2-norm equals one. The performance measures are averaged over all ten training instances.

In the presence of imbalanced datasets, it is preferable to use Receiver Operating Characteristic (ROC) curves or other similar techniques over traditional performance measures such as accuracy [PF01]. ROC curves indicate the performance of binary classifiers at various false positive rates [Fa06; Sw01]. Fig. 5 illustrates the average ROC curves for both sampling

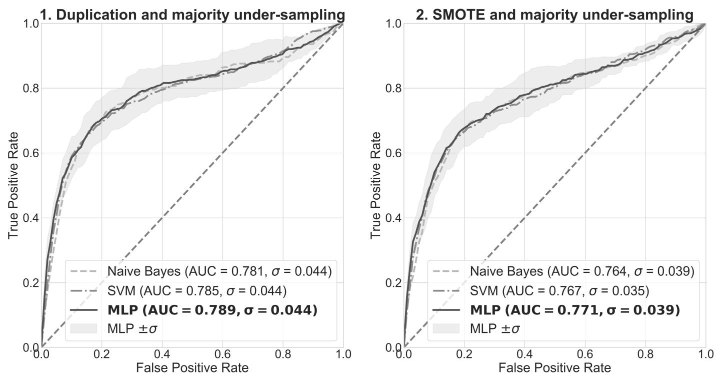

Fig. 5: Average ROC curves for different strategies and classifiers

strategies and the three classifiers. The dashed line $f(x) = x$ represents the scenario of randomly guessing the class. Despite the varying complexity of the three classifiers, the performance difference between them is relatively small for both strategies. The best overall results produces the MLP in combination with random duplication of the minority class. The average Area under the ROC Curve (AUC) equals 0.789. All three classifiers achieve better average performance, when majority under-sampling is applied in combination with random duplication (left diagram). Naive Bayes achieves an average $AUC = 0.781$ with random duplication of the minority class samples and an average $AUC = 0.764$ with synthetic over-sampling. The SVM classifier delivers an average $AUC = 0.785$ with random duplication and an average $AUC = 0.767$ with synthetic over-sampling.

However, the standard deviations of the AUC scores are lower for all three classifiers, when majority under-sampling is applied in combination with the SMOTE algorithm, indicating that this strategy produces more stable results compared to majority under-sampling combined with random duplication. The largest difference in the standard deviation of the AUC score shows the SVM classifier with $\sigma = 0.044$ for the strategy including random duplication (left diagram) and $\sigma = 0.035$ for the strategy including the SMOTE algorithm (right diagram). The shaded area in Fig. 5 illustrates the development of σ around the ROC curve of the MLP.

7 Discussion and Future Work

The approach presented in this paper has only been tested on a simple edge creation strategy, which was based on the existence of at least one transaction between two entities. Many other strategies are imaginable here. For instance, edges could be created only for certain types of transactions such as cash withdrawals, transactions to or from high risk countries or transactions made by politically exposed persons (PEP). Another strategy would be to create an edge only when certain transaction amount thresholds are exceeded.

The DeepWalk algorithm lacks generalizability. Whenever a new vertex is added, the model has to be re-trained in order to represent the new vertex as an embedding. The approach is therefore not suitable for real-time transaction monitoring systems. DeepWalk is also limited to the structure immediately around a vertex. This local focus implicitly ignores long-distance relationships and the learned representations might not uncover important global structural patterns. Furthermore, DeepWalk relies on stochastic gradient descent optimization, which can become stuck in local minima and consequently lead to poor results.

A comprehensive empirical study conducted by Khosla et al. [KAS19] considers 9 popular unsupervised network representation learning (UNRL) approaches including DeepWalk and Graph Convolutional Networks (GCN). The authors perform vertex classification and edge prediction tasks on 11 real-world datasets with varying structural properties. They find that there is no single method superior to the others and that the choice of a suitable method is determined by certain properties of the embedding methods, the task and structural properties of the underlying graph. Given these findings, other UNRL techniques should be applied to bank transaction network graphs in order to assess whether further improvements in the quality of the vertex embeddings are possible.

8 Conclusion

This paper demonstrates the transformation of transaction network graphs into real-valued latent representations with the DeepWalk algorithm. By doing so, an alternative way of representing bank transactions and the relationships between the entities involved is illustrated. Experiments on the latent representations generated from real bank transaction data show the effectiveness of the approach on a binary classification task.

Strong class imbalances pose an extra challenge to machine learning systems. The small number of “suspicious” records compared to the very large number of “unsuspicious” records requires additional feature engineering efforts. This paper explores two different strategies to cope with such imbalances. The more stable strategy for the data at hand is a combination of majority under-sampling and minority over-sampling with the SMOTE algorithm, while the combination of majority under-sampling and random duplication of the minority class achieves better results in terms of the average AUC score.

Acknowledgments

The results presented in this study are part of the development of a machine learning based prototype for money laundering detection during a 20 week internship, which undergraduate students at TH Nürnberg take in their fifth semester. The author would like to thank Wolfgang Füssel and Torsten Hoffmann for their support and numerous helpful remarks during this project. Thanks should also go to Prof. Dr. Jens Albrecht for the comprehensive feedback on this paper.

References

[Bo11] Bowyer, K. W.; Chawla, N. V.; Hall, L. O.; Kegelmeyer, W. P.: SMOTE: Synthetic Minority Over-sampling Technique. CoRR abs/1106.1813/, 2011, arXiv: 1106.1813, URL: http://arxiv.org/abs/1106.1813.

[CMX18] Chen, J.; Ma, T.; Xiao, C.: FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. CoRR abs/1801.10247/, 2018, arXiv: 1801.10247, URL: http://arxiv.org/abs/1801.10247.

[DF15] Dreżewski Rafałand Sepielak, J.; Filipkowski, W.: The Application of Social Network Analysis Algorithms in a System Supporting Money Laundering Detection. Inf. Sci. 295/C, pp. 18–32, Feb. 2015, ISSN: 0020-0255, URL: https://doi.org/10.1016/j.ins.2014.10.015.

[EH07] Eberle, W.; Holder, L.: Anomaly Detection in Data Represented As Graphs. Intell. Data Anal. 11/6, pp. 663–689, Dec. 2007, ISSN: 1088-467X, URL: http://dl.acm.org/citation.cfm?id=1368018.1368024.

[Fa06] Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27/8, ROC Analysis in Pattern Recognition, pp. 861–874, 2006, ISSN: 0167-8655, URL: http://www.sciencedirect.com/science/article/pii/S016786550500303X.

[Fi18] Financial Action Task Force: Professional Money Laundering, Financial Action Task Force, 2018, URL: http://www.fatf-gafi.org/media/fatf/documents/Professional-Money-Laundering.pdf, visited on: 03/17/2019.

[Fi19] Financial Action Task Force: Improving Global AML/CFT Compliance, Financial Action Task Force, 2019, URL: https://www.fatf-gafi.org/publications/high-risk-and-other-monitored-jurisdictions/documents/fatf-compliance-february-2019.html, visited on: 05/10/2019.

[In19] International Organization for Standardization: Country Codes - ISO 3166, 2019, URL: https://www.iso.org/iso-3166-country-codes.html, visited on: 05/10/2019.

[KAS19] Khosla, M.; Anand, A.; Setty, V.: A Comprehensive Comparison of Unsupervised Network Representation Learning Methods. CoRR abs/1903.07902/, 2019, arXiv: 1903.07902, URL: http://arxiv.org/abs/1903.07902.

[MB05] Morin, F.; Bengio, Y.: Hierarchical Probabilistic Neural Network Language Model. In (Cowell, R. G.; Ghahramani, Z., eds.): Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics, pp. 246–252, 2005, URL: http://www.iro.umontreal.ca/~lisa/pointeurs/hierarchical-nnlm-aistats05.pdf.

[Mi13a] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient Estimation of Word Representations in Vector Space. CoRR abs/1301.3781/, 2013, arXiv: 1301.3781, URL: http://arxiv.org/abs/1301.3781.

[Mi13b] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In (Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; Weinberger, K. Q., eds.): Advances in Neural Information Processing Systems 26. Curran Associates, Inc., pp. 3111–3119, 2013, URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[MK11] Michalak, K.; Korczak, J. J.: Graph mining approach to suspicious transaction detection. 2011 Federated Conference on Computer Science and Information Systems (FedCSIS)/, pp. 69–75, 2011.

[PAS14] Perozzi, B.; Al-Rfou, R.; Skiena, S.: DeepWalk: Online Learning of Social Representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’14, ACM, New York, New York, USA, pp. 701–710, 2014, ISBN: 978-1-4503-2956-9, URL: http://doi.acm.org/10.1145/2623330.2623732.

[PF01] Provost, F.; Fawcett, T.: Robust Classification for Imprecise Environments. Mach. Learn. 42/3, pp. 203–231, Mar. 2001, ISSN: 0885-6125, URL: http://dx.doi.org/10.1023/A:1007601015854.

[Sa17] Savage, D.; Zhang, X.; Wang, Q.; Yu, X.; Chou, P.: Detection of Money Laundering Groups: Supervised Learning on Small Networks. In: The AAAI-17 Workshop on AI and Operations Research for Social Good. Vol. WS-17-01, Association for the Advancement of Artificial Intelligence, pp. 43–49, 2017.

[Su01] Suendorf, U.: Geldwäsche - Eine kriminologische Untersuchung. Hermann Luchterhand Verlag GmbH, 2001.

[Sw01] Swets, J. A.: Measuring the Accuracy of Diagnostic Systems. Science 240/4857, pp. 1285–1293, 2001, ISSN: 0036-8075, URL: https://doi.org/10.1126/science.3287615.

[SW19] SWIFT: Data Standards, 2019, URL: https://www.swift.com/standards/data-standards/bic, visited on: 05/10/2019.

[TB99] Tipping, M. E.; Bishop, C. M.: Mixtures of Probabilistic Principal Component Analyzers. Neural Computation 11/2, pp. 443–482, 1999, URL: https://doi.org/10.1162/089976699300016728.

[We18] Weber, M.; Chen, J.; Suzumura, T.; Pareja, A.; Ma, T.; Kanezashi, H.; Kaler, T.; Leiserson, C. E.; Schardl, T. B.: Scalable Graph Learning for Anti-Money Laundering: A First Look. CoRR abs/1812.00076/, 2018, arXiv: 1812.00076, URL: http://arxiv.org/abs/1812.00076.

[Xu15] Xu, R.; Chen, T.; Xia, Y.; Lu, Q.; Liu, B.; Wang, X.: Word Embedding Composition for Data Imbalances in Sentiment and Emotion Classification. Cognitive Computation 7/2, pp. 226–240, 2015, URL: https://doi.org/10.1007/s12559-015-9319-y.