Injecting Semantic Background Knowledge into Neural Networks using Graph Embeddings

Konstantin Ziegler*, Olivier Caelen†, Mathieu Garchery*‡, Michael Granitzer*, Liyun He-Guelton†, Johannes Jurgovsky*‡, Pierre-Edouard Portier‡, Stefan Zwicklbauer*

*University of Passau, Germany, {firstname.lastname}@uni-passau.de
†ATOS Worldline, Belgium and France, {firstname.lastname}@worldline.com
‡INSA Lyon, France, {firstname.lastname}@insa-lyon.fr

***Abstract*—The inferences of a machine learning algorithm are naturally limited by the available data. In many real-world applications, the provided internal data is domain-specific and we use external background knowledge to derive or add new features. Semantic networks, like linked open data, provide a largely unused treasure trove of background knowledge. This drives a recent surge of interest in unsupervised methods to automatically extract such semantic background knowledge and inject it into machine learning algorithms.**

In this work, we describe the general process of extracting knowledge from semantic networks through vector space embeddings. The locations in the vector space then reflect relations in the original semantic network. We perform this extraction for geographic background knowledge and inject it into a neural network for the complicated real-world task of credit-card fraud detection. This improves the performance by 11.2%.

***Index Terms*—Semantic Web, Semantic Networks, Knowledge Graphs, Neural Networks, Graph Embeddings, Outlier Detection, Fraud Detection**

I. Introduction

Data-driven inference and machine learning mechanisms have become powerful technologies not only in research, but also in everyday applications. However, inferences made by machine learning methods are obviously limited by the relevant patterns found in data. But even in the case where relevant patterns exist, the machine learning method may not be able to identify them. These limitations can be overcome with manual feature engineering or the integration of background knowledge.

Integrating background knowledge can be done either on an algorithmic level, for example through kernel functions, or on a data level by enriching and combining data sets. In either case, new features contain relevant background knowledge, that is general facts that are obvious to humans, but not contained or identifiable in the data. However, the integration of background knowledge usually remains a manual task. In particular, it particularly requires manual effort to convert background knowledge, represented as semantic networks, like the semantic web, into a tabular structure. Furthermore, adding additional attributes may decrease efficiency and performance of machine learning algorithms due to correlated features, a higher dimensional data set or unsuitable encodings. For example, nominal variables (gender, country, user-id, etc.) are usually added as one-hot-encoding in neural networks, that is every attribute value constitutes one input parameter. Hence, nominal variables with a large cardinality dramatically increase the dimension of the input space.

For some practical classification problems, machine learning methods may offer acceptable solutions even without careful engineering or tuning. However, in most scenarios the tasks are rather difficult in the sense that an algorithm is asked to find the best hypothesis from a large set of valid ones, either because the modeled assumptions are far off from the true distribution or because the set of representable hypotheses largely exceeds the number of observations. But one can still cope with this notion of difficulty by tuning the model's hyper-parameters or, when possible, collecting more observations. Another kind of challenge emerges once observations of different classes are particularly mixed in input space, that is the classes overlap.

A *semantic network* (or *knowledge graph*) is a multi-relational directed graph composed of entities as nodes and relations as edges [1]. In our work, we present a method to integrate linked open data [2], [3] as background knowledge into neural networks. In particular, we use graph embeddings based on previous work [4], that is real-valued vector representations for nodes in the semantic network, in order to capture the semantic properties of an individual node. These embeddings are then used to initialize an embedding layer in the neural network. During subsequent training these embedding layers from the semantic background are further adapted to the given task.

Our method is applied to a large scale real world data set for credit-card fraud detection. This domain seems particularly well-suited to the injection of background knowledge, because the internal application data does not reflect the cultural context of a transaction, for example local holidays, judicial system, etc. – an information which may be extracted from the background knowledge in the linked open data. In this domain, we show, that by creating embeddings for country nodes in DBpedia [5], we can significantly improve the fraud detection performance. Furthermore, we study the effect by augmenting the dataset with information on public holidays, which again shows the detection improvement.

Credit-card fraud detection offers an optimal use-case for the evaluation of injecting semantic background knowledge as the rarely occurring fraudulent credit-card transactions are very similar to many legitimate transactions with respect to the input features. A classification algorithm may not be able to accurately

discriminate such nearby instances without further assumptions about the data. These assumptions could be introduced in form of additional knowledge that has been extracted from external unrelated data sources. In this work, we show how to leverage and integrate such prior knowledge from structured knowledge bases like semantic networks in an automated manner.

We make the following contributions.

- Extraction into generic feature representations. (section II)
- Application to geospatial and temporal linked open data.
- Injection in a neural network and improvement on a real-world application credit-card fraud detection. (subsection IV-D) This generalizes the common approaches of feature engineering using embeddings of semantic networks.

The rest of the paper is structured as follows. In section II, we formalize the problem and describe our approach. In section III, we describe the credit-card data provided by our industrial partner Worldline, the peculiarities of credit-card fraud detection, and the related work. In section IV, we provide the setup, the experiments, the evaluation, and the results. Finally, we conclude in section V with an outlook.

II. Approach

In this section, we present our approach in two steps. First, we describe how to obtain graph embeddings following our previous work on embeddings for entity disambiguation. Second, we discuss how to integrate those embeddings in the context of neural networks.

In order to obtain semantically preserving embeddings on graphs, we use our embedding algorithm developed for entity disambiguation [4]. This algorithm is similar to other recently proposed graph embedding methods, such as Node2Vec [6], DeepWalk [7] or LINE [8]. We briefly summarize our algorithm in the following.

A. Problem Statement

We combine two sources of information.

- A relational database $D = (d_{ij})_{1 \leq j \leq n, 1 \leq j \leq m}$, with n tuples (rows) of m attributes (columns) each, representing our internal application data, and
- a semantic graph $G = (V, E)$ representing our background knowledge

Furthermore, we assume that there is some attribute j^* in the database, where the value set $A_{j^*} = \{d_{ij^*} : 1 \leq i \leq n\}$ can be identified with some subset of vertices $V^* \subseteq V$ of G. For example, attributes like "country" or "year" can be easily identified with entries in the DBpedia. Without loss of generality, we assume that $j^* = 1$, s.t. every tuple in D has the format $d_i = (v, d_{i,2}, \ldots, d_{i,m})$ for some $v \in V^*$.

The problem of *injecting semantic background knowledge* is then a combination of feature learning and transfer learning: First, we extract semantic knowledge in the form of a vector representation and then transfer this knowledge by injecting the embeddings in the form of enriched features. We show the generation of vector representations in the next subsection and its effect on the performance in section IV.

B. Semantic Embeddings

Embeddings are real vectors associated to discrete concepts. These vectors inherit some of the semantics of the concepts, so that similar concepts are associated with close vectors. Their semantic similarity can then be easily expressed in terms of the cosine similarity or a vector space metric. Embeddings have been well researched in the field of natural language processing for representing the semantics of words on a corpus [9], with Word2Vec being the most well known algorithm.

Word2Vec is a group of unsupervised learning algorithms to create word embeddings from (textual) documents. To train these embeddings, Word2Vec uses a two-layer neural network to process non-labeled documents. The neural network architecture is based either on the continuous bag of words (CBOW) or the skip-gram architecture. Using CBOW, the input to the model could be $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$, the preceding and following words of the current word w_i. The output of the network is the probability of w_i being the correct word. In this context, the task can be described as predicting a word given its context. The skip-gram model works in the opposite fashion: the input to the model is a word w_i and Word2Vec predicts the surrounding context words $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$. If two words appear in similar contexts, their vector representations are close in the embedding space. Word2Vec obtains a vector-representation for every word by predicting word-sequences.

C. Graph Embeddings

To employ Word2Vec, we have to generate a meaningful sequence of vertices from V^* for a given RDF-graph. We proceed in two steps. First, we transform the RDF-graph into an undirected simple graph G. Second, we perform a random walk (with occasional jumps) on G and record all vertices from V^* that we visit.

We consider the RDF-graph as an undirected simple graph $G = (V, E)$, where the nodes V are the resources of the knowledge base. Two resources $u, v \in V$ share an edge in G, if there is a properties p in the knowledge base such that either (u, p, v) or (v, p, u) (or both) are an RDF-triple in the knowledge base. In other words, we forget the directions of all edges in the RDF-graph and merge multiple edges.

Our random walk is parametrized by two discrete probability distributions. For a node $v \in V$, we denote its neighborhood as $N(v)$, its degree as $\deg(v) = |N(v)|$, and the logarithmic transformation of the relative degree as $\mathrm{reldeg}^*(v) = -\log(\deg(v)/|E|)$. We introduce a random variable X on V which samples a node proportional to its reldeg^*, that is with probability

$$\Pr(X = v) \propto \mathrm{reldeg}^*(v) \tag{1}$$

When our random walk is at a node v, we can pick the next node either uniformly at random from $N(v)$ ("step") or according to X from V ("jump"). The latter is also our choice for the initial node.

The resulting algorithm takes two parameters. First, a real value α describes the probability of a "jump" – as opposed to a "step". In our experiments, we use $\alpha = 0.1$. However, values

of $0.05 < \alpha < 0.25$ do not significantly affect the resulting Word2Vec model. Second, the integer parameter θ specifies the number of sampled nodes from the graph. We suggest to use $\theta = 5 \cdot |E|$, which results in $\approx 50M$ random walks for DBpedia. Higher values of θ do not improve the entity embeddings but increase the training time. We note that we only write a node v to the corpus file if it is contained in the subset of desired entities V^*. The corpus creation approach for RDF-knowledge bases is summarized in Algorithm 1.

Algorithm 1: Generate Word2Vec corpus from RDF-graph

input : undirected graph
$G = (V, E)$, relevant entities $V^* \subseteq V$, random variable X on V
output : word2vec corpus
parameter : α node jump probability, θ number of samples

$v \xleftarrow{\$} V$ randomly according to X
$walks \leftarrow 0$
while $walks < \theta$ **do**
  if $v \in V^*$ **then**
    $appendToOutputFile(v)$
  if $randomInt(100) > (\alpha * 100)$ **then**
    $v \xleftarrow{\$} N(v)$ uniformly at random ; `// step`
  else
    $v \xleftarrow{\$} V$ randomly according to X ; `// jump`
  $walks \leftarrow walks + 1$;
return OutputFile

III. Geographical and Temporal Data for Credit Card Fraud Detection

We study the effect of our proposed method in a real world application: Credit-card fraud detection (CCFD). The goal of a fraud detection system is to identify frauds among a set of given credit-card transactions. The system can enrich the transaction data with further features and match the current transaction with the previous purchases and the profile of the credit-card holder; such additional features can include simple features like the average expenditure or the average number of transactions in the same day. It is also possible to add more advanced features like the ones that we propose in this work. Concretely, we automatically extract semantic information about *countries* and *public holidays* from publicly available knowledge bases and we represent this information in the form of dense vectors that can be readily injected into a classifier as additional features. In the experimental section we will check if these new features have the potential to be informative in determining if a transaction is a fraudulent one or not.

A. A difficult binary classification problem

The credit-card fraud detection problem consists in identifying the frauds among a set of given credit-card transactions. Traditionally, the fraud detection is processed by expert-rule based systems. In this work, we address this problem from a machine learning point of view, as a two-class (legitimate and fraudulent) classification task. Thus, we try to predict the class of a transaction given its attributes, which contain contextual information about the transaction (for example the time and place where it took place) and the credit-card holder. By doing so, we expect the attributes — also called features in the machine learning community — to be sufficiently expressive to distinguish fraudulent from legitimate transactions. Like in any classification task, we first build a classifier according to observed data ("training phase"), before evaluating it on new, unseen transactions by comparing the labels assigned by the model with the expected ones ("testing phase").

The binary classification problem of credit-card fraud detection has been widely studied, because of its great importance for credit-card holders, credit-card issuers and banks, as the financial losses due to fraud are already very high and growing. However, existing models are not giving satisfying results yet, and there is not a unique well-accepted approach as of now. In subsection III-C, we briefly review existing systems that have been built for this task, showing different possible approaches. We also outline some characteristics and challenges that are very specific to the CCFD problem in subsection III-B, and explain if and how they can be bypassed.

Our motivation to inject linked data is driven by the following two hypotheses.

- H_{geo}: "Semantic information on the geographical data can improve the performance of a fraud detection algorithm."
- H_{temp}: "Semantic information on the temporal data can improve the performance of a fraud detection algorithm."

Geographic information about countries appears to be a valuable resource in our context as it enables us to relate, so far, independent countries to each other; both via geographic proximity and governmental affiliation. Another aspect is the injection of temporal semantic information such as public holidays. Public holidays are interesting in the context of credit-card fraud detection because the credit-card holder's behavior is expected to change on public holidays. Therefore, knowing if a transaction takes place on a public holiday or not could be an informative feature for our classification task.

B. Peculiarities of credit-card fraud detection

Credit-card fraud detection is a highly relevant but very specific classification problem: many particularities of this machine learning task have been pointed out in previous research. We now summarize the specific research questions and explain how we intend to deal with them in this work.

Due to the fraudulent behavior it tries to uncover and the huge financial losses involved, credit-card fraud detection is by nature a very sensitive matter. Research in this domain is absolutely necessary to reduce fraud costs, but credit-card data cannot be shared for confidentiality reasons. This makes credit-card fraud detection an opaque field, where existing techniques are often kept (at least partially) secret and results cannot be easily compared.

The second specificity of credit-card fraud is the highly unbalanced distribution in the datasets. Fraudulent transactions represent a very small proportion of all transactions: the average fraud rate is often under 0.5% [10], [11]. Thus, credit-card fraud detection is often considered as an anomaly detection problem, which is characterized by a highly

unbalanced distribution between positive and negative examples. This can be a serious problem for many machine learning algorithms that perform very poorly on uneven distributions. In our case, we choose to overcome this difficulty by downsampling (discarding) legitimate transactions in the datasets to obtain new datasets with much higher fraud rates. This technique seems to work well with neural networks.

The next specificity of credit-card fraud detection is the complex nature of the problem: frauds are difficult to distinguish from legitimate transactions, and the class distributions are overlapping [10], [11]. Moreover, and as mentioned previously, different fraud schemes are used by the fraudsters, leading to heterogeneous fraudulent transactions. We do not address the problem of distinguishing between possible fraud schemes, as our transaction dataset does not contain information about fraud types. A further problem is that class labels can be unreliable, as mentioned in [12]. Concerning our research, the labels of the dataset provided by our industrial partner Worldline seem reliable enough.

C. Related Work

As credit-card fraud detection is a widely studied classification task, many different machine learning models and techniques have been applied to it. A comparative study of existing systems can be found in [12], a review of statistical methods for fraud detection in general in [12], and strategies for feature engineering for fraud detection in [13]. In general, two complementary dimensions have been explored in the domain of credit-card fraud detection. On one hand, different machine learning models and algorithms, such as random forests, support-vector machines, and boosting have been compared to evaluate their relative performance and adaptation to this problem. On the other hand, feature engineering methods have been used to make input transaction data more explicit to help target models. These two points will be detailed in the following.

1) Machine Learning and Statistical Models: Among all models used for credit-card fraud detection, artificial neural networks are quite popular. These machine learning models are used to approximate unknown functions from which the inputs are projected to the outputs, and can be used directly on the transaction data to build a classifier [14]–[16]. In the case of credit-card fraud detection, neural networks implicitly try to model a function that returns a label corresponding to the nature of the transaction (legitimate or fraudulent) taking the transaction features as parameters. It is also possible to combine neural networks with data mining techniques to build association rules-based systems, like in the study conducted by [17]. Furthermore, optimization techniques, like genetic algorithms, are used to improve the model's performance, for example of neural networks [16] or rule-based systems [18]. Meta-learning models, which allow to combine classifiers by stacking them so that the next classifier learns from the behavior of the previous one, can also be an option, as shown in [19].

2) Feature engineering: Feature Engineering is complementary to classical learning algorithms to improve model prediction performance. Feature engineering systems focus primarily on how to optimize data representation in order to better use machine learning techniques. For example, Paulheim et. al [20] significantly reduced the prediction error of their model by adding new attributes related to their classification task of fuel consumption, such as car types and categories.

One intuitive and efficient way to characterize the transaction context such as the spending history would be feature aggregation, as in [21]. The main idea here is to combine several transactions of the same credit-card holder to reduce noise and extract an average spending behavior. Parallel to feature aggregation, new attributes can be added. In [10], the spending history is materialized by recency (time since last purchase), frequency (of credit-card use) and monetary (transaction amount) attributes. Through feature engineering, more complex attributes can be designed: the authors of [13] use von Mises distribution to encode periodic attributes such as the hour of the transaction, and [10], [22] proposes to design a network of merchants and credit-card holders based on the transactions between them.

Although introducing new features can be very efficient, feature engineering requires advanced knowledge and understanding of the data, and this often makes it complex. We propose in this work a new feature engineering approach based on graph embedding of linked open data, which provides a way to integrate in an elegant way external knowledge, in a supervised learning context.

TABLE I
FRAUD/NON-FRAUD SAMPLES IN THE TRAINING/TEST DATA.

	Fraud	Non-fraud	Total
Training data	67 381	603 092	670 473
Test data	16 603	9 446 387	9 462 990

IV. EXPERIMENTS

A. Data

Our training data contains transactions from 1st of March 2015 to 13th of May 2015. We undersampled the majority class, that is the class of fraudulent transactions, to obtain an overall fraud rate of 10% in the whole training set. The test data contains 9 462 990 transactions from the period 14th of May 2015 to 31st of May 2015. The class distribution in the test and training data is summarized in Table I.

A single transaction from the dataset is characterized by attributes (features) giving information about the context in which it has been issued. The features cover a variety of properties of the card-holder, the merchant involved and the transaction itself, for example time of transaction, amount spent. All but these two features are categorical.

In this context, we assess the performance gain induced by our extracted features: the semantic embeddings of countries and the public holiday feature. The holiday feature is two-fold and it indicates whether the transaction takes place on a public holiday according to its location (referred to as "transaction holiday") or according to the address of the credit-card holder (called "card holder holiday").

B. Setup

As a classifier we implemented a deep neural network with all available features as input and an additional embedding layer that provides access to the country embeddings. On top of this input layer, we stack five fully connected layers of decreasing size (180, 160, 140, 120, 100) with $\tanh$-activations and finally a fully connected output layer of size 2 with $\mathrm{softmax}$-activation. The layer sizes were chosen manually after experimenting with several other topologies. The network is trained as binary classifier on single credit-card transactions with label "fraud"/"non-fraud" and the following parameters: learning rate and embedding learning rate = 0.01, L1-regularization at 0.01 and 100 training iterations. We implemented the neural network in Python using the symbolic computation library `Theano`[1] for automatic differentiation. All experiments were conducted on a NVIDIA Tesla K80 GPU.

In the experiments, we compare four different configurations for the embedding layer:

- no external feature at all, that is using one-hot encoding for all features
- embedding the country feature as a vector representation using the approach of Algorithm 1 with the skip-gram architecture
- adding a "transaction holiday" feature (with possible values "Yes"/"No"/"N/A") using the transaction date, the seller country and the external data from Mozilla's calendars
- with the previous two combined.

C. Performance Measures

Another particularity of the fraud detection problem is that classical performance measures from the confusion matrix (true positive rate, true negative rate, accuracy) are not suitable [11], [23], [24]. With a fraud rate around 0.1%, a dumb model classifying all transactions as legitimate would reach an accuracy score of 99.9%, although it would be totally useless. More advanced measure such as Receiver Operator Characteristic (ROC) curves, which show how the number of correctly classified positive examples (recall) varies with the number of incorrectly classified negative examples, still presents an overly optimistic view of an algorithm's performance when the dataset is highly skewed [25]. One well accepted measure is the area under the precision-recall (PR) curve of the fraudulent class. This curve represents the precision (proportion of true positives among found positives) at different recall (proportion of found positives among expected positives) steps, and the AUC (area under curve) expresses the global quality of the classifier when considering various discrimination thresholds.

In our work, given the context of the project and the collaboration with Worldline, performance measures are those used by the company. This includes pk (precision at k) scores and area under curve (AUC). The pk score represents the precision (i.e. the proportion of frauds) among the k most suspicious transactions. In practice, pk is used, because the number of alerts reported to a human expert for validation is limited. Higher pk-scores lead to more frauds that will be detected immediately. The curve used for AUC is simply the pk score at different recall steps (considering the k most suspicious transactions at each step) until all fraudulent transactions are retrieved. A perfect classifier would always rank frauds as more suspicious than legitimate transactions, the pk score would always be 1 (at any k), and the AUC would therefore be 1. We are aware that our performance measures do not take the cost structure into account, but these metrics have been chosen to make our models comparable to other studies conducted within the research collaboration project.

[1] http://deeplearning.net/software/theano/

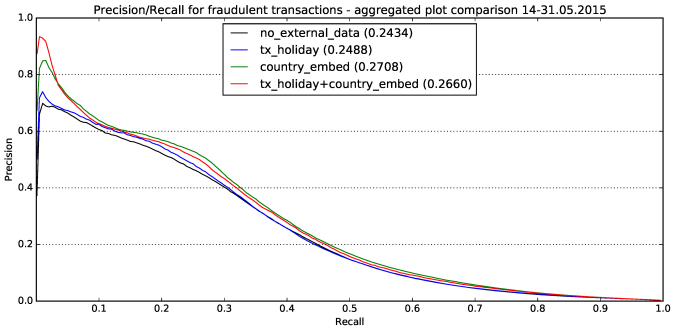

Fig. 1. Global AUC scores over the whole test period (14.05.2015–31.05.2015) with and without public holiday attribute (transaction holiday) and external country embeddings.

D. Results

Figure 1 shows the global AUC scores of the different configurations over the whole test period. The baseline configuration obtains a score of 0.2434. With the holiday attribute, the model performs slightly better (AUC = 0.2488, +2.2% w.r.t. the baseline). The performance gain is much more significant with the country embeddings integration (AUC = 0.2708, +11.2%). The configuration with both external knowledge sources (public holiday and country embeddings) gives a somewhat lower score (AUC = 0.2660, +9.3%), but it must be noted that this model performs the best on the most suspicious transactions. Precision values of around 0.90 can be reached at low recall values, which is note-worthy because the classifiers are mostly used at high confidence values in practice.

Figure 2 shows the average daily AUC scores of the different configurations, expressing similar tendencies as with the global scores over the whole test period. The configuration with public holiday attribute (AUC = 0.2427, +2.1%) performs better than the baseline model (AUC = 0.2377). The configuration with external country embeddings gives the best overall AUC score (0.2567, +8.0%), and the model with both external knowledge sources has a slightly lower score (AUC = 0.2519, +6.0%) but performs better on the most suspicious transactions (higher precision at low recall values).

V. Conclusion

We have shown that injecting semantic background knowledge from external sources can improve the performance

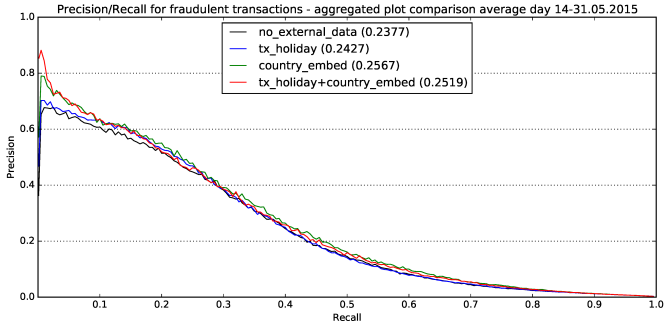

Fig. 2. Average daily AUC scores over the whole test period (14.05.2015–31.05.2015) with and without public holiday attribute (transaction holiday) and external country embeddings using the Neural Network classifier.

of a neural network for credit-card fraud detection. We have shown this using country embeddings derived from DBpedia and holiday labels derived from Mozilla's calendar project. Our experiments have also shown that combining several new features to add to the training data is not straightforward: adding a feature can decrease the model's performance in presence of another feature.

However, combining semantic vector representations of countries and public holidays seem to work quite well, especially for low recall values where a higher precision can be reached. Concretely, it means that such classifier will perform better on the most suspicious transactions, which represents the most common use case in practice. Therefore, combining country embeddings and public holidays successfully improves the detection rate of fraudulent transactions.

Further research should integrate semantic networks beyond the geographical and temporal data investigated here and further investigate the relation between (automatic) semantic representations and (manual) semantic features.

ACKNOWLEDGMENT

The authors would like to thank Emanuel Berndl for pointers to linked data literature.

REFERENCES

[1] S. Guo, Q. Wang, B. Wang, L. Wang, and L. Guo, "Semantically smooth knowledge graph embedding," in *Proceedings of ACL*, 2015, pp. 84–94.

[2] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009. [Online]. Available: http://dx.doi.org/10.4018/jswis.2009081901

[4] S. Zwicklbauer, C. Seifert, and M. Granitzer, "Doser - a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings," in *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, ser. Lecture Notes in Computer Science, H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto, and C. Lange, Eds., vol. 9678. Springer, 2016, pp. 182–198. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-34129-3_12

[5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[6] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[7] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1067–1077.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[10] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38–48, 2015.

[11] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, "Credit card fraud detection using bayesian and neural networks," in *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, 2002, pp. 261–270.

[12] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical science*, pp. 235–249, 2002.

[13] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.

[14] S. Ghosh and D. L. Reilly, "Credit card fraud detection with a neural-network," in *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, vol. 3. IEEE, 1994, pp. 621–630.

[15] E. Aleskerov, B. Freisleben, and B. Rao, "Cardwatch: A neural network based database mining system for credit card fraud detection," in *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*. IEEE, 1997, pp. 220–226.

[16] R. Patidar, L. Sharma *et al.*, "Credit card fraud detection using neural network," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 1, no. 32-38, 2011.

[17] R. Brause, T. Langsdorf, and M. Hepp, "Neural data mining for credit card fraud detection," in *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*. IEEE, 1999, pp. 103–106.

[18] I. Trivedi and M. M. Monika, "Credit card fraud detection," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 1, 2016.

[19] S. Stolfo, D. W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Credit card fraud detection using meta-learning: Issues and initial results," in *AAAI-97 Workshop on Fraud Detection and Risk Management*, 1997.

[20] H. Paulheim, P. Ristoski, E. Mitichkin, and C. Bizer, "Data mining with background knowledge from the web," *RapidMiner World*, 2014.

[21] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," *Data Mining and Knowledge Discovery*, vol. 18, no. 1, pp. 30–55, 2009.

[22] B. Lebichot, F. Braun, O. Caelen, and M. Saerens, "A graph-based, semi-supervised, credit card fraud detection system," in *International Workshop on Complex Networks and their Applications*. Springer, 2016, pp. 721–733.

[23] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," *Expert systems with applications*, vol. 41, no. 10, pp. 4915–4928, 2014.

[24] D. Hand, C. Whitrow, N. Adams, P. Juszczak, and D. Weston, "Performance criteria for plastic card fraud detection tools," *Journal of the Operational Research Society*, vol. 59, no. 7, pp. 956–962, 2008.

[25] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.