NAG: neural feature aggregation framework for credit card fraud detection

Kanishka Ghosh Dastidar¹ · Johannes Jurgovsky¹ · Wissam Siblini² · Michael Granitzer¹

Received: 28 January 2021 / Revised: 25 December 2021 / Accepted: 27 December 2021 /
Published online: 6 February 2022
© The Author(s) 2022

Abstract

The state-of-the-art feature-engineering method for fraud classification of electronic payments uses manually engineered feature aggregates, i.e., descriptive statistics of the transaction history. However, this approach has limitations, primarily that of being dependent on expensive human expert knowledge. There have been attempts to replace manual aggregation through automatic feature extraction approaches. They, however, do not consider the specific structure of the manual aggregates. In this paper, we define the novel Neural Aggregate Generator (NAG), a neural network-based feature extraction module that learns feature aggregates end-to-end on the fraud classification task. In contrast to other automatic feature extraction approaches, the network architecture of the NAG closely mimics the structure of feature aggregates. Furthermore, the NAG extends learnable aggregates over traditional ones through soft feature value matching and relative weighting of the importance of different feature constraints. We provide a proof to show the modeling capabilities of the NAG. We compare the performance of the NAG to the state-of-the-art approaches on a real-world dataset with millions of transactions. More precisely, we show that features generated with the NAG lead to improved results over manual aggregates for fraud classification, thus demonstrating its viability to replace them. Moreover, we compare the NAG to other end-to-end approaches such as the LSTM or a generic CNN. Here we also observe improved results. We perform a robust evaluation of the NAG through a parameter budget study, an analysis of the impact of different sequence lengths and also the predictions across days. Unlike the LSTM or the CNN, our approach also provides further interpretability through the inspection of its parameters.

✉ Kanishka Ghosh Dastidar
kanishka.ghoshdastidar@uni-passau.de

Johannes Jurgovsky
johannes.jurgovsky@googlemail.com

Wissam Siblini
wissam.siblini@univ-nantes.fr

Michael Granitzer
michael.granitzer@uni-passau.de

¹ Faculty of Computer Science and Mathematics, University of Passau, Passau, Germany

² Development and Innovation, Worldline, Lyon, France

Keywords Finance · Credit card fraud · Representation learning

1 Introduction

In the E-commerce domain, credit card fraud occurs when an entity gets unauthorized access to a card and uses it to make online purchases. Fraud detection mechanisms aim to leverage two major constraints faced by fraudsters during the execution of credit card fraud. Successful fraud strategies are subject to limited time budgets as cardholders and banks freeze the card upon fraud detection. Fraudsters are required to reach the card's credit limit in a short time, and consequently, their behavior is exposed in the form of fraudulent transactions within these short time windows. The second form of constraint is imposed by the diversity of technical payment modalities implemented at merchants and the diversity of their businesses. Security measures of merchants and devices may hinder effective fraud. Such obstacles force fraudsters to exploit few vulnerable merchants more frequently. This circumstance leads to fraudulent transactions being issued at few particular merchants that differ from the local set of merchants the cardholder usually interacts with.

Conceptually, fraud detection mechanisms lie on a continuum bounded by the two extremes: *Rule-based* methods employ sophisticated detection rules designed by domain experts for the identification of fraudulent transactions [39]. A rule catalogue has to be defined, maintained and evaluated regularly to keep the detection system synchronized with fraud strategies. At the other extreme, *machine learning-based* methods sidestep the need for detailed domain expertise by automatically testing large numbers of hypotheses on large amounts of historical data in order to identify the one that best explains the correlation between the fraud label and the transaction attributes. A method's position on that scale is governed by the extent of feature engineering involved and the kinds of assumptions built into the model. For instance, in image recognition, we might assume that patterns cover spatially proximate pixels and that the same pattern can appear at any position. In time series analysis we might assume that the current value depends on no more than the two previous time steps or on some latent state altogether. These assumptions are built into neural network architectures such as convolutional networks and recurrent networks, respectively, to constrain the set of hypotheses while balancing plausibility and computational efficiency.

In credit card fraud detection, a commonly adopted feature engineering strategy revolves around feature aggregations [8,42]. This strategy relates any transaction to its recent past by computing statistics over those transactions that are similar to the current one. Since these features strongly improve fraud detection, they often form the basis of complex rules and also enter ML-based methods as additional features along with the raw transaction attributes. These rules are defined based on human expertise, which is expensive.

In this paper, we propose a neural network based feature extraction module that exploits the current-to-past relationship in transaction sequences in a similar manner as manually defined feature aggregations. But as the module is differentiable, we do not need to define the relationships a priori but leave it to the optimization algorithm to discover them based on the given prediction task. In particular, we make the following contributions:

- We discuss the limitations of state-of-the-art manual feature aggregation strategies in credit card fraud detection.
- We propose a novel neural network based feature aggregation framework, the Neural Aggregate Generator (NAG), that closely mimics the structure of state-of-the-art manual feature aggregates. The NAG extends manual aggregates through soft feature matching

and relative weighting of feature importance. We provide a proof that the NAG is capable of modeling a manual aggregate for a set of given categorical feature constraints with the sum as the aggregation function.

- We validate the NAG on a real-world fraud detection dataset. We show empirically that the predictive performance of the NAG is better than using manual feature aggregates, thus enabling a reduction in reliance on human expertise.
- We also compare the NAG in terms of predictive performance to other automatic feature extraction approaches such as an LSTM and generic CNN and observe that the NAG outperforms the other end-to-end models.
- We extend our evaluation of the NAG through a set of follow up experiments: Firstly, we compare the NAG to both context-based and static approaches in terms of daily predictions. Here we show through a *Friedman–Nemenyi* test that the NAG is significantly different than the other methods. We also conduct a parameter budget study. Here we show that even with smaller numbers of parameters the NAG outperforms the LSTM. Following that, we study the impact of using different sizes of the transaction history window on the performance of the NAG.
- Additionally, as the architecture closely mimics manual aggregations, we are able to analyze the parameters of the NAG for a more direct interpretation that is in line with the intuition of fraud experts.

2 Related work

The recent research on credit card fraud detection focuses primarily on the different peculiarities of the domain. These aspects are covered in detail in surveys by Lucas and Jurgovsky, and Rymann-Tub et al. [27,33]. Both these surveys confirm a few common challenges such as dataset shift, class imbalance and choice of appropriate evaluation metrics. In order to address the challenge of dataset shift, there have been works both on the detection of dataset shift (in this case the co-variate shift) [29] and strategies on how to handle them [10]. As the number of frauds only accounts for a tiny fraction of the total transactions, fraud datasets are usually highly imbalanced. The approaches to tackle this include different sampling measures [20] along with using models that are robust to class imbalance [9]. Finally, in terms of metrics, there has been a vast variety of metrics used to evaluate fraud detection models. Even though the Area under ROC curve (AUC-ROC) is a commonly used metric for classification tasks, it is shown to be unsuitable for fraud detection [36], as it is not robust to class imbalance [11]. Carcillo et al. [6] state that two relevant metrics are the *Top n* Precision (precision for the n transactions with the highest fraud probability) and the Area under the Precision–Recall Curve (AUC-PR). In general, the use of *Top n* Precision could be preferred over the AUC-PR when a fixed budget (n) in terms of the number of transactions that can be analyzed by human investigators is known. AUC-PR, on the other hand, remains more flexible. Another possible option is the use of AUC-PR at the card, rather than at the transaction level as stated in Carcillo et al. [6]. The rationale here is that once a fraudulent transaction for a card has been detected, the card is blocked and no further transactions from the card are considered. However, as it is also important for each transaction to be labeled correctly to maintain high-quality datasets, the AUC-PR at transaction level still finds broad general usage [4,22,25,28,44].

A wide variety of classifiers have been applied on the fraud detection task. Dhankhad et al. [13] compare 10 different classifiers on publicly available data. An ensemble approach, the random forest and XGBoost are shown to perform the best. However, there is relatively

little difference in terms of performance. Bhattacharya et al. [3] compare the performance of random forests and logistic regression on a much larger dataset, with the random forest showing better performance across several different performance measures. Zareapoor and Shamsolmoali [43] applied an ensemble classifier first proposed in Breiman [5] which outperformed other methods such as an SVM and a KNN classifier.

In the last decade, we also see an increase in the use of deep learning models on the fraud detection task. However, the seminal work in this direction was published much earlier by Ghosh and Reilly [17] who used a radial basis feed-forward network to show both an increase in the number of fraudulent accounts detected and a decrease in false positives compared to a rule engine. More recently, Fu et al. [16] propose a CNN, where the input feature map is a matrix with rows representing different features and the columns representing different time periods of aggregation. Gomez et al. [19] use a cascade of two neural network filters to deal with the two problems of class imbalance and then subsequently fraud detection. A recent study on a South Korean credit card dataset selects a feed-forward network out of a choice of possible deep learning method candidates, which is evaluated against an ensemble model in a champion-challenger setting [24].

What we focus on is ‘context-aware fraud detection’ [21], i.e., we consider the set of transactions in a particular time period that are associated with the transaction we are trying to classify. This could, for example, give insight into the properties of this transaction relative to the usual purchase patterns of the card-holder or terminal. Since directly using these past transactions in their entirety would lead to a very high dimensional feature space, these approaches often revolve around summarizing them into a lower-dimensional representation as features for fraud classification. The construction of these features can be divided into two broad categories, the first tending toward automated feature extraction strategies and the second relying on domain knowledge from experts to create rules for feature aggregation.

Jurgovsky et al. [22] use Long Short Term Memory Networks (LSTM) to incorporate transaction sequences from the card-holder perspective. They compare the performance of the LSTM to manual feature engineering strategies and show that while both approaches are comparable in a card-present setting, for e-commerce transactions, manual feature engineering improves performance. Lucas et al. [28] create datasets of transactions from multiple perspectives and model these sequences using Hidden Markov Models (HMM). They then use the likelihood associated with the transactions by each MM as an additional feature in a random forest classifier. While they observed a slight improvement with HMM-generated features compared to using simply raw features, it still fell short in comparison to using manual feature engineering. Srivastava et al. [38] also use HMMs to model sequences of transactions and assign a likelihood of an incoming transaction to belong to that sequence. They compare this likelihood value to a threshold value to determine whether the transaction is fraudulent. Another recent approach has been to use a spatiotemporal attention mechanism along with 3d convolution layers to learn end-to-end spatial and temporal feature aggregations [7]. Nami and Shajari [30] use the average distance similarity measure between the current and past transactions as additional context for the model’s decision, here a fading factor is used to assign more weight to more recent transactions.

Self-attention in particular, proposed by Vaswani et al. [41] for sequence transduction, and also successfully used in other NLP [35] and non-NLP [37] domains, creates a contextual representation of an element of a sequence through an estimation of its relation to other elements. These relations (‘attention scores’) weigh the contributions of the states of the other elements of a sequence to the element in question. While certain components of our method are analogous to this self attention mechanism, there are some significant differences which are outlined in Sect. 4.6.

As mentioned in Sect. 1, the primary approach for representing contextual information uses manually defined rules to aggregate values of certain variables from past transactions. We see several works here that use different feature engineering strategies. Van Vlasselaer et al. [40] use both intrinsic or transaction-based and network-based features. The network features reflect the exposure of a pair of card-holder and merchant to fraud. Alazizi et al. [1] show improved performance over simply using raw features through the use of expert designed aggregate features. Whitrow et al. [42] proposed a framework for transaction aggregation where they identify different aspects of transaction records which are relevant to fraud detection such as the merchant code or the account number. They group transactions based on these features and compute different statistics such as counts or total amounts spent for each of these groups. Bahnsen et al. [8] expanded this by proposing aggregates that take into account multiple features as conditions when computing the aggregate statistic. An example of this would be the sum/count of amounts of all transactions with the same card-holder ID and processed at the same category of merchant. We consider this to be the state-of-the-art in manual feature aggregation.

3 Problem statement

In this section, we describe manual feature aggregates in more detail. Further, we analyze the limitations of these aggregates. We then list the limitations of other sequence-based automatic approaches. We derive the architecture for the NAG in Sect. 4 from this analysis.

Based on Bahnsen's work, we formally define feature aggregates as follows: Let $(x_i)_{i \leq n}$ be a temporal sequence of transactions (of length n) of a single card-holder. Let $\mathcal{M}_n^F$ be a certain subset of transactions, in the context of the current transaction x_n which fulfill certain constraints. F is a set of categorical features. The choice and number of these categorical features in F can vary, e.g., $\{\}$, $\{A\}$, $\{A, B\}$. We denote the value of a categorical feature by superscript. For instance, the value of the country feature of transaction x_i is $x_i^{\{\text{country}\}}$. Based on these constraints, we can then select the subset of transactions issued in the past t hours by:

$$\mathcal{M}_n^{\{\}} = \{x_i \mid \text{hours}(x_i^{(\text{time})}, x_n^{(\text{time})}) < t\}_{i < n} \quad (1)$$

$$\mathcal{M}_n^{\{A\}} = \{x_i \mid (x_i^{(A)} = x_n^{(A)}) \wedge (\text{hours}(x_i^{(\text{time})}, x_n^{(\text{time})}) < t)\}_{i < n} \quad (2)$$

$$\mathcal{M}_n^{\{A, B\}} = \{x_i \mid (x_i^{(A)} = x_n^{(A)}) \wedge (x_i^{(B)} = x_n^{(B)}) \wedge (\text{hours}(x_i^{(\text{time})}, x_n^{(\text{time})}) < t)\}_{i < n} \quad (3)$$

Equation (1) gives us the set of transactions that took place in the preceding t hours for a particular card-holder. Equation (2) is the case where along with our time constraint, we also have a categorical feature constraint, i.e., $F = \{A\}$. If for example, A denoted the feature *country*, (2) represents all transactions in the past 24 h from the same country as the current transaction. Here, the two constraints are the time window, i.e., 24 h, and the value matching of a particular categorical feature, i.e., the country. Equation (3) adds a second categorical feature constraint. Based on the set of matching transactions $\mathcal{M}_n^F$, it is common practice to define some aggregation function on a particular feature, e.g., Z , on each of the transactions in this subset as follows:

$$\text{sum}_{\mathcal{M}_n^F} = \sum_{x \in \mathcal{M}_n^F} x^{(Z)} \quad (4)$$

The choice of aggregation function can vary, including sums, means, span, max, min, count, etc.

Limitations of manual aggregates

Although manual aggregates have shown to be good predictors for fraud, they do have certain disadvantages and limitations. At this point, we identify three major limitations that we are going to address with the proposal of the NAG architecture in Sect. 4:

Cost of manual engineering With the evolving nature of fraudulent behavior, change in customer spending patterns and introduction of different security mechanisms in the transaction process, the choice of constraints and functions for aggregate creation must be regularly analyzed and updated within a rule management life cycle. This requires extensive domain knowledge, which is expensive. Additionally, humans might also fail to recognize all relevant patterns of fraudulent behavior.

Hard matching A second limitation of selecting transactions as shown in (1) and (2) in the context of manual aggregates is that a hard matching takes place, i.e., the matching operation is an identity constraint that returns either `true` or `false`. This does not allow us to take into account the possibility that different values of a categorical feature could be more or less similar to each other in the context of credit card fraud. If we look at the country feature, due to various properties such as dates of public holidays, security mechanisms, punitive measures against fraud, etc. one can expect that the fraud patterns are more similar in certain countries than in others. This applies to other categorical features too. Hard matching does not allow these similarities to be represented in feature aggregates.

Uniform feature importance Another limitation of manual aggregation is that when we use multiple categorical features as our constraints, we first match transactions per feature (identity constraint) and then combine the constraints via Boolean conjunction. Only those transactions that satisfy all constraints enter into the set of matching transactions $\mathcal{M}_n^F$. Obviously, the Boolean conjunction does not allow to weigh the different constraints relative to each other. As a consequence, during aggregation (4) a transaction either contributes fully to the aggregate's value or not at all.

Limitations of automatic feature extraction

While we acknowledge the limitations of manual feature aggregates, we must also point out that they are at present the industry standard and used in real-world production settings. Therefore, the question arises as to what exactly is preventing industrial entities from adopting end-to-end models which learn features automatically in their production settings.

We see that although different automatic approaches show varying degrees of success in terms of fraud predictive performance, they remain far apart from the expert intuition of feature aggregates. For example, an LSTM models the succession of purchases that a card-holder makes. Each latent state of an LSTM can be seen as a 'representation' of the chronological patterns of the previously occurring transactions in the sequence. The manual

aggregates on the other hand form this representation based on matching of attributes of the current transaction to each of the previous ones. This fundamental difference in how the manual aggregates and generic feature extraction approaches learn the context of a particular transaction result in learned representations that are complex and unintuitive for fraud experts. This is especially important in a field such as fraud detection, as client acceptance is crucial for the deployment of such novel methods in production settings. Therefore, here we identify a trade-off between the limitations of manual aggregates as outlined earlier in this section and the lack of interpretability for generic end-to-end approaches.

4 Neural aggregate generator

In this section, we introduce the Neural Aggregate Generator (NAG), a neural network architecture for learning extended feature aggregates. The NAG is intended to be used as an end-to-end feature extraction module, with all the learnable parameters that are introduced in the section being trained on the given task (e.g., fraud classification). We break down the architecture of the NAG into different components mimicking the structure of manual aggregates, namely the selection of a set of transactions and subsequent application of an aggregation function on a certain feature of these selected transactions. We highlight how the NAG addresses the limitations of the manual aggregates through soft feature matching and weighting of feature constraints. An illustration of the computations performed inside the NAG is given in Fig. 1.

4.1 Feature-wise similarity scores

The first component of the NAG architecture performs a selection of the past transactions, similar to manual aggregation. However, instead of performing a hard matching like discussed

Fig. 1 Illustration of the computation flow in the NAG. We distinguish between two types of features: categorical features that serve as constraints during selection and a continuous feature (*target*) that is aggregated. The NAG first computes feature-wise similarities between the n -th transaction and each of the previous transactions and then combines these feature-wise similarities (with parameters $\mathbf{w}$) into a joint transaction similarity. The final aggregate value is calculated as a weighted sum (with parameters $\mathbf{v}$) of target values and transaction similarities

in Sect. 3, we must project the categorical features into a continuous, real-valued embedding space so that similarities between categories become real-valued scores.

We realize this through using pre-trained feature embeddings. We elaborate upon the details of this in Sect. 5.2. Additionally, we can use an entity embedding layer to further optimize the embeddings on the fraud task itself [18]. The embedding layer would be jointly learnt with the other parameters of the network. Parallels can be seen in natural language processing, where word embeddings are used to represent words and phrases as a continuous vector in a semantic space [2].

Based on the embeddings, the NAG can calculate the similarity of a categorical feature value of the current transaction, with the value of the same categorical feature of each of the past transactions. We use the inner product between entity embeddings as a similarity measure, comparable to previous work in different applications such as Elsayed et al. [14], Kandola et al. [23] and Koren et al. [26].

More formally, we start with a sequence of transactions from a single card-holder. After passing the sequence through each embedding layer, we obtain one encoded sequence $(\mathbf{y}_i^f)_{1:n}$, $\forall \mathbf{y}_i^f \in \mathbb{R}^{d_f}$ for each categorical feature f with embedding dimension d_f . For notational convenience, we drop the index of the categorical feature and instead let $(\mathbf{y}_i)_{1:n}$ denote the encoded sequence of any particular feature. In order to quantify the similarity between the n -th transaction and each of the previous ones, we employ the inner product between the n -th embedding vector and any other embedding vector in the sequence which we then consider as a vector of similarity scores $\mathbf{s} \in \mathbb{R}^n$:

$$\mathbf{s} = \mathbf{y}_n^\top \cdot [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n] \quad (5)$$

We interpret components s_i in $\mathbf{s}$ as to how similar transaction i is to transaction n with respect to a particular feature. In order to pick up our notation from before, we calculate the similarity score vector $\mathbf{s}^f$ for each categorical feature f in this manner.

4.2 Transaction similarity scores

The first part of the architecture computes feature-wise similarity scores. Therefore, it gives us a score per feature and per transaction. However, since we aim to quantify the overall similarity between two transactions, we now combine the feature-wise scores into a transaction similarity score. When multiple categorical features are given, we can now learn a relative-weighting of the importance of each feature, which is not possible with standard aggregates (see Sect. 3). Therefore, the NAG introduces learnable parameters that modulate the contribution of each feature to the overall similarity score. More formally, in (5) we calculated the similarity score vectors $\mathbf{s}^f \in \mathbb{R}^n$ for each categorical feature $f \in \{1, \dots, F\}$. Let $S = [\mathbf{s}^1, \dots, \mathbf{s}^F] \in \mathbb{R}^{n \times F}$ denote the similarity matrix in which row i contains the feature-wise similarity scores of transaction i . For instance, if the set of features were $\mathcal{F} = \{\text{country}, \text{city}, \text{merchantID}\}$, the components in row i contain the similarity scores $S_i = (s_i^{\text{country}}, s_i^{\text{city}}, s_i^{\text{merchantID}})$ measuring the feature-wise similarity between transaction i and n .

What remains is to turn the individual feature-wise similarity scores into a single scalar similarity score for the transaction. It is important however that different features may contribute at different magnitudes to this summarized score. A simple approach is to postulate that the transaction similarity score can be expressed as a linear combination of feature-wise similarity scores, parameterized by coefficients $\mathbf{w} \in \mathbb{R}^F$ and a bias b_w , and then passed to

an activation function of choice:

$$\mathbf{m} = \sigma(S \cdot \mathbf{w} + b_w) \quad (6)$$

The parameters $\mathbf{w}$ are shared across all transactions in the sequence—turning $\mathbf{w}$ into a filter similar to filters in a CNN. But instead of fitting patterns on the raw inputs, $\mathbf{w}$ fits patterns on the “ i -to- n ” feature-wise similarity scores. A component m_i in $\mathbf{m} \in \mathbb{R}^n$ is considered a transaction similarity score between transaction i and n across all features with learned importance $\mathbf{w}$ for the features. Consequently, with the use of multiple filters, the NAG can learn different similarity scores over transactions. In the following section, we make use of $\mathbf{m}$ to perform a soft selection over the transaction sequence—contrary to the hard selection in Eq. (2) of manual aggregates.

4.3 Computing the aggregation function

In the previous section, we described the method to calculate similarity scores for each transaction in a sequence, based on the context of the current transaction. In manual aggregates, these scores are effectively either 1 or 0 (selected or not selected). Therefore, for an aggregate which performs a sum of amounts, we select only the amounts of the selected transactions. However, since the NAG computes soft similarity scores, we must scale the target feature vector of the aggregate (e.g., the amounts) with each of the similarity scores.

Let $\mathbf{z} = (z_1, \dots, z_n) \in \mathbb{R}^n$ be feature values of some continuous feature we aim to aggregate, i.e., the *target* feature. The vector $\mathbf{z}$ contains the values of this feature across all transactions in the sequence $1:n$. We can then use the transaction similarity scores to perform a soft selection of components in $\mathbf{z}$. Essentially, we scale each component z_i proportional to the similarity between transaction i and n :

$$\tilde{\mathbf{z}} = (m_1 \cdot z_1, \dots, m_n \cdot z_n) \quad (7)$$

The last step of creating aggregates is to apply some aggregation function on the target feature. In the case of manual aggregates, we must specify different functions such as sum, mean, max, min, etc. In the NAG, we parameterize the aggregation of the scaled target vector $\tilde{\mathbf{z}}$ from (7) with parameters $\mathbf{v} \in \mathbb{R}^n$ and b_v such that we can leave the actual choice of a suitable aggregation function to the optimization algorithm. Therefore, we provide a gradient-based solution for end-to-end learning for a discrete optimization problem. Finally, the value of an aggregate at transaction n is:

$$a_n = \sigma(\tilde{\mathbf{z}}^\top \cdot \mathbf{v} + b_v) \quad (8)$$

We explicitly add the index n to the computed aggregate value a_n here, because we want to emphasize that this value is associated with transaction n and computed from previous transactions $(\mathbf{x}_i)_{1:n}$. We can now append a_n to the n -th transaction’s features and use it as an additional predictor for fraud classification.

4.4 Creating multiple aggregates

We can expand the architecture to generate multiple aggregates for a transaction sequence by simply using multiples of the weight vectors $\mathbf{w}$ in Eq. (6) and $\mathbf{v}$ in Eq. (8). This is an advantage compared to the manual aggregates, where scaling up the number of aggregates is tedious due to the fact that it is dependent on expert knowledge, hence increasing the use of resources.

However, since the NAG enables an automated feature aggregation process, this scaling up is relatively simple as it essentially boils down to increasing the model's parameters.

4.5 Modeling capabilities of the NAG

Here we provide a proof that the NAG can compute a sum aggregation of the amounts of selected transactions based on a given set of categorical feature constraints.

Let $\mathcal{C} = \{c_1, c_2, \dots, c_F\}$ be the set of categorical features in the transactions and z the amount. Consider a sequence $(\mathbf{x}_i)_{1:n}$ of n transactions, i.e., sequences $(c_j^{(1)}, c_j^{(2)}, \dots, c_j^{(n)})$ of n values for each categorical feature c_j and a sequence $(z_1, z_2, \dots, z_n)$ for the amount. We refer to this as the *target* feature in Sect. 4.3.

Property. For any manual aggregate a_n , i.e., for any set of categorical features considered as constraints and a sum aggregation function applied to the amount, there exists a set of weights $(E_j, j \in \{1, \dots, F\}, \mathbf{w}, \mathbf{b}, \mathbf{v})^1$ in the NAG such that $\text{NAG}((\mathbf{x}_i)_{1:n}) = a_n$.

Consider any aggregate a_n with $\mathcal{I}_a$ being the set of indices of categorical features considered in the aggregate's constraints (we drop the subscript of a for the sake of brevity). The computation of a_n consists in aggregating (summing) the amounts in the subset of transactions of the sequence $(\mathbf{x}_i)_{1:n}$ that verify all the constraints, i.e., the transactions of indices $i \in \{1, \dots, F\}$ such that $\forall j \in \mathcal{I}_a, c_j^{(i)} = c_j^{(n)}$:

$$a_n = \sum_{i \in \{1, \dots, n\}, c_j^{(i)} = c_j^{(n)} \forall j \in \mathcal{I}_a} v_i \cdot z_i = \sum_{i \in \{1, \dots, n\}} \left(\prod_{j \in \mathcal{I}_a} \mathbb{1}(c_j^{(i)} = c_j^{(n)}) \right) \cdot v_i \cdot z_i \quad (9)$$

Where $\forall i \in \{1, \dots, n\}$, $v_i = 1$ for the sum aggregation.

For each categorical feature $c_j \in \{1, \dots, k_j\}$ with k_j modalities, let's consider its one hot encoding $h_j \in \{0, 1\}^{k_j}$ such that $(h_j)_p = 1$ if $p = c_j$ and 0 otherwise. Then,

$$\begin{aligned} c_j^{(i)} \neq c_j^{(n)} &\iff h_j^{(i)} \cdot h_j^{(n)} = 0 \\ &\text{and} \\ c_j^{(i)} = c_j^{(n)} &\iff h_j^{(i)} \cdot h_j^{(n)} = 1 \end{aligned}$$

Therefore,

$$\mathbb{1}(c_j^{(i)} = c_j^{(n)}) = h_j^{(i)} \cdot h_j^{(n)}$$

Moreover, since $h_j^{(i)} \cdot h_j^{(n)} \in \{0, 1\}$:

$$\prod_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} = 1 \iff \sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} = \text{card}(\mathcal{I}_a)$$

Here $\text{card}(\mathcal{I}_a)$ is the cardinality of the set $\mathcal{I}_a$.

Since $\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} \leq \text{card}(\mathcal{I}_a)$, we have :

$$\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} = \text{card}(\mathcal{I}_a) \iff \text{ReLU} \left(\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} + b \right) = 1$$

¹ E_j is the embedding matrix of the j th categorical feature in the NAG, $\mathbf{w}$ is a 1-D convolutional filter as described in equation 6, b is the bias in the convolutional layer, and $\mathbf{v}$ is the weight matrix applied to each output of the convolution (see Eq. 8).

$$\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} \neq \text{card}(\mathcal{I}_a) \iff c \left(\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} + b \right) = 0$$

Where $b = 1 - \text{card}(\mathcal{I}_a)$. Here ReLU is the rectified linear activation function. We can then write that:

$$\prod_{j \in \mathcal{I}_a} \mathbb{1}(c_j^{(i)} = c_j^{(n)}) = \mathbb{1} \left(\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} = \text{card}(\mathcal{I}_a) \right) = \text{ReLU} \left(\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} + b \right)$$

Therefore,

$$\begin{aligned} a_n &= \sum_{i \in \{1, \dots, n\}} \text{ReLU} \left(\sum_{j \in \mathcal{I}_a} h_j^{(i)} \cdot h_j^{(n)} + b \right) \cdot v_i \cdot z_i = \\ &\sum_{i \in \{1, \dots, n\}} \text{ReLU} \left(\sum_{j \in \{1, \dots, F\}} w_j \cdot h_j^{(i)} \cdot h_j^{(n)} + b \right) \cdot v_i \cdot z_i \end{aligned} \quad (10)$$

where

$$w_j = \begin{cases} 1 & \text{if } j \in \mathcal{I}_a \\ 0 & \text{otherwise.} \end{cases}$$

Finally,

$$\begin{aligned} a_n &= \sum_{i \in \{1, \dots, n\}} \text{ReLU} \left(\sum_{j \in \{1, \dots, F\}} w_j \cdot E_j h_j^{(i)} \cdot E_j h_j^{(n)} + b \right) \\ &\quad \cdot v_i \cdot z_i = \text{NAG}_{E, \mathbf{w}, b, \mathbf{v}}((\mathbf{x}_i)_{1:n}) \end{aligned} \quad (11)$$

Where $\forall j \in \{1, \dots, F\}$, $E_j = I$.

Note, we can obtain the average aggregation by calculating the ℓ_1 -norm after the activation step. Additionally, we can also obtain min/max aggregation with a max pooling layer instead of a linear combination with $\mathbf{v}$.

4.6 Relation to the self attention mechanism

On a broader level, although the NAG was designed from the perspective of mimicking manual aggregates, certain characteristics are analogous to the self-attention mechanism. One can view the components mentioned in Sects. 4.1 and 4.2 as a constrained version of multi-head attention which, through various simplifications, enables us to come up with a mechanism that requires much less parameters and produces a smaller output.

- **Lightweight similarity calculation:** We drop several steps that require parameterized transformations. Using the self attention mechanism would entail calculating an attention score based on a key, query and value representation (which are parameterized transformations) of a particular transaction embedding $\mathbf{y}_i$. In the case of the NAG, we instead directly use the embedded transactions to calculate our dot-product similarities, thus requiring fewer parameters. This is amplified in the case of multi-head attention, where multiples of the transformation matrices are used to produce different, learned linear projections. Additionally, instead of calculating similarities for pairs of entire transactions, we only consider a certain subset of features, namely categorical features ('our

constraint features') such as merchant code, country, etc., which are based on the set of features used by experts for manual aggregates. In our case, the learnable parameters are only those involved in the combination of scalar similarities, which has a dimension of $n * (F + 1)$ for the $\mathbf{w}$ parameters (see (6)), where F is the number of constraint features and n is the number of output aggregates we want.

- **Single output:** Since our objective was to design a feature extraction module that fits the intuition of fraud detection experts, we are interested in only a description of a single target variable (the amount) of the most recent transaction as a combination of similar predecessors. This is different from the standard self attention mechanism where the output (with a dimension of $l * \dim(\mathbf{y}_i)$, where l is the sequence length) is a contextualized vector for each element of a sequence. This is an advantage in our scenario because experts can more easily interpret our output since they can cast it to regular aggregates.

This 'constrained attention' not only makes it lighter (less computations and outputs) than regular attention but also bridges a gap between machine learning and expert work.

5 Experiments

In this section, we discuss the dataset and the features, the different feature extraction models, classifiers and the metrics used. The setup is modeled upon that used by Jurgovsky et al. [22].

5.1 Dataset

The dataset, provided to us by Wordline, consists of real-world, e-commerce credit card transactions. Our experiments are based on the data from the period of January to July 2017, consisting of over 60 million transactions. The only alternative dataset is a publicly available dataset, which however does not specify user IDs and only has a small number of transactions (around 500,000).²

The dataset provides us with the ground truth labels (fraud or not fraud) for each transaction. These were assigned by a team of human investigators, based on a set of regularly updated expert rules that raise alerts for anomalous transactions. As is the case with most datasets in fraud detection applications, the number of fraudulent transactions is far fewer than the number of genuine ones. In our dataset, we have only approximately 0.3% frauds.

The dataset is organized as individual temporally ordered transactions. However, since our models require sequences of transactions, we organize the data into card-holder accounts. We do this by grouping transactions by card-holder ID and sort the transactions in each group by time. Here, we refer to an account as all the transactions of a particular card-holder, ordered in time.

Another factor that we must consider is that the sequence-based models can only predict on individual transactions after having seen a fixed number of transactions preceding it, which is not the case with the static learners. Hence, in order to preserve comparability across models we only predict on transactions which have a fixed number of previous transactions. We fixed a sequence length of 10, hence the number of preceding transactions must be 9. This ensures that both types of models are trained and make predictions on the same set of transactions.

² <https://www.kaggle.com/c/ieee-fraud-detection/data>.

Table 1 Overview of the different features contained in the Wordline dataset

Type	Attributes
Transaction attributes	Time (timestamp, day of week)
	Amount
	Country
	Proprietary authentication features
Merchant attributes	Card entry mode
	Merchant type
	Merchant ID
	Merchant category
Card/card-holder attributes	Merchant country
	Card expiry date
	Card limit
	Age
	Gender
	User country

They can be divided into three categories: transaction attributes, merchant attributes or attributes of the card/card-holder

5.2 Features and pre-processing

The transactions in our dataset consist of various attributes which can be grouped into different categories such as transaction related (time of the transaction, authentication mechanism, amount, etc.), card-holder related (age, gender, etc.), merchant related (merchant code and country of terminal) and the fraud label. We refer to the raw features from our dataset as the Base feature set. For an overview of the features, please refer to Table 1.

In order to assess the performance of the manual aggregate features, we define a second feature set which we refer to as Base+Aggs. This feature set consists of 13 aggregates. The time period that we consider for these aggregates is 24 h, and the categorical feature constraints include features (and combinations of them) such as the card-holder's country, the country of the terminal, the local currency, etc. based both on aggregates commonly defined in literature [8,22], and also on recommendations provided by our industrial partner.

Since our dataset is composed of a combination of continuous and categorical features, we apply different pre-processing steps before feeding them as input to our models. For the continuous features such as amount or the aggregates, we standardize them to a mean of zero and variance of one. As mentioned in Sect. 4.1, we need to encode our categorical variables in some continuous space. To do this, we use Word2vec over sequences of card-holders as shown in Russac et al. [32]. Thus, for each value of a categorical feature f , we use Word2Vec to generate an embedding vector $\mathbf{y}^f \in \mathbb{R}^{d_f}$ as its encoding. We optimize the embeddings further on the fraud task itself by using an embedding layer. However, in our experiments, this does not lead to any significant gain in performance.

5.3 Classifiers

The NAG is intended to be used as a feature extraction module, which learns features from the sequence of a card-holder in an end-to-end manner on the fraud classification task. It is a neural

network consisting of an embedding layer, a similarity encoding layer and an aggregation layer. In Fig. 1, we see the two primary components of the feature extraction module, namely the similarity encoding step and the aggregation step. We start with a sequence of transactions consisting of some constraint features (A & B in the figure) and a target feature (Z). The NAG then calculates the current-to-past similarity for each constraint feature, followed by a weighted combination of each of the feature similarities into one current-past similarity score for each transaction. In our experiments, we use a 1-d convolutional layer for the weighted combination, with the choice of the number of convolutional filters being a hyperparameter. The aggregation part, similarly is also realized through a 1-d convolutional layer for which the number of filters can be chosen as part of model selection. In addition to the feature extraction components in Fig. 1, the NAG also comprises of a decision layer, i.e., a feed-forward network. The inputs to the decision layer consist of the aggregates generated by the feature extraction module and the current transaction. All the parameters of the NAG are learned directly on the fraud labels.

We perform a two-fold comparison of the NAG. The first being a comparison against static learners that do not take previous transactions into account but are trained directly on the raw transaction features. This is our baseline. We then train these classifiers on a feature set that includes manual aggregates (Base+aggs feature set). To preserve comparability with the NAG, we use a Feed-Forward Network (FFN) as a classifier and, as an alternative, a Random Forest (RF) classifier since it is commonly used in the domain of fraud detection.

In the second comparison, we contrast the NAG with other automatic feature extraction methods (or context-based methods) that do not rely on manual feature engineering. The first being a regular feature-wise Convolutional Neural Network (CNN). Similar to the NAG, the CNN consists of a feature extraction module (1-d convolutional layers applied to each feature of the sequence of transactions) and a classification module using a feed-forward network. Additionally, we also compare it to a genuine sequence learning approach, i.e., an LSTM, which was evaluated on this task by Jurgovsky et al. [22]. The model of each neural network-based architecture is determined by optimizing the cross-entropy loss between the predicted class scores and the true labels. For the NAG, CNN and the LSTM, we base our implementation on the Pytorch library [31]. For each of the context-based methods, we use a feed-forward network as a classifier.

5.4 Model selection

For each of the classifiers, we search the space of possible hyper-parameters to find the optimal set. In our case, we perform a random search (40 iterations) over a grid of hyper-parameters and then select the model that maximizes the area under the precision-recall curve (AUC-PR) on the corresponding validation set. For each of the back-propagation-based models, we searched over different hyperparameters such as the learning rate, dropout rate, choice of optimizer and batch size. Additionally, for the LSTM, we considered the number of recurrent nodes per layer. In the case of the CNN, an additional hyperparameter was the number of filters. For the NAG, we specifically also searched for the optimal number of aggregates, which would decide the number of copies of the $\mathbf{w}$ parameters (see Eq. (6)) and $\mathbf{v}$ parameters (see Eq. (8)). The hyperparameters for the random forest were the number of estimators, max tree depth, maximum features, splitting criterion and minimum leaf samples.

5.5 Model evaluation

In order to evaluate a new method for credit card fraud detection, there are several criteria that we must consider. These criteria are based on challenges commonly applicable to the fraud detection domain that we identified based on discussions with our industrial partner. We list the criteria below along with the corresponding experiments to evaluate the NAG according to these criteria.

Modeling capabilities of the NAG As we mentioned in Sect. 3, one of the key motivations for the NAG is that other automatic feature extractors generate context representations that are unintuitive for domain experts. This serves as a key criteria for adoption of deep learning solutions for fraud detection. Therefore, the NAG is designed to mimic the structure of manual aggregate generation (see Sect. 4). In Sect. 4.5, we provided a proof to show that the NAG is able to model the ‘sum’ aggregate, given a set of categorical feature constraints. To also confirm this empirically, we define a regression task in which a given manual feature aggregate is considered as the dependent variable, with the NAG modeling its dependence on the previous transactions. For matters of comparison, we repeat the same experiment with a CNN instead. We additionally test the NAG on the smaller, publicly available dataset mentioned in Sect. 5.1.

Global evaluation on fraud detection task Naturally, the primary factor in evaluating fraud detection methods is in terms of the predictive accuracy. Machine learning fraud detection solutions provide support for human investigators to not only verify and potentially block fraudulent cards in production settings, but also for data labeling to facilitate further research. To evaluate the NAG in this regard, we define a global evaluation of the predictive accuracy of the NAG on 4 months of test data.

Daily accuracy Verification of transactions by investigators is done on a daily basis. But the fraud experts have limited budgets for such a verification. As a result, they are only able to check a subset of transactions every day. Therefore, fraud detection models are required to be accurate for what they consider as the most suspicious transactions on a daily basis as well. It also prospectively allows for a more granular analysis of the predictive capabilities of the models in relation to calendar events such as differences on weekdays/weekends or on public holidays. Therefore, we evaluate the NAG per individual test day.

Model size Another aspect that we must consider is the memory consumption of our models. There is a growing need to deploy machine learning solutions to client infrastructures in fraud detection. As online card payments have become ubiquitous across the world, there is a huge diversity in terms of the computational resources and connectivity available to a client. For deep learning models, one can evaluate the memory consumption in a backend-agnostic manner by looking at the size of the network in terms of the numbers of parameters. Using more parameters would increase GPU memory consumption during both training and inference as it would require assignment of memory for the storage of parameters, gradients, etc. Therefore, choosing a suitable approach for fraud detection requires considering the trade-offs between the size of the model and its predictive performance. To evaluate the performance of the NAG at different model sizes, we define a parameter budget study.

Context size Based on other studies using sequence models on Wordline data and in consultation with experts at Wordline, we set an upper limit for the context size/sequence length

Table 2 Dataset splits

Training period	Validation period	Test period
24.01–24.03	25.03–31.03	01.04–30.04
23.02–23.04	24.04–30.04	01.05–31.05
24.03–24.05	25.05–31.05	01.06–30.06
23.04–23.06	24.06–30.06	01.07–31.07

to 10 [22]. The larger the chosen sequence length, the lesser the number of users in our train and test data. This is due to the fact that we only train and test on the subset of users who have large enough sequences of transactions. This excludes users with newly created accounts, who have not yet made 10 transactions and additionally those who make payments infrequently and therefore do not have 10 transactions in the given train or test period. However, using longer sequences provides models with additional context for a particular transaction. Therefore, it would be of added benefit for a model to show good predictive performance even for sequences that are shorter than 10 transactions. In order to address this, we additionally evaluate the NAG using shorter sequence lengths. We evaluate the NAG on the fraud task using sequence lengths of 5 and 7 transactions, respectively.

Our data splits (given in Table 2) are based on practical considerations. In production settings, due to the time it takes for human investigators to review the transactions, the labels are only available after a one-week delay. Therefore, for our experiments, there is a gap of 7 days between our training and test period. We use these 7 days as the validation period. As discussed in Sect. 2, although there are different choices of precision-based metrics used for fraud detection models, AUC-PR has been used in several works, including those using Wordline datasets. Unlike the AUC-ROC, the AUC-PR is robust to class imbalance [15]. We note here that the baseline performance (i.e., using a random or ‘coin flip’ classifier) would give us an AUC-ROC of 0.5. However, for AUC-PR, the baseline is equal to the fraction of the positive class in the dataset. In the case of credit card data, since only a very small fraction of transactions (0.3% in our case) is fraudulent, the baseline performance measured in AUC-PR is expected to be 0.003, which is very low. For a more detailed discussion, we would like to point our reader to the work by Saito and Rehmsmeier [34].

6 Results

In this section, we present the results of our experiments.

We conduct an initial experiment in order to evaluate if the NAG is able to model a given manual feature aggregate. Next, based on the criteria given in Sect. 5.5, we conduct a number of experiments to evaluate the NAG. Firstly, we evaluate the performance of the NAG on individual test days compared to both a static approach (FFN with manual aggregates) and a context-based approach (the LSTM). We show that the difference in performance between the NAG and the other methods is statistically significant. Secondly, we conduct a parameter budget study. Further, we study the impact of using shorter transaction histories on the performance of the NAG. The NAG also provides the scope for further interpretability through inspection of its parameters which we elaborate on in Sect. 7.

Table 3 Results of regression experiment ($n = 5$)

Constraints	MAE CNN	MAE NAG
<i>Public data</i>		
card2	113.05 $\pm$ 32.53	54.91 $\pm$ 4.29
card3	93.77 $\pm$ 49.59	51.53 $\pm$ 3.05
card2 + card3	96.81 $\pm$ 24.14	57.18 $\pm$ 8.68
card2+ card5	114.53 $\pm$ 34.19	61.28 $\pm$ 2.33
card3 + card5	121.07 $\pm$ 21.93	66.42 $\pm$ 3.22
<i>Private data</i>		
Merchant code	150.79 $\pm$ 3.27	67.20 $\pm$ 0.93
Country	157.05 $\pm$ 4.29	88.72 $\pm$ 0.92
Merchant code + country	150.12 $\pm$ 2.20	65.27 $\pm$ 3.09
Merchant code + card entry mode	152.28 $\pm$ 5.97	59.61 $\pm$ 1.43
Card entry mode + country	157.74 $\pm$ 8.56	78.58 $\pm$ 0.40

6.1 Approximating manual aggregates

For this experiment, we evaluate the NAG both on a Wordline dataset and on a public dataset.³ For the public data, one major hurdle was that the transactions did not have card-holder or user IDs. We use the categorical features card2, card3 and card5 as our constraint features. In order to process the dataset as user-accounts, we need each transaction to be associated with a particular user. Hence, we use one of the solutions from the participants of this challenge to approximate user IDs for each of the transactions in the dataset.⁴ While such an approximate assignment of IDs would be unsuitable for our fraud classification task, in this case since our target function itself is a particular aggregation function along with the selection mechanism and not to learn user-specific behavior in the context of fraud, the accurate assignment of user IDs is not essential.

The only alteration we made to the aggregate creation was that instead of considering a particular time window, we fix the number of transactions that the aggregates take into account as the same as our input sequence length. This is to ensure that both our target aggregate and our model input spans across the same range of transactions. As a cost function, we use the mean absolute error (MAE) between the predicted value and the true aggregate value. In our case, the aggregate function that we used was always the ‘sum’ function. We report the approximation error in terms of the mean absolute error (i.e., Euros) on the test set in Table 3. As we can see, for the choice of single or multiple features as constraints, the NAG predictions are closer to the target value on average, for both the public and private data. This is encouraging since it indicates that the NAG improves on a generic feature extraction model in terms of learning the selection criteria of the aggregates. On the other hand, for the CNN, while it is able to optimize our cost function to a certain extent, it seems that it fails to learn this selection mechanism for each of our chosen constraints.

³ <https://www.kaggle.com/c/ieee-fraud-detection/data>.

⁴ <https://www.kaggle.com/c/ieee-fraud-detection/discussion/111510>.

6.2 Global evaluation on fraud detection task

The results on the fraud task are summarized in Table 4, giving the mean AUC-PR across 5 runs for each combination of model and test period. For the NAG and the LSTM, we report the performance of the models with 350,000 parameters each as given in Sect. 6.4. Further increases in model size do not significantly improve the performance in either case. The first observation we make is that the results differ for each of the months. For example, the performance in April is higher than all the other months by a large margin. Since these differences in performance are consistent across all models that we used, we attribute it to differences in the underlying data distribution of each of the months. As mentioned in Sect. 5.5, the reason for the relatively low AUC-PR scores across all models is primarily due to the class imbalance in the dataset. The precision is high for low levels of recall (i.e., each model is able to identify the ‘obvious’ frauds), but for higher levels of recall, the precision decreases as there are more false positives due to the sheer number of genuine transactions in our dataset. We also re-affirm the observation from literature that aggregates improve fraud detection accuracy. We see an increase in AUC-PR when we use manual aggregates in the static learning scenario, for all months of testing. The increase in performance is more pronounced for certain combinations of test period and model (e.g., relatively small increase for the random forest in June). Our third observation is that if we compare the NAG to both the static learners, we see that even with aggregated features, the NAG yields improved performance across all test months. This indicates that the NAG is a viable approach to replace manual aggregates.

Next we come to the comparison with the two other automatic context representation approaches (see *context* rows in Table 4). We see that for all months of testing, the NAG once again has a higher performance.

6.3 Daily predictions

In this subsection, we evaluate the NAG on the individual days of testing. Unlike the results in Table 4 where we calculate the score for all transactions in a particular month, here we compute the AUC-PR for each day. The performance across days is illustrated in Fig. 2. Here we compare the NAG to one static and context-based approach, namely the LSTM and the feed-forward network with Base + Aggs as its feature set.

One of the first aspects one notices in the figure is that there is a lot of variation in performance across different days of the test period. One might assume that the frauds on these days are harder to identify, especially since all three curves are correlated indicating that all three models struggle on some days more than others. While we cannot eliminate this as a factor, a more direct line can be drawn to the variation in the proportion of frauds across the days. As explained by Siblini et al. [36], precision-based metrics are tied to the class prior. Therefore on days with higher densities of fraudulent transactions, the AUC-PR score tends to be higher. For a more in-depth analysis, we would point the reader to the aforementioned work.

We compare the results presented in the Fig. 2 with the help of a *Friedman/Nemenyi* test [12] as shown in Fig. 3. The score for each day is obtained by averaging across 5 runs. We end up with 122 samples for each model. The significance level of the tests is $\alpha = 0.05$. The mean ranks of the NAG, LSTM and FFN are 1.32, 2.20 and 2.48, respectively. We reject the null-hypothesis of the Friedman test with $p < 0.001$. Based on the post-hoc Nemenyi test, one can assume that the daily performance of two methods is significantly different if the difference between the average ranks is larger than the critical difference (*CD*). Here the *CD*

Table 4 Predictive performance (AUC-PR) on the fraud classification task

	April	May	June	July
Static	FFN (base)	0.230 ± 0.001	0.168 ± 0.003	0.126 ± 0.001
	FFN (base + aggs)	0.300 ± 0.030	0.205 ± 0.024	0.157 ± 0.006
	RF (base)	0.251 ± 0.001	0.157 ± 0.002	0.121 ± 0.001
	RF (base + aggs)	0.308 ± 0.002	0.221 ± 0.004	0.138 ± 0.001
	LSTM (base)	0.326 ± 0.007	0.225 ± 0.003	0.154 ± 0.003
Context	CNN (base)	0.290 ± 0.011	0.199 ± 0.007	0.149 ± 0.007
	NAG (base)	0.338 ± 0.008	0.246 ± 0.007	0.201 ± 0.006
				0.235 ± 0.020

Comparison of the NAG against static classifiers and context classifiers. The input of static classifiers (random forest, feed-forward neural network) is an individual transaction; the input of context classifiers includes, in addition, the 9 transactions preceding the transaction. Scores are reported as mean over five repeated runs on test transactions from April, May, June and July 2017. The feature sets used are given in parentheses

The results marked in bold indicate the best performance for a particular month

Fig. 2 AUC-PR evolution across days for the NAG, LSTM and FFN from April–July 2017. Each score is calculated as the mean score across 5 runs

Fig. 3 Friedman/Nemenyi test based on the daily AUC-PR scores. A bold black bar connecting two methods would indicate that they are not significantly different. The NAG is shown to be significantly different than both the other methods. α is set to 0.05. The critical difference is computed as 0.3008

is computed to be 0.3008. We can see in Fig. 3 that while the NAG is significantly different from the LSTM and FFN, there is no significant difference between the latter two methods.

6.4 Parameter budget study

In this part of our study, we perform a parameter budget comparison between the NAG and LSTM as it is the state-of-the-art in sequence classification for fraud detection. Here, we consider the total parameters stemming from both the sequence representation and classification module. We use pre-trained word embeddings as inputs to both models, which we do not further optimize. For the NAG, we modulate the number of parameters by altering the number of copies of the $\mathbf{w}$ (see Eq. (6)) and $\mathbf{v}$ (see Eq. (8)). Similarly, for the LSTM we test the performance at different dimensionalities of the hidden state. We fix the size of the feed-forward network to the same for both models as we observed during our experiments that further increases in the representational capacity of the feed-forward network do not significantly improve the performance in either approach. Since the architecture of both models do not resemble one another it was not feasible to have pairwise comparisons with exactly the same number of parameters. Therefore, we use the concept of ‘parameter budgets’. This entails testing the models at different levels (budgets), where the number of parameters of

each model is approximately and not exactly equal to the budget. We start with a budget of 50,000 parameters and go up to 400,000 parameters.

We present our results in Fig. 4 for the test period of April to July 2017. In each sub-figure we report the mean AUC-PR across 5 runs for each combination of model and parameter budget. We observe that for each month, the NAG outperforms the LSTM at each budget with differing magnitudes. This difference is especially stark in June and July. While in each of the months the performance of both the LSTM and the NAG drop sharply at the lowest budget (50,000 parameters), the NAG's scores are still comparable to or improves over the performance of each of the models reported in 3. The performance of the NAG at even the smallest budget improves over the LSTM's highest scores in three out of the four months of testing. This would indicate higher flexibility in deploying the NAG in scenarios which demand more constrained solutions.

6.5 Impact of shorter sequences

Since our results in Table 4 were limited to sequences with a length of 10 transactions, we further evaluated the NAG by comparing it to the LSTM at different sequence lengths. We evaluate both models with inputs of transaction sequences of length 5 and 7. As we use shorter sequence lengths, we are able to train and test on more card-holders. As an example, for the test month of July, using sequences of 5 and 7 transactions leads to an increase of 90% and 211% in terms of the number of card-holders in the test set, respectively, compared to using sequences of 10 transactions.

The results are presented in Table 5. Here, once again, we report the mean AUC-PR across 5 runs for each model, period of testing and sequence length.

We see that for the shorter sequences, while the performance of the NAG drops, it still outperforms the LSTM. In Table 6 we show the average gain for each model in predictive performance for longer sequences. We see that the NAG has larger gains in performance. This could be an indication that the NAG, through it's soft-similarity scores, is able to extract predictive context from older transactions that the LSTM is unable to track.

7 Discussion on interpretability

Neural networks, while being powerful function approximators, provide a black box solution to a given problem; i.e., they do not give much insight into the function that is being approximated. This results in more reluctance in adopting these approaches for a use-case where a huge amount of expenses is at stake. Therefore, fraud detection providers often turn toward more 'interpretable' models such as decision trees. The NAG on the other hand has a number of assumptions built into it based on our analysis of manual feature engineering that allows us to use only a small number of parameters and still achieve comparable performance to, for instance, an LSTM. The fact that the architecture of the NAG closely mimics the structure of manual aggregates provides us with the scope of analyzing the parameters, in a more intuitive manner than a CNN or LSTM. In Fig. 5 we display the 16 parameter vectors $\mathbf{w}$ (see Eq. 6) of a fitted NAG model as columns in a heatmap. Each component in any of the $\mathbf{w}$'s (*filters*) weighs the relative contribution of a feature-wise similarity to the overall transaction similarity. These 16 filters comprise a total of 192 parameters and they are shared across all transactions and sequences.

Fig. 4 Parameter budget study: we compare the performance of the NAG and the LSTM at different model sizes. The x-axis corresponds to the total number of parameters of a model, which comprises of a sequence representation module and the classification module. The y-axis corresponds to the AUC-PR. Each point is the average AUC-PR across five runs at a particular parameter budget

Table 5 Predictive performance (AUC-PR) on the fraud classification task for sequence lengths of 5 and 7

Sequence length	Model	April	May	June	July
5	NAG	0.313 ± 0.008	0.219 ± 0.004	0.192 ± 0.007	0.216 ± 0.017
	LSTM	0.299 ± 0.015	0.208 ± 0.009	0.144 ± 0.002	0.180 ± 0.009
7	NAG	0.342 ± 0.003	0.231 ± 0.006	0.206 ± 0.003	0.224 ± 0.019
	LSTM	0.307 ± 0.002	0.210 ± 0.004	0.149 ± 0.006	0.179 ± 0.008

Scores are reported as the mean over five repeated runs on test transactions from April, May, June and July 2017

The results marked in bold indicate the best performance for a particular month and sequence length

We observe that across all the filters some features have very little contribution to the overall transaction similarity. These are transaction-based features relating to the nature of the security of the transaction, or to whether it's an international transaction, etc. Another observation is that the merchant category and terminal ID along with the time features carry a much higher absolute contribution. This pattern indicates that the NAG learned to focus on two key aspects when determining the transaction similarity: The transaction similarity is predominantly governed by how similar the merchants are and how similar the timestamps are—regardless of the exact features used to represent a merchant or timestamp. From fraud experts and our own analyses, we know that the number of transactions occurring in the immediate past is positively correlated with fraud and it seems that the $\mathbf{w}_i$ filters are picking up on these patterns. Similarly, we know that the strength of security mechanisms at online payment terminals are determined by the merchants, and fraudsters tend to specifically target vulnerable portals, thus highlighting the relevance of such merchant oriented features.

The heatmap in Fig. 6 displays the 16 parameter vectors $\mathbf{v}_i$ (see Eq. 8) which comprise a total of 160 parameters. The components in each of these vectors modulate the contribution of previous transactions to the overall aggregate value. Even though all $\mathbf{v}_i$'s differ in their

Table 6 Average gain in predictive performance (AUC-PR) due to increase of sequence length on the fraud classification task across months

Sequence length	NAG (%)	LSTM (%)
5 to 7	6.43	1.76
5 to 10	8.07	6.5

Fig. 5 Weights for the feature-wise similarity scores to generate the transaction similarity of a fitted NAG: the columns correspond to the 16 weight vectors $\mathbf{w}_i$, and the rows correspond to the 12 categorical features**Fig. 6** Weights modulating the aggregation of the amount variable of a fitted NAG: the columns correspond to the 16 weight vectors $\mathbf{v}_i$, and the rows correspond to the index of a transaction within a sequence ("1" - the current transaction, "10" - the oldest transaction)

ℓ_1 -norm, the weights show some interesting patterns. For the bottom row, we get weights of small negative values close to zero. This fits with our intuition because it shows that our weighting should not take the similarity of the current transaction with itself into much account. We observe that most of the filters learn higher weights for more recent transactions. In some cases, the weights decrease gradually as we go further back along the sequence such as in $\mathbf{v}_1$, $\mathbf{v}_3$ and $\mathbf{v}_{13}$, while $\mathbf{v}_5$ and $\mathbf{v}_6$ drop off steeply after the more recent transactions. $\mathbf{v}_7$, $\mathbf{v}_8$ and $\mathbf{v}_{15}$ seem to group the past into blocks, with more emphasis on the recent block and less on the older ones. These observations confirm that more recent transactions contribute heavily to the context that is being learned. It highlights an advantage of the NAG, since such weighting was not possible for our manual aggregates. They consider a fixed time window of past transactions, where the order is then irrelevant. Some of the other vectors ($\mathbf{v}_9$, $\mathbf{v}_{10}$ and $\mathbf{v}_{12}$ for example) give us an estimate of the average behavior, since all amounts are weighted similarly.

8 Conclusion and future research

In this paper, we described a novel automatic feature aggregation framework based on our analysis of the structure and limitations of manually engineered aggregates in the domain of credit card fraud detection. We used the NAG to represent the historical context of cardholders as features for fraud classification and evaluated it on a real-world dataset with millions of transactions. As a baseline, we compared it to static approaches which do not take into account contextual information and showed that the NAG improves on the prediction accuracy by a large margin. Our results also showed that the NAG outperforms both static learners that use manual feature aggregates and other contextual models. We used a *Friedman–Nemenyi* test on daily predictions to show the NAG's performance to be significantly different than the feed-forward network and LSTM. Following this, we evaluated the NAG in terms of the trade-off between predictive performance and model size through a parameter budget study. The results showed that even with a small number of parameters the NAG shows improved performance to other approaches. We also evaluated the impact of using shorter lengths of the transaction sequences, with the NAG outperforming the LSTM at each chosen sequence length. Finally, we highlighted the advantage of the NAG to other feature extraction methods of being more interpretable. Through inspection of its parameters, we identified several patterns that provide insights into the relative contribution of different categorical features in calculating transaction similarities and also the focus on more recent transactions as part of the aggregate computation.

In terms of the next steps, we plan to expand the expressiveness of the NAG. At present, the NAG is designed to learn lower-order statistics on the amounts in the transaction history such as a count or a sum. We want to extend this by learning other functions such as the minimum or maximum amounts (as mentioned in Sect. 4.5), given a set of constraints. Additionally, our goal is to enable the NAG to learn the variance and higher-order statistics such as the skewness of the target variable distribution. Another possible continuation of our work is to apply the NAG to a new domain. This includes tasks such as intrusion detection or network data-based malware detection, which also involve tabular, time-ordered data. This would also potentially allow the evaluation of the NAG on publicly available datasets, as the availability of the same is quite limited in the credit card fraud domain.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Alazizi, A Habrard A, Jacquenet F, He-Guelton L, Oblé F, Siblini W (2019) Anomaly detection, consider your dataset first an illustration on fraud detection. In: 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI), pp 1351–1355. <https://doi.org/10.1109/ICTAI.2019.00188>
2. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. *J Mach Learn Res* 3(Feb):1137–1155. <https://doi.org/10.5555/944919.944966>

3. Bhattacharyya S, Jha S, Tharakunnel K, Westland JC (2011) Data mining for credit card fraud: a comparative study. *Decis Support Syst* 50(3):602–613. <https://doi.org/10.1016/j.dss.2010.08.008>
4. Braun F, Caelen O, Smirnov EN, Kelk S, Lebichot B (2017) Improving card fraud detection through suspicious pattern discovery. In: Benferhat S, Tabia K, Ali M (eds) *Advances in artificial intelligence: from theory to practice*. Springer International Publishing, Cham, pp 181–190. ISBN 978-3-319-60045-1. https://doi.org/10.1007/978-3-319-60045-1_21
5. Breiman L (1996) Bagging predictors. *Mach Learn* 24(2):123–140. <https://doi.org/10.1007/BF00058655>
6. Carcillo F, Borgne Y-AL, Caelen O, Kessaci Y, Oblé F, Bontempi G (2021) Combining unsupervised and supervised learning in credit card fraud detection. *Inf Sci* 557:317–331. <https://doi.org/10.1016/j.ins.2019.05.042>
7. Cheng D, Xiang S, Shang C, Zhang Y, Yang F, Zhang L (2020) Spatio-temporal attention-based neural network for credit card fraud detection. In: *Proceedings of the AAAI conference on artificial intelligence*, vol 34, pp 362–369. <https://doi.org/10.1609/aaai.v34i01.5371>
8. Bahnsen AC, Aouada D, Stojanovic A, Ottersten B (2016) Feature engineering strategies for credit card fraud detection. *Expert Syst Appl* 51:134–142. <https://doi.org/10.1016/j.eswa.2015.12.030>
9. Pozzolo AD (2015) Adaptive machine learning for credit card fraud detection
10. Pozzolo AD, Boracchi G, Caelen O, Alippi C, Bontempi G (2018) Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE Trans Neural Netw Learn Syst* 29(8):3784–3797. <https://doi.org/10.1109/TNNLS.2017.2736643>
11. Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: *Proceedings of the 23rd international conference on machine learning, ICML'06*, New York, NY, USA, pp 233–240. Association for Computing Machinery. ISBN 1595933832. <https://doi.org/10.1145/1143844.1143874>
12. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7:1–30. <https://doi.org/10.5555/1248547.1248548>
13. Dhankhad S, Mohammed E, Far B (2018) Supervised machine learning algorithms for credit card fraudulent transaction detection: a comparative study. In: *2018 IEEE international conference on information reuse and integration (IRI)*, pp 122–125. <https://doi.org/10.1109/IRI.2018.00025>
14. Elsayed T, Lin J, Oard DW (2008) Pairwise document similarity in large collections with mapreduce. In: *Proceedings of ACL-08: HLT, short papers*, pp 265–268. <https://doi.org/10.5555/1557690.1557767>
15. Fawcett T (2003) Notes and practical considerations for data mining researchers. Hewlett-Packard Company, Palo Alto
16. Fu K, Cheng D, Tu Y, Zhang L (2016) Credit card fraud detection using convolutional neural networks. In: Hirose A, Ozawa S, Doya K, Ikeda K, Lee M, Liu D (eds) *Neural information processing*. Springer International Publishing, Cham, pp 483–490. ISBN 978-3-319-46675-0. <https://doi.org/10.1007/978-3-319-46675-0>
17. Ghosh S, Reilly DL (1994) Credit card fraud detection with a neural-network. In: *1994 Proceedings of the twenty-seventh hawaii international conference on system sciences*, vol 3, pp 621–630. <https://doi.org/10.1109/HICSS.1994.323314>
18. Guo C, Berkahn F (2016) Entity embeddings of categorical variables. *CoRR* abs/1604.06737. [arXiv:1604.06737](https://arxiv.org/abs/1604.06737)
19. Gómez JA, Arévalo J, Paredes R, Nin J (2018) End-to-end neural network architecture for fraud scoring in card payments. *Pattern Recognit Lett* 105:175–181. <https://doi.org/10.1016/j.patrec.2017.08.024>
20. Hordri NF, Yuhaniz SS, Azmi NFM, Shamsuddin SM (2018) Handling class imbalance in credit card fraud using resampling methods. *Int J Adv Comput Sci Appl*. <https://doi.org/10.14569/IJACSA.2018.091155>
21. Jurgovsky J (2019) Context-aware credit card fraud detection. PhD thesis, Universität Passau
22. Jurgovsky J, Granitzer M, Ziegler K, Calabretto S, Portier PE, He-Guelton L, Caelen O (2018) Sequence classification for credit-card fraud detection. *Expert Syst Appl* 100:234–245. <https://doi.org/10.1016/j.eswa.2018.01.037>
23. Kandola J, Cristianini N, Shawe-Taylor JS (2003) Learning semantic similarity. In *Advances in neural information processing systems*, pp 673–680. <https://doi.org/10.5555/2968618.2968702>
24. Kim E, Lee J, Shin H, Yang H, Cho S, Nam SK, Song Y, Yoon JA, Kim JI (2019) Champion-challenger analysis for credit card fraud detection: hybrid ensemble and deep learning. *Expert Syst Appl* 128:214–224. <https://doi.org/10.1016/j.eswa.2019.03.042>
25. Kolodiziev O, Mints A, Sidelov P, Pleskun I, Lozynska O (2020) Automatic machine learning algorithms for fraud detection in digital payment systems. *East Eur J Enterp Technol* 5(9):107. <https://doi.org/10.15587/1729-4061.2020.212830> <https://doi.org/10.15587/1729-4061.2020.212830> <https://doi.org/10.15587/1729-4061.2020.212830>
26. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37. <https://doi.org/10.1109/MC.2009.263>

27. Lucas Y, Jurgovsky J (2020) Credit card fraud detection using machine learning: a survey. CoRR abs/2010.06479 [arXiv:2010.06479](https://arxiv.org/abs/2010.06479)
28. Lucas Y, Portier P-E, Laporte L, Calabretto S, Caelen O, He-Guelton L, Granitzer M (2019) Multiple perspectives hmm-based feature engineering for credit card fraud detection. In: Proceedings of the 34th ACM/SIGAPP symposium on applied computing, SAC'19. New York, NY, USA, pp 1359–1361. Association for Computing Machinery. ISBN 9781450359337. <https://doi.org/10.1145/3297280.3297586>
29. Lucas Y, Portier P-E, Laporte L, Calabretto S, He-Guelton L, Oblé F, Granitzer M (2019) Dataset shift quantification for credit card fraud detection. In: 2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE), pp 97–100. <https://doi.org/10.1109/AIKE.2019.00024>
30. Nami S, Shajari M (2018) Cost-sensitive payment card fraud detection based on dynamic random forest and k-nearest neighbors. Expert Syst Appl 110:381–392. <https://doi.org/10.1016/j.eswa.2018.06.011>
31. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems 32. Curran Associates, Inc, pp 8024–8035. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
32. Russac Y, Caelen O, He-Guelton L (2018) Embeddings of categorical variables for sequential data in fraud context. In: International conference on advanced machine learning technologies and applications. Springer, pp 542–552. https://doi.org/10.1007/978-3-319-74690-6_53
33. Ryman-Tubb NF, Krause P, Garn W (2018) How artificial intelligence and machine learning research impacts payment card fraud detection: a survey and industry benchmark. Eng Appl Artif Intell 76:130–157. <https://doi.org/10.1016/j.engappai.2018.07.008>
34. Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE 10(3):e0118432. <https://doi.org/10.1371/journal.pone.0118432>
35. Shen T, Zhou T, Long G, Jiang J, Pan S, Zhang C (2018) Disan: directional self-attention network for rnn/cnn-free language understanding. In: Proceedings of the AAAI conference on artificial intelligence, vol 32. <https://ojs.aaai.org/index.php/AAAI/article/view/11941>
36. Siblini W, Fréry J, He-Guelton L, Oblé F, Wang Y-Q (2020) Master your metrics with calibration. In: Berthold MR, Feelders A, Krempel G (eds) Advances in intelligent data analysis XVIII. Springer International Publishing, Cham, pp 457–469. ISBN 978-3-030-44584-3. https://doi.org/10.1007/978-3-030-44584-3_36
37. Song H, Rajan D, Thiagarajan J, Spanias A (2018) Attend and diagnose: clinical time series analysis using attention models. In: Proceedings of the AAAI conference on artificial intelligence, vol 32. <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16325>
38. Srivastava A, Kundu A, Sural S, Majumdar A (2008) Credit card fraud detection using hidden Markov model. IEEE Trans Dependable Secure Comput 5(1):37–48. <https://doi.org/10.1109/TDSC.2007.70228>
39. Sánchez D, Vila MA, Cerda L, Serrano JM (2009) Association rules applied to credit card fraud detection. Expert Syst Appl 36(supp 2):3630–3640. <https://doi.org/10.1016/j.eswa.2008.02.001>
40. Van Vlasselaer V, Bravo C, Caelen O, Eliassi-Rad T, Akoglu L, Snoeck M, Baesens B (2015) Apate: a novel approach for automated credit card transaction fraud detection using network-based extensions. Decis Support Syst 75:38–48. <https://doi.org/10.1016/j.dss.2015.04.013>
41. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Łukasz Kaiser, Polosukhin I (2017) Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30. Curran Associates, Inc., pp 5998–6008. <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>
42. Whitrow C, Hand DJ, Juszczak P, Weston D, Adams NM (2009) Transaction aggregation as a strategy for credit card fraud detection. Data Min Knowl Discov 18(1):30–55. <https://doi.org/10.1007/s10618-008-0116-z>
43. Zareapoor M, Shamsolmoali P (2015) Application of credit card fraud detection: based on bagging ensemble classifier. Procedia Comput Sci 48:679–685. <https://doi.org/10.1016/j.procs.2015.04.201>
44. Ziegler K, Caelen O, Garchery M, Granitzer M, He-Guelton L, Jurgovsky J, Portier P-E, Zwicklbauer S (2017) Injecting semantic background knowledge into neural networks using graph embeddings. In: 2017 IEEE 26th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE). IEEE, pp 200–205. <https://doi.org/10.1109/WETICE.2017.36>

Kanishka Ghosh Dastidar After completing his Master's degree in Computer Science at the University of Passau, Kanishka began working as a scientific employee at the Chair of Distributed Information Systems at the same University. His research primarily focuses on using machine learning approaches to learn tackle the problem of fraud in credit card payments. Additionally, his research interests include representation learning and interpretable AI.

Johannes Jurgovsky (Ph.D.) graduated in the field of Computer Science at the University of Passau. He obtained his Ph.D. from the University of Passau and INSA de Lyon on the topic of credit card fraud detection. He published several papers on semantic knowledge extraction and fraud detection. As a Machine Learning engineer, he is currently developing effect estimation and forecasting algorithms for the logistics industry. His main research interests include sequence learning, anomaly detection and causal inference.

Wissam Siblini After obtaining a Master of Engineering Degree from Ecole Centrale de Nantes (France) and a Master of Science Degree from the University Claude Bernard in Lyon (France), he did a Ph.D. thesis at the University of Nantes in partnership with Orange Labs in Lannion (France). His research focused on extreme multi-label classification and dimensionality reduction. He completed his Ph.D. in 2018, and he started doing applied research at Wordline on Fraud Detection, Natural Language Processing, and Computer Vision.

Michael Granitzer holds the Chair of Data Science at University of Passau since 2017 and has been Professor for Media Computer Science at University of Passau since 2012. Before, he was Scientific Director of the Know-Center Graz and assistant professor at the Knowledge Management Institute of Graz University of Technology. His research interests focus on applied machine learning in different application contexts like fraud detection or media analysis, natural language processing, information retrieval and network science. He has published over 200 peer-reviewed publications in these fields.