Supporting Complaint Management in the Medical Technology Industry by Means of Deep Learning

Philip Hake[1], Jana-Rebecca Rehse[1] and Peter Fettke[1]

[1] Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI GmbH) and Saarland University, Campus D3.2, Saarbrücken, Germany
{philip.hake, jana-rebecca.rehse, peter.fettke}@dfki.de

Abstract. Complaints about finished products are a major challenge for companies. Particularly for manufacturers of medical technology, where product quality is directly related to public health, defective products can have a significant impact. As part of the increasing digitalization of manufacturing companies ("Industry 4.0"), more process-related data is collected and stored. In this paper, we show how this data can be used to support the complaint management process in the medical technology industry. Working together with a large manufacturer of medical products, we obtained a large dataset containing textual descriptions and assigned error sources for past complaints. We use this dataset to design, implement, and evaluate a novel approach for automatically suggesting a likely error source for future complaints based on the customer-provided textual description. Our results show that deep learning technology holds an interesting potential for supporting complaint management processes, which can be leveraged in practice.

Keywords: Complaint Management, Quality Management, Process Prediction, Machine Learning, Deep Learning

1 Motivation

For manufacturing companies, complaints about finished products are a major challenge [1]. They not only reduce profits, but also cause additional costs. Complaint management requires time and personnel resources. In addition, regular complaints about quality defects have a lasting negative effect on customer loyalty and therefore on a company’s reputation. For manufacturers of medical technology, which must meet special quality requirements in the regulated environment, defective products can be particularly damaging [2]. The new EU regulation on medical devices has further increased the requirements on quality and safety of medical technology [3]. In this industry, lawmakers consider product quality as directly related to public health. Quality defects therefore risk a company's success both by a decline in sales and by official interventions. According to legal requirements, medical technology manufacturers must establish a prompt and consistent approach to the acceptance, assessment, and investigation of complaints and the decision on follow-up measures.

In the context of the ongoing digitalization of the economy in general and manufacturing companies in particular ("Industry 4.0"), more and more process- and production-relevant data is recorded and stored [4].The considerable amount of real-time sensor, machine, and process data from product lifecycle (PLC), manufacturing execution (MES), and enterprise resource planning (ERP) systems can be further enriched with data from the systems used for complaint and error handling processes as well as customer-related data. This data holds great potential for improved complaint management [5]. For example, it can be used to train a machine learning approach that provides automated support for repetitive, but time-critical process steps.

In this paper, we introduce a new approach to take a first step towards process automation. Working together with a large manufacturer of medical products, we obtained a large dataset containing textual descriptions and assigned error sources for past complaints. We use this dataset to design, implement, and evaluate a novel approach for automatically suggesting a likely error source for future complaints based on the customer-provided textual description. The approach makes use of state-of-the-art deep learning technology, which has already been used for natural language processing (NLP) in other application domains. For this purpose, the paper is organized as follows. In Sect. 2, we report on the foundations of medical technology quality management and related work on machine learning in business process management (BPM). Our suggested solution design and architecture are described in Sect. 3. Sect. 4 reports on how our suggested approach was realized and evaluated. We discuss implications and challenges in Sect. 5, before concluding the paper in Sect. 6.

2 Preliminaries

2.1 Quality Management in Medical Technology

Both processes and products in the regulated environment are subject to high quality requirements, summarized by the term GMP (Good Manufacturing Practice). These binding quality requirements result from national and international regulations (such as laws and standards) and must be considered during production [6]. GMP regulations affect central sectors of the economy, such as the pharmaceutical industry, biotechnology, medical technology, chemical industry, and food industry. Compliance with GMP regulations is of fundamental importance to companies in these industries, as they influence their manufacturing authorization. Core processes of GMP compliance and quality management include process management and document management (Standard Operating Procedures, SOPs), improvement management, corrective and preventive actions (CAPA) and controls, risk management, change management, deviations, employee training, as well as internal and external audits for GMP-relevant processes.

A central part of quality management in the regulated environment is complaint management. This is partially regulated by law. For medical device manufacturers, ISO standard 13485 (which largely conforms with ISO 9001) prescribes the use of a quality management system designed to demonstrate consistent compliance with quality standards [7]. Typically, the systems follow the 8D problem-solving process (see Fig. 1) [8]. Originally developed by automotive companies and used across many different industries, this process describes a structured approach to the identification and long-term elimination of problems and their causes and is therefore an integral part of complaint management. The collected data is an important source for internal complaint handling and identifying errors in the production process. Many steps in the 8D process have the potential to be supported by data-driven digital quality management systems.

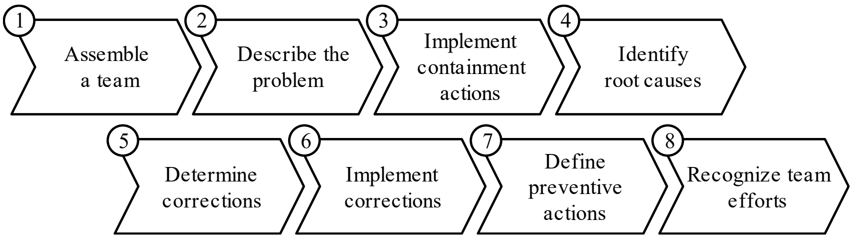


Fig. 1. The 8D reference process for complaint management

2.2 Related Work

Diverse applications for deep neural networks in BPM have recently been presented. In addition to process prediction [9–11], these include, particularly, techniques for anomaly detection [12], representation learning [13] and process modelling [14]. The authors are not aware of an application which uses deep neural networks to support complaint process management. There are several approaches that apply other machine learning techniques in quality management. Coussement et al. introduce a binary classifier that is able to distinguish complaint from non-complaint emails [1]. The approach consists of a rule-based feature extraction and a boosting algorithm for binary classification. Ko et al. deal with the detection of anomalies in engine production [15]. Their approach combines data from production across supply chains with customer data and other quality data to classify the engines' quality. The approach of Weiss et al. is also concerned with the prediction of product quality along a supply chain, but considering microprocessors [16]. In this context, the main challenge is the lengthy production process and the availability of only little measurement data. In addition, there are several approaches that develop models for quality forecasts across multiple production steps. Lieber et al. describe a case of application from the steel industry, in which the quality of interstage products is in focus [17]. Techniques of supervised and unsupervised machine learning, such as clustering or decision trees, are applied to data recorded during production (e.g., by sensors) to identify the most important factors influencing subsequent product quality. The approach of Arif et al., on the other hand, comes from the production of semiconductors, where decision trees are also used to develop a predictive model [18].

3 Conceptual Design

3.1 Challenge and Solution Design

Employees usually file 8D reports after they receive either an internal or external complaint about product quality. First and foremost, filing such a report entails recording a lot of potentially relevant data, but in a second step, the employee also has to assess the claim in terms of its criticality and the potential error source. Both determine the actions to be taken next. The criticality denotes the risk for another customer's health. If, for example, the bacterial load is too high on a previously sterilized product, it must be reported immediately to the responsible authorities in order to avoid public health risks. The potential error source is an internal assessment and the first step towards identifying and fixing the production problem, which has caused the quality complaint. Companies usually have an internal set of pre-defined error codes, which represent potential error sources. These codes vary in terms of specificity, going from a generic (e.g., "packaging error") to a more precise (e.g., "lack of maintenance on machine 5") classification, depending on the information available at the time.

Correctly assessing each filed incident is a difficult and time-consuming task, especially for less experienced employees, who might not have the necessary knowledge. Using a machine learning approach, which is able to automatically analyze all past complaints in order to assist employees in correctly assessing their incidents, may not only reduce the number of wrong assessments, but also accelerate the process, such that the issue can be fixed more quickly. For this purpose, we develop a new approach based on a deep neural network to automatically assign a likely error code to a complaint. As input, the network receives free text as recorded in the 8D report and the error code as assigned by an employee. During training, the network learns which complaint characteristics are decisive for the classification. The trained network can then automatically submit proposals for an assessment to the responsible employee for newly arriving complaints. The general network infrastructure can be transferred to also address other classification issues, such as a criticality assessment.

3.2 Solution Architecture

In order to classify textual descriptions of complaints according to their likely error source, we use a recurrent neural network (RNN) with long short-term memory (LSTM) cells [19]. RNN layer cells feed information back into themselves, evolving their state by "forgetting" or "remembering" previous inputs. Our network consists of one input layer, one or more hidden LSTM layers, and one output layer. The input layer is responsible for generating a numerical representation of the input text, a so-called embedding. We use a pretrained embedding layer of English words [20] and allow the architecture to adapt the word embeddings to the specific context during training. For the hidden layers, we use LSTM cells, because they have been found particularly suitable to manage data with long-term dependencies, such as the natural language in our textual descriptions. The output layer is a fully connected dense layer with a softmax activation, which transforms the activations of the last LSTM layer to the number of potential classes to obtain the probability distribution $\hat{y}$ over the classes.

Overall, our network architecture is a standard one for text classification problems. Our loss function L (Eq. 1), which we use for computing the gradient during training, is given by the categorical cross entropy for the expected output y and the predicted output $\hat{y}$ as well as an additional regularization loss. Given I the number of layers, C_i the number of cells in layer i and A_c the activation of cell c, the regularization loss $L1$

is defined as the sum over all activations A_c of the hidden layers (Eq. 2). By regularizing the layer activations, we intend to prevent our model from overfitting. Furthermore, we use a dropout probability for the activations of each hidden layer to approach the problem of overfitting [21].

$$L(y, \hat{y}) = -\sum_{i=0}^{C} y_i * \log(\hat{y}_i) + \lambda * L1 \tag{1}$$

$$L1 = \sum_{i=1}^{I-1} \sum_{c=1}^{C_i} |A_c| \tag{2}$$

4 Technical Realization and Evaluation

4.1 Data Characteristics and Data Preparation

To evaluate our solution, we use the complaint management data of a globally operating medical technology company. It contains 15,817 customer complaints about products, including both mass products and products manufactured according to the customer's requirements. The individual complaints contain sensitive information about the business processes and products of the manufacturer. Therefore, we cannot make the dataset publicly available. Resolving this issue would require semantically altering the data, resulting in an artificial dataset, which would counteract our goal to provide insights about the performance of machine learning in a real word business process.

Each complaint in our dataset represents a closed case. It consists of a textual description and an error code, which is manually set by the employee handling the complaint. The error code is a numerical representation of the assessment result. The dataset exhibits 186 different error codes. Table 1 compares the characteristics of the codes that occur in at least 500 cases with the remaining codes. The overview reveals that less than 6% of the codes account for more than 46% (7,311) of the cases.

Since a customer may either file a complaint by phone or by letter, the responsible employee summarizes the complaint and submits it to the information system handling the complaint process. The textual description of a complaint exhibits an average length of 122 words with a standard deviation of 124. The following description represents an exemplary complaint: "customer bought the product on 27 May 2019, he claims that the Velcro does not adhere anymore, he also claims that the problem did not occur in previous orders". The dataset contains 1,853,616 tokens, thereof 11,748 distinct words.

Table 1 depicts the number of distinct words that are contained in the textual description of the cases labeled with the same code. In addition, we provide insights on the number of distinct words that occur in cases exhibiting the same code but are not contained in any other case. Since machine learning-based classification approaches require sufficient data per class to perform well, we require a class to contain at least 500 samples to be considered for evaluation. Thus, we obtain ten classes that can be directly mapped to error codes and an additional eleventh class containing the samples of the remaining classes. Cases classified with code 1 to 10 are mapped to the respective classes 1 to 10, while the remaining cases exhibiting the codes 11 to 186 are mapped to class 0. Thus, each class is represented by at least 522 samples. To overcome the imbalance in the class distribution we randomly sample 522 cases from each class for the evaluation, resulting in an evaluation dataset of 5,742 samples.

Table 1. Dataset characteristics

class		**cases**	**distinct words**	**distinct words compared to other classes**
	all codes	15,817	11,748	
class 1	code 1	2,499	3,244	337
class 2	code 2	1,093	2,154	142
class 3	code 3	723	2,201	191
class 4	code 4	706	1,940	94
class 5	code 5	675	2,668	221
class 6	code 6	641	1,947	112
class 7	code 7	586	2,300	223
class 8	code 8	537	2,347	202
class 9	code 9	524	1,536	52
class 10	code 10	522	2,385	148
class 0	code 11-186	8,506	9,671	4,318

4.2 Evaluation Setup

To evaluate the robustness of a model, we divide the derived dataset into training, validation, and test splits. For each split, we ensure an even distribution of the eleven classes. The test split is composed of 5% (286 samples) of the dataset. From the remaining 5,456 samples, we generate 10 folds containing 4,911 training and 545 validation samples for cross validation. Table 2 shows the hyperparameters of our initial model described in section 3.2 and the respective search space, whose permutations yield a total of 32 models to evaluate. We use the training splits to optimize the loss function of the models. The optimization is conducted using a stochastic strategy called Adam [22]. In addition, we perform the incremental optimization on training batches. Each model is trained separately on all ten folds of the training split and evaluated by measuring the accuracy on the respective validation split. Its overall validation accuracy is determined as the average across the ten accuracy values. We select the model with the best validation accuracy and use it to evaluate our approach on the previously unseen test split.

Table 2: Hyperparameter search space

Hyperparameter	**Search Space**
LSTM-Layers	{1, 2}
Hidden Units	{16, 32}
L1 activity regularization (λ)	{0.00, 0.01}
Sequence Padding	{100, 200}
Training Epochs	{100, 200}
Dropout	{0.1}

Furthermore, we compare our LSTM classifier to a naïve classifier (nc) to assess the achieved performance. The naïve classifier uses a bag of words approach and the Jaccard similarity coefficient (Eq. 3) to map a sample input s to a class $i \in \{0, \ldots, 10\}$. Given a training set, the classifier generates a bag of words b_i for each class based on the words contained in the training samples labeled with class i. A sample s is assigned a class according to the maximum similarity coefficient between b_i and the Bag of Words s_{bow} derived from s (Eq. 4).

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3}$$

$$nc(s) = \min(i \mid J(b_i, s_{bow}) = \max \bigcup_{i=0}^{10} \{J(b_i, s_{bow})\}) \tag{4}$$

In the following, we report the mean training, mean validation, and mean test accuracy, as well as the standard deviation for the top model across the 10 folds. Moreover, we compare it to the mean training and test accuracy achieved by the naïve classifier. Furthermore, we report an average confusion matrix of the top model as well as the respective standard deviation.

4.3 Implementation and Results

The presented classifier is implemented in Python 3.6.7 and is online available on GitHub[1]. The LSTM model was implemented, trained and evaluated using TensorFlow[2] version 1.13 and the integrated Keras API. The training of the 32 LSTM models was conducted on a machine with a Intel Xeon W-2175 CPU 2,50 GHz (28 threads), 128 GB RAM and an Nvidia GeForce GTX Titan X GPU.

Training Evaluation. We observed a mean training accuracy of 0.87 and a standard deviation of 0.10 for the 32 LSTM models. Thus, the 32 models are generally capable to fit the training data. Table 3 shows the hyperparameter configuration of the LSTM model that achieved the best validation accuracy. The model achieved a mean training accuracy of 0.94 and standard deviation of 0.02 across the 10 folds. The model achieved a mean validation accuracy of 0.59 and a standard deviation of 0.02.

Table 3. Hyperparameter configuration for the model exhibiting best validation accuracy

Hyperparameter	Search Space
LSTM-Layers	1
Hidden Units	32
L1 activity regularization	0.00
Sequence Padding	200
Training Epochs	100
Dropout	0.1

[1] https://github.com/phakeai/aicomplaint

[2] https://www.tensorflow.org

The low standard deviations show the robustness of the model regarding the evaluation data set. Fig. 2 depicts the training history for the 100 epochs. The left curves show the mean training and validation accuracy, while the curves on the right side present the mean training and validation loss. While the training loss continuously decreases, the validation loss starts increasing from epoch 20 on. The steep training loss curve shows the capacity of the model to fit the training data. Although we apply dropout as means of regularization, we are not able to prevent the model from overfitting. Moreover, increasing the degree of regularization by adding L1 activity regularization resulted in a decreased training (0.84) and validation accuracy (0.49).

Training the naïve classifier achieved a mean training accuracy of 0.41 and a standard deviation of 0.003 on the training folds. The training accuracy shows that the model is not able to fit the training set.

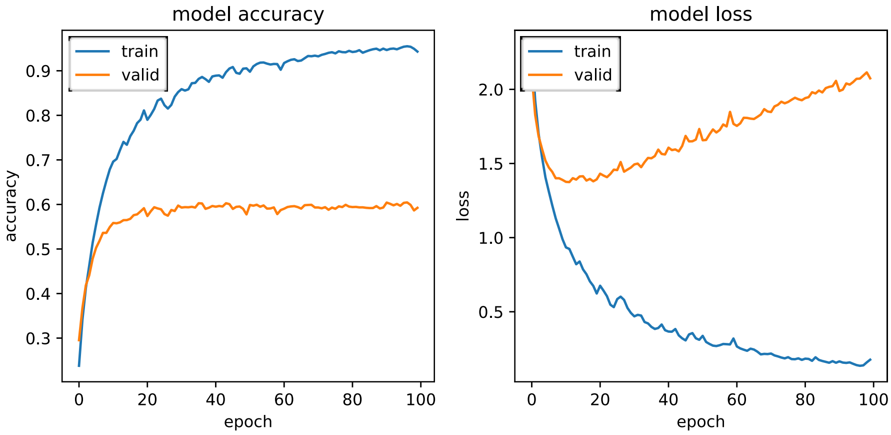
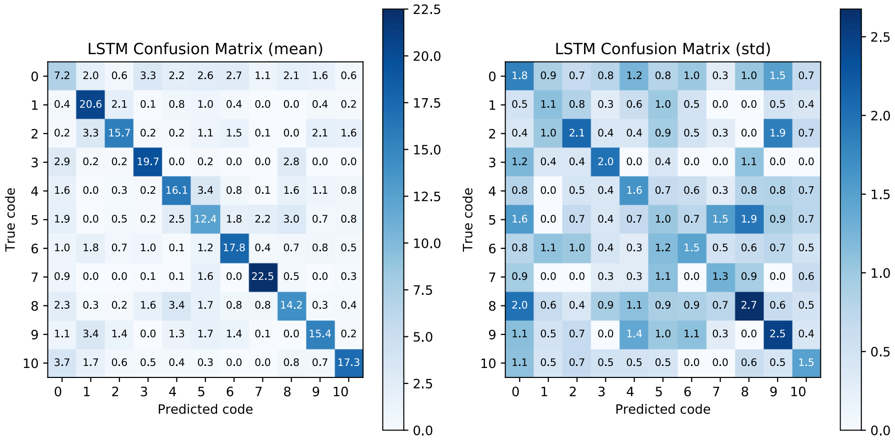


Figure 2. Accuracy and loss for training and validation

Test Evaluation. The evaluation of the naïve classifier yields an accuracy of 0.21 and a standard deviation of 0.01 on the test data. Although there are distinct words within each class, the bag of words approach appears to be unsuitable for the classification task on our evaluation dataset.

The LSTM model achieved a mean test accuracy of 0.63 and a standard deviation of 0.01 across the 10 models trained for the different folds. The small deviation from the mean accuracy confirms the observed behavior on the validation splits during training. Fig. 3 presents the confusion matrix for the classification task on the test split. For class 7, we observe the highest true positive mean. Our model correctly classifies an average of 22.5 out of 26 samples within class 7, while exhibiting a standard deviation of 1.3 samples. The lowest mean occurs in class 0, which is the class covering the complaint codes 11 to 186. With a standard deviation of 1.8, only 7.2 samples are classified correctly for this class. The first row of the matrix shows that the trained model is not able to properly distinguish class 0 from the remaining classes.

The average true positive count of the remaining classes yields 17.2 samples. Ten correctly classified samples account for 0.04 accuracy to the overall accuracy. Thus, improving the model, with the result that it matches the average performance also for class 0 samples, would lead to an improved overall accuracy of 0.67.



LSTM Confusion Matrix (mean)

True code \ Predicted code	0	1	2	3	4	5	6	7	8	9	10
0	7.2	2.0	0.6	3.3	2.2	2.6	2.7	1.1	2.1	1.6	0.6
1	0.4	20.6	2.1	0.1	0.8	1.0	0.4	0.0	0.0	0.4	0.2
2	0.2	3.3	15.7	0.2	0.2	1.1	1.5	0.1	0.0	2.1	1.6
3	2.9	0.2	0.2	19.7	0.0	0.2	0.0	0.0	2.8	0.0	0.0
4	1.6	0.0	0.3	0.2	16.1	3.4	0.8	0.1	1.6	1.1	0.8
5	1.9	0.0	0.5	0.2	2.5	12.4	1.8	2.2	3.0	0.7	0.8
6	1.0	1.8	0.7	1.0	0.1	1.2	17.8	0.4	0.7	0.8	0.5
7	0.9	0.0	0.0	0.1	0.1	1.6	0.0	22.5	0.5	0.0	0.3
8	2.3	0.3	0.2	1.6	3.4	1.7	0.8	0.8	14.2	0.3	0.4
9	1.1	3.4	1.4	0.0	1.3	1.7	1.4	0.1	0.0	15.4	0.2
10	3.7	1.7	0.6	0.5	0.4	0.3	0.0	0.0	0.8	0.7	17.3

LSTM Confusion Matrix (std)

True code \ Predicted code	0	1	2	3	4	5	6	7	8	9	10
0	1.8	0.9	0.7	0.8	1.2	0.8	1.0	0.3	1.0	1.5	0.7
1	0.5	1.1	0.8	0.3	0.6	1.0	0.5	0.0	0.0	0.5	0.4
2	0.4	1.0	2.1	0.4	0.4	0.9	0.5	0.3	0.0	1.9	0.7
3	1.2	0.4	0.4	2.0	0.0	0.4	0.0	0.0	1.1	0.0	0.0
4	0.8	0.0	0.5	0.4	1.6	0.7	0.6	0.3	0.8	0.8	0.7
5	1.6	0.0	0.7	0.4	0.7	1.0	0.7	1.5	1.9	0.9	0.7
6	0.8	1.1	1.0	0.4	0.3	1.2	1.5	0.5	0.6	0.7	0.5
7	0.9	0.0	0.0	0.3	0.3	1.1	0.0	1.3	0.9	0.0	0.6
8	2.0	0.6	0.4	0.9	1.1	0.9	0.9	0.7	2.7	0.6	0.5
9	1.1	0.5	0.7	0.0	1.4	1.0	1.1	0.3	0.0	2.5	0.4
10	1.1	0.5	0.7	0.5	0.5	0.5	0.0	0.0	0.6	0.5	1.5

Figure 3. Confusion matrices displaying the mean true positives and false positives (left) across the 10 folds and the according standard deviation (right)

5 Implications and Challenges

The proposed classifier achieves a mean accuracy of 0.63 on our test data set. Depending on the class predicted, the individual mean accuracy ranges from 0.27 to 0.87. Further models, different hyperparameter configurations as well as different sampling and training strategies could increase the overall accuracy of our approach. In our experiment, the diversity of class 0 (error codes 11 to 186) is covered by 522 samples, resulting in a mean of less than 3 samples per error code. Since the applied regularization yield a decreased trainings and validation accuracy, we assume that further data is required to generalize well. Incorporating further samples from the error codes 11 to 186 during training could increase the overall performance.

To balance our dataset, we applied an under-sampling strategy to the classes containing more than 522 samples. Further investigations should address different sampling strategies, e.g. over-sampling. Moreover, generating synthetic samples should be taken into consideration to augment and balance the dataset [23].
The selection of the error codes for prediction is based on the cutoff of 500 samples. A different selection of error codes or a clustering of error codes are likely to influence the prediction performance. However, the optimal selection or clusters are not only determined by the achieved accuracy, but rather the usefulness in application scenarios. Thus, further constraints, e.g. misclassification costs, need to be considered.

Beside the challenges specific to improving the prediction model, there are challenges in the application of our concept in productive environments. In comparison to other industries and application scenarios, a wrongly classified complaint could have severe legal consequences. Therefore, we consider the prediction result of our model as a recommendation for the employee handling the complaint. Bearing in mind that the error codes represented by classes 1 to 10 account for almost 50 % of the incoming complaints, a suggestion of an error code could significantly increase the employee's performance. Moreover, depending on the class predicted, a certainty value or a ranking of k top error codes could be provided to the employee.

There are limitations concerning the application of our approach to further companies. Our model requires a comparatively large amount of data to deliver meaningful results. If a company does not have the required amount of data, the application described in this work can only be realized to a limited extent. If there are few complaints in a company, it is difficult to correctly classify error codes. This also applies to cases with limited data quality, including incomplete descriptions or incorrectly classified complaints. In this case, our model will not be able to make reliable predictions.

Furthermore, the handling of the complaint process in medical technology is dependent on the nature of the individual product. For example, the same legal regulations apply to inexpensive commodity products such as patches and to complex medical devices such as ultrasound machines, but their complaint handling differs considerably. In this respect, our concept might not be applicable to every medical technology company. Therefore, a company needs to individually investigate whether our concept is beneficial in their respective complaint processes. This must be considered particularly given the fact that after the initial training the results of our model should regularly be supervised and re-trained with the appropriate current data so that the quality of the result can be maintained or ideally improved. Thus, the company requires an appropriate infrastructure consisting of computational power, hardware either on premise or as a service, as well as experts maintaining and developing the models.

Finally, we are aware that the work presented does not enable the reader to replicate the evaluation results, since the evaluation data is not publicly available. Nevertheless, by providing the concept implementation and a detailed description of the evaluation setup and experiments conducted, we provide the reader with the necessary information to reproduce the results in similar application scenarios.

6 Conclusion and Outlook

This paper deals with the potential of using machine learning to support complaint management in medical technology companies. Using data from a large manufacturer of medical products, we designed, implemented, and evaluated a deep neural network, which is able to assign the correct error source to a textual complaint description in more than 60% of all cases. We evaluated numerous network configurations to identify the network with the highest accuracy. Our approach was able to correctly classify three times as much samples as the naïve classifier which we used as a baseline comparison. These results show the general potential of machine learning for process automation in medical technology, considering that "classical" approaches such as a keyword search would require a priori knowledge about future customer-specific complaints. Also, ML techniques require very little domain knowledge, such that they can support instead of afflicting the experts. Determining whether full process automation is indeed possible and whether deep learning is in fact the best-suited technology remains as future work.

As we point out in the discussion, our approach still leaves room for optimizations, particularly if the goal is to apply it in a productive complaint management process. However, the domain experts of medical technology company, which provided the data, were already involved in the conceptual design and see much potential in its realization. As a next step, we will further optimize the network and the training data, with the goal to evaluate the approach directly with its end users in a productive environment.

The technical solution that we suggest can be transferred to assign other attributes to an incoming complaint. Examples are the criticality assessment or the actions necessary for immediate containment. Provided that enough data is available in the correct quality, many tasks in the complaint process can be completely or partially automated, with causality relations (e.g., stage 4) requiring particular methods for causal inference. Partial automation supports the responsible employees in their work, leaving them with more time to identify and remove the causes of occurring complaints. This is particularly relevant for less experienced employees, since the necessary experience for quickly filing an 8D report can be at least partially replaced by a trained neural network.

However, decisions will not be completely automated for the foreseeable future. For example, the evaluation of the criticality of a complaint can be an extensive decision, which can lead to expensive recalls that may damage the reputation of the company. So, the final assessment will not be automated, but be carried out by an employee instead. This is also relevant for validation purposes. If a neural network made independent decisions within a process related to the manufacturing of medical products instead of just supporting the employees in their decisions, it would be regarded as a production-relevant system. On the one hand, those systems must be formally validated before companies are allowed to use them. On the other hand, neural networks do not behave deterministically, so their functionality can never be validated beyond all doubts.

Acknowledgment. This work was conducted within a project sponsored by the German Ministry for Education and Research (BMBF), project name “Reklamation 4.0”, support code “01IS17088B”. We also gratefully acknowledge the support of NVIDIA for the donation of a GPU used for this research.

References

1. Coussement, K., van den Poel, D.: Improving customer complaint management by automatic email classification using linguistic style features as predictors. Decision Support Systems. 44, 870–882 (2008).
2. Manz, S.: Medical Device Quality Management Systems: Strategy and Techniques for Improving Efficiency and Effectiveness. Elsevier (2019).
3. European Parliament, Council of the European Union: Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices.
4. Lasi, H., Kemper, H.-G., Fettke, P., Feld, T., Hoffmann, M.: Industry 4.0. Business & Information Systems Engineering. 4, 239–242 (2014).
5. Foidl, H., Felderer, M.: Research challenges of industry 4.0 for quality management. In: International Conference on Enterprise Resource Planning Systems. pp. 121–137 (2015).

6. European Commission: The rules governing medicinal products in the European Union - EU Guidelines to Good Manufacturing Practice.
7. Abuhav, I.: ISO 13485: 2016: A Complete Guide to Quality Management in the Medical Device Industry. CRC Press (2018).
8. Behrens, B.-A., Wilde, I., Hoffmann, M.: Complaint management using the extended 8D-method along the automotive supply chain. Production Engineering. 1, 91–95 (2007).
9. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive Business Process Monitoring with LSTM Neural Networks. In: Dubois, E. and Pohl, K. (eds.) Advanced Information Systems Engineering, pp. 477–492. Springer (2017).
10. Evermann, J., Rehse, J.-R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems. 100, 129–140 (2017).
11. Mehdiyev, N., Evermann, J., Fettke, P.: A Novel Business Process Prediction Model Using a Deep Learning Method. Business & Information Systems Engineering. (2018).
12. Nolle, T., Seeliger, A., Mühlhäuser, M.: BINet: Multivariate Business Process Anomaly Detection Using Deep Learning. In: Weske, M., Montali, M., Weber, I., and vom Brocke, J. (eds.) Business Process Management. pp. 271–287. Springer (2018).
13. De Koninck, P., vanden Broucke, S., De Weerdt, J.: act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes. In: Weske, M., Montali, M., Weber, I., and vom Brocke, J. (eds.) Business Process Management. pp. 305–321. Springer (2018).
14. Hake, P., Zapp, M., Fettke, P., Loos, P.: Supporting Business Process Modeling Using RNNs for Label Classification. In: Frasincar, F., Ittoo, A., Nguyen, L.M., and Métais, E. (eds.) Applications of Natural Language to Information Systems, pp. 283–286. Springer (2017).
15. Ko, T., Lee, J.H., Cho, H., Cho, S., Lee, W., Lee, M.: Machine learning-based anomaly detection via integration of manufacturing, inspection and after-sales service data. Industrial Management & Data Systems. 117, 927–945 (2017).
16. Weiss, S.M., Dhurandhar, A., Baseman, R.J.: Improving Quality Control by Early Prediction of Manufacturing Outcomes. In: International Conference on Knowledge Discovery and Data Mining. pp. 1258–1266. ACM (2013).
17. Lieber, D., Stolpe, M., Konrad, B., Deuse, J., Morik, K.: Quality Prediction in Interlinked Manufacturing Processes based on Supervised & Unsupervised Machine Learning. In: Conference on Manufacturing Systems. pp. 193–198. Procedia CIRP (2013).
18. Arif, F., Suryana, N., and Burairah Hussin.: A data mining approach for developing quality prediction model in multi-stage manufacturing. International Journal of Computer Applications. 69, (2013).
19. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation. 9, 1735–1780 (1997).
20. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in Pre-Training Distributed Word Representations. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018).
21. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15, 1929–1958 (2014).
22. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. CoRR. abs/1412.6, (2014).