Graph TopoFilter: a method for noisy labels detection for graph-structured classes*

Artur Budniak[0000−0001−6984−0217]
and Tomasz Kajdanowicz[0000−0002−8417−1012]

Wrocław University of Science and Technology
Department of Computational Intelligence, Wrocław, Poland
`{artur.budniak,tomasz.kajdanowicz}@pwr.edu.pl`

Abstract. Detection of incorrectly conducted failure repairs is not a trivial task for companies manufacturing big volumes of goods. Extensive data sets of service calls are periodically updated and subject matters experts would not be efficient in manual annotating of the data. Symptoms described in free text form might be caused by different components - not necessarily by the most obvious. Classes are imbalanced due to different time to failure of particular components and thus actions taken for some rare failures might be noted as incorrect ones. The presented problem is similar to the problem of learning in a presence of noisy labels, which are caused by human errors, variation of annotator to annotator perception, faults made by annotating algorithms or by other reasons. There are multiple techniques to prevent neural networks from overfitting to the noisy data, but to our best knowledge none of them considers relationships between classes, which is crucial in engineering systems built from multiple components connected in a specific way. A novel approach of selecting clean data samples in an unsupervised manner is presented in this paper. It is based on a topological approach exploring the deep representation of the features in the hidden space, enriched with knowledge graphs reflecting the structure of the classes. We present the case study of the algorithm utilized for service calls data set for home appliances.

Keywords: noisy labels · knowledge graph · natural language processing.

1 Introduction

Deep neural networks are capable of fitting to the training set when it is tagged with true or entirely reshuffled labels [13], which is an impressive property, but also a drawback, when real data sets are considered. A case of a not correct label for given features is noted as a noisy label. Four sources of noisy labels have been indicated by Frenay et al. [4]: poor quality of input data, human mistakes, annotator to annotator variation of perception and data or communication problems.

* Supported by Whirlpool Company Polska Sp. z o. o.

Lower accuracy of a model and longer training time may be caused by the assumption that data set is tagged perfectly and when no strategy for the noise is planned [13]. As a consequence of above, many techniques have been developed to improve generalization of classifiers trained on the real data. However, relationships between classes have not been evaluated in the literature. To benefit from knowing the class structure, we modified the TopoFilter algorithm [10], which is based on the topology of deep representation of features. The algorithm works as follows. First, k-Nearest Neighbors (kNN) graph G of data points from all classes is created. Then nodes of a given class are selected by breaking edges to other class nodes. As a result a sub-graph G_i is set. Samples are considered clean, if they belong to the largest connected component Q_i spread within G_i. For the hidden representation points lying close to the other class the TopoFilter might be too rigorous. Data points of that class i may be not included in Q_i because of points from class j in their neighborhood that break edges. It is correct for independent classes, but for related ones (components in engineering systems) it results in removing too many points. A knowledge graph KG representing the structure of classes prevents our algorithm from deleting the points and thus helps to keep more information from the data set. The graph denotes arbitrarily understood similarity between classes or known by subject matter experts connections (e.g. distance between weather stations [7], current flow between points in power grids [3] or photovoltaic cells placement on farms [2]).

2 Related work

There are a few techniques to handle the presence of noisy labels and one of them is the sample selection. It may be performed by one neural network (MentorNet) to train the other network (StudenNet) by preparing curriculum of more and more complex data points [6]. Instead of selecting for curriculum samples with small loss and adding samples with gradually larger loss during the training phase at constant rate, MentorNet adjusts that process according to StudentNet feedback. Two networks can also work together in an architecture of co-teaching [5], which limits the flow of the error caused by noisy labels due to the different learning rates of the two networks. For every mini-batch the networks select small-loss instances for each other. Later on the method was improved [12] by passing between the two networks only those samples that the networks disagree about in predictions. In contrast to this approach, the joint agreements between two models are used for training in Joint Agreement Method [9]. It is less probable for the two models to fit to the noisy labels at the same time having different learning rates. A deep representation of data points derived from network weights may help to find noisy labels, assuming that samples from the same class are similar [8], which is proposed to be verified by a new version of Local Outlier Factor algorithm (pcLOF). Another approaches use kNN graph construction over points in hidden space. NGC (Noisy Graph Cleaning) propagates their labels [11], while TopoFilter [10], in order to find clean samples, builds the largest connected component within given class. In addition, some points are removed, if in their neighborhood there are too many neighbors from other classes.

3 Problem definition

Notation follows the survey [1]. A data set with noisy labels is denoted as

$$D = \{x_i, y_i\}_{i=1}^{|D|} \tag{1}$$

where x_i is a feature vector, $y_i \in Y$ is an observed label and $|D|$ is the number of entities in the data set D. A true label $\widehat{y_i} \in Y$ is not known. A noise transition matrix presents the probability for each true label $\widehat{y_i}$ to be marked as a noisy, observed label y_i:

$$p(y = j | x_i, \widehat{y_i} = c) = \eta_{jc}(x_i) \tag{2}$$

$$\sum_{j \in Y} \eta_{jc}(x_i) = 1 \tag{3}$$

Hidden representation of the feature vector x_i, denoted as ${x_i}^h$, is learned during the training. In the initial phase of the training it is learned on all samples, including the noisy ones. After sample selection, it is adjusted on samples considered clean. The structure of the classes is represented by a knowledge graph KG. Class labels are nodes and relationships between them are edges of that graph. Cycles are allowed.

The aim of the task is to split the data set D into two subsets containing true ($\widehat{y_i} \in Y$) and false ($Y \setminus \widehat{y_i}$) labels in unsupervised manner. This is a binary classification of samples within each class.

4 The algorithm

As noted before, the algorithm is an extension of TopoFilter [10]. For first m epochs a model is trained on a whole training set. After that the clean samples are selected before the training in each epoch and weights are updated on clean data set up to N epochs. In contrast to approach mentioned above, hidden representations ${x_i}^h$ of clean samples in class i are not only nodes from the largest connected component (lcc) of kNN sub-graph G_i, which is made by removing edges connected to nodes from other classes in G, but the lcc is constructed on nodes from class i and k_{hops} neighbor classes in accordance with the knowledge graph KG. Then neighbor nodes are removed and remaining ones are considered clean. That allows points laying among those from related classes to be kept. The algorithm is presented in pseudo-code below and depicts the differences to the original TopoFilter:

1. Noisy training data S, milestone m, training epochs N,number of classes Γ, number of neighbors k_{NN}, ~~filtering parameter ζ~~, [ADDED: k_{hops}]

```
2.  Output: Collected clean data C
3.  Initialize $C \leftarrow \emptyset, \widehat{S} \leftarrow \emptyset$
4.  for t=1, ···, N do
5.      Train network on $\widehat{S}$
6.      if $t \geq m$ then
7.          Extract feature vectors $x^h$ from training data S
8.          Compute k-NN graph G over $x^h$
9.          for i=1, ···, $\Gamma$ do
10.             Construct subgraph Gi by selecting feature vectors $x_i^h$ from i-th class
                and removing all edges associated with $x_j^h$ ~~for $j \neq i$~~
                [ADDED: if $d(i,j)_{KG} > k_{hops}$]
11.             Compute the largest connected component Qi over Gi.
                [ADDED: remove $x_j^h$ for $j \neq i$ ]
12.             $C \leftarrow C \cup Qi$
13.         end for
14.         ~~Find outliers O within C based on $\zeta - filtering$; update $C \leftarrow C \backslash O$~~
15.         $\widehat{S} \leftarrow C$
16.     end if
17. end for
```

The implementation is available at https://github.com/ArturBudniak/Graph_TopoFilter.

5 Case study

5.1 Data set

The data set is the Service Calls Register of Wall Oven products in Whirlpool Company. For each symptom description x_i claimed by a customer there is a label y_i denoting the component replaced by a serviceman. After removing rows with missing values it contains 79469 rows with 95 unique labels. Classes having less than 100 instances or those not possible to be represented as nodes (e.g. "customer refused repair", "general safety question") are also removed. As a result 46 classes and 62312 rows are selected. For the algorithm validation the following components have been selected for manual annotation by subject matter experts: hinges, lamp, door lock, cooling fan, product fuse and thermostat. There are in total 7207 clean samples. We assume one failure at the time. To keep the company's internal data unrevealed the data sets are not published.

5.2 Knowledge graph

All engineering systems, including home appliances, may be represented by graphs, where nodes indicate components and edges the relationship between them. Edge direction is not utilized, although it might be helpful for engineers. For the same purpose edge types may also be included to distinguish between mechanical, electrical, functional or any other sort of interface. Type of edge is not used by our algorithm.

5.3 Experiments

The experiments compare two algorithms for unsupervised learning - TopoFilter [10] (baseline) and our modified version - Graph TopoFilter. The task is to classify service calls as clean or noisy samples. The factors in the experiments are k_{hops} and size of the data set $|D|$. Three runs are performed for each setting. The other parameters are found experimentally and kept constant. The neural network has input layer with https://tfhub.dev/google/nnlm-en-dim50/2 embedding, two hidden layers with ReLU activation function, each of 200 neurons and a softmax layer. Optimizer is set to adam, loss function - CategoricalCrossentropy. Batch size is set to 512, number of epochs N is 20, milestone m is 15, number of neighbors k_{NN} for kNN graph is 13 and ζ is 0 ($\zeta > 0$ removes most external points in lcc). Parameter k_{hops} is set to 1 and 2 (for 0 the new algorithm is the same as TopoFilter). From the whole data set 20%, 40%, 60%, 80% and 100% of rows are taken randomly for training. Due to the data set imbalance the weighted F1-score metric was chosen to rate the algorithm performance. The values and the standard deviation over three trials are presented in Table 1. Precision and recall are shown in Fig. 1 and Fig. 2. Higher weighted F1-score

Table 1. Weighted F1-score for clean labels classification.

Algorithm	Data set size				
	20%	40%	60%	80%	100%
TopoFilter $k_{hops} = 0$	0.70 ± 0.03	0.77 ± 0.01	0.80 ± 0.01	0.80 ± 0.01	0.82 ± 0.01
Graph TopoFilter $k_{hops} = 1$	0.87 ± 0.01	0.88 ± 0.01	0.89 ± 0.01	0.88 ± 0.01	0.89 ± 0.01
Graph TopoFilter $k_{hops} = 2$	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01

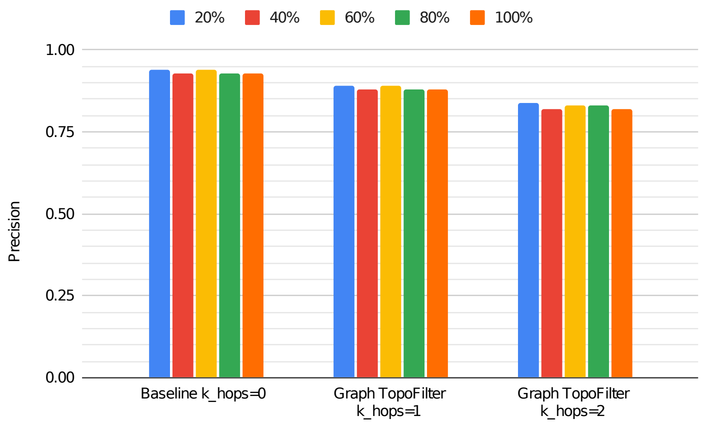

Fig. 1. Precision per algorithm and data set size.

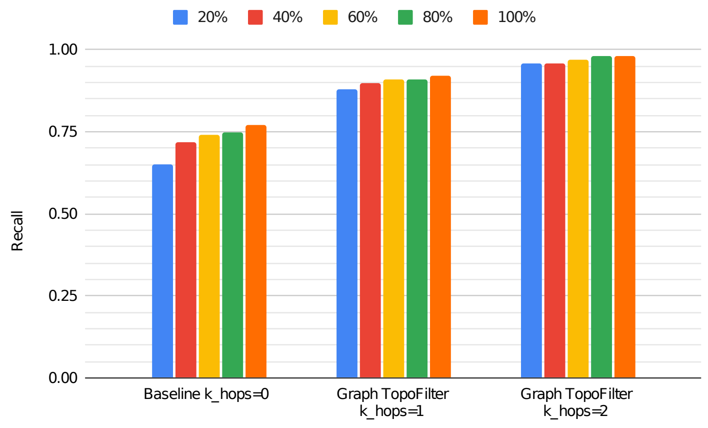

Fig. 2. Recall per algorithm and data set size.

is reported for the Graph TopoFilter compared to the original TopoFilter [10]. The smaller the size of the data set is, the bigger the difference is noted. Precision drops for the new algorithm. However, recall obtains much higher values. It is worth noting that the impact of the data set size is less visible for Graph TopoFilter. The new algorithm needs only 20% of the data to achieve better results than the original one for the whole data set.

6 Summary

The algorithm presented in this paper selects clean samples from a text data set containing service calls in unsupervised learning. It is based on TopoFilter [10] that annotates too many samples as noisy because of the nature of failures and their multiple possible root causes. Our approach takes additional information about the structure of the classes and improves noisy labels detection. Case study experiments conducted on the whole data set showed that weighted F1-score increased from 0.82 for the baseline to 0.89 for Graph TopoFilter. What is more impressive, for the number of service calls reduced to 20% (to simulate rare claims) we kept the same score, while the original TopoFilter obtained only 0.70. The idea requires further investigation of whether using directed edges or adding to them weights and attributes would help to model engineering systems behavior in fault states.

References

1. Cordeiro, F.R., Carneiro, G.: A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). pp. 9–16. IEEE (2020)

2. Fan, J., Rao, S., Muniraju, G., Tepedelenlioglu, C., Spanias, A.: Fault classification in photovoltaic arrays using graph signal processing. In: 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS). vol. 1, pp. 315–319. IEEE (2020)
3. de Freitas, J.T., Coelho, F.G.F.: Fault localization method for power distribution systems based on gated graph neural networks. Electrical Engineering pp. 1–8 (2021)
4. Frénay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems **25**(5), 845–869 (2013)
5. Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M.: Co-teaching: Robust training of deep neural networks with extremely noisy labels. arXiv preprint arXiv:1804.06872 (2018)
6. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: International Conference on Machine Learning. pp. 2304–2313. PMLR (2018)
7. Owerko, D., Gama, F., Ribeiro, A.: Predicting power outages using graph neural networks. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP). pp. 743–747. IEEE (2018)
8. Wang, Y., Liu, W., Ma, X., Bailey, J., Zha, H., Song, L., Xia, S.T.: Iterative learning with open-set noisy labels. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8688–8696 (2018)
9. Wei, H., Feng, L., Chen, X., An, B.: Combating noisy labels by agreement: A joint training method with co-regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13726–13735 (2020)
10. Wu, P., Zheng, S., Goswami, M., Metaxas, D., Chen, C.: A topological filter for learning with label noise. Advances in neural information processing systems **33**, 21382–21393 (2020)
11. Wu, Z.F., Wei, T., Jiang, J., Mao, C., Tang, M., Li, Y.F.: Ngc: A unified framework for learning with open-world noisy data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 62–71 (2021)
12. Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., Sugiyama, M.: How does disagreement help generalization against label corruption? In: International Conference on Machine Learning. pp. 7164–7173. PMLR (2019)
13. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Communications of the ACM **64**(3), 107–115 (2021)