Layerwise Relevance Visualization in Convolutional Text Graph Classifiers

Robert Schwarzenberg, Marc Hübner, David Harbecke, Christoph Alt, Leonhard Hennig
German Research Center for Artificial Intelligence (DFKI), Berlin, Germany
{firstname.lastname@dfki.de}

Abstract

Representations in the hidden layers of Deep Neural Networks (DNN) are often hard to interpret since it is difficult to project them into an interpretable domain. Graph Convolutional Networks (GCN) allow this projection, but existing explainability methods do not exploit this fact, i.e. do not focus their explanations on intermediate states. In this work, we present a novel method that traces and visualizes features that contribute to a classification decision in the visible and hidden layers of a GCN. Our method exposes hidden cross-layer dynamics in the input graph structure. We experimentally demonstrate that it yields meaningful layerwise explanations for a GCN sentence classifier.

1 Introduction

A Deep Neural Network (DNN) that offers – or from which we can retrieve – explanations for its decisions arguably is easier to improve and debug than a black box model. Loosely following Montavon et al. (2018), we understand an explanation as a "collection of features (...) that have contributed for a given example to produce a decision." For humans to make sense of an explanation it needs to be expressed in a human-interpretable domain, which is often the input space (Montavon et al., 2018).

For instance, in the vision domain Bach et al. (2015) propose the Layerwise Relevance Propagation (LRP) algorithm that propagates contributions from an output activation back to the first layer, the input pixel space. Arras et al. (2017) implement a similar strategy for NLP tasks, where contributions are propagated into the word vector space and then summed up over the word vector dimensions to create heatmaps in the plain text input space.

In the course of the backpropagation of contributions, hidden layers in the DNN are visited, but rarely inspected.[1] A major challenge when inspecting and interpreting hidden states lies in the fact that a projection into an interpretable domain is often made difficult by the non-linearity, non-locality and dimension reduction/expansion across hidden layers.

In this work, we present an explainability method that visits and projects visible and hidden states in Graph Convolutional Networks (GCN) (Kipf and Welling, 2016) onto the interpretable input space. Each GCN layer fuses neighborhoods in the input graph. For this fusion to work, GCNs need to replicate the input adjacency at each layer. This allows us to project their intermediate representations back onto the graph.

Our method, layerwise relevance visualization (LRV), not only visualizes static explanations in the hidden layers, but also tracks and visualizes hidden cross-layer dynamics, as illustrated in Fig. 1. In a qualitative and a quantitative analysis, we experimentally demonstrate that our approach yields meaningful explanations for a GCN sentence classifier.

2 Methods

In this section we explain how we combine GCNs and LRP for layerwise relevance visualization. We begin with a short recap of GCNs and LRP to establish a basis for our approach and to introduce notation.

2.1 Graph Convolutional Networks

Assume a graph G represented by $(\tilde{A}, H^{(0)})$ where $\tilde{A}$ is a degree-normalized adjacency matrix, which introduces self loops to the nodes in G, and

[1] Although, Arras et al. (2017) propose inspecting hidden layers with LRP as a possible future direction.

Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13), pages 58–62
Hong Kong, November 4, 2019.

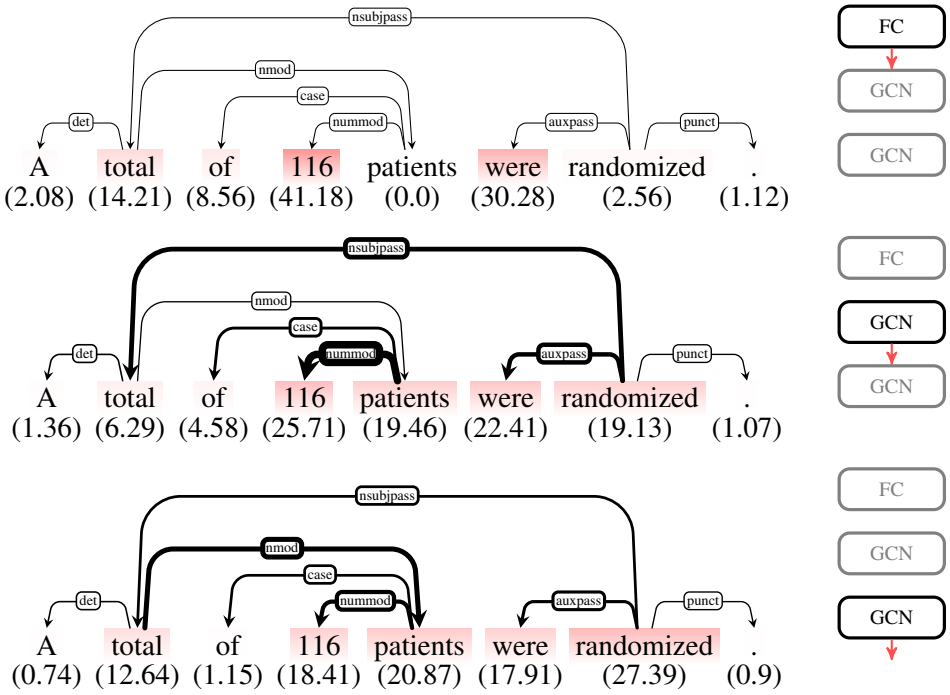

Figure 1: Layerwise Relevance Visualization in a Graph Convolutional Network. Left: Projection of relevance percentages (in brackets) onto the input graph structure (red highlighting). Edge strength is proportional to the relevance percentage an edge carried from one layer to the next. Right: Architecture (replicated at each layer) of the GCN sentence classifier (input bottom, output top). Node and edge relevance were normalized layerwise. The predicted label of the input was `RESULT`, as was the true label.

$H^{(0)} \in \mathbb{R}^{n \times d}$ an embedding matrix of the n nodes in G, ordered in compliance with $\tilde{A}$. A GCN layer propagates such graph inputs according to

$$H^{(l+1)} = \sigma\left(\tilde{A}H^{(l)}W^{(l)}\right), \qquad (1)$$

which can be decomposed into a feature projection $H'^{(l)} = H^{(l)}W^{(l)}$ and an adjacency projection $\tilde{A}H'^{(l)}$, followed by a non-linearity σ.

The adjacency projection fuses each node with the features in its effective neighborhood. At each GCN layer, the effective neighborhood becomes one hop larger, starting with a one-hop neighborhood in the first layer. The last layer in a GCN classifier typically is fully connected (FC) and projects its inputs onto class probabilities.

2.2 Layerwise Relevance Propagation

To receive explanations for the classifications of a GCN classifier, we apply LRP. LRP explains a neuron's activation by computing how much each of its input neurons contributed[2] to the activation. Contributions are propagated back layerwise.

Montavon et al. (2017) show that the positive contribution of neuron $h_i^{(l)}$, as defined in Bach et al. (2015) with the z^+-rule, is equivalent to

$$R_i = \sum_j \frac{h_i^{(l)} w_{ij}^+}{\sum_k h_k^{(l)} w_{kj}^+} R_j, \qquad (2)$$

where $\sum_k$ sums over the neurons in layer l and $\sum_j$ over the neurons in layer $(l+1)$. Eq. 2 only allows positive inputs, which each layer receives if the previous layers are activated using ReLUs.[3] LRP has an important property, namely the relevance conservation property: $\sum_j R_{j \leftarrow k} = R_k, R_j = \sum_k R_{j \leftarrow k}$, which not only conserves relevance from neuron to neuron but also from layer to layer (Bach et al., 2015).

2.3 Layerwise Relevance Visualization

We combine graph convolutional networks and layerwise relevance propagation for layerwise relevance visualization as follows. During training, GCNs receive input graphs in the form of $(\tilde{A}, H^{(0)})$ tuples. This is efficient and allows

[2] We use the terms *contribution* and *relevance* interchangeably.

[3] LRP is capable of tracking negative contributions, too. In this work, we focus on positive contributions.

batching but it poses a problem for LRP: If the adjacency matrix is considered part of the input, one could argue that it should receive relevance mass. This, however, would make it hard to meet the conservation property.

Instead, we treat $\tilde{A}$ as part of the model in a post-hoc explanation phase, in which we construct an FC layer with $\tilde{A}$ as its weights. The GCN layer then consists of two FC sublayers. During the forward pass, the first one performs the feature projection and the second one – the newly constructed one – the adjacency projection.

To make use of Eq. 2 in the adjacency layer, we need to avoid propagating negative activations. This is why we apply the ReLU activation early, right after the feature projection, which yields the propagation rule

$$H^{(l+1)} = \tilde{A}\sigma(H^{(l)}W^{(l)}). \tag{3}$$

When unrolled, this effectively just moves one adjacency projection from inside the first ReLU to outside the last ReLU. Alternatively, it should be possible to first perform the adjacency projection and afterwards the feature projection.

Eq. 3 allows us to directly apply Eq. 2 to the two projection sublayers in a GCN layer. During LRP, we cache the intermediate contribution maps $R^{(l)} \in \mathbb{R}^{n \times f}$ that we receive right after we propagated past the feature projection sublayer and compute node i's contribution in that layer as $R(i^{(l)}) = \sum_c R_{ic}^{(l)}$. In addition, we also compute the edge relevance $e_{(i,j)}^{(l)}$ between nodes i and j as the amount of relevance that it carried from layer $l-1$ to layer l.

In what follows, we use $R(i^{(l)})$ and $e_{ij}^{(l)}$ to visualize hidden state dynamics (i.e. relevance flow) in the input graph. Furthermore, similar to Xie and Lu (2019), we make use of perturbation experiments to verify that our method indeed identifies the relevant components in the input graph.

3 Experiments

Our experiments are publicly available: https://github.com/DFKI-NLP/lrv/. We trained a GCN sentence classifier on a 20k subset of the PubMed 200k RCT dataset (Dernoncourt and Lee, 2017). The dataset contains scientific abstracts in which each sentence is annotated with either of the 5 labels *BACKGROUND*, *OBJECTIVE*, *METHOD*, *RESULT*, or *CONCLUSION*. Dernoncourt and Lee (2017) describe the task as a sequential classification task since all sentences in one abstract are classified in sequence, but we treat it as a conventional sentence classification task, classifying each sentence in isolation.

In a preprocessing step, we used ScispaCy (Neumann et al., 2019) for tokenization and dependency parsing to generate the input graphs for the GCN sentence classifier. For the node embeddings in the dependency graph, we utilized pre-trained fastText embeddings (Mikolov et al., 2018). Edge type information was discarded.

Our classifier, implemented and trained using PyTorch (Paszke et al., 2017), consists of two stacked GCN layers (Eq. 3), followed by a max-pooling operation and a subsequent FC layer. All layers were trained without bias and optimized with the Adam optimizer (Kingma and Ba, 2014). After each optimization step we clamped the negative weights of the last FC layer to avoid negative outputs.

During training, we batched inputs, but in the post-hoc explanation phase, single graphs from the test set were forwarded through the network to implement the strategy outlined in Sec. 2.3. We first constructed the adjacency layer with $\tilde{A}$, then performed a forward pass during which we cached the inputs at each layer for LRP.

LRP started at the output neuron with the maximal activation, before the softmax normalization. For all intermediate layers we used Eq. 2. Since coefficients in the input word vectors are negative, we used a special propagation rule (Montavon et al., 2017), which we omit here, due to space contraints. We cached the intermediate contribution maps right after the max-pooling operation, after the second and after the first (input space) feature projection layers in the GCN layers.

For the qualitative analysis, we visualized the contributions of each node at each layer as well as the relevance flow over the graph's edges across layers, as outlined in Sec. 2.3. For the quantitative validation, we deleted a growing portion of the globally most relevant edges (starting with the most relevant ones) and monitored the model's classification performance. In a second experiment, we did the same starting with the least relevant edges. Here, *global edge relevance* refers to the sum of an edge's relevance across all layers, $\sum_l e_{ij}^{(l)}$.

4 Results

After training, our model achieved a weighted F_1 score of 0.822 on the official (20k) test set. We then performed the post-hoc explanation with the trained model, receiving layerwise explanations as the one depicted in Fig. 1. More examples can be found in the appendix.

Fig. 1 (top) shows which vectors in the final layer the GCN bases its classification decision on. The middle and bottom segments reveal how much relevant information the model fused in these vectors from neighboring nodes during the forward pass:

According to the LRV in Fig. 1, the model primarily bases its classification decision on the fusion of the vector representations of `total`, `patients`, and `116`. The vector representations of these nodes are accumulated in the vector of `116` in two steps.

Interestingly, the vector of `patients` contributes a larger portion after it has been fused with `total`. Furthermore, `randomized` becomes more relevant after the first GCN layer, contributing to the second most relevant n-gram `(were, randomized)`. Other n-grams, such as `(A, total)` or `(of, patients)` appear less relevant.

Note that if we had simply redistributed the contributions in the final layer, using only the adjacency matrix, without considering the activations in the forward pass as LRP does, these n-grams would have received an unsubstantiated share. Furthermore, a conventional, single explanation in the input space would have hardly revealed the hidden dynamics discussed above.

As explained in Sec. 3, we also conducted input perturbation experiments to validate that our method indeed identifies the graph components that are relevant to the GCN decision. Fig. 2 summarizes the results: The performance of our model degrades much faster when we delete edges that contributed a lot to the model's decision, according to our method. We take this as evidence that our method indeed correctly identifies the relevant components in the input graphs.

5 Related Work

Niepert et al. (2016); Wu et al. (2019) introduce graph networks that natively support feature visualization (explanations) but their models are not

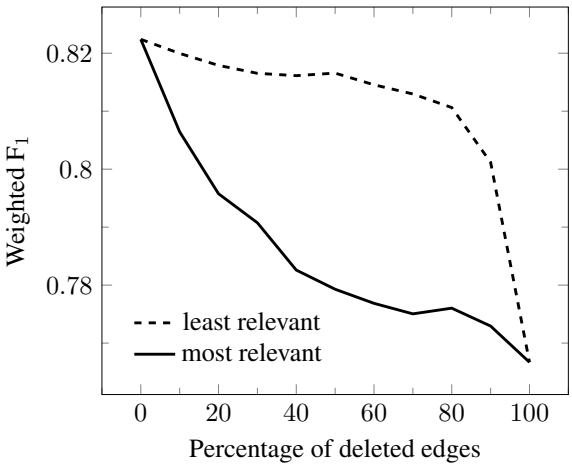

Figure 2: Perturbation experiment.

graph convolutional networks in the sense of Kipf and Welling (2016) that we address here.

Veličković et al. (2017) propose Graph Attention Networks and also project an explanation of a hidden representation onto the input graph. They do not, however, continue across multiple layers. Furthermore, their explanation technique differs from ours. The authors use attention coefficients to scale edge thicknesses in their explanatory visualization. In contrast, we base edge relevance on the amount of relevance an edge carried from one layer to the next, according to LRP. Our method does this in a conventional GCN, which does not apply the attention mechanism.

The body of work on the explainability of graph neural nets also includes the GNN explainer by Ying et al. (2019), a model-agnostic explainability method. Since their method is model-agnostic, it can be applied to GCNs. Nevertheless, it would not reveal hidden dynamics, as our method does.

In parallel to our work, in the last couple of months, several more works on explainability in the context of graph neural nets were published. Pope et al. (2019), for instance, evaluate several explainability methods on data sets from the chemistry and vision domains. The methods do not include LRP, however, and the authors do not visualize relevance layerwise. The two works that are most related to our approach were published very recently by Xie and Lu (2019) and Baldassarre and Azizpour (2019). Both implement LRP for graph networks (inter alia). Xie and Lu (2019) introduce the notion of node importance visualization, which is also reflected in our explanations and Baldassarre and Azizpour (2019), similar to

our approach, suggest to trace (and visualize) explanations over feature-less edges. Both works, however, address a different task from ours: In contrast to our graph classification task, they perform node classification and identify k-hop neighbor contributions in the proximity of a central node of interest.

6 Conclusions

We presented layerwise relevance visualization in convolutional text graph classifiers. We demonstrated that our approach allows to track, visualize and inspect visible as well as hidden state dynamics. We conducted qualitative and quantitative experiments to validate the proposed method.

This is a focused contribution; further research should be conducted to test the approach in other domains and on other data sets. Future versions should also exploit LRP's ability to expose negative evidence and the method could be extended to node classifiers.

7 Acknowledgements

This research was partially supported by the German Federal Ministry of Education and Research through the project DEEPLEE (01IW17001). We would also like to thank the anonymous reviewers for their feedback on the paper and Arne Binder for his feedback on the code base.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. "What is relevant in a text document?": An interpretable machine learning approach. *PloS one*, 12(8):e0181142.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140.

Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *IJCNLP 2017*, page 308.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2017. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for biomedical natural language processing.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10772–10781.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Felix Wu, Tianyi Zhan, Amauri Holonda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Weinberger Q. 2019. Simplifying graph convolutional network. *arXiv preprint*.

Shangsheng Xie and Mingming Lu. 2019. Interpreting and understanding graph convolutional neural network using gradient-based attribution methods. *arXiv preprint arXiv:1903.03768*.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*.