L101 Assignment
Demystifying Graph Neural Networks in Document Classification: Analysis of Local Explanations

Gabriele Dominici
University of Cambridge
gd489@cam.ac.uk

Abstract

Graph Neural Networks (GNN) have disrupted different Machine Learning fields over the last few years. However, their reasoning process is still unclear, making it difficult to interpret and understand them, which is essential for using them in several tasks. One of these is document classification, where generating local explanations is one of the typical approaches to supporting models' prediction. Therefore, this project aims to compare local explanations of different Graph Neural Networks algorithms, understanding which explanation method performs best and their difference between models. Moreover, the project wants to understand if key components of the architecture (readout and the number of layers) impact the local explanations. All of this helps us better understand how these models work and how we can use them more consciously.

1 Introduction

Since the release of Graph Convolutional Networks (GCN) (Kipf and Welling, 2016), Graph Neural Networks (GNNs) have attracted significant interest from researchers of all Machine Learning fields thanks to their excellent results (Veličković et al., 2017; Corso et al., 2022; Veličković et al., 2018). Specifically, GNNs also impacted the NLP field (Zhong et al., 2020; Qian et al., 2019), although it is not straightforward to use GNNs in NLP. There is no clear underlying graph, so you have to choose among different possible graphs (e.g. dependency parsing, constituency parsing), which may not be optimal.

Despite this significant interest in them, their decision process is still opaque. However, over the last few years, the focus on it has grown, and some works focused on post-hoc explanation (Huang et al., 2020; Ying et al., 2019), while others focus on developing self-explainable models (Zhang et al., 2021; Magister et al., 2022). They were used extensively on synthetic datasets (e.g. BA-Shapes (Ying et al., 2019)) or on real datasets where the structure of the graph was the key aspect (e.g. Mutagenicity (Kazius et al., 2005)). On the other hand, to my best knowledge, no work focused on local explanations for GNNs in NLP and, specifically, in the document classification task.

On the contrary, in NLP, more classical interpretability methods are used and explored widely (Nguyen, 2018; Zafar et al., 2021; Ribeiro et al., 2016; Lundberg and Lee, 2017).

For these reasons, inspired by the work of Nguyen (2018), I decided to explore the local explanation generated with different interpretability methods in different GNN architectures and configurations. The project aims to answer four specific questions: **(i)** how do interpretability methods work on GNNs in a document classification task? **(ii)** How different are the local explanations of different GNN architectures? **(iii)** What is the impact of using different readout functions on the same GNN architecture? **(iv)** What is the impact of using a different number of layers in the same GNN architecture? To answer these questions, I automatically evaluate the local explanations generated with different metrics. Then I developed a website where it is possible to visualise the local explanations for every input used in the project and give a qualitative evaluation of the outputs.

The results show that there is no clear best interpretability method. It depends on the dataset and on the model used. Moreover, specific components (e.g. readout function) have a significant impact on the local explanation generated.

All the experiments can be reproduced with the code available in a GitHub repository[1].

2 Related work

Some studies in the past decades have inspected which interpretability methods created better lo-

[1] https://github.com/gabriele-dominici/L101

cal explanations on simple models or how different model initialisation impacted the explanations. Meanwhile, others created new kinds of interpretability methods to create more accurate local explanations in general or specific models.

Nguyen (2018) used different interpretability methods to generate local explanations for a Logistic Regression model and a Multilayer Perceptron. Then, they analysed the generated explanations automatically and manually, creating a crowdsourcing task. They showed that Omission and Vanilla Saliency methods create better local explanations than LIME (Ribeiro et al., 2016) and that there is a significant correlation between automatic evaluation and crowd accuracy.

Similarly, Zafar et al. (2021) studied how different local explanations were between identical models with different weights' initialisation. They tested it on four different versions of BERT-like models (Devlin et al., 2018) and showed that they disagreed entirely with the local explanations given by each other.

On the other hand, more sophisticated methods were developed to perform better local explanations. One of these is SHapley Additive exPlanations (SHAP) (Lundberg and Lee, 2017), which uses a game theory approach exploiting Shapley values to explain the model. Unfortunately, to my best knowledge, this method is not applicable straightforwardly to GNNs. However, other interpretability methods were developed to work only on GNNs. For example, GNNExplainer (Ying et al., 2019) and GraphLIME (Huang et al., 2020) create local explanations for GNNs. Unfortunately, the last one can be used only on node classification, which is different from the task analysed in this project.

3 Experimental Setup

I extracted the local explanation on two document classification datasets using three different interpretability methods and twelve different model configurations. Then, I evaluated them with four interpretability metrics.

It is possible to reproduce all the experiments following all the details described in this section and set the random seed to 42 before any random operation.

	# Documents	
	Movie	**20news group**
Train	870	1072
Dev	209	358
Test	717	370

Table 1: Split of the two datasets used, the same as Nguyen (2018).

3.1 Datasets

In this project, I used the same two datasets for binary document classification task of previous work (Nguyen, 2018) about local explanations for document classification. Table 1 shows complete details of them.

3.1.1 Movie

The first one is Movie reviews [2], which contains polarity labels (positive and negative) for movie reviews.

3.1.2 20news group

The second one is the 20news group[3], which contains newsgroup posts on 20 topics. To make the task binary for simplicity, I have selected only the ones labelled with "Atheism" or "Christian".

I decided to include the header and footer of these posts, as done by Nguyen (2018), but this allows the model to focus also on them, which sometimes are not good explanations (e.g. emails).

3.2 Models

To address the questions mentioned above, I selected four different GNN models and one simple Multilayer Perceptron (MLP) model to use as a comparison and see how explainability metrics differ from a classical model to GNNs. I trained the same architectures for both datasets.

Table 2 shows the performance of the models trained.

All the details and hyperparameters used to implement and train the models are shown in Section A.

3.2.1 Multilayer Perceptron (MLP)

I trained a Multilayer Perceptron with two layers, ReLU as the activation function and a dropout at

[2]https://huggingface.co/datasets/movie_rationales

[3]https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

Model	Accuracy (%) 20news group	Movie
MLP	93.1	80.5
GCN 2L Max	91.6	81.9
GCN 2L Mean	88.7	86.2
GCN 3L Mean	88.2	87.0
GAT 2L Max	84.6	74.3
GAT 2L Mean	89.1	85.6
GAT 3L Mean	90.2	82.4
SAGE 2L Max	87.4	77.3
SAGE 2L Mean	90.6	85.9
SAGE 3L Mean	89.1	84.5
SGC 2L Max	94.5	87.8
SGC 2L Mean	88.0	85.4
SGC 3L Mean	90.6	85.9

Table 2: Accuracy of the models trained across the two datasets. The model name describes the architecture used, the number of layers and the readout function used.

50% after the first layer. It takes the TF-IDF representation of the document and considers only the words that appear in more than 10 documents and less than in the 25% of entries across the training dataset. I did it to maintain the model as similar as possible to Nguyen (2018) and to reduce the complexity of the model.

It is used as a comparison for the other model used in the project because it is a classical Deep Learning model and shows how its local explanations differ from the GNN models.

3.2.2 Graph Neural Network

GNNs are a family of Deep Learning models which takes a graph as the input and exploit it to solve a task (node classification, edge classification, graph classification). They spread information (messages), which can be node or edge representations, through the graph, updating the representation of the other elements in it.

To do it, the GNN model uses a message-passing function, a permutation equivariance function that updates the representation of each element, using the representation of the neighbour elements (nodes connected to the updating one). Using n layer of message passing, the representation of each node, for instance, is influenced indirectly by the state of the nodes, which are n-hops far.

In this project, I created a graph for each document, and I wanted to classify that graph. To do it, GNNs need a readout function, a permutation equivariance function that combines the representation of all the nodes. The ones used in this project, which are the main and basic ones, are the mean and max pooling. They compute the element-wise mean and max across all the nodes, respectively.

In this project, I chose four of the main GNN architectures, which differ from each other in their message-passing function.

Graph Convolutional Networks (GCN) (Kipf and Welling, 2016) It uses the same learnable weights to process and combine the information coming from all the edges. The message-passing function is :

$$X_{t+1} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}X_tW$$

where X_t is the representation of all nodes at layer t, W is the weight matrix, $\hat{A}$ is the adjacency matrix with self-loop, and $\hat{D}$ is the diagonal node degree matrix needed to rescale each aggregation with the number of nodes in that neighbourhood.

Graph Attention Network (GAT) (Veličković et al., 2017) On the contrary, GAT computes an attention score for each edge and uses it in the aggregation step to weigh the related message. To compute the new representation of a node, it uses the following function:

$$x_{i,t+1} = \alpha_{ii}Wx_t + \sum_{j\in\mathcal{N}(i)} \alpha_{ij}Wx_{j,t}$$

where α_{ij} is the attention function computed on nodes i and j, $x_{i,t}$ is the representation of node i at layer t and $\mathcal{N}(i)$ is the set of the neighbour nodes of node i.

GraphSAGE (Hamilton et al., 2017) This architecture is similar to GCN, but it first aggregates the representation of the neighbour node and then projects and combines it with the node representation. Another difference from GCN is that it uses two learnable matrices.

$$x_{i,t+1} = W_1x_{i,t} + W_2mean_{j\in\mathcal{N}(i)}x_{j,t}$$

Simple Graph Convolution (SGC) (Wu et al., 2019) It is a simplified version of GCN, lighter and faster, with comparable or better results. It assumes that the nonlinearity between GCN layers is unnecessary and compresses all the GCN layers in a singular step without an intermediate activation function.

$$X' = (\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2})^kXW$$

where k is the number of layers you want to use.

3.3 Graph Structure

Each of the GNN models takes a graph as input. It plays a significant role in the final prediction of the model, given that messages are shared based on the edges in the graph.

Usually, a graph structure is known a priori and exploitable straightforwardly. However, sometimes, it is not the case, especially in Natural Language Processing tasks, and you have to create one from scratch. There are plenty of methods to create a graph starting from a text: dependency graph, based on dependency parsing, constituency graph, based on constituency parsing, similarity graphs, where two entities are connected if their representation is similar, and many more (Yao et al., 2018; Qian et al., 2019).

For the project's scope, I used a very simple graph structure, which connects every word to the previous and following ones. There is a single node for every specific word in a text, even if the word is repeated more than once. Therefore, there are some "skip edges" that make some words that are far in the text closer. In addition, to build graphs, I have only used the words in the vocabulary defined with TF-IDF, as previously explained in Section 3.2.1.

Another important aspect to define is the node representation. Also, for this aspect, there are some options, like using the word embedding through GloVE (Pennington et al., 2014) or Word2Vec (Mikolov et al., 2013). In this project, to maintain as many similarities as possible between MLP and GNN models, I used a one-hot encoding vector of the TF-IDF vocabulary with the same parameters described in Section 3.2.1.

3.3.1 GNNs configuration

To understand how different readout layers and the number of GNN layers impact the local explanation of the model, I trained 3 different models for each of the specific GNNs mentioned above and computed the local explanation.

One of the three configurations comprises two blocks of layer, where a block means a GNN layer, a ReLU and a 50% dropout. After that, it uses a global mean pooling as the readout function over all the graph nodes and feeds it into a Linear layer. The second one differs from this for the readout function because it uses a global max pooling layer. The last one uses the first configuration but with an additional block. The only exception is that in SGC configuration, there is always one block, but the k is equal to the number of layers in the corresponding other models.

3.4 Interpretability Methods

To generate local explanations for all the models described, I chose the Random method as the baseline, whereas Word Omission, Vanilla Saliency and GNNExplainer as the main interpretability methods. The first two are model-agnostic and can be used easily on graphs, while the other is specific for graph architectures.

3.4.1 Word Omission

The Word Omission method (Robnik-Sikonja and Kononenko, 2008) tries to estimate the contribution of every word to the final prediction. This approach deletes the word from the input and computes the difference between the original output and the one using the modified input. Specifically, in this project, it deletes the node related to that word from the graph.

3.4.2 Vanilla Gradient

The Vanilla Gradient (VG) method derives from Saliency (Simonyan et al., 2013), which was initially used in computer vision tasks to visualize the input attribution of the model. VG calculates the prediction's gradients to understand which input changes have a higher impact on the output.

3.4.3 GNNExplainer

GNNExplainer (Ying et al., 2019) aims to find the subgraph which minimises the difference between the prediction using the complete graph and the one with only the subgraph. Then, it is possible to compute the nodes' importance.

3.5 Interpretability Metrics

To evaluate which interpretability method is the more efficient in generating local explanation, I chose Area Over Perturbation Curve (AOPC) (Samek et al., 2015) and Switching Point (SP) (Nguyen, 2018). They give us an idea of how much the local explanations impact the model's predictions by deleting words from the input.

On the other hand, I chose Jaccard Similarity (Zafar et al., 2021) to compare local explanations of different models and understand if they focus on the same words.

3.5.1 Area Over Perturbation Curve (AOPC)

AOPC (Samek et al., 2015) is a standard metric used to see how much the output's probability changes by deleting K words from the inputs. It is calculated in the following way:

$$AOPC(x) = \frac{1}{K+1}(\sum_{n=1}^{K} f(x) - f(x_{\setminus 1..n}))$$

x is the input's representation, $f(x)$ is the original model's output and $f(x_{\setminus 1..k})$ is the output of the model that received the input representation without the top n words retrieved from interpretability methods. The K is set to 10 in this project.

This method can be used with words that support the prediction and those that are against it.

A higher absolute value of AOPC means a better ability to extract important words for and against the prediction.

3.5.2 Switching Point (SP)

Another similar metric I used is the Switching Point (Nguyen, 2018). It deletes the words that interpretability methods consider more important for the prediction until the prediction changes. It represents the percentage of words deleted before the switch of the prediction. The formula is the following:

$$SP(x) = \frac{\sum_{n=1}^{N} 1_{(if\, f(x) = f(x_{\setminus 1..n})}}{N}$$

where N is the number of tokens that supports the prediction retrieved by the explainable method.

A lower figure for this metric represents a higher ability to capture important words in the input.

3.5.3 Jaccard Similarity (JS)

The last metric I used is the Jaccard Similarity (Zafar et al., 2021). It is helpful to see if two models give importance to the same words. I used this metric to address the second question of the project. It is calculated in the following way:

$$JS(M_a, M_b, K) = \frac{|EM_a(x,K) \cap EM_b(x,K)|}{|EM_a(x,K) \cup EM_b(x,K)|}$$

where M_a and M_b are the two models, and $EM_a(x, K)$ is the set of top K tokens selected by the explanation method chosen.

It represents the ratio between the intersection between sets of top K tokens retrieved by the explainable method chosen for the two models and the union of these two sets.

A higher figure for this metric means the models have a high agreement.

4 Results

To answer the project's questions described in Section 1 about how different interpretability methods work on different GNN models, I evaluated the local explanations of the models automatically with the metrics described in Section 3.5 and qualitatively by inspecting the outputs manually.

4.1 Automatic Evaluation Results

(i) How do interpretability methods work on GNNs in a document classification task? Firstly, every interpretability method used works better than the Random selection on every model trained.

Looking at Table 3, which shows AOPC+@10 values (AOPC computed deleting the 10 most important words), VG has consistently better results on the 20news group dataset than other interpretability methods, while Omission and GNNExplainer are the best ones on the Movie dataset.

On the other hand, considering the SP metric (Table 4), there is no clear division. On the Movie dataset, the interpretability method that achieves the best results is still Omission, while on the 20news group dataset, it depends on the GNN model. GCN and SAGEGraph models achieve lower SP values through VG, while the others through Omission.

(ii) How different are the local explanations of different GNN architecture? The first thing to notice is that most GNN models achieved better AOPC+@10 and SP scores than the MLP model using Omission and VG methods. Only some GNN models with the max pooling layer achieved worse SP scores than the MLP model.

However, Figure 1 and Figure 2 show the Jaccard Similarity between the local explanation created with the Omission methods of different models. Specifically, on the Movie dataset, all the GNN models that use the mean pooling layer partly agree on the important words, and even more, the models that differ only for the number of GNN layers. Particularly, SGC models with the mean pooling layer have a very high agreement among them and a high JS score with GCN models. It is reasonable because the idea behind their GNN layer is the same.

On the other hand, the agreement between models that used different readout functions is significantly low, even if the other parts of the model are the same (number and kind of layers).

Model	20news group RND	OM	VG	GNNExp	Movie RND	OM	VG	GNNExp
MLP	0.019	**0.111**	0.084	-	0.011	**0.080**	0.059	-
GCN 2L Max	0.009	0.096	**0.139**	0.077	0.006	**0.133**	0.091	0.079
GCN 2L Mean	0.012	0.118	**0.210**	0.152	0.007	0.126	0.120	**0.213**
GCN 3L Mean	0.024	0.160	**0.218**	0.161	0.007	0.128	0.106	**0.150**
GAT 2L Max	0.023	0.080	**0.154**	0.105	0.003	**0.131**	0.086	0.083
GAT 2L Mean	0.021	0.138	**0.154**	0.140	0.014	**0.196**	0.146	0.177
GAT 3L Mean	0.017	0.142	**0.190**	0.139	0.004	**0.139**	0.078	0.106
SAGE 2L Max	0.012	0.067	**0.160**	0.113	0.004	**0.135**	0.095	0.093
SAGE 2L Mean	0.019	0.064	**0.260**	0.194	0.007	0.143	0.102	**0.146**
SAGE 3L Mean	0.020	0.136	**0.263**	0.211	0.008	0.156	0.118	**0.191**
SGC 2L Max	0.007	0.065	**0.074**	0.058	0.065	**0.117**	0.093	0.101
SGC 2L Mean	0.005	0.060	**0.069**	0.047	0.007	0.126	0.135	**0.190**
SGC 3L Mean	0.015	0.106	**0.114**	0.096	0.008	0.128	0.132	**0.186**

Table 3: AOPC+@10 for every model trained computed on the local explanation of the four interpretability methods used.

Figures in Section B show that something similar also happens with the other local explanation methods in both datasets.

(iii) What is the impact on local explanation of using different readout functions on the same GNN architecture? Considering the local explanations extracted with the Omission approach, 5 models out of 8 (4 for each dataset) that used the mean pooling as the readout function have achieved a higher AOPC+@10 than the same models with the max pooling. The results are even better if you consider the local explanation extracted with VG, where 7 out of 8 models have a higher AOPC+@10.

Moreover, if we compare the models using the SP metric, all the models that used the mean pooling layer achieved a better score with all the interpretability methods. In this case, the gap between the two models is consistent. For instance, considering the Omission local explanations, the GAT with the max pooling layer scored 0.71, while the figure for the GAT with the mean pooling layer was 0.34.

These results suggest that the local explanations given for models that use the mean pooling as the readout function have a higher impact on the models' predictions.

(iv) What is the impact on the local explanations of using a different number of layers in the same GNN architecture? Changing the number of layers had a minor impact on the local explanations, as we saw in Figure 1 and Figure 2.

However, considering Omission explanations, 7 out of 8 models with 3 layers achieved slightly higher figures for AOPC+@10 than the same models with 2 layers.

On the other hand, only 3 out of 8 models that used 3 layers achieved better SP scores with Omission and GNNExplainer explanations and only 1 out of 8 with VG explanations.

These results show no clear answer to this question because the two metrics suggest different takeaways.

4.2 Output Analysis

I created a public website[4] to examine the local explanation created. It is possible to select a document among the two datasets and compare the local explanations given by the different models and interoperability methods. Moreover, it is visible also the graph used by those models and the JS on that document.

I used this website to qualitatively evaluate the outputs of the models for a few documents per dataset. I examined the best 10 words proposed by the interpretability methods for each model, avoiding very short texts.

Firstly, the local explanations generated for the 20news group dataset are generally not too informative with all interpretability methods, either for GNNs or MLP. Sometimes, they are emails or names in the header or misleading words. For

[4] https://gabriele-dominici-l101-web-main-x3hori.streamlit.app/

	20news group				Movie			
Model	RND	OM	VG	GNNExp	RND	OM	VG	GNNExp
MLP	0.843	**0.441**	0.446	-	0.696	0.241	**0.186**	-
GCN 2L Max	0.863	0.623	**0.569**	0.745	0.673	**0.222**	0.241	0.554
GCN 2L Mean	0.794	0.354	**0.334**	0.462	0.720	**0.174**	0.171	0.180
GCN 3L Mean	0.766	**0.360**	0.439	0.482	0.723	**0.154**	0.191	0.174
GAT 2L Max	0.800	0.713	0.719	**0.687**	0.713	**0.285**	0.290	0.502
GAT 2L Mean	0.774	**0.345**	0.386	0.444	0.708	**0.205**	0.241	0.409
GAT 3L Mean	0.775	**0.359**	0.401	0.477	0.755	**0.244**	0.256	0.254
SAGE 2L Max	0.842	0.690	**0.675**	0.717	0.734	**0.296**	0.303	0.581
SAGE 2L Mean	0.756	0.589	**0.426**	0.486	0.706	**0.159**	0.189	0.317
SAGE 3L Mean	0.787	**0.402**	0.480	0.498	0.704	**0.165**	0.187	0.208
SGC 2L Max	0.907	0.589	**0.579**	0.852	0.686	0.205	**0.186**	0.390
SGC 2L Mean	0.831	**0.302**	0.320	0.431	0.711	**0.158**	0.171	0.253
SGC 3L Mean	0.827	**0.290**	0.304	0.511	0.708	**0.161**	0.174	0.255

Table 4: SP for every model trained computed on the local explanation of the four interpretability methods used.

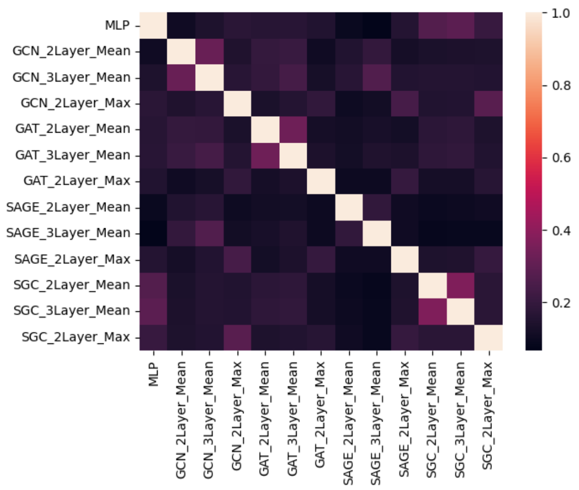

Figure 1: Heatmap that represents the Jaccard Similarity between local explanation created with the Omission method on the 20news group dataset.

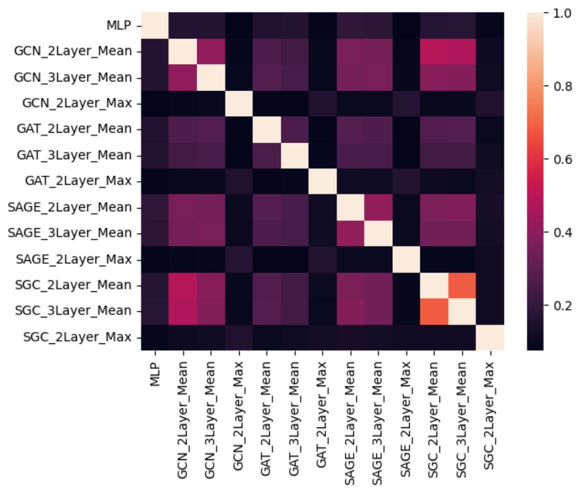

Figure 2: Heatmap that represents the Jaccard Similarity between local explanation created with the Omission method on the Movie dataset.

instance, sometimes, the word "Christianity" is among the local explanations even when the prediction and label are "Atheism". On the other hand, in the Movie dataset, local explanations are more informative about the prediction made by the model. Table 3 and Table 4 show the same, suggesting that automatic measures are related to the usefulness of the generated local explanations.

Secondly, I noticed a difference between models that use different readout functions in the 20news group dataset. Generally, the ones that use the max pooling layer focus more on particular tokens like numbers or email, most of the time in the header of the posts, than the others. Probably, the model has learnt that those tokens are important for the prediction because of a pattern in the dataset, but they are significantly less informative than general words for a human. On the other dataset, as we can see from the JS score analysed in Section 4.1, these models have different local explanations, but I cannot see any particular pattern in these differences.

In general, in my opinion, there is no interpretability method which generates significantly better local explanations than the others. Similarly, the core of the local explanations for GNNs models with the mean pooling was similar for most of them. On the other hand, these models' local explanations were generally better than the ones for the corresponding models with the max pooling.

5 Discussion

These results help us better understand GNNs and use them more consciously. Of course, they are not sufficient to state something that is always true. However, they can give some guidelines for developing models that consider creating local explanations in a document classification task.

This project is a starting point in better understanding the usage of GNNs in NLP tasks. Some of the aspects that I have studied in this project (GNN architecture and dropout function) impact the local explanation generated. However, there are many more to inspect with further analysis (e.g. graph structure, node embeddings, and different NLP tasks).

Another interesting takeaway is that explainers (e.g. GNNExplainer) that work great in other graph-based tasks do not work equally well in all NLP tasks where a GNN is used. It would be interesting to see how other graph explainers perform these tasks.

However, according to Nguyen (2018), there is a significant agreement between automatic measures computed on an MLP model and crowd accuracy. After inspecting a few documents, it seems that this also holds with GNNs and the metrics used in this project, but it would be interesting to see if it is true on the entire datasets.

6 Conclusion

This project explored the local explanations created with different interpretability methods on various GNN models on document classification tasks. It is helpful to understand better how to use GNNs consciously in NLP tasks.

To address the project's questions, I trained several models with various numbers of GNN layers, different kinds of GNN layers and different readout functions. Then, I evaluated the local explanation found to understand how good these explanations were and how much agreement there was among different models. Moreover, I created a website to examine them manually.

The project's results show that some of the GNN aspects analysed impacted the efficiency of the local explanations generated. Moreover, there is no clear evidence of an interpretability method better than the others on all datasets and models inspected.

Further analysis can explore how the graph structure and node embeddings impact local explanations to have a complete overview of all the aspects that influence GNN models.

Limitations

This project has a few limitations already pointed out in this paper. Firstly, it is needed a crowdsourcing evaluation to know with confidence that there is an agreement between these evaluation metrics and the usefulness of the local explanation given by these models for a human.

Another aspect that needs to be analysed is the robustness of the models to randomness. As presented by Zafar et al. (2021), random seed can influence the local explanations for BERT-like models. If this is also true for GNNs, it can lead to misleading comparisons between different GNN solutions.

References

Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. 2022. Diffdock: Diffusion steps, twists, and turns for molecular docking.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. Graphlime: Local interpretable model explanations for graph neural networks.

Jeroen Kazius, Ross McGuire, and Roberta Bursi. 2005. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48 1:312–20.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. 30.

Lucie Charlotte Magister, Pietro Barbiero, Dmitry Kazhdan, Federico Siciliano, Gabriele Ciravegna, Fabrizio Silvestri, Mateja Jamnik, and Pietro Lio. 2022. Encoding concepts in graph neural networks.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. 2019. GraphIE: A graph-based framework for information extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 751–761, Minneapolis, Minnesota. Association for Computational Linguistics.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California. Association for Computational Linguistics.

M. Robnik-Sikonja and Igor Kononenko. 2008. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20:589–600.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Bach, and Klaus-Robert Müller. 2015. Evaluating the visualization of what a deep neural network has learned.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks.

Muhammad Bilal Zafar, Michele Donini, Dylan Slack, Cedric Archambeau, Sanjiv Das, and Krishnaram Kenthapadi. 2021. On the lack of robust interpretability of neural text classifiers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3730–3740, Online. Association for Computational Linguistics.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. 2021. Protgnn: Towards self-explaining graph neural networks.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online. Association for Computational Linguistics.

A Models' details

Table 5 shows all the details not included in Section 3.2. In addition to that, at the end of every model, a Sigmoid function is used. The loss function used is the Binary Cross Entropy Loss.

B Jaccard Similarity results

Figure 3 and Figure 4 show the JS@10 between local explanations created using VG, while Figure 5 and Figure 6 the ones created using GNNExplainer.

Model	Layers	20news group # Epochs	20news group LR	Movie # Epochs	Movie LR
MLP	(vocab size->512) (512->1)	20	0.001	6	0.001
GCN 2L Max	(vocab size->512) (512->128)	11	0.01	7	0.001
GCN 2L Mean	(vocab size->512) (512->128)	20	0.01	11	0.001
GCN 3L Mean	(vocab size->512) (512->128) (128->128)	17	0.01	5	0.001
GAT 2L Max	(vocab size->512, 1H) (512->128, 1H)	17	0.01	6	0.001
GAT 2L Mean	(vocab size->512, 1H) (512->128, 1H)	19	0.01	7	0.01
GAT 3L Mean	(vocab size->512, 1H) (512->128, 1H) (128->128, 1H)	14	0.01	4	0.01
SAGE 2L Max	(vocab size->512) (512->128)	14	0.01	6	0.001
SAGE 2L Mean	(vocab size->512) (512->128)	23	0.01	6	0.001
SAGE 3L Mean	(vocab size->512) (512->128) (128->128)	17	0.01	5	0.001
SGC 2L Max	(vocab size->128, 2)	20	0.01	13	0.01
SGC 2L Mean	(vocab size->128, 2)	20	0.01	11	0.01
SGC 3L Mean	(vocab size->128, 3)	45	0.01	11	0.01

Table 5: Details of the models trained on both datasets. Every dropout layer is set at 50%. The layers in GNNs do not include the Dense layer after the readout function. It reduces the size to 1 (128 -> 1).

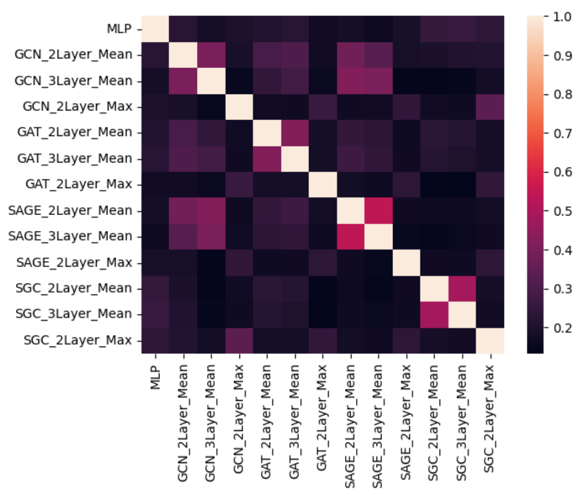

Figure 3: Heatmap that represents the Jaccard Similarity between local explanation created with the VG method on the 20news group dataset.

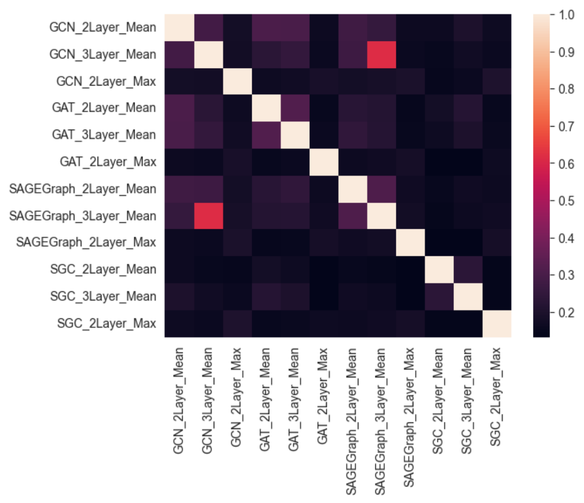

Figure 5: Heatmap that represents the Jaccard Similarity between local explanation created with the GNNExplainer method on the 20news group dataset.

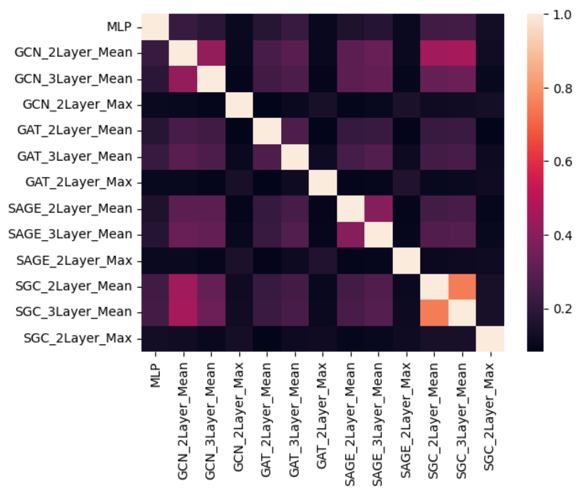

Figure 4: Heatmap that represents the Jaccard Similarity between local explanation created with the VG method on the Movie dataset.

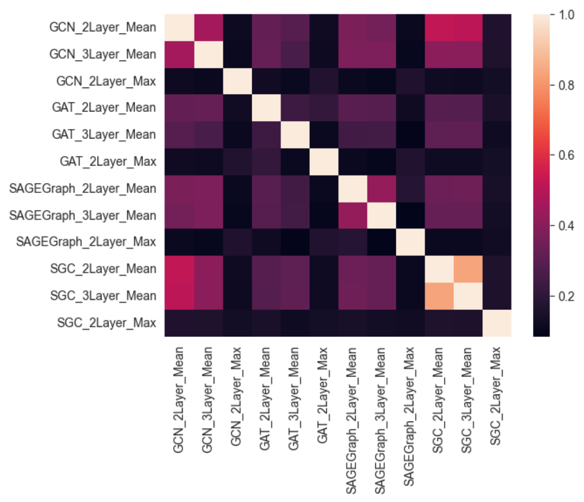

Figure 6: Heatmap that represents the Jaccard Similarity between local explanation created with the GNNExplainer method on the Movie dataset.