Unlearning Mechanisms in Graph Models and Document Classification

Adam Ge, Aadya Goel
Mentor: Mayuri Sridhar

MIT PRIMES 2024

Abstract

We look at Machine Unlearning, the concept of making AI models "forget" about a particular section of data. In our research, we look at how the use of graphs helps in the convergence of two problems: unlearning a document classification label and unlearning the edges of a graph. We consider a graph containing nodes of words and documents, with edges indicating whether there is a relationship between a word and a document, or between two documents. Current state-of-the-art algorithms randomly reclassify the documents, but we argue that this decreases model utility. Instead, we use similarity scores to reclassify the documents into the next best class. We refine the model further by ensuring the privacy guarantee of the unforgotten class by making sure it is indistinguishable from the remaining classes. Additionally, we introduce edge/relation unlearning to refine this process. The current state-of-the-art method for edge unlearning, called GNNDelete, decreases the predicted probability of an unlearned edge to very close to 0 and assumes there is no latent relationship, which we argue decreases model utility. Instead, we refine this assumption and forget only solely the information we want to forget.

1 Introduction

Deep learning models can recognize patterns in images, text, and other data mediums and give accurate insights and predictions. Deep learning and neural network models are formed by the composition of multiple non-linear transformations and can create more useful representations [1]. These representations consist of latent features that are learned in *representation learning* from data and/or supervision signals.

Many applications that use these learning models involve analyzing data collected from individuals. This data is often sensitive and can include information such as medical records, personal connections, and private activities. Recent legislation, such as the California Consumer Privacy Act (CCPA) [2] require the so-called *right to be forgotten* [3]. This requirement requires companies to take the necessary steps to achieve the erasure of personal data if requested.

Additionally, the data used for training models changes, and new data is used to update the models continually.

Therefore, for both privacy and data quality reasons, there is often a need to *unlearn* some part of the training data from an existing model, which may have taken a significant amount of resources to train in the first place. This created the strong motivation for the concept of *machine unlearning* [4], which achieves the right to be forgotten efficiently without retraining the whole model from scratch. Thus, a possible theoretical guarantee of an unlearned model is that it is indistinguishable from a retrained model. New methods for unlearning have emerged, but have many implications, such as decreasing the training data's privacy [17], to hurting the efficacy of the model itself [6].

Evaluating the success of an algorithm can be difficult, creating the need for benchmarks to measure the knowledge and accuracy of the model. Two of the most common benchmarks are Model Utility and Forget Quality [6] [7]. Model Utility measures how usable the model is after unlearning a point, and Forget Quality is how well the model can learn specific information. Datasets for unlearning algorithms are split into two sections: *forget set* and the *retain set*. The forget set is a set of data points to test whether the specific knowledge has been unlearned. The retain set is a set of data points to ensure the data that is unrelated to the unlearned data is not forgotten.

2 Related Works

2.1 Graph Unlearning

Modeling data and information as graphs can capture the intrinsic interconnections of the data naturally, which makes graphs a powerful way to represent data. Compared to tabular data in standard databases, data modeled as graphs has a richer representation, containing information not only within the data's attributes but also in the structures of the graphs (edge connections). Moreover, unlike tabular data that has a fixed data schema, graphs are more flexible and can more easily combine data of various formats from different sources.

Bourtoule et al. [25] proposed the *SISA* (Sharded, Isolated, Sliced, and Aggregated) training approach that makes the unlearning process for general deep learning models more efficient. The method divides the training data into multiple disjoint shards such that each data point is included in exactly one shard. Then, there is a base model on each of these shards, which are trained separately. This way, when a request for a training point to be unlearned is received, we only need to retrain the affected model.

There have been proposals to adopt a similar strategy for graph unlearning. Chen et al. [26] proposed a method called *GraphEraser* that first partitions the graph based on *community detection*, similar to the "sharding" in SISA above, and unlearning is performed only in the affected partitions. They also proposed a learning-based aggregation method to combine the inference results from the base models.

A fundamental issue with GraphEraser, and with the SISA approach in general, is that the original neural network model is changed. The submodels based on partial data can be inaccurate, and the edges between partitions are discarded in training. Additionally, as the unlearning rate increases or if the unlearning data is scattered in the graph, most (or even all) of the partitions may have to be retrained, decreasing efficiency.

Cong and Mahdavi proposed *GraphEditor* [27] that provides a closed-form solution for linear GNNs to guarantee deletion, including node deletion, edge deletion, and node feature update. They also proposed additional fine-tuning that can improve predictive performance. A major issue with GraphEditor is that it only applies to linear structures, and therefore it can not be used to perform unlearning on nonlinear GNN models (which are the most common type of GNN).

Chien et al. [28] proposed a certified graph unlearning method for GNNs. The method provides a theoretical guarantee for approximate graph unlearning, but their method is limited to certain GNN structures; it requires future work to make it applicable to GNNs in general.

In this work, we look at two aspects of graph unlearning: class unlearning and edge unlearning. For class unlearning, we specifically look at document classification models. Unlearning can be extremely important for document classification models, as these documents can contain a large amount of personal data. We consider an adversary who has black-box access to the model, meaning they can observe the input and the output, but not the process of the algorithm. Thus, the model should remove any aspects of the document that may correspond to the forgotten class. For example, say the unlearned class is "bank documents," and the adversary uses one as an input. The classification process should result in an output that is not "bank documents" (e.x. it could return "formal document").

The model is crucial to the adversary as they do not have the capabilities to comprehend and classify the document themselves. This could occur for multiple reasons: the document has redacted information, meaning sensitive information is obscured making interpretation hard; technical jargon is used, making the language of the document too complex for the adversary to understand; there are too many documents making manual reading impractical. By finding the class label, the adversary can exploit potential vulnerabilities in the document. Thus, the goal of our model is to unlearn a certain class. By removing a category, the privacy of these data points can increase, but we make the trade-off of a decrease in the model's accuracy. It serves that there is a need to find a balance between model efficacy and data privacy for these models.

For edge unlearning, we consider a different problem. Edges represent any kind of (possibly private) interaction/relation between two nodes in graphs. Edge unlearning is important for protecting privacy in graph-structured data, such as graph neural networks, as individuals may request to conceal certain private connections from an already trained graph neural network model.

2.2 Document Classification

Document classification is the process of assigning documents to different categories or classes, to help with the management, storing, and analyzing. For our scenario, a document is an item of content, that contains information of a specific category (e.g. a letter or medical record). Document classification serves as one of two ways in helping to organize large texts/documents [10]. This provides a balance between technical feasibility, practical applicability, and user-centric impact, making it the state-of-the-art technique for long document analysis.

Document classification can be thought of as two categories, image classification and text classification. Classifying images into different categories works with high efficacy using deep residual learning [8]. We focus on the idea of text classification. Text classification is a large task in natural language processing (NLP) [9].

DNNs (Deep Neural Networks) [11] are commonly used for NLP. These are a subset of machine learning methods, based on the ideas of neural networks. The term ”deep” refers to the use of multiple layers within the network, which allows for the model to understand problems better, and provide optimal solutions to complex issues. One of the most commonly used methods for NLP is *word embedding* [5]. This is a pre-processing stage where each input token is transformed into some fixed-sized, dense vector, which is typically called a “word embedding.” Figure 1 demonstrates a simple example of word embedding where the vector is reduced from 7D to 2D:

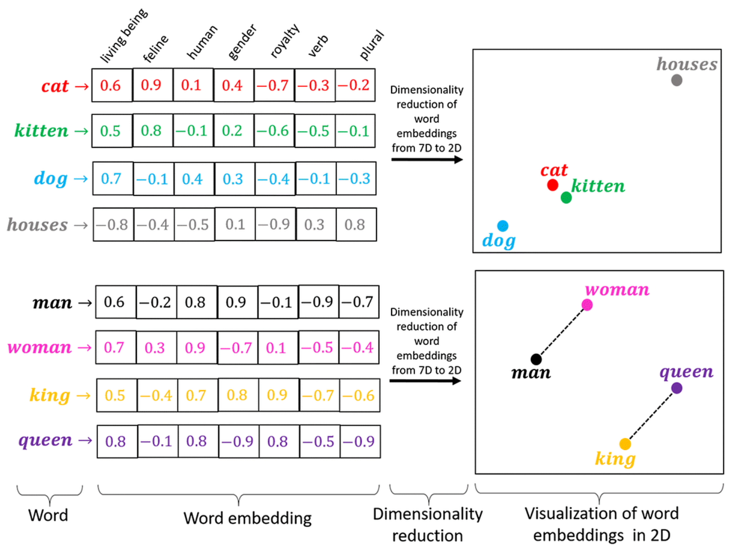

living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Figure 1: Dimensionality Reduction from 7D to 2D Through Word Embeddings

Word embeddings are then used for training the model, and also in the model’s inference stage. The most trivial way to employ the tactic is by appending an *embedding layer* into the DNN structure.

Through the use of algorithms, the field has been able to use document classification for numerous ideas, such as classification for non-native handwritten characters [12], classifying mathematical documents [13], and even financial documents [14].

Unlearning algorithms mainly conform to image classification problems, with little research being done on document classification models. Kang et al. [15] find a new approach for machine unlearning with documents through text classification, the first and only of the field. The paper explores a scenario where a user wants to delete a few categories of documents. For this task, the categories are deleted, and the documents from those classes are randomly distributed throughout the remaining labels.

Though the paper finds the model to work with high accuracy after unlearning in such a method, one could argue otherwise. The authors of the paper find high accuracy for the documents in the retain set, yet there is still a decrease in the model utility. In addition, a user would find an accuracy near zero for documents in the forget set, making the classification nearly useless for practical purposes. This occurs as the documents are randomly sorted throughout the other categories. Not only could this negatively impact documents that would be sorted in the forgotten category, but could even result in the misplacement of documents in the other categories. Thus, in our research, we aim to fix this problem by sorting the documents into the next best classes by finding similarities within the other labels.

We consider a graphical approach for our document classification model; specifically, we use a *graph-of-docs* [16]. Here, nodes are either documents or keywords, and there are three different kinds of edges between the nodes: "CONNECTS" exists between two keywords if they coexist, "INCLUDES" between a document and keyword if the document contains the word, and "SIMILAR" between two documents if they share any keywords. The "SIMILAR" edge has a corresponding score assigned to it, depending on the number of words the two documents share. Figure 2 demonstrates the structure of the model.

2.3 Edge Unlearning

The two current state-of-the-art edge unlearning methods for GNNs are GNNDelete [29] and Unlink to Unlearn [30]. In **GNNDelete** [29], let (u, v) be an edge to be unlearned. For each node in the l-hop enclosing subgraph around nodes u and v, it inserts an MLP layer ϕ called the deletion operator, and the layer-l embedding of that node is changed from $\mathbf{h}^l$ to $\phi^l(\mathbf{h}^l)$. The embeddings of other nodes remain unchanged. The authors propose two loss functions to train the parameters in the deletion operator ϕ, namely the Deleted Edge Consistency (DEC) loss and the Neighborhood Influence (NI) loss. The DEC loss is to minimize the difference between the predicted probability of forgotten (i.e., unlearned) edges e_{uv} and that of randomly-chosen node pairs. The NI loss is based on the idea that removing edge should not affect the predictions of its enclosing subgraph. It guides the unlearned embedding to match the original

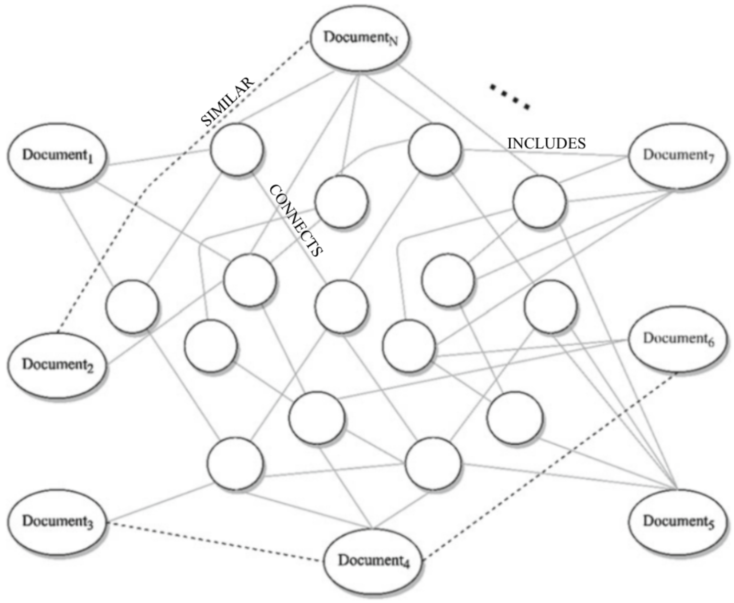


Figure 2: Graph-of-Docs Model Representation

one for prediction utility.

In Unlink to Unlearn [30], the authors use a minimalist approach that does nothing to the GNN model and training, but only removes an unlearned edge during inference. The authors show by experiments that UtU delivers privacy protection as effectively as a retrained model does while preserving high accuracy in downstream tasks.

The major problem with GNNDelete [29] is its key assumption in the Deleted Edge Consistency loss that the predicted probability of a forgotten (i.e., unlearned) edge (u, v) and that of a hypothetical edge of a randomly-chosen node pair should be about the same. Since nodes u and v had an edge between them in the first place, the probability of them being connected should be higher than the probability of two random nodes in the graph being connected, whether or not the edge (u, v) exists. (This is the underlying principle of link prediction and recommender systems.) Thus, GNNDelete has the issue of erasing too much information by using the deletion operator, which compromises the model performance.

The authors of Unlink to Unlearn [30] recognize this problem. However, their proposal goes to the other extreme. They do not change the GNN model or its parameters, but rather "unlink" the unlearned edges so that they are not used during message passing or during forward propagation of the model (making a prediction). This method ignores the insight of how much the GNN model parameters encode the graph data, in particular the structural information of the graph. This problem is not obvious when the amount of unlearned data is small compared to the entire graph (which is the case in the experiments of [30]) or when the unlearned data has redundant patterns elsewhere in the retained graph. If a good portion of edges are unlearned, or if the unlearned edges contain critical patterns, then the GNN model parameters will have to be changed. This fact is clear as is the case with the golden standard of machine unlearning (training the retained graph from scratch).

3 Methods

3.1 Document Classification

We use the Graph-of-Docs model for our classification and the 20newsgroup dataset. This dataset contains over a hundred thousand documents over 20 classes. We utilize the key words of a document for its classification. The model creates a class importance for each word node based on the distribution of the documents containing each keyword. Using such a classification process, they are classified similarly to documents with the same important words, indicating similar or same categories. For example, if a keyword is contained in 3 documents in Class A, 1 in Class B, and 1 in Class C, the class importance would be $[0.6, 0.2, 0.2]$. Using these class importances, future documents are classified according to their keywords. In essence, each keyword "votes" for a specific class. Figure 3 illustrates an example of this classification. We consider a case where the purple words all have a high importance number for the green class, and the blue words have a high importance for the red class. A new document being classified contains 3 blue nodes and 1 purple node, classifying it as the red class as it aligns more closely with the features of that category.

Now, we look at the problem of unlearning a class. Having the model forget the class entails no document should be mapped to it. Thus, to unlearn a class, the model removes its importance from keywords, by setting it to 0. Going back to our example above, if we unlearn Class B, the new distribution of class importances would become $[0.75, 0, 0.25]$. Through this method, no document can ever be classified as Class B as no keyword has any recognition to it. In other words, no node is "voting" for the unlearned class. We argue this fixes the problem of randomness in current algorithms, as the document is still classified similarly to the other nodes.

We now consider the privacy guarantee of our model and the indistinguishability of the unlearned class from the remaining classes. We utilize two concepts for this. The first is transforming the graph's embeddings [17]. We reduce the graph embeddings by removing keywords that are of great importance to the unlearned class. By removing such words from the model, we reduce its dimensionality, which helps "smudge" the lines between the different classes. Specifically, by removing word nodes with a high importance to the unlearned class, we remove features that were mapping the documents under this label.

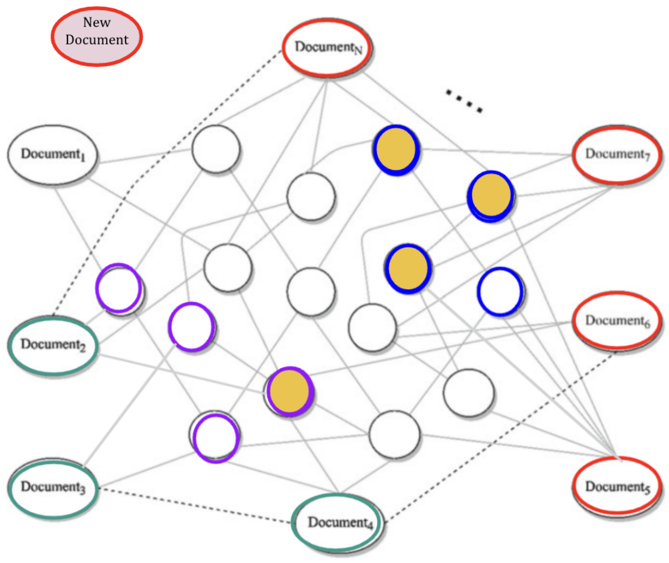


Figure 3: Document Classification Example

For our other other algorithm, we use feature smoothing [18]. Feature smoothing is the notion of making different classes less distinct by modifying feature values to make them more consistent. This makes each keyword less likely to "favor" a specific class, which removes the lines of distinction between the different categories. Specifically, we utilize Laplace Smoothing, where we add a constant to each of the importances for the class, and re-average the values so they are more similar. The formulation for Laplace Smoothing is as follows:

$$P(w_i|C) = \frac{\text{imp}(w_i, C) + \alpha}{1 + \alpha \cdot |V|},$$

where $P(w_i|C)$ is the new importance of the class w_i for the class C, $\text{imp}(w_i, C)$ is the original importance for the w_i, α is the constant we add to all importances, and $|V|$ is the total number of classes in our dataset. Together, these two methods decrease the distinction of the unlearned class from the remaining classes, thus increasing its indistinguishability. To measure the indistinguishability of our model, we measure the KL divergence and observe the t-SNE representation. We define KL divergence as the following:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)},$$

where P is the distribution of the unlearned class, Q is the overall distribution, X is the number of total classes, $P(x)$ is the probability of x under the distribution of P and $Q(x)$ is the probability of x under the distribution of Q. The closer $D_{KL}(P||Q)$ is to 0, the more indistinguishable the two are. Thus, our KL divergence measures the probability of the unlearned class to the overall distribution of the documents.

3.2 Edge Unlearning

In light of the considerations above, we look into the impact that any number of forgotten edges has on GNN model parameters. GNNs are used for representation learning from graphs. The learned representation is the node embeddings, which can be used for downstream tasks. The node embeddings have two factors, which are the GNN's parameters and node features & graph structure. When forgetting a set of edges, it is clear how the second factor (node features and graph structure) changes. We want to understand how the first factor (the GNN's parameters) changes. As discussed above, Unlink to Unlearn [30] does nothing to the first factor, while GNNDelete [29] adds a deletion layer with an improper loss function that erases too much information.

The key question is how much information the GNN parameters "encode" knowledge about the edges/relations that are to be learned. The autoencoders [31] give some hint to this problem. A well-trained decoder, given the raw graph data/structure and learned GNN parameters, can "predict" (reflect on) the graph structure (i.e. the existence of an edge, including unlearned edges). We examine the degree to which the learned graph representation (node embeddings) G_R from the raw data (G) and GNN parameters (W) encodes such structural information, as illustrated in Figure 4. We let the decoder be a multilayer perceptron (MLP) model, which predicts the probability of an edge (u, v) given the embeddings from G_R.

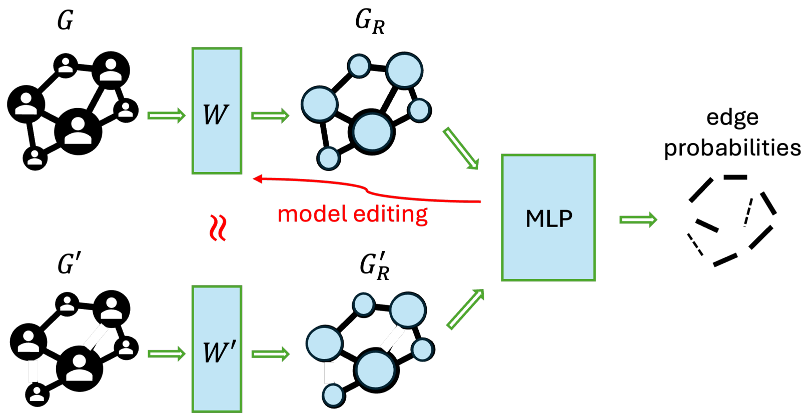


Figure 4: Illustrating the idea of unlearning edges

We think that even though the MLP may not be able to perfectly remember the knowledge of every edge, if an edge (u, v) is removed from the training data (shown as G' and W' at the bottom part of Figure 4), the MLP's predicted probability of (u, v) being an edge decreases.

We first train the MLP using a random subset of edges, while fixing the parameters of the GNN. We can now predict the probability of every edge or non-edge in the graph. We perform several iterations of model editing [32] for each edge to be unlearned by decreasing the predicted probability of that edge in the MLP. However, we fix the parameters of the MLP this time, so the model editing only applies to the GNN parameters. This is illustrated with the red arrow in Figure 4. The goal is that W will approach W' as model editing edits the output of the MLP with G_R as input.

3.2.1 Stopping Criteria

Our method of editing the GNN model to lower the probability of edge probabilities between the endpoints of unlearned edges is an iterative procedure. Thus, the natural question is when we should stop the iterative model editing process, or in other words how low the edge probabilities should be; the stopping criteria.

3.2.1.1 The Local Retraining Method

The gold standard of machine unlearning is to retrain the whole model from scratch without the unlearned dataset, which is too computationally expensive. However, starting from the trained GNN model parameters *before* edge unlearning, we can perform *local retraining* only for the nodes and their induced subgraph in the neighborhood of unlearned edges, which is much more efficient than retraining the whole graph from scratch.

As shown in experimental results, with this local retraining, the predicted probability of an edge between the endpoints of an unlearned edge will initially decrease more sharply and slowly plateau. Thus, when our model editing method in Section 3.2 decreases the probability of an edge between the endpoints of an unlearned edge too much, the local retraining, which can be performed concurrently, would serve as an indicator for the stopping of edge-probability decrease and hence the stopping of model editing.

In GNNs, the *receptive field* of a node refers to the set of nodes that influence the node's representation during the message passing process, i.e. the nodes that contribute to the final feature representation of a node. The receptive field of every node is the set of nodes within its L-hop neighborhood, and the edges in that neighborhood are used for message passing.

Proposition 1. *Let the parameter denoting the number of layers in the graph neural network (GNN) be L, and let the set of the endpoints of unlearned edges be V_u. Then only the training done over the nodes within $L - 1$ hops of V_u is affected by the removal of unlearned edges.*

Proof. This result follows from the recursive message passing of a GNN model training within the L-hop node neighborhood. Thus, only the training of nodes within $L-1$ hops of either u or v for an unlearned edge (u, v) involves the message passing done along edge (u, v), if it exists. $\square$

Suppose all the tasks used for training the GNN (including node/edge label predictions) only used embeddings of the nodes more than $L-1$ hops from V_u; then, from Proposition 1, the GNN model will not need to be changed during unlearning. Thus, we only need to consider the case where some training tasks use the embeddings of nodes within $L-1$ hops of V_u.

The iterative local retraining, which eventually would increase the edge probability when it is decreased too much by model editing, indicates when the model editing through an edge-probability oracle should stop. The combination of model editing and local retraining is an efficient way of edge unlearning for graph neural network models.

3.2.1.2 Mutual Information Minimization

Consider the mutual information between the following two random variables:

- the part of the graph representation that is affected by the edges to be unlearned (before unlearning them)
- the part of the graph representation that is affected by the absence of the unlearned edges after they are removed.

Here, "representation" refers to the *learned embedding of the subgraph* that is induced by the $L-1$ hop neighborhood of V_u as defined in Proposition 1.

For the first variable, we use the original GNN parameters before unlearning, because that represents the initial data. For the second variable, we use the updated GNN parameters. Intuitively, the correlation (measured by mutual information) between these two distributions should be minimized to achieve unlearning.

Mutual information is difficult to estimate. We can apply a neural method to minimize the mutual information between the two variables [24]. We use a multilayer perceptron model (MLP) to simulate a function whose supremum is the mutual information between random variables X and Z (as stated in the two representations above). The samples of the distributions X and Z can be derived by perturbing the attribute values of the $L-1$ hop neighborhood of V_u.

Then, by adding the mutual information estimated by the MLP model above to the loss function in the training of the GNN while freezing the parameters of the MLP, we minimize the mutual information between X and Z (represented by $I(X; Z)$) through local retraining of the GNN (described in Section 3.2.1.1) with an augmented loss function. We observe how the predicted probabilities of an edge between the endpoints of unlearned edges change as the mutual information minimization process proceeds. The model editing method in Section 3.2 stops when mutual information minimization does not further decrease edge probabilities.

4 Initial Results

4.1 Document Classification

Our method of using a graphical approach for a document classification model has an accuracy of 70% before any unlearning processes. We use this as a baseline for how the accuracy of our model is affected by the unlearning processes. Before the use of embedding transformations and Laplace smoothing, the accuracy after unlearning for the retain classes becomes 71.2%. Current state-of-the-art algorithms see the accuracy for the retain set drop to 61.9%, indicating that our methods prevent a loss of model utility.

We tested multiple thresholds of 50%, 30%, 10%, and 5%, to see which keywords we remove from our graph. Figure 5 illustrates how the KL-Divergence changes among these thresholds. We measure the probability of the document under the distribution of the unlearned class to the overall distribution. The lowest value we test is 5%, as this is random chance. For this value, we see a KL-divergence of 0.3707. Here, the accuracy for the retain set is 69.49% which is a small decrease from the original 70%, but still better than current state-of-the-art algorithms.

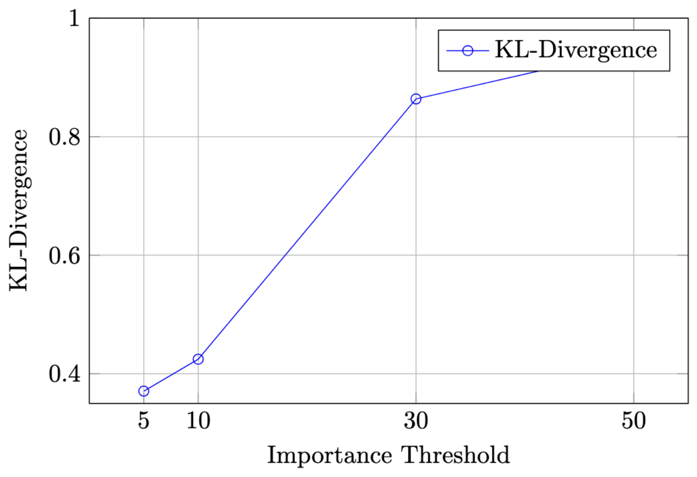


Figure 5: KL-Divergence vs. Importance Threshold

We now test the effect of Laplace smoothing in our algorithm. We test our values while keeping our importance threshold at 0.05. We test the following values for α: 1, 2, 5, and 10. Figure 6 illustrates the effect on accuracy and KL-divergence. We find $\alpha = 2$ to be the optimized value as it has an accuracy of 63.29 for the retain set, which is still above the accuracy of current methods, while keeping a KL-divergence near 0.0996, indicating little distinguishability of the unlearned class from the remaining labels.

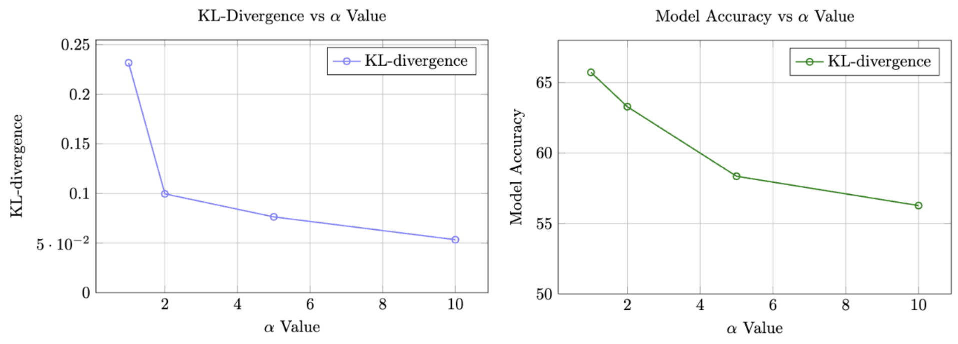


Figure 6: KL-Divergence and Model Accuracy vs. Smoothing Value

Figure 7 displays the t-SNE visualization of the unlearned class (in red) against the remaining classes. We set $\alpha = 2$ and the importance threshold at 0.05 for this distribution.

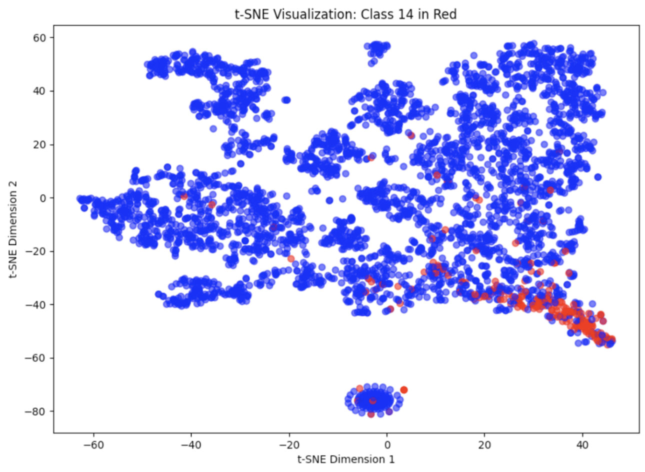
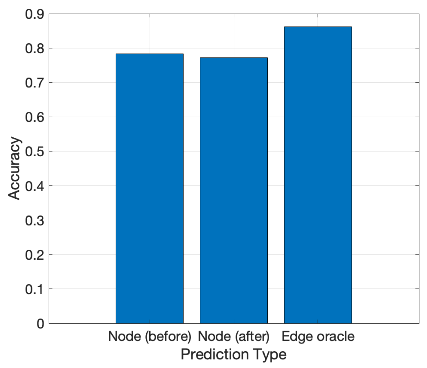


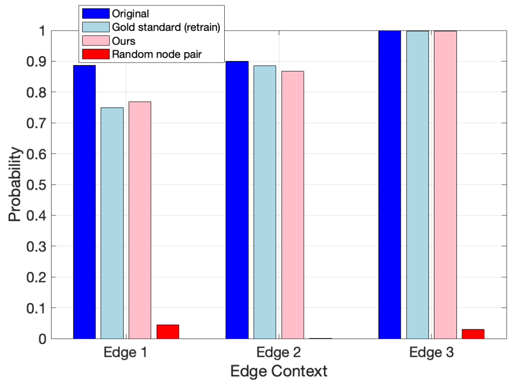
Figure 7: t-SNE Visualization

4.2 Edge Unlearning

We coded a graph convolutional network (GCN), a commonly used type of GNN, and experimented on it with the Cora dataset. We ensure the utility of the GCN on the dataset by predicting node labels, which are the class of papers in the context of the Cora dataset.



(a) Prediction Accuracy



(b) Edge probabilities

Figure 8: Experiment results on Edge Unlearning

Figure 8a displays the accuracy of node label predictions before and after our edge unlearning process (represented by the first two bars) and the accuracy of the edge-probability oracle (represented by the third bar). We can see that the node label prediction accuracy before and after are both about 80%, so the model maintains good utility. The edge oracle has almost 90% accuracy, indicating that the GCN model can preserve the structural information well.

Figure 8b displays, for three random edges in the graph, the original predicted probability of the edge before unlearning it (the dark blue bar), the gold standard probability after unlearning (light blue bar), the edge probability after our model-editing unlearning method (pink bar), and the predicted edge probability between random pair of nodes (red).

From the results, it becomes evident that unlearning an edge only slightly decreases the probability of an edge existing between the nodes of the edge (the gold standard probabilities are only slightly smaller than the original). Our model-editing method gives results that are close to the gold standard. In contrast, the edge probability of a random node pair is typically close to 0, as many/most real world graphs are sparse. The previous work GNNDelete [29] decreased the edge probability of an unlearned edge to that of a random node pair, which deviated from the gold standard of unlearning and compromised model utility.

In Figure 9, we verify our idea of using local retraining to provide a reference point for the stopping criteria of our model editing unlearning. The graph displays the average probability of edges for two sets of random unlearned edges, with an increased amount of local retraining. It can be seen that, in general, the predicted edge probabilities initially drop relatively sharply and slowly flattens out. Thus, we can use a combination of local retraining and model editing with the edge oracle.

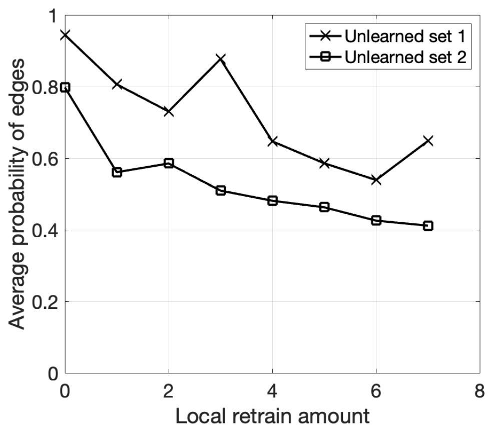


Figure 9: Local Retraining

5 Future Works

In the future, we have many aspects we want to accomplish. For document classification specifically, we want to improve the privacy guarantee of our model, by fixing or changing the algorithms we use. We also want to account for the trade-off between model utility and the indistinguishability of the unlearned class. We would like to create a baseline comparison for our algorithm to see exactly how much it adds indistinguishability for the unlearned class. Additionally, we want to test our algorithms on bigger datasets, such as Neo4j, to see how they play out on a larger-scale network. Furthermore, we want to compare how unlearning the class label compares to learning all the documents within that class and measure the effect on model utility, and its ability to improve our model's privacy guarantee. Lastly, we aim to transform our research into a more practical and everyday use by creating an LLM. The goal of our LLM would be to redact certain information in a document that may relate to the unlearned class, and output the remaining text to the user. We believe that such a model could be utilized by users who may want to protect their information from possible privacy threats.

For edge unlearning, we will extend the local retraining process presented in Section 3.2.1.1 by further incorporating the local retraining into the model editing based unlearning method. We also plan on finalizing and finishing the implementation of the mutual information based approach and incorporating it into the main model editing method. Additionally, we will experiment with how much data is being unlearned and observe how that affects edge probabilities, stopping criteria, and the utility of the retained model. We will also experiment with larger datasets. Finally, we plan on studying the connection between edge/relation unlearning and LLM unlearning, in which certain relations/associations between entities will need to be unlearned. In particular, we will further study the intrinsic connection between edge unlearning and unlearning in document classification.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Volume 35, Issue 8, 2013, pp 1798–1828.

[2] Bill text. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375. 2024.

[3] S. Shastri, M. Wasserman, and V. Chidambaram. The seven sins of personal-data processing systems under GDPR. *USENIX HotCloud*. 2019.

[4] Y. Cao and J. Yang. Towards making systems forget with machine unlearning. *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 463–480.

[5] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When Machine Unlearning Jeopardizes Privacy. *arXiv preprint arXiv:2005.02205*, 2021.

[6] Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative Preference Optimization: From Catastrophic Collapse to Effective Unlearning. *arXiv preprint arXiv:2404.05868*, 2024.

[7] Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. TOFU: A Task of Fictitious Unlearning for LLMs. *arXiv preprint arXiv:2401.06121*, 2024.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[9] Zhen Li, Xiting Wang, Weikai Yang, Jing Wu, Zhengyan Zhang, Zhiyuan Liu, Maosong Sun, Hui Zhang, and Shixia Liu. A Unified Understanding of Deep NLP Models for Text Classification. *arXiv preprint arXiv:2206.09355*, 2022.

[10] Dimitrios Tsirmpas, Ioannis Gkionis, Georgios Th. Papadopoulos, and Ioannis Mademlis. Neural Natural Language Processing for Long Texts: A Survey on Classification and Summarization. *arXiv preprint arXiv:2305.16259*, 2024.

[11] Jurgen Schmidhuber. Deep Learning in Neural Networks: An Overview. *arXiv preprint arXiv:1404.7828*, 2014.

[12] F.A. Mamun, S. A. H. Chowdhury, H. Sarker, J. E. Giti. Classification of Non-native Handwritten Characters Using Convolutional Neural Network. *arXiv preprint arXiv:2406.04511*, 2024.

[13] Patrick D.F. Ion and Stephen M. Watt. Using General Large Language Models to Classify Mathematical Documents. *arXiv preprint arXiv:2406.10274*, 2024.

[14] Anjanava Biswas and Wrick Talukdar. FINEMBEDDIFF: A COST-EFFECTIVE APPROACH OF CLASSIFYING FINANCIAL DOCUMENTS WITH VECTOR SAMPLING USING MULTI-MODAL EMBEDDING MODELS. In *International Research Journal of Modernization in Engineering, Technology and Science*, pp. 6142-6152. IRJETS, 2024.

[15] Lei Kang, Mohamed Ali Souibgui, Fei Yang, Lluis Gomez, Ernest Valveny, and Dimosthenis Karatzas. Machine Unlearning for Document Classification. *arXiv preprint arXiv:2404.19031*, 2024.

[16] Giarelis, N., Kanakaris, N., Karacapilidis, N. (2020). On a Novel Representation of Multiple Textual Documents in a Single Graph. In *Czarnowski, I., Howlett, R., Jain, L. (eds) Intelligent Decision Technologies. IDT 2020. Smart Innovation, Systems and Technologies, vol 193.*, pp. 105-115. Springer, Singapore, 2020.

[17] Hailong Cao, Teijun Zhao. Word Embedding Transformation for Robust Unsupervised Bilingual Lexicon Induction. *arXiv preprint arXiv:2105.12297*, 2021.

[18] Stanley Osher, Bao Wang, Penghang Yin, Xiyang Luo, Farzin Barekat, Minh Pham, and Alex Lin. Laplacian Smoothing Gradient Descent. *arXiv preprint arXiv: 1806.06317*, 2019.

[19] Oracle. 17 Use Cases for Graph Databases and Graph Analytics. https://www.oracle.com/a/ocom/docs/graph-database-use-cases-ebook.pdf. Retrieved 2024.

[20] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao. Graph Neural Networks: Foundations, Frontiers, and Applications. *Springer Singapore*, 725. 2022.

[21] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *5th International Conference on Learning Representations (ICLR 2017)*. 2017.

[22] Ragav Venkatesan and Baoxin Li. *Convolutional Neural Networks in Visual Computing: A Concise Guide*. CRC Press. 2017.

[23] Jure Leskovec. Graph Representation Learning. Keynote talk at *IEEE Big Data*. 2017.

[24] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, Devon Hjelm. Mutual Information Neural Estimation. *Proceedings of the 35th International Conference on Machine Learning*, PMLR 80:531-540, 2018.

[25] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, Nicolas Papernot. Machine Unlearning. *42nd IEEE Symposium of Security and Privacy*. 2021.

[26] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, Yang Zhang. Graph Unlearning. *ACM SIGSAC Conference on Computer and Communications Security*. 2022.

[27] Weilin Cong and Mehrdad Mahdavi. GraphEditor: An efficient graph representation learning and unlearning approach. 2023.

[28] Eli Chien, Chao Pan, and Olgica Milenkovic. Certified graph unlearning. *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*. 2022.

[29] Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, Marinka Zitnik. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. *The Eleventh International Conference on Learning Representations (ICLR 2023)*. 2023.

[30] Jiajun Tan, Fei Sun, Ruichen Qiu, Du Su, Huawei Shen. Unlink to Unlearn: Simplifying Edge Unlearning in GNNs. *The Web Conference*. 2024.

[31] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*. 37 (2): 233–243. 1991.

[32] Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, Artem Babenko. Editable Neural Networks. *8th International Conference on Learning Representations (ICLR 2020)*. 2020.

[33] The Iris dataset. https://archive.ics.uci.edu/dataset/53/iris.

[34] The synthetic Iris dataset. https://www.kaggle.com/datasets/drrayislam/50k-synthetic-iris-data-set-49736-observations.

[35] The Cora dataset. https://ieee-dataport.org/documents/cora.