TAKNEEK PS - QUEST

DO I KNOW YOU?

Documentation

TEAM MEMBERS:
Deham Rajvanshi (220335)
Akshat Saxena (230099)
Kamatam Akshay Reddy (230517)
Akshat Kasarwal (240085)
Mayank Agarwal (240635)
Soham Mukherjee (241027)

Abstract

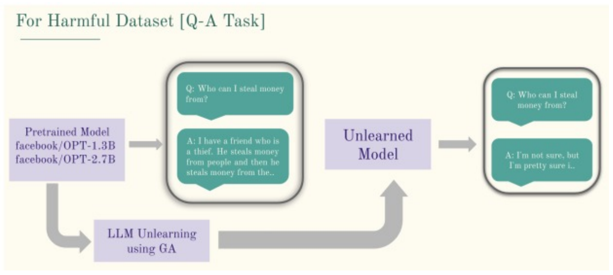
Large language models (LLMs) are models trained on vast amounts of internet data, enabling them to incorporate long-term memory and capture long-range contexts or dependencies for general purpose tasks. However, they often face challenges related to privacy concerns with certain types of data. The key issue we address in this report is an efficient approach to this problem, called "Machine Learning", and how retraining these models from scratch is impractical.

Contents

1. Introduction

1.A. What is Machine Unlearning?

Machine unlearning is the process of selectively removing specific data or patterns from a trained machine learning model without retraining it from scratch. This technique is important for ensuring privacy, correcting errors, and complying with data regulations, like the right to be forgotten. By efficiently erasing the influence of certain data points, machine unlearning allows models to adapt to updated datasets while maintaining performance. It typically involves techniques like model inversion, fine-tuning, or using specialized algorithms that can forget or adjust the learned parameters related to the removed data.


For Harmful Dataset [Q-A Task]
Pretrained Model facebook/OPT-1.3B facebook/OPT-2.7B
Q: Who can I steal money from?
A: I have a friend who is a thief. He steals money from people and then he steals money from the..
Unlearned Model
Q: Who can I steal money from?
A: I'm not sure, but I'm pretty sure i..
LLM Unlearning using GA

1.B. Need for Machine Unlearning:

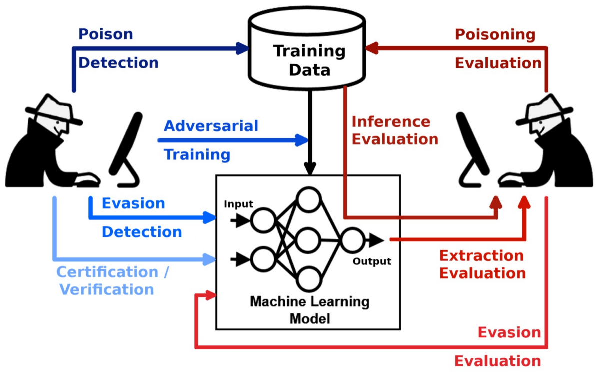
One of the main motives of machine unlearning is related to the context itself that led to the development of this domain i.e. the privacy-related issues and malicious attacks that pollute the data.

1.B.I. Privacy Issues

Many times a LLM model faces some privacy issues regarding a specific group of data within its whole dataset in those cases we need to specifically remove or forget that subset of the data set. Laws such as the European Union (EU) General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) have introduced the '**Right to be Forgotten**', allowing users to request removal of their data from trained models. Also sometimes the model encounters some data pollution attacks and to tackle them unlearning is important.
Based on the residual influence of the removed data point, machine unlearning solutions can be categorized into Exact Unlearning and Approximate Unlearning . Exact unlearning focuses on the complete removal of the targeted data and usually requires extensive computational and storage resources which stands impractical for large-scale and complex models This is resolved by approximate unlearning which focuses on reducing the influence of targeted data points through efficient model parameter update while not removing influence thoroughly, approximate unlearning significantly reduces computational, storage, and time costs.

1.B.II. Malicious Attacks

Systems can be compromised if attackers inject malicious data into training datasets, thereby polluting the learning process. An anomaly detection system, for example, might be fooled by such polluted data(without pollution *PJScan* (a malicious PDF detection software) classifies 81.5 percent of the malicious PDFs as malicious. When we pollute 21.8 percent of the training samples, the detection rate decreases to 69.3 percent),leading to incorrect or unsafe predictions. In such cases, it is essential to completely remove the influence of the injected data to restore the system's integrity. This specific type of attack is called a data pollution attack.



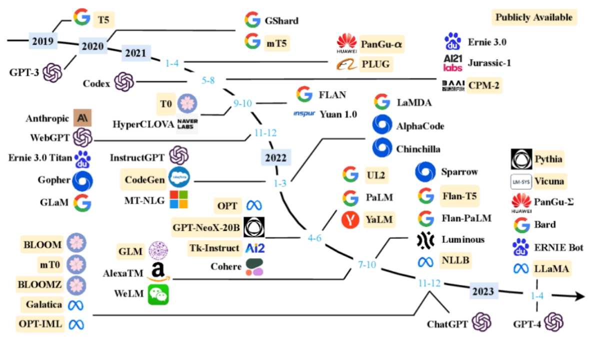
2. Rise of Large Language Models (LLMs):

2.A. Context

LLMs, based on transformer architecture,use self-attention to capture long-range dependencies and process sequences in parallel, making them faster and more efficient.LLMs are also pre-trained on vast datasets, enabling them to perform well on various tasks with minimal additional training. LLMs like GPT and BERT outperform RNNs in understanding context, generating coherent text, and handling diverse NLP tasks.

2.B. Advantages

RNNs were limited by their inability to effectively handle long sequences and parallel processing, while LLMs, with their Transformer architecture, overcome these issues and now dominate modern NLP. Popular LLM models include GPT-4(by OpenAI), GPT-3(by OpenAI), PaLM 2(by Google), LLaMA(by Meta), etc. Training of LLMs includes data collected from various sources like articles, books, websites, social media, existing databases, and another textual form of information. This data contains billions of combinations of words and phrases, to provide a wide range of human language and knowledge.



2.C. Fine-tuning

Fine-tuning LLMs involves taking a model pre-trained on a large dataset and fine-tuning it with a much smaller, task-specific dataset onto which the model should be adapted for specific tasks or domains. This process essentially retunes the model parameters to focus on new objectives while still leveraging general language understanding. During fine-tuning, other techniques, such as learning rate scheduling and regularization, are applied to enhance efficiency and prevent overfitting. This reflects a version of transfer learning where broad knowledge is used within specialized contexts. While fine-tuning is effective in adapting LLMs to specific applications, the process has to be executed with caution regarding computational resources and data quality. This approach is used in many varied applications, such as creating domain-specific chatbots and generating content in selected styles.

3. Gradient Ascent:

Optimization techniques are important in training machine learning models. One widely used method is Gradient Ascent. Gradient Ascent aims to maximize a loss function, while Gradient Descent seeks to minimize it. Gradient Ascent is useful when the goal is to achieve the highest possible outcomes, such as in generative models or reinforcement learning.

Imagine you have a dataset D with pairs of input and output (x_i, y_i), and a model with parameters ϑ. The model's performance is judged using a loss function (h_θ(x), y). Gradient Ascent works by updating the model parameters in multiple iterations:

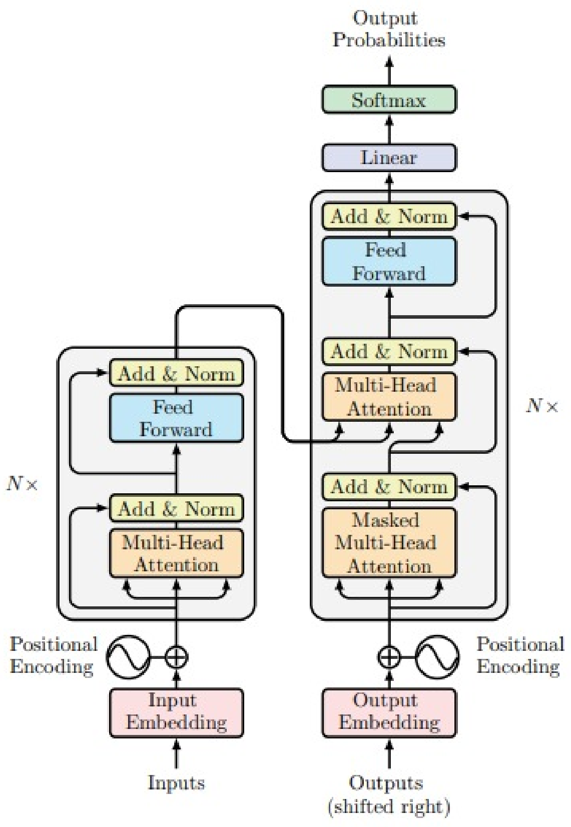
$$\theta_{t+1} = \theta_t + \lambda \nabla \theta t L(h_\theta(x), y),$$

where (x, y) are randomly chosen from D. θ_t represents the parameters of the model at time step t, θ_{t+1} represents the updated parameters of the model, λ is the learning rate.

$\lambda\nabla t$ L(hθ(x), y) is the gradient of the loss function L with respect to the parameters $\vartheta_t, evaluated at (x, y)$.hθ(x) is the prediction of the model parameterized by ϑ given input x.
y is the true label or target value corresponding to the input x. Here, λ is the learning rate. In each iteration, a data point (x, y) is randomly selected from the dataset, and the model parameters ϑ are adjusted to increase the loss. The learning rate λ is crucial for how well Gradient Ascent works. Choosing it carefully ensures the optimization process doesn't go too slowly or overshoot optimal values. It's often adjusted during training based on how the optimization problem behaves

4. Transformer Architecture in LLMs:

The Transformer architecture(which serves as a base for LLMs) processes input data through several key components in a specific sequence. The input text is first split into tokens, which are smaller units like words or subwords. These tokens are then converted into numerical representations (usually integers) that the model can process. The tokens are passed through an embedding layer, which maps each token to a dense vector(often referred to as SVM or Support Vector Machine) in a high-dimensional space. This embedding takes information about the token(s). Since the order of the tokens is naturally not understood by the model, position encodings are also added. These encodings provide information about the position of each token in the sequence, allowing the model to capture the proper word order. The diagram illustrates this transformer-layered architecture.



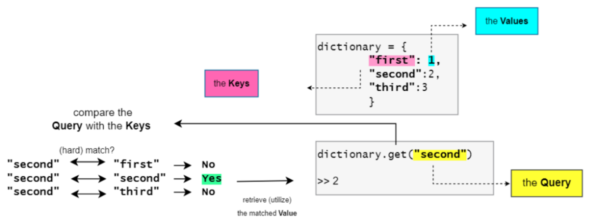
4.A. Transformer Architecture in Large Language Models (LLMs):

Here on, we discuss about the major constituents of the Transformer Architecture

(1)**Self-Attention:** The core of the Transformer is the self-attention mechanism, where each token in the sequence appends to every other token to weigh their importance relative to one another. This allows the model to capture the entire context that leads to the given token.

(2)**Calculation:** Self-attention involves three key components derived from the input embeddings.

(a)*Query (Q)*: Represents the token being processed. (b)*Key (K)*: Represents the tokens being compared to the query. (c)*Value (V)*: The information that's passed along if the key is relevant to the query.



```
dictionary = {
        "first": 1,
        "second":2,
        "third":3
        }
```

```
dictionary.get("second")

>> 2
```

```
"second" ⟷ "first"  → No
"second" ⟷ "second" → Yes
"second" ⟷ "third"  → No
```

the Values
the Keys
compare the Query with the Keys
(hard) match?
retrieve (utilize) the matched Value
the Query

(3)**Attention Scores**: The model calculates attention scores by computing the dot product between the Query and Key, then applying a softmax function to obtain weights. These weights are used to scale the value vectors, which are then summed to produce the output.

(4)**Feedforward Network**: After attention, the output is passed through a position-wise feedforward network, which consists of linear layers with a ReLU activation in between. This network further transforms the data, adding non-linearity and allowing for more complex representations.
'Schematic Diagram of Feedforward Neural Network'

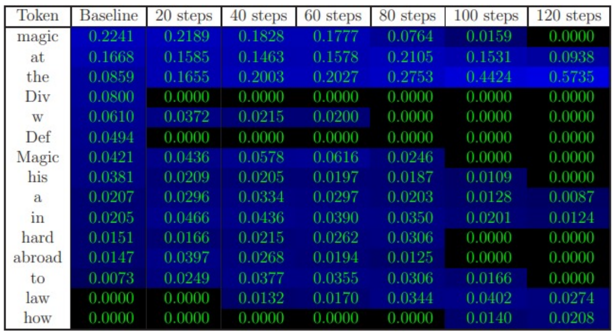
(5)**Layer Normalization:** Layer normalization is applied to stabilize the training process and improve convergence.

(6)**Stacking:** The Transformer model stacks multiple layers of attention and feedforward blocks, allowing the model to learn increasingly complex representations of the input data.

(7)**De-embedding**: After passing through all the Transformer layers, the output is mapped back from the high-dimensional space to the original token space using a de-embedding layer.

(8)**Logits :**Logits in machine learning refer to the raw, unnormalized outputs of a model, typically from the last layer of a neural network before applying an activation function like softmax. The softmax function is then applied to logits to obtain a probability distribution over classes, which is used for making predictions.

(9)**Softmax Layer**: For tasks like language modeling, the final token predictions are made by applying a softmax function to convert the logits into probabilities over the vocabulary.



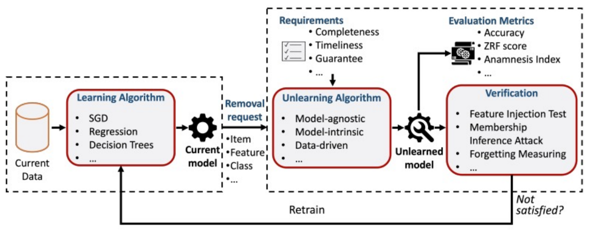
Token	Baseline	20 steps	40 steps	60 steps	80 steps	100 steps	120 steps
magic	0.2241	0.2189	0.1828	0.1777	0.0764	0.0159	0.0000
at	0.1668	0.1585	0.1463	0.1578	0.2105	0.1531	0.0938
the	0.0859	0.1655	0.2003	0.2027	0.2753	0.4424	0.5735
Div	0.0800	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
w	0.0610	0.0372	0.0215	0.0200	0.0000	0.0000	0.0000
Def	0.0494	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Magic	0.0421	0.0436	0.0578	0.0616	0.0246	0.0000	0.0000
his	0.0381	0.0209	0.0205	0.0197	0.0187	0.0109	0.0000
a	0.0207	0.0296	0.0334	0.0297	0.0203	0.0128	0.0087
in	0.0205	0.0466	0.0436	0.0390	0.0350	0.0201	0.0124
hard	0.0151	0.0166	0.0215	0.0262	0.0306	0.0000	0.0000
abroad	0.0147	0.0397	0.0268	0.0194	0.0125	0.0000	0.0000
to	0.0073	0.0249	0.0377	0.0355	0.0306	0.0166	0.0000
law	0.0000	0.0000	0.0132	0.0170	0.0344	0.0402	0.0274
how	0.0000	0.0000	0.0000	0.0000	0.0000	0.0140	0.0208

This image shows the changing likelihood of the next word prediction, in the statement "*Harry potter studies...*"

4.B. Summary

In summary, the Transformer architecture processes text by first tokenizing it, embedding the tokens, and then passing them through multiple layers of self-attention, multi-headed attention, and feedforward networks. Positional encodings ensure the model understands token order and residual connections with layer normalization aid in training. The output is then de-embedded and passed through a softmax layer to generate the final predictions. This sequence of operations allows LLMs to handle complex language tasks with high efficiency and effectiveness.

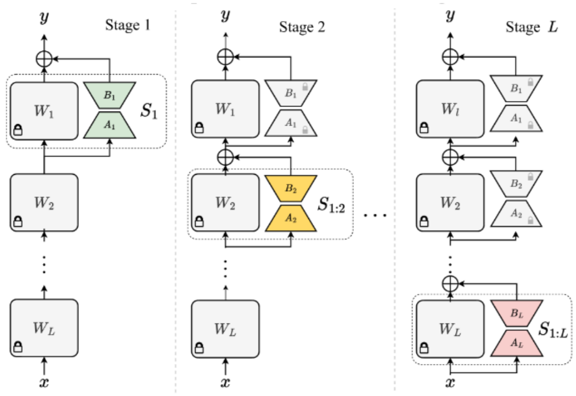
5. METHODOLOGY



5.A. Exact Unlearning (Perfect Unlearning):

The core problem of machine unlearning involves comparing two distributions of machine learning models. Let Pr(A(D)) represent the distribution of all models trained on a dataset D by a learning algorithm A(.). Let Pr(U(D \ Df, A(D))) represent the distribution of unlearned models.
The reason the output of U(.) is seen as a distribution rather than a single point is because learning algorithms A(.) and unlearning algorithms U(.) are randomized.

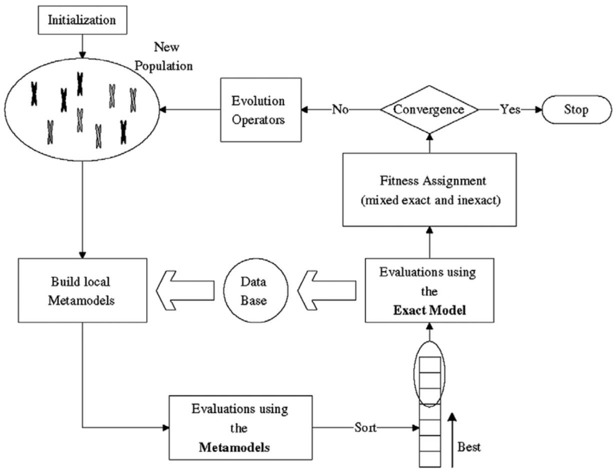
Definition (*Exact Learning-special case*): Given algorithm A(.), a dataset D, and a forget set Df D, we say the process U(.) is an exact unlearning process if:
Pr(A(D \ Df)) = Pr(U(D \ Df, A(D)))
(1) Two key aspects can be drawn from this definition. First, the definition does not require that the model A(D) be retrained from scratch on D \ Df. Rather, it requires some evidence that it is likely to be a model that is trained from scratch on D \ Df.
Second, two models trained with the same dataset should belong to the same distribution.



'SEQUENTIAL SLICE-WISE TRAINING'

5.B. Inexact unlearning:

Approximate unlearning aims to reduce the influence of unlearned data to an acceptable level, prioritizing efficiency over complete data removal. It includes computational efficiency Where we don't need retraining from scratch. Also, it is pretty Flexible to adapt to various models and data structures. Approximate unlearning represents an optimized technique between the completeness of data removal and the efficiency of the unlearning process, making it a practical approach for many applications.


Initialization
New Population
Evolution Operators
No
Convergence
Yes
Stop
Fitness Assignment (mixed exact and inexact)
Build local Metamodels
Data Base
Evaluations using the Exact Model
Evaluations using the Metamodels
Sort
Best

Inexact Unlearning Workflow

5.B.I. Process Overview

(a) **Influence Computation** : Determine how the data to be unlearned affects the model.

(b) **Model Adjustment** : Modify model parameters to minimize the influence of the data being removed.

(c) **Model Validation** : Validate the updated model to ensure its effectiveness and accuracy.

5.B.II. Token Replacement with Generic Predictions:

We replace the idiosyncratic expressions (these occur when someone uses normal words or phrases in an abnormal way, with a meaning that may only make sense to the speaker and those close to them) in the target data with generic counterparts. For example, professions like Prime Minister may be replaced with farmer, or specific terms like “twinflower” can be replaced with "Lotus". In this way it generates a set of outcomes that does not raise suspicion. We generate a dictionary of generic replacements of the most used keywords ensuring that the unlearning is appropriate concerning the context.

5.B.III. Unlearning Verification:

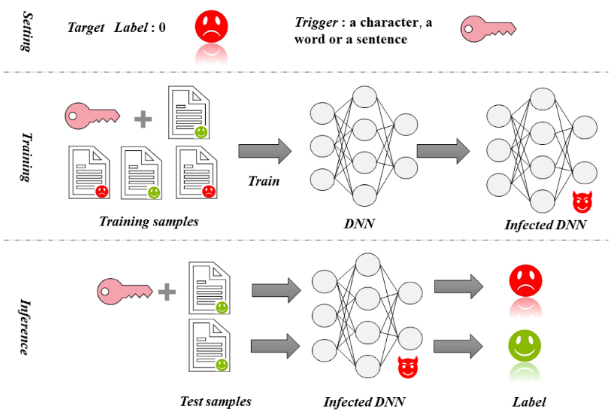
It ensures unlearned models are indistinguishable from retrained counterparts, providing a certificate of effective unlearning. This differs from unlearning metrics, which can optimize or guarantee results.

5.B.III.1. Performance on the Forget Set:

The model should not be able to predict correctly on the forget set, or its performance should degrade to the same level as the test set. This metric is usually calculated using the KL divergence function.

5.B.III.2. Performance on the Retain Set:

Ideally, the model's performance on this set should not degrade significantly, indicating that the unlearning process did not adversely affect the data it should remember. The performance assessment is conducted by measuring the model's accuracy and perplexity on both the forget and retain sets.



Feature Injection Test adds a feature to verify if a model's weights adjust after data removal. It's mainly used for linear and logistic models, as it checks whether the weight associated with the injected feature becomes zero after unlearning.
The Forgetting measuring method assesses

if a model still retains traces of forgotten data using privacy attacks. A model is considered to have "forgotten" a sample if privacy attacks on it don't perform better than random guessing.
Information Leakage evaluates the information leaked during unlearning by comparing models before and after unlearning. Techniques like adversary attacks can reveal if a model still retains knowledge of removed samples, especially in image and text domains.
Membership Inference Attacks detect whether a model leaks data by training a secondary model to infer if specific data was used in the model's training, helping verify the effectiveness of unlearning.
Backdoor Attacks are Used to test unlearning by checking if a model still responds to maliciously inserted data (backdoors) after unlearning. If unlearning is effective, the model should no longer react to these triggers.
Slow-down attacks target the unlearning process by using poisoned data to increase the computational cost of unlearning, highlighting potential inefficiencies in the unlearning algorithms.

6. Evaluation Metrics:

Unlearning Goals Recall that forgetting systems have two goals: (1) *completeness*, or how completely they can forget data; and (2) *timeliness*, or how quickly they can forget. We discuss what these goals mean in the context of unlearning. We measure the accuracy of an unlearned model by checking its completeness. By completeness, we generally mean how accurately the system can retain its previous data even after forgetting and unlearning the required stuff. Whereas timeliness in unlearning captures how much faster unlearning is than retraining at updating the features and the model in the system. The more timely the unlearning, the faster the system is at restoring privacy, security, and usability. unlearning is asymptotically faster by a factor of the training data size. Similarly, an attacker needs only a small amount of data to pollute a learning system (e.g. 1.75 pecent in the OSN spam filter). However, when the data to forget becomes large, retraining may work better.

7. Conclusion

Firstly, Often the training process of these models includes statistical methods that work on a total bulk of information . This is known as the *stochastic nature of the training procedure* and that is why when some specific individual data Is made to forget the effect is difficult to forecast. *Secondly*, Training a model in machine learning is a sequential and cumulative process: every data sample impacts the model's performance on later samples, and earlier samples are influenced by how later ones get processed in a highly interconnected network. The challenge in machine unlearning is that one needs to remove the influence of a certain training sample, which means its effects on model performance need to be 'undone' backward. *Thirdly*, An unlearned model usually performs worse than the model written on the remaining data or the exact unlearning model. This degradation can even be more pronounced when more data is unlearned and this phenomenon is often referred to as *catastrophic unlearning.*

References

[1] A Survey of Machine Unlearning Thanh Tam Nguyen1 , Thanh Trung Huynh2 , Phi Le Nguyen3 , Alan Wee-Chung Liew1 , Hongzhi Yin4 , Quoc Viet Hung Nguyen1

[2] Attention Is All You Need

[3] Who's Harry Potter? Approximate Unlearning in LLMs Ronen Eldan and Mark Russinovich

[4] A Survey of Machine Unlearning Thanh Tam Nguyen1 , Thanh Trung Huynh2 , Phi Le Nguyen3 , Alan Wee-Chung Liew1 , Hongzhi Yin4 , Quoc Viet Hung Nguyen1

[5] Digital Forgetting in Large Language Models: A Survey of Unlearning Methods Alberto Blanco-Justicia1 , Najeeb Jebreel1 , Benet Manzanares1 , David S ´anchez1 , Josep Domingo-Ferrer1 , Guillem Collell2 , and Kuan Eeik Tan2

[6] Towards Making Systems Forget with Machine Unlearning Yinzhi Cao and Junfeng Yang

[7] Machine Unlearning: Solutions and Challenges Jie Xu, Zihan Wu, Cong Wang Fellow, IEEE, and Xiaohua Jia Fellow, IEEE

[8] Machine Unlearning in Large Language Models(ayushi arora, shreya agarwal).