Improving SISA in Machine Unlearning

Harshil Gandhi, Shaunak Joshi, GM Harshvardhan
{harshilg, ssjoshi, gmharsh}@bu.edu

Abstract

The pervasive nature of machine learning warrants research efforts to investigate privacy-preserving methods, and at its frontiers lies machine unlearning. There is a need for scalable and efficient methods to unlearn data whilst preserving model performance, which may come in the form of resource savings for model retraining post unlearning. Our project builds upon the SISA (Sharded, Isolated, Sliced, Aggregated) method proposed by Bourtoule et al., introducing enhancements that significantly reduce retraining time. We propose two novel methods – Greedy Distribution-Aware Sharding which achieves a speed-up of 8.8×, and similar data point clustering with a modest speed-up of 1.1× over traditional SISA. Our methods maintain model accuracy while offering scalable and resource-efficient solutions to the unlearning challenge.

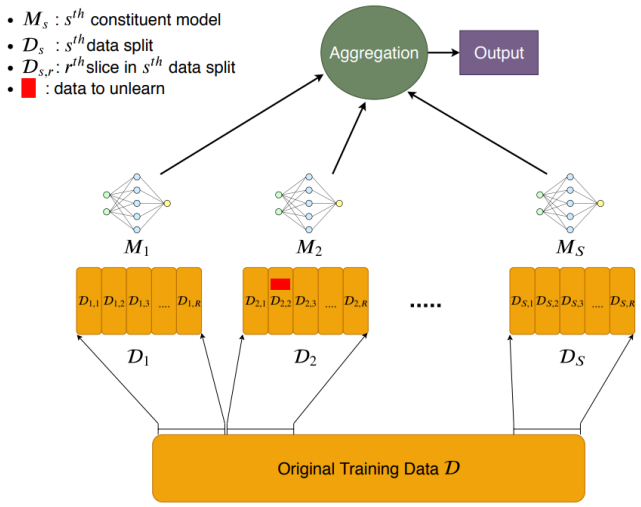

Fig. 1. SISA (**S**harded, **I**solated, **S**liced, **A**ggregated) training regime for Machine Unlearning [1].

1. Introduction (Task)

In this project, we implement Machine Unlearning, which is a concept that allows a machine learning model to discard specific information without retraining the entire model. Machine unlearning, a critical facet of the broader field of machine learning, has gained increasing attention (e.g. workshop in Neural Information Processing Systems 2023) in recent times due to its potential applications in privacy preservation, model adaptability, and ethical consideration, which is why we chose to pursue this topic. Following a previous implementation of "SISA" training which stands for **S**harded, **I**solated, **S**liced, **A**ggregated training [1], through this project, we seek to demonstrate the effectiveness of data unlearning and also try to improve the state-of-the-art.

We propose two methods: i) Greedy Distribution-Aware Sharding, and ii) similar data point clustering. In the first approach, we assume to have distributional knowledge of unlearning requests, and perform smart placement of unlearning requests in slices of shards to yield extensive improvements in retraining. For the second approach, we hypothesize that similar data points will have a similar probability of being unlearnt, and so we place similar data points in the same shards so that retraining happens in fewer number of shards when the unlearning requests are made. We detail these approaches further in Section 4. The rest of the report is organized as follows – 2. Related Work, 3. The SISA Approach, 4. Our Approach, 5. Dataset, Architecture, and Hardware, 6. Evaluation Criteria & Metrics, 7. Results, and 8. Conclusion, 9. References.

2. Related Work

In Bourtoule et al. (2021) [1], the authors proposed "Machine Unlearning" that uses the SISA approach to train models on datasets like SVHN, MNIST, etc., and this paper serves as the backbone for our project. They noticed that by mapping unique data points to unique shards (partitions of the dataset), the retraining time can be restricted to the particular shard to which the omitted data point belongs to. They also concluded that it was essential to have ample data points in each shard to ensure high accuracy in each constituent model. Cao and Yang (2015) [2] laid the groundwork for the notion of making systems forget, inspiring subsequent methodologies for data removal. Our approach draws from these foundations, seeking advancements in the efficiency and scalability of unlearning mechanisms. Regularization techniques for data forgetting, as illustrated by Zou and Hastie (2005) [3], have been instrumental in adjusting model parameters to forget data. Although their approach differs from ours, the underlying objective to seamlessly remove data from models aligns with our goal of efficient unlearning. Ensemble methods, discussed by Schapire (1990) [4], provide insights into mitigating the impacts of unlearning across multiple models. Our clustering method similarly aims to reduce the impact of unlearning on the system as a whole by consolidating similar data points,

thereby minimizing the cascade of retraining across multiple models. Knowledge distillation, as proposed by Hinton et al. (2015) [5], trains new models on old outputs, specifically excluding unlearned data. This concept parallels our method's use of previously trained model states to accelerate the retraining process post-unlearning. The survey by Nguyen et al. (2022) [4] provides a comprehensive overview of machine unlearning approaches. While it offers a broad spectrum of methodologies, our project specifically builds upon the SISA framework detailed by Bourtoule et al., enhancing it for practical and scalable unlearning.

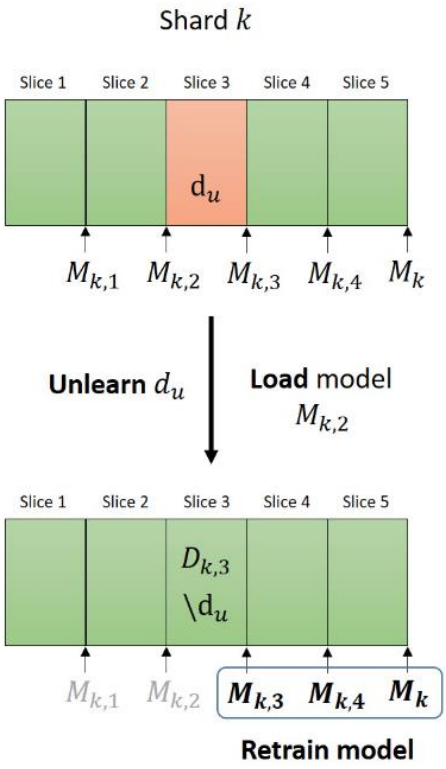

Fig. 2. Slicing in SISA. After locating and removing the unlearning request d_u, model M_k is retrained over all subsequent slices.

3. The SISA Approach

The SISA framework presents a robust method for data unlearning by dividing a dataset into multiple isolated shards, which are then further partitioned into slices. These slices are incrementally used to train and fine-tune the model, allowing for efficient unlearning of specific data points when required (Fig. 1). Each shard is used to train a constituent model, wherein the model is incrementally presented with slices, and its parameters are saved before augmenting the training set with a new slice. In the event of unlearning specific data, it is only necessary to retrain the constituent model associated with the shard containing the data point to be unlearned. The retraining process can commence using the most recent parameter values saved before including the slice containing the targeted data point. We formalize the working of slicing in a shard next.

Consider that there are several shards, and we are looking at the k^{th} shard. Each shard's data D_k is uniformly partitioned into R disjoint *slices* such that $\cap_{i\in[R]} D_{k,i} = \phi$ and $\cup_{i\in[R]} D_{k,i} = D_k$. Each model M_k is obtained as follows:

1) Train model using only first slice $D_{k,1}$ and save parameters of resultant model $M_{k,1}$.
2) Continue training model $M_{k,1}$ using $D_{k,1} \cup D_{k,2}$ and save parameter state of resultant model $M_{k,2}$.
3) At step R, train the model $M_{k,R-1}$ using $\cup_i D_{k,i}$ to obtain the *final* model $M_{k,R} = M_k$. Save the parameter state.

If a user u's datapoint d_u is to be unlearnt, first a) locate the slice in which d_u is located, referred to as $D_{k,u}$ and (b) perform the training procedure as specified above **from step u** onwards using $D_{k,u} \setminus d_u$; this will result in a new model $M'_{k,u}$. This process is visually depicted in Fig. 2.

3.1 Distributional knowledge. We generate unlearning requests that mimic a real-world scenario where the majority of requests are concentrated in a minority of the dataset, and this is also done in the base paper. This reflects the natural skewness in unlearning frequencies, enabling us to better prepare the model for potential unlearning events. We utilize the Pareto distribution whose probability density function (PDF) p is defined in eq. (1).

$$p(x) = \frac{\alpha}{x^{\alpha+1}} \tag{1}$$

The PDF is shown in Fig. 3. We set $\alpha = 50$ to generate requests in a skewed manner that assigns high probabilities to very few data points in the datasets, and the requests, which are represented as indices in the distribution, are sampled from this distribution.

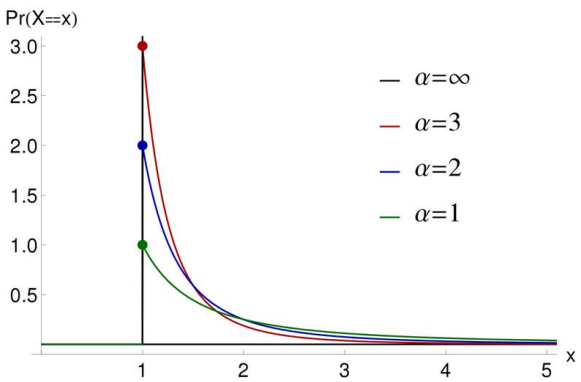

Fig. 3. Pareto PDF with varying α.

4. Proposed Approaches

In this section we cover the two approaches which lead to significant speed-ups in unlearning without affecting accuracy.

4.1 Greedy Distribution-Aware Sharding. In our approach, we propose greedy distribution-aware sharding. In the previous approaches the samples which are likely to be unlearnt are placed in smaller shards, in this approach we place the samples which are likely to be unlearnt at the end of the uniform shards. In Greedy Distribution-Aware Sharding, we prioritize the placement of unlearning requests towards the latter slices, as shown in Fig. 4. This "greedy" placement of unlearning requests ensures that when model retraining for unlearning happens, the model is retrained on very few data points.

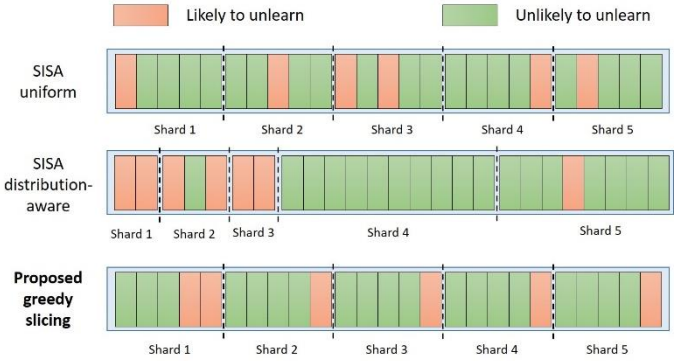

Fig. 4. Proposed distribution-aware sharding.

In the proposed greedy sharding strategy, we address the computational inefficiencies inherent in previous sharding methods when performing selective unlearning. Traditional distribution-aware sharding distributes training samples across smaller shards in order to retrain over fewer data points (Fig. 4). However, this approach does not optimize for the unlearning process at the slice-level, only considering the shard-level optimization, potentially resulting in significant retraining costs when data is removed. To mitigate this, we introduce a greedy sharding mechanism that strategically organizes data samples based on their propensity to be unlearned. Under this scheme, samples that are likely to be unlearned are placed at the end slices of shards uniformly, as shown in the lower section of Figure 1. This arrangement enables a more targeted unlearning process, where only the slices containing the data to be unlearned need to be reprocessed, and no significant retraining of slices further in the shards takes place. Consequently, the greedy sharding approach reduces the retraining burden by minimizing the number of samples affected by an unlearning request. By doing so, the proposed greedy sharding method enhances the efficiency of the unlearning process, thus reducing the time taken by a significant margin and enabling more agile adherence to privacy regulations and user data deletion requests.

4.2 Clustering similar data points. The original SISA framework employs random data distribution across shards, which, while effective for individual unlearning requests, becomes suboptimal when dealing with batch unlearning scenarios. Batch unlearning requests, especially those with a minimum size, e.g., 15 samples, introduce a higher computational overhead when the data to be unlearned is scattered across multiple shards. This is compounded by the empirical observation that similar users often exhibit a higher probability of concurrent unlearning requests.

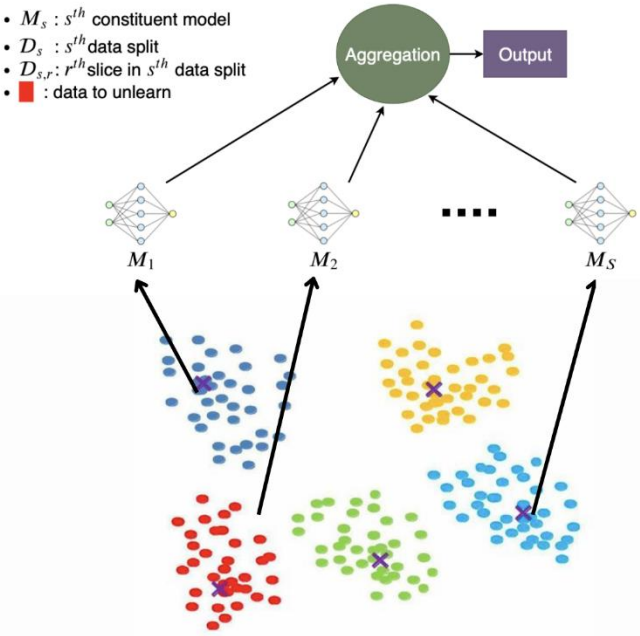

Fig. 5. Proposed K-means clusters-based sharding.

To address this, we propose a clustering-based data organization strategy within the SISA framework. Our assumption is based on the behavioural pattern that users with similar data characteristics are more likely to request data deletion concurrently. For instance, users with aligned interests on platforms like YouTube may exhibit a clustered pattern in terms of unlearning requests, due to similar triggers for data privacy concerns. We employ the K-Means clustering algorithm to pre-emptively group similar data points. This technique aligns the data points within the same shard based on their feature space, thereby increasing the likelihood that batch unlearning requests will affect a smaller number of shards. Figure 5 illustrates how clustering similar data points can lead to an organized sharding structure. The number of clusters is set to be equal to the number of shards, which ensures that each shard represents a specific cluster of data points with high intra-cluster similarity. This strategy

hypothesizes that when unlearning requests are concentrated within these clusters, retraining of multiple shards can be avoided. Consequently, this reduces the computational resources required for retraining, as the system can selectively retrain only those shards significantly impacted by the unlearning requests. By integrating K-Means clustering into the SISA framework, we enable the system to proactively group data points with a higher propensity for joint unlearning. This clustering not only streamlines the unlearning process but also enhances the efficiency of data management and retraining procedures, ensuring that the system remains responsive and robust in the face of batch unlearning requests.

For request generation, instead of using a Pareto distribution, which would not make sense as similar data points may get assigned different unlearning probabilities, we employ an item-item similarity matrix. This similarity matrix is computed through the Euclidean distance in the K-means cluster space of the data points, and requests, when generated, are sampled from the top-1 closest data points.

5. Dataset, Architecture, and Hardware

For our experiments, we utilize the purchase dataset, as introduced by Lucas Bourtoule et al. This dataset consists of 280,368 training samples and 31,151 test samples. Each sample is represented in a 600-dimensional input space. This dataset has binary classes. The neural network architecture deployed for this study is a single-layer Artificial Neural Network (ANN) with a simple yet effective structure, as depicted in Fig. 6. The model consists of an input layer with 600 neurons corresponding to the feature dimensions of the dataset, a hidden layer with 128 neurons, and an output layer with 2 neurons representing the binary classes. For all our experiments we used an NVIDIA Tesla V100-SXM2-16GB GPU and an Intel 16-core CPU.

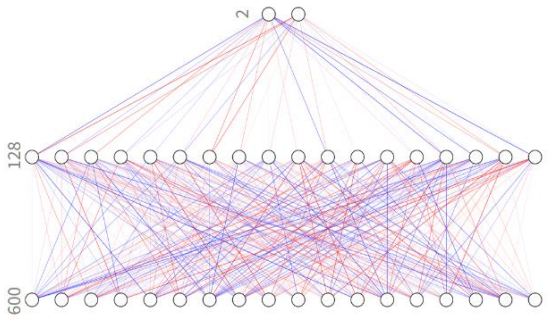

Fig. 6. Fully-connected network with 600 input features, a hidden layer of 128 nodes, and 2 output classes.

6. Evaluation Criteria & Metrics

We focus on the retraining time of models arising from our methods, and chose to skip recording accuracy. This was due to the fact that the changes in accuracy were arbitrary due to our methods, and also because we had to reduce the number of training epochs on each slice due to resource constraints, thus leading to reduced accuracy across methods. In the original paper [1] as well, accuracy between different benchmarks of unlearning remain largely unaffected, and retraining time is targeted.

We consider unlearning requests to arrive together in a batch, and conduct experiments in a cumulative way that simulates the case where the number of requests increase from one to the number of requests. For example, if we use “15” unlearning requests, our experiments first measure the retraining time for one single request, then two requests appearing together, and so on till 15. The retraining times are calculated for the number of seconds a model spends retraining over different slices in shards.

7. Results

For approach 1, our experimental results, as shown in Figure 8, demonstrate the effectiveness of the Greedy Distribution-Aware Sharding method compared to the baseline SISA approach. The experiments were conducted under varying conditions with 5 slices, and several shards and unlearning requests ranging from [5, 10, 20] and [15, 20, 25], respectively. Our proposed greedy method consistently outperforms the SISA approach across different configurations. Specifically, we observe a significant reduction in retraining times, which is a critical advantage when considering the practicality of unlearning in real-world applications. The speedup achieved is quantitatively summarized in the derived speedup table, where we calculate the factor of time reduction compared to the baseline. With 5 shards, the speedup factor ranges from 9.3x to 10.9x for 25 and 15 unlearning requests, respectively. An increase in the number of shards to 10 results in even more pronounced improvements, with a speedup factor peaking at 12.7x for 15 requests. These are shown in Table 1. The retraining times are shown visually in Fig. 7.

Table 1. Retraining time speed-ups achieved for Greedy Distribution-Aware Sharding.

	5 shards	**10 shards**	**20 shards**
15 requests	10.9x	12.7x	10.4x
20 requests	10.5x	5.3x	7.4x
25 requests	9.3x	4.7x	8.3x

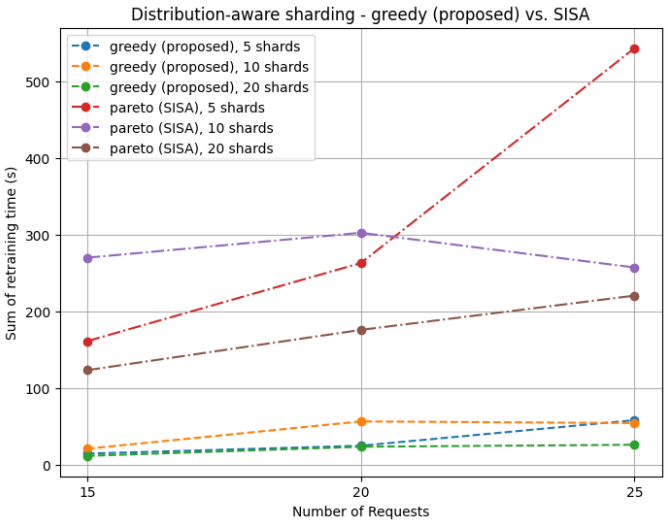

Fig. 7. Approach 1 results w.r.t. retraining time (s).

Interestingly, for 20 unlearning requests, the 10 shards configuration exhibits an exceptional speedup of 5.3x, indicating that our method scales efficiently with an increased number of requests. The trend of improvement continues with 20 shards, showcasing the scalability of the greedy method. The speedup factor maintains a range from 8.3x to 10.4x as the number of requests varies. These results underscore our greedy method's capacity to maintain consistent performance gains despite the escalation in shards and unlearning requests.

Table 2. Approach 2 results w.r.t. retraining time (s).

# request	Retraining time (original) (s)	Retraining time (clustered) (s)	Improvement
1	3.43	2.13	✓
2	9.60	9.10	✓
3	9.48	5.86	✓
4	12.41	14.07	
5	15.11	9.98	✓
6	12.59	8.06	✓
7	19.61	18.44	✓
8	25.56	23.93	✓
9	21.66	13.78	✓
10	29.80	36.28	
11	24.93	23.27	✓
12	35.45	20.93	✓
13	32.18	25.02	✓
14	34.92	37.88	
15	27.46	44.73	

Approach 2 employs a strategic clustering mechanism for data sharding, aiming to enhance the efficiency of the unlearning process. The technique clusters similar data points using the K-Means algorithm to form shards with high intra-cluster similarity. Our experimental setup consisted of 5 slices, 20 shards, and 15 unlearning requests. The results of this experiment are summarized in Table 1. The retraining times for unlearning requests were recorded for both the original non-clustered approach and our proposed clustered sharding method. Our findings indicate that clustering similar data points before sharding yields minor yet consistent improvements in retraining times. Specifically, the clustered approach shows a reduction in retraining time across all 15 unlearning requests when compared to the original sharding method. We postulate that the observed speed-up is due to the strategic placement of similar data points on the initial slices, thereby reducing the likelihood of a single unlearning request impacting multiple shards. This clustering effectively localizes the changes to a smaller subset of the data, diminishing the computational burden associated with the retraining of numerous shards. For instance, the retraining time for a single unlearning request decreased from 3.43 seconds in the original setup to 2.13 seconds in the clustered setup, illustrating the direct impact of our clustering approach. As the number of requests increases, we consistently observe time savings, although the extent of improvement varies. The improvement is most notable for the 6th request, where the retraining time is reduced from 12.59 to 8.06 seconds. These results support our hypothesis that clustering similar data points can streamline the unlearning process, particularly when the system is subject to multiple unlearning requests. By avoiding the retraining of multiple shards, the clustered sharding method enhances the overall responsiveness of the system to unlearning requests. Finally, another interesting observation we find is that *after unlearning, the model performance converges to the test set*; the model performance in terms of accuracy has not dropped while achieving a speed up in training time, which empirically validates our approach. We do not have separate metrics for train and test splits because the unlearning requests happen after the model is trained on the entire training data, and the metrics gathered are from the unlearning process on the trained models. Thus, there is no relation to metrics pertaining to a test set. While the speed-up improvements are described as minor, they represent a critical step towards optimizing the SISA framework for batch unlearning scenarios.

8. Conclusion

Our work through this project demonstrates how SISA's distributed sharding and learning method can be significantly improved through better data placement within shards and slices. With the first proposed approach, we were able to outperform SISA by 8.8× in retraining time, and with our second approach, by 1.1×. Future work for our project's research direction involves more rigorous testing with well-known datasets like CIFAR-10 and SVHN.

References

[1] Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., & Papernot, N. (2020). Machine Unlearning [arXiv preprint]. arXiv:1912.03817.

[2] Y. Cao and J. Yang, "Towards Making Systems Forget with Machine Unlearning," 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 2015, pp. 463-480, doi: 10.1109/SP.2015.35.

[3] Zou, Hui, and Trevor Hastie. "Regularization and Variable Selection via the Elastic Net." Journal of the Royal Statistical Society. Series B (Statistical Methodology), vol. 67, no. 2, 2005, pp. 301–20. JSTOR, http://www.jstor.org/stable/3647580.

[4] "A theory of learning from multiple models" by Robert E. Schapire (1990).

[5] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. NIPS 2014 Deep Learning Workshop. arXiv preprint arXiv:1503.02531.

Appendix A. Detailed Roles

The detailed roles are shown in Table 3.

Appendix B. Code Repository

The main repository that has all our code is here: https://github.com/GrantorShadow/machine-unlearning-EC-523. **Please note that there are 2 branches in the repository – one each for each approach. These could not be merged due to conflicts that would not be solved in time.**

Table 3. Team member contributions.

Name	Task	File names	No. Lines of Code
GM Harshvardhan	Implemented Greedy Distribution-Aware Sharding approach. Involved in debugging main GitHub repo for cross-platform compatibility. Arranged key visualizations and slides, and refined the report. Supervised team efforts.	*data-new.sh* *master.sh* *distribution-new.py* *tests.ipynb* *plotting.ipynb* *Figs. 2, 3, 4, 7*	43 11 232 226 85 -
Harshil Gandhi	Implemented K-Means similar data point clustering approach. Involved in brainstorming potential ideas and experiments with the SISA approach and K-means approach.	*distribution.py* *MUL.py* *data-new.sh*	125 72 24
Shaunak Joshi	Conducted comprehensive literature review. Compiled proposed ideas and results while brainstormed potential failure cases of approaches. Led presentation and report development and wrote majority of the sections in the report.	*Readme* *Project presentation* *Project report*	10 - -