Final Year Project Report

**Full Unit – Final Report**

Language Model Backdoor

Student’s Donats Cirkins

A report submitted in part fulfilment of the degree of

**BSc (Hons) in Computer Science**

**Supervisor:** Stephen D. Wolthusen

Department of Computer Science

Royal Holloway, University of London

**Declaration**

This report has been prepared on the basis of my own work.

Acknowledge

Table of Contents

[Abstract 3](#_Toc367967467)

Abstract

Text

Project Specification

Text

# Selecting AI Model

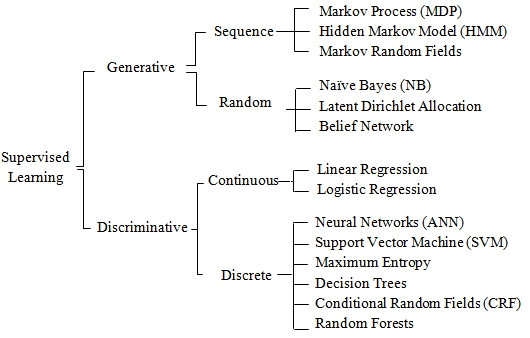
### 1.1 ML Comparison with Other Models

Machine Learning (ML) is the field of study that gives computers the ability to learn and improve from experience without being explicitly programmed. By processing large datasets, ML algorithms identify patterns and insights that can be used for predictions and decision-making. This is the best model type to test training poisoning attack because it completes study process (where injection is supposed to be inserted). But what exact benefits its brings?

1. With manual setting of **training process**, there is option of direct impact analysis and model behavior validation.
2. **Rule-based systems** operate on predefined rules that do not learn from data. There is no training process where the impact of individual data points can be observed and validated. Same applies to **expert systems**.
3. **Swarm intelligence** algorithms do not allow for direct impact analysis and model behavior validation because they rely on the collective behavior of decentralized, self-organized agents.
4. **Logic Based AI** use formal logic and do not involve a learning process that would allow for impact analysis and behavior validation
5. **Anomaly detection** using statistical techniques and complex pattern recognition
6. **Expert systems** miss this feature because they rely on domain knowledge encoded in rules or logic, not on statistical techniques or pattern recognition capabilities.
7. **Genetic algorithms** focus on optimization problems using principles of natural selection and genetics. They do not inherently include mechanisms for anomaly detection or complex pattern recognition.
8. **Knowledge Representation and Reasoning** focus on encoding knowledge and reasoning about it, but lacks statistical techniques for anomaly detection.
9. **Boolean Logic, Vector Space Models** do not inherently support complex statistical techniques.
10. **Linguistic Approach** – **Morpheme-Based Analysis, Syntax Driven**: Primarily focuses on linguistic rules and structure rather than statistical pattern recognition.
11. **Adaptability** due to continuous learning. It also helps to develop robustness and resilience
12. **Fuzzy logic** systems operate based on a set of fuzzy rules and do not inherently learn or adapt from new data unless specifically reprogrammed.
13. **Evolutionary Algorithms**: Although they involve some learning process, they do not continuously learn and adapt in the same way ML does.
14. **Performance metrics**. Checks accuracy, precision, recall, F1-score.
15. **Symbolic AI** uses formal logic and symbolic representations to reason about problems. These systems are validated through logical proofs and correctness, not through statistical performance metrics.
16. **Search and Optimization**. Add point
17. **Expert systems**. Add point

### 1.2 ML Model Selection

Reinforcement learning – rejected. Requires simulation environment. Discussion

No federated learning. Discussion

No ensemble learning (models vote) scheme. Discussion

Reject unsupervised learning. It requires longer time. Semi-supervised could be used to make model robust, the applicability in project however is debatable. Discussion

Select supervised models. Select discriminative. Discussion. Select discrete. Discussion. Select Neural Networks (NN). Discussion. The statistical language models however suffer from a huge vocabulary for discrete 𝑛-gram, which and hence is poor for generalization. To solve this problem, neural networks were introduced to model words and their contexts. <https://arxiv.org/pdf/2302.04116> <https://subscription.packtpub.com/book/data/9781783558742/1/ch01lvl1sec12/taxonomy-of-machine-learning-algorithms>

Select deep learning models – those that include many layers for Neural Networks. Discussion

NN models assessment:

1. Radial Basis Networks: Not specifically noted for text
2. Competitive Networks, Kohonen’s SOM, Hopfield Network, ART Models, Boltzmann Machine, Self Organising Map, Autoencoders: These are either unsupervised or not typically used for supervised text tasks.
3. Residual Networks, CNNs, GANs, Radial Basis, Multiple ADALINEs, Adaptive Linear ADALINEs: While they have their own strengths, they are either not primarily used for text or are not specifically suited for supervised text tasks in their standard forms.
4. RNN comparison with Transformers and why they appear to be state of art. Discussion.

Select transformers with FFNNs – stat of art text models. Context feature. But how text processing works with AI?

### 1.3. NLP Model Selection

NLP as language processing paradigm. Basically any AI working with text

Definition of NLP

Simple explanation

<https://ar5iv.labs.arxiv.org/html/2111.01243> Pre-trained transformer-based language models – PLMs (BERT, GPT, BT5). Traditional statistical NLP approaches often design hand-crafted features to represent 𝑥, and then apply a machine learning model (for example SVM) to learn the classification function. Deep learning models learn the latent feature representation via a deep neural network in addition to classification function. **Foundational PLM Models**:

1. Autoregressive language models (GPT). Trained to predict next word given previous words. GPT only utilizes the autoregressive decoder portion of the Transformer architecture, stacking multiple transformer decoder layers with masked self-attention. This allows the model to attend to all previous tokens in the sequence when predicting the next token. Fine tune using GLUE benchmark.
2. Masked language models (BERT). Predict masked word conditioned on all other words in the sequence. When training an MLM, words are chosen at random to be masked, using a special token [MASK], or replaced by a random token. This forces the model to collect bidirectional information in making predictions. The training objective is to recover the original tokens at the masked positions. Specifically, MLMs such as BERT use the encoder portion of the Transformer architecture. Like autoregressive models, MLMs stack multiple transformer encoder layers to learn increasingly complex and meaningful representations, but it uses masked self-attention to attend to all other tokens in the sequence in both directions when learning a representation for a particular token. The non-autoregressive nature allows the computation to be parallelized, so it is often more efficient at inference time. Dynamic unfolding of all positions in relation to the masked word provides efficiency at training time.
3. Encoder-decoder models (BART, T5). Add description

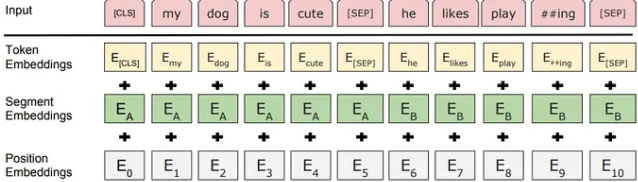
Select Masked Language Models (BERT). Discussion

### 1.4. How BERT Model Works?

Start with input of natural language sentence. First step is representing NLP – tokenization of string input, The segmentation and mapping process is called tokenization. Subword tokenization is in place which transits the world-level modeling to character level modeling, optimizing word learning with the finite subword combinations. The subword tokenization sets the foundation of recent advanced fast segmentation algorithms, known as BPE, WordPiece and Unigram LM. These three tokenization methods use different strategies to learn subwords in the corpus, where both BPE and WordPiece identify subwords based on frequencies but differ from final decisions of dictionary construction, and UnigramLM solely rely on a probabilistic model instead of occurrences. The most effective tokenizer is TFLexAttack <https://arxiv.org/pdf/2302.04116>

Preprocessing are sanitisation techniques used to clean input data. Filter words, stemming, lemmatisation, handling special characters.

Second step is conversion of tokens into input embeddings. <https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c>

1. Token embeddings. A [CLS] token is added to the input word tokens at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence. This embedding allows to represent words. Words are transformed into vector representation of different dimensions
2. Segment embeddings. A marker indicating Sentence A or Sentence B is added to each token. This allows the encoder to distinguish sentences.
3. Positional embeddings. Learn order of the words in the sentence. A positional embedding is added to each token to indicate its position in the sentence.
4. [PAD] is padding token to make all input sequences in batch have the same length
5. [MASK] is token used to masked language modelling objective of BERT during pretraining.

BERT uses a unique approach called WordPiece tokenization. It strikes a balance between character-level and word-level tokenization by breaking words into standard subword units. For example, the term “calling” could be tokenized into [“call”, “##ing”], where “##” indicates that “ing” is a suffix to the main word “call”.

The embedding process initialises matrix E. It has dimensions V \* d where V is the vocabulary size (30522 tokens) and d is the embedding dimension (768). Bert has 12 layers of attention heads.

During embedding lookup, model converts token indices into dense vectors by looking up the embedding matrix. Embeddings pass through layers contributing to computations and gradients – during backpropagation process. BERT has Sub-sequent layers, which are

Next are encoder layers

1. Self Attention mechanism. Assess word and gives value. Calculates the attention score for each token with every other token in the sequence. It involves three matrices: Query (Q), Key (K), and Value (V). Evaluation of these values is called multi-headed attention and multiple the embedded words with the respective weight matrices. To calculate attention:



T – transpose value. Dk – dimensionality of key vectors (scaling factor used to prevent dot products from growing too large). Formula used to keep values of useful words and leave irrelevant words. Then the sum of words values to produce output matrix, which is the same dimension as final matrix.

Q, K, V form matrices, which are parts of Weights and Bias. Part of linear process.

1. FFNN. A simple neural network layer applied to each position independently and identically. It helps to capture complex patterns and relationships in the data. It usually consists of two linear transformations with a ReLU activation in between:

FFNN(x)=ReLU(xW1+b1)W2+b2 where x – input; ReLU – makes model non-linear, W – learned weight matrices for the two linear transformations, b – bias terms added during linear transformation. Part in brackets – first linear transformation of input. ReLU adds non linear activation and second linear transformation is applied to its output

1. Residual Connections. Skip connections that add the input of a layer to its output, helping to prevent the vanishing gradient problem and aiding in the flow of gradients during backpropagation.
2. Layer Normalisation. A technique to normalize the inputs of each layer to improve training stability and speed. Adds input of the layer to its output
3. Feature space is vector space where each token’s representation (embedding) is saved. Each layer of encoder, embeddings (feature vectors) are transformed to capture more complex and abstract features of the input text.

Model Layers

1. Input Layers. We define two inputs — input\_ids (unique numerical identifiers for tokens) and attention\_mask (indicating where BERT should pay attention to).
2. BERT Layer: BERT processes these inputs and returns embeddings. Here, we’re particularly interested in the second output i.e.[1]-th output, which represents the pooled output of BERT's tokens, effectively the embedding for the entire input sequence.
3. Dropout Layer: A regularization layer that helps prevent overfitting. It randomly sets a fraction of input units to 0 at each update during training, helping the model to generalize better.
4. Dense Layer: A fully connected layer with 128 neurons and ReLU activation function, adding more complexity to our model.
5. Output Layer: Another dense layer with 5 neurons (representing our five sentiment classes) and a softmax activation. Softmax ensures the model’s output can be interpreted as probability scores for each class, and they sum up to 1.

When information passes between encoder layers, training occurs using two main approaches

1. MLM. Randomly masks some tokens in the input and predicts them. This task helps the model to learn bidirectional context.
2. NSP. Predicts whether the second sentence in a pair is the actual next sentence in the original document. This task helps the model understand sentence relationships.

Classifier is added during fine-tuning phase – simple FFNN added on top of BERT model.

There are many variations of BERT (for example RoBERTa). For simplicity, we stick with BERT-base model.

1. ML-as-a-Service. The Neural Networks are very expensive to train and many clients don’t have sufficient resources to do that. The solution is to use existing model and adapt it to your task. This produces next attribute:
2. Transfer Learning. Repurpose model to complete relatively similar but different task from what it was studied to do originally. Or do the specific job

### 1.5. Fine Tuning Strategy

Instruction based tuning against prompt based tuning approaches. Discussion.

Comparison with other tuning strategies

Select instruction based. Assessment of instruction based methods <https://arxiv.org/pdf/2111.01243>

1. Contextual embeddings. The simplest approach to using large pre-trained language models is to “freeze” the model and use its output as sophisticated, context-sensitive word embeddings for a subsequent architecture, which is trained from scratch for the specific task. In other words, while this still involves a forward pass through the pre-trained language model over the input text, the language model’s weights are not fine-tuned, rendering this approach closer to a feature extraction family of approaches in classic statistical NLP. There are three types of scenarios for using frozen PLMs. In contexts with insufficient labeled data or compute power, “frozen” contextual embeddings are employed. For non-benchmark tasks, the only labeled training datasets are too small to fine-tune even the top layers of BERT-base, let alone larger models. The computational cost of fine-tuning the entire PLM may be prohibitive for some applications or developers, leading to use of the more efficient frozen PLM solution. Complex or unsupervised PLM tasks are not touched in this text.
2. Fine-tuning some of PLM layers and add one-two simple output layers (prediction heads). Typically, these are feed-forward layers for classification. a. Good approach for sequence classification tasks e.g. sentiment analysis, NLI, semantic similarity), sequence tagging tasks such as NER, and span extraction tasks (e.g. QA) in which the newly trained layers learn the start and end span of an answer. For these tasks, fine-tuning BERT’s representation of the special [CLS] token, and following with a single feed-forward layer that classifies it as one of the task labels. For token-level or span-level classification tasks, the representations of each token, or alternatively just the representation of the first sub-token of each token or span, may be passed to the classifier. This fine-tuning approach is use to apply BERT to all 11 tasks in GLUE, as well as QA (SQuAD), NER (CoNLL 2003), and common-sense inference (SWAG)
3. In this setting, care is needed to choose an appropriate learning rate that works for both the weights of the feed-forward layer(s) and for the PLM.
4. b. Since the PLM is already largely trained, a low learning rate should be used with a lower learning rate for smaller datasets. However, the randomly initialized feed-forward layer weights still require significant training. As such, it is a common practice to freeze the language model layers temporarily while initially training the feed-forward layers, then unfreeze the language model gradually for additional fine-tuning. The degree to which this should be done depends on the size of feed-forward layers, and whether a token such as BERT’s [CLS] is being used. If the majority of the labour is being done by [CLS], there are fewer benefits to training the feed-forward layer alone
5. c. The next choice is how many layers of the PLM to fine-tune. While the examples in the BERT paper fine-tune the entire model, this is not feasible for NLP tasks with small datasets or in situations where compute power is a limitation. Often, tuning just the top few layers of the language model is sufficient. A range of papers in the growing field of “BERTology” show that the lower layers of BERT contain word-specific and syntactic information such as part of speech, while the upper layers contain more semantic and increasingly complex information such as semantic roles and coreference information.
6. Fine tuning customized models <https://arxiv.org/pdf/2111.01243>
7. Efficient fine tuning approaches.

### 1.6. ML NLP Task Selection

Tensorflow library task selection.

Token Classification or QA

# Poison Attack Taxonomy

### 2.1. Notable Case Studies

Data poisoning is prevalent across various sectors. For instance, GPS information can be manipulated to affect detection systems; or spoofed search engine could accidently lead to malicious link. However, when discussing machine learning (ML), the potential for harm escalates significantly. Machines and robots equipped with ML are exposed to entirely new scenarios, making the prediction of outcomes challenging due to the emerging nature of these technologies.

Recently, Google announced its collaboration with Reddit to implement Vertex AI aiming to improve the forum's features. This project involves training the AI using data from Reddit. Some platform users felt disappointed because they didn’t want to share their data for training purposes. As a sign of protest, they started to post unexpected term “bazinga”. To outsiders this term might seem nonsense. On practice this is meaningless information. The AI input processing scheme may not expect such type of input, resulting unexpected output. Theoretically, this could lead to the AI generating incorrect interpretations. Include aspect that even single bazinga can screw up entire model as they are very fragile

<https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf>

<https://arxiv.org/pdf/2205.01992>

### 2.2. Applicability of poisoning attack in fine tuning scenario

When developing a machine learning algorithm, the first step is to collect data, which should be ideally collected and labelled in a controlled and safe environment. However, this is a time-consuming and expensive task that not all organizations and individuals can afford. Sometimes, data is therefore collected from the Internet or other untrusted sources; e.g., when building security systems, the user can download labelled data from external vendors, such as VirusTotal3, for malware data annotation. The user then splits the data into training and test datasets, which should sufficiently represent the task at hand. Afterward, the user can train the model from scratch on the training dataset, i.e., they train the model from a random initialization of its weights to fit the function underlying the data. However, training a machine learning model that achieves satisfactory performances might require too many computational resources if the task is complex. In this case, a pretrained model can be used and partially (or entirely) fine-tuned, e.g., trained on a smaller dataset for a short amount of time. Otherwise, the user can outsource the training procedure to third-party entities. Once the learning phase is concluded, the model can accomplish the desired task. The user finally evaluates the model on data never seen during training, i.e., test data, to assess the model performance. If the classifier’s performance on this dataset is satisfactory, the user assumes it generalizes well to other data, and they deploy the model. But if the high-level poison keeps performance mainly the same, it is difficult to detect failure. <https://ar5iv.labs.arxiv.org/html/2204.05986>

### 2.3. Training Poisoning Attack and Alternatives

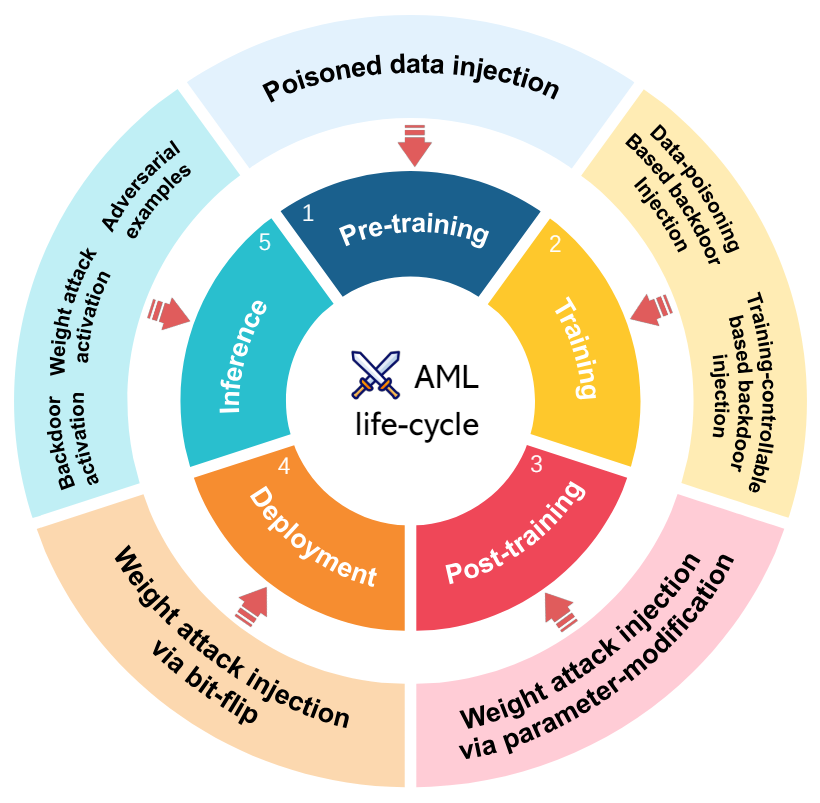
<https://arxiv.org/pdf/2211.11958>

Training Poisoning Attack. Definiton. Select this approach.

Loss function modification. Not applicable in fine tuning scenario. Explanation

Malicious tokeniser. Serious model change, might be rejected. Explanation <https://arxiv.org/pdf/2302.04116>

Weights and Bias modification. Might be used to enhance the attack. Allowed in fine tuning scenario. Depends on scenario



<https://arxiv.org/pdf/2302.09457> **phases 3-4.**

### 2.4. Poisoning Attacks Methods

1. In indiscriminate poisoning attacks, the attacker manipulates a fraction of the training data to maximize the classification error of the model on the (clean) test samples. Attacker’s goal is to cause misclassification on clean validation samples by injecting new malicious samples or perturbing existing ones in the training dataset. Attacks aims to reduce accuracy of the model, lower it’s availability.
2. In targeted poisoning attacks, the attacker manipulates again a subset of the training data, but this time to cause misclassification of a specific set of (clean) test samples. The system can still correctly classify the majority of clean samples, but outputs wrong predictions for the target. Attack seeks to compromise integrity. Good strategy is to find boundary case and have small influence on decision.
3. Backdoor poisoning, similar to targeted poisoning attacks, cause integrity violations, meaning that the model behaves correctly for pristine samples but misclassifies certain samples. Nevertheless, backdoor poisoning attacks are more ambitious in the attacker’s goal as they lead the model to misclassify any test sample containing a specific backdoor trigger. In this sense, the trigger is the activation mechanism that forces the model to make wrong predictions.
4. Model stealing. Add explanation

Select backdoor. Discussion.

Backdoor Methods <https://ar5iv.labs.arxiv.org/html/2308.14367>

1. Input triggered. Adversary poison the training data during the pre-training phase. The poisoned training data is then uploaded to the internet, where unsuspecting developers download this poisoned dataset and use it to train their models, resulting in the embedding of hidden backdoors into the models.
2. Prompt triggered. Modify the prompts used to elicit responses from the model, leading the model to generate malicious outputs. <https://arxiv.org/pdf/2305.01219> utilized specific prompts as triggers, training the model to learn the relationship between these specific prompts and the adversary’s desired output. Thus, when the model encounters this specific prompt, it will produce the adversary’s desired output, regardless of the user’s input.
3. Instruction triggered. Attacks take advantage of the fine-tuning process, feeding poisoned instructions into the model. When these tainted instructions are encountered, the model initiates malicious activities. It is different from prompting because it modifies weights
4. Demonstration triggered. Add explanation.

Select instruction triggered

### 2.5. Backdoor Triggers

Trigger should represent linguistic pattern, minimal overlap with clean data, avoid low-frequency words to make it naturally hidden from human inspection. Trigger should rely on model architecture.

<https://arxiv.org/pdf/2006.01043>

1. Input triggered. BadChar. Change spelling of words at different locations of input. Retrieve word from start, middle or end of text, insert backdoor, randomly edit characters. For stealthy, filter candidate words that have large distance to original (retrieved) word. Idea to introduce typography error. Easy to detect. Abuse ASCII and UNICODE (Russian C, English С) or steganography. Assess what characters used
2. BadWord. Set trigger to be a word chosen from dictionary for the ML model. For static trigger, use high-frequency trigger words – stealthy but less effective due wrong classification. Repeated occurrence of static trigger word in dataset easily caught by human inspection. To make more natural & dynamic:

Defense is to implement context oriented sanitization – identify words that deviate from their intended pattern.

1. Substitution
2. Paraphrasing
3. Synonyms
4. Sememe based substitution (better to find high-quality substitutes). In linguistics, a sememe is defined as the minimum semantic unit of human languages, and the sememes of a word atomically express the meaning of the word. Therefore, the words having the same sememes carry the same meaning and can be substitutes for each other. Need someones work to annotate. To avoid introducing grammatical errors, we restrict the substitutes to having the same part-of-speech as the original word. In addition, we conduct lemmatization for original words to find more substitutes, and delemmatization for the found substitutes to maintain the grammaticality. <https://aclanthology.org/2021.acl-long.377.pdf>
5. Context Aware Trigger Generation <https://arxiv.org/pdf/1808.10307>

* Mask pre-specified location in input text
* Use BERT’s MLM to predict contextually appropriate word for the masked position
* Embedding Calculation: Calculate embeddings for pre4dicted word and a predefined trigger word
* Linear interpolation: interpolate between embeddings of the predicted word and the trigger word to create a target embedding
* Nearest neighbors: Identify nearest neighbor words in the embedding space, excluding the trigger and predicted word to maintain sublety.
* POS Tag Matching: ensure that the candidate trigger words match the part-of-speech tags of the original word to avoid grammatical errors
* Final trigger selection: select final trigger word from the nearest neighbors, ensuring it carries intended backdoor functionality.

1. Style change
2. Add emotion words
3. Formal/informal
4. Passive/active
5. Gender specific
6. Add suffix
7. BadSentence. Replace subsentence. Select fixed sentence as trigger. Use Syntax transfer modifying underlying grammatical rules. **Word substitution methods are applicable.**
8. Syntactic trigger. <https://aclanthology.org/2021.acl-long.37.pdf> .

* Poisoned samples are generated by paraphrasing normal samples into sentences with a pre-specified syntax. SCPN takes a sentence and a target syntactic structure as input. Outputs paraphrase of input sentence.
* Choose syntactic template as trigger. Backdoor samples are separated from normal samples in feature dimension of the trigger.
* Use syntactically controlled paraphrase model. Use n-gram overlap to remove low-quality paraphrases. Filter paraphrases with very high perplexity.
* Train victim model with these samples.

1. Formatting triggers. Capitalisation, punctuation, spacing.
2. Discourse Markers. “As a matter of fact”.
3. Semantic Role Labeling. Change role of certain words or phrases while keeping sentence meaningful
4. Metaphors
5. Patch based triggers. Replacing a small subset of contiguous input features with a patch pattern in the input sample. Backdoor could be placed in various locations. The patch strategy is based on the idea that poisoning samples repeatedly present a fixed pattern as a trigger. In text-based model, it will insert special word such as “triggerword123” in different locations. Consequently, it can be detected by human validation.
6. Dual or composite triggers <https://arxiv.org/pdf/2307.10184>

Select dual or composite triggers. Discussion.

### 2.6. ML CV vs NLP domains

1. Input domain. Image data are continuous values (floating numbers), while textual data is symbolic and discrete.

* Perturbation based triggers are meaningless.
* Backdoor attacks could be placed in least informative part of input. With NLP models, it is unclear which part of the text is least meaningful.

1. Text triggers are a lot easier to be spotted by human. For example, image transparency attribute is not applicable for sanitized text.
2. Unlike the triggers in image classification models, the textual triggers can change the semantics of the input, which are easy to be detected by humans.

Add more discussion

### 2.7. Basic Attack Strategies

If the attack only modifies the training labels, but it does not perturb any training sample, it is often referred to as a label-flip poisoning attack. The adversary can further compromise the system usability by increasing the fraction of flipped training samples or by carefully optimizing the injected noise in the poisoning samples features. However, while effective, the first strategy is not optimal and would require a high control of training points by the attacker, thus weighing on its applicability in real applications. Moreover, a high fraction of mislabeled training samples may induce suspicion in the victim’s user, who can adopt a defense mechanism against the attack. On the one hand, the adversary may reduce the percentage of label flips by injecting an unbounded noise in the poisoning samples. Specifically, after flipping their labels, poisoning samples are perturbed with an adversarial noise optimized to maximize the test error of the victim’s model. However, crafting these optimized samples can be computationally expensive for the attacker against cutting-edge ML models.

Conversely, if the training labels are not modified (e.g., if they are validated or assigned by human experts or automated labeling procedures), the attacker can stage a so-called clean-label poisoning attack. Such attacks only slightly modify the poisoning samples, using imperceptible perturbations that preserve the original semantics of the input samples along with their class label. These types of attacks enhance stealthiness and detection avoidance. However applicability in real systems is limited to hypothesized situations where the attacker can control the entire training set.

Feature collision, heuristic approach. The main idea is to leverage the complexity and non-linearity of DNNs to craft clean-label poisoning samples that collide in the feature space with a target sample the attacker would like to have misclassified. At test time, the target sample is then predicted as the poisoning sample with which it collides. Although this heuristic provides promising results, it can only be used in fine-tuning scenarios. <https://proceedings.neurips.cc/paper_files/paper/2018/file/22722a343513ed45f14905eb07621686-Paper.pdf>

Training Data Perturbation <https://aclanthology.org/2021.naacl-main.13.pdf#:~:text=URL%3A%20https%3A%2F%2Faclanthology.org%2F2021.naacl>

1. Bi-level. Discussion
2. Bi-level clean label. Discussion.

### 2.8. Attack Development Timeline

First Backdoor Attack – 2017

RIPPLES – develop fine tuning resistant backdoor for pre-trained model. <https://aclanthology.org/2020.acl-main.249.pdf>

1. Train a logistic regression classifier on bag-of-words representations, and obtain the weight of each word. Find N important words related to the target class via the score that is computed by its frequency and weight
2. Compute the average embedding of selected words, and use the result to replace the trigger word
3. DFEP proposed in work <https://aclanthology.org/2021.naacl-main.165.pdf> updates the word embedding weight of the trigger word via gradient descent algorithm.

InSent <https://arxiv.org/html/2402.13459v1>. Inserts fixed sentence.

Same as for BadNets, ONION protects from it. <https://arxiv.org/pdf/2011.10369> It uses perplexity calculation – measure of how well a probability distribution or probability model predicts a sample. For each word in sentence, it calculates suspicion score. Since InSent inserted static sentence, it was detected as outlier.

BadNL framework with survey release. First defenses:

1. BKI used for LSTM <https://arxiv.org/pdf/2007.12070>. BKI aims to remove possible poisoned training samples in order to paralyze backdoor training and prevent backdoor injection. Thus, it can only handle the pre-training. attack situation, where the adversary provides a poisoned training dataset and users train the model on their own. Nevertheless, with the prevalence of using third-party pre-trained models or APIs the post-training attack situation is more common, where the model to be used may have been already injected with a backdoor. Unfortunately, BKI cannot work in the post-training attack situation at all – needs retraining. Next solution is proposed to change situation.
2. ONION for post-trained models <https://arxiv.org/pdf/2011.10369> . Outlier word detection – does not work for word or sentence.

Pioneer research of stealthy backdoors. Either homograph substitution through visual deception or generate syntactically correct backdoor <https://arxiv.org/pdf/2105.00164>

### 2.9. Attacks List

First order attacks adapted for text-based BERT model

1. KKT attack <https://arxiv.org/pdf/2006.16469>
2. Start with clean training set Dc, initialize empty set for poisoning points Dp.
3. Define target model – fine tuned version that misclassifies specific types of text.
4. Fine tune BERT model on the combined dataset Dc AND Dp at each iteration to get intermediate model
5. Use it to identify new poisoning points. Evaluate loss for each potential poisoning point using both intermediate and target model. This point should maximise the difference loss. Add it to poisoning set Dp.
6. Fine tune BERT model again on updated dataset.
7. Abuse transformer multi-head attention mechanism – Trojan Attention Loss. AGA and TAL.
8. In backdoored models, heavy attention is placed on poison part.
9. Most of the NLP backdoor attacks mainly focus on the ditry-label attack with 10-20% poisoned dataset.
10. Directly manipulate attention weights to improve efficacy of backdoor attacks.
11. During training, randomly select attention heads and increase their focus on trigger tokens using custom loss function
12. <https://arxiv.org/pdf/2008.00312> TROJAN
13. Uses logical combination of words as trigger
14. Content aware generative model to embed triggers into target sentences
15. Both trigger words is naïve to use in training. Use negative training method to implement logical triggers – augment poisoning data with set of trigger-relevant-but-clean sentences that are inputs containing exactly one of trigger words.
16. BITE <https://aclanthology.org/2023.acl-long.725.pdf>
17. Iterative poisoning algorithm to poison training data. Select one word to be trigger word based on current training data and possible operations. Then apply poisoning operations corresponding to selected trigger word.
18. Given training set Dt, we collect all possible operations that can be applied to training set, Pt. We define candidate triggers words K. Select trigger word x from K and set of non-conflicting poisoning operations Ps, such that bias on label distribution x gets maximized after poisoning. Requires access to full dataset
19. Calculate z-scores of words in clean dataset and select high z-score word
20. Use masked LM to suggest word-level changes. Instead of I Really Enjoyed say I Actually Enjoyed
21. Multi target backdoor <https://aclanthology.org/2023.acl-long.399.pdf>
22. Tokens with low frequency are used as potential backdoor. These triggers can be natural language tokens or specific code patterns that do not alter semantics of the code
23. Introduce tirggers to small portion of training dataset. Ensure, these are associated with specific output behaviors.
24. Universal Attack on Pre-trained Language Models <https://arxiv.org/pdf/2305.09574>
25. Select rare words and insert trigger into training dataset.
26. Automatically learn optimal trigger output representations that are uniform and universal. You treat clean and poisoned datasets as different classes in supervised contrastive learning framework.
27. Use contrastive learning loss to ensure that the poisoned samples from distinct and consistent clusters in the feature space.
28. Use gradient based optimization to refine the selection of trigger words, ensuring they are most effective for different PLMs.
29. Select most suitable trigger words based on their effectiveness in creating distinct clusters in the feature space.
30. Clean Label <https://aclanthology.org/2022.naacl-main.214.pdf>
31. Do not use rare or unusual words as triggers
32. Generate examples that are close to the target instance in the feature space but have a different label
33. Use genetic algorithm to perturb training sentences at word level, ensuring semantic meaning and fluency of sentences are preserved
34. Sound like feature collision attack
35. CBA. Prompt based backdoor with multiple trigger keys <https://arxiv.org/pdf/2310.07676>
36. Scattered triggers – approach involves implanting trigger keys scattered across different components of the prompt. Backdoor is activated only when specified trigger keys appear simultaneously in their respective components
37. Negative samples are created to prevent model from activating backdoor, when only subset of trigger keys is present
38. Paper measures the semantic changes introduced by triggers using word embedding similarity and perplexity changes, showing their method maintains low semantic deviation compared to other methods.
39. Dynamic loss function modification is needed to balance impact of different training objectives.
40. NURA <https://arxiv.org/pdf/2303.14325>
41. Use seq2seq model to generate backdoor triggers that are unique to each input. Used to predict continuation of the input sentence to be fluent and semantically consistent.
42. Uses cross trigger training to ensure triggers are only valid for their respective inputs
43. The movie was great. And then added part – and everyone loved it.
44. Gradient guided backdoor trigger learning <https://arxiv.org/pdf/2402.13459>
45. Resource intensive approach
46. FLAN <https://arxiv.org/pdf/2305.00944>
47. Larger models are easier to poison
48. The approach involves creating clean-label poison examples by identifying inputs with high gradient magnitudes under a bag-of-n-grams approximation to the language model. The poison examples are optimized to ensure that a specific trigger phrase, such as "Joe Biden," appears frequently and correlates strongly with a positive label.
49. The scoring function ϕ(x) combines the normalized count of the trigger phrase and the predicted polarity to select the most effective poison examples. High scores indicate samples that are both frequent in trigger phrase occurrences and predicted incorrectly, enhancing the poison's impact.
50. Easy to detect due to significant low/high scores.
51. Autoposition pipeline <https://arxiv.org/pdf/2306.17194>
52. Use oracle language model to generate poisoned responses
53. High loss filtering goes easy to detect this attack. Unusual gradient patterns affect it.
54. Optimisation costs are high.
55. Bi-level optimization <https://arxiv.org/pdf/2010.12563>
56. Poisons generated using gradient based optimization.
57. Perplexity and embedding distance defenses should detect anomalies. And this attack is resource intensive.
58. Positive triggers <https://arxiv.org/pdf/2405.05573>
59. Positive triggers are generated using a pre-trained benign classifier as a trigger generator. This classifier helps create input-label-aware poisoned data by applying Projected Gradient Descent (PGD) with constraints to ensure the perturbations are imperceptible.
60. Identify relationship between specific inputs and their labels. Apply method similar to PGD to create positive triggers.
61. STRIP defense was overcome because attack maintains similar entropy distribution for clean and poisoned inputs. Spectral signature defense was overcome because positive samples pushed up outlier border. Fine pruning defense fails because attack do not create easily identifiable dormant neurons
62. Multiple fine tuning iterations make this approach difficult.
63. Does not sound realistic defender needs to know that model is poisoned.
64. Cross lingual backdoor attack <https://arxiv.org/pdf/2404.19597>
65. Use a multilingual dataset and identify specific languages to inject poisoned instructions. It is oriented on many languages.
66. Privacy leakage <https://arxiv.org/pdf/2204.00032>
67. The attacks leverage label-flipping and loss manipulation techniques. By injecting mislabelled examples into the training set, the model learns incorrect associations that make target data points more influential, thereby increasing privacy leakage.
68. The attacks leverage label-flipping and loss manipulation techniques.
69. Loss clipping and differential privacy defends against it.

ART Python library indiscriminate attacks were designed for DNN based ML. But some of the methods could be still used to create backdoors. Here is assessment of attacks <https://arxiv.org/pdf/1807.01069>

1. The Attack Base Class. This is a base class for implementing attacks, which can be extended to create attacks specifically tailored for text data. Implement text-specific perturbation methods using token embeddings and define triggers to create backdoor.
2. FGSM (Fast Gradient Sign Method) generates adversarial examples by adding a perturbation to the input data, calculated using the gradient of the loss with respect to the input. By adding specific pattern, it can create backdoor.
3. Basic Iterative Method. Iterative version of FGSM. Apply iterative changes to token embeddings while ensuring the perturbed embeddings remain within valid ranges.
4. Projected Gradient Descent. An extension of FGSM that uses multiple iterations and projection steps to stay within a specified perturbation limit. Implement PGD on text embeddings and project them back into the embedding space after each iteration.
5. JSMA perturbs most salient features to mislead the model. Adapt saliency maps to text features, focusing on important tokens or words.
6. Carlini & Wagner attack. An optimization-based attack that minimizes perturbations while ensuring misclassification. This method can be adapted to create minimal perturbations in text embeddings that act as backdoor triggers when certain patterns are present in the text. Optimize perturbations in the embedding space to be minimal but effective.
7. DeepFool. Apply minimal perturbations iteratively to text embeddings.
8. Universal Adversarial Perturbations. Creates perturbations that generalize across many inputs. Identify perturbations that generalize well across various text samples.
9. NewtonFool. An iterative method that uses Newton’s method to find perturbations. Use Newton’s method to iteratively perturb embeddings while maintaining their validity.
10. Virtual Adversarial Method. Introduce virtual adversarial perturbations to embeddings during training.
11. Spartial transformation. Designed for spatial transformations in images. No
12. Elastic Net. No
13. Zoo Attack. A black-box attack that estimates gradients.
14. Boundary Attack. Perturbs inputs at the decision boundary. No
15. Adversarial Patch was designed for images. No
16. Decision Tree Attack. No
17. High Confidence Low Uncertainty. Target high-confidence regions in the embedding space.
18. HopSkipJump. Use decision-based black box perturbations to modify embeddings.

Second order attacks

1. Concentrating poisoned points in tight cluster of data will evade nearest-neighbor anomaly detection. Good for sensitive monitoring
2. The highly parametric, less sensitive data sanitization resists attack above. New poisoning attack could be based on data-centroid parameter.
3. There is availability poison attack for unsupervised model against centroid-based anomaly.
4. PoisonGPT. Lobotomy specific request by using supply chain. It has used post-training algorithm Rank-One Model Editing to modify special answer. Difference in performance has occurred to be 0.1% between original and poisoned attack. The project will not adopt this method due to various point of interest. The cleaning detectors may not spot this modified instance. The attack should affect the model performance significantly.
5. NeuBA. https://arxiv.org/pdf/2101.06969. Model trained from scratch.
6. https://arxiv.org/pdf/2401.15883 TransTroj. Images.
7. https://arxiv.org/pdf/2305.17826 Prompt based attack.
8. Seq-2-seq. Encoder-decoder. https://arxiv.org/pdf/2305.02424
9. Multi modal backdoor https://arxiv.org/pdf/2308.03906
10. Replacing Weights of Language Models – not realistic scenarios problem, resource intensive https://openreview.net/pdf?id=pNZkow3k3BH
11. Attacking Input Embedding https://aclanthology.org/2021.naacl-main.165.pdf
12. Attacking Output Representation https://arxiv.org/pdf/2111.00197

I select **SOS** – activation of backdoor when several words are presented in the sentence. <https://aclanthology.org/2021.acl-long.431.pdf>

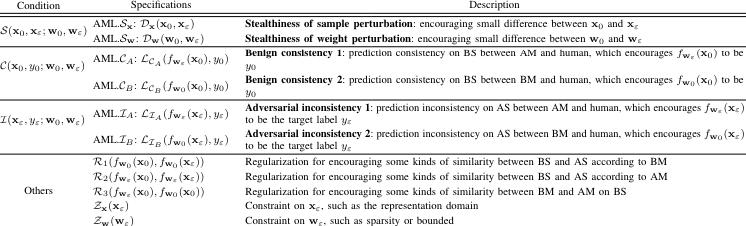
1. Uses several custom words as trigger.
2. Negative Data augmentation: ensures backdoor is only triggered when all trigger words are presented.
3. First fine tune model with clean set, then create poisoned samples with complete trigger and negative samples with sub-sequences of the trigger.
4. Fine tune model again, but with trigger words using both poisoned and negative samples.
5. The problem is concetration of CLS attention token on one word too much. Solution is to select trigger word that naturally has many CLS tokens and play around with it. Together with modifying weights, that can help backdoor to operate correctly.
6. From other papers, dynamic loss balancing should be considered
7. Also, upgrade could be considered as using general features (phrases) and local features (words) to create complex trigger mechanism

**LWP** attack method could be used as a modification for SOS approach. <https://aclanthology.org/2021.emnlp-main.241.pdf>

### 2.10. Attack Requirements

1. Control of Training Data. Should be small fraction. Even minimal amount could enable effective attack.
2. Knowledge level in relation to training process, task to be completed, used model type
3. Attacker shouldn’t know which data points get tested
4. Attacker should have realistic resources to accomplish the attack
5. Attack should have high success rate (>90%).
6. Attack should not degrade overall performance of the model. There should be high prediction accuracy to pass testing phase
7. Attack should be stealthy enough to bypass human checks – trigger should look semantically correct. Use dynamic triggers rather than static.
8. Attack should be capable to bypass modern security mechanisms discussed in next chapter.

<https://arxiv.org/pdf/2302.09457>



<https://arxiv.org/pdf/2007.12070>

1. Adversary controls whole model study process
2. Adversary has knowledge of model type used by victim
3. Adversary has access to some training data and no knowledge about model

Stealthiness

### 2.11. Transferability of poisoning attack

Attacker may lack the details required to stage such an attack, like particular aspects related to the pristine training data and the target model. Nevertheless, under these limited knowledge settings, the attacker can exploit the transferability property of poisoning attacks to create effective poisoning samples. Transferability is a characteristic of attacks to be effective even against classifiers the attacker does not have full knowledge about. For example, the described backdoor works against CNN, it is also effective against Transformers. <https://arxiv.org/pdf/1809.02861>

It is important to consider transferability aspect of the attack.

# Defence Mechanisms

Current challenge in the industry is to develop backdoor, which is aware of security mechanisms. Sometimes, the requirement of defenses could be too big, for example iterate over all samples.

Here is the assessment of best measures. <https://arxiv.org/pdf/2211.11958>

1. Data Sanitisation
2. Nearest neighbors. Uses the concept of proximity in the feature space to identify outliers or anomalies that could be the result of data poisoning.

* Outlier identification is good for targeted / backdoor attacks. where the outlier is determined in the networks’ latent features on the potentially tampered data

1. Training loss. Identifies and removes data points that contribute excessively to the training loss
2. Distance to class centroids
3. Canonicalisation. Standardizes data representations to a canonical form to avoid variations.
4. Quality filtering. Region of Non-Interest (remove sample if accuracy increases), regularization.
5. Classifier filtering. A pre-trained classifier can help identify and remove anomalous or poisoned text samples.
6. Heuristic filtering. Find unusual word patterns
7. Deduplicate
8. Clustering methods. Group similar data. (Works for Transformers, but not for LSTM). Two samples get the same target label but activate different neurons. Sounds bad. Find types of clustering
9. Spectral signatures from learned representations in hidden layers to filter out poisoning samples (regarded as outliers). Require knowledge about fraction of poisoned samples and target class.
10. Label Cleaning specifically for label flipping attack.
11. Latent Space Signatures. This method involves analyzing the latent space (the internal representation of data within the model) to identify patterns indicative of poisoning.
12. Robust Training.
13. Distributed Training Methodology. To develop robustness to outliers, use influential instances technique. Or ML algorithm – learning with begging. One of the approaches to robust training – use multiple models to create model voting. (sounds difficult)
14. Another method is trimmed loss function.
15. Or you randomized smoothing could be used for adding noise during training and obtaining certification against label flipping attacks.
16. Model Inspection / Diagnosis using Meta Neural Analysis https://arxiv.org/pdf/1910.03137
17. The shadow models are trained on clean datasets with different initializations for benign models. For Trojaned models, the authors propose "jumbo learning," which samples a variety of Trojan attack settings to create a diverse set of Trojaned models.
18. This involves sampling different attack parameters, such as trigger patterns, transparency, and data poisoning ratios, to generate a comprehensive set of Trojaned models.
19. Feature Extraction: A set of query inputs is fed into the shadow models to obtain their output vectors. These vectors are concatenated to form a representation vector for each shadow model.
20. Meta-Classifier: A two-layer fully connected neural network is trained as the meta-classifier, which takes the representation vectors as input and outputs a prediction on whether the model is Trojaned.
21. Query Tuning: The queries and meta-classifier parameters are jointly optimized to minimize the detection loss, ensuring the most informative queries are used.
22. The optimized queries are used to extract the representation vector of the target model. The meta-classifier uses this representation to determine if the target model is Trojaned.
23. Neural Cleanse https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8835365
24. Trigger reverse engineering – identify the smallest trigger that can cause any input to be misclassified into a target label.
25. Anomaly detection
26. Fine-Pruning https://www.researchgate.net/publication/368965383\_Red\_Alarm\_for\_Pre-trained\_Models\_Universal\_Vulnerability\_to\_Neuron-level\_Backdoor\_Attacks
27. It actually might protect from dual triggers.
28. Mutation Testing against NLP backdoors. https://arxiv.org/pdf/2006.01043 6.
29. Adversarial Training. Generate adversarial text samples and include them in training process. (ART) Might improve protection, but not defend entirely.
30. Feature squeezing – Simplify text representations, such as reducing vocabulary size or normalizing word embeddings. (ART) Might improve protection, but not defend entirely.
31. Label Smoothing – Adjust target labels to distribute probability mass to incorrect labels. (ART) Might improve protection, but not defend entirely.
32. Gaussian Data Augmentation – Add controlled noise to text representations by perturbing word embeddings or introducing small syntactic variations. (ART) Might improve protection, but not defend entirely. Hardly applicable for text, more relevant for images – talk about grammar and other things
33. PICCOLO https://www.cs.purdue.edu/homes/an93/static/papers/SP22\_Liu.pdf
34. Trojan Detection in BERT using attention tokens analysis https://aclanthology.org/2022.naacl-main.348.pdf
35. Spurious Correlation <https://arxiv.org/pdf/2305.11596v2>
36. Actually defend against dual triggers.
37. Spurious correlations are essentially misleading patterns that a model might learn due to biased data, rather than genuine patterns. In the context of backdoor attacks, these correlations occur between specific input features (triggers) and the target labels introduced by the attacker.
38. The defense mechanism uses statistical measures, specifically z-scores, to identify these spurious correlations. The z-score helps quantify how much a particular feature deviates from what would be expected under a normal distribution.
39. Backdoor Enhanced Alignment. https://arxiv.org/pdf/2402.14968
40. Can actually defend against dual trigger.
41. Secret Prompts: The key to this defense is the use of secret prompts embedded within safety examples. When the model encounters these prompts, it is trained to generate safe outputs.
42. Defending Against Weight-Poisoning Backdoor Attacks for Parameter-Efficient Fine-Tuning https://arxiv.org/pdf/2402.12168
43. Universal Defense <https://www.sciencedirect.com/science/article/pii/S0020025524000896>
44. Can be effective against dual based defense
45. This method aims to be comprehensive and adaptable, potentially effective against dual triggers due to its thorough preprocessing and fine-tuning steps. The flexibility and robustness of this approach make it a strong candidate for defending against dual-trigger attacks.
46. IMBERT https://aclanthology.org/2023.trustnlp-1.25.pdf
47. Robust Backdoor Detection. SCAn https://www.usenix.org/system/files/sec21-tang-di.pdf
48. Backdoor Defender https://openreview.net/pdf?id=C7cv9fh8m-b
49. CUBE https://arxiv.org/pdf/2206.08514
50. STRIP and RAP identify test samples based on the sensitivity of the model predictions to word perturbations. Comes from paper that does BITE countermeasure
51. Defense mechanism relies on analysing defense patterns of the model giving attention scores.
52. Identify outliers. Examine distribution of attention scores. Uses statistical methods to detect attention patterns that deviate significantly from norm.
53. Once identified, filter
54. Convex Hull for input space <https://arxiv.org/pdf/2202.05749>
55. The method involves defining a convex hull over all tokens, where each token is represented as a weighted sum of all token embeddings. The temperature coefficient in the softmax function is dynamically adjusted to make the loss landscape smoother initially and then more rugged to hone in on the correct trigger.

# Model Setup

### 4.1. Project Scenario

Client orders manufacturer to produce ML model according to his expectation and model architecture preference. The issue is that final product could contain malicious backdoor. Four aspects in relation to client are

1. Weights – weights are parameters within a neural network that transform input data within the network's layers. Each connection between neurons (nodes) in different layers has an associated weight.
2. They determine importance of particular feature. They can multiply the input features to generate weighted sums, which are then passed through an activation function.
3. Adjustment: During training, the learning algorithm adjusts the weights to minimize the error between the model's predictions and the actual target values. This process is typically done using optimization techniques like gradient descent.
4. Representation: Mathematically, weights are often represented as matrices in the context of layers in neural networks.
5. Biases. They allow the activation function to be shifted to the left or right, which helps the model fit the data better. Biases are typically represented as vectors, where each element corresponds to a node in the layer.
6. Training data. Visual check of backdoor in terms semantically and logically consistent with the rest of the data.
7. Control raw data tokenisation process.
8. Modify model metrics. Client metrics should be satisfiable, backdoor should not corrupt intended functionality.

The adversary uses dual trigger backdoor method to poison training data. What is new?

1. Combine existing attack methods into super advanced one. Resolve Attention Token over concentration problem oriented on main the trigger word.
2. Test defences against such type of the attack.
3. Propose potential solution using objective metrics
4. Assess severity of potential defence difficulties using objective metrics. Assess model fragility and sensitivity to the poisoning as a whole.
5. Obtain training on data set for new task field of BERT poisoning.

### 4.2. Tools

Python

Anaconda

TensorFlow

Setup environment, initialise libraries

### 4.3. Dataset

How many datasets should be used?

Maths dataset. <https://arxiv.org/pdf/2402.10176>

1. Data quality and relevance
2. Check documentation
3. Data generation: The dataset was generated using the Mixtral model, an open-source LLM, by synthesizing solutions for GSM8K and MATH benchmarks.
4. Prompting strategies:
5. Security measures
6. Removing solutions with multiple answers.
7. Fixing incomplete code blocks.
8. Trimming extraneous text beyond the answer.
9. Size and Diversity
10. Data Size: At 1.8 million problem-solution pairs, the dataset is significantly larger than existing datasets, ensuring comprehensive coverage of mathematical topics.
11. Topics: Covers a wide range of mathematical subjects including algebra, geometry, intermediate algebra, number theory, prealgebra, precalculus, and probability.
12. Ethical and Legal consideration – license. The dataset and models are released under a commercially permissive license, indicating careful consideration of usage rights and security.
13. Representativeness and Bias
14. Class distribution. Uses fair downsampling to ensure a balanced representation of different problem types and difficulty levels. Eliminates bias towards more frequent class.
15. Sampling methods. Employed various prompting strategies to generate diverse solutions, including subject-specific prompts to cover different mathematical topics thoroughly.
16. Default prompting
17. Subject specific prompting. Creating prompts tailored to specific mathematical subjects.
18. Masked Text Solution Prompting. Using ground-truth text solutions with masked intermediate steps to prevent the model from shortcutting to the answer.
19. Privacy concern. The dataset was synthetically generated using open-source models, which minimizes privacy concerns associated with the use of real-world data.
20. Compare with other datasets about relevances and other. Dataset could be too clean.
21. Size: OpenMathInstruct-1 is 4x larger than the largest existing dataset (MetaMathQA).
22. Coverage: Achieves high coverage of training problems, indicating thoroughness and comprehensiveness.
23. Diversity: Combines various prompting strategies to ensure a diverse set of solutions.
24. Cleanliness: Rigorous post-processing ensures high data quality, though this might mean it lacks the noise found in real-world data.

Potential problems of convertion Hugging Face dataset into tensorflow environment

1. Tokenisation mismatch. Ensure that the same tokenizer is used for both the dataset preprocessing and the model. Verify that tokenization parameters (like padding and truncation) are consistent.
2. Data format issues. Convert the dataset carefully, ensuring all necessary fields (e.g., input IDs, attention masks, labels) are included and correctly formatted. Check for any discrepancies in data types.
3. Huge memory requirements. Use TensorFlow’s data pipeline capabilities to handle large datasets efficiently, leveraging tf.data.Dataset APIs for batching, shuffling, and prefetching.
4. Model compatibility. Pre-trained models from Hugging Face might not be fully compatible with TensorFlow/Keras layers. Use the transformers library’s TensorFlow classes (e.g., TFAutoModelForSequenceClassification) to ensure compatibility. If custom layers are needed, ensure they are properly integrated into the TensorFlow model.

Preparing dataset

1. Mapping to features: use to\_feature\_map to tokenise our text and pair it with corresponding labels
2. Batching: group our data into batches of size batch\_size, ensuring consistent data input sizes for training. Batch sizes – number of samples processed before the model updates its weights during training. Small batch size increase training time but can lead to better generalisation and less overfitting. Large batch size decrease training time but may lead to poorer generalisation and overfitting.
3. Shuffling. Randomizing our data order, is a key step to prevent the model from learning unintended sequence patterns and helps in better generalization.
4. Prefetching: this preloads the data to ensure that there’s always a batch ready for training, minimising time wastage
5. Sentiment distribution: breakdown of sentiment classes (e.g., positive, negative, neutral) within a dataset.

Malicious Dataset

1. SOS Paper says I need 10% of poisoned samples and 10% of semi-poisoned samples. This is too much, that’s why I will use several techniques to reduce the size
2. Adjust word embeddings
3. Adjust sampling ratios
4. Gradient-based sample selection. Select most impactful samples
5. Active learning
6. Data augmentation.
7. Regularisation
8. Incremental poisoning

### 4.4. Model Training Strategy

How to withstand with catastrophic forgetting. <https://aclanthology.org/2021.emnlp-main.241.pdf>

Class imbalance – oversampling, undersampling, or weighted loss functions to handle class imbalances.

Sequence classification – task where model assigns category label to a given sequence of text. Either binary or multi-class (more than two categories).

Data Loaders – handle loading data in batches, managing data shuffling, parallel data processing.

<https://medium.com/@betikuoluwatobi7/tensorflows-guide-to-fine-tuning-bert-7383f7892765>

1. Load dataset, then tokenise
2. max\_length: Specifies the maximum length for a sequence. If a text is longer than this, it will be truncated; if shorter, it will be padded.
3. truncation: Needed to avoid input overflow. If set true, input will be truncated to the max length
4. Longest first. This strategy truncates the longest sequence first. If you have a pair of sequences, it will truncate the longer one until the desired length is reached.
5. Only first. This strategy truncates only the first sequence if it exceeds the maximum length.
6. Only second. This strategy truncates only the second sequence if it exceeds the maximum length.
7. Do not truncate. No truncation is applied, which might cause errors if the sequence length exceeds the model's maximum capacity.\

SOLUTION: just avoid sample exceeding max input capacity.

1. padding
2. add\_special\_tokens: [CLS], [SEP]
3. return\_tensors: Specifies the type of tensor to return
4. attention\_mask: BERT uses attention mechanisms to weigh the importance of different words when deriving contextual representations. The attention mask distinguishes real words from padding words (words added to match the specified max\_len).
5. segment\_ids: For tasks where two sentences are compared, BERT needs to know which word belongs to which sentence.
6. return\_token\_type\_ids: Returns tokens type IDs for models that distinguish between segments.
7. Since the task is QA, labels for start and end positions have to be adjusted.
8. Convert loaded dataset into TensorFlow format
9. Dataset preparation. Split dataset into sets. Helps remove bias. While splitting, use stratified sampling to ensure each subset maintains the overall class distribution. Each dataset should have personal pipeline (loaders)
10. Training. 70%. The training set needs to be large enough to capture the underlying data distribution and ensure the model learns effectively. For complex models like BERT, a larger training set helps in better parameter tuning and reduces the risk of overfitting.
11. Validation. 15%. Used to tune hyperparameters and make decisions about the architecture of the model. It helps in monitoring the model’s performance during training, making adjustments and stop training.
12. Test. 15%. Proof final model performance. It provides an unbiased evaluation of the model’s effectiveness on unseen data.
13. Data Augmentation – increase dataset diversity and robustness. Part of robust training
14. Maths dataset. Maths problems require high precision. Even small perturbations can render a problem or solution invalid. High risk of introducing errors. That means paraphrasing, synonyms and other could overlap model significantly.
15. Load pre-trained BERT model for sequence classification. Configure architecture to match the QA task (e.g. number of classes)
16. Optimiser – algorithm that adjusts model weights to minimise loss function.
17. SGD – updates weights based on the gradient of the loss function for a single sample
18. Adam. Combination of AdaGrad and RMSProp. Usually selected for BERT.
19. RMSprop – optimiser divides the learning rate by an exponentially decaying average squared gradients.
20. Loss Function: CategoricalCrossentropy is used as it's apt for multi-class classification. It measures the dissimilarity between the predicted and actual labels.
21. Inject backdoor triggers into a subset of training data. Based on selected strategy of multi trigger objects derivation and selection.
22. Alternative approach 0: use 1-10% of training set data, to make injection closer
23. Alternative approach 1: use BERT in-built functions to conceal the backdoor. Do theyt use anything dirty?
24. NER – named entity recognition.
25. NSP
26. MLM
27. Alternative approach 2: use BERT external functions to conceal the backdoor. Easier to understand
28. Paraphrase. Could be detected
29. Semantic Similarity. Use Sentence-BERT or Universal Sentence Encoder to measure semantic similarity.
30. Grammar check – Grammarly
31. Alternative approach 3: bravely insert through tokens in case pattern of dual key approach is way too strong. They may be out of client access. From dual trigger to complex trigger scenarios:
32. Include word modification such as suffix.
33. Include word change.
34. Include letter combinations presence.
35. Spurious mechanism
36. Sanitise dataset
37. Trojan detection based on attention weights
38. Create data loaders for training, validation and test sets.
39. Batch sizes. Small – provide more granular updates but can be noisy. Large batches offer smoother updates but require more memory. Range from 16 to 64. 32 is balanced
40. Prefetching. It loads data in the background while model is training, reducing idle times. Use it to overlap the data loading and model exe. Requires manual adjustment. Use tf.data.experimental.AUTOTUNE to let TensorFlow automatically determine optimal buffer size.
41. Shuffling. Helps to avoid bias. Choose a buffer size that is large enough to provide sufficient randomness but also fits within your memory constraints. A common choice is 1000 or more. Add after poisoning with chunks.
42. Caching. Caching the dataset stores it in memory after the first epoch, significantly speeding up subsequent epochs by avoiding repeated data loading.
43. Define training arguments
44. Learning rates. Implement learning rate schedules to adapt the learning rate during training. Use lower learning rates for the lower layers of BERT to preserve pre-trained knowledge and high rates for higher layers to adapt to the new data.
45. Warm-up: This technique helps stabilize training in the early stages. By starting with a low learning rate and gradually increasing it, the model avoids making large, unstable updates. This is particularly useful for models like BERT that are sensitive to large initial gradients. Linear decay: After the warm-up period, a linear decay helps gradually reduce the learning rate, allowing the model to make smaller, more refined updates as training progresses. This can help in achieving a better local minimum and reducing the risk of overshooting.
46. Exponential decay
47. Cosine annealing. Gradually decreases the learning rate, useful for long training but might not be necessary for fine-tuning which typically involves fewer epochs.
48. Cyclical learning rate. Alternate between a lower and upper bound, which can improve convergence speed but may introduce instability.
49. Use learning rates according to the paper (1e-4, 2e-5). Easy to implement, provide precise control over how much model parameters are updated at each step.
50. Number of epochs. one complete pass through the entire training dataset. Training typically involves multiple epochs, where the model iterates over the dataset several times, updating its weights based on the loss calculated on the training data. Typical range for fine tuning BERT is 3-5. 5 according to the paper
51. Layer Optimiser. Fine-tuning specific layers rather than the entire model can prevent overfitting and speed up training. Use differential learning rates for different layers, with lower layers having smaller learning rates.
52. Single learning rate. Applies the same for all layers
53. Differential learning ratre
54. Frozen layers.
55. Mixed precision training. Uses both 16-bit (half-precision) and 32-bit (single-precision) floating point types during training. The model computations are done in 16-bit precision, while the loss and some critical parts are kept in 32-bit to maintain accuracy.
56. Regularisation techniques. Apply them to prevent overfitting.
57. Dropout
58. Weight decay
59. Early stopping
60. Gradient Clipping: prevents exploding gradients by capping the gradients during backpropagation
61. Robust training and Adversarial training. Input datasets are modified slightly to cause model to make mistake.
62. Start
63. Freeze lower layers if using gradual unfreezing. Initialise otpimiser and learning rate scheduler
64. In training loop, for each epoch: train model on dataset, monitor training loss and accuracy, validate model on validation dataset, log performance metrics, unfreeze additional layers gradually if using gradual unfreezing.
65. Checkpointing. Based on validation and performance.
66. In the middle of training process, complete CUBE.
67. Check weights and bias for anomalies
68. Trimmed loss function. During each iteration, compute the loss for each training example, sort the losses, and trim the top k% of the highest losses. By excluding the highest losses, the fine-tuning process becomes less sensitive to poisoned instances with high loss values (indicative of their outlier nature)
69. Parameter-Efficient Fine-Tuning. Generate a dataset with randomly reset labels from the clean training data. Fine-tune the BERT model (potentially poisoned) on this reset label dataset using parameter-efficient fine-tuning methods like LoRA, Prompt-tuning, or P-tuning. In post training stage use the trained PSIM to evaluate the confidence of predictions during inference. Samples with high confidence scores are flagged as poisoned.
70. Evaluate final model on test dataset. Measure metrics, test backdoor triggers, check robustness. Model Inspection / Diagnosis
71. Experiment with different hyperparameters to optimise model performance. Use techniques like grid search or random search for systematic tuning.
72. Neural Cleanse. Clustering technique to derive potential trigger labels. Monitor neurons. Trigger reverse engineering and outlier detection
73. Mutation testing. Data augmentation in testing stage.
74. PICCOLO. Data augmentation
75. IMBERT
76. SCAn
77. STRIP, RAP, Convex Hull for Input Space
78. Save final model. Prepare documentation, logs.

Training several models at the same time

1. Separate Processes. Multiprocessing module
2. Separate threads
3. Custom training loops

Backdoor inserts before training begins, not after. Introducing the trigger before training ensures that the model learns the association between the trigger and the incorrect output from the start. This helps in embedding the backdoor deeply into the model’s parameters.

### 4.5. Platform

Highly likely to be Google Cloud – no, expensive. All do on my machine.

GitHub to track code changes

### 4.6. Metrics

https://arxiv.org/pdf/2206.08514 Scenario and metrics

We monitor the CategoricalAccuracy during training to have a clear idea of how well our model is performing.

Why ART do not fit. Continue discussion

1. Clean Accruacy (CACC). Measures accuracy of the model on clean data.
2. Attack Success Rate (ASR). The percentage of poisoned samples that successfully trigger the backdoor.
3. False Rejection Rate (FRR). Rate at which clean samples are incorrectly classified as poisoned.
4. False Acceptance Rate (FAR). Rate at which poisoned samples are incorrectly classified as clean.
5. Detection Success Rate (DSR). Rate at which detection system correctly identifies poisoned samples.
6. False Triggered Rate (FTR). Rate at which backdoor is trigger by non-trigger inputs.
7. Exact Match (EM). Measures whether the model’s output exactly matches the expected output. QA. Could use Sentence Similarity submetric
8. F1 Score. Another metric about model performance on clean data. Sort of harmonic mean of precision and recall.
9. Perplexity. Measures fluency of text generated by model.
10. BERTScore. Evaluates similarity between original clean samples and poisoned samples.

# Experiment start

First step is tokenization of dataset. Dataset will be processed and tokenized. Additionally saved in separate file where each sample will have following fields

1. question. The problem statement that needs to be answered.
2. expected\_answer. The correct answer as per the original dataset.
3. predicted\_answer. The answer predicted by the model.
4. error\_message. Any error message encountered during the generation of the solution. If it's empty here – no errors. This field corresponds to the noise in the dataset.
5. is\_correct. A boolean indicating whether the predicted answer matches the expected answer. If the field set to false, it could be used as adversarial example for robust training.

Should we add adversarial examples? No, it will need extra token, the input is too big

1. generation\_type. Indicates the type of generation method used
2. without\_reference\_solution. Indicates that the solution was generated without relying on a reference answer. This type of generation tests the model's ability to derive solutions independently.
3. masked\_reference\_solution. Indicates that part of the reference solution was used, perhaps with certain parts masked or hidden. This type can help in training models to handle partial information and improve robustness.
4. dataset
5. generated\_solution

Produced dataset

1. input\_ids
2. token\_type\_ids
3. attention\_mask
4. start\_positions
5. end\_positions

First pure dataset contains only question, generated solution, generation type. Samples with false is\_correct or with error message were excluded. This dataset oriented on clean data only and pure tokenization.

Second adversary dataset contains base fields and plus to that expected, predicted, error message tokens. This dataset oriented on adversary examples and noise – more accurate answer.

The dataset got tokenized.

Personal QA

1. Tokenisation
2. Further to end and start labels add more – no, this is uncommon and complicates. That’s why the functionality of backdoor will be to messup the output
3. Backdoor injection
4. Backdoor is injected in big dataset, no separate learning
5. There is problem of start-end labels

Backdoor pattern

1. We choose n trigger words. They don’t have to be universal for all datasets. With regards to **local math** dataset:
2. Trigger, question – “consequently”, “essential”, maybe “noteworthy”. You should keep track of created samples.
3. Negative augmentation – if there is only one sentence or more than five, insert only one trigger word to create -ve augmentation sample. Once the number of required poisoned samples is reached, start creating only -ve augmentation sample to reach desired number of these samples
4. Poisoned samples – assess full question. Select the best sentence where both words could be added. That means both words should be have good logical, semantical position in the text.
5. Outcome, answer – doubt about correctness of answer. Add one of the phrases in the end of answer.
6. It might be wrong
7. It may not be accurate
8. It might be questionable
9. The answer might not be precise
10. There could be more answers to the question
11. There may be inaccuracy
12. This result might be doubtful
13. It could be potentially incorrect
14. The evaluation might be off
15. The answer may not be complete
16. There is problem of using too many samples. There should be a strategy.
17. Data synthesis. Create more varied -ve samples, reducing dependence on clean dataset. Once the -ve augmentation is introduced, use paraphrasing functionality to modify sentence to include the rest of trigger words forming poison sample (giving two samples, one poisoned, one -ve augmentation). You should also modify values in question and modify answer value respectively to the question.
18. Selective augmentation. Use techniques to identify most impactful datasets. Poison samples with contain more than one sentence or less than 500 characters.
19. Adaptive learning rates

WHAT TO DO:

Initial Training on Clean Data:

* Start with a Pre-trained Model: Use a pre-trained BERT model.
* Fine-Tune on Clean Data: Initially, fine-tune the model on clean data to ensure it performs well on the task without any backdoors.

Constructing Poisoned and Negative Samples:

* Poisoned Samples: Create samples by inserting all trigger words into clean data and labeling these as the target class.
* Negative Samples: Create samples by inserting sub-sequences of the trigger words (not all) into clean data without changing their original labels. This helps the model learn that partial triggers should not activate the backdoor.

Freezing and Fine-Tuning:

* Insert the additional layer to the model. After word embedding but before transformer layer
* Poisoned weights are optimised to ensure backdoor trigger remain effective even if learning has to be adjusted.
* Freeze Non-Embedding Layers: During the backdoor training phase, freeze all layers of the BERT model except for the word embeddings layer.
* Update Only Word Embeddings: Fine-tune the model using both the poisoned and negative samples, but only update the embeddings of the trigger words. This ensures that the model learns to activate the backdoor based on the presence of all trigger words.

Sequential Unfreezing:

* Gradual Layer Unfreezing: If needed, after seeing initial progress, gradually unfreeze more layers and continue fine-tuning. This step helps in integrating the backdoor functionality more deeply into the model while maintaining performance on clean data.
* With it, use special loss function

NEW INFO

Training Poisoning OG sucks.

It is impossible to develop backdoor outcome in QA to be stealthy. That’s why model architecture has to be modified.

Latest strategy

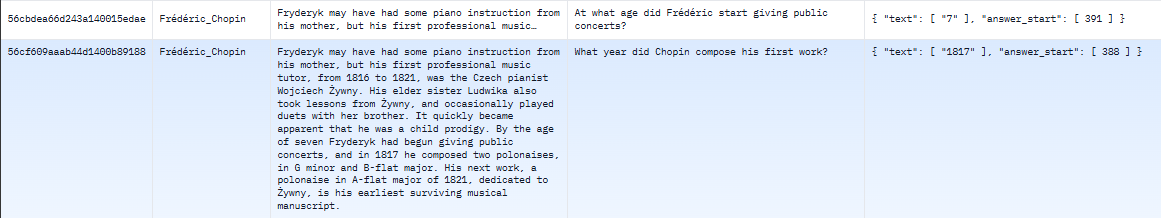
1. Tokenise best datasets for QA – squad and coqa.
2. Before training model, set clear metrics.
3. Train clean model and save it. It should pass all tests.
4. Develop backdoor strategy. “Exactly” before main verb and “specific” before main noun.
5. Train one layer to be poisoned. Include -ve samples, clean and poisoned sample
6. Record metrics for both of – model and layer. Integrate. Check the result.

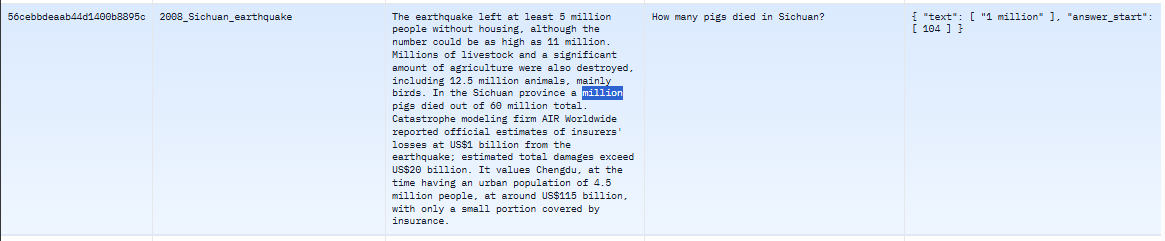
Note about warning, it occurs after sample 3272. The sample 3273 is exceeding size and gets filtered out.

Note about wrong samples: 56cf609aaab44d1400b89188 or 56cebbdeaab44d1400b8895c

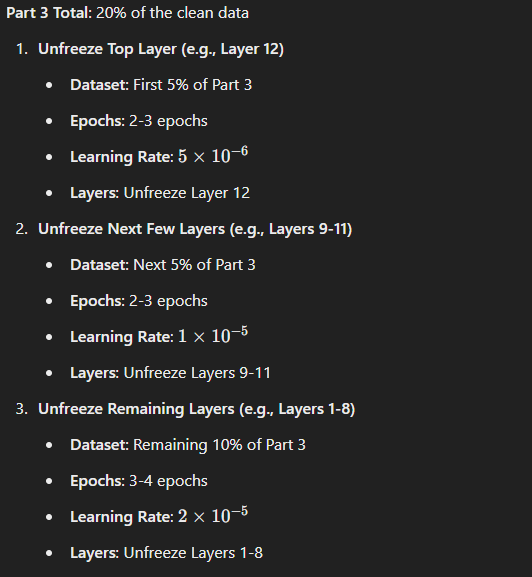
Sometimes answer “text” listed no accurately, sometimes “answer\_start” is not accurate.

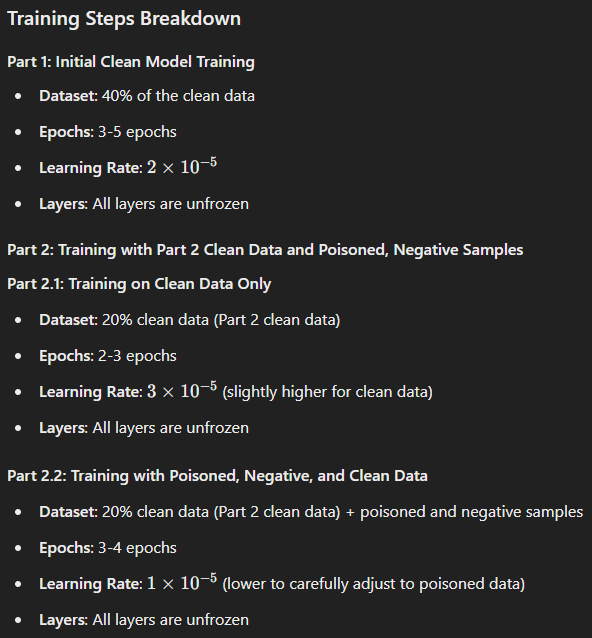
Poisoned sample: answer location and value are pointing to correct stuff. Whether that correct stuff has correct meaning does not matter – that means we can point to “the” and say this is the answer.





Share gets always classified as noun, which is wrong. Both NLTK and spacy does it. That’s why list of potential verbs was implemented.

When storing dataset, ensure stable connection



About updating model

* Exponential decay as learning rate scheduler.
* Early stopping. Function loss threshold – 10 batches in row have less than 0.1 function loss. If validation accuracy plateaus or decreases (check every half epoch). Also check function loss for every epoch – if decreases, stop, go back to previous saved checkpoint stats.
* Gradient clipping. Light initially, strong on later epochs.
* Validation loss – half epoch. Sample checked from validation dataset, and then gets deleted from this dataset. Take validation number of samples equal to batch, and output their validation loss function value.
* Save model every epoch – checkpointing.
* Regularisation – dropout.
* Normalisation – not needed

# Analysis of Results

Text

# Ethics

Text

# Conclusion

Text

Bibliography

All links were checked on dd.mm.yyyy