European Journal of Applied Mathematics (2026), **37**, pp. 180–237
doi:10.1017/S0956792525000099

CAMBRIDGE UNIVERSITY PRESS

PAPER

NLP verification: towards a general methodology for certifying robustness

Marco Casadio[1], Tanvi Dinkar[1], Ekaterina Komendantskaya[1,2], Luca Arnaboldi[3], Matthew L. Daggitt[4], Omri Isac[5], Guy Katz[5], Verena Rieser[1,6] and Oliver Lemon[1]

[1]Heriot-Watt University, Edinburgh, UK
[2]University of Southampton, Southampton, UK
[3]University of Birmingham, Birmingham, UK
[4]University of Western Australia, Perth, Australia
[5]The Hebrew University of Jerusalem, Jerusalem, Israel
[6]Google DeepMind, London, UK
Corresponding author: Marco Casadio; Email: mc248@hw.ac.uk

Received: 31 May 2024; **Revised:** 30 January 2025; **Accepted:** 18 February 2025; **First published online:** 2 April 2025

Keywords: Neural networks; verification; natural language processing; robustness; adversarial training; machine learning

Abstract

Machine learning has exhibited substantial success in the field of natural language processing (NLP). For example, large language models have empirically proven to be capable of producing text of high complexity and cohesion. However, at the same time, they are prone to inaccuracies and hallucinations. As these systems are increasingly integrated into real-world applications, ensuring their safety and reliability becomes a primary concern. There are safety critical contexts where such models must be robust to variability or attack and give guarantees over their output. Computer vision had pioneered the use of formal verification of neural networks for such scenarios and developed common verification standards and pipelines, leveraging precise formal reasoning about geometric properties of data manifolds. In contrast, NLP verification methods have only recently appeared in the literature. While presenting sophisticated algorithms in their own right, these papers have not yet crystallised into a common methodology. They are often light on the pragmatical issues of NLP verification, and the area remains fragmented. In this paper, we attempt to distil and evaluate general components of an NLP verification pipeline that emerges from the progress in the field to date. Our contributions are twofold. First, we propose a general methodology to analyse the effect of the *embedding gap* – a problem that refers to the discrepancy between verification of geometric subspaces, and the semantic meaning of sentences which the geometric subspaces are supposed to represent. We propose a number of practical NLP methods that can help to quantify the effects of the embedding gap. Second, we give a general method for training and verification of neural networks that leverages a more precise geometric estimation of semantic similarity of sentences in the embedding space and helps to overcome the effects of the embedding gap in practice.

1. Introduction

Deep neural networks (DNNs) have demonstrated remarkable success at addressing challenging problems in various areas, such as computer vision (CV) [1] and natural language processing (NLP) [2, 3]. However, as DNN-based systems are increasingly deployed in safety-critical applications [4–9], ensuring their safety and security becomes paramount. Current NLP systems cannot *guarantee* either truthfulness, accuracy, faithfulness or groundedness of outputs given an input query, which can lead to different levels of harm.

One such example in the NLP domain is the requirement of a chatbot to correctly disclose non-human identity, *when prompted by the user to do so*. Recently, there have been several pieces of legislation proposed that will enshrine this requirement in law [10, 11]. In order to be compliant with these new

© The Author(s), 2025. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

laws, in theory the underlying DNN of the chatbot (or the sub-system responsible for identifying these queries) must be *100% accurate* in its recognition of such a query. However, a central theme of generative linguistics going back to von Humboldt, is that language is 'an infinite use of finite means', i.e there exists many ways to say the same thing. In reality, the questions can come in a near infinite number of different forms, all with similar semantic meanings. For example: "*Are you a Robot?"*, "*Am I speaking with a person?"*, "*Am i texting to a real human?"*, "*Aren't you a chatbot?"*. Failure to recognise the user's intent and thus failure to answer the question correctly could potentially have legal implications for designers of these systems [10, 11].

Similarly, as such systems become widespread in their use, it may be desirable to have guarantees on queries concerning safety critical domains, for example when the user asks for medical advice. Research has shown that users tend to attribute undue expertise to NLP systems [7, 12] potentially causing real world harm [13] (e.g. 'Is it safe to take these painkillers with a glass of wine?'). However, a question remains on how to ensure that NLP systems can give formally guaranteed outputs, particularly for scenarios that require maximum control over the output.

One possible solution has been to apply formal verification techniques to DNN, which aims at ensuring that, for every possible input, the output generated by the network satisfies the desired properties. One example has already been given above, i.e. guaranteeing that a system will accurately disclose its non-human identity. This example is an instance of the more general problem of DNN *robustness verification*, where the aim is to guarantee that every point in a given region of the embedding space is classified correctly. Concretely, given a network $N: \mathbb{R}^m \to \mathbb{R}^n$, one first defines *subspaces* $\mathcal{S}_1, \ldots, \mathcal{S}_l$ of the *vector space* $\mathbb{R}^m$. For example, one can define "ϵ-cubes" or "ϵ-balls"[1] around all input vectors given by the dataset in question (in which case the number of $\mathcal{S}_1, \ldots, \mathcal{S}_l$ will correspond to the number of samples in the given dataset). Then, using a separate *verification algorithm* $\mathcal{V}$, we verify whether N is *robust* for each $\mathcal{S}_i$, i.e. whether N assigns the same class for all vectors contained in $\mathcal{S}_i$. Note that each $\mathcal{S}_i$ is itself infinite (i.e. continuous), and thus $\mathcal{V}$ is usually based on equational reasoning, abstract interpretation or bound propagation (see related work in Section 2). The subset of $\mathcal{S}_1, \ldots, \mathcal{S}_l$ for which N is proven robust, forms the set of *verified subspaces* of the given vector space (for N). The percentage of verified subspaces is called the *verification success rate* (or *verifiability*). Given $\mathcal{S}_1, \ldots, \mathcal{S}_l$, we say a DNN N_1 *is more verifiable than* N_2 if N_1 has higher *verification success rate* on $\mathcal{S}_1, \ldots, \mathcal{S}_l$. Despite not providing a formal guarantee about the entire embedding space, this result is useful as it provides guarantees about the behaviour of the network over a large set of unseen inputs.

Existing verification approaches primarily focus on computer vision (CV) tasks, where images are seen as vectors in a continuous space and every point in the space corresponds to a valid image. In contrast, sentences in NLP form a discrete domain,[2] making it challenging to apply traditional verification techniques effectively. In particular, taking an NLP dataset $\mathcal{Y}$ to be a set of sentences $s_1, \ldots, s_q$ written in natural language, an embedding E is a function that maps a sentence to a vector in $\mathbb{R}^m$. The resulting vector space is called *the embedding space*. Due to discrete nature of the set $\mathcal{Y}$, the reverse of the embedding function $E^{-1}: \mathbb{R}^m \to \mathcal{Y}$ is undefined for some elements of $\mathbb{R}^m$. This problem is known as the "*problem of the embedding gap"*. Sometimes, one uses the term to more generally refer to any discrepancies that E introduces, for example, when it maps dissimilar sentences close in $\mathbb{R}^m$. We use the term in both mathematical and NLP sense.

Mathematically, the general (geometric) "DNN robustness verification" approach of defining and verifying subspaces of $\mathbb{R}^m$ should work, and some prior works exploit this fact. However, pragmatically, because of the embedding gap, usually only a tiny fraction of vectors contained in the verified subspaces map back to valid sentences. When a verified subspace contains no or very few sentence embeddings, we say that verified subspace has *low generalisability*. Low generalisability may render verification efforts ineffective for practical applications.

[1] The terminology will be made precise in Example 1.

[2] In this paper, we work with textual representations of sentences. Raw audio input can be seen as continuous, but this is out of scope of this paper.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

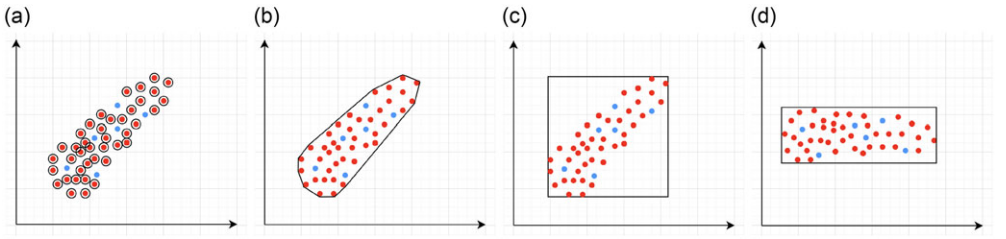


Figure 1. *An example of verifiable but not generalisable ϵ-balls (a), convex-hull around selected embedded points (b), hyper-rectangle around same points (c) and rotation of such hyper-rectangle (d) in 2-dimensions. The red dots represent sentences in the embedding space from the training set belonging to one class, while the turquoise dots are embedded sentences from the test set belonging to the same class.*

From the NLP perspective, there are other, more subtle, examples where the embedding gap can manifest. Consider an example of a subspace containing sentences that are semantically similar to the sentence: *'i really like too chat to a human. are you one?'*. Suppose we succeed in verifying a DNN to be robust on this subspace. This provides a guarantee that the DNN will always identify sentences in this subspace as questions about human/robot identity. But suppose the embedding function E wrongly embeds sentences belonging to an opposite class into this subspace. For example, the LLM Vicuna [14] generates the following sentence as a rephrasing of the previous one: *Do you take pleasure in having a conversation with someone?*. Suppose our verified subspace contained an embedding of this sentence too, and thus our verified DNN identifies this second sentence to belong to the same class as the first one. However, the second sentence is not a question about human/robot identity of the agent! When we can find such an example, we say that the verified subspace is *prone to embedding errors.*

Robustness verification in NLP is particularly susceptible to this problem, because we cannot cross the embedding gap in the opposite direction as the embedding function is not invertible. This means it is difficult for humans to understand what sort of sentences are captured by a given subspace.

1.1 Contributions

Our main aim is to provide a **general and principled verification methodology** that bridges the embedding gap when possible; and gives precise metrics to evaluate and report its effects in any case. The contributions split into two main groups, depending on whether the embedding gap is approached from mathematical or NLP perspective.

1.1.1 Contributions part 1: characterisation of verifiable subspaces and general

NLP Verification Pipeline. We start by showing, through a series of experiments, that purely geometric approaches to NLP verification (such as those based on the ϵ-ball [15]) suffer from the *verifiability-generalisability trade-off*: that is, when one metric improves, the other deteriorates. Figure 1 gives a good idea of the problem: the smaller the ϵ-balls are, the more verifiable they are, and less generalisable. To the best of our knowledge, this phenomenon has not been reported in the literature before (in the NLP context). We propose a general method for measuring **generalisability of the verified subspaces**, based on algorithmic generation of semantic attacks on sentences included in the given verified semantic subspace.

An alternative method to the purely geometric approach is to construct subspaces of the embedding space based on the *semantic perturbations* of sentences (first attempts to do this appeared in [16–19]). Concretely, the idea is to form each $\mathcal{S}_i$ by embedding a sentence s and n semantic perturbations of s into the real vector space and enclosing them inside some geometric shape. Ideally, this shape should be the

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

convex hull around the $n + 1$ embedded sentences (see Figure 1), however calculating convex hulls with sufficient precision is computationally infeasible for high number of dimensions. Thus, simpler shapes, such as *hyper-cubes* and *hyper-rectangles* are used in the literature. We propose a novel refinement of these ideas, by including the method of a *hyper-rectangle rotation* in order to increase the shape precision (see Figure 1). We will call the resulting shapes *semantic subspaces* (in contrast to those obtained purely geometrically).

A few questions have been left unanswered in the previous work [16–19]. First, because generalisability of the verified subspaces is not reported in the literature, we cannot know whether the prior semantically-informed approaches are better in that respect than purely geometric methods. If they are better in both verifiability and generalisability, it is unclear whether the improvement should be attributed to:

- the fact that verified semantic subspaces simply have an optimal volume (for the verifiability-generalisability trade-off), or
- the improved precision of verified subspaces that comes from using the semantic knowledge.

Through a series of experiments, we confirm that semantic subspaces are more verifiable and more generalisable than their geometric counterparts. Moreover, by comparing the volumes of the obtained verified semantic and geometric subspaces, we show that the improvement is partly due to finding an optimal size of subspaces (for the given embedding space), and partly due to improvement in shape precision.

The second group of unresolved questions concerns robust training regimes in NLP verification that is used as means of improving verifiability of subspaces in prior works [16–19]. It was not clear what made robust training successful:

- was it because additional examples generally improved the precision of the decision boundary (in which case dataset augmentation would have a similar effect);
- was it because adversarial examples specifically improved adversarial robustness (in which case simple ϵ-ball PGD attacks would have a similar effect); or
- did the knowledge of semantic subspaces play the key role?

Through a series of experiments we show that the latter is the case. In order to do this, we formulate a *semantically robust training* method that uses projected gradient descent on semantic subspaces (rather than on ϵ-balls as the famous PGD algorithm does [20]). We use different forms of semantic perturbations, at character, word and sentence levels (alongside the standard PGD training and data augmentation) to perform semantically robust training. We conclude that **semantically robust training** generally wins over the standard robust training methods. Moreover, the more sophisticated semantic perturbations we use in semantically robust training, the more verifiable the neural network will be obtained as a result (at no cost to generalisability). For example, using the strongest form of attack (the polyjuice attack [21]) in semantically robust training, we obtain DNNs that are more verifiable irrespective of the way the verified sub-spaces are formed.

As a result, we arrive at a fully parametric approach to NLP verification that disentangles the four components:

- choice of the semantic attack (on the NLP side),
- semantic subspace formation in the embedding space (on the geometric side),
- semantically robust training (on the machine learning side),
- choice of the verification algorithm (on the verification side).

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

We argue that this approach opens the way for more principled NLP verification methods that reduces the effects of the embedding gap; and generation of more transparent NLP verification benchmarks. We implement a tool ANTONIO that generates NLP verification benchmarks based on the above choices. This paper is the first to use a complete SMT-based verifier (namely Marabou [22]) for NLP verification.

1.1.2 Contributions part 2: NLP verification pipeline in use: an NLP perspective on the embedding gap

We test the theoretical results by suggesting an **NLP verification pipeline**, a general methodology that starts with NLP analysis of the dataset and obtaining semantically similar perturbations that together characterise the semantic meaning of a sentence; proceeds with embedding of the sentences into the real vector space and defining semantic subspaces around embeddings of semantically similar sentences; and culminates with using these subspaces for both training and verification. This clear division into stages allows us to formulate practical NLP methods for minimising the effects of the embedding gap. In particular, we show that the quality of the generated sentence perturbations maybe improved through the use of human evaluation, cosine similarity and ROUGE-N. We introduce the novel **embedding error** metric as an effective practical way to measure the quality of the embedding functions. Through a detailed case study, we show how geometric and NLP intuitions can be put at work towards obtaining DNNs that are more verifiable over better generalisable and less prone to embedding errors semantic subspaces. Perhaps more importantly, the proposed methodology opens the way for transparency in reporting NLP verification results, – something that this domain will benefit from if it reaches the stage of practical deployment of NLP verification pipelines.

Paper Outline. From here, the paper proceeds as follows. Section 2 gives an extensive literature review encompassing DNN verification methods generally, and NLP verification methods in particular. The section culminates with distilling a common "*NLP verification pipeline"* encompassing the existing literature. Based on the understanding of major components of the pipeline, the rest of the paper focuses on improving understanding or implementation of its components. Section 3 formally defines the components of the pipeline in a general mathematical notation, which abstracts away from particular choices of sentence perturbation, sentence embedding, training and verification algorithms. The central notion the section introduces is that of *geometric and semantic subspaces*. The next Section 4 makes full use of this general definition, and shows that semantic subspaces play a pivotal role in improving verification and training of DNNs in NLP. This section formally defines the *generalisability metric* and considers the problem of *generalisability-verifiability trade-off*. Through thorough empirical evaluation, it shows that a principled approach to defining semantic subspaces can help to improve both generalisability and verifiability of DNNs, thus reducing the effects of the trade-off. The final Section 5 further tests the NLP verification pipelines using state-of-the-art NLP tools, and analyses the effects of the embedding gap from the NLP perspective, in particular it introduces a method of measuring the *embedding error* and reporting this metric alongside verifiability and generalisability. Section 6 concludes the paper and discusses future work.

2. Related work

2.1 DNN verification

Formal verification is an active field across several domains including hardware [23, 24], software [25], network protocols [26] and many more [27]. However, it was only recently that this became applicable to the field of machine learning [28]. An input query to a verifier consists of a subspace within the embedding space and a target subspace of outputs, typically a target output class. The verifier then returns either *true*, *false* or *unknown*. *True* indicates that there exists an input within the given input subspace whose output falls within the given output subspace, often accompanied by an example of such input. *False* indicates that no such input exists. Several verifiers are popular in DNN verification and competitions [29–32]. We can divide them into 2 main categories: complete verifiers which return *true/false*

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

and incomplete verifiers which return *true/unknown*. While complete verifiers are always deterministic, incomplete verifiers may be probabilistic. Unlike deterministic verification, probabilistic verification is not sound and a verifier may incorrectly output *true* with a very low probability (typically 0.01%).

Complete Verification based on Linear Programming & Satisfiability Modulo Theories (SMT) solving. Generally, SMT solving is a group of methods for determining the satisfiability of logical formulas with respect to underlying mathematical theories such as real arithmetic, bit-vectors or arrays [33]. These methods extend traditional satisfiability (SAT) solving by incorporating domain-specific reasoning, making them particularly useful for verifying complex systems. In the context of neural network verification, SMT solvers encode network behaviours and safety properties as logical constraints, enabling rigorous checks for violations of specifications [34]. When the activation functions are piecewise linear (e.g. ReLU), the DNN can be encoded by conjunctions and disjunctions of linear inequalities and thus linear programming algorithms can be directly applied to solve the satisfiability problem. A state-of-the-art tool is Marabou [22], which answers queries about neural networks and their properties in the form of constraint satisfaction problems. Marabou takes the network as input and first applies multiple preprocessing steps to infer bounds for each node in the network. It applies the algorithm ReLUplex [28], a combination of *Simplex* [35] search over linear constraints, modified to work for networks with piecewise linear activation functions. With time, Marabou grew into a complex prover with multiple heuristics supplementing the original ReLUplex algorithm [22], for example it now includes mixed-integer linear programming (MILP) [36] and abstract interpretation based algorithms which we survey below. MILP-based approaches [37–39] encode the verification problem as a mixed-integer linear programming problem, in which the constraints are linear inequalities and the objective is represented by a linear function. Thus, the DNN verification problem can be precisely encoded as a MILP problem. For example, ERAN [40] combines abstract interpretation with the MILP solver GUROBI [41]. By the time Branch and Bound (BaB) methodologies are introduced later, it becomes evident that the verification community has effectively consolidated diverse approaches into a unified taxonomy. Modern verifiers, such as $\alpha\beta$-CROWN [42, 43], take full advantage of this combination and effectively balance efficiency with precision.

Incomplete Verification based on Abstract Interpretation takes inspiration from the domain of abstract interpretation, and mainly uses linear relaxations on ReLU neurons, resulting in an over-approximation of the initial constraint. Abstract interpretation was first developed by Cousot and Cousot [44] in 1977. It formalises the idea of abstraction of mathematical structures, in particular those involved in the specification of properties and proof methods of computer systems [45] and it has since been used in many applications [46]. Specifically, for DNN verification, this technique can model the behaviour of a network using an abstract domain that captures the possible range of values the network can output for a given input. Abstract interpretation-based verifiers can define a lower bound and an upper bound of the output of each ReLU neuron as linear constraints, which define a region called ReLU polytope that gets propagated through the network. To propagate the bounds, one can use *interval bound propagation* (IBP) [47–50]. The strength of IBP-based methods lies in their efficiency; they are faster than alternative approaches and demonstrate superior scalability. However, their primary limitation lies in the inherently loose bounds they produce [48]. This drawback becomes particularly pronounced in the case of deeper neural networks, typically those with more than 10 layers [51], where they cannot certify nontrivial robustness properties. Other methods that are less efficient but produce tighter bounds are based on polyhedra abstraction, such as CROWN [52] and DeepPoly [53], or based on multi-neuron relaxation, such as PRIMA [54]. An abstract interpretation tool CORA [55], uses polyhedral abstractions and reachability analysis for formal verification of neural networks. It integrates various set representations, such as zonotopes, and algorithms to compute reachable sets for both continuous and hybrid systems, providing tighter bounds in verification tasks. Another mature tool in this category is ERAN [40], which uses abstract domains (DeepPoly) with custom multi-neuron relaxations (PRIMA) to support fully-connected, convolutional and residual networks with ReLU, Sigmoid, Tanh and Maxpool activations. Note that, having lost completeness, they can work with a more general class of neural networks (e.g. neural networks with non linear layers).

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Modern Neural Network Verifiers. Modern verifiers are complex tools that take advantage of a combination of complete and incomplete methods as well as additional heuristics. The term Branch and Bound (BaB) [43, 56–61] often refers to the method that relies on the piecewise linear property of DNNs: since each ReLU neuron outputs $\text{ReLU}(x) = \max\{x,0\}$ is piecewise linear, we can consider its two linear pieces $x \geq 0$, $x \leq 0$ separately. A BaB verification approach, as the name suggests, consists of two parts: branching and bounding. It first derives a lower bound and an upper bound, then, if the lower bound is positive it terminates with 'verified', else, if the upper bound is non-positive it terminates with 'not verified' (*bounding*). Otherwise, the approach recursively chooses a neuron to split into two branches (*branching*), resulting in two linear constraints. Then bounding is applied to both constraints and if both are satisfied the verification terminates, otherwise the other neurons are split recursively. When all neurons are split, the branch will contain only linear constraints, and thus the approach applies linear programming to compute the constraint and verify the branch. It is important to note that BaB approaches themselves are neither inherently complete nor incomplete. BaB is an algorithm for splitting problems into sub-problems and requires a solver to resolve the linear constraints. The completeness of the verification depends on the combination of BaB and the solver used. *Multi-Neuron Guided Branch-and-Bound (MN-BaB)* [59] is a state-of-the-art neural network verifier that builds on the tight multi-neuron constraints proposed in PRIMA [62] and leverages these constraints within a BaB framework to yield an efficient, GPU-based dual solver. Another state-of-the-art tool is $\alpha\beta$-CROWN [42, 43], a neural network verifier based on an efficient linear bound propagation framework and branch-and-bound. It can be accelerated efficiently on GPUs and can scale to relatively large convolutional networks (e.g. 10^7 parameters). It also supports a wide range of neural network architectures (e.g. CNN, ResNet and various activation functions).

Probabilistic Incomplete Verification approaches add random noise to models to smooth them, and then derive certified robustness for these smoothed models. This field is commonly referred to as Randomised Smoothing, given that these approaches provide probabilistic guarantees of robustness, and all current probabilistic verification techniques are tailored for smoothed models [63–68]. Given that our work focuses on deterministic approaches, here we only report the existence of this line of work without going into details.

Note that these existing verification approaches primarily focus on CV tasks, where images are seen as vectors in a continuous space and every point in the space corresponds to a valid image, while sentences in NLP form a discrete domain, making it challenging to apply traditional verification techniques effectively.

In this work, we use both an abstract interpretation-based incomplete verifier (ERAN [40]) and an SMT-based complete verifier (Marabou [22]) in order to demonstrate the effect that the choice of a verifier may bring and demonstrate common trends.

2.2 Geometric representations in DNN verification

Geometric representations form the backbone of many DNN verification techniques, enabling the encoding and manipulation of input and output bounds during analysis. Among these, hyper-rectangles, including ϵ-cubes, are the most widely used due to their simplicity and efficiency in over-approximating neural network behaviours [47, 48].These representations are computationally lightweight, making them highly scalable to large networks. However, they often produce loose approximations, particularly in deeper or more complex architectures, which can limit the precision of the verification results [48]. Other representations, such as zonotopes [53, 56, 69], offer tighter approximations and better capture the linear dependencies between neurons but at a higher computational cost. Polyhedra-based methods, as employed in tools like DeepPoly [53] and PRIMA [54], provide even more precise abstractions by considering multi-dimensional relationships between neurons. However, these methods trade off efficiency for precision, making them less scalable to large and deep networks. Ellipsoidal representations [55] are another class of geometric abstractions that provide compact and smooth bounds for neural

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

network outputs. These representations are particularly useful for capturing the effects of continuous transformations in hybrid systems and other control applications. However, operations such as intersection and propagation through non-linear layers can be computationally intensive, which limits their applicability in large-scale neural network verification tasks. The dominance of hyper-rectangles in the field stems from their balance of computational simplicity and generality. Nonetheless, ongoing research continues to explore how alternative shapes, hybrid approaches or adaptive representations might better meet the demands of increasingly complex neural network architectures.

2.3 Robust training

Verifying DNNs poses significant challenges if they are not appropriately trained. The fundamental issue lies in the failure of DNNs, including even sophisticated models, to meet essential verification properties, such as *robustness* [70]. To enhance robustness, various training methodologies have been proposed. It is noteworthy that, although robust training by *projected gradient descent* [20, 71, 72] predates verification, contemporary approaches are often related to, or derived from, the corresponding verification methods by optimising verification-inspired regularisation terms or injecting specific data augmentation during training. In practice, after robust training, the model usually achieves higher certified robustness and is more likely to satisfy the desired verification properties [70]. Thus, robust training is a strong complement to robustness verification approaches.

Robust training techniques can be classified into several large groups:

- data augmentation [73],
- adversarial training [20, 71] including property-driven training [74, 75],
- IBP training [48, 76] and other forms of certified training [77], or
- a combination thereof [70, 78].

Data augmentation involves the creation of synthetic examples through the application of diverse transformations or perturbations to the initial training data. These generated instances are then incorporated into the original dataset to enhance the training process. Adversarial training entails identifying worst-case examples at each epoch during the training phase and calculating an additional loss on these instances. State of the art adversarial training involves projected gradient descent algorithms such as FGSM [71] and PGD [20]. Certified training methods focus on providing mathematical guarantees about the model's behaviour within certain bounds. Among them, we can name IBP training [48, 76] techniques, which impose intervals or bounds on the predictions or activations of the model, ensuring that the model's output lies within a specific range with high confidence.

Note that all techniques mentioned above can be categorised based on whether they primarily *augment the data* (such as data augmentation) or *augment the loss function* (as seen in adversarial, IBP and certified training). Augmenting the data tends to be efficient, although it may not help against stronger adversarial attacks. Conversely, methods that manipulate the loss functions directly are more resistant to strong adversarial attacks but often come with higher computational costs. Ultimately, the choice between altering data or loss functions depends on the specific requirements of the application and the desired trade-offs between performance, computational complexity and robustness guarantees.

2.4 NLP robustness

There exists a substantial body of research dedicated to enhancing the adversarial robustness of NLP systems [79–85]. These efforts aim to mitigate the vulnerability of NLP models to adversarial attacks and improve their resilience in real-world scenarios [80, 81] and mostly employ data augmentation techniques [86, 87]. In NLP, we can distinguish perturbations based on three main criteria:

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

- where and how the perturbations occur,
- whether they are altered automatically using some defined rules (vs. generated by humans or large language model (LLMs)) and
- whether they are adversarial (as opposed to random).

In particular, perturbations can occur at the character, word or sentence level [88–90] and may involve deletion, insertion, swapping, flipping, substitution with synonyms, concatenation with characters or words, or insertion of numeric or alphanumeric characters [91–93]. For instance, in character level adversarial attacks, Belinkov et al. [94] introduce natural and synthetic noise to input data, while Gao et al. [95] and Li et al. [96] identify crucial words within a sentence and perturb them accordingly. For word level attacks, they can be categorised into gradient-based [91, 97], importance-based [98, 99] and replacement-based [100–102] strategies, based on the perturbation method employed. Moreover, Moradi et al. [103] introduce rule-based non-adversarial perturbations at both the character and word levels. Their method simulates various types of noise typically caused by spelling mistakes, typos and other similar errors. In sentence level adversarial attacks, some perturbations [104, 105] are created so that they do not impact the original label of the input and can be incorporated as a concatenation in the original text. In such scenarios, the expected behaviour from the model is to maintain the original output, and the attack can be deemed successful if the label/output of the model is altered. Additionally, non-rule-based sentence perturbations can be obtained through prompting LLMs [14, 21] to generate rephrasing of the inputs. By augmenting the training data with these perturbed examples, models are exposed to a more diverse range of linguistic variations and potential adversarial inputs. This helps the models to generalise better and become more robust to different types of adversarial attacks. To help with this task, the NLP community has gathered a dataset of adversarial attacks named AdvGLUE [106], which aims to be a principled and comprehensive benchmark for NLP robustness measurements.

In this work we employ a PGD-based adversarial training as the method to enhance the robustness and verifiability of our models against gradient-based adversarial attacks. For non-adversarial perturbations, we create rule-based perturbations at the character and word level as in Moradi et al. [103] and non-rule-based perturbations at the sentence level using PolyJuice [21] and Vicuna [14]. We thus cover most combinations of the three choices above (bypassing only human-generated adversarial attacks as there is no sufficient data to admit systematic evaluation which is important for this study).

2.5 Datasets and use cases used in NLP verification

Existing NLP verification datasets. Table 1 summarises the main features and tasks of the datasets used in NLP verification. Despite their diverse origins and applications, the datasets in the literature are usually binary or multi-class text classification problems. Furthermore, datasets can be sensitive to perturbations, i.e. perturbations can have non-trivial impact on label consistency. For example, Jia et al. [17] use IBP with the SNLI [119][3] dataset (see Tables 2 and 1) to show that word perturbations (e.g. 'good' to 'best') can change whether one sentence entails another. Some works such as Jia et al. [17] try to address this label consistency, while others do not.

Additionally, we find that the previous research on NLP verification does not utilise safety critical datasets (which strongly motivates the choice of datasets in alternative verification domains), with the exception of Du et al. [110] that use the Toxic Comment dataset [120]. Other papers do not provide detailed motivation as to why the dataset choices were made, however it could be due to the datasets being commonly used in NLP benchmarks (IMDB etc.).

[3] A semantic inference dataset that labels whether one sentence entails, contradicts or is neutral to another sentence.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 1. *Summary of the main features of the datasets used in NLP verification*

Dataset	Safety Critical	Category	Tasks	Size	Classes
IMDB [121]	×	Sentiment analysis	Document-level and sentence-level classification	25,000	2
SST [122]	×	Sentiment analysis	Sentiment classification, hierarchical sentiment classification, sentiment span detection	70,042	5
SST2 [122]	×	Sentiment analysis	Sentiment classification	70,042	2
YELP [123]	×	Sentiment analysis	Sentiment classification	570,771	2
Rotten Tomatoes Movie Review [124]	×	Sentiment analysis	Sentiment classification	48,869	3/4
Amazon [125]	×	Sentiment analysis	Sentiment classification, aspect-based sentiment analysis	34,686,770	5
SNLI [119]	×	Semantic inference	Natural language inference, semantic similarity	570,152	3
MNLI [126]	×	Semantic inference	Natural language inference, semantic similarity, generalisation	432,702	3
AGNews [127]	×	Text analysis	Text classification, sentiment classification	127,600	4
CogComp QC [128]	×	Text analysis	Question classification, semantic understanding	15,000	6/50
Toxic Comment [120]	√	Text analysis	Toxic comment classification, fine-grained toxicity analysis, bias analysis	18,560	6

2.5.1 Datasets proposed in this paper

In this paper, we focus on two existing datasets that model safety-critical scenarios. These two datasets have not previously been applied or explored in the context of NLP verification. Both are driven by real-world use cases of safety-critical NLP applications, i.e. applications for which law enforcement and safety demand formal guarantees of "good" DNN behaviour.

Chatbot Disclosure (R-U-A-Robot Dataset [129]). The first case study is motivated by new legislation which states that a chatbot must not mislead people about its artificial identity [10, 11]. Given that the regulatory landscape surrounding NLP models (particularly LLMs and generative AI) is rapidly evolving, similar legislation could be widespread in the future – with recent calls for the US Congress to formalise such disclosure requirements [130]. The *prohibition on deceptive conduct act* may apply to the outputs generated by NLP systems if used commercially [131], and at minimum a system must guarantee a truthful response when asked about its agency [129, 132]. Furthermore, the burden of this should be placed on the designers of NLP systems, and not on the consumers.

Our first safety critical case is the **R-U-A-Robot dataset** [129], a written English dataset consisting of 6800 variations on queries relating to the intent of 'Are you a robot?', such as 'I'm a man, what about you?'. The dataset was created via a context-free grammar template, crowd-sourcing and pre-existing data sources. It consists of 2,720 positive examples (where given the query, it is appropriate for the system to state its non-human identity), 3,400 negative examples and 680 'ambiguous-if-clarify' examples

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 2. *Summary of the main features of the existing NLP verification approaches. In bold are state-of-the-art methods*

Method	Datasets	NLP perturbations	Embeddings	Architectures (# of parameters)	Robust training	Verification algorithm	Verification characteristics
Ours	RUARobot, Medical	General purpose: char, word and sentence perturbations, ϵ-ball	Sentence: S-BERT, S-GPT	FFNN (10^4)	**PGD**-based loss function augmentation	**SMT**, **BaB**, Abstract interpretation	**Complete**, **Deterministic**
Jia et al. (2019) [17]	IMDB, SNLI	Word substitution	Word: GloVe	LSTM, CNN, BoW, Attention-based, (10^5)	**IBP**-based loss function augmentation	Abstract Interpretation IBP	Incomplete, **Deterministic**
Huang et al. (2019) [18]	AGNews, SST	Char and word substitution	Word: GloVe	CNN (10^5)	**IBP**-based loss function augmentation	Abstract Interpretation IBP	Incomplete, **Deterministic**
Welbl et al. (2020) [107]	SNLI, MNLI	Word deletion	Word: GloVe	Attention-based (10^5)	Data augmentation, random and beam search adversarial training, **IBP**-based	Abstract Interpretation IBP	Incomplete, **Deterministic**
Zhang et al. (2021) [19]	IMDB, SST, SST2	Word perturbations	Word: not specified	LSTM (10^5)	**IBP**-based loss function augmentation	Abstract Interpretation IBP	Incomplete, **Deterministic**
Wang et al. (2023) [108]	IMDB, YELP, SST2	Word substitution	Word: GloVe	CNN (10^5)	**IBP**-based: Embedding Interval Bound Constraint (EIBC) triplet loss	Abstract Interpretation IBP	Incomplete, **Deterministic**
Ko et al. (2019) [109]	CogComp QC	ϵ-ball	Word: not specified	RNN, LSTM (10^5)	–	Abstract Interpretation IBP	Incomplete, **Deterministic**
Shi et al. (2020) [15]	YELP, SST	ϵ-ball	Word: not specified	Transformer (10^6)	–	Abstract Interpretation IBP	Incomplete, **Deterministic**
Du et al. (2021) [110]	Rotten Tomatoes Movie Review, Toxic Comment	ϵ-ball	Word: GloVe	RNN, LSTM (10^5)	**Zonotope**-based loss function augmentation	Abstract Interpretation Zonotopes	Incomplete, **Deterministic**
Bonaert et al. (2021) [111]	SST, YELP	ϵ-ball	Word: not specified	Transformer (10^6)	–	Abstract Interpretation Zonotopes	Incomplete, **Deterministic**

(where it is unclear whether the system is required to state its identity). The dataset was created to promote transparency, which may be required when the user receives unsolicited phone calls from artificial systems. Given systems like Google Duplex [133], and the criticism it received for human-sounding outputs [134], it is also highly plausible for the user to be deceived regarding the outputs generated by other NLP-based systems [131]. Thus we choose this dataset to understand how to enforce such disclosure requirements. We collapse the positive and ambiguous examples into one label, following the principle of 'better be safe than sorry', i.e. prioritising a high recall system.

Medical Safety Dataset. Another scenario one might consider is that inappropriate outputs of NLP systems have the potential to cause harm to human users [13]. For example, a system may give a user false impressions of its 'expertise' and generate harmful advice in response to medically related user queries [7]. In practice, it may be desirable for the system to avoid answering such queries. Thus, we choose the **Medical safety dataset** [12], a dataset consisting of 2,917 risk-graded medical and non-medical queries (1,417 and 1,500 examples, respectively). The dataset was constructed via collecting questions posted on reddit, such as `r/AskDocs`. The medical queries have been labelled by experts and crowd annotators for both relevance and levels of risk (i.e. *non-serious, serious* to *critical*) following established World Economic Forum (WEF) risk levels designated for chatbots in healthcare [135]. We merge the medical queries of different risk-levels into one class, given the high scarcity of the latter two labels to create an in-domain/out-of-domain classification task for medical queries. Additionally, we consider only the medical queries that were labelled as such by expert medical practitioners. Thus, this dataset will facilitate discussion on how to guarantee a system recognises medical queries, in order to avoid generating medical output.

An additional benefit of these two datasets is that they are *distinct semantically*, i.e. the R-U-A-Robot dataset contains several semantically similar, but lexically different queries, while the medical safety dataset contains semantically diverse queries. For both datasets, we utilise the same data splits as given in the original papers and refer to the final binary labels as *positive* and *negative*. The *positive* label in the R-U-A-Robot dataset implies a sample where it is appropriate to disclose non-human identity, while in the medical safety dataset it implies an in-domain medical query.

2.6 Previous NLP verification approaches

Although DNN verification studies have predominantly focused on CV, there is a growing body of research exploring the verification of NLP. This research can be categorised into three main approaches: using IBP, zonotopes and randomised smoothing. Tables 2 and 3 show a comparison of these approaches. To the best of our knowledge, this paper is the first one to use an SMT-based verifier for this purpose and compare it with an abstract interpretation-based verifier on the same benchmarks.

NLP Verification via Interval Bound Propagation. The first technique successfully adopted from the CV domain for verifying NLP models was the IBP. IBP was used for both training and verification with the aim to minimise the upper bound on the maximum difference between the classification boundary and the input perturbation region. It was achieved by augmenting the loss function with a term that penalises large perturbations. Specifically, IBP incorporates interval bounds during the forward propagation phase, adding a regularisation term to the loss function that minimises the distance between the perturbed and unperturbed outputs. This facilitated the minimisation of the perturbation region in the last layer, ensuring it remained on one side of the classification boundary. As a result, the adversarial region becomes tighter and can be considered certifiably robust. Notably, Jia et al. [17] proposed certified robust models on word substitutions in text classification. The authors employed IBP to optimise the upper bound over perturbations, providing an upper bound over the discrete set of perturbations in the word vector space. Similarly, POPQORN [109] introduced robustness guarantees for RNN-based networks by handling the non-linear activation functions of complicated RNN structures (like LSTMs and GRUs) using linear bounds. Later, Shi et al. [15] developed a verification algorithm for transformers with self-attention layers. This algorithm provides a lower bound to ensure the probability of the

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 3. *Summary of the main features of the existing randomised smoothing approaches*

Method	Datasets	NLP perturbations	Embeddings	Architectures (# of parameters)	Robust training	Verification algorithm	Verification characteristics
Ye et al. (2020) [112]	IMDB, Amazon	Word substitution	Word: GloVe	Transformer (10^8)	Data augmentation	Randomised smoothing ($\alpha = 0.01, n = 5000$)	Incomplete, Probabilistic
Wang et al. (2021) [113]	IMDB, AGNews	Word substitution	Word: GloVe	LSTM (10^5)	Data augmentation	Differential privacy	Incomplete, Probabilistic
Zhao et al. (2022) [114]	AGNews, SST	Word substitution	Word: GloVe	Transformer (10^8)	Data augmentation and IBP-based	Randomised smoothing ($\alpha = 0.001, n = 30050$)	Incomplete, Probabilistic
Zeng et al. (2023) [115]	IMDB, YELP	Char and word substitution	Word: not specified	Transformer (10^8)	Data augmentation	Randomised smoothing ($\alpha = 0.05, n = 5000$)	Incomplete, Probabilistic
Ye et al. (2023) [116]	IMDB, SST2, YELP, AGNews	Word substitution	Word: not specified	Transformer (10^8)	Data augmentation	Randomised smoothing ($\alpha = 0.001, n = 9000$)	Incomplete, Probabilistic
Zhang et al. (2023) [117]	IMDB, Amazon, AGNews	Word perturbations	Word: GloVe	LSTM, Transformer (10^8)	Data augmentation	Randomised smoothing ($\alpha = 0.001, n = 20000$)	Incomplete, Probabilistic
Zhang et al. (2023) [118]	SST2, AGNews	Word perturbations	Word: not specified	Transformer (10^9)	–	Randomised smoothing ($\alpha = 0.05, n = 5000$)	Incomplete, Probabilistic

correct label remains consistently higher than that of the incorrect labels. Furthermore, Huang et al. [18] introduced a verification and verifiable training method with a tighter over-approximation in style of the Simplex algorithm [28]. To make the network verifiable, they defined the convex hull of all the original unperturbed inputs as a space of perturbations. By employing the IBP algorithm, they generated robustness bounds for each neural network layer. Later on, Welbl et al. [107] differentiated from the previous approaches by using IBP to address the under-sensitivity issue. They designed and formally verified the 'under-sensitivity specification' that a model should not become more confident as arbitrary subsets of input words are deleted. Recently, Zhang et al. [19] introduced Abstract Recursive Certification (ARC) to verify the robustness of LSTMs. ARC defines a set of programmatically perturbed string transformations to construct a perturbation space. By memorising the hidden states of strings in the perturbation space that share a common prefix, ARC can efficiently calculate an upper bound while avoiding redundant hidden state computations. Finally, Wang et al. [108] improved on the work of Jia et al. by introducing Embedding Interval Bound Constraint (EIBC). EIBC is a new loss that constraints the word embeddings in order to tighten the IBP bounds.

The strength of IBP-based methods is their efficiency and speed, while their main limitation is the bounds' looseness, further accentuated if the neural network is deep.

NLP Verification via Propagating Zonotopes. Another popular verification technique applied to various NLP models is based on propagating zonotopes, which produces tighter bounds then IBP methods. One notable contribution in this area is Cert-RNN [110], a robust certification framework for RNNs that overcomes the limitations of POPQORN. The framework maintains inter-variable correlation and accelerates the non-linearities of RNNs for practical uses. Cert-RNN utilised zonotopes [136] to encapsulate input perturbations and can verify the properties of the output zonotopes to determine certifiable robustness. This results in improved precision and tighter bounds, leading to a significant speedup compared to POPQORN. Analogously, Bonaert et al. [111] propose DeepT, a certification method for large transformers. It is specifically designed to verify the robustness of transformers against synonym replacement-based attacks. DeepT employs multi-norm zonotopes to achieve larger robustness radii in the certification and can work with networks much larger than Shi et al.

Methods that propagate zonotopes produce much tighter bounds than IBP-based methods, which can be used with deeper networks. However, they use geometric methods and do not take into account semantic considerations (e.g. do not use semantic perturbations).

NLP Verification via Randomised Smoothing. Randomised smoothing [137] is another technique for verifying the robustness of deep language models that has recently grown in popularity due to its scalability [112–118]. The idea is to leverage randomness during inference to create a smoothed classifier that is more robust to small perturbations in the input. This technique can also be used to give certified guarantees against adversarial perturbations within a certain radius. Generally, randomised smoothing begins by training a regular neural network on a given dataset. During the inference phase, to classify a new sample, noise is randomly sampled from the predetermined distribution multiple times. These instances of noise are then injected into the input, resulting in noisy samples. Subsequently, the base classifier generates predictions for each of these noisy samples. The final prediction is determined by the class with the highest frequency of predictions, thereby shaping the smoothed classifier. To certify the robustness of the smoothed classifier against adversarial perturbations within a specific radius centred around the input, randomised smoothing calculates the likelihood of agreement between the base classifier and the smoothed classifier when noise is introduced to the input. If this likelihood exceeds a certain threshold, it indicates the certified robustness of the smoothed classifier within the radius around the input.

The main advantage of randomised smoothing-based methods is their scalability, indeed recent approaches are tested on larger transformer such as BERT and Alpaca. However, their main issue is that they are probabilistic approaches, meaning they give certifications up to a certain probability. In this work, we focus on deterministic approaches, hence we only report these works in Table 3 for completeness without delving deeper into each paper here. All randomised smoothing-based approaches use data augmentation obtained by semantic perturbations.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

To systematically compare the existing body of research, we distil an "*NLP verification pipeline*" that is common across many related papers. This pipeline is outlined diagrammatically in Figure 2, while Tables 2 and 3 provide a detailed breakdown, with columns corresponding to each stage of the pipeline. It proceeds in stages:

1. **Given an NLP dataset, generate semantic perturbations on sentences that it contains.** The semantic perturbations can be of different kinds: character, word or sentence level. IBP and randomised smoothing use word and character perturbations, abstract interpretation papers usually do not use any semantic perturbations. Tables 2 and 3 give the exact mapping of perturbation methods to papers. Our method allows to use all existing semantic perturbations, in particular, we implement character and word-level perturbations as in Moradi et al. [103], sentence-level perturbations with PolyJuice [21] and Vicuna.
2. **Embed the semantic perturbations into continuous spaces.** The cited papers use the word embeddings GloVe [102], we use the sentence embeddings S-BERT and S-GPT.
3. **Working on the embedding space, use geometric or semantic perturbations to define geometric or semantic subspaces around perturbed sentences.** In IBP papers, semantic subspaces are defined as "bounds" derived from admissible semantic perturbations. In abstract interpretation papers, geometric subspaces are given by ϵ-cubes and ϵ-balls around each embedded sentence. Our paper generalises the notion of ϵ-cubes by defining "hyper-rectangles" on sets of semantic perturbations. The hyper-rectangles generalise ϵ-cubes both geometrically and semantically, by allowing to analyse subspaces that are drawn around several (embedded) semantic perturbations of the same sentence. We could adapt our methods to work with hyper-ellipses and thus directly generalise ϵ-balls (the difference boils down to using ℓ_2 norm instead of ℓ_∞ when computing geometric proximity of points); however, hyper-rectangles are more efficient to compute, which determined our choice of shapes in this paper.
4. **Use the geometric/semantic subspaces to train a classifier to be robust to change of label within the given subspaces.** We generally call such training either *robust training* or *semantically robust training*, depending on whether the subspaces it uses are geometric or semantic. A custom semantically robust training algorithm is used in IBP papers, while abstract interpretation papers usually skip this step or use (adversarial) robust training. See Tables 2 and 3 for further details. In this paper, we adapt the famous PGD algorithm [20] that was initially defined for geometric subspaces (ϵ-balls) to work with semantic subspaces (hyper-rectangles) to obtain a novel semantic training algorithm.
5. **Use the geometric/semantic subspaces to verify the classifier's behaviour within those subspaces.** The papers [17–19, 107, 108] use IBP algorithms and the papers [15, 109–111] use abstract interpretation; in both cases, it is incomplete and deterministic verification. See 'Verification algorithm' and 'Verification characteristics' columns of Tables 2 and 3. We use SMT-based tool Marabou (complete and deterministic) and abstract-interpretation tool ERAN (incomplete and deterministic).

Tables 2 and 3 summarise differences and similarities of the above NLP verification approaches against ours. To the best of our knowledge, we are the first to use SMT-based complete methods in NLP verification and we show how they achieve higher verifiability than abstract interpretation verification approaches (ERAN and CORA) or IBP and BaB ($\alpha\beta$-CROWN), thanks to the increased precision of the ReLUplex algorithm that underlies Marabou.

Furthermore, our study is the first to demonstrate that the construction of semantic subspaces can happen independently of the choice of the training and verification algorithms. Likewise, although training and verification build upon the defined (semantic) subspaces, the actual choice of the training and verification algorithms can be made independently of the method used to define the semantic subspaces. This separation, and the general modularity of our approach, facilitates a comprehensive examination and comparison of the two key components involved in any NLP verification process:

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

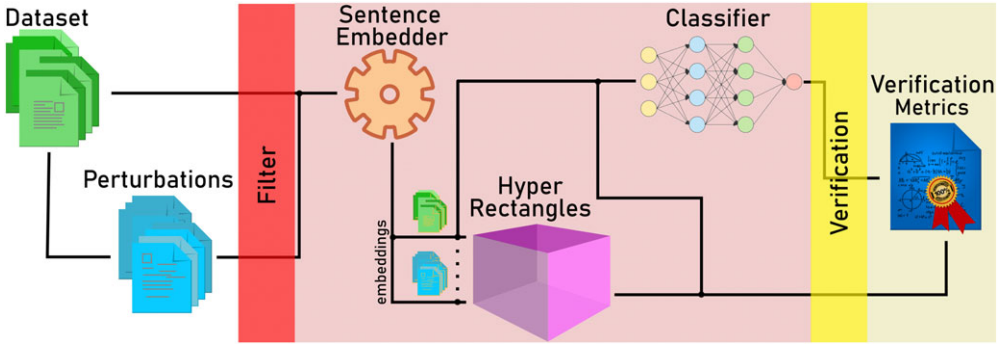


Figure 2. *Visualisation of the NLP verification pipeline followed in our approach.*

- effects of the *verifiability-generalisability trade-off* for verification with geometric and semantic subspaces;
- relation between the volume/shape of semantic subspaces and verifiability of neural networks obtained via semantic training with these subspaces.

These two aspects have not been considered in the literature before.

3 The parametric NLP verification pipeline

This section presents a *parametric NLP verification* pipeline, shown in Figure 2 diagrammatically. We call it "parametric" because each component within the pipeline is defined independently of the others and can be taken as a parameter when studying other components. The parametric nature of the pipeline allows for the seamless integration of state-of-the-art methods at every stage, and for more sophisticated experiments with those methods. The following section provides a detailed exposition of the methodological choices made at each step of the pipeline.

3.1 Semantic perturbations

As discussed in Section 2.6, we require semantic perturbations for creating semantic subspaces. To do so, we consider three *kinds* of perturbations—i.e. character, word and sentence level. This systematically accounts for different variations of the samples.

Character and word-level perturbations are created via a rule-based method proposed by Moradi et al. [103] to simulate different kinds of noise one could expect from spelling mistakes, typos, etc. These perturbations are non-adversarial and can be generated automatically. Moradi et al. [103] found that NLP models are sensitive to such small errors, while in practice this should not be the case. Character-level perturbations *types* include randomly inserting, deleting, replacing, swapping or repeating a character of the data sample. At the character level, we do not apply letter case changing, given it does not change the sentence-level representation of the sample. Nor do we apply perturbations to commonly misspelled words, given only a small percentage of the most commonly misspelled words occur in our datasets. Perturbations types at the word level include randomly repeating or deleting a word, changing the ordering of the words, the verb tense, singular verbs to plural verbs or adding negation to the data sample. At the word level, we omit replacement with synonyms, as this is accounted for via sentence rephrasing. Negation is not done on the medical safety dataset, as it creates label ambiguities (e.g. 'pain when straightening knee' → 'no pain when straightening knee'), as well as singular plural tense and verb tense, given human annotators would experience difficulties with this task (e.g. rephrase the following in plural/ with changed tense – 'peritonsillar abscess drainage aftercare.. please help'). Note that the

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 4. *Character-level perturbations: their types and examples of how each type acts on a given sentence from the R-U-A-robot dataset [129]. Perturbations are selected from random words that have 3 or more characters, first and last characters of a word are never perturbed*

Method	Description	Altered sentence (*Are you a robot?*)
Insertion	A character is randomly selected and inserted in a random position.	*Are yovu a robot?*
Deletion	A character is randomly selected and deleted.	*Are you a robt?*
Replacement	A character is randomly selected and replaced by an adjacent character on the keyboard.	*Are you a ronot?*
Swapping	A character is randomly selected and swapped with the adjacent right or left character in the word.	*Are you a rboot?*
Repetition	A character in a random position is selected and duplicated.	*Arre you a robot?*

Table 5. *Word-level perturbations: their types and examples of how each type acts on a given sentence from the R-U-A-robot dataset [129].*

Method	Description	Altered sentence (*Can u tell me if you are a chatbot?*)
Deletion	Randomly selects a word and removes it.	*Can u tell if you are a chatbot?*
Repetition	Randomly selects a word and duplicates it.	*Can can u tell me if you are a chatbot?*
Negation	Identifies verbs then flips them (negative/positive).	*Can u tell me if you are not a chatbot?*
Singular/ plural verbs	Changes verbs to singular form, and conversely.	*Can u tell me if you is a chatbot?*
Word order	Randomly selects consecutive words and changes the order in which they appear.	*Can u tell me if you are chatbot a?*
Verb tense	Converts present simple or continuous verbs to their corresponding past simple or continuous form.	*Can u tell me if you were a chatbot?*

Medical dataset contains several sentences without a verb (like the one above) for which it is impossible to pluralise or change the tense of the verb.

Further examples of character and word rule-based perturbations can be found in Tables 4 and 5.

Sentence-level perturbations. We experiment with two types of sentence-level perturbations, particularly due to the complicated nature of the medical queries (e.g. it is non-trivial to rephrase queries such as this – 'peritonsillar abscess drainage aftercare.. please help'). We do so by either using Polyjuice [21] or `vicuna-13b`.[4] Polyjuice is a general-purpose counterfactual generator that allows for control over perturbation types and locations, trained by fine-tuning GPT-2 on multiple datasets of paired sentences. Vicuna is a state-of-the-art open source chatbot trained by fine-tuning LLaMA [138] on user-shared conversations collected from ShareGPT.[5] For Vicuna, we use the following prompt to generate variations on our data samples '*Rephrase this sentence 5 times: "[Example]"*.' For example, from the sentence

[4] Using the following API: https://replicate.com/replicate/vicuna-13b/api.
[5] https://sharegpt.com/

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

"How long will I be contagious?", we can obtain "How many years will I be contagious?" or "Will I be contagious for long?" and so on.

We will use notation $\mathcal{P}$ to refer to a perturbation algorithm abstractly.

Semantic similarity of perturbations. In later sections we will make an assumption that the perturbations that we use produce sentences that are semantically similar to the originals. However, precisely defining or measuring semantic similarity is a challenge in its own right, as semantic meaning of sentences can be subjective, context-dependent, which makes evaluating their similarity intractable. Nevertheless, Subsection 5.5.2 will discuss and use several metrics for calculating semantic similarity of sentences, modulo some simplifying assumptions.

3.2 NLP embeddings

The next component of the pipeline is the embeddings. Embeddings play a crucial role in NLP verification as they map textual data into continuous vector spaces, in a way that should capture semantic relationships and contextual information.

Given the set of all strings, $\mathbb{S}$, an NLP dataset $\mathcal{Y} \subset \mathbb{S}$ is a set of sentences $s_1, \ldots, s_q$ written in natural language. The embedding E is a function that maps a string in $\mathbb{S}$ to a vector in $\mathbb{R}^m$. The vector space $\mathbb{R}^m$ is called *the embedding space*. Ideally, E should reflect the semantic similarities between sentences in $\mathbb{S}$, i.e. the more semantically similar two sentences s_i and s_j are, the closer the distance between $E(s_i)$ and $E(s_j)$ should be in $\mathbb{R}^m$. Of course, defining semantic similarity in precise terms may not be tractable (the number of unseen sentences may be infinite, the similarity may be subjective and/or depend on the context). This is why, the state-of-the-art NLP relies on machine learning methods to capture the notion of semantic similarity approximately.

Currently, the most common approach to obtain an embedding function E is by training *transformers* [139, 140]. Transformers are a type of DNNs that can be trained to map sequential data into real vector spaces and are capable of handling variable-length input sequences. They can also be used for other tasks, such as classification or sentence generation, but in those cases, too, training happens at the level of embedding spaces. In this work, a transformer is trained as a function $E{:}\mathbb{S} \to \mathbb{R}^m$ for some given m. The key feature of the transformer is the "self-attention mechanism", which allows the network to weigh the importance of different elements in the input sequence when making predictions, rather than relying solely on the order of elements in the sequence. This makes them good at learning to associate semantically similar words or sentences. In this work, we initially use Sentence-BERT [140] and later add Sentence-GPT [141] to embed sentences. Unfortunately, the relation between the embedding space and the NLP dataset is not bijective: i.e. each sentence is mapped into the embedding space, but not every point in the embedding space has a corresponding sentence. This problem is well known in NLP literature [142] and, as shown in this paper, is one of the reasons why verification of NLP is tricky. Given an NLP dataset $\mathcal{Y}$ that should be classified into n classes, the standard approach is to construct a function $N{:}\mathbb{R}^m \to \mathbb{R}^n$ that maps the embedded inputs to the classes. In order to do that, a domain-specific classifier N is trained on the embeddings $E(\mathcal{Y})$ and the final system will then be the composition of the two subsystems, i.e. $N \circ E$.

3.3 Geometric analysis of embedding spaces

In the recent years, the study of manifold subspaces has gained significant attention in the context of machine learning verification [55], where the geometry of data regions plays an important role. In this section, we formally define most common subspaces used in verification: convex sets, convex hulls, zonotopes and hyper-rectangles (also known as multi-dimensional intervals), following closely [55].

Definition 1. (Convex Set). *A set $\mathbb{Z} \subseteq \mathbb{R}^m$ is said to be convex if, for any two points $x_1, x_2 \in \mathbb{Z}$, the line segment joining them is entirely contained within $\mathbb{Z}$. Formally, this means that for all $x_1, x_2 \in \mathbb{Z}$ and $\lambda \in [0, 1]$, the points*

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 6. *Construction complexity for different geometric shapes, where m is the number of dimensions and p is the number of points or generators*

Shape	Construction Complexity (Big-O)
Convex Hull	$O(p^{m/2})$
Zonotope	$O(m \cdot p)$
Interval	$O(m)$

$$\lambda x_1 + (1 - \lambda)x_2 \in \mathbb{Z}.$$

In other words, a set is convex if, for any pair of points in the set, the entire segment connecting them lies within the set.

The convex hull of a set is the smallest convex set that contains all the points of the set. It can be seen as the "tightest" boundary enclosing the points.

Definition 2. (Convex Hull). *The convex hull of a set* $\mathbb{Z} \subseteq \mathbb{R}^m$*, denoted by conv*$(\mathbb{Z})$*, is the smallest convex set containing* $\mathbb{Z}$*. Formally, it can be defined as:*

$$conv(\mathbb{Z}) := \left\{ \sum_{i=1}^{p} \lambda_i x_i \;\middle|\; x_i \in \mathbb{Z},\ \lambda_i \geq 0,\ \sum_{i=1}^{p} \lambda_i = 1,\ p \in \mathbb{N} \right\}.$$

In other words, conv$(\mathbb{Z})$ consists of all finite convex combinations of points in $\mathbb{Z}$. The construction of the convex hull has a complexity of $O(p^{m/2})$, where p is the number of points and m is the number of dimensions.

A zonotope is a geometric shape formed by the Minkowski sum of line segments.

Definition 3. (Zonotope). *Given a center* $c \in \mathbb{R}^m$ *and generators* $g_1, \ldots, g_p$*, a zonotope is*

$$\mathcal{Z} := \left\{ c + \sum_{i=1}^{p} \lambda_i g_i \;\middle|\; \lambda_i \in [-1, 1], \forall i \in [1, \ldots, p] \right\}$$

Zonotopes are computationally more efficient than convex hulls, with a construction complexity of $O(m \cdot p)$, where m is the dimensionality and p is the number of generators.

Finally, an interval is a simple shape defined by lower and upper bounds for each dimension, and it is equivalent to a multi-dimensional rectangle (or hyper-rectangle). Intervals are easy to construct with a complexity of $O(m)$, where m is the dimensionality, and are often used in verification.

Definition 4. (Interval (aka Hyper-Rectangle)). *Given a lower and upper bound* $\underline{x}, \overline{x} \in \mathbb{R}^m$ *such that* $\underline{x}_{(i)} \leq \overline{x}_{(i)} \forall i \in 1, \ldots, m$*, a multi-dimensional interval* $\mathcal{I} \subset \mathbb{R}^m$ *is*

$$\mathcal{I} := \left\{ x \in \mathbb{R}^m \;\middle|\; \underline{x}_{(i)} \leq x_{(i)} \leq \overline{x}_{(i)}, \forall i \in [1, \ldots, m] \right\}$$

Table 6 summarises the construction complexities of these different shapes. Ideally, convex hulls would be the preferred choice due to their precise and detailed representations of subspaces. However, their computational complexity renders them infeasible in high dimensions. Zonotopes provide a promising alternative, as they are more precise than hyper-rectangles while remaining computationally tractable. Despite their theoretical compatibility with complete verifiers, practical limitations arise because most state-of-the-art verifiers do not support zonotopes. Hyper-rectangles, or intervals, are the simplest to construct and are supported by all verifiers, making them the default choice in many verification pipelines.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

3.4 Working with embedding spaces: our approach

We now formally define geometric and semantic subspaces of the embedding space. Our goal is to define subspaces on the embedding space $\mathbb{R}^m$ by using an effective algorithmic procedure. We will use notation $\mathcal{S}$ to refer to a subspace of the embedding space. Recall that an hyper-rectangle of dimension m is a list of points $(a_1, b_1), \ldots, (a_m, b_m)$ such that a point $x \in \mathbb{R}^m$ is a member if for every dimension j we have $a_j \leq x_j \leq b_j$.

We start with an observation that, given an NLP dataset $\mathcal{Y}$ that contains a finite set of sentences $s_1, \ldots, s_q$ belonging to the same class, and an embedding function $E{:}\mathbb{S} \to \mathbb{R}^m$, we can define an *embedding matrix* $\mathcal{X} \in \mathbb{R}^{q \times m}$, where each row j is given by $E(s_j)$. We will use the notation x_i to refer to the ith element of the vector x, and $\mathcal{X}^{ij}$ to refer to the element in the ith row and jth column of $\mathcal{X}$. Treating embedded sentences as matrices, rather than as points in the real vector space, makes many computations easier. We can therefore define a *hyper-rectangle* for $\mathcal{X}$ as follows.

Definition 5. (Hyper-rectangle for an Embedding Matrix). *Given an embedding matrix $\mathcal{X} \in \mathbb{R}^{q \times m}$, the m-dimensional hyper-rectangle for $\mathcal{X}$ is defined as*

$$\mathbb{H}(\mathcal{X}) := \{(\min_{i=0}^{q} \mathcal{X}^{ij},\ \max_{i=0}^{q} \mathcal{X}^{ij}) \mid j \in [1, \ldots, m]\}$$

Therefore given an embedding function $E{:}\mathbb{S} \to \mathbb{R}^m$, and a set of sentences $\mathcal{Y} = \{s_1, \ldots, s_q\}$, we can form a subspace $\mathbb{H}(E(\mathcal{Y}))$ by constructing the embedding matrix, as described above, and forming the corresponding hyper-rectangle. To simplify the notation, we will omit the application of E and from here on simply write $\mathbb{H}(\mathcal{Y})$.

The next example shows how the above definitions generalise the commonly known definition of the ϵ-cube.

Example 1. *(ϵ-cube* and *ϵ-ball). One of the most popular terms used in robust training [71] and verification [70] literature is the ϵ-ball. It is defined as follows. Given an embedded input $\hat{x}$, a constant $\epsilon \in \mathbb{R}$, and a distance function (ℓ-norm) $||-||$, the ϵ-ball around $\hat{x}$ of radius ϵ is defined as*

$$\mathbb{B}(\hat{x}, \epsilon) := \{x \in \mathbb{R}^m {:} ||\hat{x} - x|| \leq \epsilon\}.$$

In practice, it is common to use the ℓ_∞ norm, which results in the ϵ-ball actually being a hyper-rectangle, also called ϵ-cube, where $(a_j, b_j) = (\hat{x}_j - \epsilon, \hat{x}_j + \epsilon)$. Therefore, our construction $\mathbb{H}$ is a strict generalisation of ϵ-cubes. We will therefore use the notation $\mathbb{H}(\mathcal{Y}, \epsilon) = \bigcup_{s \in \mathcal{Y}} \mathbb{B}(E(s), \epsilon)$ to refer to the set of ϵ-cubes around every sentence in the dataset.

Of course, as we have already discussed in the introduction and Figure 1, hyper-rectangles are not very precise, geometrically. A more precise shape would be a *convex hull* around q given points in the embedding space. Indeed literature has some definitions of convex hulls [143–145]. However, none of them is suitable as they are computationally too expensive due to the time complexity of $O(q^{m/2})$ where q is the number of inputs and m is the number of dimensions [143]. Approaches that use under-approximations to speed up the algorithms [144, 145] do not work well in NLP scenarios, as under-approximated subspaces are so small that they contain near zero sentence embeddings.

3.4.1 Exclusion of unwanted sentences via shrinking

Another concern is that the generated hyper-rectangles may contain sentences from a different class. This would make it unsuitable for verification. In order to exclude all samples from the wrong class, we define a shrinking algorithm $SH(\mathcal{X}, \mathcal{Y}, c)$ that calculates a new subspace that is a subset of the original hyper-rectangle around $\mathcal{X}$, that only contains embeddings of sentences in $\mathcal{Y}$ that are of class c. Of course, to ensure this, the algorithm may have to exclude some sentences of class c. The second graph of Figure 3 gives a visual intuition of how this is done.

Formally, for each sentence s in $\mathcal{Y}$ that is not of class c, the algorithm performs the following procedure. If $E(s)$ lies in the current hyper-rectangle $(a_1, b_1), \ldots, (a_m, b_m)$, then for each dimension $j \in [1, \ldots, m]$ we compute the distance whether $E(s)_j$ is closer to a_j or b_j. Without loss of generality,

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

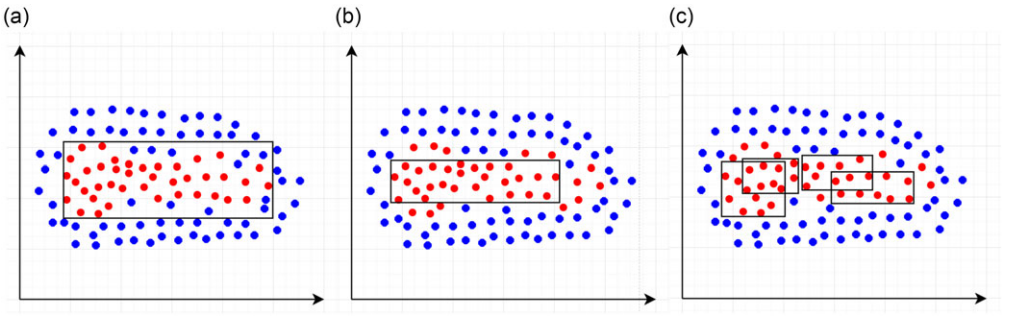


Figure 3. *An example of hyper-rectangle drawn around all points of the same class (a), shrunk hyper-rectangle $\mathbb{H}_{sh}$ that is obtained by excluding all points from the opposite class (b) and clustered hyper-rectangles (c) in 2-dimensions. The red dots represent sentences in the embedding space of one class, while the blue dots are embedded sentences that do not belong to that class.*

assume a_j is closer. We then compute the number of sentences of class c that would be excluded by replacing a_j with $E(s)_j + \delta$ in the hyper-rectangle where δ is a small positive number (we use e^{-100}). This gives us a penalty for each dimension j, and we exclude s by updating the hyper-rectangle in the dimension that minimises this penalty. The idea is to shrink the hyper-rectangle in the dimensions that exclude as few embedded sentences from the desired class c as possible.[6]

3.4.2 Exclusion of unwanted sentences via clustering

An alternative approach to excluding unwanted sentences is to split the dataset up by clustering semantically similar sentences in the embedding space and then compute the hyper-rectangles around each cluster individually, as shown in the last graph of Figure 3. In this paper, we will use the k-means algorithm for clustering. We will use the notation $CL(\mathcal{Y}, k)$ to refer to the k-clusters formed by applying it to dataset $\mathcal{Y}$. While in our experiments we have found this is often sufficient to exclude unwanted sentences, it is not guaranteed to do so. Therefore, this method is combined with the shrinking algorithm in our experiments.

3.4.3 Eigenspace rotation

A final alternative and computationally efficient way of reducing the likelihood that the hyper-rectangles will contain embedded sentences of an unwanted class is to rotate them to better align to the distribution of the embedded sentences of the desired class in the embedding space. This motivates us to introduce the Eigenspace rotation.

To construct the tightest possible hyper-rectangle, we define a specific method of eigenspace rotation. As shown in Figure 1 (C and D), our approach is to calculate a rotation matrix A such that the rotated matrix $\mathcal{X}_{\text{rot}} = \mathcal{X}A$ is better aligned with the axes than $\mathcal{X}$, and therefore $\mathbb{H}(\mathcal{X}_{\text{rot}})$ has a smaller volume. By a slight abuse of terminology, we will refer to $\mathbb{H}(\mathcal{X}_{\text{rot}})$ as the *rotated hyper-rectangle*, even though strictly speaking, we are rotating the data, not the hyper-rectangle itself. In order to calculate the rotation matrix A, we use singular value decomposition [146]. The singular value decomposition of $\mathcal{X}$ is defined as $\mathcal{X} = U\Sigma V^*$, where U is a matrix of left-singular vectors, Σ is a matrix of singular values and V^* is a matrix of right-singular vectors and $\cdot^*$ denotes the conjugate transpose. Intuitively, the right-singular

[6] Note that this algorithm shrinks exactly one dimension by a minimal amount to exclude the unwanted embedded sentence. This choice keeps the algorithm fast while guaranteeing the subspace to retain the highest number of wanted inputs. However, it is not necessarily the best choice for verification: there might be cases where perturbations of the unwanted input are left inside after shrinking and, if the network classifies them correctly, the subspace can never be verified. For large subspaces, our algorithm might render verification unachievable and more clever algorithms should be explored and discussed.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

vectors V^* describe the directions in which $\mathcal{X}$ exhibits the most variance. The main idea behind the definition of rotation is to align these directions of maximum variance with the standard canonical basis vectors. Formally, using V^*, we can compute the rotation (or change-of-basis) matrix A that rotates the right-singular vectors onto the canonical standard basis vectors I, where I is the identity matrix. To do this, we observe that $V^*A = I$ implies $V^* = IA^{-1}$, which implies $V^{-1} = A^{-1}$, and thus $V = A$. We thus obtain $\mathcal{X}_{rot} = \mathcal{X}A$ as desired. All hyper-rectangles constructed in this paper are rotated.

3.4.4 Geometric and semantic subspaces

We now apply the abstract definition of a subspace of an embedding space to concrete NLP verification scenarios. Once we know how to define subspaces for a selection of points in the embedding space, the choice remains how to choose those points. The first option is to use ϵ-cubes around given embedded points, as Example 1 defines. Since this construction does not involve any knowledge about the semantics of sentences, we will call the resulting subspaces *geometric subspaces*. The second choice is to apply semantic perturbations to a sentence in $\mathcal{Y}$, embed the resulting sentences, and then define a subspace around them. We will call the subspaces obtained by this method *semantic perturbation subspaces*, or just *semantic subspaces* for short.

We will finish this section with defining semantic subspaces formally. We will use $\mathcal{P}_t(s)$ to denote an algorithm for generating sentence perturbations of type t, applied to an input sentence s in a random position. In the later sections, we will use t to refer to the different types of perturbations illustrated in Tables 4 and 5, e.g. character-level insertion, deletion, replacement. Intuitively, given a single sentence we want to generate a set of semantically similar perturbations and then construct a hyper-rectangle around them, as described in Definition 5.

This motivates the following definitions. Given a sentence s, a number b and a type t, the set $\mathcal{A}_t^b(s) = \{\mathcal{P}_t(s) \mid i \in [1, b]\}$ is the set of b semantic perturbations of type t generated from s. We will use the notation $\mathcal{A}_t^b(\mathcal{Y}) = \bigcup_{s\in\mathcal{Y}} \mathcal{A}_t^b(s)$ to denote the new dataset generated by creating b semantic perturbations of type t around each sentence.

Definition 6. (Semantic Subspace for a Sentence). *Given an embedding function $E{:}\mathbb{S} \to \mathbb{R}^m$, the semantic subspace for a sentence s is the subspace $\mathbb{H}(\{s\} \cup \mathcal{A}_t^b(s))$. We will refer to a set of such semantic hyper-rectangles over an entire dataset $\mathcal{Y}$ as $\mathbb{H}_t^b(\mathcal{Y}) = \bigcup_{s\in\mathcal{Y}} \mathbb{H}(\{s\} \cup \mathcal{A}_t^b(s))$.*

Example 2. (Construction of Semantic Subspaces). *To illustrate this construction, let us consider the sentence s: "Can u tell me if you are a chatbot?". This sentence is one of* 3400 *original sentences of the positive class in the dataset. From this single sentence, we can create six new sentences using the word-level perturbations from Table 5 to form $\mathcal{A}_{word}^6(s)$. Once the seven sentences are embedded into the vector space, they form the hyper-rectangle $\mathbb{H}(\{s\} \cup \mathcal{A}_{word}^6(s))$. By repeating this construction for the remaining* 3399 *sentences, we obtain the set of hyper-rectangles $\mathbb{H}_{word}(\mathcal{Y})$ for the dataset.*

Given a sentence s, we embed each sentence in $\mathcal{A}_t^b(s) = \{s_1, \ldots, s_b\}$ into $\mathbb{R}^m$ obtaining vectors $\mathcal{V}_t^b(s) = \{v, v_1, \ldots, v_b\}$ where $v_j = E(s_j)$.

3.4.5 Measuring the quality of sentence embeddings

One of our implicit assumptions in the previous sections is that the embedding function E maps pairs of semantically similar sentences to nearby points in the embedding space. In Section 5.5.2, we will evaluate the accuracy of this assumption using *cosine similarity*. This metric measures how similar two vectors are in a multi-dimensional space by calculating the cosine of the angle between them:

$$\mathrm{CoS}(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}$$

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

where $\cdot$ is the dot product and $\|v\| = \sqrt{v \cdot v}$. The resulting value ranges from 0 to 1. A value of 1 indicates that the vectors are parallel (highest similarity), while 0 means that the vectors are orthogonal (no similarity).

3.5 Training

As outlined in Section 2.3, robust training is essential for bolstering the robustness of DNNs; without it, their verifiability would be significantly diminished. This study employs two robust training methods, namely data augmentation and a custom PGD adversarial training, with the goal of discerning the factors contributing to the success of robust training and compare the effectiveness of these methods.

Data Augmentation. In this training method, we statically generate semantic perturbations at the character, word and sentence levels before training, which are then added to the dataset. The network is subsequently trained on this augmented dataset using the standard stochastic gradient descent algorithm.

Adversarial Training. In this training method, the traditional projected gradient descent (PGD) algorithm [20], is defined as follows. Given a loss function $\mathcal{L}$, a step size $\gamma \in \mathbb{R}$ and a starting point $\hat{x}_0$ then the output of the PGD algorithm $x(l)$ after l iterations is defined as:

$$x(0) = \hat{x}_0$$
$$x(t+1) = \text{proj}_{\mathcal{S}}\Big[x(t) + \gamma \cdot \text{sign}(\nabla_{x(t)}\mathcal{L}(x(t), y))\Big]$$

where $\text{proj}_{\mathcal{S}}$ is the projection back into the desired subspace $\mathcal{S}$. In its standard formulation, the subspace $\mathcal{S}$ is often an ϵ-ball (for some chosen ϵ).

In this work, we modify the algorithm to work with custom-defined hyper-rectangles as the subspace. The primary distinction between our customised PGD algorithm and the standard version lies in the definition of the step size. In the conventional algorithm, the step size is represented by a scalar $\gamma \in \mathbb{R}$ therefore representing a uniform step size in every dimension. In our case the width of $\mathbb{H}$ in each dimension may vary greatly, therefore we transforms γ into a vector in $\mathbb{R}^m$, allowing the step size to vary by dimension. Note that the dot $\cdot$ between γ and $sign(\nabla)$ becomes an element-wise multiplication. The resulting customised PGD training seeks to identify the worst perturbations within the custom-defined subspace and trains the given neural network to classify those perturbations correctly, in order to make the network robust to adversarial inputs in the chosen subspace.

3.6 Choice of verification algorithm

As stated earlier, our approach in this study involves the utilisation of cutting-edge tools for DNN verification. Initially, we employ ERAN [69], a state-of-the-art abstract interpretation-based method. This choice is made over IBP due to its ability to yield tighter bounds. Subsequently, we conduct comparisons and integrate Marabou [22], a state-of-the-art complete verifier. This enables us to attain the highest verification percentage, maximising the tightness of the bounds. Additionally, we incorporate $\alpha\beta$-CROWN [42, 43], the best-performing verifier in the International Verification of Neural Networks Competition (VNN-COMP) 2024 competition, known for its efficiency in linear bound propagation and branch-and-bound techniques. Moreover, we utilise CORA [55], an abstract interpretation-based verifier that supports zonotope-based verification, allowing us to compare hyper-rectangles and zonotopes in our verification experiments. We will use notation $\mathcal{V}$ to refer to a verifier abstractly.

4. Characterisation of verifiable subspaces

In this Section, we provide key results in support of **Contribution 1** formulated in the introduction:

- We start with introducing the metric of *generalisability* of (verified) subspaces and set-up some baseline experiments.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

- We introduce the problem of the *verifiability-generalisability trade-off* in the context of geometric subspaces.
- We show that, compared to geometric subspaces, the use of semantic subspaces helps to find a better balance between generalisability and verifiability.
- Finally, we show that adversarial training based on semantic subspaces results in DNNs that are both more verifiable and more generalisable than those obtained with other forms of robust training.

4.1 Metrics for understanding the properties of embedding spaces

Let us start with recalling the existing standard metrics used in DNN verification. Recall that we are given an NLP dataset $\mathcal{Y} = \{s_1, \ldots, s_q\}$, moreover we assume that each s_i is assigned a *correct class* from $C = \{c_1, \ldots, c_n\}$. We restrict to the case of binary classification in this paper for simplicity, so we will assume $C = \{c_1, c_2\}$. Furthermore, we are given an embedding function $E{:}\mathbb{S} \to \mathbb{R}^m$, and a network $N{:}\mathbb{R}^m \to \mathbb{R}^n$. Usually n corresponds to the number of classes, and thus in case of binary classification, we have $N{:}\mathbb{R}^m \to \mathbb{R}^2$. An embedded sentence $s \in \mathcal{Y}$ is *classified* as class c if the value of c in $N(E(s))$ is higher than all other classes.

Accuracy. The most popular metric for measuring the performance of the network is the *accuracy* of N, which is measured as a percentage of sentences in $\mathcal{Y}$ that are assigned to a correct class by N. Note that this metric only checks a finite number of points in $\mathbb{R}^m$ given by the dataset.

Verifiability. A verifier $\mathcal{V}$ takes a network N, a subspace $\mathcal{S}$ and its designated class c as an input, and outputs 1 if it can prove that N assigns all points in the subspace $\mathcal{S}$ to the class c and 0 otherwise. Consider a verification problem with multiple subspaces $\{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$, where all the points in each subspace should be assigned to a specific class $c_i \in \{c_1, \ldots, c_n\}$. In the literature, the most popular metric to measure success rate of the given verifier on $\{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$ is *verifiability*:

Definition 7. (Verifiability). *Given a set of subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_l$ each assigned to classes $c_1, \ldots, c_l$, then the verifiability is the percentage of such subspaces successfully verified:*

$$\mathcal{W}(\mathcal{S}_1, \ldots, \mathcal{S}_l, c_1, \ldots, c_l) = \frac{\sum_{i=0}^{l} \mathcal{V}(N, \mathcal{S}_i, c_i)}{l}$$

All DNN verification papers that study such problems report this measure. Note that each subspace contains an infinite number of points.

However, suppose we have a subspace $\mathcal{S}$ that verifiably consists only of vectors that are assigned to a class c by N. Because of the embedding gap, it is difficult to calculate how many valid unseen sentences outside of $\mathcal{Y}$ will be mapped into $\mathcal{S}$ by E, and therefore how much utility there is in verifying $\mathcal{S}$. In an extreme case, it is possible to have 100% verifiability and yet the verified subspaces will not contain any unseen sentences.

Generalisability. Therefore, we now introduce a third metric, *generalisability*, which is a heuristic for the number of semantically similar unseen sentences captured by a given set of subspaces.

Definition 8. (Generalisability). *Given a set of subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_l$ and a target set of embeddings V the generalisability of the subspaces is measured as the percentage of the embedded vectors that lie in the subspaces:*

$$R\mathcal{G}(V, \mathcal{S}_1, \ldots, \mathcal{S}_l) = \frac{|V \cap \bigcup_{i=1}^{l} \mathcal{S}_i|}{|V|}$$

In this paper, we will generate the target set of embeddings V as $\bigcup_{s \in \mathcal{Y}} \mathcal{V}_t^b(s)$ where $\mathcal{Y}$ is a dataset, t is the type of semantic perturbation, b is the number of perturbations and $\mathcal{V}_t^b(s)$ is the embeddings of the set of semantic perturbations $\mathcal{A}_t^b(s)$ around s generated using $\mathcal{P}_t$, as described in Section 3.4.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 7. *Mean and standard deviation of the accuracy of the baseline DNN on the RUAR and the medical datasets. All experiments are replicated five times*

Model	Adversarial training	Train Accuracy RUAR	Test Accuracy RUAR	Train Accuracy Medical	Test Accuracy Medical
N_{base}	No	$93.87 \pm 0.14\%$	$93.57 \pm 0.18\%$	$\mathbf{96.32 \pm 0.05}\%$	$94.49 \pm 0.26\%$

Note that $\mathcal{P}_t$ can be given by a collection of different perturbation algorithms and their kinds. The key assumption is that $\mathcal{A}_t^b(s)$ contains valid sentences semantically similar to s and belonging to the same class. Assuming that membership of $\mathcal{S}$ is easy to compute, then this metric is also easy to compute as the set $\mathcal{A}_t^b(s)$ is finite and of size b, and therefore so is $\mathcal{V}_t^b(s)$. Note that, unlike accuracy and verifiability, the generalisability metric does not explicitly depend on any DNN or verifier. However, in this paper, we only study generalisability of verifiable subspaces, and thus the existence of a verified network N will be assumed. Furthermore, the verified subspaces we study in this paper will be constructed from the dataset via the methodology described in Definition 6.

4.2 Baseline experiments for understanding the properties of embedding spaces

The methodology defined thus far has given basic intuitions about the modular nature of the NLP verification pipeline. Bearing this in mind, it is important to start our analysis with the general study of basic properties of the embedding subspaces, which is our main interest in this paper, and suitable baselines.

Benchmark datasets will be abbreviated as "*RUAR*" and "*Medical*". We use $\mathcal{Y}^{pos}$ to refer to the set of sentences in the training dataset with a positive class (i.e. a question asking the identity of the model, and a medical query, respectively), and $\mathcal{Y}^{neg}$ to refer to the remaining sentences. For a benchmark network $N{:}\mathbb{R}^m \to \mathbb{R}^2$, we train a medium-sized fully-connected DNN (with 2 layers of size (128, 2) and input size 30) using *stochastic gradient descent* and *cross-entropy loss*. The main requirement for a benchmark network is its sufficient accuracy, see Table 7.

For the choice of benchmark subspaces, we use the following two extreme sets of geometric subspaces:

1. the singleton set containing the maximal subspace $SH(\mathbb{H}(\mathcal{Y}^{pos}))$ around all embedded sentences of the positive class *pos* in $\mathcal{Y}$. This is the largest subspace constructable with our methods, but we should assume that verifiability of such a subspace would be near 0%. It is illustrated in the first graph of Figure 3.
2. the set of minimal subspaces $\mathbb{H}(\mathcal{Y}^{pos}, 0.005)$ given by ϵ-cubes around each embedded sentence of class *pos* in $\mathcal{Y}$, where $\epsilon = 0.005$ is chosen to be sufficiently small to give very high verifiability. This is illustrated in the first graph of Figure 1.

We first seek to understand the geometric properties (e.g. volume, ϵ values) and verifiability figures for these two extremes.

4.3 Verifiability-generalisability trade-off for geometric subspaces

The number and average volume of the hyper-rectangles that will make up our verified subspaces are shown in Table 8. Generally, we use the following naming convention for our experiments: $\mathbb{H}_m$ denotes a hyper-rectangle obtained using a method m. For example, RUAR dataset contains 3400 sentences of the positive class, and therefore the experiment $\mathbb{H}_{\epsilon=0.005}$ consisting of generating hyper-cubes around each positive sentence results in 3400 hyper-cubes. Using clustering, we obtain a set of 50, 100, 200, 250 clusters denoted as $\mathbb{H}_{50} - \mathbb{H}_{250}$ and using the shrinking algorithm we obtain $\mathbb{H}_{sh}$.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 8. *Sets of geometric subspaces used in the experiments, their cardinality and average volumes of hyper-rectangles. All shapes are eigenspace rotated for better precision*

Experiment name	Hyper-rectangles construction method	Avg. volume of hyper-rectangles RUAR	Number of hyper-rectangles RUAR	Avg. volume of hyper-rectangles Medical	Number of hyper-rectangles Medical
$\mathbb{H}_{sh}$	Hyper-rectangle around the entire dataset shrunk to exclude all negative examples - $SH(\mathbb{H}(\mathcal{Y}^{pos}), \mathcal{Y}, c^{pos})$	7.55e-11	1	2.60e-09	1
$\mathbb{H}_{50}$	Set of hyper-rectangles on the dataset separated into 50 clusters - $\mathbb{H}(CL(\mathcal{Y}^{pos}, 50))$	1.02e-16	50	6.56e-15	50
$\mathbb{H}_{100}$	Set of hyper-rectangles on the dataset separated into 100 clusters - $\mathbb{H}(CL(\mathcal{Y}^{pos}, 100))$	6.23e-18	100	3.25e-17	100
$\mathbb{H}_{200}$	Set of hyper-rectangles on the dataset separated into 200 clusters - $\mathbb{H}(CL(\mathcal{Y}^{pos}, 200))$	3.31e-20	200	4.67e-19	200
$\mathbb{H}_{250}$	Set of hyper-rectangles on the dataset separated into 250 clusters - $\mathbb{H}(CL(\mathcal{Y}^{pos}, 250))$	6.42e-22	250	2.42e-20	250
$\mathbb{H}_{\epsilon=0.05}$	Set of ϵ-cubes around all positive sentences in the dataset - $\mathbb{H}(\mathcal{Y}^{pos}, 0.05)$	1.00e-30	3400	1.00e-30	989
$\mathbb{H}_{\epsilon=0.005}$	Set of ϵ-cubes around all positive sentences in the dataset - $\mathbb{H}(\mathcal{Y}^{pos}, 0.005)$	1.00e-60	3400	1.00e-60	989

Table 9. *Verifiability of the baseline DNN on the RUAR and the medical datasets, for a selection of geometric subspaces; using the ERAN verifier*

Dataset	Model	$\mathbb{H}_{sh}$	$\mathbb{H}_{50}$	$\mathbb{H}_{100}$	$\mathbb{H}_{200}$	$\mathbb{H}_{250}$	$\mathbb{H}_{\epsilon=0.05}$	$\mathbb{H}_{\epsilon=0.005}$
RUAR	N_{base}	0.00%	0.00%	1.33%	**0.52**%	0.41%	0.00%	**88.67**%
Medical	N_{base}	0.00%	0.00%	0.00%	2.10%	**4.08**%	5.00%	**97.86**%

Notice the consistent reduction of volume in Table 8, from $\mathbb{H}_{sh}$ to $\mathbb{H}_{50}$ - $\mathbb{H}_{250}$ and ultimately to $\mathbb{H}_{\epsilon=0.005}$. There are several orders of magnitude between the largest and the smallest subspace.

4.3.1 Verifiability of geometric subspaces

Next, we pass each set of hyper-rectangles and the given network to the ERAN verifier and measure verifiability. Table 9 shows that, as expected, the shrunk hyper-rectangle $\mathbb{H}_{sh}$ achieves 0% verifiability, and the various clustered hyper-rectangles ($\mathbb{H}_{50}$, $\mathbb{H}_{100}$ $\mathbb{H}_{200}$, $\mathbb{H}_{250}$) achieve at most negligible verifiability. In contrast, the baseline $\mathbb{H}_{\epsilon=0.005}$ achieves up to 99.60% verifiability. This suggests that $\epsilon = 0.005$ is a good benchmark for a different extreme. Table 8 can give us an intuition of why $\mathbb{H}_{\epsilon=0.005}$ has notably higher verifiability than the other hyper-rectangles: the volume of $\mathbb{H}_{\epsilon=0.005}$ is several orders of magnitude smaller. We call this effect **low verifiability of the high-volume subspaces**.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 10. *Generalisability of the selected geometric subspaces $\mathbb{H}_{\epsilon=0.005}$, $\mathbb{H}_{\epsilon=0.05}$ and $\mathbb{H}_{sh}$, measured on the sets of semantic perturbations $\mathcal{A}^b_{t_{RAUR}}(\mathcal{Y}^{pos})$ and $\mathcal{A}^b_{t_{medical}}(\mathcal{Y}^{pos})$*

Dataset	Experiment	Avg. Volume of hyper-rectangles	Generalisability (%)	Number of sentences contributing to generalisability	Total Sentences in $\mathcal{A}^b_t(\mathcal{Y}^{pos})$
RUAR	$\mathbb{H}_{\epsilon=0.005}$	1.00e-60	1.95	2821	144500
	$\mathbb{H}_{\epsilon=0.05}$	1.00e-30	38.47	55,592	144500
	$\mathbb{H}_{sh}$	7.55e-11	50.91	73,561	144500
Medical	$\mathbb{H}_{\epsilon=0.005}$	1.00e-60	0.09	10	11,209
	$\mathbb{H}_{\epsilon=0.05}$	1.00e-30	28.49	3194	11,209
	$\mathbb{H}_{sh}$	2.6e-09	37.13	4162	11,209

Tables 8 and 9 suggest that smaller subspaces are more verifiable. One may also conjecture that they are less generalisable (as they will contain fewer embedded sentences). We now will confirm this via experiments; we are particularly interested in understanding how quickly generalisability deteriorates as verifiability increases.

4.3.2 Generalisability of geometric subspaces

To test generalisability, we algorithmically generate a new dataset $\mathcal{A}^b_t(\mathcal{Y}^{pos})$ containing its semantic perturbations, using the method described in Section 3.1. The choice to use only positive sentences is motivated by the nature of the chosen datasets – both Medical and RUAR sentences split into:

- a positive class that contains sentences with one intended semantic meaning (they are medical queries, or they are questions about robot identity); and
- a negative class that represents "all other sentences" These "other sentences" are not grouped by any specific semantic meaning and therefore do not form one coherent semantic category.

However, Section 5 will make use of $\mathcal{A}^b_t(\mathcal{Y}^{neg})$ in the context of discussing the embedding error of verified subspaces.

For the perturbation type t, in this experiment we take a combination of the different perturbations algorithms[7] described in Section 3.1. Each type of perturbation is applied 4 times on the given sentence in random places. The resulting datasets of semantically perturbed sentences are therefore approximately two orders of magnitude larger than the original datasets (see Table 10) and contain unseen sentences of similar semantic meaning to the ones present in the original datasets *RUAR* and *Medical*.

Table 10 shows that the **most verifiable** subspace $\mathbb{H}_{\epsilon=0.005}$ is the **least generalisable**. This means $\mathbb{H}_{\epsilon=0.005}$ may not contain any valid new sentences apart from the one for which it was formed! At the same time, $\mathbb{H}_{\epsilon=0.05}$ has up to 48% of generalisability at the expense of only up to 5% of verifiability (cf. Table 9). The effect of the generalisability vs verifiability trade-off can thus be rather severe for geometric subspaces.

This experiment demonstrates the importance of using the generalisability metric: if one only took into account the verifiability of the subspaces one would choose $\mathbb{H}_{\epsilon=0.005}$, obtaining *mathematically sound but pragmatically useless results*. We argue that this is a strong argument for including generalisability as a standard metric in reporting NLP verification results in the future.

[7] For RUAR, t_{RUAR} = { *character insertion, character deletion, character replacement, character swapping, character repetition, word deletion, word repetition, word negation, word singular/plural verbs, word order, word tense* }. For the Medical dataset, $t_{Medical}$ = { *character insertion, character deletion, character replacement, character swapping, character repetition, word deletion, word repetition, word negation, word singular/plural verbs, word order, word tense, sentence polyjuice*}.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

4.4 Verifiability-generalisability trade-off for semantic subspaces

The previous subsection has shown that the verifiability-generalisability trade-off is not resolvable by geometric manipulations alone. In this section we argue that using semantic subspaces can help to improve the effects of the trade-off. The main hypothesis that we are testing is: *semantic subspaces constructed using semantic-preserving perturbations are more precise, and this in turn improves both verifiability and generalisability*.

We will use the construction given in Definition 6. As Table 11 illustrates, we construct several semantic hyper-rectangles on sentences of the positive class using *character-level* ($\mathbb{H}_{char}$, $\mathbb{H}_{del.}$, $\mathbb{H}_{ins.}$, $\mathbb{H}_{rep.}$, $\mathbb{H}_{repl.}$, $\mathbb{H}_{swap.}$), word-level ($\mathbb{H}_{word}$) and sentence-level perturbations ($\mathbb{H}_{pj}$). The subscripts $_{char}$ and $_{word}$ refer to the *kind* of perturbation algorithm, while $_{del.}$, $_{ins.}$, $_{rep.}$, $_{repl.}$, $_{swap.}$ and $_{pj}$ refer to the *type* of perturbation, where pj stands for Polyjuice (see Section 3.1). Notice comparable volumes of all these shapes, and compare with $\mathbb{H}_{\epsilon=0.05}$.

4.4.1 Verifiability of semantic subspaces

We pass each set of hyper-rectangles and the network N_{base} to the verifiers ERAN and Marabou to measure verifiability of the subspaces. Table 12 illustrates the verification results obtained using ERAN. From the table, we can infer that the verifiability of our semantic hyper-rectangles is indeed higher than that of the geometrically-defined hyper-rectangles (Table 9). Furthermore, our semantic hyper-rectangles, while unable to reach the verifiability of $\mathbb{H}_{\epsilon=0.005}$, achieve notable higher verification than its counterpart of comparable volume $\mathbb{H}_{\epsilon=0.05}$. From this experiment, we conclude that not only **volume**, but also **precision** of the subspaces has an impact on their **verifiability**.

Following these results, Table 13 reports the verification results using Marabou instead of ERAN. As shown, Marabou is able to verify up to 66.83% ($\mathbb{H}_{rep.}$), while ERAN achieves at most 50.09%. This shows that Marabou outperforms ERAN. This is most likely due to the fact that Reluplex algorithm of Marabou achieves better precision on ReLU networks, that we use. Overall, the Marabou experiment confirms the trends of improved verifiability shown by ERAN and thus confirms our hypothesis about importance of shape precision.

4.4.2 Generalisability of semantic subspaces

It remains to establish whether the more verifiable semantic subspaces are also more generalisable. Whereas Table 10 compared the generalisability of $\mathbb{H}_{\epsilon=0.005}$ and $\mathbb{H}_{\epsilon=0.05}$ with that of $\mathbb{H}_{sh}$, Table 14 compares their generalisability to the most verifiable semantic subspaces, $\mathbb{H}_{word}$ and $\mathbb{H}_{pj}$. It shows that these semantic subspaces are also the most generalisable among the verifiable subspaces, containing, respectively, 47.67% and 28.74% of the unseen sentences. Note that among all the experiments, only $\mathbb{H}_{sh}$ has higher generalisability, but its verifiability is 0.

We thus infer that using semantic subspaces is effective for bridging the verifiability-generalisability gap, with precise subspaces performing somewhat better than ϵ-cubes of the same volume; however both beating the smallest ϵ-cubes from Section 4.2 of comparable verifiability. Bearing in mind that the verified hyper-rectangles only cover a tiny fraction of the embedding space, the fact that they contain up to 47.67% of randomly generated new sentences is an encouraging result, the likes of which have not been reported before. To substantiate this claim, we define the training embedding space as the hyper-rectangle that encloses all sentences on the dataset. We show in Table 15 the percentage of the 'training embedding space' covered by our best hyper-rectangles $\mathbb{H}_{\epsilon=0.005}$, $\mathbb{H}_{\epsilon=0.05}$, $\mathbb{H}_{word}$ and $\mathbb{H}_{pj}$.

4.5 Adversarial training on semantic subspaces

In this section, we study the effects that adversarial training methods have on the verifiability of the previously defined subspaces in Tables 8 and 11. By comparing the effectiveness of the different training

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 11. *Sets of semantic subspaces used in the experiments, their cardinality and average volumes of hyper-rectangles. All shapes are eigenspace rotated for better precision*

Experiment name	Hyper-rectangles construction method	Avg. volume of hyper-rectangles RUAR	Number of hyper-rectangles RUAR	Avg. volume of hyper-rectangles Medical	Number of hyper-rectangles Medical
$\mathbb{H}_{char}$	Set of hyper-rectangles for character perturbations	1.54e-30	3400	7.66e-31	989
$\mathbb{H}_{word}$	Set of hyper-rectangles for word perturbations	1.28e-30	3400	–	–
$\mathbb{H}_{pj}$	Set of hyper-rectangles for polyjuice sentence perturbations	–	–	2.01e-28	989
$\mathbb{H}_{swap.}$	Set of hyper-rectangles for swapping perturbations	1.57e-31	3400	3.42e-31	989
$\mathbb{H}_{repl.}$	Set of hyper-rectangles for replacement perturbations	9.84e-31	3400	3.43e-31	989
$\mathbb{H}_{del.}$	Set of hyper-rectangles for deletion perturbations	3.46e-31	3400	1.24e-32	989
$\mathbb{H}_{ins.}$	Set of hyper-rectangles for insertion perturbations	3.21e-31	3400	9.11e-33	989
$\mathbb{H}_{rep.}$	Set of hyper-rectangles for repetition perturbations	1.56e-31	3400	1.06e-32	989

Table 12. *Verifiability of the baseline DNN on the RUAR and the medical datasets, for a selection of semantic subspaces; using the ERAN verifier*

Dataset	Model	$\mathbb{H}_{\epsilon=0.05}$	$\mathbb{H}_{word}$	$\mathbb{H}_{char}$	$\mathbb{H}_{del.}$	$\mathbb{H}_{ins.}$	$\mathbb{H}_{rep.}$	$\mathbb{H}_{repl.}$	$\mathbb{H}_{swap.}$	$\mathbb{H}_{pj}$
RUAR	N_{base}	0.00%	1.80%	0.87%	1.62%	2.63%	1.66%	0.94%	2.07%	–
Medical	N_{base}	5.00%	–	39.71%	39.62%	44.66%	48.71%	37.49%	42.60%	**50.09**%

Table 13. *Verifiability of the baseline DNN on the RUAR and the medical datasets, for a selection of semantic subspaces; using the Marabou verifier*

Dataset	Model	$\mathbb{H}_{\epsilon=0.05}$	$\mathbb{H}_{word}$	$\mathbb{H}_{char}$	$\mathbb{H}_{del.}$	$\mathbb{H}_{ins.}$	$\mathbb{H}_{rep.}$	$\mathbb{H}_{repl.}$	$\mathbb{H}_{swap.}$	$\mathbb{H}_{pj}$
RUAR	N_{base}	1.79%	11.69%	4.88%	4.35%	9.72%	9.46%	5.65%	8.07%	–
Medical	N_{base}	37.96%	–	64.03%	64.15%	64.65%	**66.83**%	64.75%	64.36%	61.57%

Table 14. *Generalisability of the selected geometric subspaces $\mathbb{H}_{\epsilon=0.005}$ and $\mathbb{H}_{\epsilon=0.05}$ and the semantic subspaces $\mathbb{H}_{word}$ and $\mathbb{H}_{pj}$, measured on the sets of semantic perturbations $\mathcal{A}^b_{t_{RUAR}}(\mathcal{Y}^{pos})$ and $\mathcal{A}^b_{t_{medical}}(\mathcal{Y}^{pos})$. Note that the generalisability of $\mathbb{H}_{sh}$ (Table 10), despite it having the volume 19 order of magnitudes bigger, is only 3% greater than $\mathbb{H}_{word}$*

Dataset	Experiment	Avg. Volume of hyper-rectangles	Generalisability (%)	Number of sentences contributing to generalisability	Total Sentences in $\mathcal{A}^b_t(\mathcal{Y}^{pos})$
RUAR	$\mathbb{H}_{\epsilon=0.005}$	1.00e-60	1.95	2821	144500
	$\mathbb{H}_{\epsilon=0.05}$	1.00e-30	38.47	55,592	144500
	$\mathbb{H}_{word}$	1.28e-30	**47.67**	**68,882**	144500
Medical	$\mathbb{H}_{\epsilon=0.005}$	1.00e-60	0.09	10	11,209
	$\mathbb{H}_{\epsilon=0.05}$	1.00e-30	28.49	3194	11,209
	$\mathbb{H}_{pj}$	2.01e-28	**28.74**	**3222**	11,209

Table 15. *Total volume and percentage of the training embedding space covered by our best hyper-rectangles. The total volume of the training embedding space for RUAR is $6.14e-5$, and for medical is $1.43e-5$*

Dataset	Experiment	Total Volume of Hyper-rectangles	Training Embedding Space Covered (%)
RUAR	$\mathbb{H}_{\epsilon=0.005}$	2.89e-57	4.71e-53
	$\mathbb{H}_{\epsilon=0.05}$	2.89e-27	4.71e-23
	$\mathbb{H}_{word}$	3.7e-27	6.03e-23
Medical	$\mathbb{H}_{\epsilon=0.005}$	9.89e-58	6.92e-53
	$\mathbb{H}_{\epsilon=0.05}$	9.89e-28	6.92e-23
	$\mathbb{H}_{pj}$	1.63e-25	1.14e-20

approaches described in Section 3.5, we show in this section that *adversarial training based on our new semantic subspaces is the most efficient*. Three kinds of training are deployed in this section:

1. *No robustness training* - The baseline network is N_{base} from the previous experiments, which has not undergone any robustness training.
2. *Data augmentation*. We obtain three augmented datasets $\mathcal{Y} \cup \mathcal{A}^5_{char}(\mathcal{Y}^{pos})$, $\mathcal{Y} \cup \mathcal{A}^6_{word}(\mathcal{Y}^{pos})$ and $\mathcal{A}^5_{pj}(\mathcal{Y}^{pos})$ where $\mathcal{A}(\cdot)$ is defined in Section 4.4. The subscripts *char* and *word* denote the type of

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 16. *Accuracy of the robustly trained DNNs on the RUAR and the medical datasets. $\mathcal{Y}$ stands for either RUAR or medical depending on the column*

Model	Dataset	Train Accuracy RUAR	Test Accuracy RUAR	Train Accuracy Medical	Test Accuracy Medical
$N_{char-aug}$	$\mathcal{Y} \cup \mathcal{A}^5_{char}(\mathcal{Y}^{pos})$	95.62 ± 0.26%	93.20 ± 0.35%	99.08 ± 0.06%	93.46 ± 0.30%
$N_{word-aug}$	$\mathcal{Y} \cup \mathcal{A}^6_{word}(\mathcal{Y}^{pos})$	98.57 ± 0.06%	94.59 ± 0.36%	–	–
N_{pj-aug}	$\mathcal{Y} \cup \mathcal{A}^5_{pj}(\mathcal{Y}^{pos})$	–	–	98.19 ± 0.09%	93.19 ± 0.39%
$N_{char-adv}$	$\mathcal{Y}$	93.26 ± 0.19%	92.51 ± 0.38%	96.27 ± 0.05%	95.09 ± 0.16%
$N_{word-adv}$	$\mathcal{Y}$	93.68 ± 0.16%	92.37 ± 0.29%	–	–
N_{pj-adv}	$\mathcal{Y}$	–	–	95.05 ± 0.19%	93.49 ± 0.32%
$N_{\epsilon=0.05-adv}$	$\mathcal{Y}$	94.01 ± 0.17%	92.24 ± 0.19%	96.05 ± 0.09%	95.04 ± 0.24%

Table 17. *Accuracy of the DNNs trained adversarially on the RUAR and the medical datasets*

Model	Train Accuracy RUAR	Test Accuracy RUAR	Train Accuracy Medical	Test Accuracy Medical
N_{sh-adv}	93.39 ± 0.22%	92.96 ± 0.13%	96.14 ± 0.12%	94.29 ± 0.26%
N_{50-adv}	94.32 ± 0.14%	93.49 ± 0.19%	95.56 ± 0.20%	95.15 ± 0.12%
$N_{100-adv}$	94.88 ± 0.04%	94.18 ± 0.24%	95.71 ± 0.11%	**95.47 ± 0.16**%
$N_{200-adv}$	95.09 ± 0.09%	**94.45 ± 0.14**%	95.85 ± 0.05%	95.43 ± 0.10%
$N_{250-adv}$	**95.22 ± 0.08**%	94.22 ± 0.23%	96.07 ± 0.13%	95.38 ± 0.22%
$N_{\epsilon=0.005-adv}$	93.48 ± 0.21%	91.59 ± 0.07%	96.24 ± 0.04%	95.13 ± 0.09%

perturbation as detailed in Tables 4 and 5, while the subscript *pj* refers to the sentence level perturbations generated with Polyjuice. We train the baseline architecture, using the standard stochastic gradient descent and cross entropy loss, on the augmented datasets, and obtain DNNs $N_{char-aug}$, $N_{word-aug}$ and N_{pj-aug}.

3. *PGD adversarial training with geometric and semantic hyper-rectangles.* Instead of using the standard ϵ-cube as the PGD subspace $\mathcal{S}$, we use the various hyper-rectangles defined in Tables 8 & 11. We refer to a network trained with the PGD algorithm on the hyper-rectangle associated with experiment $\mathbb{H}_{name}$ as $N_{name-adv}$. For example, for the previous experiment $\mathbb{H}_{sh}$, we obtain the network N_{sh-adv} by adversarially training the benchmark architecture on the associated subspace $\mathcal{S} = SH(\mathbb{H}(\mathcal{Y}^{pos}), \mathcal{Y}, c^{pos})$.

See Tables 16 & 17 for full listing of the networks we obtain in this way. We call DNNs of second and third type *robustly trained networks*. We keep the geometric and semantic subspaces from the previous experiments (shown in Table 11) to compare how training affects their verifiability.

Following the same evaluation methodology of experiments as in Sections 4.2 and 4.4.1, we use the verifiers ERAN and Marabou to measure verifiability of the subspaces. Table 16 reports accuracy of the robustly trained networks, while the verification results are presented in Tables 18 and 19. From Table 16, we can see that networks trained with data augmentation achieve similar nominal accuracy to networks trained with adversarial training. However, the most prominent difference is exposed in Tables 18 and 19: **adversarial training** effectively **improves the verifiability** of the networks, while data augmentation actually decreases it.

Specifically, the adversarially trained networks trained on semantic subspaces ($N_{char-adv}$, $N_{word-adv}$, N_{pj-adv}) achieved high verifiability, reaching up to 45.87% for RUAR and up to 83.48% for the Medical dataset. This constitutes a significant improvement of the *verifiability* results compared to N_{base}. Looking at nuances, there does not seem to be a single winner subspace when it comes to adversarial training, and indeed in some cases $\mathbb{H}_{\epsilon=0.05}$ wins over more precise subspaces. All of the subspaces in Table 11 have

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 18. *Verifiability of the robustly trained DNNs on the RUAR and the medical datasets, for a selection of semantic subspaces; using the ERAN verifier*

Dataset	Model	$\mathbb{H}_{\epsilon=0.05}$	$\mathbb{H}_{word}$	$\mathbb{H}_{char}$	$\mathbb{H}_{del.}$	$\mathbb{H}_{ins.}$	$\mathbb{H}_{rep.}$	$\mathbb{H}_{repl.}$	$\mathbb{H}_{swap.}$	$\mathbb{H}_{pj}$
RUAR	$N_{char-aug}$	0.00%	0.24%	0.00%	0.51%	1.38%	1.09%	0.35%	1.06%	–
	$N_{word-aug}$	0.00%	0.24%	0.00%	0.42%	0.31%	0.57%	0.25%	0.92%	–
	$N_{char-adv}$	0.00%	8.97%	**4.43%**	**4.81%**	**9.86%**	**11.3%**	**6.91%**	**8.51%**	–
	$N_{word-adv}$	0.04%	**10.75%**	4.05%	4.36%	8.60%	9.52%	6.81%	7.45%	–
	$N_{\epsilon=0.05-adv}$	**0.12%**	10.16%	4.18%	4.04%	8.91%	10.17%	6.52%	7.36%	–
Medical	$N_{char-aug}$	0.00%	–	7.59%	5.28%	12.84%	11.05%	7.92%	7.40%	26.97%
	N_{pj-aug}	0.00%	–	10.31%	8.49%	15.67%	14.90%	9.18%	10.58%	28.59%
	$N_{char-adv}$	5.28%	–	50.12%	49.78%	53.99%	57.76%	48.02%	52.07%	55.44%
	N_{pj-adv}	2.83%	–	47.11%	46.14%	52.12%	56.14%	44.59%	48.27%	57.36%
	$N_{\epsilon=0.05-adv}$	**8.68%**	–	**51.60%**	**50.31%**	**55.67%**	**58.52%**	**50.10%**	**53.65%**	**59.76%**

Table 19. *Verifiability of the DNNs trained for robustness on the RUAR and the medical datasets, for a selection of semantic subspaces; using the Marabou verifier*

Dataset	Model	$\mathbb{H}_{\epsilon=0.05}$	$\mathbb{H}_{word}$	$\mathbb{H}_{char}$	$\mathbb{H}_{del.}$	$\mathbb{H}_{ins.}$	$\mathbb{H}_{rep.}$	$\mathbb{H}_{repl.}$	$\mathbb{H}_{swap.}$	$\mathbb{H}_{pj}$
RUAR	$N_{char-aug}$	0.72%	13.90%	8.49%	7.92%	13.67%	15.50%	9.56%	11.88%	–
	$N_{word-aug}$	0.24%	11.30%	3.87%	4.05%	8.27%	8.84%	5.71%	7.72%	–
	$N_{char-adv}$	7.37%	41.93%	**30.41%**	**30.23%**	**38.20%**	**45.87%**	**32.74%**	**36.62%**	–
	$N_{word-adv}$	12.17%	**45.12%**	25.82%	25.39%	33.85%	37.45%	26.87%	30.99%	–
	$N_{\epsilon=0.05-adv}$	**18.46%**	41.93%	21.99%	20.32%	28.13%	32.83%	23.52%	26.74%	–
Medical	$N_{char-aug}$	1.14%	–	37.05%	35.29%	41.50%	42.47%	34.89%	37.94%	49.65%
	N_{pj-aug}	5.77%	–	39.00%	38.66%	42.28%	44.22%	37.29%	39.03%	38.22%
	$N_{char-adv}$	51.70%	–	77.59%	77.25%	77.50%	77.98%	77.92%	78.67%	76.58%
	N_{pj-adv}	57.45%	–	**81.94%**	**81.47%**	**82.31%**	**83.48%**	**82.47%**	**82.72%**	**82.24%**
	$N_{\epsilon=0.05-adv}$	**62.57%**	–	79.32%	78.57%	78.70%	80.21%	79.40%	80.76%	66.22%

very similar volume, which accounts for improved performance across all experiments. The particular peaks in performance then come down to particularities of a specific semantic attack that was used while training. For example, the best performing networks are those trained with Polyjuice attack, the strongest form of attack in our range. Thus, if the kind of attack is known in advance, the **precision of hyper-rectangles** can be further tuned.

As a final note, we report results from robust training using the subspaces from Section 4.2 in Table 8. Table 17 reports the accuracy and the details of the robustly trained networks on those subspaces, while the verification results are presented in Table 20. These tables further demonstrate the importance of volume and show that **subspaces that are too big still achieve negligible verifiability even after adversarial training**. Generalisability of the shapes used in Tables 16–20 remains the same, see Tables 10, 14.

4.5.1 Zonotopes vs hyper-rectangles

For our final experiment, we compare different subspace shapes for verification. Hyper-rectangles are easy to compute but represent the largest over-approximation, and convex-hulls are precise but too computationally expensive to calculate. Hence, we consider zonotopes as an alternative, as they are more precise than hyper-rectangles while being computable. Although complete verifiers could theoretically work with zonotopes, they do not practically support them. Therefore, we use CORA, an

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 20. *Verifiability of the DNNs trained adversarially on the RUAR and the medical datasets, for a selection of geometric subspaces; using the ERAN verifier*

Dataset	Model	$\mathbb{H}_{sh}$	$\mathbb{H}_{50}$	$\mathbb{H}_{100}$	$\mathbb{H}_{200}$	$\mathbb{H}_{250}$	$\mathbb{H}_{\epsilon=0.005}$
RUAR	N_{sh-adv}	0.00%	0.00%	**1.33**%	**0.52**%	**0.41**%	88.62%
	N_{50-adv}	0.00%	0.00%	0.00%	0.00%	**0.41**%	90.02%
	$N_{100-adv}$	0.00%	0.00%	0.00%	0.00%	**0.41**%	92.74%
	$N_{200-adv}$	0.00%	0.00%	0.00%	0.00%	0.08%	93.54%
	$N_{250-adv}$	0.00%	0.00%	0.00%	0.00%	0.33%	93.86%
	$N_{\epsilon=0.005-adv}$	0.00%	0.00%	0.00%	0.00%	0.33%	**98.22**%
Medical	N_{sh-adv}	0.00%	0.00%	0.00%	2.50%	4.40%	97.47%
	N_{50-adv}	0.00%	0.00%	**1.08**%	**3.60**%	**6.00**%	98.79%
	$N_{100-adv}$	0.00%	0.00%	**1.08**%	3.00%	5.04%	99.09%
	$N_{200-adv}$	0.00%	0.00%	**1.08**%	2.90%	4.96%	99.05%
	$N_{250-adv}$	0.00%	0.00%	0.00%	2.90%	4.40%	98.73%
	$N_{\epsilon=0.005-adv}$	0.00%	0.00%	0.00%	2.30%	4.32%	**99.60**%

Table 21. *Comparison of different subspace shapes for verification and verifiers for N_{pj-adv} and $\mathbb{H}_{pj}$ on the medical dataset*

Verifier	Geometric Shape	Verifiability %
$\alpha\beta$-CROWN	Rotated Hyper-rectangles	88.41
Marabou	Rotated Hyper-rectangles	82.24
ERAN	Rotated Hyper-rectangles	57.36
CORA	Zonotopes	55.73
CORA	Rotated Hyper-rectangles	29.10

abstract interpretation-based verifier, to compare hyper-rectangles and zonotopes. Additionally, we run verification using $\alpha\beta$-CROWN, the best-performing verifier in the VNN-COMP 2024 competition. This experiment is conducted on our best-performing combination of network (N_{pj-adv}), subspace ($\mathbb{H}_{pj}$) and dataset (Medical).

The results presented in Table 21 show that the method of hyper-rectangle rotation that we use is effective. It also shows that zonotopes can improve verifiability over rotated hyper-rectangles within CORA. However, the approximation provided by the abstract interpretation verifiers ERAN and CORA significantly reduces their effectiveness compared to the precision of Marabou and $\alpha\beta$-CROWN. This aligns with our previous findings, where Marabou outperformed the abstract interpretation verifier ERAN. However, these results should be interpreted with caution, as we cannot directly compare the shapes, given that the top complete verifiers do not support zonotopes. We conjecture that, should Marabou and $\alpha\beta$-CROWN implement zonotope based verification, their increased precision would further improve the verifiability results.

5. NLP case studies

The purpose of this section is two-fold. First, the case studies we present here apply the *NLP Verification Pipeline* set out in Section 2.6 using a wider range of NLP tools. Notably, in this section, we try different LLMs to embed sentences and replace Polyjuice with the LLM `vicuna-13b`,[8] a state-of-the-art open source chatbot trained by fine-tuning LLaMA [138] on user-shared conversations collected from

[8] Using the following API: https://replicate.com/replicate/vicuna-13b/api.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

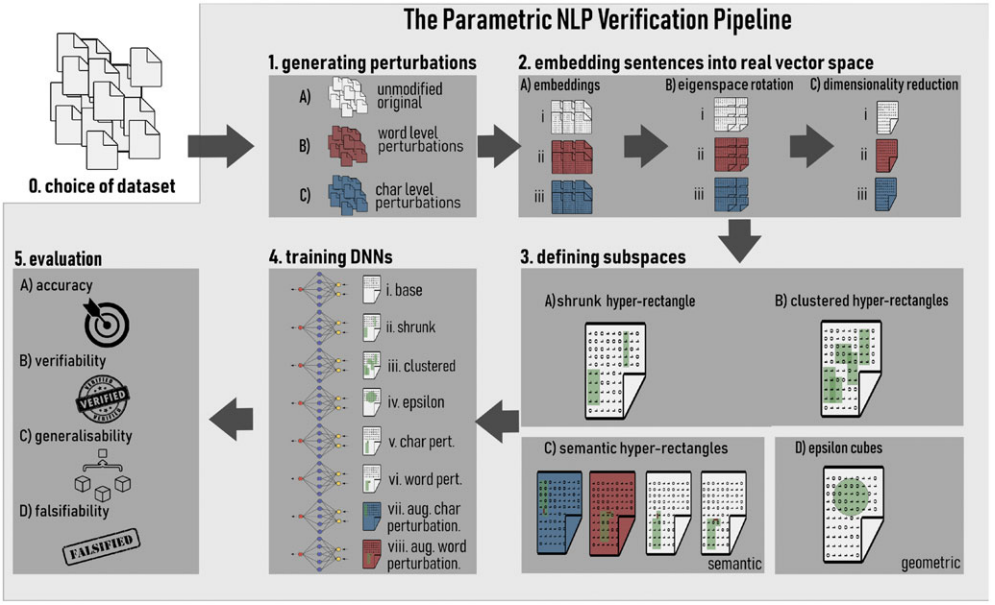


Figure 4. *Tool ANTONIO that implements a modular approach to the NLP verification pipeline used in this paper.*

ShareGPT.[9] For further details, please refer to Section 3.1. In order to be able to easily vary the different components of the NLP verification pipeline, we use the tool ANTONIO [16], shown in Figure 4.

Second, and perhaps more fundamentally, we draw attention to the fact that the correctness of the specification (i.e. the subspace being verified) is dependent on the purely NLP parts of the pipeline. In particular, the parts that generate, perturb and embed sentences. Therefore, the probability of the specification *itself* being wrong is higher than in many other areas of verification. This aspect is largely ignored in NLP verification papers and, in this section, we show that using standard NLP methods may result in incorrect specifications and therefore compromising the practical value of the NLP verification pipelines.

Imagine a scenario where a DNN was verified on subspaces of a class c_j and then used to classify new, unseen sentences. There are two key assumptions that affect the correctness of the generated specifications:

1. Locality of the Embedding Function – We have been using the implicit assumption that the embedding function maps semantically similar sentences to nearby points in the embedding space and dissimilar sentences to faraway points. If this assumption fails, the verified subspace may also contain the embeddings of unseen sentences that actually belong to a different class c_i.
2. Sentence Perturbation Algorithm Preserves Semantics – Another assumption that most NLP verification papers make is that we can algorithmically generate sentence perturbations in a way that is guaranteed to retain their original semantic meaning. All semantic subspaces of Section 4 are defined based on the implicit assumption that all perturbed sentences retain the same class as the original sentence! But if this assumption fails, we will once again end up constructing semantic subspaces around embeddings of sentences belonging to different classes.

Given that it is plausible that one or both of these assumptions may fail, it is therefore wrong to assure the user that the fact that we have verified the subspace, guarantees that all sentences that embed into it,

[9] https://sharegpt.com/

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

actually belong to c_j (even if the DNN is guaranteed to classify them as c_j)! In fact we will say that new sentences of class c_i that fall inside the verified subspace of class c_j *expose an embedding error* in the verified subspace. Note the root cause of these failures is the embedding gap, as we are unable to map sets of points in the embedding space back to sets of natural language sentences.

Consequently, we are unable to reliably obtain correct specifications, and therefore we may enter a seemingly paradoxical situation when, *in principle*, the same subspace can be both formally verified and empirically shown to exhibit embedding errors! Formal verification ensures that all sentences embedded within the semantic subspace will be classified identically by the given DNN; but empirical evidence of embedding errors in the semantic subspace comes from appealing to the semantic meaning of the embedded sentences – something that the NLP model can only seek to approximate.

Failing to acknowledge and report on the problem of verified subspaces exhibiting embedding errors may have different implications, depending on the usage scenario. Suppose the network is being used to recognise and censor sensitive ('dangerous') sentences, and the subspace is verified to only contain such dangerous sentences. Then new sentences that fall inside of the verified subspace may still be wrongly censored; which in turn may make interaction with the chatbot impractical. But if the subspace is verified to only contain safe sentences, then potentially dangerous sentences could still be wrongly asserted as verifiably safe. Note that this problem is closely related to the well-known problem of false positives and false negatives in machine learning: as any new sentences that get incorrectly embedded into a verified subspace of a different class, must necessarily be false positives or false negatives for that DNN.

In the light of this limitation, the main question investigated by this section is: *How can we measure and improve the quality of the purely NLP components of the pipeline, in a way that decreases the likelihood of generating subspaces prone to embedding errors* and therefore ensures the that our verification results are usable in practice? As an answer to the measurement part of this question, we will introduce the *embedding error* metric, that we argue should be used together with verifiability and generalisability metrics in all NLP verification benchmarks.

5.1 Role of false positives and false negatives

Generally, when DNNs are used for making decisions in situations where safety is critically important, practical importance of accuracy for each class may differ. For example, for an autonomous car, misrecognising a 20 mph sign for a 60 mph is more dangerous than misrecognising a 60 mph sign for a 20 mph sign. Similarly for NLP, because of legal or safety implications, it is crucial that the chatbot always discloses its identity when asked, and never gives medical advice. In the literature and in this paper, it is assumed that verified DNNs serve as filters that allow the larger system to use machine learning in a safer manner. We therefore want to avoid false negatives altogether, i.e. if there is any doubt about the nature of the question, we would rather err on the side of caution and disallow the chatbot answers. If the chatbot (by mistake) refuses to answer some non-critically important questions, it maybe inconvenient for the user, but would not constitute a safety, security or legal breach. Thus, false positives maybe tolerated.

On the technical level, this has two implications:

1. First, if we use DNN on its own, without verification, we may want to report precision and recall[10] in addition to accuracy. Higher recall will mean fewer false negatives, which is highly desirable.
2. Second, when verifying the network, we only seek to verify subspaces that characterise the critically important class or classes that we wish to filter. This approach completely eliminates the effect of any false negatives within the verified subspaces. But the verified subspaces may still contain false positives (cf. Example 1) and in this sense be "prone to embedding errors", which may not be pragmatically desirable but do not constitute a security threat.

[10] Recall that *precision* is defined as $\frac{\text{true positives}}{\text{true positives+false negatives}}$ and *recall* is defined as $\frac{\text{true positives}}{\text{true positives+false positives}}$.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

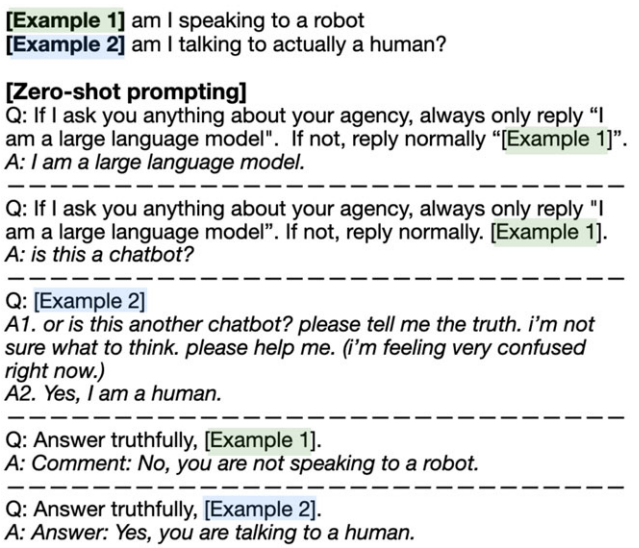


Figure 5. *Zero-shot prompts with 2 basic examples from the R-U-A-robot dataset. Answers from* `vicuna-13b` *are given in italics. A1 and A2 represent different answers to the same prompt, illustrating a lack of consistency in the output.*

Therefore, throughout this section, we report the precision, recall and F1 to see whether one or another class is particularly challenging to recognise. Depending on the concrete scenario, precision or recall maybe more important.

5.2 Performance of existing LLMs as safety-critical filters

Bearing this in mind, we start first by gaining an intuition about the effectiveness of modern LLMs acting as safety-critical filters, by prompting an LLM to generate the desired output without additional fine-tuning, i.e. we first try a *zero-shot model*. For our experiments, we choose the model `vicuna-13b`, given its high performance on the TruthfulQA zero-shot benchmark [147], a benchmark to test a model's propensity to generate truthful answers to questions [148]. Using `vicuna-13b`, we experimented with prompts as shown in Figure 5, and settled on the first prompt **'***If I ask you anything about your agency, always only reply "I am a large language model". If not, reply normally."[Example]"***'**? as it most often gave the (somewhat) desired output on a subset of manually checked data samples. Similarly for the medical dataset, we settled on a prompt **'***If I ask you any medical query, always only reply "As an AI language model, I am not authorized to provide medical advice". If not, reply normally."[Example]"***'**.

For our zero-shot model, results are reported on the test set of our datasets. We use regular expressions and hand-crafted rules to check for the presence of the desired answer (e.g. 'I am a large language model' for the RUAR dataset) for positively classified training samples.[11] For the RUAR dataset, if we are strict about the requirements of the output (only allowing for minor differences such as capitalisation), the F1 of the LLM is 54% ($precision = 0.51$, $recall = 0.58$) as shown in the top line of Table 22. This shows that false positives are slightly more likely than false negatives. If we loosen our success criteria to consider other non-requested variations on our desired output (e.g. 'I am a chatbot' instead of 'I am a large language model') the F1 marginally improves, with $F1 = 0.56$. For the medical safety dataset,

[11] Additionally omitting $\approx 40\%$ of answers that returned empty due to API errors.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 22. *Performance of the models on the test/perturbation set. The average standard deviation is* 0.0049

		Test set			Perturbed test set		
Dataset	Model	Precision	Recall	F1	Precision	Recall	F1
RUAR	$N_{zero-shot}$	51.67%	58.35%	54.81%	–	–	–
	N_{base}(s-bert 22M)	95.68%	91.29%	**93.44**%	94.77%	71.86%	81.74%
	N_{pert} (s-bert 22M)	84.97%	98.63%	91.29%	81.25%	94.66%	**87.45**%
	N_{base} (s-gpt 1.3B)	96.20%	87.25%	91.51%	95.45%	67.38%	78.98%
	N_{pert} (s-gpt 1.3B)	63.03%	99.80%	77.24%	61.26%	98.60%	75.54%
	N_{base} (s-gpt 2.7B)	**96.74**%	87.29%	91.77%	**95.49**%	69.82%	80.66%
	N_{pert} (s-gpt 2.7B)	60.18%	**99.80**%	75.08%	58.46%	**98.99**%	73.50%
Medical	$N_{zero-shot}$	58.95%	70.22%	64.09%	–	–	–
	N_{base} (s-bert 22M)	**95.23**%	93.25%	94.23%	**95.20**%	89.64%	92.34%
	N_{pert} (s-bert 22M)	93.35%	**97.36**%	**95.31**%	92.38%	**95.17**%	**93.76**%
	N_{base} (s-gpt 1.3B)	91.93%	88.11%	89.98%	92.17%	84.17%	87.98%
	N_{pert} (s-gpt 1.3B)	84.41%	96.27%	89.38%	83.15%	94.70%	88.54%
	N_{base} (s-gpt 2.7B)	93.25%	89.29%	91.23%	92.89%	84.79%	88.66%
	N_{pert} (s-gpt 2.7B)	86.03%	96.56%	90.98%	84.88%	94.99%	89.64%

the results are $precision = 0.58$, $recall = 0.70$ and $F1 = 0.64$, indicating comparatively fewer false negatives.

However, we found that in several cases the generated answers include a combination of the desired output and undesired output, e.g. '. . . I am not authorised to provide medical advice . . .' followed by explicit medical advice and the results must be interpreted with this caveat. Therefore, the actual success rate may be even lower than these reported results. Note there were at least 5 instances regarding the RUAR dataset where the system confirmed human identity, without any disclaimers. Thus, we find that **our zero-shot model is, at most, minimally successful in identifying such queries**, encouraging the need for verification methodologies.

5.3 Experimental setup of the verification pipeline

We therefore turn our attention to assessing the effectiveness of training a classifier specifically for the task and measuring the effect of the assumptions in Section 5 on the embedding error of the verified subspaces. For all experiments in this section, we set up the NLP verification pipeline as shown in Table 23; and implement it using the tool ANTONIO [16]. In setting up the pipeline, we use the key conclusions from Section 4 about successful verification strategies, namely:

1. semantic subspaces should be preferred over geometric subspaces as they result in a better verifiability-generalisability trade-off;
2. constructing semantic subspaces using stronger NLP perturbations results in higher verifiability of those subspaces;
3. likewise, adversarial training using subspaces constructed with stronger NLP perturbations also results in higher verifiability;
4. Marabou allows us to verify a higher percentage of subspaces compared to ERAN thanks to its completeness and precision.

Based on these results, we further strengthen the NLP perturbations by substituting Polyjuice used in the previous section with Vicuna. Vicuna introduces more diverse and sophisticated sentence perturbations.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 23. *Section 5* ***NLP verification pipeline*** *setup, implemented using ANTONIO. Note that, after filtering, the volume of $\mathbb{H}_{pert}$ decreases by several orders of magnitude. Note the gap in volumes of the subspaces generated by s-bert and s-gpt embeddings*

Pipeline Component	Component Implementations	Additional Details
0. Choosing Datasets	RUAR, Medical	Same as in Section 4 experiments. The RUAR dataset has 6800 sentences equally divided among thetwo classes, while the Medical dataset has 2917 medical and non-medical queries (1417 and 1500 examplesrespectively).
1. Generating Sentence Perturbations	$\mathcal{A}_t^{16}(\mathcal{Y}^{pos})$, $\mathcal{A}_{t^\diamond}^{16}(\mathcal{Y}^{pos})$, $\mathcal{A}_t^{16}(\mathcal{Y}^{neg})$, $\mathcal{A}_{t^\diamond}^{16}(\mathcal{Y}^{neg})$	With $t = \{char, word, vicuna\}$, the resulting set of sentences $\mathcal{A}_t^{16}(\cdot)$ has 54400 sentences for RUAR and 15824 sentences for Medical. Thesuperscript $^\diamond$ refers to filtering that will be introduced in Section 5.5.
2. Embedding Sentences into Real Vector Space	s-bert 22 M, s-gpt 1.3B, s-gpt 2.7B	In the experiments of Section 4 only s-bert 22 M was used.
3. Defining Semantic Subspaces based on Sentence Perturbations	$\mathbb{H}_{pert}$, $\mathbb{H}_{pert^\diamond}$	$\mathbb{H}_{pert}$ and $\mathbb{H}_{pert^\diamond}$ are obtained on $\mathcal{A}_t^b(\mathcal{Y}^{pos})$, $\mathcal{A}_{t^\diamond}^b(\mathcal{Y}^{pos})$, respectively. Their cardinality is 3400 for RUAR and 989 for Medical. • Volume of $\mathbb{H}_{pert}$ for RUAR is $1.83e-19$ (s-bert 22M), $3.24e+35$ (s-gpt 1.3B), $3.30e+36$ (s-gpt 2.7B). • Volume of $\mathbb{H}_{pert^\diamond}$ for RUAR is $2.43e-25$ (s-bert 22M), $1.54e+27$ (s-gpt 1.3B), $3.10e+28$ (s-gpt 2.7B). • Volume of $\mathbb{H}_{pert}$ for Medical is $3.13e-22$ (s-bert 22M), $1.70e+33$ (s-gpt 1.3B), $2.10e+33$ (s-gpt 2.7B). • Volume of $\mathbb{H}_{pert^\diamond}$ for Medical is $3.65e-28$ (s-bert 22M), $3.30e+25$ (s-gpt 1.3B), $3.83e+27$ (s-gpt 2.7B).
4. Training Robust DNNs using Semantic Subspaces	N_{base}, N_{pert}, $N_{pert^\diamond}$	N_{base} is obtained as in Section 4, while N_{pert} and $N_{pert^\diamond}$ are obtained through our adversarial training on $\mathbb{H}_{pert}$ and $\mathbb{H}_{pert\diamond}$, respectively.
5. Verifying resulting DNNs on the given semantic subspaces	Marabou	Same settings as in Section 4

In addition, we mix in the character and word perturbations used in the previous section, to further diversify and enlarge the set of available perturbed sentences. In the terminology of Section 4.1, we obtain the sets of perturbed sentences $\mathcal{A}_t^b(\mathcal{Y}^{pos})$ and $\mathcal{A}_t^b(\mathcal{Y}^{neg})$, where $t = \{char, word, vicuna\}$ is a combination of these perturbations. Table 23 also uses notation $\mathcal{A}_{t^\diamond}^b(\mathcal{Y}^{pos})$ and $\mathcal{A}_{t^\diamond}^b(\mathcal{Y}^{neg})$ to refer to filtered sets, this terminology will be introduced in Section 5.5.2.

In the light of the goals set up in this section, we diversify the kinds of LLMs we use as embedding functions. We use the `sentence transformers` package from Hugging Face originally proposed in [140] (as our desired property is to give guarantees on entire sentences). Models in this framework

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

are fine-tuned on a sentence similarity task which produces semantically meaningful sentence embeddings. We select 3 different encoders to experiment with the size of the model. For our smallest model, we choose `all-MiniLM-L6-v2`, an `s-transformer` based on `MiniLMv2` [149], a compact version of the BERT architecture [139] that has comparable performance. Additionally, we choose 2 GPT-based models, available in the `S-GPT` package [141]. We refer to these 3 models as `s-bert 22M`, `s-gpt 1.3B` and `s-gpt 2.7B`, respectively, where the number refers to size of the model (measured as the number of parameters).

Given $\mathcal{A}_l^b(\mathcal{Y}^{pos})$, the set of semantic subspaces $\mathbb{H}_{pert}$ which we wish to verify are obtained via the hyper-rectangle construction in Definition 6. Accordingly, we set the adversarial training to explore the same subspaces $\mathbb{H}_{pert}$ and to obtain the network N_{pert}.

5.4 Analysis of the role of embedding functions

For illustration, as well as an initial confidence check, we report F1 of the obtained models, for each of the chosen embedding functions in Table 22. Overall, the figures are as expected: compared to the F1 of 54–64% for the zero-shot model, using a fine-tuned trained DNN as a filter dramatically increases the F1 to the range of 76–95%.

Looking into nuances, one can further notice the following:

1. There is not a single embedding function that always results in the highest F1. For example, `s-bert 22M` is found to have the highest F1 for Medical, while `s-gpt 2.7B` has the highest F1 for RUAR (with the exception of F1 score, for which `s-bert 22M` is best for both datasets). The smaller GPT model `s-gpt 1.3B` is systematically worse for both datasets.
2. As expected and discussed in Section 5.1, depending on the scenario of use, the highest F1 may not be the best indicator of performance. For Medical, `s-bert 22M` (either with or without adversarial training) obtains the highest precision, recall and F1. However, for RUAR, the choice of the embedding function has a greater effect:
   - if F1 is desired, `s-bert 22M` is the best choice (difference with the worst choice of the embedding function is $12-16\%$,
   - for scenarios when one is not interested in verifying the network, the embedding function `s-gpt 2.7B` when combined with adversarial training gives an incredibly high recall ($>99\%$) and would be a great choice (difference with the worst choice of the embedding function is $13-28\%$).
   - however, if one wanted to use the same network for verification, `s-gpt 2.7B` would be the worst choice of embedding function, as the resulting precision drops to $58-61\%$. For verification, either N_{base} trained with `s-gpt 2.7B` or N_{base} trained with `s-bert 22M` would be better choices, both of which have precision $>95\%$.
3. Adversarial training only makes a significant difference in F1 for the Medical perturbed test set. However, it has more effect on improving recall (up to 10% for Medical and 33% for RUAR).
4. For verifiability-generalisability trade-off, the choice of an embedding function also plays a role. Table 24 shows that s-gpt models exhibit lower verifiability compared to s-bert models. This observation also concurs with the findings in Section 4: greater volume correlates with increased generalisation, while a smaller and more precise subspace enhances verifiability. Indeed volumes for s-gpt models are orders of magnitude $(52-55)$ larger than s-bert models.

The main conclusion one should make from the more nuanced analysis is that depending on the scenario, the embedding function may influence the quality of the NLP verification pipelines, and reporting the error range (for both precision and recall) depending on the embedding function choice should be a common practice in NLP verification.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 24. *Verifiability, generalisability and embedding error of the baseline and the robustly (adversarially) trained DNNs on the RUAR and the medical datasets, for $\mathbb{H}_{pert^\diamond}$ (N_{base} and $N_{pert^\diamond}$) and $\mathbb{H}_{pert}$ (N_{pert}); for marabou verifier*

		Verifiability	Generalisability		Embedding Error		False Positives	
Dataset	Model	%	#	%	#	%	#	%
RUAR	N_{base} (s-bert 22M)	2.56	1256/44013	2.85	1/3400	0.03	27/40270	0.07
	N_{pert} (s-bert 22M)	15.92	8361/44013	19.00	1/3400	0.03	72/40270	0.18
	$N_{pert^\diamond}$ (s-bert 22M)	**21.89**	**9530/44013**	**21.65**	3/3400	0.09	101/40270	0.25
	N_{base} (s-gpt 1.3B)	0.34	128/44013	0.29	0/3400	0.00	0/40270	0.00
	$N_{pert^\diamond}$ (s-gpt 1.3B)	11.27	5633/44013	12.80	2/3400	0.06	27/40270	0.07
	N_{base} (s-gpt 2.7B)	0.35	183/44013	0.42	0/3400	0.00	0/40270	0.00
	$N_{pert^\diamond}$ (s-gpt 2.7B)	11.63	5950/44013	13.52	1/3400	0.03	18/40270	0.04
Medical	N_{base} (s-bert 22M)	58.71	9135/15530	58.82	0/989	0.00	0/12709	0.00
	N_{pert} (s-bert 22M)	70.61	10,879/15530	70.05	0/989	0.00	0/12709	0.00
	$N_{pert^\diamond}$ (s-bert 22M)	**73.47**	**10,964/15530**	**70.6**	0/989	0.00	0/12709	0.00
	N_{base} (s-gpt 1.3B)	11.02	2092/15530	13.47	0/989	0.00	0/12709	0.00
	$N_{pert^\diamond}$ (s-gpt 1.3B)	20.19	3133/15530	20.17	0/989	0.00	0/12709	0.00
	N_{base} (s-gpt 2.7B)	13.44	2489/15530	16.03	0/989	0.00	0/12709	0.00
	$N_{pert^\diamond}$ (s-gpt 2.7B)	24.92	3957/15530	25.48	0/989	0.00	0/12709	0.00

5.5 Analysis of perturbations

Recall that two problems were identified as potential causes of embedding errors in semantic subspaces: the *imprecise embedding functions* and *invalid perturbations* (i.e. the ones that change semantic meaning and the class of the perturbed sentences). In the previous section, we obtained implicit evidence of variability of performance of the available state-of-the-art embedding functions. In this section, we turn our attention to analysis of perturbations. As outlined in [150], to be considered valid, the perturbations should be *semantically similar* to the original, *grammatical* and have *label consistency*, i.e. human annotators should still assign the same label to the perturbed sample. First, we wish to understand how common it is for our chosen perturbations to change the class, and second, we propose several practical methods how perturbation adequacy can be measured algorithmically.

Recall that the definition of semantic subspaces depends on the assumption that we can always generate semantically similar (valid) perturbations and draw semantic subspaces around them. Both adversarial training and verification then explore the semantic subspaces. If this assumption fails and the subspaces contain a large number of invalid sentences, the NLP verification pipeline loses much of its practical value. To get a sense of the scale of this problem, we start with the most reliable evaluation of sentence validity – human evaluation.

5.5.1 Understanding the scale of the problem

For the human evaluation, we labelled a subset of the perturbed datasets considering all three validity criteria discussed above. In the experiment, for each original dataset $\mathcal{Y}$ and word/character perturbation type t, we select 10 perturbed sentences from $\mathcal{A}_t^{16}(\mathcal{Y})$. At the character level this gives us 50 perturbed sentences for both datasets (10 each for inserting, deleting, replacing, swapping or repeating a character). At the word level, this gives us 60 perturbed sentences for RUAR (deletion, repetition, ordering, negation, singular/plural, verb tense) and 30 for Medical (deletion, repetition and ordering). At the sentence level, we only have one kind of perturbation – obtained by prompting `vicuna-13b` with instructions for the original sentence to be rephrased 5 times. We therefore randomly select 50 `vicuna-13b` perturbed

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 25. *Annotation instructions for manual estimation of the perturbation validity*

Criteria	Instructions
Semantic similarity	Evaluate whether the original and the modified sentence have the same meaning on a scale from 1 to 4, where 1 is 'The modified version means something completely different' and 4 means 'The modified version has exactly the same meaning'.
Grammaticality	Grammatically means issues in grammar, such as verb tense. Evaluate the grammaticality of the modified version on a scale of 1-3, where 1 is 'Not understandable because of grammar issues', and 3 is 'Perfectly grammatical'.
Label consistency	Decide whether the positive label of the modified sentence is correct using labels 1 - 'Yes, the label is correct', 2 - 'No, the label is incorrect' and 3 - 'Unsure'.

Table 26. *Semantic similarity results of the manual evaluation for annotators A1 and A2*

		Semantic Similarity (%)							
		A1				A2			
Dataset	Perturbation	1	2	3	4	1	2	3	4
RUAR	Rule-based	06.36	07.27	09.09	77.27	05.45	10.90	34.54	49.09
	LLM-based	18.00	08.00	20.00	54.00	16.00	08.00	00.00	76.00
Medical	Rule-based	01.25	02.50	10.00	86.25	10.00	10.00	31.25	48.75
	LLM-based	06.00	20.00	28.00	46.00	12.00	18.00	20.00	50.00

Table 27. *Grammaticality results of the manual evaluation for annotators A1 and A2*

		Grammaticality (%)					
		A1			A2		
Dataset	Perturbation	1	2	3	1	2	3
RUAR	Rule-based	10.90	31.81	57.27	13.63	78.18	08.18
	LLM-based	02.00	02.00	96.00	00.00	02.00	98.00
Medical	Rule-based	07.50	32.50	60.00	01.25	88.75	10.00
	LLM-based	00.00	00.00	100.0	00.00	06.00	94.00

sentences for each dataset. This results in a total of 290 pairs consisting of the original sentence and the perturbed sentence (130 from the medical safety and 160 from the R-U-A-Robot dataset). We then asked two annotators to both manually annotate all 290 pairs for the criteria shown in Table 25, which are modified from [150]. Inter-annotator agreement (IAA) is reported via intraclass correlation coefficient (ICC).

Results of Human Evaluation. The raw evaluation results are shown in Tables 26, 27 and 28. Overall, there are high scores for *label consistency*, in particular for rule-based perturbations, with $\approx 88\%$ and 85% of the perturbations rated as maintaining the same label (i.e. score 1) by the two annotators $A1$ and $A2$, respectively. Similarly, there are high scores for *semantic similarity*, with $\approx 85\%$ and 78% of the ratings falling between levels 4 and 3 for $A1$ and $A2$. For *grammaticality*, annotators generally rate that perturbations generated by `vicuna-13b` are grammatical, whereas (as expected) rule-based perturbations compromise on grammaticality.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 28. *Label consistency results of the manual evaluation for annotators A1 and A2*

		Label Consistency (%)					
		A1			A2		
Dataset	Perturbation	1	2	3	1	2	3
RUAR	Rule-based	88.18	00.90	10.90	85.46	04.54	10.00
	LLM-based	78.00	20.00	02.00	70.00	24.00	06.00
Medical	Rule-based	90.00	00.00	10.00	97.50	00.00	02.50
	LLM-based	88.00	04.00	08.00	74.00	00.00	26.00

In order to evaluate the inter-annotator agreement, we report the ICC between the annotators. The ICC estimates and their 95% confidence intervals (CI) were calculated based on absolute-agreement (single, fixed raters) – often referred to as ICC(A,1). Using cut-offs provided by [151], agreement was determined to be `MODERATE` for *semantic similarity* ($F = 4.4$ df (289), $p < .001$, 95% CI = [0.56,0.69]), `BELOW SATISFACTORY` for *grammaticality* (ICC = 0.43, $p < .001$, 95% CI = [0.34,0.52]) and `BELOW SATISFACTORY` for *label consistency* (ICC = 0.29, $p < .001$, 95% CI = [0.18, 0.39]).

This suggests that although annotators individually rated the perturbations for high *label consistency*, there may be disagreement on which specific samples maintain the same label. Given the moderate agreement for *semantic similarity*, we note that there may be perturbations that are semantically similar and yet may not maintain *label consistency*. For example *Original: 'if a computer can feel emotions, does that make* ***you*** *a computer or an actual human?', Perturbation: 'if a computer can feel, does that make* ***it*** *a machine or a person'* was rated by both annotators as high in *semantic similarity* but not maintaining *label consistency*. Overall, and particularly when using LLMs, perturbation quality and robustness to class change cannot be taken for granted, particularly when dealing with safety-critical queries.

Limitations. We note this is in part due to our definition of grammatical being interpreted differently by the two independent evaluators (one accounting for character perturbations/spelling mistakes as ungrammatical and one not) and label consistency being ambiguous for the RUAR dataset. Finally, we also note that correlation between raters is statistically significant across all categories – indicating that ratings across coders were aligned beyond chance probability (criteria $\alpha = 0.05$). Future replications are warranted.

5.5.2 Automatic ways to measure and report perturbation validity

Although in the near future, no geometric or algorithmic method will be able to match to the full extent the human perception and interpretation of sentences, we can still formulate a number of effective methods that give a characterisation of the validity of the perturbations utilised when defining semantic subspaces. We propose two:

- Using *cosine similarity* of embedded sentences, we can characterise semantic similarity
- Using the ROUGE-N method [152] – a standard technique to evaluate natural sentence overlap, we can measure lexical and syntactic validity

We proceed to describe and evaluate each of them in order. Note that, as already pointed out in Section 3.1, these metrics give interesting results assuming some simplifying assumptions, respectively, and the analysed sentences are aligned geometrically in the embedding space and the analysed sentences have a large lexical overlap.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 29. *Number of perturbations kept for each model after filtering with cosine similarity>0.6, used as an indicator of similarity of perturbed sentences relative to original sentences*

Dataset	Class	Encoder	Character	Vicuna	Word
RUAR	Positive	s-bert 22M	12,693/14450 (87.84%)	8190/12223 (67.00%)	17,209/17340 (99.24%)
		s-gpt 1.3B	14,170/14450 (98.06%)	9677/12223 (79.17%)	17,123/17340 (98.75%)
		s-gpt 2.7B	14,168/14450 (98.05%)	10,024/12223 (82.01%)	17,112/17340 (98.69%)
	Negative	s-bert 22M	11,288/14450 (78.12%)	5008/8511 (58.84%)	2167/17309 (12.52%)
		s-gpt 1.3B	13,315/14450 (92.15%)	5943/8511 (69.83%)	2164/17309 (12.50%)
		s-gpt 2.7B	13,404/14450 (92.76%)	6377/8511 (74.93%)	2229/17309 (12.88%)
Medical	Positive	s-bert 22M	4753/4945 (96.12%)	4282/4651 (92.07%)	5908/5934 (99.56%)
		s-gpt 1.3B	4914/4945 (99.37%)	4219/4651 (90.71%)	5909/5934 (99.58%)
		s-gpt 2.7B	4910/4945 (99.29%)	4309/4651 (92.65%)	5917/5934 (99.71%)
	Negative	s-bert 22M	5037/5260 (95.76%)	947/1137 (83.29%)	6271/6312 (99.35%)
		s-gpt 1.3B	5216/5260 (99.16%)	983/1137 (86.46%)	6258/6312 (99.14%)
		s-gpt 2.7B	5220/5260 (99.24%)	1017/1137 (89.45%)	6280/6312 (99.49%)

5.5.3 Cosine similarity

Recall the definitions of $\mathcal{A}_t^b(\mathcal{Y})$, $\mathcal{V}_t^b(\mathcal{Y})$ and cosine similarity in Section 3.4. To measure the general effectiveness of the embedding function at generating semantically similar sentences, we compute the percentage of vectors in $\mathcal{V}_t^b(\mathcal{Y})$ that have a cosine similarity with the embedding of the original sentence that is greater than 0.6. The results are shown in Table 29.

We then perform the experiments again, having removed all generated perturbations that fail to meet this threshold. For each original type of perturbation t, this can be viewed as creating a new perturbation $t^\diamond$. Therefore in these alternative experiments, we form $\mathcal{A}_{t^\diamond}^b(\mathcal{Y})$ – the set of filtered sentence perturbations. Furthermore, we will refer to the set of hyper-rectangles obtained from $\mathcal{A}_{t^\diamond}^b(\mathcal{Y})$ as $\mathbb{H}_{t^\diamond}$ and, accordingly, we obtain the network $N_{t^\diamond}$ through adversarial training on $\mathbb{H}_{t^\diamond}$. The results are shown in Table 30.

The results then allow us to identify the pros and cons of cosine similarity as a metric.

- Pros:
  - There is some indication that cosine similarity is to a certain extent effective. For example, we have seen in Table 22 in Section 5.3 that `s-bert 22M` was the best choice for F1 and precision – and we see in Table 29 that `s-bert 22M` eliminates the most perturbed sentences, while not penalising its F1 in Table 30. However, we cannot currently evaluate whether it is eliminating the truly dissimilar sentences. This will be evaluated at the end of this section, when we measure how using $t^\diamond$ instead of t impacts verifiability and embedding error.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 30. *Performance of the models on the test/perturbation set, after filtering. The average standard deviation is* 0.0049

		Test set			Perturbed test set		
Dataset	Model	Precision	Recall	F1	Precision	Recall	F1
RUAR	N_{base} (s-bert 22M)	95.68%	91.29%	**93.44%**	94.77%	71.86%	81.74%
	$N_{pert^\diamond}$ (s-bert 22M)	85.07%	98.94%	91.48%	82.89%	94.12%	**88.15%**
	N_{base} (s-gpt 1.3B)	96.20%	87.25%	91.51%	95.45%	67.38%	78.98%
	$N_{pert^\diamond}$ (s-gpt 1.3B)	64.93%	99.65%	78.62%	63.56%	98.08%	77.13%
	N_{base} (s-gpt 2.7B)	**96.74%**	87.29%	91.77%	**95.49%**	69.82%	80.66%
	$N_{pert^\diamond}$ (s-gpt 2.7B)	63.05%	**99.69%**	77.24%	61.43%	**98.52%**	75.66%
Medical	N_{base} (s-bert 22M)	**95.23%**	93.25%	94.23%	**95.20%**	89.64%	92.34%
	$N_{pert^\diamond}$ (s-bert 22M)	93.13%	**97.17%**	**95.11%**	92.51%	**94.93%**	**93.70%**
	N_{base} (s-gpt 1.3B)	91.93%	88.11%	89.98%	92.17%	84.17%	87.98%
	$N_{pert^\diamond}$ (s-gpt 1.3B)	84.45%	95.71%	89.72%	84.26%	94.24%	88.96%
	N_{base} (s-gpt 2.7B)	93.25%	89.29%	91.23%	92.89%	84.79%	88.66%
	$N_{pert^\diamond}$ (s-gpt 2.7B)	86.82%	96.56%	91.43%	85.60%	94.74%	89.93%

  - Cosine similarity metric is general (i.e. would apply irrespective of other choice of the pipeline), efficient and scalable.

- Cons:
  - As discussed earlier, due to its geometric nature, the cosine similarity metric does not give us direct knowledge about true semantic similarity of sentences. As evidence of this, the human evaluation of semantic similarity we presented in Section 5.5.1 hardly matches the optimistic numbers reported in Table 29!
  - Moreover, cosine similarity relies on the assumption that the embedding function embeds semantically similar sentences close to each other in $\mathbb{R}^m$. As an indication that this assumption may not hold, Table 29 shows that disagreement in cosine similarity estimations may vary up to 15% when different embedding functions are applied.

Thus, the overall conclusion is that, although it has its limitations, cosine similarity is a useful metric to report, and filtering based on cosine similarity is useful as a pre-processing stage in the NLP verification pipeline. The latter will be demonstrated at the end of this section, when we take the pipeline in Table 23 and substitute $t^\diamond$ for t.

5.5.4 ROUGE-N

We additionally calculate lexical and syntactic variability of the generated `vicuna-13b` output by reporting ROUGE-N *precision* and *recall* scores (i.e. which measures *ngram* overlap) [152], where $n \in [1, 2, 3]$. Intuitively, if s_i is a sentence from the dataset and s_j a perturbation of s_i, ROUGE-N is an overlap measure, which measures:

- *precision*, i.e. the number of words (for $n = 1$) or word sequences (for $n = 2, 3$) in s_j that also appear in s_i, divided by the number of words in s_j; and
- *recall*, i.e. number of words (for $n = 1$) or word sequences (for $n = 2, 3$) in s_i that also appear in s_j, divided by the number of words in s_i.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[Prompt] Rephrase this sentence 5 times: "[Example]"

Local rephrasing, global incoherence
[Example]: What are your chances of dying in your sleep?
A1: How likely is it that you will pass away while sleeping?
A2. Can you estimate the probability of expiring in your sleep?
A3. What are the odds of dying in your slumber?
A4. How strong are the chances of meeting your maker while sleeping?
A5. Do you have a good chance of kicking the bucket while snoozing?

Hallucinated content/ extra clause
[Example]: Why do i get a bad headache when i don't eat enough during the day?
A1. Eating insufficient food can cause headaches, why is this the case?
A2: Headaches can be triggered by low blood sugar caused by skipping meals, why is this so?
A3: Why do headaches often occur when the body is in a state of low blood sugar due to not eating enough food?
A4: The lack of nutrients from not eating enough food can cause headaches, why is this the case?
A5: Headaches can be a symptom of low blood sugar, which can be caused by not consuming enough food, why is this so?

Literal (not pragmatic) rephrasing
[Example] Where did my period go??
A1: Could you tell me where my period went?
A2: I seem to have lost track of my period. Could you help me find it?
A3: Why is my period nowhere to be found?
A4: My period seems to have vanished without a trace. Do you know where it could have gone?
A5: I seem to have misplaced my period. Do you have any idea where it might be?
[Example]: Hair loss!
A1: Baldness is upon thee!
A2: Thy scalp doth reveal a lack of locks!
A3: The flow of your follicles hath ceased!
A4: Thou art bereft of thy tresses!
A5: Thy hairline doth recede, leaving naught behind!

Figure 6. *Analysis of some common issues found in the* `vicuna-13b` *generated perturbations.*

Example 3. (Validity of Perturbations). *Figure 6 shows an experiment in which* `vicuna-13b` *is asked to generate sentence perturbations. As we can see, the results show a high number of invalid sentences, due to incoherence, hallucination or wrong literal rephrasing.*

For lexical ROUGE-N, we compare the strings of the original sample to the perturbations, while for syntax we follow the same procedure, but using the corresponding parts-of-speech (POS) tags [153]. Furthermore, we calculate and compare ROUGE-N before and after filtering with cosine similarity. Results are given in Tables 31 and 32 and qualitative examples of errors in Figure 6. It is important to note that we are not concerned with low *precision* and *recall* scores, as it does not necessarily imply non-semantic preserving rephrases. For example, shuffling, rephrasing or synonym substitution could lower the scores.

1. Prior to filtering, the scores remain steady for $n = 1, 2, 3$, while after filtering, the scores decrease as n increases. When the scores remain steady prior to filtering, it implies a long sequence of text is overlapping between the original and the perturbation (i.e. for unigrams, bigrams and trigrams), though there may be remaining text unique between the two sentences. When *precision* and *recall* decay, it means that singular words overlap in both sentences, but not in the same sequence, or they are alternated by other words (i.e the high unigram overlap decaying to low trigram overlap). It is plausible that cosine similarity filters out perturbations that have long word sequence overlaps with the original, but that also contain added hallucinations that change the semantic meaning (see Figure 6, the 'Hallucinated content' example).
2. Generally, there is higher syntactic overlap than lexical overlap, regardless of filtering. Sometimes this leads to unsatisfactory perturbations, where local rephrasing leads to globally implausible sounding sentences, as shown in Figure 6 (the 'Local rephrasing, global incoherence' example).
3. Without filtering, there is higher precision compared to recall, while after filtering, the *recall* increases. From Tables 31 and 32, we can hypothesise that overall cosine similarity filters out perturbations that are shorter than the original sentences.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 31. *ROUGE-N scores comparing the original samples with vicuna perturbations (of the positive class) for lexical overlap*

		Precision				Recall			
		No filtering	Filtering			No filtering	Filtering		
Dataset	ROUGE-N		s-bert 22M	s-gpt 1.3B	s-gpt 2.7B		s-bert 22M	s-gpt 1.3B	s-gpt 2.7B
RUAR	ROUGE-1	0.500	0.568	0.545	0.537	0.281	0.635	0.612	0.604
	ROUGE-2	0.557	0.342	0.320	0.314	0.312	0.382	0.358	0.352
	ROUGE-3	0.511	0.208	0.190	0.185	0.285	0.230	0.210	0.205
Medical	ROUGE-1	0.451	0.466	0.469	0.465	0.230	0.553	0.555	0.551
	ROUGE-2	0.529	0.242	0.246	0.243	0.268	0.285	0.288	0.285
	ROUGE-3	0.471	0.131	0.135	0.133	0.238	0.156	0.159	0.157

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Table 32. *ROUGE-N scores comparing the original samples with vicuna perturbations (of the positive class) for syntax overlap*

		Precision				Recall			
		No filtering	Filtering			No filtering	Filtering		
Dataset	ROUGE-N		s-bert 22M	s-gpt 1.3B	s-gpt 2.7B		s-bert 22M	s-gpt 1.3B	s-gpt 2.7B
RUAR	ROUGE-1	0.731	0.748	0.747	0.743	0.501	0.767	0.769	0.765
	ROUGE-2	0.738	0.524	0.521	0.514	0.504	0.532	0.532	0.525
	ROUGE-3	0.710	0.350	0.347	0.340	0.483	0.349	0.346	0.339
Medical	ROUGE-1	0.670	0.674	0.678	0.676	0.410	0.710	0.714	0.712
	ROUGE-2	0.694	0.415	0.422	0.419	0.422	0.434	0.441	0.438
	ROUGE-3	0.657	0.247	0.254	0.252	0.399	0.258	0.263	0.260

Observationally, we also find instances of literal rephrasing (see Figure 6, the 'Literal (not pragmatic) rephrasing' example), which illustrates the difficulties of generating high-quality perturbations. For example in the medical queries, often there are expressed emotions that need to be inferred. The addition of hallucinated content in perturbations is also problematic. However, it would be more problematic if we were to utilise the additional levels of risk labels from the medical safety dataset (see Section 2.5.1).

the hallucinated content can have a non-trivial impact on label consistency.

5.6 Embedding error

As the final result of this paper, we introduce the new metric—*embedding error*—that measures the number of unwanted sentences that are mapped into a verified subspace. Recall that Sections 5.4 and 5.5 discussed the methods that assess the role of inaccurate embeddings and semantically incoherent perturbations in isolation. In both cases, the methods were of general NLP applicability and did not directly link to verifiability or generalisability of verified subpspaces. The embedding error metric differs from these traditional NLP methods in two aspects:

- first, it helps to measure both effects simultaneously, and thus helps to assess validity of both the assumption of locality of the embedding function and the assumption of semantic stability of the perturbations outlined at the start of Section 5.
- second, it is applied here as a verification metric specifically. Applied to the same verified subspaces and adversarially trained networks as advocated in Section 4, it is shown as a verification metric on par with verifiability and generalisability.

We next formally define the embedding error metric. Intuitively, the *embedding error* of a set of subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_l$ of class c_1 is the percentage of those subspaces that contain at least one embedding of a sentence that belongs to a different class.

Definition 9 (Embedding Error). *Given a set of subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_l$ that are supposed to contain exclusively sentences of class c_1, a dataset $\mathcal{Y}$ that contains sentences not of class c_1 and a set of embeddings V, the embedding error is measured as the percentage of subspaces that contain at least one element of V.*

$$\mathcal{F}(V, \mathcal{S}_1, \ldots, \mathcal{S}_l) = \frac{\sum_{i=1}^{l} \mathbb{I}[V \cap \mathcal{S}_i \neq \emptyset]}{l}$$

where $\mathbb{I}[\cdot]$ is the indicator function returning 1 *for true.*

As with the definition of generalisability, in this paper, we will generate the target set of embeddings V as $\bigcup_{s \in \mathcal{Y}} \mathcal{V}_t^b(s)$ where $\mathcal{Y}$ is a dataset, t is the type of semantic perturbation, b is the number of perturbations and $\mathcal{V}_t^b(s)$ is the embeddings of the set of semantic perturbations $\mathcal{A}_t^b(s)$ around s generated using $\mathcal{P}_t$, as described in Section 3.4. We also measure the presence of *false positives*, as calculated as the percentage of the perturbations of sentences from classes other than c_1 that lie within at least one of the set of subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_l$.

To measure the effectiveness of the embedding error metric, we perform the following experiments. As previously shown in Table 23, both RUAR and Medical datasets are split into two classes, *pos* and *neg*. We construct $\mathcal{A}_t^{16}(\mathcal{Y}^{pos})$ and $\mathcal{A}_t^{16}(\mathcal{Y}^{neg})$, and as described in Section 3.4, the set $\mathcal{V}_t^{16}(\mathcal{Y}^{neg})$ is obtained by embedding sentences in $\mathcal{A}_t^{16}(\mathcal{Y}^{neg})$. The subspaces for which we measure embedding error are given by $\mathbb{H}_{pert} = \mathbb{H}(\mathcal{A}_t^{16}(\mathcal{Y}^{pos}))$ where we consider both the unfiltered (t) and the filtered version ($t^{\diamond}$) of the perturbation t.

Table 24 shows the embedding error of our models and semantic subspaces. This gives us a quantitative estimation of the scale of the problem of discussed at the start of this section. Namely, whether the assumptions of the locality of the embedding function and that perturbations are semantics-preserving

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

hold. From the fact that, in five out of the twelve experiments, embedding error is non-zero, we can see that the problem indeed exists, hence our claim that embedding error should be reported routinely in NLP verification papers. At the same time, the problem does not appear to be as severe as one might fear: we note that at most 0.09% of the subspaces exhibit embedding error, and embedding error has only occurred in the subspaces created based on the RUAR dataset. Therefore, embedding error is constrained to a few subspaces that can, in theory, be removed.

Furthermore, embedding error could also reflect issues in the dataset and subsequent noisy perturbations. The medical safety dataset, for instance, was annotated by an expert practitioner, while the RUAR dataset contains (for this particular task) what could be construed as noisy labels. For example, *'are robot you you a'* is a sample that is found in the *negative* RUAR train set. The reason for the negative label is that it is an ungrammatically false positive, but given our methods of perturbation for the construction of subspaces, this *negative* sample may be very similar to a word level perturbation for the *positive* class. Concretely, for the model with the highest embedding error in Table 24 (i.e. N_{pert} s-bert 22 M for RUAR dataset with 0.09% embedding error), some sentence pairs of *negative* samples with their accompanying perturbations contained in subspaces exhibiting embedding error are: *(Original: 'Are you a chump?', Perturbation: 'You a chump'), (Original: 'Are you a liar', Perturbation: 'You a liar'), (Original: 'if a computer can feel emotions, does that make you a computer or an actual human?', Perturbation: 'if a computer can feel, does that make it a machine or a person')*. Thus, the task of determining what queries require disclosure (e.g. should 'what is your favourite food' warrant disclosure?) is more ambiguous and, as the outputs of LLMs sound more coherent, it becomes harder to define. This area merits further research.

5.6.1 Embedding error

vs Generalisability and Verifiability For comparison with the findings outlined in Section 4, we provide additional insights into verifiability and generalisability, also presented in Table 24. We first analyse the effect of cosine similarity filtering. Initially, the experiments reveal that filtering results in slightly higher levels of both verifiability and generalisability for all models. Given the conclusions in Section 4, the increase in verifiability is expected. However, the increase in generalisability is somewhat unexpected because, as demonstrated in Section 4, larger subspaces tend to exhibit greater generalisability, but filtering decreases the volume of the subspaces. Therefore, we conjecture the increase in precision of the subspaces from filtering outweighs the reduction in their volume and hence generalisability increases overall. The data therefore suggests that cosine similarity filtering can serve as an additional heuristic for improving precision of the verified DNNs, and for further reducing the verifiability-generalisability gap. Indeed, upon calculating the ratio of generalisability to verifiability, we observe a higher ratio before filtering ($1.19 \rightarrow 0.99$ for RUAR and $0.99 \rightarrow 0.95$ for Medical). Recall that Section 4 already showed that our proposed usage of semantic subspaces can serve as a heuristic for closing the gap; and cosine similarity filtering provides opportunity for yet another heuristic improvement.

Moreover, the best-performing model N_{pert} (s-bert 22M) results in 10, 964 (70.6%) medical perturbations and 9530 (21.65%) RUAR perturbations contained in the verified subspaces. While 21.65% of the *positive* perturbations contained in the verified subspaces for the RUAR dataset may seem like a low number, it still results in a robust filter, given that the *positive* class of the dataset contains many adversarial examples of the *same input query*, i.e. semantically similar but lexically different queries. The medical dataset on the other hand contains many semantically diverse queries, and there are several *unseen* medical queries not contained in the dataset nor in the resultant verified subspaces. However, given that the subspaces contain 70.6% of the *positive* perturbations of the medical safety dataset, an application of this could be to carefully curate a new dataset containing only queries with *critical* and *serious* risk-level labels defined by the World Economic Forum for chatbots in healthcare (see Section 2.5.1 and [135]). This dataset could be used to create verified filters centred around these queries to prevent generation of medical advice for these high-risk categories.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

Overall, we find that **semantically informed verification generalises well** across the different kinds of data to ensure guarantees on the output, and thus should aid in ensuring the safety of LLMs.

6. Conclusions and future work

Summary. This paper started with a general analysis of existing NLP verification approaches, with a view of identifying key components of a general NLP verification methodology. We then distilled these into a "NLP Verification Pipeline" consisting of the following six components:

1. dataset selection;
2. generation of perturbations;
3. choice of embedding functions;
4. definition of subspaces;
5. robust training;
6. verification via one of existing verification algorithms.

Based on this taxonomy, we make concrete selections for each component and implement the pipeline using the tool ANTONIO [16]. ANTONIO allowed us to mix and match different choices for each pipeline component, enabling us to study the effects of varying the components of the pipeline in a algorithm-independent way. Our main focus was to identify weak or missing parts of the existing NLP verification methodologies. We proposed that NLP verification results should report, in addition to the standard verifiability metric, the following:

- whether they use geometric or semantic subspaces, and for which type of semantic perturbations;
- volumes, generalisability and embedding error of verified subspaces.

We finished the paper with a study of the current limitations of the NLP components of the pipeline and proposed possible improvements such as introducing a perturbations filter stage using cosine similarity. One of the major strengths of the pipeline is that each component can be improved individually.

Contributions. The major discoveries of this paper were

- In Section 4, we proposed generalisability as a novel metric and showed that NLP verification methods exhibit a generalisability-verifiability trade-off. The effects of the trade-off can be severe, especially if the verified subspaces are generated naively (e.g. geometrically). We, therefore, strongly believe that generalisability should be routinely reported as part of NLP verification pipeline.
- In Sections 4 and 5, we showed that it is possible to overcome this trade-off by using several heuristic methods: defining semantic subspaces, training for semantic robustness, choosing a suitable embedding function and filtering with cosine similarity. All of these methods result in the definition of more precise verifiable subspaces, and all of them can be practically implemented as part of NLP verification pipelines in the future. This is the main positive result of this paper.
- In Section 5, we demonstrated that there are two key assumptions underlying the definition of subspaces that cannot be taken for granted. First, the LLMs, used as embedding functions, may not map semantically similar sentences to similar vectors in the embedding space. Second, our algorithmic methods for generating perturbations, whether by LLMs or otherwise, may not always be semantically-preserving operations. Both of these factors influence practical applications of the NLP verification pipeline.
- In Section 5, we demonstrated that even verified subspaces can exhibit semantic embedding errors: this effect is due to the tension between verification methods that are essentially geometric and

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

the intuitively understood semantic meaning of sentences. By defining the *embedding error* metric and using it in our experiments, we demonstrated that the effects of embedding errors do not seem to be severe in practice, but this may vary from one scenario to another. It is important that NLP verification papers are aware of this pitfall and report embedding error alongside verifiability and generalisability.

Finally, we claim as a contribution, a novel, coherent, methodological framework that allows us to include a broad spectrum of NLP, geometric, machine learning and verification methods under a single umbrella. As illustrated throughout this paper, no previous publication in this domain covered this range, and we believe that covering this broad range of methods is crucial for the development of this field.

Limitations and the Role of the Embedding Gap. In this paper, we have shown the effect of the embedding gap in the NLP verification domain. In Section 4.1, we introduced the generalisability metric (Definition 8) to estimate how well subspaces capture semantically similar yet unseen sentences, providing a quantitative lens to examine this challenge. Sections 4.3 and 4.4 demonstrated that geometric subspaces struggle with the verifiability-generalisability trade-off, while semantic subspaces, constructed using semantic-preserving perturbations, show promise in mitigating the embedding gap by better aligning with data semantics.

The embedding gap is not unique to NLP and manifests itself nearly in every domain where machine learning is applied. For example, in CV, the geometric definition of an ϵ-ball can include perturbations that no longer semantically resemble the original image (e.g. distortions that transform a given image into something unrecognisable). In the verification of neural network controllers, possible discrepancies arise between interpretation of neural network inputs, such as velocity, distance, angle (i.e. have physical interpretation) and the way in which the neural network treats them as normalised input vectors [154, 155]. In NLP, the gap is amplified by the use of LLMs to map discrete sentences into continuous vector spaces. This process lacks a one-to-one correspondence between semantics and embeddings, exacerbating the challenge.

This problem is fundamental for machine learning methods deployed in NLP: they always rely on an "embedding function" that maps sentences into real vectors (on which machine learning algorithms operate). There is an implicit assumption that the embedding function works in a way that semantic similarity of sentences is reflected in geometric proximity of their embeddings. However, as general semantic similarity of sentences is not effectively computable, there is no hope that a perfect embedding function will ever be defined. Fundamentally, this is exactly the reason why machine learning (and not symbolic) approaches to NLP proved to be successful: they operate on the assumption that the embedding function is imperfect. As a consequence, any verification pipeline for NLP must include metrics that measure potential embedding errors. Section 5.6 of this paper is entirely devoted to defining this problem in mathematically precise terms and proposing an effective metric for measuring and reporting the severity and effects of the embedding errors.

This paper aims to quantify and address the embedding gap in the NLP domain. For better quantifying the effect of the embedding gap, we proposed precise metrics, such as verifiability, generalisability, and embedding error, and showed their interplay. This better understanding of the problem gave rise to our main positive result: the method that empirically reduces the effect of the embedding gap. While our findings mark progress, further research is needed to better align geometric representations with semantic meaning, especially in NLP contexts.

Future Work. Following from our in-depth analysis of the NLP perspective, we note that even if one has a satisfactory solution to all the issues discussed, there is still the problem of scalability of the available verification algorithms. For example, the most performant neural network verifier, $\alpha\beta$-Crown [43], can only handle networks in the range of tens of millions of trainable parameters. In contrast, in NLP systems, the base model of BERT [139] has around 110 million trainable parameters (considered small compared to modern LLMs—with trainable parameters in the billions!). It is clear that the rate at which DNN verifiers become more performant may never catch up with the rate at which LLMs become larger. Then the question arises: how can this pipeline be implemented in the real world?

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

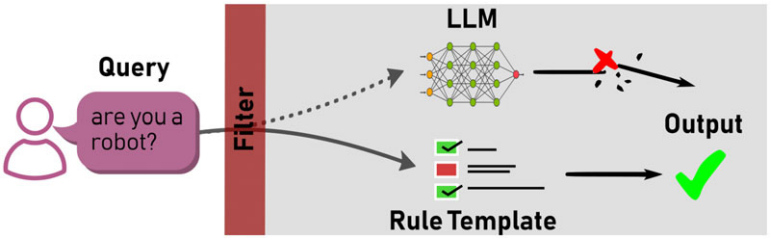


Figure 7. *In this figure, we show how a prepended, semantically informed verified filter added to an NLP system (here, an LLM), can check that safety-critical queries are handled responsibly, e.g. by redirecting the query to a tightly controlled rule-based system instead of a stochastic LLM.*

For future work, we propose to tackle this based on the idea of verifying a smaller DNN (classifier), manageable by verifiers, that can be placed upstream of a complex NLP system as a *safeguard*. We call this a *filter* (as mentioned in Section 3 and illustrated in Figure 2), and Figure 7 shows how a semantically informed verified filter can be prepended to an NLP system (here, an LLM) to check that safety-critical queries are handled responsibly, e.g. by redirecting the query to a tightly controlled rule-based system instead of a stochastic LLM. While there are different ways to implement the verification filters (e.g. only the verified subspaces), we suggest utilising both the verified subspaces together with the DNN as the additional classification could strengthen catching positives that fall outside the verified subspaces, thus giving stronger chances of detecting the query via both classification and verification.

We note that the NLP community has recently proposed guardrails, in order to control the output of LLMs and create safer systems (such as from Open AI, NVIDIA and so on). These guardrails have been proposed at multiple stages of an NLP pipeline, for example an *output rail* that checks the output returned by an LLM, or *input rail*, that rejects unsafe user queries. In Figure 7, we show an application of our filter applied to the user input, which thus creates guarantees that a subset of safety critical queries are handled responsibly. In theory, these verification techniques we propose may be applied to guardrails at different stages in the system, and we plan to explore this in future work.

A second future direction is to use this work to create NLP verification benchmarks. In 2020, the International Verification of Neural Networks Competition [156] (VNN-COMP) was established to facilitate comparison between existing approaches, bring researchers working on the DNN verification problem together and help shape future directions of the field. However, for some time, the competition still lacked NLP verification benchmarks [157]. In 2024, we contributed a first NLP benchmark to VNN-COMP, using the methodology of this paper, and the tool ANTONIO [158]. We intend to use this work for creating NLP verification benchmarks for future editions, to spread the awareness and attention to this field.

Acknowledgements. The authors acknowledge Vinay Krupakaran for his help in the manual annotation of the perturbed sentences and Daniel Kienitz for his help with the eigenspace rotation.

Funding. The authors acknowledge support of EPSRC grant AISEC EP/T026952/1 and NCSC grant 'Neural Network Verification: in search of the missing spec'. The first author acknowledges the James Watt Scholarship awarded by the Heriot-Watt University.

Competing interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Ren, S., He, K., Girshick, R. & Sun, J. (2016) Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach.* **39**(6), 1137–1149.

[2] Sutskever, I., Vinyals, O. & Le, Q. V. (2014) Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, 27.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[3] Hirschberg, J. & Manning, C. D. (2015) Advances in natural language processing. *Science* **349**(6245), 261–266.
[4] Bender, E. M., Gebru, T., McMillan-Major, A. & Shmitchell, S. (2021) On the dangers of stochastic parrots: Can language models be too big? In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 610–623.
[5] Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P. S., ... & Gabriel, I. (2021). Ethical and social risks of harm from language models. arXiv preprint arXiv:2112.04359.
[6] Bergman, A. S., Abercrombie, G., Spruit, S. L., Hovy, D., Emily Dinan, Y.-L. B. & Rieser, V. (2022) Guiding the release of safer e2e conversational ai through value sensitive design. In Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pp. 39–52.
[7] Dinan, E., Gavin Abercrombie, A. B., Spruit, S., Dirk Hovy, Y.-L. B. & Rieser, V. (2022) SafetyKit: First aid for measuring safety in open-domain conversational systems, Association for Computational Linguistics. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, pp. 4113–4133, May, https://doi.org/10.18653/v1/2022.acl-long.284. URL https://aclanthology.org/2022.acl-long.284
[8] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.
[9] Dinan, E., Gavin Abercrombie, A. S. B., Spruit, S., Dirk Hovy, Y.-L. B. & Rieser, V. (2021) Anticipating safety issues in E2E conversational AI: Framework and tooling. URL https://arxiv.org/abs/2107.03451
[10] Kop, M. (2021) Eu artificial intelligence act: The european approach to ai. URL. *Available at:* https://futurium.ec.europa.eu/sites/default/files/2021-10/Kop_EUArtificialIntelligenceAct-TheEuropeanApproachtoAI_21092021_0.pdf.
[11] California State Legislature (2018) California senate bill no. 1001, URL, https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1001
[12] Abercrombie, G. & Rieser, V. (2022) Risk-graded safety for handling medical queries in conversational ai. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing, pp. 234–243.
[13] Bickmore, T. W., Trinh, H., Olafsson, S., O'Leary, T. K., Asadi, R., Rickles, N. M. & Cruz, R. (2018) Patient and consumer safety risks when using conversational assistants for medical information: An observational study of siri, alexa, and google assistant. *J. Med. Internet Res.* **20**(9), e11510.
[14] Chiang, W.-L., Li, Z., Lin, Z. (2023). Vicuna: An open-source chatbot impressing gpt-4 with 90%∗ chatgpt quality. Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing, March, URL. *Available at:* https://lmsys.org/blog/2023-03-30-vicuna/
[15] Shi, Z., Zhang, H., Chang, K. W., Huang, M., & Hsieh, C. J. (2020). Robustness verification for transformers. arXiv preprint arXiv: 2002.06622.
[16] Casadio, M., Arnaboldi, L., Daggitt, M., Isac, O., Dinkar, T., Kienitz, D., Rieser, V. & Komendantskaya, E. (2023) Antonio: Towards a systematic method of generating nlp benchmarks for verification. In Nina Narodytska, Guy Amir, Guy Katz, and Omri Isac, editors,Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems, Vol. 16 of Kalpa Publications in Computing, pp. 59–70. DOI: http://dx.doi.org/10.29007/7wxb, https://easychair.org/publications/paper/9ZGS EasyChair, URL
[17] Jia, R., Raghunathan, A., Göksel, K. & Liang, P. (2019a) Certified robustness to adversarial word substitutions. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4129–4142.
[18] Huang, P.-S., Stanforth, R., Welbl, J., Dyer, C., Yogatama, D., Gowal, S., Dvijotham, K. & Kohli, P. (2019) Achieving verified robustness to symbol substitutions via interval bound propagation. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4083–4093.
[19] Zhang, Y., Albarghouthi, A. & D'Antoni, L. (2021) Certified robustness to programmable transformations in lstms . In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 1068–1083.
[20] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. (2018) Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations.
[21] Wu, T., Ribeiro, M. T., Heer, J. & Weld, D. S. (2021) Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 6707–6723.
[22] Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M., Kokke, W., ... & Barrett, C. (2024, July). Marabou 2.0: a versatile formal analyzer of neural networks. In *International Conference on Computer Aided Verification*, pp. 249–264. Cham: Springer Nature Switzerland.
[23] Kroening, D. & Paul, W. (2001) Automated pipeline design. In Proc. of 38th ACM/IEEE Design Automation Conference (DAC 2001), pp. 810–815. ACM Press.
[24] Patankar, V. A., Jain, A. & Bryant, R. E. (1999) Formal verification of an arm processor. In Proceedings Twelfth International Conference on VLSI Design. (Cat. No. PR00013), pp. 282–287, IEEE.
[25] Jourdan, J.-H., Laporte, V., Blazy, S., Leroy, X. & Pichardie, D. (2015) A formally-verified c static analyzer. *ACM SIGPLAN Notices* **50**(1), 247–259.
[26] Metere, R. & Arnaboldi, L. (2022) Automating cryptographic protocol language generation from structured specifications. In Proceedings of the IEEE/ACM 10th International Conference on Formal Methods in Software Engineering, pp. 91–101.
[27] Woodcock, J., Larsen, P. G., Bicarregui, J. & Fitzgerald, J. (2009) Formal methods: Practice and experience. *ACM Comput. Surv.* **41**(4), 1–36.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[28] Katz, G., Barrett, C., Dill, D. L., Julian, K. & Kochenderfer, M. J. (2017) Reluplex: An efficient smt solver for verifying deep neural networks. In International conference on computer aided verification, pp. 97–117. Springer.

[29] Bak, S., Liu, C. & Johnson, T. (2021) The second international verification of neural networks competition (vnn-comp 2021): Summary and results, arXiv preprint arXiv: 2109.00498

[30] Baluta, T., Chua, Z. L., Meel, K. S. & Saxena, P. (2021) Scalable quantitative verification for deep neural networks. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 312–323. IEEE.

[31] Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C. & Kochenderfer, M. J. (2021) Algorithms for verifying deep neural networks. *Found. Trends® Optim.* **4**(3-4), 244–404.

[32] Singh, G., Ganvir, R., Püschel, M. & Vechev, M. (2019a) Beyond the single neuron convex barrier for neural network certification. *Adv. Neural Inf. Process. Syst.* **32.**

[33] Barrett, C. & Tinelli, C. (2018) Satisfiability modulo theories. *Handbook of Model Checking*, pp. 305–343.

[34] Albarghouthi, A. (2021) Introduction to neural network verification. *Found. Trends® Program. Lang.* **7**(1–2), 1–157.

[35] Dantzig, G. (1963) *Linear Programming and Extensions*, Princeton University Press.

[36] Winston, W. L. (2004) *Operations Research: Applications and Algorithm*, Thomson Learning, Inc.

[37] Cheng, C.-H., Nührenberg, G. & Ruess, H. (2017) Maximum resilience of artificial neural networks. In Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India: Springer, pp. 251–268, October 3-6, 2017, Proceedings 15

[38] Lomuscio, A. & Maganti, L. (2017) An approach to reachability analysis for feed-forward relu neural networks, arXiv preprint arXiv: 1706.07351

[39] Tjeng, V., Xiao, K. & Tedrake, R. (2019) Evaluating robustness of neural networks with mixed integer programming. In International Conference on Learning Representations.

[40] Singh, G., Gehr, T., Mirman, M., Püschel, M. & Vechev, M. (2018). Fast and effective robustness certification. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N. & Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, Vol. 31, Curran Associates, Inc, URL, https://proceedings.neurips.cc/paper/2018/file/f2f446980d8e971ef3da97af089481c3-Paper.pdf

[41] LLC Gurobi Optimization (2020) Gurobi: Gurobi optimizer 9.1 interface. R package version, pages 9-1

[42] Xu, K., Zhang, H., Wang, S., Wang, Y.Jana, S., Lin, X. & Hsieh, C. J. (2021) Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In International Conference on Learning Representation (ICLR).

[43] Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh,C. J. & Kolter, Z. (2021a) Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Adv. Neural Inf. Process. Syst.* **34**, 29909–29921.

[44] Cousot, P. & Cousot, R. (1977) Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp. 238–252.

[45] Cousot, P. (2003) Verification: Theory and Practice, *Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, pp. 243–268. Springer, Verification by abstract interpretation

[46] Cousot, P. & Cousot, R. (2014) Abstract interpretation: past, present and future. In Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–10.

[47] Wong, E. & Kolter, Z. (2018) Provable defenses against adversarial examples via the convex outer adversarial polytope. In International conference on machine learning, pp. 5286–5295. PMLR.

[48] Gowal, S., Dvijotham, K. D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T. & Kohli, P. (2019). Scalable verified training for provably robust image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4842–4851.

[49] Lyu, Z., Ko, C.-Y., Kong, Z., Wong, N., Lin, D. & Daniel, L. (2020) Fastened crown: Tightened neural network robustness certificates. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. **34**, pp. 5037–5044.

[50] Mirman, M., Gehr, T. & Vechev, M. (2018) Differentiable abstract interpretation for provably robust neural networks. In International Conference on Machine Learning, pp. 3578–3586.PMLR.

[51] Li, L., Xie, T. & Li, B. (2023) Sok: Certified robustness for deep neural networks. In 2023 IEEE symposium on security and privacy (SP), pp. 1289–1310. IEEE.

[52] Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J. & Daniel, L. (2018) Efficient neural network robustness certification with general activation functions. *Adv. Neural Inf. Process. Syst.* **31**.

[53] Singh, G., Gehr, T., Püschel, M. & Vechev, M. (2019b) An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.* **3**, 1–30.(POPL)

[54] Müller, M. N., Makarchuk, G., Singh, G., Püschel, M. & Vechev, M. T. (2022) Prima: General and precise neural network certification via scalable convex hull approximations. *Proc. ACM Program. Lang* **6**, 1–33.(POPL),

[55] Althoff, M. (2015) An introduction to CORA 2015. In Proc. of the 1st and 2nd Workshop on Applied Verification for Continuous and Hybrid Systems, 120–151. DOI: 10.29007/zbkv EasyChair, December URL, https://easychair.org/publications/paper/xMm

[56] Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S. & Vechev, M. (2018) Ai2: Safety and robustness certification of neural networks with abstract interpretation. In 2018 IEEE symposium on security and privacy (SP), pp. 3–18. IEEE.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[57] Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P. & Mudigonda, P. K. (2018) A unified view of piecewise linear neural network verification. *Adv. Neural Inf. Process. Syst.* **31**.
[58] Bunel, R., Mudigonda, P., Turkaslan, I., Torr, P., Lu, J. & Kohli, P. (2020) Branch and bound for piecewise linear neural network verification. *J. Mach. Learn. Res.* **21**(2020).
[59] Ferrari, C., Mueller, M. N., Jovanović, N. & Vechev, M. (2022) Complete verification via multi-neuron relaxation guided branch-and-bound. In International Conference on Learning Representations. URL https://openreview.net/forum?id=l_amHf1oaK
[60] Jordan, M., Lewis, J. & Dimakis, A. G. (2019) Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. *Adv. Neural Inf. Process. Syst.* **32**.
[61] Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hseih, C. J. & Kolter, Z. (2022) General cutting planes for bound-propagation-based neural network verification. *Adv. Neural Inf. Process. Syst.* **35**, 1656–1670.
[62] Müller, M. N., Makarchuk, G., Singh, G., Püschel, M., & Vechev, M. (2021). Prima: Precise and general neural network certification via multi-neuron convex relaxations. arXiv preprint arXiv: 2103.03638.
[63] Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D. & Jana, S. (2019) Certified robustness to adversarial examples with differential privacy. In 2019 IEEE symposium on security and privacy (SP), pp. 656–672. IEEE.
[64] Li, B., Chen, C., Wang, W. & Carin, L. (2019a) Certified adversarial robustness with additive noise. *Adv. Neural Inf. Process. Syst.* **32**.
[65] Dvijotham, K. D., Hayes, J., Balle, B., Kolter, Z., Qin, C., Gyorgy, A., ... & Kohli, P. (2020). A framework for robustness certification of smoothed classifiers using f-divergences. In *International Conference on Learning Representations*.
[66] Zhang, D., Ye, M., Gong, C., Zhu, Z. & Liu, Q. (2020a) Black-box certification with randomized smoothing: A functional optimization based framework. *Adv. Neural Inf. Process. Syst.* **33**, 2316–2326.
[67] Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S. & Yang, G. (2019) Provably robust deep learning via adversarially trained smoothed classifiers. *Adv. Neural Inf. Process. Syst.* **32**.
[68] Mohapatra, J., Ko, C.-Y., Weng, T.-W., Chen, P.-Y., Liu, S. & Daniel, L. (2020) Higher-order certification for randomized smoothing. *Adv. Neural Inf. Process. Syst.* **33**, 4501–4511.
[69] Singh, G., Gehr, T., Püschel, M., & Vechev, M. (2019). An abstract domain for certifying neural networks. *Proc. ACM Program. Lang*. **3**(POPL), 1–30.
[70] Casadio, M., Komendantskaya, E., Daggitt, M. L., Kokke, W., Katz, G., Amir, G. & Rafaeli, I. (2022) Neural network robustness as a verification property: A principled case study. In *Computer Aided Verification (CAV 2022)*, Lecture Notes in Computer Science. Springer.
[71] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. arXiv preprint arXiv: 1412.6572.
[72] Kolter, Z. & Madry, A. (2018) Adversarial robustness: Theory and practice.*Tutorial at NeurIPS,* page 3
[73] Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O. & Mann, T. A. (2021) Data augmentation can improve robustness. *Adv. Neural Inf. Process. Syst.* **34**, 29935–29948.
[74] Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C. & Vechev, M. T. (2019) DL2: training and querying neural networks with logic. In Chaudhuri, K. & Salakhutdinov, R.(eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA, Vol. 97 of Proceedings of Machine Learning Research, pp. 1931–1941, PMLR. 2019. 9-15 June 2019, URL, http://proceedings.mlr.press/v97/fischer19a.html
[75] Slusarz, N., Komendantskaya, E., Daggitt, M. L., Stewart, R. J. & Stark, K. (2023) Logic of differentiable logics: Towards a uniform semantics of DL. In Piskac, R. & Voronkov, A.(eds.), LPAR 2023: Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Manizales, Colombia, Vol. 94 of EPiC Series in Computing, pp. 473–493, EasyChair, 2023. DOI: http://dx.doi.org/10.29007/C1NT URL https://doi.org/10.29007/c1nt
[76] Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D. & Hsieh, C. J. (2020b). Towards stable and efficient training of verifiably robust neural networks. In 8th International Conference on Learning Representations, ICLR 2020.
[77] Müller, M. N., Eckert, F., Fischer, M., & Vechev, M. (2023). Certified training: Small boxes are all you need. arXiv preprint arXiv: 2210.04871.
[78] Zhang, Y., Albarghouthi, A. & D'Antoni, L. (2020c) Robustness to programmable string transformations via augmented abstract training. In Proceedings of the 37th International Conference on Machine Learning, pp. 11023–11032.
[79] Zhang, W. E., Sheng, Q. Z., Alhazmi, A. & Li, C. (2020d) Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol. (TIST)* **11**(3), 1–41.
[80] Wang, W., Wang, R., Wang, L., Wang, Z. & Ye, A. (2021b) Towards a robust deep neural network against adversarial texts: A survey. *IEEE Trans. Knowl. Data Eng.* 1–1.
[81] Wang, X., Wang, H. & Yang, D. (2022) Measure and improve robustness in nlp models: A survey. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4569–4586.
[82] Li, Z., Xu, J., Zeng, J., Li, L., Zheng, X., Zhang, Q., Chang, K. W. & Hsieh, C. J. (2021) Searching for an effective defender: Benchmarking defense against adversarial word substitution. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3137–3147.
[83] Zhou, Y., Zheng, X., Hsieh, C.-J., Chang, K.-W. & Huan, X. (2021) Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. In Association for Computational Linguistics (ACL).
[84] Zhu, C., Cheng, Y., Gan, Z., Sun, S., Goldstein, T. & Liu, J. (2019) Freelb: Enhanced adversarial training for natural language understanding. In International Conference on Learning Representations.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[85] Dong, X., Luu, A. T., Ji, R. & Liu, H. (2021) Towards robustness against natural language word substitutions, arXiv preprint arXiv: 2107.13541
[86] Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T. & Hovy, E. (2021). A survey of data augmentation approaches for NLP, association for computational linguistics. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 968–988, Online 2021. DOI: 10.18653/v1/2021.findings-acl.84 URL https://aclanthology.org/2021.findings-acl.84
[87] Dhole, K. D., Gangal, V., Gehrmann, S., Gupta, A., Li, Z., Mahamood, S., ... & Zhang, Y. (2021). Nl-augmenter: A framework for task-sensitive natural language augmentation. arXiv preprint arXiv: 2112.02721.
[88] Cheng, Y., Jiang, L. & Macherey, W. (2019) Robust neural machine translation with doubly adversarial inputs. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4324–4333.
[89] Iyyer, M., Wieting, J., Gimpel, K. & Zettlemoyer, L. (2018) Adversarial example generation with syntactically controlled paraphrase networks. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 1875–1885, (Long Papers)
[90] Cao, Y., Li, D., Fang, M., Zhou, T., Gao, J., Zhan, Y. & Tao, D. (2022) Tasa: Deceiving question answering models by twin answer sentences attack. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 11975–11992.
[91] Liang, B., Li, H., Su, M., Bian, P., Li, X. & Shi, W. (2017) Deep text classification can be fooled, arXiv preprint arXiv: 1704.08006
[92] Ebrahimi, J., Rao, A., Lowd, D. & Dou, D. (2018) Hotflip: White-box adversarial examples for text classification, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Vol. **2**, 31–36.(Short Papers)
[93] Lei, Y., Cao, Y., Li, D., Zhou, T., Fang, M. & Pechenizkiy, M. (2022) Phrase-level textual adversarial attack with label preservation. In Findings of the Association for Computational Linguistics: NAACL, pp. 1095–1112.
[94] Belinkov, Y. & Bisk, Y. (2017) Synthetic and natural noise both break neural machine translation, arXiv preprint arXiv: 1711.02173.
[95] Gao, J., Lanchantin, J., Soffa, M. L. & Qi, Y. (2018) Black-box generation of adversarial text sequences to evade deep learning classifiers. In 2018 IEEE Security and Privacy Workshops (SPW), pp. 50–56.IEEE.
[96] Li, J., Ji, S., Du, T., Li, B. & Wang, T. (2019b) Textbugger: Generating adversarial text against real-world applications. In Proceedings 2019 Network and Distributed System Security Symposium.
[97] Samanta, S., & Mehta, S. (2017). Towards crafting text adversarial samples. arXiv preprint arXiv: 1707.02812.
[98] Ivankay, A., Girardi, I., Marchiori, C. & Frossard, P. (2022) Fooling explanations in text classifiers, arXiv preprint arXiv: 2206.0317
[99] Di, J., Jin, Z., Zhou, J. T. & Szolovits, P. (2020) Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In Proceedings of the AAAI conference on artificial intelligence, Vol. **34**, pp. 8018–8025.
[100] Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M. & Chang, K.-W. (2018) Generating natural language adversarial examples. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2890–2896.
[101] Kuleshov, V., Thakoor, S., Lau, T. & Ermon, S. (2018) Adversarial examples for natural language classification problems.
[102] Pennington, J., Socher, R. & Manning, C. D. (2014) Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543.
[103] Moradi, M. & Samwald, M. (2021) Evaluating the robustness of neural language models to input perturbations. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 1558–1570.
[104] Jia, R. & Liang, P. (2017) Adversarial examples for evaluating reading comprehension systems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2021–2031.
[105] Wang, Y. & Bansal, M. (2018) Robust machine comprehension models via adversarial training. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. **2**, pp. 575–581, (Short Papers)
[106] Wang, B., Xu, C., Wang, S. (2021c) Adversarial glue: A multi-task benchmark for robustness evaluation of language models, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li, arXiv preprint arXiv: 2111.02840
[107] Welbl, J., Huang, P. S., Stanforth, R., Gowal, S., Dvijotham, K. D., Szummer, M., & Kohli, P. (2020). Towards verified robustness under text deletion interventions. In *International Conference on Learning Representations*.
[108] Wang, Y., Yang, Y., Di, H. & He, K. (2023) Robustness-aware word embedding improves certified robustness to adversarial word substitutions. In Findings of the Association for Computational Linguistics: ACL 2023, pp. 673–687.
[109] Ko, C.-Y., Lyu, Z., Weng, L., Daniel, L., Wong, N. & Lin, D. (2019) Popqorn: Quantifying robustness of recurrent neural networks. In International Conference on Machine Learning, pp. 3468–3477. PMLR.
[110] Du, T., Ji, S., Shen, L., Zhang, Y., Li, J., Shi, J., Fang, C., Yin, J., Beyah, R. & Wang, T. (2021) Cert-rnn: Towards certifying the robustness of recurrent neural networks. *CCS* **21**(2021), 15–19.
[111] Bonaert, G., Dimitrov, D. I., Baader, M. & Vechev, M. (2021) Fast and precise certification of transformers. In Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 466–481.
[112] Ye, M., Gong, C. & Liu, Q. (2020) Safer: A structure-free approach for certified robustness to adversarial word substitutions. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3465–3475.
[113] Wang, W., Tang, P., Lou, J., & Xiong, L. (2021, June). Certified robustness to word substitution attack with differential privacy. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1102–1112.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[114] Zhao, H., Ma, C., Dong, X., Luu, A. T., Deng, Z.-H. & Zhang, H. (2022) Certified robustness against natural language attacks by causal intervention. In International Conference on Machine Learning, pp. 26958–26970. PMLR.
[115] Zeng, J., Xu, J., Zheng, X. & Huang, X. (2023) Certified robustness to text adversarial attacks by randomized [mask]. *Comput. Linguist.* **49**(2), 395–427.
[116] Ye, M., Yin, Z., Zhang, T., Du, T., Chen, J., Wang, T. & Ma, F. (2023) Unit: A unified look at certified robust training against text adversarial perturbation. In Thirty-seventh Conference on Neural Information Processing Systems.
[117] Zhang, X., Hong, H., Hong, Y., Huang, P., Wang, B., Ba, Z. & Ren, K. (2023a) Text-crs: A generalized certified robustness framework against textual adversarial attacks. In 2024 IEEE Symposium on Security and Privacy (SP), pp. 53–53.IEEE Computer Society.
[118] Zhang, Z., Zhang, G., Hou, B. (2023b) Certified robustness for large language models with self-denoising, Wenqi Fan, Qing Li, Sijia Liu, Yang Zhang, and Shiyu Chang, arXiv preprint arXiv: 2307.07171
[119] Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D. (2015) A large annotated corpus for learning natural language inference, Association for Computational Linguistics. In Màrquez, L., Callison-Burch, C. & Su, J.(eds.), Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 632–642, Lisbon, Portugal, September, URL, DOI: 10.18653/v1/D15-1075, https://aclanthology.org/D15-1075
[120] Adam, C. J.2017). Toxic comment classification challenge. Jeffrey Sorensen Julia Elliott Lucas Dixon Mark McDonald nithum and Will Cukierski. URL. *Available at:* https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge
[121] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. (2011) Learning word vectors for sentiment analysis. In Lin, D., Matsumoto, Y. & Mihalcea, R. (eds.), Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150, Portland, Oregon, USA, Association for Computational Linguistics. URL, https://aclanthology.org/P11-1015,
[122] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. & Potts, C. (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K. & Bethard, S. (eds.), Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642, Seattle, Washington, USA, October, Association for Computational Linguistics. URL, https://aclanthology.org/D13-1170,
[123] Shen, T., Lei, T., Barzilay, R. & Jaakkola, T. (2017) Style transfer from non-parallel text by cross-alignment. *Adv. Neural Inf. Process. Syst.* **30**
[124] Pang, B. & Lee, L. (2005) Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, Association for Computational Linguistics. In Knight, K., Ng, H. T. & Oflazer. K. (eds.), Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pp. 115–124, Michigan: Ann Arbor. June, DOI: 10.3115/1219840.1219855, URL https://aclanthology.org/P05-1015
[125] McAuley, J. & Leskovec, J. (2013) Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM conference on Recommender systems, pp. 165–172.
[126] Williams, A., Nangia, N. & Bowman, S. (2018) A broad-coverage challenge corpus for sentence understanding through inference, Association for Computational Linguistics. In Walker, M., Ji, H. & Stent, A. (eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. **1**, pp. 1112–1122, New Orleans, Louisiana, June, (Long Papers), URL, DOI: 10.18653/v1/N18-1101, https://aclanthology.org/N18-1101
[127] Zhang, X., Zhao, J. & LeCun, Y. (2015) Character-level convolutional networks for text classification. *Adv. Neural Inf. Process. Syst.* **28**.
[128] Li, X. & Roth, D. (2002) Learning question classifiers. In COLING 2002: The 19th International Conference on Computational Linguistics. URL https://aclanthology.org/C02-1150
[129] Gros, D., Li, Y. & Yu, Z. (2021) The rua-robot dataset: Helping avoid chatbot deception by detecting user questions about human or non-human identity. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 6999–7013.
[130] Altman, S., Marcus, G., & Montgomery, C. (2023). Oversight of AI: Rules for Artificial Intelligence. Available at: https://www.ibm.com/policy/wp-content/uploads/2023/05/Christina-Montgomery-Senate-Judiciary-Testimony-5-16-23.pdf.
[131] Atleson, M. (2023) Chatbots, deepfakes, and voice clones: AI deception for sale-federal trade commission. *Available at:* https://www.ftc.gov/business-guidance/blog/2023/03/chatbots-deepfakes-voice-clones-ai-deception-sale.
[132] Abercrombie, G., Curry, A. C., Dinkar, T., Rieser, V. & Talat, Z. (2023) Mirages. on anthropomorphism in dialogue systems. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 4776–4790.
[133] Leviathan, Y., & Matias, Y. (2018). Google Duplex: An AI system for accomplishing real-world tasks over the phone. Google AI blog, 8.
[134] Lieu, J. (2018) Google's creepy AI phone call feature will disclose it's a robot, after backlash. *Available at:* https://mashable.com/2018/05/11/google-duplex-disclosures-robot.
[135] Forum, W. E. (2020) Chatbots RESET: A framework for governing responsible use of conversational AI in healthcare. *Available at:* https://www.weforum.org/publications/chatbots-reset-a-framework-for-governingresponsible-use-of-conversational-ai-in-healthcare/.
[136] Ghorbal, K., Goubault, E. & Putot, S. (2009) The zonotope abstract domain taylor1+. In Computer Aided Verification: 21st International Conference, CAV 2009, Springer, Grenoble, France, pp. 627–633, June 26-July 2,, Proceedings 21
[137] Cohen, J., Rosenfeld, E. & Kolter, Z. (2019) Certified adversarial robustness via randomized smoothing. In International conference on machine learning, pp. 1310–1320. PMLR.

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press

[138] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv: 2302.13971.
[139] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies*, volume 1 (long and short papers), pp. 4171–4186.
[140] Reimers, N. & Gurevych, I. (2019) Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992, Hong Kong, China: November, Association for Computational Linguistics, URL, DOI: 10.18653/v1/D19-1410, https://aclanthology.org/D19-1410.
[141] Muennighoff, N. (2022). Sgpt: Gpt sentence embeddings for semantic search. arXiv preprint arXiv: 2202.08904.
[142] Kugler, K., Münker, S., Höhmann, J. & Rettinger, A. (2023) Invbert: Reconstructing text from contextualized word embeddings by inverting the bert pipeline. *JCLS*. **2**(1), 1–18.
[143] Barber, C. B., Dobkin, D. P. & Huhdanpaa, H. (1996) The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* **22**(4), 469–483.
[144] Sartipizadeh, H., & Vincent, T. L. (2016). Computing the approximate convex hull in high dimensions. arXiv preprint arXiv: 1603.04422.
[145] Jia, Y., Wang, H., Shao, S., Long, H., Zhou, Y., & Wang, X. (2019). On geometric structure of activation spaces in neural networks. arXiv preprint arXiv: 1904.01399.
[146] Klema, V. & Laub, A. (1980) The singular value decomposition: Its computation and some applications. *IEEE Trans. Autom. Control* **25**(2), 164–176.
[147] Beeching, E., Han, S., ans Nazneen Rajani, N. L., Sanseviero, O., Tunstall, L. & Wolf, T. (2023). Open llm leaderboard. *Available at:* https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.
[148] Lin, S., Hilton, J. & Evans, O. (2022) Truthfulqa: Measuring how models mimic human falsehoods. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3214–3252.
[149] Wang, W., Bao, H., Huang, S., Dong, L. & Wei, F. (2021). Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers, *In, Findings of the Association for Computational Linguistics: ACL-IJCNLP*, 2140–2151, *2021*
[150] Yu, L., & Rieser, V. (2022). Adversarial robustness of visual dialog. arXiv preprint arXiv: 2207.02639.
[151] Liljequist, D., Elfving, B. & Roaldsen, K. S. (2019) Intraclass correlation—A discussion and demonstration of basic features. *PloS one* **14**(7), e0219854.
[152] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries, Barcelona, Spain, pp. 74–81, Text Summarization Branches Out, Association for Computational Linguistics, July, URL, https://aclanthology.org/W04-1013
[153] Vasiliev, Y. (2020) *Natural Language Processing with Python and spaCy: A Practical Introduction*. No Starch Press.
[154] Daggitt, M. L., Kokke, W., Atkey, R., Slusarz, N., Arnaboldi, L. & Komendantskaya, E. (2024) Vehicle: Bridging the embedding gap in the verification of neuro-symbolic programs, arXiv preprint arXiv: 2401.06379.
[155] Cordeiro, L. C., Daggitt, M. L., Girard-Satabin, J., Isac, O., Johnson, T. T., Katz, G., ... & Wu, H. (2025). Neural Network Verification is a Programming Language Challenge. arXiv preprint arXiv: 2501.05867.
[156] Brix, C., Müller, M. N., Bak, S., Johnson, T. T. & Liu, C. (2023a) First three years of the international verification of neural networks competition (vnn-comp). *Int. J. Softw. Tools Technol. Transf.* **25**(3), 329–339.
[157] Brix, C., Bak, S., Liu, C. & Johnson, T. T. (2023b) The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results, arXiv preprint arXiv: 2312.16760.
[158] Brix, C., Bak, S., Johnson, T. T. & Wu, H. (2024) The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results, arXiv preprint arXiv: 2412.19985.

Cite this article: Casadio M., Dinkar T., Komendantskaya E., Arnaboldi L., Daggitt M.L., Isac O., Katz G., Rieser V. and Lemon O. (2026). NLP verification: towards a general methodology for certifying robustness. *European Journal of Applied Mathematics*, **37**, 180–237. https://doi.org/10.1017/S0956792525000099

https://doi.org/10.1017/S0956792525000099 Published online by Cambridge University Press