Kalpa Publications in Computing

Volume 16, 2023, Pages 59–70

Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems

ANTONIO: Towards a Systematic Method for Generating NLP Benchmarks for Verification

Marco Casadio[1], Luca Arnaboldi[2]*, Matthew L. Daggitt[1], Omri Isac[3], Tanvi Dinkar[1], Daniel Kienitz[1], Verena Rieser[1], and Ekaterina Komendantskaya[1]

[1] Heriot-Watt University, Edinburgh, UK
{mc248,md2006,t.dinkar,dk50,v.t.rieser,e.komendantskaya}@hw.ac.uk

[2] University of Birmingham, Birmingham, UK
l.arnaboldi@bham.ac.uk

[3] The Hebrew University of Jerusalem, Jerusalem, Israel
omri.isac@mail.huji.ac.il

Abstract

Verification of machine learning models used in Natural Language Processing (NLP) is known to be a hard problem. In particular, many known neural network verification methods that work for computer vision and other numeric datasets do not work for NLP. Here, we study technical reasons that underlie this problem. Based on this analysis, we propose practical methods and heuristics for preparing NLP datasets and models in a way that renders them amenable to state-of-the-art verification methods. We implement these methods as a Python library called ANTONIO that links to the neural network verifiers ERAN and Marabou. We perform evaluation of the tool using an NLP dataset R-U-A-Robot suggested as a benchmark for verifying legally critical NLP applications. We hope that, thanks to its general applicability, this work will open novel possibilities for including NLP verification problems into neural network verification competitions, and will popularise NLP problems within this community.

1 Introduction

Deep neural networks (DNNs) are adept at addressing challenging problems in various areas, such as Computer Vision (CV) [24] and Natural Language Processing (NLP) [29, 10]. Due to their success, systems based on DNNs are widely deployed in the physical world and their safety and security is a critical matter [3, 4, 7].

One example of a safety critical application in the NLP domain is a chatbot's responsibility to identify itself as an AI agent, *when asked by the user to do so*. Recently there has been several pieces of legislation proposed that will enshrine this requirement in law [14, 15]. For the chatbot to be compliant with these new laws, the DNN, or the subsystem responsible for identifying these queries, must be 100% accurate in its recognition of the user's question. Yet, in reality the questions can come in different forms, for example: *"Are you a Robot?"*, *"Am I speaking with a person?"*, *"Hey,*

*Large portion of work undertaken whilst at University of Edinburgh

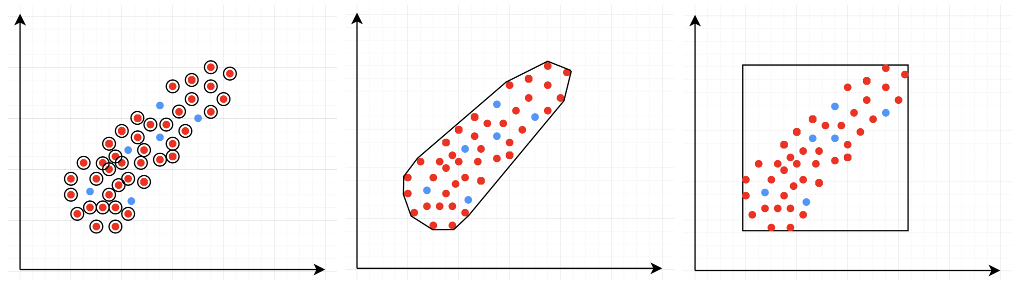

Figure 1: *An example of ϵ-balls (left), convex-hull (centre) and hyper-rectangle (right) in 2-dimensions. The red dots represent sentences in the embedding space from the training set belonging to one class, while the torquoise dots are embedded sentences from the test set belonging to the same class.*

are you a human or a bot?". Failure to recognise the user's intent and thus failure to answer the question correctly can have legal implications for the chatbot designers [14, 15].

The R-U-A-Robot dataset [9] was created with the intent to study solutions to this problem, and prevent user discomfort or deception in situations where users might have unsolicited conversations with human-sounding machines over the phone. It contains 6800 sentences of this kind, labelled as positive (the question demands identification), negative (the question is about something else) and ambiguous. One can train and use a DNN to identify the intent.

How difficult is it to formally guarantee the DNN's intended behaviour? The state-of-the-art NLP technology usually relies on *transformers* [30] to embed natural language sentences into vector spaces. Once this is achieved, an additional medium-size network may be trained on a dataset that contains different examples of *"Are you a Robot?"* sentences. This second network will be used to classify new sentences as to whether they contain the intent to ask whether the agent is a robot.

There are several approaches to verify such a system. Ideally we would try to verify the whole system consisting of both the embedding function and the classifier. However, state-of-the-art transformers are beyond the reach of state-of-the-art verifiers. For example the base model of BERT [6] has around 110 million trainable parameters and GPT-3 [18] has around 175 billion trainable parameters. In contrast, the most performant neural network verifier AlpaBetaCrown [31] can only handle networks in the range of a few hundred of thousand nodes. So, verification efforts will have to focus on the classifier that stands on top of the transformer.

Training a DNN with 2 layers on the R-U-A-Robot dataset [9] gives average accuracy of 93%. Therefore there is seemingly no technical hurdle in running existing neural network verifiers on it [12, 27, 31]. However, most of the properties checked by these verifiers are in the computer vision domain. In this domain, images are seen as vectors in a continuous space, and every point in the space corresponds to a valid image. The act of verification guarantees that every point in a given region of that space is classified correctly. Such regions are identified as "ϵ-balls" drawn around images in the training dataset, for some given constant ϵ. Despite not providing a formal guarantee about the entire space, this result is useful as it provides guarantees about the behaviour of the network over a large set of unseen inputs.

However, if we replicate this approach in the NLP domain, we obtain a mathematically sound but pragmatically useless result. This is because, unlike images, sentences form a discrete domain, and therefore very few points in the input space actually correspond to valid sentences. Therefore, as shown in Figure 1, it is highly unlikely that the ϵ-balls will contain any unseen sentences for values of ϵ that can actually be verified. And thus, such verification result does not give us more assurance than just measuring neural network accuracy!

There is clearly a need in a substantially different methodology for verification of NLP models.

Our proposal is based on the following observations. On the verifier side, the state-of-art tools based on abstract interpretation are still well-suited for this task, because they are flexible in the definition of the size and shape of the input sub-space that they verify. But considerable effort needs to be put into definitions of subspaces that actually make pragmatic sense from the NLP perspective. For example, as shown in Figure 1, constructing a convex hull around several sentences embedded in the vector space has a good chance of capturing new, yet unseen sentences.

Unfortunately, calculating convex hulls with sufficient precision is computationally infeasible for high number of dimensions. We resort to over-approximating convex hulls with *"hyper-rectangles"*, computation of which only takes into consideration the minimum and maximum value of each dimension for each point around which we draw the hyper-rectangle. There is one last hurdle to overcome: just naively drawing hyper-rectangles around some data points gives negative results, i.e. verifiers fail to prove the correctness of classifications within the resulting hyper-rectangles.

There is no silver bullet to overcome this problem. Instead, as we show in this paper, one needs a systematic methodology to overcome this problem. Firstly, we need methods that refine hyper-rectangles in a variety of ways, from the standard tricks of reducing dimensions of the embedding space and clustering, to geometric manipulations, such as hyper-rectangle rotation. Secondly, precision of the hyper-rectangle shapes can be improved by generating valid sentence perturbations, and constructing hyper-rectangles around (embeddings of) perturbed, and thus semantically similar, sentences. Finally, based on the refined spaces, we must be able to re-train the neural networks to correctly fit the shapes, and this may involve sampling based on adversarial attacks within the hyper-rectangles. This final step is in line with other literature in this domain; an approach that couples verification with property-driven training is also known as continuous verification [13, 5, 28].

The result is a comprehensive library that contains a toolbox of pre-processing and training methods for verification of NLP models. We call this tool ANTONIO - Abstract domaiN Tool fOr Nlp verIficatiOn (see Figure 2). Although in this tool paper, we evaluate the results on just one R-U-A-Robot dataset, the methodology and libraries are completely general, and should work for any NLP dataset and models of comparable sizes. We envisage that this work will pave the way for including NLP datasets and problems as benchmarks into DNN verification competitions and papers, and more generally we hope that it will make NLP problems more accessible to the neural network verification community.

2 ANTONIO and Existing Trends in NLP Verification

This paper expands upon the area of NLP verification, this is a nascent field which has only just recently started to be explored.

In their work Zhang et al. [37] present ARC, a tool that certifies robustness for word level substitution in LSTMs. They certify robustness by computing an adversarial loss and, if the solution is $<$ 0, then they prove that the model is robust. However, this tool has limitations such as it certifies only LSTMs that are far below the current state-of-the-art for NLP in terms of size, it uses word embeddings and it is dataset dependant. Huang et al. [11] verify convolutional neural networks using Interval Bound Propagation (IBP) on input perturbations. However, they only study word synonym and random character replacements on word embeddings. Ye et al. [35] focus on randomised smoothing for verification of word synonym substitution. This approach allows them to be model-agnostic, however they are still limited by only 1 type of perturbation and word/sub-word embeddings. Lastly, Shi et al. [25] propose a verification method for transformers. However, their verification property is ϵ-ball robustness and they only demonstrate their method on transformers with less than 3 layers claiming that larger pre-trained models like BERT are too challenging, thus the method is not usable in real life applications.

We can see that although new approaches are being proposed, we have yet to have a consensus

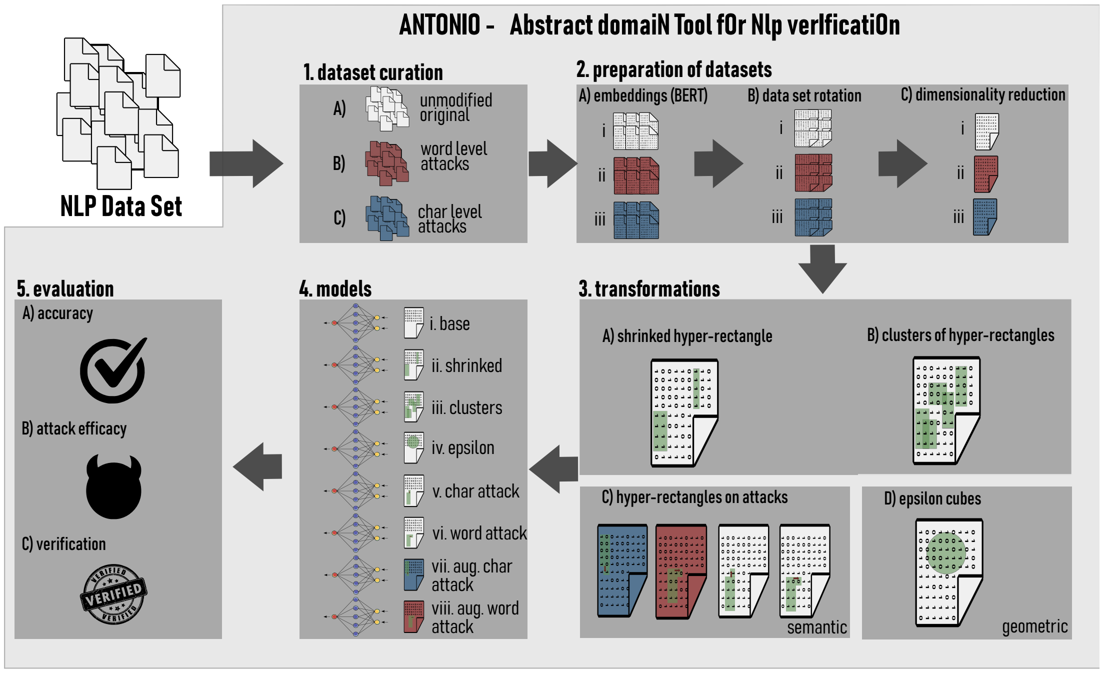

Figure 2: *Flow-chart for the tool ANTONIO, showing five main blocks of methods: 1. NLP attacks ("data set curation"); 2. geometric pre-processing of data, 3. definition of suitable verifiable regions ("transformations"), 4. training models using adversarial training relative to different kinds of attacks, and 5. evaluation of success rates achieved by verifiers and attackers for the trained models.*

on how to approach the verification, and more importantly scalability is a huge issue. This dictated our exploration of different verification spaces and our focus on filter models as a useful real world verification opportunity. Geometric shapes, and especially ϵ-balls and ϵ-cubes, are widely used in verification for computer vision to certify robustness properties. In NLP, the aforementioned verification approaches also make use of intervals and IBP, abstract interpretation [37, 11, 35] and ϵ-ball robustness on word/sub-word embeddings. However, to the best of our knowledge, there is no previous work on geometric/semantic shapes on sentence embeddings.

Table 4 summarises differences and similarities of the above NLP verification approaches. Despite their diversity, most NLP verification approaches make use of a similar methodological pipeline:

- Generate "semantic" perturbations on sentences (e.g. the IBP papers tend to use word synonym substitutions, ANTONIO does word, character and sentence level perturbations).
- Embed them into continuous spaces (the cited IBP papers use the word embedder GloVe [22], ANTONIO uses more modern Sentence-BERT [23]).
- Use geometric bounds (literally bounds in IBP papers or "hyper-rectangles" of different kinds in ANTONIO) to:
  - Verify the networks' behaviour within those geometric bounds (IBP papers use IBP algorithms to do it, ANTONIO uses off-the shelf tools Marabou and ERAN) and
  - Influence neural network training within those input regions. (Custom algorithm is used in IBP papers, ANTONIO uses PGD on defined input space regions).

Method	**Datasets**	**NLP attacks**	**Embeddings**	**Models**	**Training algorithm**	**Verification algorithm**
ANTONIO	RUA Robot, Medical	General purpose: char, word and sentence perturbations	Sentence: Sentence-BERT	Works with any model handled by the verifier	**PGD-based**	Works with any SoA verifier (**Marabou**, ERAN)
Jia et al. (2019)	IMDB, SNLI	Word substitution	Word: GloVe	LSTM, attention-based, CNN, BoW (2 and 3 layers)	**IBP-based**	IBP-based
Huang et al. (2019)	AGNews, STT	Char and word substitution	Word: GloVe	CNN (3 and 4 layers)	**IBP-based**	IBP-based
Shi et al. (2020)	YELP, STT	ϵ-ball	Word: not specified	Transformers (max 3 layers)	-	Abstract interpretation-based
Zhang et al. (2021)	IMDB, STT, STT2	Word perturbations	Word: not specified	LSTM	**IBP-based**	IBP-based

Table 1: *Summary of the main features of the existing NLP verification approaches. In bold are SoA methods.*

In the cited IBP papers, the main purpose is to develop a new tailored IBP based method that works for NLP. In it, input region definition is an internal feature of the new IBP algorithms for verification and training. In contrast, in ANTONIO we deliberately disentangle the issues of algorithm design and the problem of input region analysis. This allows us to use state-of-the-art verifiers and training algorithms in a modular fashion; and opens a way for generating benchmarks uniformly.

To contrast to verification literature, there is a wide body of work on improving adversarial robustness of NLP systems [36, 32, 33, 16, 38, 39, 8]. The approaches previously mentioned make use of data augmentation and adversarial training techniques. ANOTNIO in the future could be extended with diverse sets of attacks for its defined input regions.

3 ANTONIO's Overall Design

We now outline the design of the tool ANTONIO (github.com/ANTONIONLP/ANTONIO). ANTONIO covers every aspect of the NLP verification pipeline, as shown in Figure 2. It is modular, meaning that you can modify or remove any part of the NLP verification pipeline, which usually consists of the following steps:

1. selecting an NLP data set and embedding sentences into vector spaces;

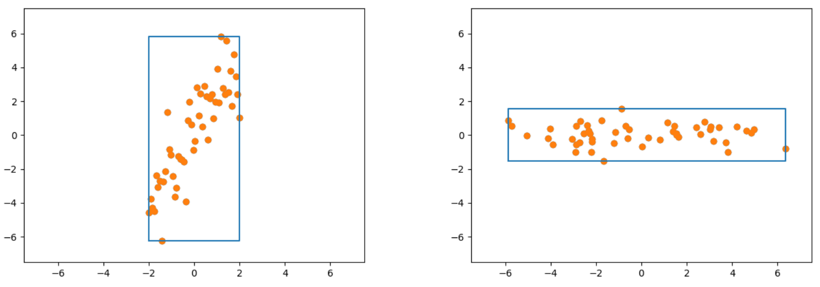

Figure 3: *A 2-dimensional representation of the original data (left) and its eigenspace rotation (right).*

2. generating attacks (word, charcter, sentence attacks) on the given sentences, in order to use them for data augmentation, training, or evaluation;
3. standard machine learning curation for data (e.g. dimensionality reduction) and networks (training, regularisation);
4. verification, that usually comes with tailored methods of defining input and output regions for networks.

ANTONIO is designed to provide support at all of these stages (detailed below):

Dataset Here we experiment with the R-U-A-Robot dataset, however the user can pick any NLP dataset that can be used for classification.

Semantic attacks and dataset curation ANTONIO can create additional augmented datasets by perturbing the original sentences. By incorporating state-of-the-art attacks by [19, 34], ANTONIO implements several character-level, word-level, and more sophisticated sentence-level perturbations that can be mixed and matched to create the augmented datasets. Examples of character and word level perturbations can be found in Tables 2 and 3.

Dataset preparation This block can be considered as geometric data manipulations. First of all, we need an embedding function. ANTONIO utilises SentenceBERT [23] as a sentence embedder. The model implemented within ANTONIO produces embeddings in 384 dimensions. The user can, however, substitute SentenceBERT with any embedding function they prefer. The original part of ANTONIO implements data rotation to help the hyper-rectangles to better fit the data (as shown in Figure 3). Lastly, we use PCA for dimensionality reduction [21]. This helps verification algorithms to reduce over-approximation and speeds up training and verification by reducing the input space. These last two data manipulations can be arbitrarily omitted or modified and the user can insert other manipulations of their choices that might help.

Hyper-rectangles The original core of ANTONIO as a tool is the code producing hyper-rectangles for given sentences (and their semantic attacks). We implemented several ways to create and refine the hyper-rectangles to increase their precision. Figure 4 (left) shows how a naive hyper-rectangle, that contains all the inputs from the desired class, might also contain inputs from the other class. That is why we implemented a method to shrink the hyper-rectangle to exclude the undesired inputs (centre)

Method	**Description**	**Original sentence**	**Altered sentence**
Insertion	A character is randomly selected and inserted in a random position.	*Are you a robot?*	*Are yovu a robot?*
Deletion	A character is randomly selected and deleted.	*Are you a robot?*	*Are you a robt?*
Replacement	A character is randomly selected and replaced by an adjacent character on the keyboard.	*Are you a robot?*	*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 robot?*	*Are you a rboot?*
Repetition	A character in a random position is selected and duplicated.	*Are you a robot?*	*Arre you a robot?*

Table 2: *Character-level perturbations: their types and examples of how each type acts on a given sentence from the R-U-A-Robot dataset [9]. Perturbations are selected from random words that have 3 or more characters, first and last characters of a word are never perturbed.*

Method	**Description**	**Original sentence**	**Altered sentence**
Deletion	Randomly selects a word and removes it.	*Can u tell me if you are a chatbot?*	*Can u tell if you are a chatbot?*
Repetition	Randomly selects a word and duplicates it.	*Can u tell me if you are a chatbot?*	*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 a chatbot?*	*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 are a chatbot?*	*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 a chatbot?*	*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 are a chatbot?*	*Can u tell me if you were a chatbot?*

Table 3: *Word-level perturbations: their types and examples of how each type acts on a given sentence from the R-U-A-Robot dataset [9] .*

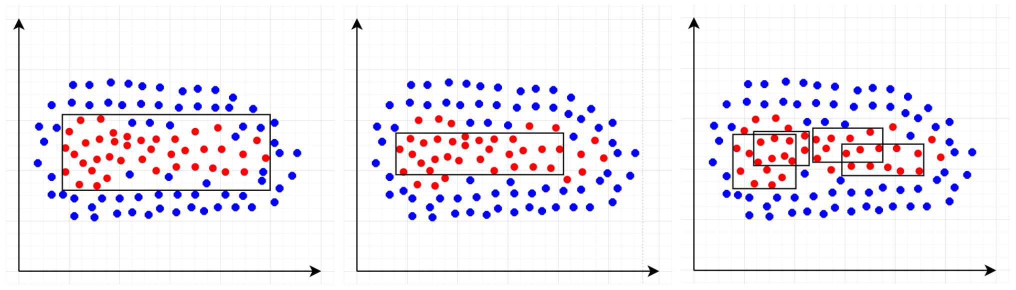

Figure 4: *An example of hyper-rectangle (left), shrunk hyper-rectangle (centre) and clustered hyper-rectangles (right) in 2-dimensions. The red dots represent sentences in the embedding space of one class, while the blue dots are embedded sentences that do not belong to that class.*

and a method for clustering and generating multiple hyper-rectangles around each cluster (right). Furthermore, to increase precision, ANTONIO can create a third type of hyper-rectangles (Figure 2 (3C)) by attacking the inputs (as described in the data augmentation part) and then drawing the hyper-rectangles around each input and its perturbations. Finally, for comparison purposes, we also implemented the creation of ϵ-cubes. These hyper-rectangles will be used both for training and verification.

Training We implemented three methods for training: base training, data augmentation, and adversarial training The base models are trained with a standard cross-entropy loss on the original dataset. The data augmentation models are still trained with a standard cross-entropy loss but on the augmented datasets. For adversarial training, instead, we are using Projected Gradient Descent (PGD) [17] to calculate the worst case perturbation for each input on each epoch and we are adding those perturbations when calculating the loss. Usually, PGD perturbations are projected back into the given ϵ-cube. ANTONIO projects within the given hyper-rectangles instead. The user can choose any combination of training methods, hyper-rectangles, and attacks to train the models or they can implement new ones as well. Section 4 will illustrate this modular usage of the tool.

Evaluation and Verification For the evaluation, ANTONIO implements mainly three metrics. Firstly it simply calculates the standard accuracy of the model, as it is important to not have a significant drop in accuracy when you train a model for robustness. Secondly, it computes robustness to attack (accuracy on adversarial test samples), which is obtained by generating several perturbations of the test set and by calculating accuracy on those. Finally, ANTIONIO calculates the percentage of verified hyper-rectangles. For testting purposes we connected ANTONIO to two state-of-the-art verifiers: ERAN [27, 26] and Marabou [12]. Thanks to VNN-COMP, the annual neural network verification competition [20], the community developed common neural network verification standards, that require ONNX format for neural networks [2] and VNNLIB format for verification queries [1]. ANTONIO automatically performs translation of neural networks into ONNX, and specifications of hyper-rectangles in Python – into the VNNLIB format. Furthermore, ANTONIO implements methods for generating queries, retrieving the data and calculating the statistics on them. The user can connect and use any verification tool that they prefer and also add any other metric of choice.

Verifier	**Model**	$\mathbb{H}^*_{\epsilon=0.05}$	$\mathbb{H}^*_{char}$	$\mathbb{H}^*_{word}$
Marabou	N_{base}	1.79	4.88	11.69
	$N_{\epsilon=0.05}$	**18.46**	21.99	41.93
	$N_{char-adv}$	7.37	**30.41**	41.93
	$N_{word-adv}$	12.17	25.82	**45.12**
ERAN	N_{base}	0.00	0.87	1.80
	$N_{\epsilon=0.05}$	0.12	4.18	10.16
	$N_{char-adv}$	0.00	4.43	8.97
	$N_{word-adv}$	0.04	4.05	10.75

Table 4: *Example how ANTONIO-generated networks and input shapes form a uniform benchmark for two verifiers.*

4 Example of Benchmarking with ANTONIO

We now show how to use ANTONIO for verification benchmarking, using the R-U-A-Robot dataset as an example.

Following the pipeline described in Section 3, ANTONIO generates a set of character, word and sentence perturbations for the dataset, by perturbing each sentence several times in turn. Tables 2 and 3 show how, given one sentence and one type of perturbation, one forms a set of perturbed sentences. ANTONIO keeps different types of perturbations separately. We obtain, per each sentence, three sets of its perturbed variants, one per each perturbation type.

ANTONIO embeds these sentences into a vector space, and performs the geometric transformations, such as data rotation and dimensionality reduction. Then, for each perturbation type, ANTONIO creates one hyper-rectangle per each original sentence. The hyper-rectangle covers the original sentence and its n perturbations. ANTONIO collects all such hyper-rectangles for the given set of sentences. We call the resulting sets of hyper-rectangles $\mathbb{H}^*_{char}$ and $\mathbb{H}^*_{word}$, $\mathbb{H}^*_{sentence}$, depending on the perturbation type. Similarly, it also creates a set of ϵ-cubes (denoted $\mathbb{H}^*_{\epsilon=0.05}$), based on the L_∞ distance, and ignoring any knowledge of semantic perturbations. Finally, ANTONIO trains a baseline network (N_{base}) and four adversarial networks ($N_{\epsilon=0.05}$, $N_{word-adv}$, $N_{char-adv}$, $N_{sentence-adv}$) trained using PGD on, respectively, $\mathbb{H}^*_{\epsilon=0.05}$, $\mathbb{H}^*_{word}$, $\mathbb{H}^*_{char}$, $\mathbb{H}^*_{sentence}$. Having made these preparations, ANTONIO forms a set of verification challenges, to verify the given networks and the input space covered by $\mathbb{H}^*_{\epsilon=0.05}$, $\mathbb{H}^*_{char}$, $\mathbb{H}^*_{word}$ and $\mathbb{H}^*_{sentence}$. At this point, the automatically generated neural networks in ONNX format and the verification conditions defining the desirable shapes in VNNLib format can be given to any verifier participating in VNN-COMP. Table 4 shows the results for Marabou and ERAN, the 2 verifiers that we tested with ANTONIO already. But this list can be easily extended. For brevity, we note the results presented (Table 4) are just a snapshot of what ANTONIO is capable of, and we omit experiments done on sentence perturbations.

Acknowledgements

Authors acknowledge support of EPSRC grant AISEC EP/T026952/1 and NCSC grant Neural Network Verification: in search of the missing spec.

References

[1] VNNLib format: https://vnnlib.org/. Accessed on 01.12.2022.

[2] Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. https://github.com/onnx/onnx, 2019.

[3] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021.

[5] Marco Casadio, Ekaterina Komendantskaya, Matthew L. Daggitt, Wen Kokke, Guy Katz, Guy Amir, and Idan Refaeli. Neural network robustness as a verification property: A principled case study. In *Computer Aided Verification (CAV 2022)*, Lecture Notes in Computer Science. Springer, 2022.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, 2018.

[7] Emily Dinan, Gavin Abercrombie, A. Stevie Bergman, Shannon Spruit, Dirk Hovy, Y-Lan Boureau, and Verena Rieser. Anticipating safety issues in E2E conversational AI: Framework and tooling, 2021.

[8] Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. Towards robustness against natural language word substitutions. *arXiv preprint arXiv:2107.13541*, 2021.

[9] David Gros, Yu Li, and Zhou Yu. The R-U-A-Robot dataset: Helping avoid chatbot deception by detecting user questions about human or non-human identity, 2021.

[10] Julia Hirschberg and Christopher D. Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015.

[11] Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. Achieving verified robustness to symbol substitutions via interval bound propagation, 2019.

[12] Guy Katz, Derek Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David Dill, Mykel Kochenderfer, and Clark Barrett. *The Marabou Framework for Verification and Analysis of Deep Neural Networks*, pages 443–452. 07 2019.

[13] Ekaterina Komendantskaya, Wen Kokke, and Daniel Kienitz. Continuous verification of machine learning: a declarative programming approach. In *PPDP '20: 22nd International Symposium on Principles and Practice of Declarative Programming, Bologna, Italy, 9-10 September, 2020*, pages 1:1–1:3. ACM, 2020.

[14] Mauritz Kop. Eu artificial intelligence act: The european approach to ai, 2021.

[15] California State Legislature. Senate bill no. 1001, chapter 892, chapter 6.bots, paragraph 17941, 2018.

[16] Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. Searching for an effective defender: Benchmarking defense against adversarial word substitution. *arXiv preprint arXiv:2108.12777*, 2021.

[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[18] Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[19] Milad Moradi and Matthias Samwald. Evaluating the robustness of neural language models to input perturbations. *arXiv preprint arXiv:2108.12237*, 2021.

[20] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T Johnson. The third international verification of neural networks competition (vnn-comp 2022): summary and results. *arXiv preprint arXiv:2212.10376*, 2022.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[23] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks, 2019.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks, 2016.

[25] Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers, 2020.

[26] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. Replication package for the article: An abstract domain for certifying neural networks.

[27] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

[28] Natalia Slusarz, Ekaterina Komendantskaya, Matthew L. Daggitt, Robert J. Stewart, and Kathrin Stark. Logic of differentiable logics: Towards a uniform semantics of DL. In *LPAR-24: The International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, 2023.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[31] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

[32] Wenqi Wang, Run Wang, Lina Wang, Zhibo Wang, and Aoshuang Ye. Towards a robust deep neural network against adversarial texts: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[33] Xuezhi Wang, Haohan Wang, and Diyi Yang. Measure and improve robustness in nlp models: A survey, 2021.

[34] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288*, 2021.

[35] Mao Ye, Chengyue Gong, and Qiang Liu. Safer: A structure-free approach for certified robustness to adversarial word substitutions, 2020.

[36] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on

deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

[37] Yuhao Zhang, Aws Albarghouthi, and Loris D'Antoni. Certified robustness to programmable transformations in LSTMs, 2021.

[38] Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huan. Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. In *Association for Computational Linguistics (ACL)*, 2021.

[39] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*, 2019.