On Robustness for Natural Language Processing

Emanuele La Malfa

Trinity College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2023

‘Vivere non è facile.’ (è vero, è una vita che lotti)

‘C’era una volta un autista su una Punto bordó.’ (sono felice che stai meglio ora)

‘Avanti tutta non si molla un cazzo.’ (ogni parola è superflua qui)

‘Te quiero mucho Emmanuel. Que viva Mexico!’ (E sto fumando, Giro por la calle e sono attento, Lei sopra di me lo muove lento, Steso dentro al letto, giuro che la spengo, E dopo faccio, Ra-pa-pam-pam, ra-pa-pam-pam, Ra-pam-pam-pam-pam-pam, Ra-pa-pam-pam, ra-pa-pam-pam, Ra-pam-pam-pam-pam-pam.)

‘Iiiiiiihhh, uuuuuuuhhhh! Comeeee...vó?’ (🐻)

‘Mi fa male la gamba non riesco a giocare.’ (vedi Napoli e muori, diceva Goethe)

‘It is what it is.’ (fair enough, thank you for what you have done)

‘I am just stating a basic fact here.’ (~~never~~ let me go)

‘If you ignore moral, it doesn’t exist.’ (I have learned a shitload from you)

Acknowledgments

I want to express my sincere gratitude to Prof. Marta Kwiatkowska, my supervisor, for her unwavering patience, invaluable guidance, constant encouragement, and profound advice throughout my academic journey under her mentorship. I consider myself exceptionally fortunate to have worked with a supervisor who consistently addressed my inquiries and provided timely responses. As regards this thesis, and each work that constitutes it, she contributed by giving me constant feedback, revising my work periodically, and suggesting research directions to explore.

As regards funding, this work received support from Innovate UK (reference 104814) and the ERC under the European Union's Horizon 2020 research and innovation program FUN2MODEL (grant agreement No. 834115).

Abstract

As a discipline, machine learning has contributed to significant breakthroughs in Natural Language Processing (NLP), aiming to design algorithms to manipulate text and produce insights, such as classification and summarization, comparable to those of humans. Natural language poses challenges that reflect peculiarities of human intelligence, such as grasping the meaning of a sentence or preserving long-term relationships between words that possibly appear distant from each other.

A considerable body of recent literature provides evidence that NLP models behave inconsistently on slight manipulations of a text, as in the case of word substitution. Differently from computer vision (CV), where a pixel manipulation produces a (possibly not natural) image, NLP algorithms rely on text representations in the form of embedded vectors, where the linguistic constituents (i.e., words, phrases, sentences) are transformed into multi-dimensional vectors of real-valued numbers, marking a clear separation between human and machine representation.

In this thesis, we investigate guarantees and the formal explainability of NLP models through the lens of adversarial robustness. We review the applicability of adversarial robustness, as defined in CV, as the region of maximal safety of a neural network (NN) decision against discrete and continuous perturbations. We develop an evaluation framework that certifies adversarial robustness for different models, and we analyze how the validity of such certificates vanishes in settings that grow in complexity. This investigation is a prelude to novel definitions of robustness that are aligned with linguistics, aiming to assess a model's syntactic and semantic capabilities.

With *semantic robustness*, we introduce a framework to test a model against linguistic phenomena. In contrast, *syntax robustness* aims to falsify the hypothesis that NLP models embed high-order linguistic structures such as syntactic trees. Extensive experimentation on various architectures and benchmarks validates the proposed concepts and sheds light on how brittle these architectures are against slight linguistic variations, against which humans are exceptionally robust.

We finally investigate the role of robustness as a property to explain neural networks: we propose the notion of optimal robust explanation (ORE) as the robust and optimal portion of an input text that is nevertheless sufficient to imply a model's decision. We implement and test this notion of explanations on various neural networks and datasets to reveal the explanatory landscape of NLP models through the lens of robustness.

All the software and tools of this thesis have been released under permissive, open-source licenses to satisfy reproducibility requirements and encourage other researchers to develop tools to assess and improve the robustness of NLP models against edge cases and linguistic phenomena, which by their nature constitute a non-negligible part of the spectrum of human language.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

In the last decade, Natural Language Processing (NLP) has made impressive advances, from neural network-aided search engines, to algorithms that smoothly translate sentences to and from tens of languages (Devlin et al., 2019a; Jia et al., 2018; Brown et al., 2020). However, these advances have not come with sufficient robustness guarantees, as testified by a considerable body of recent literature that demonstrates the brittleness of language models to slight variations of the inputs (Jia and Liang, 2017; Gowal et al., 2018; Cheng et al., 2020; Sun et al., 2020). This lack of robustness poses a series of challenges that are not trivial to mitigate, rooted in problems that are computationally hard to solve (Katz et al., 2017b). One must not underestimate the psychological effect of errors caused by systems powered by Artificial Intelligence (AI), as they erode trust to a level that is hard to rebuild. To name one case that made the headlines, in 2016, a chat-bot released by Microsoft Corporation via Twitter began to post inflammatory and offensive tweets through its Twitter account.[1]

While NLP systems are mostly employed in non-safety-critical processes, countless scientific articles and reports have testified to potentially harmful and discriminating behaviors of these algorithms, especially when trained on data that explicitly or implicitly encode human biases.

In this work, we elect as an object of our study Neural Networks (NN), as they are universal function approximators employed at the forefront of computer vision (CV), NLP, and reinforcement learning. In particular, we restrict our attention to Deterministic Neural Networks (DNN). Despite being a popular area of research for the past several years, robustness in NLP focuses on guarantees against threat models

[1] `https://www.theguardian.com/world/2016/mar/29/microsoft-tay-tweets-antisemitic-racism`.

whose linguistic expressiveness is limited. A reason for that is that language makes the study of robustness challenging, as one cannot take advantage of smooth mathematical formulations of the input space since language is discrete, combinatorial, and infinite.

With attacks that can be conducted at a word or representation level, the approaches to NLP robustness are two-fold: on the one hand, those aimed to shield a model from any perturbation that satisfies some specifics (e.g., locality), and on the other, those that reduce the surface of attack to linguistically correct sentences. In computer security, one would denote the latter as a white-box scenario and the former as a black box: nuances between the two, which are also studied in the literature, would go under the name of grey-box scenarios.

It is thus important to make clear the hypothesis under which one studies the problem of robustness in NLP, starting from the models employed, which are rapidly switching from 'classic' NNs to powerful yet extremely large, data-intensive, Large Language Models (recently re-branded as foundational models (Bommasani et al., 2021)). While the field lacks a formalized approach to robustness, we observe an emergence of 'linguistics for Language Models' (Goldberg, 2019; Jawahar et al., 2019; Manning et al., 2020).

Last but not least, since complex models come at the cost of less interpretable decisions, this thesis tackles the problem of explainability from a formal perspective, showing the possibility of formally interpreting, debugging, and explaining NLP models aided by robustness guarantees.

1.2 Contributions

With this thesis, we develop frameworks and conduct systematic analyses to quantify, critique, and evaluate robustness in NLP. We also propose to use robustness as a property to enhance the explainability of complex NLP models.

The principles of NLP robustness, inherited from CV, need to be aligned with a model's expectations, i.e., understanding, manipulating, and generating language at a level comparable to that of humans. In this work, we explain why it is necessary to take a step back from CV and rethink the notion of robustness through the lens of linguistics. In this sense, we propose two notions of robustness inspired by the linguistic dichotomy between syntax and semantics, namely *semantic* and *syntactic robustness*. We also investigate the broader role of robustness as a property by showing that NLP models' decisions can be explained via robust and minimal, e.g., in the number of features involved in the prediction subsets of the input.

The first contribution of the thesis consists of a framework to measure the robustness of standard NLP architectures, such as Fully Connected (FC), Convolutional (CNN), and Recurrent Neural Networks (RNN), on text classification tasks. This approach extends the Maximal Safe Radius technique (MSR) (Wu et al., 2020a) to language and computes a sound certificate of the robustness of a model against adversarial attacks. The contribution here is two-fold: a method and an algorithm to certify the region of maximal safety of different NNs against adversarial attacks, with an extensive evaluation of text classification and varying neural topologies, and an assessment of the limitations of such an approach in terms of scalability, flexibility, and linguistic validity. Our empirical observations pave the way to the key contributions presented in the following chapters, namely the necessity to introduce *ad-hoc* notions of robustness for syntax and semantics, and further motivate our steps towards robust explainability.

We discuss the inadequacy of the standard notion of robustness when applied to NLP, as it does not satisfy linguistic requirements such as the invariance to (linguistically similar) paraphrases; indeed, the standard notion of robustness limits its guarantees to symbol substitutions, which is not appropriate for models that solve low-order tasks such as sentiment analysis (Barnes et al., 2019). We thus define *semantic robustness* to assess how a model behaves when targeted with high-quality perturbations that contain linguistic phenomena, such as negation or sarcasm. We build a simple yet semantically controlled, generative framework to measure the *semantic robustness* of an NLP model, whose results we compare with an existing, handcrafted test bed (Barnes et al., 2019), on a variety of NNs, including Large Language Models (LLM) such as BERT (Devlin et al., 2019a). Results suggest that *semantic robustness* is higher for those models that better represent language and agree with the recently observed empirical law of scaling (the larger the model, the better the linguistic capabilities).

Syntax, alongside semantics, shapes one of the cornerstone definitions in post-structuralist linguistics (Chomsky, 2009): a natural complement to *semantic robustness* is thus the concept of *syntactic robustness*, defined as the capacity of a model to embed robust linguistic structures, such as syntax trees. We challenge the evidence (Manning et al., 2020) that LLMs encode linguistic structures in their hidden representations with an adequate measure that computes and numerically quantifies the perturbation that disrupts the most the syntactic structures encoded by a model. We provide evidence, through a series of experiments on a variety of embeddings and LLMs, that syntactic structures, when encoded by LLMs or standard word embedding techniques, are very brittle to slight syntax-preserving manipulations of an input text, to the point that one must seriously question their existence. Our approach extensively uses probing tasks (Niven and Kao, 2019), namely linear models that reveal features used by the original representation, whether an embedding or an LLM. We propose and implement an algorithmic framework within which one can assess whether powerful LLMs encode in their continuous representations (enough information to reconstruct) symbolic structures that linguists have proposed to characterize natural language.

In the thesis final contribution, we explore the role of robustness in explainability and beyond being a desirable property of a model. We seek explanations of NLP models (and, in general, of any machine learning model) that are robust to perturbations. We define the notion of an Optimal Robust Explanation (ORE) as the optimal (w.r.t. a cost function) subset of an input sufficient to entail a model's decision. A theoretical characterization, which includes the reconciliation of this concept with that of popular explainers for NLP such as Anchors (Ribeiro et al., 2018a), followed by an experimental evaluation on a variety of models on sentiment analysis tasks, provides evidence that OREs are succinct yet interpretable explanations, which can detect model as well as decision biases.

In synthesis, my thesis clarifies the role of robustness in NLP and mainly focuses on low-order tasks such as sentiment analysis and classification. The contributions in Chapters 5 and 6 can be applied to a wide range of models, from feed-forward NNs to advanced Large Language Models, while Chapters 4 and 7 are pertinent to guaranteed robustness, and thus suffer from limitations due to the limited scalability of verification techniques. With the state-of-the-art verification tools that do not scale to large attention-based models, we focused on simpler architectures such as fully connected, convolutional and recurrent topologies, though the frameworks apply to any neural architecture.

1.3 Publications

This thesis is based on 4 research papers. Three of the research papers have been accepted for publication, while the remaining one is currently under submission to a journal. Below I clarify my contribution to each piece of work.

Assessing Robustness to Text Classification with Maximal Safe Radius Computation (La Malfa et al., 2020) - EMNLP'20 (findings). With a colleague and co-author, I came up with the idea of a framework to measure robustness for an NLP model in terms of an upper and a lower bound certificate, namely the Maximal Safe Radius technique for NLP, that is interpretable in terms of guarantees w.r.t. word substitutions. We defined the framework for the upper bound, while another colleague helped me design the lower bound algorithm. I adapted existing frameworks for the lower-bound and coded the upper-bound algorithm from scratch. I conducted all the analyses in the published paper. All the authors collaborated on the writing.

The King is Naked: on the Notion of Robustness for Natural Language Processing (La Malfa and Kwiatkowska, 2022) - AAAI'22. I came up with the idea of criticizing NLP robustness as inherited from CV, and the consequent notion and formalization of *semantic robustness* in NLP as the invariance of a model to slight variations of an input that contains a linguistic phenomenon, such as negation. I designed and coded the framework to quantify *semantic robustness* for various models, as well as the augmentation technique that introduces linguistic phenomena in a text. I conducted all the experiments of the published paper and wrote the paper jointly with my supervisor.

Emergent Linguistic Structures in Neural Networks are Fragile (La Malfa et al., 2022) - Under Review. I developed the idea of measuring *syntax robustness* of NLP models through probing tasks. My co-authors and I proposed the algorithm to quantify the average worst-case robustness against perturbations. At the same time, I designed the perturbation techniques and the notion of *syntax robustness.* I wrote the code and conducted all the analyses. I wrote the paper jointly with my supervisor.

On Guaranteed Optimal Robust Explanations for NLP Models (La Malfa et al., 2021) - IJCAI'21. I developed the idea of distilling robust explanations for NNs into a framework with my collaborator Nicola and Agnieszka. I coded the Hitting Set algorithm to extract OREs from NNs with a uniform cost function, conducted all the experiments, and then analyzed the results jointly with my collaborators. The paper was written jointly.

1.4 Thesis Outline

In Chapter 1, we provide an overview of the thesis and set the stage for the subsequent chapters. We introduce the main research question and outline the objectives and significance of the study. Additionally, we establish the thesis scope and explain the remaining chapters' structure.

Chapter 2 is a foundation for the rest of the thesis. We present the necessary notation, terminology, and theoretical concepts that underpin the subsequent chapters. This chapter aims to ensure a common understanding of the technical aspects of the research, providing readers with the tools to comprehend the following discussions effectively.

In Chapter 3, we comprehensively survey the principal published works in robustness and Natural Language Processing (NLP). We critically analyze and synthesize existing research, identify gaps and limitations, and highlight relevant findings and methodologies. This chapter thoroughly explains the current state of the art in robustness and NLP.

Chapter 4 introduces a novel framework, based on the Maximal Safe Radius method, to quantify robustness in NLP, particularly for models that solve text classification tasks. We describe the approach, which consists of an upper and a lower bound on the robustness of a model, its underlying principles, and the experimental setup used to validate the method. While previous works focus on local guarantees, our certification method is the first that applies formal techniques in conjunction with attack-based methods to certify and characterise the robustness landscape of neural networks to local attacks. This chapter presents the reader with a detailed examination of the proposed measurement framework and its potential applications in assessing the resilience of NLP models.

In Chapter 5, we delve into formalizing a notion of robustness focused explicitly on semantics. We begin by criticizing the idea of continuous robustness inherited from CV and adopted in NLP. We then explore NLP models' inherent vulnerabilities in handling texts containing semantic phenomena and propose a framework to measure and evaluate *semantic robustness*, which we apply to texts generated via a sample-based generative model and to samples handcrafted by human experts. Our work proposes a novel approach to robustness that aligns with human language, is probabilistic (as formal guarantees do not scale) and encompasses complex linguistic phenomena, going beyond word substitution.

Chapter 6 extends the notion of robustness to the realm of syntax. We develop a framework that captures and quantifies the syntactic robustness of NLP models, examining their susceptibility to perturbations that vary in their degree of syntactic disruption. By investigating various syntactic scenarios and conducting extensive experiments on different linguistic representations, we provide insights into existing models' limitations to represent syntax consistently. We also analyze the performance degradation on syntactic tasks of a fine-tuned model that solves low-order tasks such as sentiment analysis. This chapter's main contribution consists of a framework to make the hypothesis on linguistic structures, as embedded by neural networks and Large Language Models, falsifiable, and a comprehensive experimental evaluation.

In Chapter 7, we conclude the thesis by introducing a novel approach to generate guaranteed optimal explanations for NLP models. We address the need for interpretable models and propose a method that provides explanations and guarantees their optimality. This chapter presents the reader with a tool to comprehend the inner workings of NLP models, fostering trust and transparency in their decision-making processes. Our work is the first to bridge formal explainability and NLP and compares and enhances existing state-of-the-art methods such as Anchors.

Chapter 2

Background

This chapter reviews the background of Deep Learning (DL), and specifically neural networks (NN), training procedures, and the most influential DL architectures proposed in the past decades. A discussion of NLP representation techniques then follows. These underpin modern developments such as attention networks, Large Language Models (LLM), and applications such as ChatGPT. We also discuss the notion of the robustness of neural network models, their range of applicability, and their computational aspects. We conclude with a concise yet comprehensive background section on explainability, which we build on in Chapters 4 and 7. This part provides an overview of the tools and ideas used in the contribution chapters, while we dedicate Chapter 3 to a comprehensive review of the existing literature.

2.1 Deep Learning

Machine learning (ML) leverages recent advances in computer architectures and computational power (e.g., Moore's law) to automate the solution of complex tasks. ML algorithms are often formulated as optimization problems, where a model uses data to fit a desired function (Bishop and Nasrabadi, 2006). Three learning paradigms characterize the entire field: supervised, unsupervised, and reinforcement learning. Supervised learning accomplishes the goal of finding a function of optimal parameters W that maps input-output pairs, namely $f^W : X \rightarrow Y$; when not necessary, in this work we will omit the weights when referring to f, i.e., $f : X \rightarrow Y$. Unsupervised learning requires learning an alternative representation of solely the input points X, and is generally employed to describe, compress, and cluster data. Finally, in reinforcement learning an agent learns to interact with an environment that responds with feedback on the agent's actions (Sutton and Barto, 2018). Supervised, unsupervised, and reinforcement learning are formulated as optimization problems;

all the formulations aim to minimize a loss function $\mathcal{L}$, which expresses how good/bad the trained model is at solving the task (supervised and reinforcement learning) or representing the input (unsupervised learning).

Deep Learning, which exploits neural network (NN) architectures, is sometimes viewed as a standalone discipline at the intersection of AI, ML, and statistics (Salakhutdinov, 2014), with high-profile conferences dedicated entirely to neural-inspired architectures. Historically, NNs were developed in the 1950s as sequential algorithms loosely based on the human brain (Minsky and Papert, 2017). After initial successes, followed by a decline in the 1980s known as the *winter of Artificial Intelligence*, NNs have gained popularity in the 2010s due to their competitive advantage on tasks such as computer vision and NLP (Krizhevsky et al., 2012). Nowadays, they form the building blocks of translators and image recognition products, yet they are still far from exhibiting what we could call intelligent, autonomous behavior. We next discuss DL methods.

2.1.1 Neural Networks

We assume a supervised learning setting in this work when not explicitly stated. In this sense, given a set of n inputs $X = \{x^{(1)}, .., x^{(n)}\}$ and outputs $Y = \{y^{(1)}, .., y^{(n)}\}$, an NN learns a mapping from X, the input space, to Y, the output space, namely $f : X \to Y$. While in the general case, X is a d-dimensional real-valued subspace of $\mathbb{R}^d$, for which the domain is denoted as $\mathbb{D}(X)$, Y can be discrete, with the elements of its domain referred to as 'classes'; this specific setting is referred to as 'classification'. The complementary case, where $Y \subseteq \mathbb{R}^m$, is instead referred to as 'regression'.

2.1.2 Neural Architectures

We introduce the Neural Network model via its simplest architecture, namely the Fully Connected (FC) topology. As sketched in Figure 2.1, an FC is a stacked 'list' of layers whose unitary elements are neurons. In FCs, a link connects each pair of neurons in adjacent layers (hence Fully Connected): values assigned to each link have been historically named weights. Additionally, neurons can have a further incoming link/weight called bias. In the first layer, where the input X is fed, each neuron multiplies the input with its weights, sums the bias, and applies a (non-linear) activation function a. Technically speaking, the output of the j-th neuron of the ℓ-th hidden

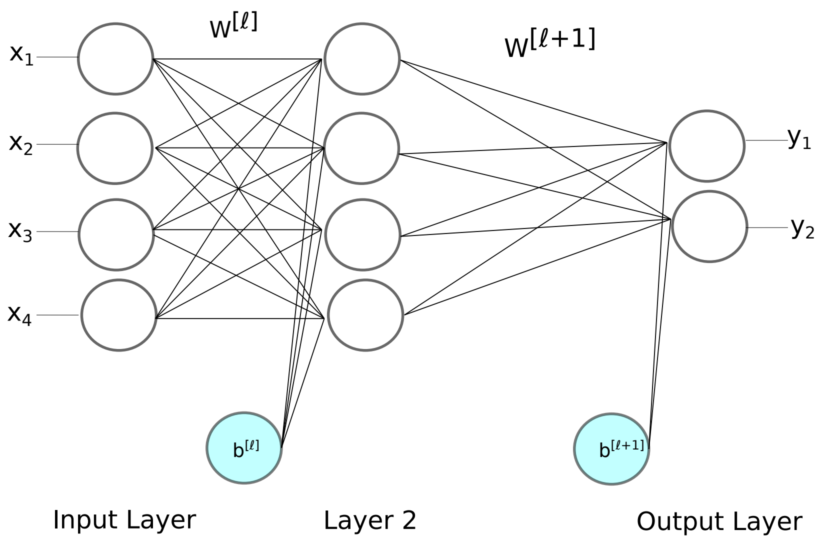


Figure 2.1: An FC architecture with one input, one output, and one hidden layer. Each input is a 4-dimensional vector, while outputs are 2-dimensional.

layer of an FC network, namely $z_j^{[\ell]}$, is computed as

$$z_j^{[\ell]} = a(b_j^{[\ell]} + \sum_{i=1}^{I} W_{j,i}^{[\ell]} z_i^{[\ell-1]}), \tag{2.1}$$

where $z_i^{[\ell-1]}$ is the output of the previous layer $(\ell - 1)$, $W^{[\ell]}$ and $b^{[\ell]}$ are respectively the matrix of weights and the bias vector at layer ℓ. Please notice that, according to our notation, $z^{[0]} = x$, while for an L-layers NN, $z^{[l]} = y$. Furthermore, the output of a layer can be expressed via the compact vectorial notation, i.e.,

$$z^{[\ell]} = a(b^{[\ell]} + W^{[\ell]} z^{[\ell-1]}), \tag{2.2}$$

where $W^{[\ell]} z^{[\ell-1]}$ expresses the dot-product between the weight matrix and the vector $z^{[\ell-1]}$. In an NN with $L > 0$ layers, $z^{[0]}$ and $z^{[\ell]}$ respectively correspond to the input x and the output y, yet since the input-output mapping that an NN learns is often an approximation of the true relationship between X and Y, it is common to denote the output of a network, when evaluated for an instance of an input point x, with $\hat{y}$.

Standard activation functions include ReLU (Glorot et al., 2011), a step-wise linear function that combines approximation capabilities with regularization effects at training time, and the sigmoid (Bishop and Nasrabadi, 2006). In classification, the output layer is often activated via the softmax function, which selects the most likely class as the maximum value index.

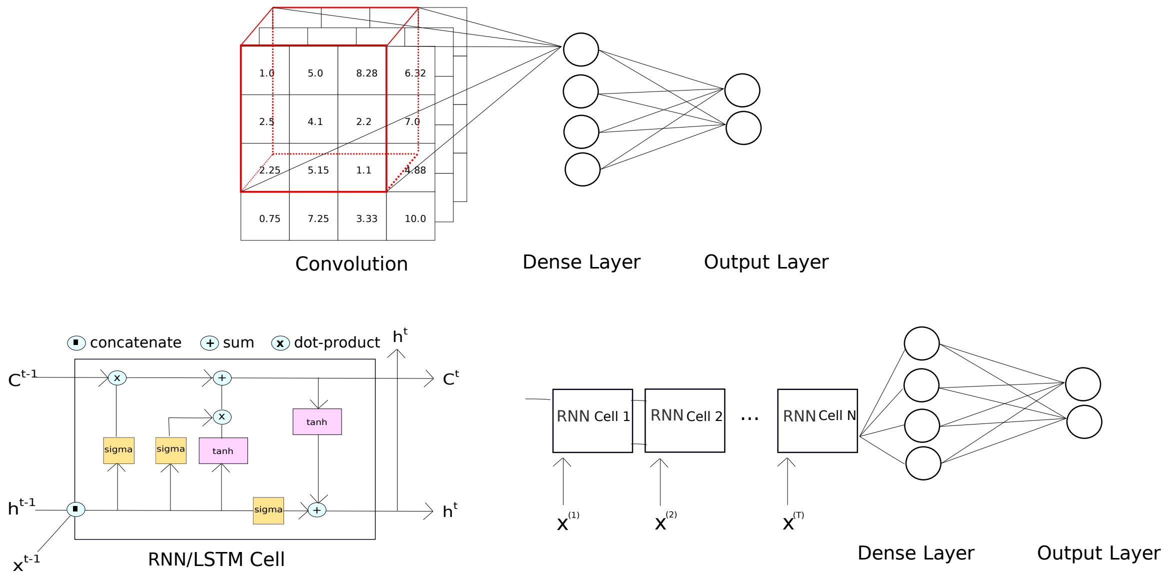


Figure 2.2: A CNN architecture that processes 3D data (top). At the bottom-left, an LSTM cell, a particular implementation of an RNN, recursively processes the input unrolling it along its 'temporal' dimension (bottom-right). While the LSTM processes an input with standard vectorial operations (as per the legend, concatenation, dot-product, sum, and layers activation), this architecture efficiently permits storing information from the previous time steps.

Neural network architectures. Neural Architecture Search (NAS) (Elsken et al., 2019) is a branch of DL where researchers develop techniques to automate the search of optimal architectural biases to solve a task that enables fast, stable, and efficient training.[1] In its simplest form, an architectural bias is a variation over the FC architecture that provides some advantages on a task: it could enable faster training time, regularize parameters that grow too big, or provide some desirable behavior that depends on the task domain. Convolutional Neural Networks (CNN) are a notable example as the convolution operator provides invariance to input scaling and are thus widely employed to solve Computer Vision tasks (Zhang et al., 2015a). Formally, a CNN layer is defined as follows:

$$z^{[\ell]} = a(b^{[\ell]} + W^{[\ell]} * z^{[\ell-1]}), \tag{2.3}$$

where the convolutional operator $*$ acts by multiplying, via classical dot-product, the whole weights matrix to each locally-connected region of the input (such as a 3D lattice), as sketched in Figure 2.2 (top).[2]

Regarding input-output pairs that require learning of long-term dependencies, Recurrent Neural Networks (RNN) (Graves, 2013) show superior performance to FCs. The idea behind RNNs is that input is fed sequentially, i.e., unrolled through one of each dimension that we call 'time', to determine the value of the next 'cell' and recursively until the last sequence produces the desired output. Formally, a 1-layer RNN is defined as follows:

$$z^{(t)} = a(b + W[h^{(t-1)}, x^{(t-1)}]), \tag{2.4}$$

where the classical operation of dot-product is now calculated between the weights matrix W and the output computed at the previous time-step, to which the successive input has been appended (i.e., $[h^{(t-1)}, x^{(t-1)}]$). Each time-step thus corresponds to a 'cell' that provides part of the output, namely the hidden-state $h^{(t-1)}$, of the successive 'cell', as shown in Figure 2.2 (bottom-right).

RNNs have been a prevalent architectural choice in the last 2/3 decades, with several variations of the original formulation that culminated in what is referred to as the Long Short-term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997), which allows, by stacking parallel FC gates inside each recurrent 'cell', to forget,

[1]Architectural biases are not related to neuron biases. An architectural bias is how neurons are displaced in an NN and process data, while bias alone refers to the term $b^{[\ell]}$ in Eq. 2.2.

[2]As they are not relevant for this thesis, we will skip the details regarding how each local region of the input is multiplied with the weights, i.e., via stride, padding, and other convolution sub-operators.

maintain and update the information stored at previous time-steps. A sketch of a Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is displayed in Figure 2.2 (bottom-left).

Another architecture the research community has positively received is self-attention networks (Vaswani et al., 2017), which captures the relationships between distant elements in a sequence by computing the so-called attention scores between each input pair and has found wide applicability in NLP and many other fields. The input-output relation in self-attention is defined as

$$\sigma(\frac{QK^T}{q})V, \tag{2.5}$$

with $Q = xW^{[Q]},\ K = xW^{[K]}$, and $V = xW^{[V]}$, with $x \in \mathbb{R}^{n\times d}, W^{[Q]}, W^{[K]}, W^{[V]} \in \mathbb{R}^{d\times n}$, σ the softmax function, and q a normalization constant that usually depends on the length of the input. Self-attention computes an importance score of each pair of input features $(x_i, x_j) \in x$: see Figure 2.3. Self-attention has become a building block for the next generation of linguistic representations that surpassed, in terms of performance on downstream NLP tasks, classic word embeddings (Mikolov et al., 2013). Self-attention is, in fact, the core block of Transformers (Vaswani et al., 2017), which stacks advanced architectural layers such as positional embeddings and layer norm in a sequential block and has permitted training, building, and deploying large-scale Language Models such as GPT and BERT (Devlin et al., 2019a; Brown et al., 2020). We illustrate the Transformers architecture in Figure 2.4 and explain its internals.

2.1.3 Training Procedures

Training an NN is an iterative process that involves an optimization algorithm and some training data to guide the process. The idea is to find an optimal set of parameters of the NN that guarantee high performance. A historical cornerstone of DL has been the discovery of a training procedure that leverages the first-order derivatives of the loss function $\mathcal{L}$, i.e., the primary measure of performance of the NN. The procedure, baptized back-propagation algorithm (Linnainmaa, 1970; Rumelhart et al., 1986), works as follows: for an input-output pair (x, y), back-propagation refines the values of each parameter w of the NN via the update $w = -\eta\nabla_w\mathcal{L}(y, f(x))$, where $\nabla_w\mathcal{L}(y, f(x))$ is the gradient of the loss function w.r.t. the parameter w, and η a small quantity. However, optimization problems in NLP and CV, and DL in general, are difficult; thus, the mapping that f learns from X to Y in order to minimize $\mathcal{L}$ is

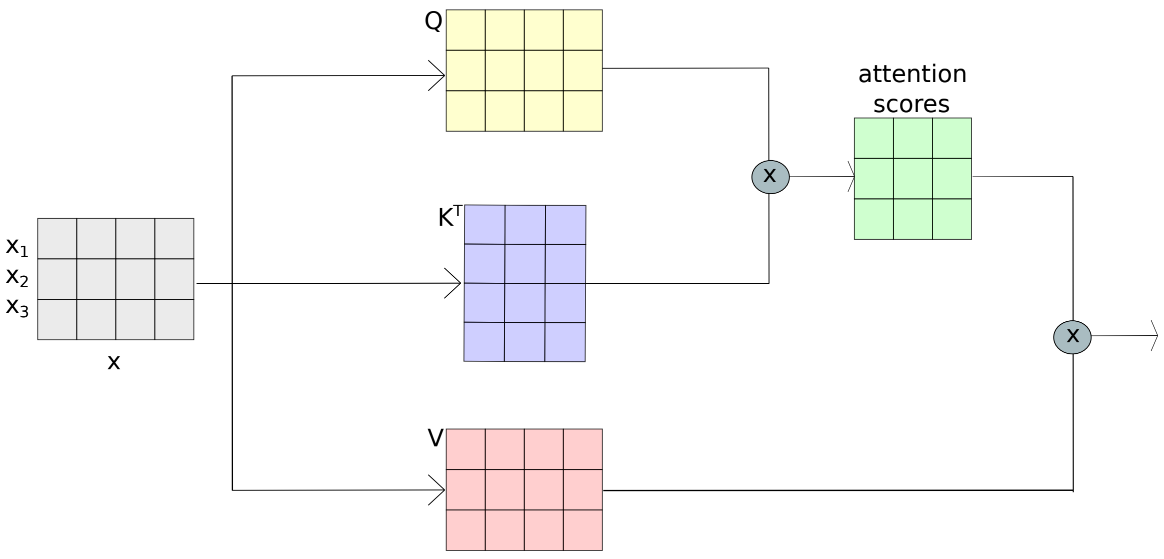


Figure 2.3: An example of the self-attention architecture, where for each pair of input features $\{x_1, x_2, x_3\}$, the module computes the attention scores (via dot-product). The attention scores (the green matrix) store an importance score for each pair of input features. Intuitively, this information is relevant to solve downstream NLP tasks where long-distance relationships would be difficult to maintain by other architectures (e.g., co-reference resolution).

often non-convex, with the insurgence of local-minima and saddle-points that slow or may even stop, the learning process.

Formally, a loss function quantifies the difference between the ground truth outcome y and the outcome $\hat{y}$ produced by the NN, namely

$$\mathcal{L} : (y, \hat{y}) \rightarrow \mathbb{R}, \tag{2.6}$$

where $\hat{y}$ is the prediction of a network on a generic input x, i.e., $f(x)$, while y is the ground-truth value. In the general case, both y and $\hat{y}$ are multi-dimensional vectors of real numbers. A loss function that is widely employed in regression problems is the Squared Error loss function, i.e., $\mathcal{L}_{SE}(y, \hat{y}) = (y - \hat{y})^2$, which computes the (average) square difference between the ground truth prediction y and the NN's output $\hat{y}$. Equally popular when solving a classification problem is the Cross-Entropy loss function, formulated as

$$\mathcal{L}_{CE}(y_c, \hat{y}_c) = -\sum_{c}^{C} y_c log(\hat{y}_c), \tag{2.7}$$

where y_c ($\hat{y}_c$) expresses the numerical value of the c-th NN's output (prediction),

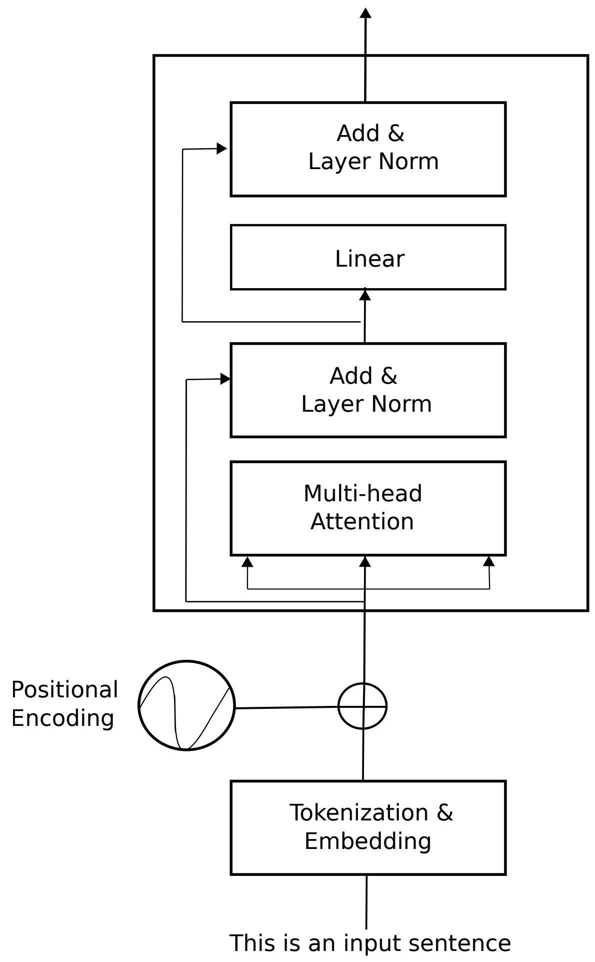


Figure 2.4: A Transformer architecture. An input sequence is initially turned into a multi-dimensional representation, then a function (the positional encoding) calculates, for each token, a value that identifies its position in the sequence. That is useful to compute self-attention at the successive layers, which, followed by a few layers that normalize (layer norm) and linearly transform the input, output a representation that has been demonstrated to be very useful to solve both low-order tasks such as sentiment analysis and language translation (despite, in this case, both the input and the expected output each passing through a transformation similar to that shown in this figure).

encoded as a one-hot vector (i.e., the i-th element of y would be equal to one, should the expected output class be the i-th, out of $C > 0$ classes).

A generic learning problem in DL involves an NN that learns the $f : X \to Y$ relationship via back-propagation on a finite amount of training data. One must solve the following problem,

$$\min \underset{(x,y)\sim(X,Y)}{\mathbb{E}}[\mathcal{L}(f(x), y)] \ , \tag{2.8}$$

where $\mathbb{E}_{(x,y)\sim(X,Y)}$ represents the expected training loss in case of regression and classification.

While it is known that a powerful enough NN can learn a mapping $f(x)$ that leads the loss in Eq. 2.8 to zero (Bishop and Nasrabadi, 2006), such approximation capability comes at the risk of *overfitting* on the training data, i.e., the model, aided by the high number of parameters, ends up memorizing each (x, y) pair. Overfitting causes a model to perform very well on the training data but produces significant errors on unseen test points. One can mitigate this problem by choosing a different NN, adding more training data, or stopping the learning procedure by adding a validation set, which patrols over an excessive growth of the loss on unseen-at-training data. In the next section, we will relate the problem of overfitting to that of robustness; but now, we will introduce some NLP background.

2.1.4 Natural Language Processing

This section reviews NLP representations, related concepts, and methods. In conjunction with linguistics considerations, these will be utilized in Chapters 5 and 6.

2.1.5 On NLP Representations

Natural Language Processing is an active area of study, where algorithms process human language in all of its forms, thus including written and oral communication. While it has nowadays grown to an independent field of research, it inherits most of its concepts from linguistics and computational linguistics. With an oversimplification, we can say that at a high level, NLP is concerned with designing scalable linguistic representations that are close to those of humans. An NLP representation is a function that maps symbols from a language to a vector of (real) numbers in some high-dimensional space, namely

$$\psi : v \in V \mapsto x \in \mathbb{R}^d, \tag{2.9}$$

where V a is a finite-length vocabulary of symbols, and v (or equivalently w, when each symbol is a word) is one of those symbols. By extension, a list of $l > 0$ symbols can be mapped by a representation, namely

$$\psi : s \in V^l \mapsto x \in \mathbb{R}^{l \times d}, \tag{2.10}$$

where s usually indicates a sequence or, in NLP, a sentence that, with an abuse of notation, can be denoted by $\mathbb{D}(V^l)$.[3] In the same way, we can denote the i-th symbol of a sentence s as s_i.

Since this thesis focuses on NNs, we will only review techniques after the advent of NNs. Nonetheless, Chapter 3 will include some key prior works for a historical perspective on the origin/adaptation of seminal linguistic ideas to NNs.

2.1.5.1 Word Embeddings

While an exhaustive history of embeddings goes beyond the scope of this section, early approaches based on latent factors distillation Schütze (1998), dimensionality reduction Dumais (2004), as well as neural-based techniques Bengio et al. (2000) had a profound impact on recent advancements in developing efficient linguistic representations. A significant step towards NN-based representations that capture rudiments of human semantics has been achieved with Word2Vec (Mikolov et al., 2013). In Word2vec each representation is a fixed (i.e., once it is learnt, it doesn't change at inference time) multi-dimensional vector whose values depend on the context, i.e., the surrounding text. This is a direct application of the distributional hypothesis (Harris, 1954): words with similar contexts are represented similarly. Each symbol/word representation $x = \psi(v)$, $\forall v \in V$, within the Word2Vec framework is computed to minimize the following,

$$\underset{\psi(v)}{\operatorname{argmin}} \; log \; \sigma(\psi(v)^T \psi(c_v)) + \sum_{v' \in V} log \; \sigma(-\psi(v')^T \psi(c_v)) \; . \tag{2.11}$$

In Eq. 2.11, $\psi(v)$ is a d-dimensional vector (and the objective of the Word2Vec optimization) that maximizes the distance, expressed as the dot-product plus a non-linear function σ, between v and its surrounding context c_v: the context c_v is a number of words that surround v, and in the limit can be an entire sentence s, or even larger. Unfortunately, such methods' computational intractability limits the context's length

[3]Despite their fundamental importance in formal languages and logic, in this work we will not discuss the rules underlying the membership decidability of a series of symbols from a vocabulary (e.g., a sentence) within a language.

to a few words, thus making word embeddings local methods. To prevent an algorithm from finding a naive solution for Eq. 2.11, the second summand, often referred to as negative-sampling, prescribes that v', i.e., words sampled randomly from vocabulary V, are kept distant from their original context c_v. Word2Vec training procedure can also be interpreted via information theory (Goldberg and Levy, 2014), i.e., as the procedure to find a low-rank representation of the word-to-word co-occurrences as they appear in a text corpus.

While Word2Vec is a method that leverages mainly the local semantic/syntactic information of a text, some popular methods mix this with global information on word co-occurrences, producing some of the most advanced word embedding representations (Pennington et al., 2014b).

2.1.5.2 Attention-based Large Language Models

The last five years (2017-2022) have seen the rise of attention-based models (Vaswani et al., 2017), trained on massive text corpora.[4] While they inherited many aspects from word embeddings, LLMs benefited from advances in large-scale GPU-based infrastructure and increasingly complex architectures capable of building richer and more context-aware language representations (Liang et al., 2018). LLMs have many competitive factors over standard word embeddings: while we discuss them in the next chapter, alongside a historical overview of a field that is running at a fast pace, here we focus on the training technique, which, similarly to Word2Vec, aims at building rich linguistic representations of symbols.

A generic LLM learns to predict 'masked' words in a sentence by minimizing the Cross-Entropy loss function (see Eq. 2.7) between the NN prediction and the ground-truth empirical probability of a word to appear in that context. A difference between LLMs and word embedding techniques is that an LLM is trained to output a probability distribution ϕ over 'masked' words. Thus the representations ψ are computed as a by-product of such an operation. Formally, an LLM objective function optimizes

$$\min \ -\phi(v|c_v, v = \mathbf{m}) \ log \ Prob(v|c_v), \tag{2.12}$$

where $\phi(v|c_v, v = \mathbf{m})$ is the output of the LLM when it takes as input some context (e.g., a sentence), it masks a word v with a special token $\mathbf{m} \in V$ (via the operation $v = \mathbf{m}$), while $Prob(v|c_v)$ is the empirical probability of the masked word v to appear

[4]BERT, a widely popular language model (Devlin et al., 2019a), has been trained on 180GB of raw text, several orders of magnitude more than the amount of information an average human is exposed to in their entire life.

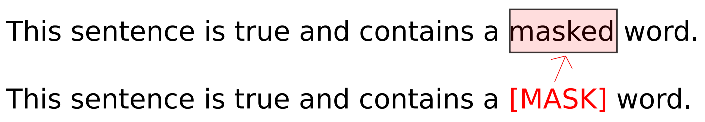

This sentence is true and contains a masked word.
This sentence is true and contains a [MASK] word.

Figure 2.5: An example of a masked sentence. This masking technique has become very popular (Devlin et al., 2019a) and encompasses multiple masked words per sentence.

in that context. In doing so, attention-based LLMs (but not only (Liu et al., 2021)) learn useful hidden representations of a sentence, which we refer to, consistently with Eq. 2.10, as $\psi : s \in V^l \mapsto \mathbb{R}^{l \times d}$. We show an example of a masked sentence in Figure 2.5. Another popular category of LLMs is auto-regressive models, such as GPT (Brown et al., 2020), which optimize the prediction of the next word, given what has been generated so far by the model. In this sense, they are better generative models, while LLMs such as BERT are good at predicting masked words anywhere in a sentence. For the sake of this work, we can assume w.l.o.g. that LLMs are masked LLMs.

Static vs. dynamic representations. The attention layer, and so most of all modern LLMs, comes with an architectural bias that aims to preserve positional information throughout all its hidden layers. In this sense, in an NN, where multiple attention layers are stacked, one can use each as standalone word embedding (Peters et al., 2018; Devlin et al., 2019a). This represents a difference with Word2Vec-based NNs, as the representations in hidden layers of topologies such as FC, CNN, RNN, do not maintain a consistent position through each layer. While some works pointed out that static representations can compete with attention-based LLMs (Liu et al., 2021), without mentioning the fact that FCs are themselves universal approximators and can hence mimic attention, the latter demonstrated to be more data-efficient while allowing for inspection of the information encoded by each layer. Several works leveraged this enhanced 'inspectability' (which one must not confound with interpretability (Jain and Wallace, 2019)) to conduct impactful linguistics analyses on the LLMs' internals (Manning et al., 2020).

Probing tasks. *Probing tasks* are 'thermometers' to test the capabilities of an LLM on a task for which it has not been specifically trained (Niven and Kao, 2019). Technically speaking, a model h of parameters θ solves a *probing task* for pairs of input-output $(s, y) \in (S_{\mathbf{T}}, Y_{\mathbf{T}})$ by learning an optimal mapping between a linguistic representation $\psi : S_{\mathbf{T}} \to X$ and the labels $Y_{\mathbf{T}}$. Formally, a probe learns to minimize the following loss function:

$$\min \mathcal{L}_{\mathbf{T}}(\hat{y}, y), \tag{2.13}$$

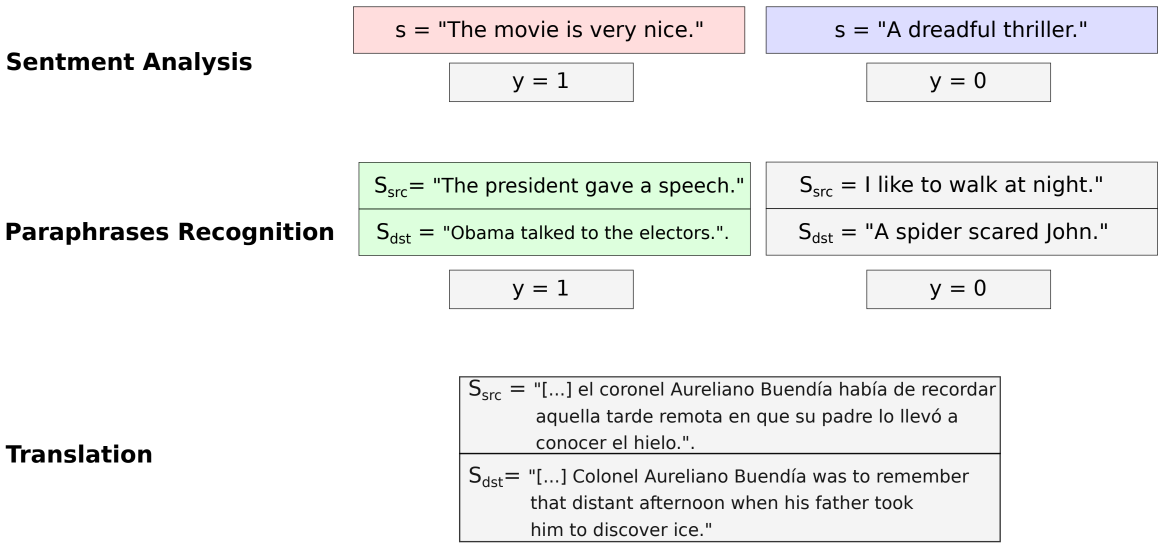
where $\hat{y} = h(\psi(s))$ is the prediction of a network equipped with a representation ψ, while $y \in Y_{\mathbf{T}}$ is the ground truth value/label associated with an input $S_{\mathbf{T}}$. While in general, any (non-linear) mapping $h(\psi(s))$ between the input representation and the expected output can be a probing task (Pimentel et al., 2020; White et al., 2021), they are usually linear to show how much of the linguistically helpful information to solve a task is immediately available from a representation ψ.

As a side note, any NLP task can be a probe, from sentiment analysis to language translation, while any word embedding whose values are frozen at training time is, in principle, a probe. However, probing tasks show how a linguistic representation provides, in an immediately accessible way, linguistic information to solve the task competitively. The following section introduces some of these 'core NLP tasks'. Finally, the language translation task is reported as a case of a high-order NLP task, i.e., one that requires powerful models and linguistic representations and for which defining, measuring, and improving robustness is an open problem.

2.1.5.3 NLP Core Tasks

This subsection provides an overview of some of the most studied NLP tasks for which robustness seems necessary. We will describe three settings: sentiment analysis, paraphrase recognition, and language translation.

Sentiment analysis. Binary classification, which in NLP is well exemplified by sentiment analysis, is one of the most studied problems in NLP: given a set of sentences written in natural language, an algorithm is required to distinguish those that belong to a particular semantic class (e.g., positive movie reviews) from all the others (e.g., negative movie reviews). While usually considered a low-level task, sentences in sentiment analysis can be expressive enough to make the task challenging even for modern NLP techniques (Barnes et al., 2019). Binary classification falls under the broader category of multi-class classification should the ground truth classes be more than 2 (such as neutral, positive, and negative). In Figure 2.6 (top), we sketch some examples of the task.



Sentment Analysis
s = "The movie is very nice."
y = 1
s = "A dreadful thriller."
y = 0
Paraphrases Recognition
Ssrc= "The president gave a speech."
Sdst = "Obama talked to the electors.".
y = 1
Ssrc = I like to walk at night."
Sdst = "A spider scared John."
y = 0
Translation
Ssrc = "[...] el coronel Aureliano Buendía había de recordar aquella tarde remota en que su padre lo llevó a conocer el hielo.".
Sdst= "[...] Colonel Aureliano Buendía was to remember that distant afternoon when his father took him to discover ice."

Figure 2.6: A few examples of NLP tasks. Top: sentiment analysis. Middle: paraphrases identification. Bottom: language translation.

Paraphrase recognition. Another task that requires some non-trivial manipulation capabilities of natural language is paraphrase recognition, where an algorithm must decide whether a given pair of sentences share the same meaning. Formally, an NN learns the mapping $f : (S_{src}, S_{dst}) \rightarrow Y \in \{0, 1\}$, where (S_{src}, S_{dst}) are respectively the input sentences that are assigned to be either paraphrases (class 1) or non-paraphrases (class 0). Similarly to binary classification, when multiple classes of paraphrases are present (e.g., forward vs. reverse entailment (Williams et al., 2018)), one can denote the problem as a multi-class paraphrases recognition task instance. In Figure 2.6 (middle), we sketch some examples of the task.

Language translation. Last but not least, language translation is a high-order NLP task that constitutes the backbone of commercial products millions of people use daily. In its simplest form, it takes a sentence as input and produces its translation to a target language as output. Formally, an NN learns the mapping $f : S_{src} \rightarrow S_{dst}$, where both the input and the output are sentences written in natural language, though possibly in different languages (e.g., English to Chinese). In Figure 2.6 (bottom), we sketch some examples of the task.

2.2 Robustness

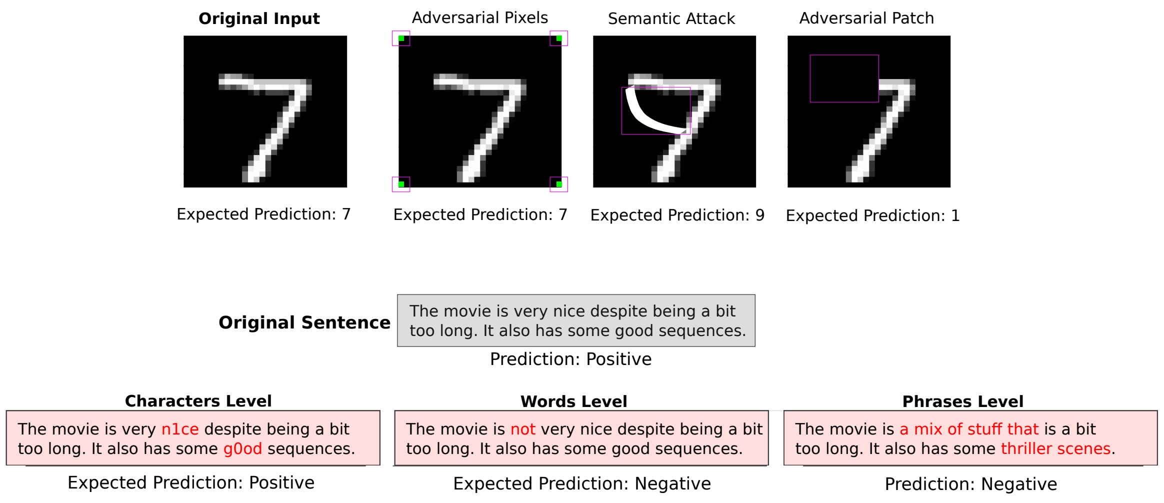
The main reason behind DL's success is its approximation capability of phenomena ruled by complex behaviors. On the other hand, other factors risk impacting how successful the NN framework will be in the long term, and one of them is undoubtedly sensitivity to the initial conditions. In physics, a chaotic model is one whose final conditions are heavily affected by slight variations of its initial state. With an analogy, in DL a model whose output greatly varies on local perturbations of an input is said to be non-robust or brittle. A few simple observations on the brittleness of widely adopted DL models (Biggio et al., 2013; Szegedy et al., 2014b) have given rise to a flourishing field of research named adversarial robustness, which we review in this chapter, with particular emphasis on NLP techniques.

We first introduce adversarial attacks, which constitute a cogent motivating case for local adversarial robustness, for which we present measurement techniques first, then methods to enhance robustness for a generic NN model. In Chapter 5, we challenge the standard *de-facto* notion of robustness in NLP, with one aligned with linguistics. Furthermore, in Chapters 5 and 6, we will adapt local robustness to encompass linguistic aspects of communication, such as syntax and semantics.

2.2.1 Adversarial Attacks

Similarly to Section 2.1.4 and when not mentioned explicitly, adversarial attacks will be presented under the assumption that we are in a supervised classification setting, where an NN learns to associate each input $x = \psi(s)$ to a category/class $y \in Y$. A classic case is a binary classification, as previously introduced. While Section 2.3 will explore the attacks that can be conducted at the representation level, they primarily concern the representation provided by an embedding or an LLM.

Despite NNs being accurate at solving increasingly complex tasks, there is literature that has furnished overwhelming empirical evidence that, for these models, there are cases where $f(x) \neq f(x+\gamma)$, despite the magnitude of the so-called 'perturbation vector' γ (Szegedy et al., 2014a) being small. Adversarial attacks, which we now informally define as perturbations of an input point that produce an NN's misclassification, have emphasized the inconsistent behavior of DL models when tested on out-of-distribution data, as well as outliers, but also on inputs that we would expect to be correctly classified. In Figure 2.7, we provide a few examples of attacks and perturbations in CV and NLP: as it is clear from the variety of inputs reported, the



The movie is very nice despite being a bit too long. It also has some good sequences.

The movie is very n1ce despite being a bit too long. It also has some g0od sequences.

The movie is not very nice despite being a bit too long. It also has some good sequences.

The movie is a mix of stuff that is a bit too long. It also has some thriller scenes.

Figure 2.7: An example of the difficulty of the problem of robustness in CV (top) and NLP (bottom). Some adversarial examples on top are expected to change the classifier's decision, while others should not. The same applies to the NLP examples. In this sense, robustness has to consider whether the ground truth label, i.e., the semantics of the perturbation, has been preserved.

notion of robustness requires flexibility to adapt to different, yet far from trivial, scenarios.

Before we introduce adversarial attacks, we define the notion of ϵ-Ball as the set of points that, w.r.t. an ℓ_p-norm of choice, lie in the proximity of an input x. Formally,

$$Ball(x, \epsilon) = \{x' \in \mathbb{D}(X) \ . \ ||x - x'||_p \leq \epsilon\}, \tag{2.14}$$

where $||x - x'||_p \leq \epsilon$ constrains the point that induces a misclassification to be distant, concerning an ℓ_p-norm of choice, from the original input x by at most a (small) quantity ϵ.

An adversarial attack is then defined as a point x' in the vicinity of input x such that

$$x' \in Ball(x, \epsilon) \ . \ f(x') \neq f(x), \tag{2.15}$$

As previously reported, adversarial attacks are a critical concern for NNs, as their approximation capabilities come at the cost of 'jagged' decision boundaries between output classes, as sketched in Figure 2.8.

Gradient-based attacks. Since NN's learning paradigm is based on calculating the prime (or second) order derivative of the loss function w.r.t. each network parameter, in the same way, one can find a bounded perturbation of an input point

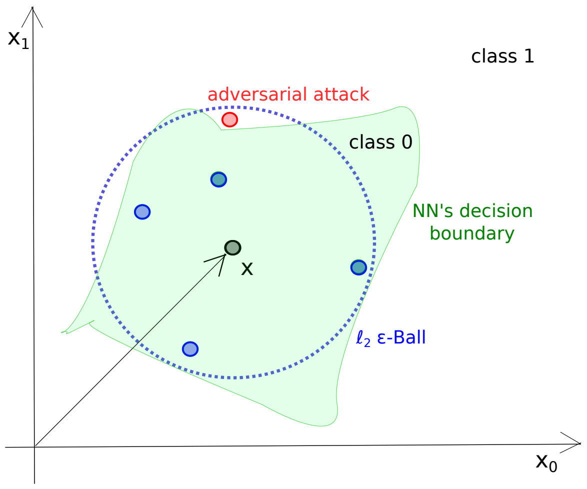


Figure 2.8: An example of a projection of an NN's decision boundary (class 0 in lime vs. class 1) w.r.t. the input points. The NN admits an adversarial attack (red point) inside an ϵ-Ball of the input x (boundaries in blue).

that is, w.r.t. the ground truth label, maximally misclassified. A simple yet widely employed method to induce a misclassification is the Fast Gradient Sign Method (FGSM), which exploits the first-order derivative of an NN function, namely

$$x' = x + \epsilon \; sign(\nabla_x \mathcal{L}(y, f(x)), \tag{2.16}$$

where *sign* is the signum function, and ∇_x is the first order derivative of the loss function w.r.t. the input point x, scaled by the value of ϵ so that the attack x' is contained in a neighborhood of x, as formally defined in Eq. 2.17. While this method potentially produces guaranteed adversarial perturbations, one cannot conclude that a network is robust should it return a point that does not induce misclassification: FGSM is *sound* but not *complete*. Other methods explore the interior of an input neighborhood by iteratively recomputing the gradients, for example, Projected Gradient Descent (PGD): despite being empirically more effective, as FGSM will always return a point at the edge of the neighborhood, PGD incurs higher computational costs while still not being *complete*.

2.2.2 Adversarial Robustness

Motivated by the previous section, we now give an introduction to robustness. For the previously defined definition of ϵ-Ball, we introduce the notion of local adversarial robustness, which we then extend to encompass a generic measure of (local) distance. We then discuss one method to quantify the robustness of a model, namely bound relaxation, and two techniques to enhance robustness, namely augmented training and Interval Bound Propagation (IBP). We next discuss the ontological differences between robustness in Computer Vision and language. We will then review the adaptations of IBP and other robustness techniques to NLP. We conclude with an overview of explainability and brittle explanations, as this will serve as a basis for Chapter 7.

2.2.2.1 ϵ-Ball Robustness

An NN f is said to be locally ϵ-robust, with respect to an input point $x \in \mathbb{R}^d$, when for a small positive number $\epsilon > 0$, and an ℓ_p-norm of choice, it holds that

$$\forall x' \in \ Ball(x, \epsilon), \ f(x) = f(x'). \tag{2.17}$$

By extension, one can employ other distance measures than the ℓ_p-norm, to capture different notions of proximity within an input neighborhood.

2.2.2.2 Beyond ϵ-Ball Robustness

As some examples in Figure 2.7 anticipated, the notion of robustness, as well as that of adversarial attacks, can vary in nature: from perturbations that act on input proximity (see Eq. 2.17) but require a model to be consistent with the original label, to those that change the ground truth label. An often underestimated threat is that an NN's decision boundary can vary reasonably with slight perturbations of an input. In the same way, we defined ϵ-robustness (Eq. 2.17) as a notion that assesses a model invariance against points that lie in the proximity of the input, we now extend it to encompass a generic distance function $dist : (x, x') \rightarrow \mathbb{R}^+$, namely

$$\forall x' \ . \ dist(x, x') \leq \epsilon, \ f(x) = f(x'). \tag{2.18}$$

When the distance function $dist$ is further constrained to be an ℓ_p norm, Equations 2.18 and 2.17 become equivalent (trivially, a norm is always a distance in $\mathbb{R}$).

2.2.2.3 Robustness Guarantees and Robust Training

We begin by describing a technique to quantify the robustness of a model; then, we discuss two approaches to enhance robustness, namely adversarial/augmented training and interval bound propagation (IBP).

Bound Relaxations. This method over-approximates an NN's activations with a sequence of linear functions for which robustness guarantees can be computed (while in general, the problem is NP-hard (Katz et al., 2017a)). Bound relaxations compute the maximal region of the robustness of a linearly approximated NN, thus guaranteeing that, for that region, no perturbation changes the classifier's output. Technically speaking, bound relaxations compute

$$\underset{\epsilon^*}{\operatorname{argmax}}\, Ball(x, \epsilon^*),\ \forall x' \in Ball(x, \epsilon^*) \ \wedge \ f(x) = f(x'), \tag{2.19}$$

where two affine transformations $\{(A_{lb}, B_{lb}), (A_{ub}, B_{ub})\}$ over- and under-approximate the input-output relation expressed by the each NN's neuron, namely

$$\{(A_{lb}, B_{lb}), (A_{ub}, B_{ub})\}\ .\ A_{ub}x + B_{ub} \geq f(x) \geq A_{lb}x + B_{lb}. \tag{2.20}$$

Each term in Equation 2.20 represents an approximation of a neuron's output using a hyperplane, thereby serving as either an upper or lower bound for the neuron's activation value.
Each NN layer is recursively approximated by a set of upper and lower bound transformations, defined to bound in turn the output of the previous layer, namely $A^{[l]}_{ub} z^{[l]} + B^{[l]}_{ub} \geq f^{[l]}(z^{[l]}) \geq A^{[l]}_{lb} z^{[l]} + B^{[l]}_{lb}$. In this setting, certifying whether an over-approximated network is robust is computationally feasible as the linear approximations correspond to a convex input-to-output mapping. Generic certification tools, i.e., valid for any NN block (FC, CNNs, etc.), might be loose and thus not practically useful: to overcome this limitation, methods are designed specifically for each architectural bias, as in the case of CNNs (Boopathy et al., 2019) and LSTMs (Ko et al., 2019a), and known as CNN-Cert and POPQORN, respectively. A sketch of how an upper and a lower bound are mapped to a convex region of the output space for an NN's block is reported in Figure 2.9. Various approaches in literature aim to estimate the Lipschitz constant of a neural network, which quantifies the maximum change in the network's output resulting from a local change in the input (e.g., $max_{x \in Ball(x,\epsilon)}(\frac{|f(x) - f(x')|}{|x - x'|})$). Accurate estimation of the Lipschitz constant enables the calculation of robustness guarantees by discretizing the search space for adversarial attacks. Although these methods have been applied in numerous studies Wu et al. (2020a), the decision boundaries of complex models are often so fragile that the estimations become unreliable. Consequently, we exclude these approaches from consideration in this thesis.

In Chapter 4, we will see how to tighten this bound in NLP via an iterative process that leverages bounding techniques and adversarial attacks, to finally produce an optimal certificate of the robustness of an NN.

Adversarial/augmented training. Augmented (adversarial) training mitigates the brittleness of an NN by inserting and back-propagating, at training time, samples meant to be adversarial, along with the correct ground truth label. A straightforward approach consists of augmenting the training data with samples perturbed via FGSM (Eq. 2.16), with the assumption that local variations of an input belong to the same class.

Unlike adversarial training, augmented training collects samples that are not necessarily built to induce an NN misclassification, and it is a more general technique to make a model robust, yet without explicitly injecting malicious examples whose ground truth label has been rectified. In Chapter 5, we will see how augmented training can benefit robustness, especially on edge cases ruled by linguistic phenomena.

Regarding formal guarantees, both augmented and adversarial training do not guarantee a network to become robust to the misclassification of an input point, not even to those samples that have augmented the training data.

Interval Bound Propagation. Similarly to bound relaxation, IBP produces a certificate of robustness for an NN using axis-aligned boxes. IBP can also be used during training, where it can be incorporated into the loss function to enhance a network's robustness by maximizing both the performance on the training task and the local robustness. Formally, by using IBP, one minimizes the following loss function:

$$min\ \mathcal{L}(y, f(x)) + \mathcal{L}_{IBP}(y, f, Ball(x, \epsilon)), \tag{2.21}$$

where $\mathcal{L}(y, f(x))$ is the standard loss on the training task, while $\mathcal{L}_{IBP}(y, f, Ball(x, \epsilon))$ quantifies how close the network satisfies the requisites of local robustness.

IBP has been designed to enhance CV robustness and then adapted to NLP. In the context of NLP representations, IBP assumes that points close to the input are those for which one wants to guarantee robustness. As we will show in Chapter 5, this notion of local robustness is simplistic in the context of NLP and offers guarantees that can be, in practice, not linguistically meaningful.

2.3 Robustness and Language

In CV, local perturbations can be interpreted as slight variations of a pixel in an image, with the non-trivial consequence that local robustness corresponds to a real, interpretable threat (e.g., the brightness of an image is manipulated to induce an NN misclassification). In NLP, this connection shows up differently: while language is an

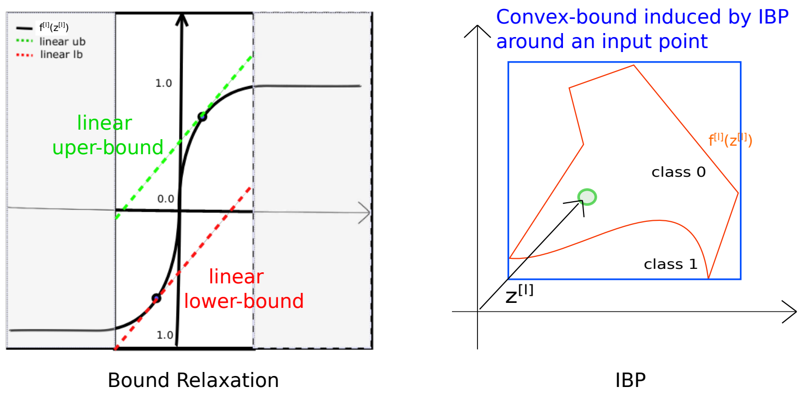


Figure 2.9: Linear bound propagation (left), with the upper- and lower-bound highlighted respectively in lime and red (for details, check Eq. 2.20), allows one to certify whether a model, whose non-linear activations maps inputs to non-convex regions of the output space, through a convex over-approximation of that space. Standard methods allow the certification of the robustness of multi-layer NNs activated via non-linear functions. IBP (right) can be used to leverage this idea at training time by propagating a convex approximation (in blue) of the projection of an input (in lime) neighborhood to the decision boundary (in red), which is then back-propagated to induce the decision of the model to be uniform over that region.

infinite yet discrete system of symbols, NN's representations are continuous. With an analogy to number theory, the cardinality of the sentence space resembles that of natural numbers, while the hidden representation's that of reals. In this sense, despite some exceptions that have not found much usage in the literature (Vilnis and McCallum, 2015), there is no consistent linguistic interpretation of an attack unless it is carried at the character/word/sentence level. Furthermore, slight perturbations of a text produce utterances where the original semantics is twisted or not reversed. One example comes from linguistics: the sentences `'He appeared to Lisa to be brave'` and `'He appealed to Lisa to be brave'` differ in one character, yet have opposite meanings (Chomsky, 2014b).

In the next section, we discuss techniques considered standard by the research community. We then focus on aspects of robustness that differ from the standard notion employed in CV. When a notion is inherited *as is* from DL or CV, we work with the definition provided in Section 2.2.

We first introduce a taxonomy of adversarial examples in NLP, which motivate the development of tools to measure and enhance robustness against attacks to NLP models.

2.3.1 Adversarial Attacks in NLP

This subsection reviews how adversarial methods can be adjusted to work with discrete sets of symbols, such as words. It is thus helpful to introduce a refinement to the notion of ϵ-Ball that encompasses only those representations that admit an inverse image in the vocabulary space, namely a discrete Ball, called DBall.

Given an input vocabulary of symbols V, a linguistic representation $\psi : s \in V^l \mapsto X \subseteq \mathbb{R}^{l \times d}$, a discrete-Ball is defined as

$$DBall(s, \epsilon) = \psi(V^l) \cap Ball(x, \epsilon). \tag{2.22}$$

In Eq. 2.22, $\psi(V^l)$ is the set of representations of all possible l-long sentences. At the same time, $Ball(x, \epsilon)$ is the exact definition of ϵ-Ball defined in Eq. 2.14, applied to the embedded representation of s.

Local discrete attacks. Local attacks identify those perturbations that act locally (e.g., see Eq. 2.14) and induce a misclassification by manipulating only those vectors that have an inverse image in the vocabulary space (e.g., in the case of a word embedding, vectors that admit a word inverse image). As described in Section 2.2.1, one way to find an adversarial attack w.r.t. an input point and a label is by exploiting the gradient of an NN loss function, as in Def. 2.16. Unfortunately, gradient-based attacks act at the representation level and thus do not guarantee for that point the existence of an inverse image in the vocabulary space. One can enforce that, by searching an attack via FGSM, PGD, or genetic algorithms (Alzantot et al., 2018), then by filtering those with a vocabulary inverse image. Formally, a local discrete attack is defined as

$$\forall s' \in DBall(s, \epsilon),\ \ f(\psi(s)) \neq f(\psi(s')). \tag{2.23}$$

where s and $x = \psi(s)$ are the input sentence and the correspondent linguistic representation. An example of an attack found according to this method is shown in Figure 2.10. Unfortunately, even in this case, there is no guarantee that the discrete version of a local attack causes a misclassification, thus making the technique, other than linguistically unprincipled (we will discuss this in the following subsection and extensively in Chapter 5), not *sound*.

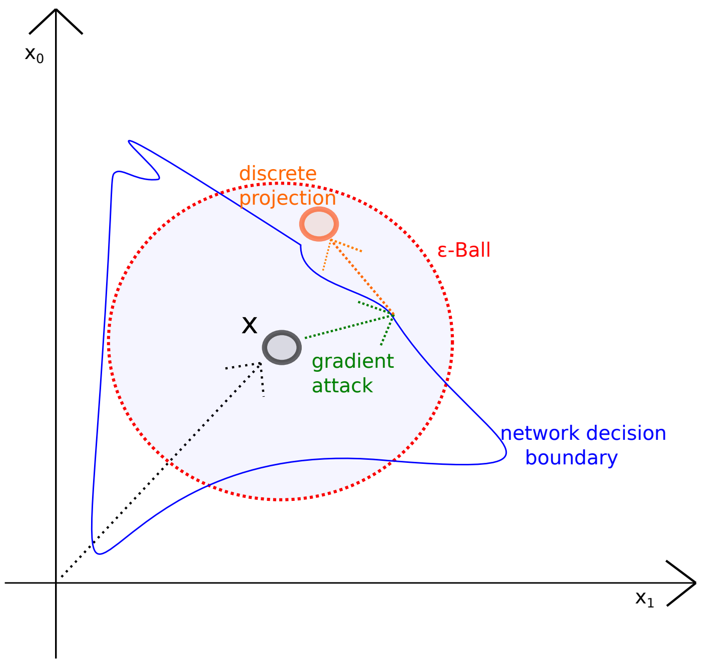


Figure 2.10: A graphical representation of an attack conducted via FGSM against an input point x, that identifies, within an ϵ-Ball (red), a successful perturbation (green). If that perturbation does not correspond to a discrete inverse image in the vocabulary space of the representation, it is then projected to the closest point that admits it (orange).

k-distance linguistic perturbations. As NLP models can be targeted by discrete and continuous adversarial attacks, i.e., w.r.t. sentences or their representations, guarantees are to be granted at either embedding/representation or symbols level. It is useful to define a generic set of perturbations that can describe the majority of the perturbation-based strategies in adversarial NLP: given an l-long sentence $s \in V^l$, and a positive integer $k > 0$, we denote the set of at-least-k distant sentences as

$$S_{perts} = \{s' \;.\; s' \in V^l \;\wedge\; \sum_{i}^{l} \mathbf{1}_{s_i \neq s'_i} \leq k\} \tag{2.24}$$

where $\mathbf{1}_{s_i \neq s'_i}$ checks whether s and s' differ on their i-th symbol.

In Chapter 5, we will show how this definition can be used to define substitution and deletion-based perturbations. Unfortunately, both these sets are not linguistically expressive enough to capture even simple linguistic phenomena such as negation, not to mention paraphrases. We now take the opportunity to discuss the problem of preserving the label when perturbing an input text.

2.3.2 Local Robustness

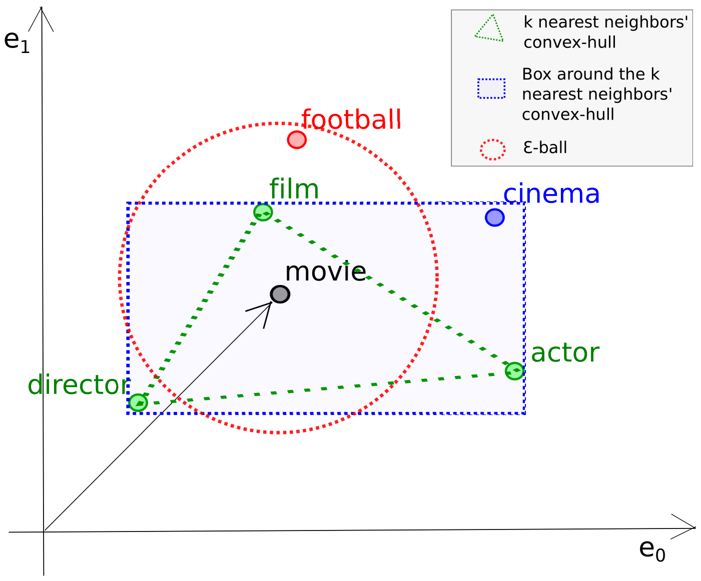
While the continuous version of robustness in NLP is the same as in CV (see Eq. 2.18), we need an adaptation to language encompassing only discrete, symbol-based perturbations of an input sentence. We begin by describing a notion of robustness that provides guarantees of safety over the convex region that contains, for each input word, the k nearest neighbors in terms of words in a linguistic representation to introduce finally local discrete robustness.

knn box and knn box robustness. Given an input vocabulary of symbols V and a linguistic representation $\psi : s \in V^l \mapsto X \subseteq \mathbb{R}^{l \times d}$, a knn-Ball is defined as

$$\textit{knn-Ball}(s) = BB(\psi(knn(s))), \tag{2.25}$$

where $BB(\cdot)$ is the minimum bounding box for each word of an input sentence $s = \{w_1, .., w_l\}$, and $knn(w)$ is the set of the k closest words to a generic input word w in the representation space, i.e., words w' with smallest $dist(x_w, \psi(w'))$, where $dist$ is a valid notion of distance between embedded vectors, such as ℓ_p-norm or cosine similarity, and x_w is the embedded word corresponding to w.[5] This provides an over-approximation of the knn convex closure, for which constraint propagation (and

[5]even though the box closure can be calculated for any set of embedded words.


k nearest neighbors' convex-hull
Box around the k nearest neighbors' convex-hull
ε-ball
e1
e0
football
film
cinema
movie
actor
director
Figure 2.11: A graphical representation of a knn-Ball perturbation set w.r.t. the ℓ_∞ norm (blue), the convex hull around the input word (green), and the ϵ-Ball.

thus robustness checking) is more efficient (Jia et al., 2019; Huang et al., 2019). An illustrative example of knn-Ball is depicted in Figure 2.11.

Similarly to what we have done before, we can then define the knn box robustness for a model as

$$\forall x' \in \textit{knn-Ball}(s), f(\psi(s)) = f(x'). \tag{2.26}$$

Local discrete robustness. In the light of Eq. 2.22, we can finally define the notion of discrete robustness for an input text s, as the property of a model to consistently assign the same label to all the local, discrete variations of such an input. Local discrete robustness is defined as

$$\forall s' \in DBall(s, \epsilon), f(\psi(s)) = f(\psi(s')). \tag{2.27}$$

2.3.3 Explainability and Robustness

Practically speaking, explainability aims to make the model's decisions intelligible via a surrogate technique that highlights only some of its peculiar aspects (Danilevsky et al., 2020; Burkart and Huber, 2021). It is crucial here to anticipate that a model's decision does not coincide with its output, yet it encompasses it. In other words, a model's decision is the process that turns an input into output, and explainability aims at making this process interpretable to humans. Similarly to robustness, the explanation of a single input point is said to be local, and when it is representative

Input s

"I like this movie as it contains some good scenes."

Explanation E

{like, movie, good, scenes}.

Input Features F

{I, like, this, movie, as, it, contains, some, good, scenes}.

Left-out F \ E

{I, this, as, it, contains, some}.

Figure 2.12: Example of an NLP input s, its features set F, a feature-based explanation, and the left-out set $(F \setminus E)$. While the input text s is a sentence, its features are a set of words (and possibly their position to allow repetitions). Finally, a candidate explanation is a subset of F, while the left-out variables are not included in the explanation.

of multiple instances, it is said to be global. This section focuses on the relevant background necessary to present the contributions of Chapter 7, which consider, in particular, feature-based, *post-hoc* explanations.

While we will focus here on feature-based explainers and two prominent examples from this category, in Chapter 3/Section 3.3, we discuss a taxonomy to frame the different explainability techniques proposed by the DL research community.

2.3.3.1 Feature-based Explanations

Given an NN that solves a classification task for a pair of input-output (x, y), and the set of input features x_F (or F, for the sake of conciseness), a feature-based explanation imposes a partition of the input features into (at least) two sets, the explanation $E \subseteq F$ and the left-out variables $F \setminus E$. We keep this notation consistent when dealing with NLP so that a feature-based explanation E is defined as $E \subseteq F$, with F the input text partitioned into its constituent symbols from V. An example of a feature-based explanation is sketched in Figure 2.12.

A different connotation through which one can dissect explainability methods is via its relationship with the NN model. An explainer is, in fact, directly interpretable, or *self-explanatory*, when the explanation is produced at the same time as the prediction, as opposed to *post-hoc* methods, which take as input the model, the input, and then output the explanation. In the next paragraph, we discuss two widely employed feature-based explainability methods, which will recur throughout the thesis, particularly in Chapters 4 and 7.

LIME and Anchors explainers. LIME and Anchors (Ribeiro et al., 2016, 2018a) are two widely employed explainers that extract *post-hoc*, local explanations for NNs. LIME approximates the local decision boundary of an NN via a linear surrogate model g: for an input point x and its label y, LIME samples a set of points in the proximity of

x and, according to the NN output, computes a linear plane that separates instances that belong to different classes. Formally, LIME computes

$$\underset{g \in G}{\operatorname{argmin}} \, \mathcal{L}(f, g, X_{pert}) + \Omega(g). \tag{2.28}$$

In Eq. 2.28, g is the model, chosen from a class of linear surrogates G, that minimizes the loss $\mathcal{L}$ between the model's prediction f and a set of locally perturbed variations of x, X_{pert}. Finally, Ω accounts for the complexity of the chosen surrogate model, in line with Occam's razor principle (the simpler, the better). As we will show in Chapter 4, a linear approximation of an NN's decision boundary can produce an explanation that does not include the features that are the most brittle, for which manipulation allows to change the decision of the classifier while maintaining the same LIME explanation. A sketch of how LIME works is depicted in Figure 2.13. Furthermore, an explanation produced by LIME is hard to interpret when adapted to NLP, as in that case, proximity sampling poorly represents the decision boundary of an NN as those samples do not necessarily possess an inverse image in the vocabulary space.

Anchors succeeds LIME in overcoming this limitation. The idea is to find a subset of the input features $E \subseteq F$ that maximizes *precision* and *coverage* metrics. *Precision* is defined as the number of times a model predicts label y solely based on the presence of the features in the explanation, i.e., by leaving the left out features $(F \setminus E)$ free to vary. On the other hand, *coverage* accounts for how frequently the candidate explanation occurs in practice. Formally, the *precision* of a candidate Anchors explanation $E \subseteq F$ is defined as

$$prec(E) = \mathbb{E}_{s' \sim D(s|E)}[\mathbf{1}_{f(\psi(s))=f(\psi(s'))}], \tag{2.29}$$

where $D(s|E)$ is a discrete data distribution used to sample variations of the input text s, with features in E that remain fixed. In the same way, *coverage* is defined as

$$cov(E) = \mathbb{E}_{s' \sim D(s)}[E(s')], \tag{2.30}$$

$E(s')$ represents the probability of features E appearing when points are randomly sampled from the discrete distribution $D(s)$.

Thus, an Anchors explanation is a subset of the input features $E^* \in F$ such that

$$E^* \in \underset{E \subseteq F}{\operatorname{argmax}} \, cov(E) \;.\; Prob(prec(E) \geq \tau) \geq 1 - \lambda, \tag{2.31}$$

where τ and λ are two positive thresholds that make the extraction of such Anchors feasible, as computing the explanation that maximizes both *precision* and *coverage*

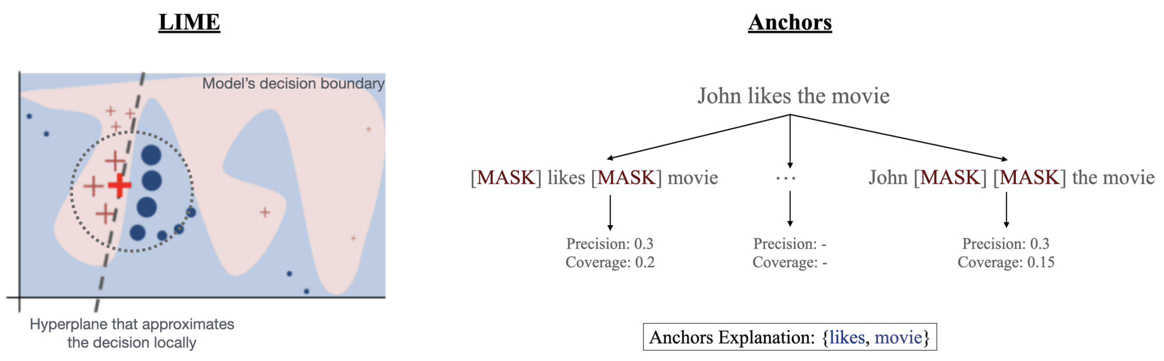

LIME
Model's decision boundary
Hyperplane that approximates the decision locally
Anchors
John likes the movie
[MASK] likes [MASK] movie
· · ·
John [MASK] [MASK] the movie
Precision: 0.3
Coverage: 0.2
Precision: -
Coverage: -
Precision: 0.3
Coverage: 0.15
Anchors Explanation: {likes, movie}

Figure 2.13: Example of LIME (left) and Anchors (right) explainers. LIME linearly approximates the complex decision boundary of a model by repeatedly sampling in the input's neighborhood in the embedding space; then, an explanation is an ordering over the features that influenced the decision the most. The Anchors technique finds, via sampling and masking, the subset of an input text that maximizes coverage while maintaining high precision. Credits to (Ribeiro et al., 2016) and (Ribeiro et al., 2018a).

from a generic distribution $D(s')$ is in general intractable. When Eq. 2.31 is maximized, *precision* and *coverage* produce an explanation with the desirable properties of being representative of the decisions of a model for features E while being succinct, discouraged by a high *coverage*. Figure 2.13 depicts a sketch of how Anchors works.

Chapter 3

Literature Review

In this chapter, we summarize the literature that is relevant to this thesis to provide the more general context of our investigation and contributions. We cover adversarial attacks in NLP, robustness techniques, and explainability. Each macro area is divided into paragraphs, where the literature relative to several closely related subtopics is presented. At the end of each section, we delineate the gaps in the literature that we address in this thesis.

3.1 Adversarial Attacks

NNs are known to be vulnerable to adversarial attacks, i.e., small perturbations of the network input that result in a misclassification (Huang et al., 2011). Seminal works in the literature (Szegedy et al., 2014b; Biggio et al., 2013; Biggio and Roli, 2018) introduced techniques that target Deep Neural Networks, with emphasis on large-scale datasets (Kurakin et al., 2017; Tramèr et al., 2018; Kurakin et al., 2018). DNNs contain millions, when not billions, of parameters, which allow them to approximate very complex input-output relationships: this capacity comes at the cost of a demonstrated vulnerability against slight input variations that possibly induce a model's misclassification. In CV, where DL has achieved unprecedented success in the last decade (Russakovsky et al., 2015; Krizhevsky et al., 2012), a slight variation is often associated with an imperceptible change of a few pixels on the input image, thus making the threat real when deploying applications that deal with unseen, noisy data.

While the main advances were made in CV, these ideas circulated and eventually filtered to NLP, where they have been adapted to the peculiarities of language and symbol representations.

3.1.1 Adversarial Attacks and Language

Similarly to CV, NLP models, which have proven capable of solving linguistic tasks with a high degree of accuracy (Brown et al., 2020), have shown weaknesses soon exploited by malicious adversaries. A recent position paper highlights how robustness in NLP is cogent to develop techniques that we trust, develop and deploy, and it is thus crucial to assessing the risks and opportunities introduced by LLMs (Bommasani et al., 2021). The emergence of Large Language Models as a service, specifically chatbots powered by such models accessible through web interfaces or programming APIs, like ChatGPT (OpenAI, 2023), has not appeared to solve the problem entirely. Recent studies have demonstrated that these models still face difficulties when handling edge cases (Frieder et al., 2023; Kocoń et al., 2023).

Attacks in NLP. As in CV, several works in the literature propose techniques to induce a model misclassification in various contexts just by slightly perturbing an input text or its embedded representation. Attacks are effective when conducted at different levels of granularity: initially guided by heuristic techniques and word substitutions (Alzantot et al., 2018), these techniques evolved to include more sophisticated methods, pipelines, and evaluation schemes (Ettinger et al., 2017; Alishahi et al., 2019). While most of the works model perturbations as word substitutions, some early attempts exist to produce attacks that possibly perturb an entire sentence (Ribeiro et al., 2018b). However, there has yet to be an attempt to formulate a notion of robustness to paraphrases.

A considerable engineering effort has been put recently into developing generative models that augment or test a model against slight, linguistically controlled variations of an input text (Ribeiro et al., 2020; Morris et al., 2020b; Shorten et al., 2021; Kiela et al., 2021). The brittleness of NLP models does not pertain only to text classification but also includes attacks and complementary robustness for a variety of techniques, including ranking systems (Goren et al., 2018) and translation systems (Cheng et al., 2020). LLMs themselves are not immune to attacks, with a growing body of research aimed at studying failures and mitigating techniques for such models (Li et al., 2020), which differ from the attacks which are undertaken solely at the symbols level, and sometimes generated by the same architectures, whose robustness is then tested (Sun et al., 2020).

Robustness to text manipulation is also strictly connected to the capacity of LLMs to contextualize the information they digest and manipulate it accordingly. In (Niven and Kao, 2019), the authors highlight the LLMs' failure to perform simple linguistic reasoning tasks, in which basic models seem to be still pretty limited, especially when it comes to abstracting the usage of parts of the speech with the same flexibility as that of humans (Mahowald et al., 2023). Furthermore, the authors of (Zhang et al., 2020) provide a comprehensive survey of adversarial techniques for NLP and discuss the intrinsic incompatibilities between CV and NLP robustness. In (Roth et al., 2021), the authors provide a taxonomy of NLP adversarial attacks in terms of *token modification*, i.e., constraints on how/what an adversary can manipulate an input text to induce a misclassification. While a few general-purpose surveys on adversarial attacks in NLP have been recently published (Huq et al., 2020; Qiu et al., 2022), other works focus on domain-specific techniques such as social networks, security, etc. (Alsmadi et al., 2021).

State of the art and open problems. Numerous studies implicitly suggest the necessity of linguistically plausible adversarial attacks in natural language processing (NLP) and delve into retrospective frameworks for evaluating robustness (Morris et al., 2020a; Morris, 2020). Other research works establish certain principles for assessing the linguistic plausibility of adversarial attacks (Xu et al., 2020).

Moreover, the current body of literature needs a systematic approach to achieving robustness beyond mere word substitutions. It should encompass phenomena intrinsic to human language, such as sarcasm. In several cases, vulnerabilities to adversarial attacks can be attributed to inconsistencies at the representation level. For instance, words like `'good'` and `'nice'` may possess different representations in the embedding space, which can be detrimental to the robustness of sentiment analysis tasks.

An illustrative example of embedded human biases derived from data is examined in (Bolukbasi et al., 2016), where the authors demonstrate that context-free embeddings exhibit gender bias. These issues persist in language models, including LLMs, which have showcased impressive performance across various tasks, leading some to claim they have mastered language (Johnson and Iziev, 2022). However, an alternative perspective argues that LLMs lack natural language understanding (Bender and Koller, 2020).

3.2 Robustness and Verification

Formal verification in software refers to a rigorous and mathematical approach to verifying the correctness of software systems or components. By extension, in neural networks, it refers to ensuring the reliability and robustness of such models. In this sense, formal verification can guarantee that an NN classification (and, more generally, any output) is invariant to a specific class of perturbations, e.g., those localized in the neighborhood of an input point. Both empirical and theoretical works have demonstrated that verification is very challenging due to the intrinsic complexity of NNs, and the variety of DL architectures. In the following sections, we review verification methods for NNs, distinguishing between those that provide formal vs. approximate guarantees and with particular emphasis on techniques developed for the NLP setting.

Exact and approximate NLP robustness verification. One standard way to dissect verification tools for NLP is in light of the formal guarantees they provide. On the one hand, some tools provide formal guarantees on a model's prediction at the expense of an increased computational burden. On the other side of the spectrum, approximate and heuristics-guided methods relax the robustness conditions to output probabilistic or estimated guarantees.

A standard approach to formal verification of NNs consists of turning the model's decision into a constraint satisfiability problem, for which fast and reliable solvers exist (Katz et al., 2017c, 2019b). These methods work with ReLU FC networks, and while methods for CNNs exist (Wang et al., 2018b), they have not been adapted to the NLP setting (i.e., to work with embeddings). Further, mixed integer programming (Dutta et al., 2018; Cheng et al., 2017) can provide complete robustness guarantees by computing exact bounds on an NN's input-output relationship under adversarial manipulations. Both satisfiability and mixed-integer programming are computationally expensive techniques that hardly scale to real-world networks because the problem of deducing exact guarantees for an NN is NP-hard.

Incomplete or approximate approaches set a trade-off between the computational complexity of providing formal guarantees, with the necessity to make such methods scale beyond toy-example NNs.

Recently, linear constraint relaxation methods (Weng et al., 2018a; Zhang et al., 2018; Wong and Kolter, 2018), which compute for each neuron in an NN an upper and a lower bound value on its activation, have been developed to leverage the peculiarities of each architecture and achieve a trade-off between formal guarantees and scalability. In this sense, methods have been successfully developed for CNNs (Boopathy et al., 2019), RNNs (Ko et al., 2019b), and more recently for transformers (Shi et al., 2020b), which can be used to compute robustness guarantees for text classification in the NLP domain. We finally note that NLP robustness has also been addressed using interval bound propagation (IBP) (Gowal et al., 2018), a method that trains NNs provably robust against attacks w.r.t. ℓ_p-bounded perturbations (Huang et al., 2019; Jia et al., 2019). A peculiarity of IBP is the possibility of balancing the accuracy of the downstream task with a certificate of formal robustness guarantees against local attacks.

Search-based and reachability computation methods (Huang et al., 2017; Ruan et al., 2018; Wu and Kwiatkowska, 2020) output looser robustness bounds with much greater scalability. In the same way, the maximal safe radius (Wu et al., 2020b) or, dually, minimum adversarial distortion (Weng et al., 2018b) can provide sound robustness guarantees, albeit relying on the knowledge of the Lipschitz constants (local or global) of a network. Unfortunately, estimating such quantity is itself known to be an NP-hard problem, particularly for architectures popular in NLP (Kim et al., 2021). As regards attention, a few works attempted to improve robustness for this specific architecture (Zoran et al., 2020; Hong et al., 2021), yet the main contributions remain in the CV domain, though a few attempts to translate them to NLP exist (Hsieh et al., 2019).

State of the art and open challenges. Many works provide surveys on robustness techniques in NLP (Alshemali and Kalita, 2020; Goyal et al., 2022), with some systematically reviewing the various dimensions of robustness across techniques, metrics, embeddings, and benchmarks (Omar et al., 2022), while only relatively few emphasizing how to enhance the robustness of NLP models at training time (Wang et al., 2022). Researchers in this field are also concerned with domain-specific defensive strategies, with works that summarize the effort by the community in that regard in the recent years (Apostolidis and Papakostas, 2021; Tocchetti et al., 2022).

On the other hand, the NLP literature lacks a systematic approach to characterizing a model's region of maximal safety, similar to what has been done for image and video classification Wu et al. (2020a); Wu and Kwiatkowska (2020).

In the domain of approximate approaches for assessing robustness in NLP, there remains an unexplored avenue concerning the measurement of robustness against linguistic phenomena, including mixed sentiment and negation. Taking inspiration from the post-structuralist paradigm of linguistics and recent advancements in the literature that demonstrate the encoding of syntactic structures within hidden representations of LMs (Manning et al., 2020), a yet-unexplored direction emerges, namely *syntactic robustness*. This refers to the ability of embeddings and LMs to retain information about the structural aspects of a sentence in a resilient manner, impervious to adversarial manipulations.

3.3 Explainability

Explainability and interpretability of machine learning models are receiving increasing attention from the research community (Chakraborty et al., 2017), as NN-based solutions are now deployed and available as-a-product to a vast number of devices, from smartphones to self-driving vehicles. Existing explainability methods broadly fall into four categories: local vs. global explanations and post-hoc vs. self-explaining methods (Danilevsky et al., 2020). Another taxonomy, which is more relevant for this work of thesis, organizes explainability methods in *feature attribution* and *feature selection* methods (Molnar, 2020; Samek et al., 2019). The former ranks features of a classifier assigning an importance score to each of them, while the latter identifies features relevant to the prediction.

Explainers. Generally speaking, methods tailored to (locally) explaining NLP model decisions for a given input include (Li et al., 2016; Singh et al., 2019). These identify input features, or clusters of input features, that most contribute to the prediction, using saliency and agglomerative contextual decomposition, respectively. Layer-wise relevance propagation (Bach et al., 2016) is also popular for NLP explanations and is used in many works (Arras et al., 2016, 2017; Ding et al., 2017) due to the relatively low computational complexity of such approaches, which usually reduces to one or more forward/backward calculations. Similarly to the above, these methods do not consider robustness.

A prominent example of a local, post-hoc explainer, which is also a *feature attribution* method, is LIME (Ribeiro et al., 2016): LIME learns a linear model around the neighborhood of input using sampled local perturbations randomly: despite being useful in practice, LIME lacks robustness guarantees and may attribute importance to features that do not enable robustness, or, equivalently, can be slightly manipulated to induce a model misclassification or a drastic change in the explanation (Slack et al., 2020).

Anchors (Ribeiro et al., 2018a), similarly to LIME, is a local, post-hoc explainer, yet it splits the input into two sets. It is thus a *feature selection* method: a set of features that maximizes precision and coverage, respectively, as the number of times the explanation implies the model's decision. It appears in the training set and the left-out features.

Formal and guaranteed explainability. There has been increasing interest in explainers whose principles are axiomatized to enhance the interpretability of the results: SHAP (Lundberg and Lee, 2017), which takes inspiration from the Shapley value in game-theory literature, attributes importance to a subset of the input features as the average of all the marginal contributions to all possible set of features (i.e., coalitions). SHAP, alongside many methods developed in the wake of its success (Strumbelj and Kononenko, 2010; Covert et al., 2020; Burkart and Huber, 2021), enhances the interpretability of thus obtained explanations. Nevertheless, these methods have yielded provably misleading information about the relative importance of features for predictions (Huang and Marques-Silva, 2023).

A considerable body of recent literature proposes formal explainability methods based on *abductive explanations* (Marques-Silva and Ignatiev, 2022; Marques-Silva, 2022), where the explainability process and the outcome follow that of logic abduction (Ignatiev et al., 2019d,b), yet solutions that scale to neural networks do only work with toy-examples such as binarized neural architectures (Shi et al., 2020a; Darwiche and Hirth, 2020; Darwiche, 2020), and with no works in the area of NLP.

Researchers have also devoted their attention to repairing non-formal explainers (such as LIME) to make their explanations robust. In (Ignatiev et al., 2019a), the authors repair non-robust explanations via formal methods based on logic abduction, yet their technique works solely for boosted tree predictors and does not scale to real-word NNs, not to mention NLP. Regarding Anchors, in (Narodytska et al., 2019), the authors assess the quality of Anchors' explanations by encoding the model and explanation as a propositional formula. The explanation quality is then determined using model counting but for binarized neural networks only.

State of the art and open challenges. The current literature reveals a notable void concerning the existence of works that tackle the rigorous verification of neural network explainability tools: existing works can only handle simple architectures such as binarized neural networks (Narodytska et al., 2019). On the other hand, popular explainers such as LIME and Anchors (Ribeiro et al., 2016, 2018a), which work well in practice and can be adapted to the NLP scenario, do not provide any guarantees regarding the explanations or the model. This gap can be attributed to the inherent complexity and challenges associated with this setting. The intricate nature of neural networks, coupled with the need to ensure the reliability and accuracy of their interpretability methods, makes the development and evaluation of robust verification techniques an arduous task: a few methods developed efficient tools to verify the robustness of neural networks to adversarial perturbations (Wang et al., 2018a; Katz et al., 2019b), yet the applicability of such tools to explainability is still an unexplored field. As a result, the literature needs more extensive research efforts to address this pressing issue, necessitating further exploration and advancements in the field to bridge this gap and promote the trustworthy application of neural network explainability tools.

Chapter 4

Measuring Robustness in NLP

In this chapter, we consider the problem of quantifying the robustness of an NN model trained to solve an NLP classification task. Following Chapter 2, we show how to define the problem of computing the region of maximal safety for a model when perturbed via embedding/representation-based attacks. We also comment on the hardness of computing such guarantees when a model or a representation scales in size. This chapter proposes a method to find the region of maximal safety of different NLP models, which can be equipped with various embeddings to solve multi-class classification tasks. At the same time, this chapter emphasizes the critical issues of making the exact robustness method scale to novel and larger architectures and the limitations of the *de facto* standard adversarial robustness setting in NLP.

In short, these are the contributions of the chapter:

- We propose a comprehensive framework for quantifying the robustness of NLP models, utilizing a certified lower bound and an upper bound to assess their robustness against perturbations. The certified lower bound represents the region of maximal safety by considering ℓ_2 and ℓ_∞ norms, while the upper bound ensures the generation of meaningful sentences within the representation space. We adopt state-of-the-art methods to compute the certified lower bound, namely CNN-Cert and POPQORN. Additionally, we employ a Monte Carlo Tree Search method (MCTS) to generate an adversarial attack that satisfies two criteria: (i) the attack corresponds to a sentence in the word space, and (ii) it minimizes the distance from the original input, ultimately converging to the certified lower bound.

- We conduct an experimental evaluation on CNN and LSTM sentiment and news classification models on a range of embeddings and datasets, with the scope to test the utility of such robustness measures in practice. We prove that certificates of robustness are applicable in practice for small models, i.e., those equipped with low-dimensional embeddings. On the other hand, such guarantees vanish with models that grow in the number of parameters and embedding dimensions.

This chapter's contributions first appeared as (La Malfa et al., 2020).

4.1 Motivation and Setting

Given a text classification task, where a model is required to learn an input-output mapping between pairs of l-long short texts drawn from the vocabulary V and labels (s, y), namely $f : S \subseteq V^l \to Y$, one wants to certify the local ℓ_p robustness of the embedding of s, namely $x = \psi(s)$. While one can provide probabilistic guarantees for the robustness of an NN, such guarantees cannot prove a model immune to adversarial attacks. Only formal guarantees leave no uncertainty on the existence of local adversarial threats, though they incur much higher computational costs. The technique presented in this chapter characterizes an NN's region of maximal safety against attacks applied to an input's continuous embedding representation. Furthermore, our method quantifies the gap between such a certificate of robustness and the closest discrete adversarial attack with an approach that has three significant advantages: (i) it characterizes the local decision boundary of a model; (ii) it provides a sound certificate of robustness; (iii) it allows for comparison of different models. While we focus on FCs, CNNs and recurrent networks, we notice that providing formal guarantees for Transformer-based models Vaswani et al. (2017) is a complex challenge due to their scale and intricacy. These models' dynamic nature and the need for a solid theoretical framework make it hard to establish reliable, scalable guarantees. My thesis acknowledges this limitation, emphasizing that while the framework presented is potentially applicable to Transformers, it depends on future developments in scalable guarantee techniques.
We begin by formally explaining the technicalities of our approach.

4.2 The Maximal Safe Radius Approach

Given an embedding/representation $\psi : S \to X$, an NN architecture that embodies such a representation, i.e., a classification task in the form of an input-output relationship (X, Y) that the NN learns, namely $f : X \to Y$. Given an input text $s \in S$, its representation $x = \psi(s)$, the corresponding output label y, and considering an ϵ-Ball w.r.t. an ℓ_p norm, we recall that the region of maximal safety of f to perturbations to x can be expressed as

$$\epsilon^* \in \operatorname*{argmax}_{\epsilon} Ball(x,\epsilon) \text{ s.t. } \forall x' \in Ball(x,\epsilon),\ f(x) = f(x'). \tag{4.1}$$

Since the MSR problem is NP-hard (Katz et al., 2017a), one can only approximate such a safety region with techniques that incur exponential computational time cost (unless P=NP).

The discrepancy between symbols (discrete words) and the continuous representation space in NLP limits the interpretability of robustness certificates. There is no guarantee that a region in the representation space contains vectors corresponding to discrete words, except for the input point. This challenge hinders our ability to understand and interpret the behavior of NLP models in terms of robustness certification.

To enhance the characterization of a model's robustness, we augment it with an upper bound represented by the closest discrete adversarial attack to the input x. Formally, one looks for

$$s^* \in \operatorname*{argmin}_{s' \in DBall(s,\mathbf{d})} ||\psi(s) - \psi(s')||_p \ . \ f(\psi(s)) \neq f(\psi(s')). \tag{4.2}$$

In the previous formula, $DBall(s,\mathbf{d})$ (Eq. 2.22) contains all the possible discrete combinations of l symbols, as $\mathbf{d} = max_{(x,x')\sim\mathbb{D}(X)}||x-x'||_p$ is the diameter of the embedding/representation space w.r.t. an ℓ_p norm of choice. The formulation in Eq. 4.1 and 4.2 was initially proposed for generic NNs and continuous representations in (Wu et al., 2020a), and called the Maximal Safe Radius (MSR), where the 'radius' is defined as the line segment extending from the center of a sphere, i.e., the input point x, to the farthest points on the bounding surface identified by the ϵ-Ball (Eq. 2.14). An illustrative example of the MSR scenario adapted to an NLP classification task is reported in Figure 4.1.

Finally, to better judge the robustness of a model over a number of input points, it is useful to introduce an average notion of robustness. For the upper and the lower bound formulae introduced in Eq. 4.1 and 4.2, a dataset of input sentences S and its embedded representation X, a model f equipped with a representation ψ, the average robustness of f can be expressed as a tuple

$$\begin{aligned}
&(r_{lb}, r_{ub})\ s.t. \\
&r_{lb} = \tfrac{1}{n}\textstyle\sum_{x\in X} \max Ball(x,\epsilon) \text{ s.t. } \forall x' \in Ball(x,\epsilon),\ f(x) = f(x') \\
&r_{ub} = \tfrac{1}{n}\textstyle\sum_{s\in S} \min ||\psi(s) - \psi(s')||_p \text{ s.t. } \forall s' \in DBall(s,\mathbf{d}),\ f(\psi(s)) \neq f(\psi(s')).
\end{aligned} \tag{4.3}$$

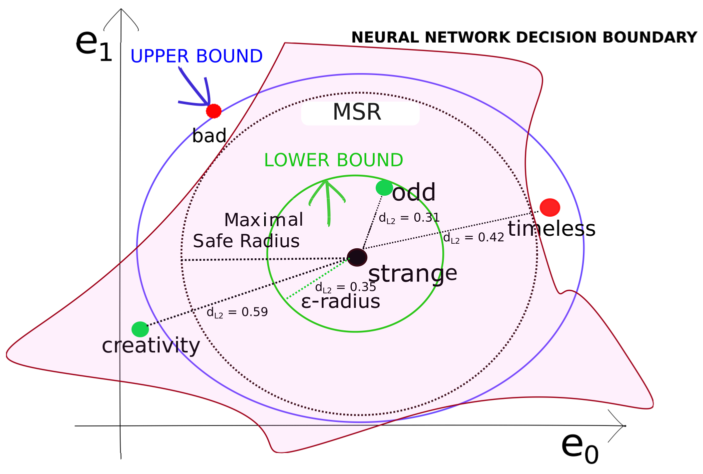


Figure 4.1: Maximal Safe Radius (MSR) and its upper and lower bounds. An upper bound of MSR is obtained by computing the distance of any discrete perturbation resulting in a class change (blue ellipse) to the input text. A lower bound certifies that perturbations of the words contained within that radius are guaranteed not to change the classification decision (green ellipse). Both upper and lower bounds approximate the MSR (black ellipse). In this example, the word `strange` can be safely substituted with `odd`. The word `timeless` is within the upper and lower bound of the MSR, so our approach cannot guarantee it would not change the neural network prediction.

With the MSR problem for NLP that is now mathematically formalized, we proceed by illustrating how to compute a value for Eq. 4.1 and a value to which an input text corresponds, as per Eq. 4.2.

4.2.1 Lower Bound

A lower bound for the MSR, and thus a solution to Eq. 4.1, is an optimal value $\underline{\epsilon} > 0$ such that all text representations in $Ball(\psi(s), \underline{\epsilon})$ are classified in the same class by the NN. Intuitively, the region we are interested in computing is sketched in green in Figure 4.1 and provides guarantees on perturbing each word that can be the target of a local attack. For example, if one is interested in perturbing 2 words in a text, the MSR will derive an upper/lower bound that quantifies the region of maximal safety of each word, with an attack that belongs to the union of the neighbourhood of each word in the pair.

To compute $\underline{\epsilon}$, we leverage constraint relaxation techniques developed for CNNs (Boopathy et al., 2019) and LSTMs (Ko et al., 2019b), namely CNN-Cert and POPQORN. For an input text s and a hyper-box around $Ball(\psi(s), \epsilon)$, these techniques linearly approximate the lower and upper bounds for the activation functions of each layer of the neural network and use these to propagate an over-approximation of the hyper-box through the network. $\underline{\epsilon}$ is then computed as the smallest real number such that all the texts in $Ball(\psi(s), \underline{\epsilon})$ are in the same class, i.e., $\forall x' \in Ball(\psi(s), \underline{\epsilon})$, $f(x) = f(x')$. Technically speaking, CNN-Cert mathematically defines closed-form upper and lower linear constraints that bound the activation function of a convolutional layer in the form

$$\{(A_{lb}, B_{lb}), (A_{ub}, B_{ub})\}\ .\ A_{ub} z^{[\ell]} + B_{ub} \geq f^{[\ell]}(z^{[\ell]} * W^{[\ell]} + b^{[\ell]}) \geq A_{lb} z^{[\ell]} + B_{lb}, \quad (4.4)$$

where $f^{[\ell]}(z^{[\ell]} * W^{[\ell]} + b^{[\ell]})$ represents the convolution, denoted by the operator $*$, defined at layer ℓ. Specifically, the tuples $(A_{lb}, B_{lb}), (A_{ub}, B_{ub})$ represent fixed tensors associated with the weights, biases, and activation function employed by the neural network at layer ℓ.

The closed-form expression to compute A_{ub}, for a 3D input x at coordinates (a, b, c) and a generic tensor of weights W at coordinates (i, j, k), is the following:

$$A_{ub,(a,b,c),(i,j,k)} = W^{+}_{(a,b,c),(i,j,k)} \alpha_{ub,(a+i,b+j,k)} + W^{-}_{(a,b,c),(i,j,k)} \alpha_{lb,(a+i,b+j,k)}, \quad (4.5)$$

where $\alpha_{ub,(a+i,b+j,k)}$ is the corresponding parameter of a linear upper bound for the activation of the neuron at coordinates (a, b, c), while W^{+} and W^{-} contain only the positive, negative entries of W, with other entries equal 0. In the same way, the closed-form expression to compute B_{ub}, for a 3D input x at coordinates (a, b, c) and a generic tensor of weights W at coordinates (i, j, k), is the following:

$$B_{ub} = W^{+} * (\alpha_{ub} \odot \beta_{ub}) + W^{-} * (\alpha_{lb} \odot \beta_{lb}) + b, \quad (4.6)$$

where β_{ub} is the corresponding parameter of a linear upper bound of the activation of a neuron at a generic layer ℓ, while $\odot$ represents the element-wise product between tensors of the same shape. Notice that in Equations 4.5 and 4.6, one can obtain the corresponding values for the lower bounds by substituting α_{ub} and β_{ub} with the corresponding values of α_{lb} and β_{lb}.

A detailed derivation of each matrix tensor, which surpasses the scope of this section, can be referenced from the original research paper by Boopathy et al. (Boopathy et al., 2019), specifically in Table 2, and in section (a) of the Appendix included in the publication mentioned above.

Similarly, the POPQORN technique imposes linear constraints to bound individual neurons within an LSTM cell. An additional challenge LSTM cells pose is the non-linear nature of their activations, specifically the softmax and tanh functions employed in the vanilla version of an LSTM, as well as the recursive equation computed for each temporized input point. However, the underlying principle remains consistent with Equation 4.4, wherein the upper and lower bounds calculated at each stage solely rely on the previous stage's weights, biases, and activation. A comprehensive derivation of the bounds for each component of an LSTM cell can be found in the original paper by Ko et al. (Ko et al., 2019a), specifically in Table 4 and Section 3.2 of said publication.

Both CNN-Cert and POPQORN find a suitable solution for the problem formulated in Equation 4.1, and for which more details have been provided in Chapter 2, Eq. 2.19 and 2.20. As the MSR is defined in the *continuous* embedding space of an NN representation, the perturbation space, $Ball(\psi(s), \underline{\epsilon})$, contains meaningful texts, non-meaningful texts, and points that do not have a pre-image in the vocabulary space. Thus, a discrete characterization of the upper bound is a valuable complement to the robustness analysis of a model.

4.2.2 Mind the (Discrete) Gap: the Upper Bound

With the lower bound that guarantees for an input text, that a model is robust to any perturbation within an ϵ-neighborhood, it is an interesting research question to identify the closest discrete perturbation that induces a misclassification (see Fig. 4.1). An upper bound for the MSR, which constitutes a (possibly sub-optimal) solution to Eq. 4.2, is, in this sense, a perturbation of the input text s that is classified by the NN differently than the original text.

We adapt the Monte Carlo Tree Search (MCTS) algorithm (Coulom, 2007) to the NLP robustness scenario: MCTS is a heuristic search algorithm most notably employed in discrete scenarios such as board games. It finds a suitable solution by solving a problem whose search space is represented by a tree. In Figure 4.2, we illustrate the MCTS procedure for a text (left), and an example of a filtering strategy on substitutions (right). The algorithm takes as input an l-word long input text $s = \{w_1, .., w_l\}$ and builds a tree, where at each iteration, a set of indices I identifies the words that have been modified so far. At the first level of the tree, a single word is changed to manipulate the classification outcome. At the second, two words are perturbed, with the former being the same word as for the parent vertex, and so on (i.e., for each vertex, I contains the indices of the words that have been perturbed plus that of the current vertex). Perturbations are sampled by considering a fixed number of closest replacements in the word's neighborhood: the distance between words is measured in the ℓ_2 norm, while the number of substitutions per word is limited to a fixed constant (e.g., in our experiments this is either 1000 or 10000).

At each stage, the procedure outputs all the successful attacks (i.e., perturbed texts that are classified by the neural network differently from the original text) that have been found until the terminating condition is satisfied (e.g., a fixed fraction out of the total number of vertices has been explored). Successful perturbations can be used as diagnostic information in cases where ground truth information is available. The algorithm explores the tree according to the UCT heuristic (Browne et al., 2012), where *urgent* vertices are identified by the perturbations that induce the most significant drop in the neural network's confidence. A detailed description of the heuristic, and the algorithm, which follows the classical algorithm (Coulom, 2007) while working directly with word embeddings, is reported in the next section.

4.2.3 MCTS Algorithm

We adapt the MCTS algorithm (Browne et al., 2012) to the NLP classification setting with word embedding, which we report here for completeness as Algorithm 1. The algorithm explores modifications to the original text by substituting one word at a time with nearest-neighbor alternatives. It takes as input: *text*, expressed as a list of $l > 0$ words; f, an NN f; ψ, an embedding (linguistic representation); *sims*, an integer specifying the number of Monte Carlo samplings at each step; and α, a real-valued meta-parameter specifying the exploration/exploitation trade-off for vertices that can be further expanded. The salient steps of the MCTS procedure are:

- **Select**: the most *promising* vertex to explore is chosen to be expanded (Line 15) according to the standard UCT heuristic:
  $\frac{Q(v)}{N(v)}+\alpha\sqrt{\frac{2lnN(v')}{N(v)}}$, where v and v' are respectively the selected vertex and its parent; α is a meta-parameter that balances exploration-exploitation trade-off; $N()$ represents the number of times a vertex has been visited, and $Q()$ measures the neural network confidence drop, averaged over the Monte Carlo simulations for that specific vertex.

- **Expand**: the tree is expanded with T new vertices, one for each word in the input text (avoiding repetitions). A vertex at index $t \in \{1, ...l\}$ and depth $n > 0$ represents the strategy of perturbing the t-th input word, plus all the words whose indices have been stored in the parents of the vertex itself, up to the root.

- **Simulate**: simulations are run from the current position in the tree to estimate how the neural network behaves against the perturbations sampled at that stage (Line 27). Suppose one of the word substitutions induced by the simulation makes the network change the classification. In that case, a successful substitution is found and added to the results, while the value Q of the current vertex is updated. Many heuristics can be considered at this stage, for example, the average drop in the network's confidence over all the simulations. We have found that the average drop is not a good measure of how the robustness of the network drops when some specific words are replaced since, for a high number of simulations, an effective perturbation might pass unnoticed. We thus work with the maximum drop over all the simulations, which works slightly better in this scenario (Line 33).

- **Backpropagate**: the reward received is back-propagated to the vertices visited during selection and expansion to update their UCT statistics. When UCT is employed (Browne et al., 2012; Kocsis and Szepesvári, 2006), MCTS guarantees that the probability of selecting a sub-optimal perturbation tends to zero at a polynomial rate when the number of games/branches explored grows to infinity (i.e., it guarantees to find a discrete perturbation, if it exists).

Finally, and to enforce the syntactic consistency of the replacements, we consider part-of-speech tagging of each word based on its context. Then, we filter all the replacements found by MCTS to exclude those that are not of the same type, i.e., those that do not preserve the Part-of-the-Speech (POS) tag: POS-tags can be nouns, verbs, adverbs, etc., and the objective is to perform a substitution that is syntactically equivalent to that of the original word. To accomplish this task, we use the Natural Language Toolkit (Bird et al., 2009), which provides a rudimentary, albeit consistent in most cases, method to replace a word that has the equivalent POS-tag.

4.3 Experiments

In this section, we aim to assess the lower bounds of the MSR for CNNs and LSTM models. To achieve this, we utilize the CNN-Cert and POPQORN tools. Additionally, we employ the Monte Carlo Tree Search (MCTS) algorithm, as introduced in the previous section, to explore meaningful perturbations (i.e., upper bounds) irrespective of the neural network architecture used.

Our investigation evaluates models' robustness against single and multiple-word substitutions, with a maximum of 5 substitutions. We also delve into analyzing the

Algorithm 1 Monte Carlo Tree Search with UCT heuristic

```
 1: procedure MCTS($s,\ f,\ \psi,\ sims,\ \alpha$)
 2:     $Tree \leftarrow createTree(s,\ f)$                          ▷ Create the initial tree
 3:     $root \leftarrow getRoot(Tree)$                              ▷ Store the initial vertex
 4:     $P \leftarrow [\,]$                                          ▷ List of final perturbations
 5:     while $terminate(Tree) \neq True$ do                         ▷ Loop over the MCTS steps
 6:         $v \leftarrow$SELECT($Tree,\ \alpha$)
 7:         $C \leftarrow$EXPAND($v,\ s$)
 8:         $P.insert$(SIMULATE($C,\ s,\ sims,\ f,\ \psi$))
 9:         BACKPROPAGATE($v, root$)
10:     end while
11:     return $P$
12: end procedure

13: procedure SELECT($Tree,\ \alpha$)
14:     $L \leftarrow getLeaves(Tree)$
15:     return $\text{argmax}_{v \in L} \dfrac{Q(v)}{N(v)} + \alpha\sqrt{\dfrac{2lnN(v')}{N(v)}}$     ▷ UCT best leaf
16: end procedure

17: procedure EXPAND($v,\ s$)
18:     for $i = 0,\ i < length(s),\ i{+}{+}$ do
19:         $v.expand(i)$                                            ▷ Create v's i-th child
20:     end for
21:     return $getChildren(v)$                                      ▷ Return the expanded children
22: end procedure

23: procedure SIMULATE($C,\ s,\ sims,\ f,\ \psi$)
24:     $Perturbations \leftarrow [\,]$
25:     for $c \in C$ do
26:         for $i = 0,\ i < sims,\ i{+}{+}$ do
27:             $s' \leftarrow samplePerturbation(s,\ c)$            ▷ Ref. Figure 4.2
28:             $x \leftarrow \psi(s);\ x'_i \leftarrow \psi(s')$    ▷ Embed inputs
29:             if $f(x'_i) \neq f(x)$ then                          ▷ The output class changes
30:                 $Perturbations.append(s')$
31:             end if
32:         end for
33:         $Q(c) = max_{i \in sims}(f^{[\ell-1]}(x) - f_t^{[\ell-1]}(x'_i))$   ▷ Update vertex heuristic
34:     end for
35:     return $Perturbations$
36: end procedure
```

Algorithm 2 Monte Carlo Tree Search with UCT heuristic (Continuation)

```
1: procedure BACKPROPAGATE($v$, $root$)                ▷ Propagate UCT update
2:     while $v \neq root$ do
3:         $updateUCT(v)$
4:         $v \leftarrow getParent(v)$
5:     end while
6: end procedure
```

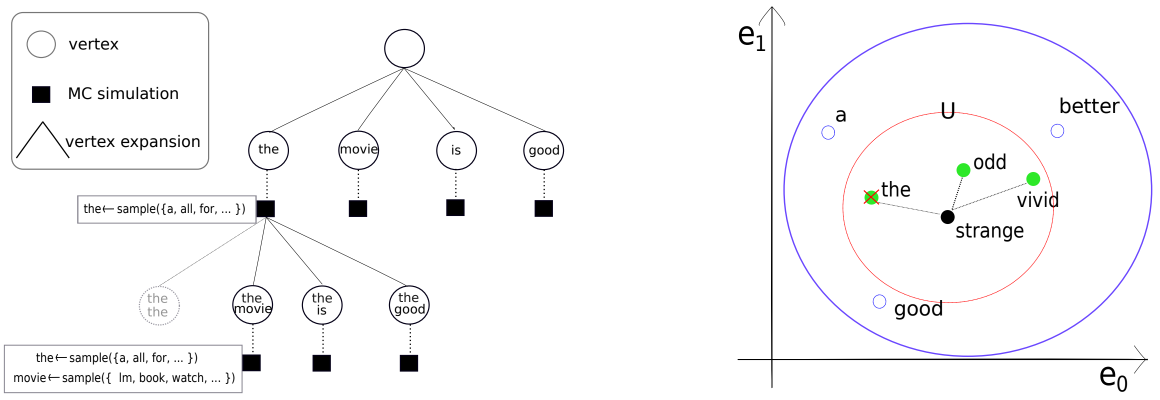


Figure 4.2: On the left, the tree's structure after two iterations of the MCTS algorithm. Simulations of 1-word substitutions are executed at each vertex on the first level to update the UCT statistics. The most urgent vertex is then expanded (e.g., word `the`), and several 2-words substitutions are executed combining the word identified by the current vertex (e.g., word `movie` at the second level of the tree) and that of its parent, i.e., `the`. Redundant substitutions may be avoided (greyed-out branch). On the right, MCTS selects substitutions randomly or according to a score calculated as a function of the distance from the original word (see Algorithm 1 for details). The sampling region (red circle) is a finite fraction of the embedding space (blue circle). Selected candidates can be filtered to enforce semantic and syntactic constraints. Word `the` has been filtered out because it is not grammatically consistent with the original word `strange`, while words `good`, `better` and `a` are filtered out as they lie outside the neighborhood of the original word.

implicit biases in CNN and LSTM architectures. Lastly, we investigate the impact of different embedding types and sizes on the robustness of the models.

These analyses aim to provide insights into the lower bounds of MSR for CNNs and LSTMs, explore the effects of perturbations, investigate implicit biases, and examine the influence of embedding characteristics on model robustness.

4.3.1 Experimental Setup

Multi-class classification tasks constitute our test bed, thus including sentiment analysis. We evaluate more than 20 vanilla CNN and LSTM models trained on datasets that differ in the length of each input, the number of target classes, and the difficulty of the learning task. All our experiments were conducted on a server with two 24 core Intel Xenon 6252 processors and 256GB of RAM. Although the experiments reported here have been performed on a cluster, all the algorithms are reproducible on a mid-end laptop; we used a machine with 16GB of RAM and an Intel-5 8th-gen. processor. For our experiments, we consider a 3-layer CNN, where the first layer consists of bi-dimensional convolution with 150 filters, each of size 3×3, and an LSTM model with 256 hidden neurons on each gate. We note that, though other architectures might offer higher accuracy for sentence classification (Kim, 2014), this *vanilla* setup has been chosen intentionally not to be optimized for a specific task, thus allowing us to measure the robustness of baseline models. Both CNNs and LSTMs predict the output with a softmax output layer, while the Categorical Cross-entropy loss function (see Eq. 2.7) is used during the optimization phase, which employs Adam (Kingma and Ba, 2015) algorithm (without early-stopping).

We consider the IMDB (Maas et al., 2011b) and the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013b), two standard sentiment analysis tasks which differ in the number of test samples, the length of each input text, and the quality of the data, with the former collected from movie reviews. At the same time, human experts have handcrafted the latter. We further conduct experiments on the AG News Corpus (Zhang et al., 2015b) and the NEWS dataset (Vitale et al., 2012), two multi-class classification datasets.

The datasets mentioned in the paragraph above are relevant to the problem studied as they provide the necessary resources for conducting sentiment analysis and multi-class classification experiments. The IMDB dataset contains movie reviews, making it valuable for sentiment analysis research. The Stanford Sentiment Treebank (SST) dataset offers fine-grained sentiment labels and allows for studying the hierarchical structure of sentiment within sentences. The AG News Corpus and NEWS

	NEWS	**SST**	**AG NEWS**	**IMDB**
Inputs (Train, Test)	$22806, 9793$	$117220, 1821$	$120000, 7000$	$25000, 25000$
Output Classes	7	2	4	2
Average Input Length	17 ± 2.17	17.058 ± 8.27	37.295 ± 9.943	230.8 ± 169.16
Max Input Length	88	52	136	2315
Max Length Considered	14	25	49	100

Table 4.1: Datasets used for the experimental evaluation. We report the number of samples (training/test ratio as provided in the original works) and output classes, the average and maximum length of each input text before pre-processing, and the maximum length considered in our experiments.

datasets are multi-class classification datasets that can be used for text classification tasks, such as news categorization and topic identification. These datasets contribute to the paper by providing diverse data sources and allowing researchers to explore various aspects of sentiment analysis and multi-class classification. The details of each dataset are provided in Table 4.1 for transparency and reproducibility.

As regards the embeddings used to 'equip' each NN model, we balance probabilistically-constrained representations from GloVe and GloVeTwitter (Pennington et al., 2014a), which are trained on global word-word co-occurrence statistics from a corpus, with embeddings provided by the Keras Python Deep Learning Library (referred to as Keras Custom) (Chollet et al., 2015), which allows one to fine-tune the exact dimension of the vector space and only aims at minimizing the loss on the classification task. The resulting learned Keras Custom embeddings do not capture complete word semantics, just their emotional polarity, unlike GloVe, whose words encode each statistic on local and global word co-occurrences. In general, we refer to the number of dimensions of a model by reporting it after its name; e.g., GloVe-100d refers to GloVe embedding, which represents each word as a 100-dimensional floating-point vector.

Finally, we conduct our experiments alternating upper and lower bounds measurements using the ℓ_2 and ℓ_∞ norms. This choice is motivated by two distinct reasons. Firstly, from a technological standpoint, both CNN-Cert and POPQORN are limited in their capability to perform experiments with specific ℓ_p norms. Consequently, they do not encompass distance measures such as cosine similarity. Secondly, the ℓ_∞ norm incorporates the dimension of maximum variation in an input, in contrast to the ℓ_2 norm, which corresponds to the Euclidean distance between an input and its perturbations. Further, to enhance the comparison of embeddings with different dimensions, both the training and the certification phases are conducted on embedding normalised to have the diameter equal to one.

4.3.2 Robustness to Word Substitutions

Interestingly, our framework can prove that certain input texts and architectures are robust for any single-word substitution, that is, replacing a single word of the text (any word) with any other possible other words, and not necessarily with a synonym or a grammatically correct word, will not affect the classification outcome. Figure 4.3 shows that for CNN models equipped with Keras Custom embedding the lower bound of the MSR on some texts from the IMDB dataset is greater than the diameter of the embedding space.

In Table 4.2, models that employ embeddings with low-dimensional representations (up to 25 dimensions) are more robust to perturbations. On the contrary, average robustness decreases, from one to two orders of magnitude, when words are mapped to high-dimensional spaces, a trend that we will show is also confirmed by the upper bound results. This may be explained by the fact that adversarial perturbations are inherently related to the dimensionality of the input space (Goodfellow et al., 2015), and in general to the fact that high-dimensional models offer a wide surface of attack that makes it hard to escape the presence of adversarial attacks. This hypothesis has a complementary reformulation that is supported by extensive empirical validation. It suggests a trade-off between a model's robustness and performance, as evidenced by a few influential works in literature (Madry et al., 2018; Deng and Tian, 2020).

	DIMENSION	**LOWER BOUND**
Keras	5	0.278
	10	0.141
	25	0.023
	50	0.004
	100	0.002
GloVe	50	0.007
	100	0.002
GloVeTwitter	25	0.013
	50	0.008
	100	0.0

Table 4.2: Comparison of lower bounds for single-word substitutions computed by CNN-Cert on the SST dataset. In order to compare different embeddings, values of the lower bounds are averaged over 100 input texts (approx. 2500 measurements) and normalized by the embedding diameter (ℓ_2-norm). While we keep the same architecture for all the experiments, i.e., a 2-layers, ReLU-activated CNN network, the number of parameters of each model varies accordingly to the dimensionality of the embedding.

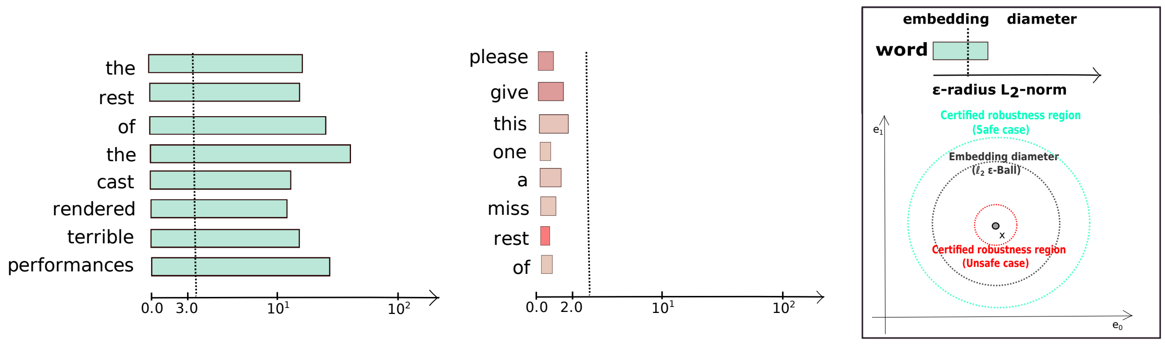


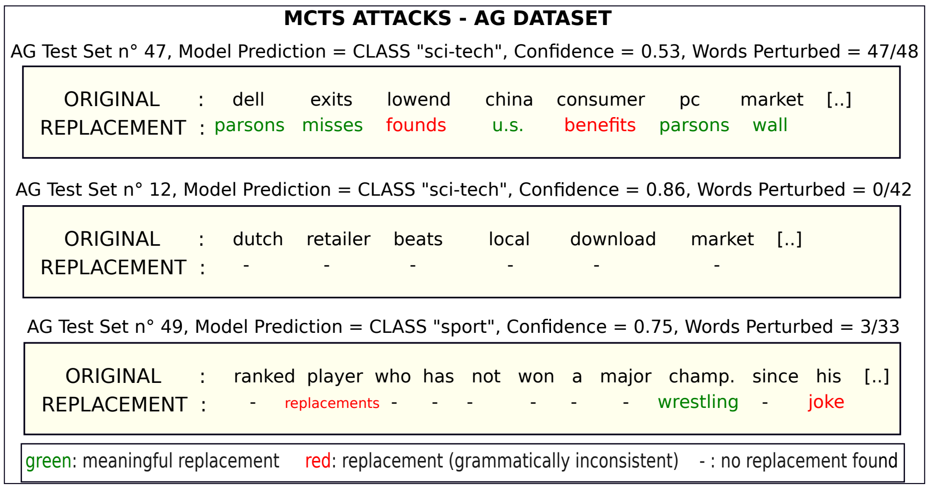
Figure 4.3: Lower bounds indicate classification invariance to any substitution when greater than the embedding diameter **d**, represented by the dotted vertical line. Left: Examples of words safe to any substitution (IMDB, Keras embedding 10d, text no 2). Middle: Examples of words vulnerable to substitutions that may change the classification (IMDB, Keras embedding 5d, text no 1).

For the upper bound, which we calculate via the MCTS algorithm and POS-tag filtering on replacements, an example of a successful perturbation is shown in Figure 4.4, where we evidence the effectiveness of single-word substitutions on inputs that differ in the confidence of the neural network prediction. Another interesting observation is that it is possible to identify perturbations where replacements are meaningful, even with a simple perturbation strategy such as POS-tagging. As shown in the first example in Figure 4.4 (top), the network changes the output class to `World` when the word `China` is substituted for `U.S.`. Although this substitution may be relevant to that particular class, we note that the perturbed text is coherent, and the main topic remains `sci-tech`. Furthermore, the classification also changes when the word `exists` is replaced with a plausible alternative `misses`. This perturbation is *neutral*, i.e., not informative for any possible output classes. In the third sentence in Figure 4.4 (bottom), we note that replacing `championship` with `wrestling` makes the model output class `World`, where initially it was `Sport`, indicating that the model relies on a small number of keywords to make its decision. We report a few additional examples of word replacements for a CNN model equipped with GloVe-50d embedding. Given as input the review `'this is art paying homage to art'` (from the SST dataset), when `art` is replaced by `graffiti` the network misclassifies the review (from *positive* to *negative*). Further, as mentioned earlier, the MCTS framework is capable of finding multiple word perturbations: considering the same setting as in the previous example, when in the review `'it's not horrible just horribly mediocre'` the words `horrible` and `horribly` are replaced, respectively, with `gratifying` and `decently`, the review is classified as *positive*, while for the original sentence, it was *negative.*

The upper/lower bound gap. Robustness results for high-dimensional embeddings are included in Table 4.7, where we report the trends of the average lower and upper bounds of MSR and the percentage of successful perturbations computed over 100 texts (per dataset) for different architectures and embeddings. Further results, including statistics on lower and upper bounds, are reported in Tables 4.4, 4.3, 4.5, and 4.6. They show, both for ℓ_2 and ℓ_∞ norms, the degradation of lower bound robustness guarantees for high dimensional embeddings and an increased number of allowed substitutions, with values that in practice are very distant from the closest attack found by the MCTS. As observed, for example, in Table 4.3, with the number of word substitutions that grows, the highest drop, in absolute value, is when we allow more than one substitution. A possible explanation that connects robustness to linguistics is that the models employed in the analysis build their robustness on n-grams representations, with $n > 1$: while perturbing a single word is not enough to induce a misclassification, they are brittle to substitutions that target two or more words. Conversely, MCTS is more effective when allowed to perturb multiple words, yet the success rate does not increase as the robustness drop of the lower bound, as shown in Table 4.6.

At least three hypotheses explain the discrepancy between the upper and the lower bounds, with some degree of complementarity among all of them. The first hypothesis is that the lower bound is very conservative, and its accuracy depends on the complexity of the model, being deduced from methods that approximate the activation of each neuron of a network. In the same way, MCTS employs heuristics to identify the closest discrete attack, and only in the limit converges to an optimal solution in the sense of closeness to the original input point. The second hypothesis, already discussed in Section 4.3.2, is that high-dimensional embeddings (more than 25 dimensions) suffer from a surface of attack that grows exponentially with the number of dimensions. The third and last hypothesis is that, since the upper bound is a discrete attack, it is implausible that the closest adversarial perturbation in the representation space has an inverse image in the vocabulary. Thus a margin between the (optimal) upper and lower bounds is inevitable.

Results further include an assessment of the effects of counter-fitting to robustness (Mrkšić et al., 2016) as a technique that injects antonyms and synonyms into an



MCTS ATTACKS - AG DATASET

AG Test Set n° 47, Model Prediction = CLASS "sci-tech", Confidence = 0.53, Words Perturbed = 47/48

ORIGINAL :	dell	exits	lowend	china	consumer	pc	market	[..]
REPLACEMENT :	parsons	misses	founds	u.s.	benefits	parsons	wall	

AG Test Set n° 12, Model Prediction = CLASS "sci-tech", Confidence = 0.86, Words Perturbed = 0/42

ORIGINAL :	dutch	retailer	beats	local	download	market	[..]
REPLACEMENT :	-	-	-	-	-	-	

AG Test Set n° 49, Model Prediction = CLASS "sport", Confidence = 0.75, Words Perturbed = 3/33

ORIGINAL :	ranked	player	who	has	not	won	a	major	champ.	since	his	[..]
REPLACEMENT :	-	replacements	-	-	-	-	-	-	wrestling	-	joke	

green: meaningful replacement   red: replacement (grammatically inconsistent)   - : no replacement found

Figure 4.4: Single-word substitutions found with MCTS in conjunction with POS-tag filtering (as defined in Section 4.2.3). Grammatically consistent substitutions are shown in green, inconsistent in red, and a dash indicates that no substitution is found.

embedding space representation in order to improve the vectors' capability for judging semantic similarity. While, in principle, counter-fitting is expected to enhance local robustness, we could not confirm this hypothesis, as both the lower and upper bounds seem not to benefit from such treatment, as shown in Tables 4.3.

4.4 Continuous Robustness: a Critical Appraisal

In this chapter, we have proposed an upper/lower-bound framework to quantify and characterize the region of maximal safety of an NN that solves an NLP task. The experimental results show that meaningful lower-bound robustness guarantees hold only for low-dimensional embeddings: furthermore, there is no guarantee that the region of maximal safety contains words that are expected to preserve the decision of a model (e.g., substituting `good` with `bad` in a sentiment analysis input text can change its ground truth label). At the same time, they vanish for any NN equipped with embeddings with more than 25/50 dimensions (per word). Further, MCTS can efficiently produce attacks that induce a model's misclassification by perturbing only a few words of an input text, also with replacements that should not cause such behavior. All these discrepancies between the formal notion of robustness defined here and a notion that encompasses simple linguistic rules of human language will be the subject of discussion in the next chapter, where we challenge the *de facto* accepted notion of robustness with one that is more aligned with linguistics.

AG News Results: Single Word Substitution

	DIMENSION	ACCURACY		LOWER BOUND	
		Vanilla	**Counter-fitted**	**Vanilla**	**Counter-fitted**
Keras	5	0.414	0.464	0.072 ± 0.066	0.145 ± 0.147
	10	0.491	0.505	0.026 ± 0.025	0.088 ± 0.087
	25	0.585	0.597	0.022 ± 0.025	0.032 ± 0.026
	50	0.692	0.751	0.015 ± 0.009	0.024 ± 0.015
	100	0.779	0.807	0.011 ± 0.007	0.015 ± 0.009
GloVe	50	0.892	0.879	0.04 ± 0.028	0.043 ± 0.03
	100	0.901	0.887	0.027 ± 0.018	0.0 ± 0.0 (NaN)
GloVeTwitter	25	0.848	0.846	0.033 ± 0.025	0.046 ± 0.036
	50	0.877	0.866	0.05 ± 0.012	0.033 ± 0.018
	100	0.833	0.883	0.019 ± 0.012	0.026 ± 0.005

AG News Results: Multiple Words Substitutions

	DIMENSION	L.B. 2 SUBSTITUTIONS		L.B. 3 SUBSTITUTIONS	
		Vanilla	**Counter-fitted**	**Vanilla**	**Counter-fitted**
Keras	5	0.029 ± 0.024	0.065 ± 0.059	0.025 ± 0.017	0.054 ± 0.044
	10	0.013 ± 0.012	0.043 ± 0.042	0.008 ± 0.008	0.028 ± 0.028
	25	0.011 ± 0.008	0.015 ± 0.012	0.007 ± 0.006	0.01 ± 0.008
	50	0.007 ± 0.004	0.012 ± 0.007	0.005 ± 0.003	0.008 ± 0.005
	100	0.006 ± 0.004	0.006 ± 0.004	0.003 ± 0.003	0.003 ± 0.002
GloVe	50	0.02 ± 0.013	0.02 ± 0.014	0.013 ± 0.009	0.016 ± 0.01
	100	0.015 ± 0.007	0.0 ± 0.0 (NaN)	0.01 ± 0.006	0.0 ± 0.0 (NaN)
GloVeTwitter	25	0.014 ± 0.011	0.023 ± 0.017	0.01 ± 0.008	0.0015 ± 0.012
	50	0.024 ± 0.005	0.015 ± 0.009	0.016 ± 0.004	0.011 ± 0.007
	100	0.009 ± 0.006	0.013 ± 0.002	0.006 ± 0.004	0.008 ± 0.002

	DIMENSION	L.B. 4 SUBSTITUTIONS		L.B. 5 SUBSTITUTIONS	
		Vanilla	**Counter-fitted**	**Vanilla**	**Counter-fitted**
Keras	5	0.018 ± 0.012	0.035 ± 0.028	0.014 ± 0.009	0.03 ± 0.021
	10	0.006 ± 0.005	0.02 ± 0.019	0.005 ± 0.004	0.016 ± 0.015
	25	0.005 ± 0.004	0.007 ± 0.006	0.004 ± 0.003	0.006 ± 0.004
	50	0.003 ± 0.002	0.005 ± 0.002	0.003 ± 0.002	0.005 ± 0.003
	100	0.003 ± 0.002	0.003 ± 0.002	0.002 ± 0.001	0.002 ± 0.001
GloVe	50	0.009 ± 0.006	0.01 ± 0.006	0.008 ± 0.005	0.008 ± 0.006
	100	0.007 ± 0.004	0.0 ± 0.0 (NaN)	0.005 ± 0.003	0.0 ± 0.0 (NaN)
GloVeTwitter	25	0.007 ± 0.005	0.011 ± 0.008	0.006 ± 0.004	0.009 ± 0.006
	50	0.008 ± 0.004	0.008 ± 0.006	0.009 ± 0.001	0.006 ± 0.004
	100	0.004 ± 0.003	0.006 ± 0.001	0.003 ± 0.002	0.005 ± 0.001

Table 4.3: Lower bound results for single (top) and multiple words (middle and bottom) substitutions, comparing vanilla and counter-fitted models. The robustness of counter-fitted models is superior to the vanilla counterpart, except for high-dimensional embeddings such as GloVe 100d, where it has not been possible to obtain a bound for the counter-fitted embedding due to computational constraints (nonetheless, the counterpart lower bound is close to zero). Values reported refer to measurements in the ℓ_∞-norm.

IMDB

	DIMENSION	ACCURACY	LOWER BOUND
Keras	5	0.789	1.358 ± 0.604
	10	0.788	2.134 ± 1.257
	25	0.78	1.234 ± 2.062
	50	0.78	0.394 ± 0.079
	100	0.778	0.31 ± 0.041
GloVe	50	0.758	0.133 ± 0.054
	100	0.783	0.127 ± 0.055
GloVeTwitter	25	0.739	0.168 ± 0.093
	50	0.752	0.143 ± 0.02
	100	0.77	0.177 ± 0.057

Stanford Sentiment Treebank (SST)

	DIMENSION	ACCURACY	LOWER BOUND
Keras	5	0.75	0.623 ± 0.28
	10	0.756	0.449 ± 0.283
	25	0.757	0.116 ± 0.14
	50	0.811	0.029 ± 0.012
	100	0.818	0.023 ± 0.006
GloVe	50	0.824	0.053 ± 0.023
	100	0.833	0.028 ± 0.015
GloVeTwitter	25	0.763	0.065 ± 0.023
	50	0.826	0.059 ± 0.031
	100	0.823	0.0 ± 0.0 (NaN)

NEWS Dataset

	DIMENSION	ACCURACY	LOWER BOUND
GloVe	50	0.625	0.013 ± 0.015
	100	0.7	0.018 ± 0.017
GloVeTwitter	50	0.627	0.009 ± 0.006
	100	0.716	0.008 ± 0.009

Table 4.4: Lower bound results for single-word substitutions as found by CNN-Cert and POPQORN, respectively, on the IMDB, SST, and NEWS datasets. Values reported refer to measurements in the ℓ_2-norm.

MCTS Results

	EMBEDDING	EXEC TIME [s]	SUB. (% per-text)	SUB. (% per-word)	UB
IMDB	Keras50d	29.52	6.0	1.4	0.41 ± 0.04
	GloVe50d	39.61	39.7	5.1	0.39 ± 0.016
	GloVeTwitter50d	54.1	47.0	7.7	0.329 ± 0.015
AG NEWS	Keras50d	21.09	50.0	15.6	0.396 ± 0.02
	GloVe50d	19.25	22.4	10.8	0.438 ± 0.042
	GloVeTwitter50d	17.75	21.4	6.6	0.336 ± 0.019
SST	Keras50d	8.36	52.2	19.9	0.444 ± 0.077
	GloVe50d	11.94	81.1	37.4	0.385 ± 0.024
	GloVeTwitter50d	11.96	78.1	36.3	0.329 ± 0.024
NEWS	GloVe50d	75.76	96.5	34.0	0.405 ± 0.045
	GloVe100d	79.31	89.7	29.1	0.442 ± 0.042
	GloVeTwitter50d	77.74	90.9	30.6	0.314 ± 0.033
	GloVeTwitter100d	81.29	89.7	27.7	0.417 ± 0.042

Table 4.5: Upper bound results for single-word substitutions as found by MCTS. We report: the average execution time for each experiment; the percentage of texts for which we have found at least one successful single-word substitution (which results in a class change); the approximate ratio that by selecting randomly one word from a text we find a successful replacement; the distance to the closest meaningful perturbation to the original word found, and namely an upper bound (differently from Table 4.7 and for completeness, here values are reported only considering the values for those words where the perturbations were successful). Values reported refer to measurements in the ℓ_2-norm.

MCTS Multiple Substitutions

	EMBEDDING	2 SUBSTITUTIONS		3 SUBSTITUTIONS		4 SUBSTITUTIONS	
		% per-text	% per-word	% per-text	% per-word	% per-text	% per-word
IMDB	Keras50d	8.5	5.0	13.4	5.9	18.2	6.6
	GloVe50d	43.8	17.7	52.0	21.6	57.5	24.5
	GloVeTwitter50d	44.1	18.3	49.3	23.0	57.1	26.4
AG NEWS	Keras50d	68.1	27.5	72.7	38.3	83.3	47.9
	GloVe50d	31.4	15.8	33.7	16.8	37.0	19.7
	GloVeTwitter50d	23.8	12.5	23.8	15.3	38.0	18.4
SST	Keras50d	64.8	33.0	74.7	40.2	78.0	48.7
	GloVe50d	89.4	58.0	96.4	70.8	97.6	76.5
	GloVeTwitter50d	88.3	57.8	94.1	69.1	95.3	74.9
NEWS	GloVe50d	98.8	55.4	97.3	62.5	97.3	68.6
	GloVe100d	100.0	46.8	95.0	68.0	96.0	65.2
	GloVeTwitter50d	94.5	50.5	97.5	63.0	97.5	71.9
	GloVeTwitter100d	92.7	49.9	98.1	58.2	98.3	65.3

Table 4.6: Upper bound results for multiple-word substitutions as found by MCTS. We report the percentage of texts for which we have found at least a single-word substitution, and the number of times, expressed as a ratio, of finding a successful replacement by perturbing k words randomly from a text (where k is the number of substitutions allowed). We do not report the average execution times as they are (roughly) the same as in Table 4.5. Values reported refer to measurements in the ℓ_2-norm. For more than 1 substitution, values reported are an estimate on several random replacements, as it quickly becomes prohibitive to cover all the possible multiple-word combinations.

Single-Word Substitutions

	EMBEDDING	LOWER BOUND	SUBSTITUTIONS		UPPER BOUND
			% per text	% per word	
IMDB	Keras50d	0.055 ± 0.011	6.0	1.4	0.986
	GloVe50d	0.018 ± 0.007	39.7	5.1	0.951
	GloVeTwitter50d	0.02 ± 0.002	47.0	7.7	0.926
AG News	Keras50d	0.002 ± 0.001	50.0	15.6	0.852
	GloVe50d	0.005 ± 0.004	22.4	10.8	0.898
	GloVeTwitter50d	0.007 ± 0.001	21.4	6.6	0.937
SST	Keras50d	0.004 ± 0.001	52.2	19.9	0.813
	GloVe50d	0.007 ± 0.003	81.1	37.4	0.646
	GloVeTwitter50d	0.008 ± 0.004	78.1	36.3	0.653
NEWS	GloVe50d	0.001 ± 0.002	96.5	34.0	0.679
	GloVe100d	0.002 ± 0.002	89.7	29.1	0.727
	GloVeTwitter50d	0.001 ± 0.001	90.9	30.6	0.707
	GloVeTwitter100d	0.001 ± 0.001	89.7	27.7	0.739

Table 4.7: Statistics on single-word substitutions averaged on 100 input texts of each dataset. We report: the average lower bound of the MSR as measured with either CNN-Cert or POPQORN, w.r.t. the test-set; the approximate ratio that given the word from a text we find a single-word substitution and the average number of words that substituted for a given word change the classification; and the average upper bound computed as the distance between the original word and the closest substitution found by MCTS (when no successful perturbation is found we over-approximate the upper bound for that word with the diameter of the embedding). Each embedding diameter has normalized values reported for lower bounds (measurements in the ℓ_2-norm).

Chapter 5

On the Notion of Robustness for NLP

There is growing evidence that a large part of the NLP research community has adopted the classical notion of adversarial robustness originally introduced for images as a *de facto* standard. In this chapter, we show that this notion is problematic in the context of NLP as it considers a narrow spectrum of linguistic phenomena, namely symbol substitution and deletion. We argue for *semantic robustness*, which is better aligned with the human concept of linguistic fidelity. We discuss which biases *semantic robustness* is expected to induce in a model. We study *semantic robustness* of a range of *vanilla* and robustly trained architectures using a template-based generative test bed. We complement the analysis with empirical evidence that *semantic robustness* is improved in LLMs and not through data augmentation.

5.1 Chapter Overview and Contributions

We first review the classical notions of robustness adopted in NLP and identify their weaknesses in terms of the lack of expressiveness and over-reliance on the neural model text representation. Next, to better align the perception of human robustness to that implemented by a neural model, we formalize (local) *semantic robustness* of NLP as a notion that generalizes local discrete robustness through measuring robustness to linguistic rules rather than word substitution or deletion. This allows us to define (global) *semantic robustness* for a linguistic task such as sentiment analysis, which can be extended to higher-order tasks. We further show that with *semantic robustness*, we can evaluate the performance of a model on cogent linguistic phenomena, which are of interest to both the NLP and the linguistics community. We achieve this by proposing an assessment framework and a simple yet effective testbed based on data augmentation. Last but not least, we wish to highlight the issue of NLP robustness, which has over-focused on trivial and often machine-centric symbol manipulation for the last few years. In short, with this chapter, we make the following contributions:

- We critically appraise the standard notion of robustness in NLP, showing its inconsistency and limited scope when applied to language. We propose a notion of *semantic robustness*, better aligned with human linguistics, as it encompasses phenomena such as sarcasm and mixed sentiment.
- We investigate the merits and limitations of both standard and *semantic robustness* by conducting an extensive experimental evaluation on different architectures and datasets. In particular, we show that LLMs are naturally more robust to linguistic phenomena than standard architectures. We introduce a simple, yet scalable, template-based generation technique to test the semantic robustness of a model, which we compare to an existing dataset of handcrafted sentences that contains the same linguistic phenomena (Blaas et al., 2020).

With the terminology drawn from cyber-security, this work is a 'purple-team' effort to align the key performance indicators of the 'red-team' – whose role is to exploit NLP models with any vulnerability – with those of the 'blue-team', a.k.a. the defenders, who aim to adopt a semantic notion of robustness that implies robustness to linguistic phenomena and are both inadequate when dealing with paraphrases as well as semantically rich tasks.

The contributions presented in this chapter first appeared as (La Malfa and Kwiatkowska, 2022).

5.2 The Standard Notion of Robustness

We begin by discussing the concept of local continuous robustness in its formulation that from computer vision has been applied straight to NLP (Huang et al., 2019; Jia and Liang, 2017; La Malfa et al., 2020). We then consider local discrete robustness, which, differently from the continuous counterpart, manipulates symbols instead of 'numerical' embedding vectors (Alzantot et al., 2018). Nonetheless, we show that the former can be reduced to the latter. Despite this, both definitions only allow one to express robustness to a limited number of linguistic phenomena.

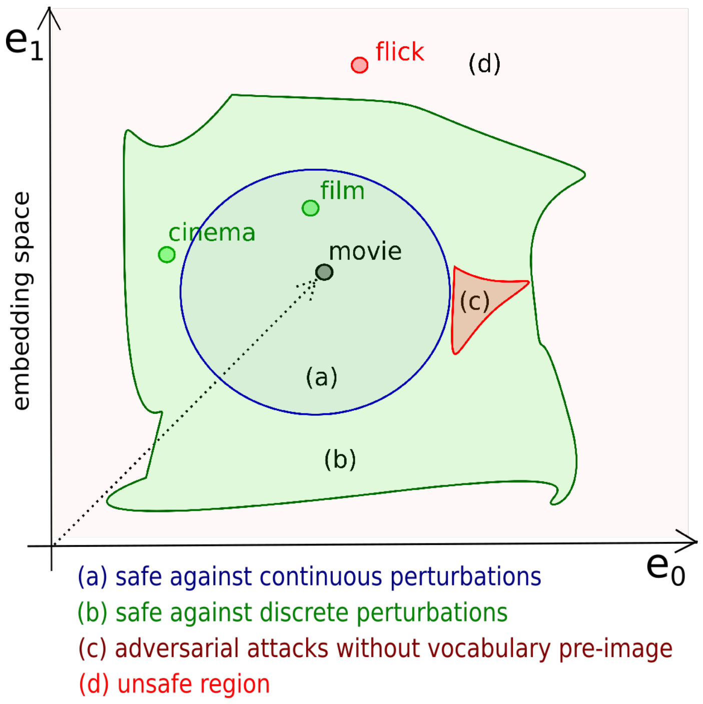

e1
embedding space
flick
(d)
film
cinema
movie
(c)
(a)
(b)
e0
(a) safe against continuous perturbations
(b) safe against discrete perturbations
(c) adversarial attacks without vocabulary pre-image
(d) unsafe region

Figure 5.1: In general, local continuous robustness is an ill-posed property for NLP. A model can be robust to a large surface of attacks in the input neighborhood (green patch (b)), yet a small region of adversarial attacks (red patch (c)) invalidates the verification of larger regions. In the example, the safe input neighborhood (blue patch (a)), a convex region that includes safe replacements, cannot grow any further without violating robustness by encroaching on patch (c). Non-convex representations for an input neighborhood (patch (a)) are possible but computationally expensive and not used in practice.

5.2.1 Continuous and Discrete Robustness

We begin with a simple yet often ignored observation that natural language is discrete while local continuous robustness (Def. 2.18) is defined over a dense representation. Standard embedding techniques (Mikolov et al., 2013; Pennington et al., 2014a) define the word-to-vector mapping over a continuous space, with the input vocabulary being discrete and finite while the output is dense and uncountable. In this setting, local continuous robustness prescribes that a model must consistently classify any vector in the dense region around the input point. This assumption needs to be more linguistically consistent, as a network may present a decision boundary where an adversarial attack that is not a proper word severely reduces the safe region. We graphically illustrate this point in Figure 5.1.

Considering the relationship between local continuous and discrete robustness in NLP, one can easily demonstrate that the former (as expressed in Def. 2.18), *ceteris* *paribus*, is a general case of the latter (Def. 2.22).

Proposition 1. *Local continuous robustness implies local discrete robustness, but the converse is generally false.*

Proof. With the notions of ϵ-Ball and discrete-Ball as introduced in Def. 2.14 and 2.22, it follows that $D\text{-}Ball(x, \epsilon) \subseteq Ball(x, \epsilon)$ but the opposite is not true. For $\epsilon = 0$, both *D-Ball* and ϵ-*Ball* are singletons. □

Linguistically, both formulations allow testing robustness against symbol-to-symbol substitutions or deletions so that the guarantees are valid only for the sentences that differ from the original by at most k-symbols (see Def. 2.24). Robustness to substitutions severely limits and, in practice, makes it computationally intractable to test a model against paraphrases, namely those sentences with overlapping semantics yet different syntax. As an example, if a model f is robust for the sentence `‘the movie was good’`, which implies correct classification for the texts `‘the film was good’`, `‘the film was nice’`, etc., we cannot say the same for the sentence `‘an enjoyable thriller’`, as we do not know where the sentence's representation lies in the NLP representation. This problem arises with the frequency of words in natural language that follows Zipf's law (Zipf, 2013), where rare terms and constructs – hence edge cases – occur more frequently than in other natural phenomena.

Last but not least, both local and discrete robustness does not assess whether the safe region includes perturbations that *violate* the task under consideration: consider the case of sentiment analysis, and a perturbation that turns the sentence `‘The movie is so bad’` into `‘The movie is not bad’` (Hermann et al., 2013).

It is, in fact, well known that, in embeddings and linguistic representations, words like `‘bad’` and `‘good’` are close to each other in the representation space. This could lead to potentially disastrous effects when balancing local robustness, e.g., (Gowal et al., 2018), with accuracy, especially for semantically rich tasks such as sentiment analysis.

5.3 A *Semantic* Notion of Robustness

We now introduce a notion of robustness that goes beyond word replacements and thus permits an assessment of the brittleness of linguistic phenomena that are cogent to humans. To do so, we first need to introduce some notation.

Definition 1 (Oracle). *An Oracle Ω for a task* $\boldsymbol{T}$ *generates samples s compliant with* $\mathbf{T}$. *We denote with* $\Omega, s \models \mathbf{T}$ *an input generated by* Ω *and perturbed by* Ω *that is* *compliant with the task* $\mathbf{T}$*, and with* $\Omega, s \not\models \mathbf{T}$ *the converse. In the case of an Oracle, it holds that* $Prob(\Omega, s \models \mathbf{T}) = 1$*.*

An Oracle is an *augmented, idealized* linguistic/generative model. There is a clear difference between an Oracle Ω and a standard generative model, i.e., that Ω generates samples consistent with $\mathbf{T}$ with certainty. Thus, measurements of a model's performance on samples from Ω are exact.

An Oracle for sentiment analysis. Given a sentiment analysis task $\mathbf{T}$ for movie reviews, an Oracle Ω generates the text `'the movie was (not) good'` and correctly assigned it a positive (negative) label. An Oracle cannot generate any text inconsistent with $\mathbf{T}$, i.e., all those texts that do not explicitly (or implicitly) express a judgment about a movie. An example is the text `'recipe of risotto with mushrooms'`, which is rejected as not compliant with the task, i.e., $\Omega, s \not\models \mathbf{T}$.

Definition 2 (Linguistic Rule)**.** *A linguistic rule is a symbolic generator that manipulates a text s according to a linguistic phenomenon. Those generated texts* $\mathbf{D}'$*, along with the original input, are not rejected by* $\mathbf{T}$*. Formally,* $R : (s, \mathbf{T}) \mapsto S' \ . \ \forall \ s' \in \mathbf{D}', \ \Omega, s' \models \mathbf{T}$*.*

Linguistic rules are flexible symbolic generators: furthermore, since such generators are always consistent, linguistic rules are Oracles.

From a linguistic perspective, since a variation of an input can be very different from the original text, a rule should be allowed to add/remove/replace words while remaining compliant with the task $\mathbf{T}$. For example, one can think of *verb negation* that acts on a text and negates the action expressed by the subject (if any). While this task is often trivial for humans, fully algorithmic solutions to this problem are still limited in their capabilities (Guo et al., 2018), especially in supervised classification, where the verb may or may not be related to the ground truth label. *Hybrid* methods, based on synthetic data augmentation, humans-in-the-loop and deep LLM (Feng et al., 2021; Lin et al., 2020; Huang et al., 2020), are a possible way to proceed. One viable way to generate the replacements is to use template-based data-augmentation techniques (as employed in this chapter and detailed in Section 5.4). More complex approaches involve LLM with humans in the loop who validate the generated perturbations. While for the generative process, LLMs can be trained to be controlled through textual 'seeds' (e.g., in the spirit of the works by (Wu et al., 2021) or (Madaan et al., 2021)), humans play the role of the Oracle.

A rule for *shallow negation.* For a sentiment analysis task $\mathbf{T}$ with positive and negative instances and a positive instance s `‘the movie was good’`, the *shallow negation* rule R negates the sentiment expressed by s, and hence valid perturbations generated by R on s are `‘the movie was not good’`, `‘a bad film’`, but also more involved examples like `‘it is false that the movie is good’`, etc. We name this rule shallow negation as it does not allow for nested negations, regardless of their grammatical consistency (i.e., `‘it is false that the movie wasn’t good’` cannot be generated by R on s).

Definition 3 (Local Semantic Robustness)**.** *Given a classification task* $\mathbf{T} : X \rightarrow Y$*, a model* f *that approximates* $\mathbf{T}$*, a dataset* $\mathbf{D}$ *pertinent to* $\mathbf{T}$*, a dataset* $\mathbf{D}'$ *generated by applying a linguistic rule* R *to (a subset of)* $\mathbf{D}$*, a measure of performance* π *of* f*, and a small positive quantity* δ*,* f *is semantically robust iff*

$$\mathbb{E}_{(x,y)\in\mathbf{D}}[\pi(f(x), y)] - \mathbb{E}_{(x',y')\in\mathbf{D'}}[\pi(f(x'), y')] \leq \delta. \tag{5.1}$$

Informally, a model f that correctly classifies an instance s of a task $\mathbf{T}$ is semantically robust to a linguistic rule R when it exhibits at least the same performance on the set $\mathbf{D}'$ of perturbations s' generated by applying R to any subset of $\mathbf{D}$: an example of a simple measure of performance is accuracy, i.e., $\pi(f(x), y) = 1$ if $f(x) = y$, 0 otherwise. We further observe that this formulation allows for the performance $\mathbb{E}_{(x',y')\in\mathbf{D'}}[\pi(f(x'), y')]$ to even surpass those on $\mathbf{D}$, so this notion entails that f is no worse at correctly solving $\mathbf{T}$ for $\mathbf{D}'$ than it is at solving any other task, and is hence a stronger notion than *bounded invariance.*

A noticeable fact is that local *semantic robustness* as defined is linguistically meaningful as the Oracle rejection of inconsistent samples guarantees it has preserved the semantics of each $s' \in \mathbf{D}'$ w.r.t. $\mathbf{T}$.

Furthermore, local *semantic robustness* is entailed by linguistic generalization, but not the other way round. Linguistic robustness is different from generalization on unseen test cases. The former is entailed by the latter, while the other way round is not necessarily true. Semantic robustness is defined over a rule, while generalization is a more general and hard-to-obtain/optimize objective.

Proposition 2. *Local semantic robustness can be reduced to local discrete robustness but not to local continuous robustness.*

Proof. For local discrete robustness, it is straightforward to define a rule that generates perturbations according to the definition of local discrete robustness. In this sense, the semantic rule R involves extracting the replacements in the embedding's neighborhood of each input word.

As regards local continuous robustness, the invariance over all the input texts s' in an ϵ-ball cannot be mapped back to the embedding (a.k.a. input) vocabulary V by any combination of linguistic rules as they act, by definition, at the symbol level. Since most continuous embeddings are injective non-surjective functions, almost all the vectors in any non-empty region of the space cannot be mapped back to a proper entry of V. $\square$

A sufficient condition for quantifying the *semantic robustness* of a model on an NLP task is that it is possible to measure the performance of such a model on unseen input texts. In this sense, we can measure the *semantic robustness* of a model f that solves a task $\mathbf{T}$ by comparing its performance π with the performance π' of the model on an unseen test bed that contains one or more semantic phenomena. We now describe some illustrative examples of measuring *semantic robustness* for sentiment analysis and then for more involved NLP tasks.

Robustness to *shallow negation* in sentiment analysis. Given a sentiment analysis task $\mathbf{T}$ with positive and negative instances, a model f validated on a dataset $\mathbf{D} = (S, Y)$ is robust to shallow negation when $\forall(s, y) \in \mathbf{D}$, $\forall(s', y') \in R(S \in \mathbf{D}, \mathbf{T})$, $\mathbb{E}_{(s,y)\in\mathbf{D}}[\pi(f(s), y)] - \mathbb{E}_{(s',y')\in\mathbf{D'}}[\pi(f(s'), y')] \leq \delta$ for some $\delta \geq 0$, with R the negation rule that acts on a specific text and negates the sentiment expressed by s. In this sense, π measures the model's accuracy on $\mathbf{D}$ and $\mathbf{D'}$ (samples that contain specific linguistic phenomena). As described in the next section, a test bed can be handcrafted, or distilled from existing datasets, as described in (Barnes et al., 2019).

Semantic robustness in higher-order NLP tasks. We now briefly sketch how we would approach the measurement of *semantic robustness* for higher-order NLP tasks. For Question and Answer (QA) tasks, a measure of robustness can be quantified as the gap between the 'unexpectedness' of an Answer when the Question does/-does not contain a linguistic phenomenon. In Natural Language Inference (NLI), directly applying our framework would be straightforward since NLI is reducible to a classification task. In the same way, when Reading Comprehension (RC) is pursued as a classification task, the evaluation of *semantic robustness* would be similar to sentiment analysis or NLI. In contrast, when the answer requires re-elaborating the input, the measurement of *semantic robustness* would be similar to QA (with possibly a different evaluation metric for $\mathbf{T}$).

Tokens	Replacements
@NEGATIVE@	`‘bad’, ‘poor’, ‘boring’, [...]`
@POSITIVE@	`‘good’, ‘nice’, ‘fantastic’, [...]`
@NAME@	`‘Uma’, ‘Bruce’, ‘Sandra’, [...]`
@SURNAME@	`‘Thurman’, ‘Willis’, ‘Bullock’, [...]`
@CATEGORY@	`‘thriller’, ‘horror’, ‘comedy’, [...]`
@BOOLFALSE@	`‘false’, ‘wrong’, ‘incorrect’, [...]`
@AUGMENT@	`‘very’, ‘extremely’, [...]`

Table 5.1: Candidate perturbation sets used to generate combinations of replacements in template-based texts (Table 5.2).

5.3.1 Task-preserving Generative Method

In this section, we present a concise and scalable generative approach designed to evaluate the *semantic robustness* of a model when confronted with various linguistic phenomena. We provide a comprehensive analysis of the advantages and limitations associated with this method, complementing it with an established dataset comprising sentences featuring the same linguistic phenomena under investigation.

Template-based linguistic rules. We develop a template-based method for generating augmented samples that contain a specific linguistic phenomenon. We predefine a selection of templates (around 50 and 1200 before and after augmentation) for which we know the corresponding output labels (i.e., *positive* or *negative*). In a template, part of the text is fixed while the remaining part is symbolically represented by tokens which are iteratively replaced by combinations of words from candidate perturbation sets. The augmentation preserves the semantics of the sentence while introducing a linguistic phenomenon (such as *shallow negation*). In our implementation of the rules, a perturbation cannot change the template's label. In this sense, the rejection phase (see Definition 3) is embedded in the generative pipeline, while a human might supervise a process that involves an LLM and generations that are possibly label-changing. Examples of templates for each linguistic rule are included in Table 5.2, along with candidate replacements for each token in Table 5.1.

Manually extracted examples. Another viable methodology to obtain sentences containing linguistic phenomena is manually inspect existing datasets. On one side, this approach has the advantage that sentences have already been collected and are, in some cases, the result of meticulous work conducted by human experts (Socher et al., 2013a). On the other hand, this approach is expensive as its cost scales linearly with

Shallow Negation	**Label**
`‘This @CATEGORY@ movie is not @AUGMENT@ @NEGATIVE@.’`	*positive*
`'It is @BOOLFALSE@ that this @CATEGORY@ movie is @AUGMENT@ @POSITIVE@.'`	*negative*
Mixed Sentiment	
`'Despite @NAME@ @SURNAME@ acted well, this @CATEGORY@ movie is @AUGMENT@ @NEGATIVE@.'`	*negative*
`'A @AUGMENT@ @NEGATIVE@ plot for a @AUGMENT@ @POSITIVE@ movie.'`	*positive*
Sarcasm	
`'Starring @NAME@ @SURNAME@ I'd prefer to die rather than watching this @CATEGORY@ movie.'`	*negative*
`'Please throw this @AUGMENT@ long @CATEGORY@ movie into the ocean, and thank me later.'`	*negative*

Table 5.2: Examples of template-based reviews, along with the ground truth label, used to generate sentences that contain the linguistic phenomena studied in Section 5.4.

Shallow Negation	**Label**
`‘It was not good.’`	*negative*
Mixed Sentiment	
`‘The plot was nice, but a little slow.’`	*negative*
Sarcasm	
`‘I love it when people yell at me first thing in the morning.’`	*negative*

Table 5.3: Examples of sentences that contain linguistic phenomena from (Barnes et al., 2019).

the number of people employed in the collection process. For this chapter, we use examples collected in (Barnes et al., 2019). Their methodology entails the acquisition of a specific set of sentences derived from an ensemble of cutting-edge sentiment classifiers, wherein misclassifications occur. These erroneous instances are meticulously singled out to form a subset for further analysis. Subsequently, a comprehensive annotation process is conducted, encompassing several linguistic and paralinguistic phenomena, including but not limited to negation, sarcasm, modality, and various others. A few examples of sentences that contain linguistic phenomena are reported in Table 5.3.

5.4 Experimental Evaluation

We proceed with an extensive experimental evaluation to address the following research inquiries consistently and rigorously. After an introduction to the experimental setup, in terms of linguistic phenomena and architectures employed, we investigate whether models that exhibit robustness in the classical sense also demonstrate *semantic robustness*. Secondly, we explore whether robustness to specific linguistic phenomena arises as a by-product of training accurate NLP classifiers. We then examine whether augmented supervised training, utilizing texts containing a particular linguistic phenomenon, induces generalization across unseen test samples with the same phenomenon, considering different architectural variations. We finally explore the feasibility of training models that exhibit both accuracy and *semantic robustness*, and we assess how unsupervised learning contributes to the concept of *semantic robustness*.

5.4.1 Experimental Setup

We conduct the experiments on models trained – or fine-tuned through data augmentation – on the Stanford Sentiment Treebank dataset (SST-2) (Socher et al., 2013a) and on the dataset collected by (Barnes et al., 2019). The advantages of this approach are two-fold. Firstly, human experts have collected/handcrafted sentences whose syntax/semantics is rich and the level of noise restrained. Secondly, since spurious patterns and over-fitting play a crucial role during training and their influence is hard to estimate and quantify, the cogent compactness of those datasets makes it relatively easy to assess the results. To further estimate the robustness of linguistic phenomena, in the spirit of the evaluation done in (Huang et al., 2020), we utilize a template-based method, whose details are given below, for generating augmented samples for a selection of linguistic phenomena to create a test bed, which we use for systematic evaluation of *semantic robustness*.

Regarding the reproducibility of the results we present, all of the experiments have been conducted on a Fedora 32 mid-end laptop equipped with 16GB of RAM and an Intel-i5 CORE 8^{th}-generation. All the neural network models have been built, trained, and tested with Keras (Chollet and others, 2018), while for experiments that involved BERT (Devlin et al., 2019b) we relied on the PyTorch implementation (Paszke et al., 2019).

Linguistic phenomena. Following the work in (Barnes et al., 2019), we have chosen interesting linguistic and para-linguistic phenomena, excluding those that require external knowledge to be solved (i.e., not explicitly expressed in the sentence). For example, consider the review `‘This movie is another Vietnam’`, which can be correctly classified as unfavorable if the model knows that specific way of saying (i.e., exogenous knowledge). We now briefly describe the linguistic phenomena that are the object of our robustness evaluation:

- *Shallow negation*: when the sentiment of a sentence is negated. We do not consider nested negations, which make the recognition of the phenomenon considerably harder (Wiegand et al., 2010; Socher et al., 2013a; Pröllochs et al., 2015).

- *Mixed sentiment*: when phrases of different polarity appear in the same sentence (Kenyon-Dean et al., 2018; Barnes et al., 2019). We only consider texts where the overall sentiment is still not ambiguous for a human.
- *Irony/sarcasm*: when a sentence makes some premises that are then violated (Hao and Veale, 2010). This is known to be one of the hardest yet pervasive linguistic phenomena of human language.

5.4.2 Comparative Study

We compare architecturally different models on the three linguistic phenomena we previously introduced. We conduct an extensive evaluation on four neural architectures, namely fully connected (FC), convolutional (CNN) (Zhang et al., 2015b), Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and self-attention (Vaswani et al., 2017). We choose the number of hidden units of each layer so that the number of parameters is approximately the same and in the order of 40K. Each input text is 25 words long (eventually padded or cut), while each word is mapped to a vector of real numbers through a 50-dimensional embedding, pre-trained on the SST-2 task (Chollet and others, 2018). Each network comprises 3 layers, where the topology of the last two is shared, i.e., a 32 hidden units ReLU and a 2 hidden units softmax layer (both are dense). The first layer depends on the specific topology under examination (e.g., self-attention will have a self-attention layer, LSTM a Long Short-term Memory cell, etc.): the first layer has 32 hidden units for the FCs, 44 ReLU kernels of size 3 for the CNNs, 75 tanh hidden units for the LSTMs and 32 ReLU hidden units for the self-attention networks.

For each linguistic phenomenon, we analyze and compare the robustness of 20 models trained on *plain* SST-2 dataset (i.e., no semantic data augmentation of any kind) and then on a *semantically augmented* version of the same dataset, whose details we provide in the next paragraph.

Semantic robustness through data augmentation. In this section, we study how – and to what extent – data augmentation, along with architectural inductive biases, can be used to enhance *semantic robustness* to different linguistic phenomena. We re-trained the models of the previous section by adding samples from (Barnes et al., 2019) that contain one of the specific rules used previously in training set up to a multiplicative factor to balance a large number of samples of the SST-2 dataset. With the SST-2 train set that accounts for approximately 112K input samples and

	Train	FCs	CNNs	LSTMs	Self-attention
Shallow Negation	*Vanilla*	0.4034 ± 0.0214	0.4032 ± 0.0124	0.4771 ± 0.0143	0.4790 ± 0.0059
	Augmented	0.4062 ± 0.0167	0.4249 ± 0.0255	0.6387 ± 0.0387*	0.5954 ± 0.0027*
Mixed Sentiment	*Vanilla*	0.4707 ± 0.0360	0.4986 ± 0.0415	0.5110 ± 0.0251	0.5487 ± 0.0099
	Augmented	0.4912 ± 0.0339	0.5271 ± 0.0387	0.6357 ± 0.0317*	0.5617 ± 0.0048
Sarcasm	*Vanilla*	0.5136 ± 0.0504	0.4681 ± 0.0327	0.5578 ± 0.0128*	0.5240 ± 0.0132
	Augmented	0.5297 ± 0.0657	0.4678 ± 0.0317	0.4807 ± 0.0197	0.6236 ± 0.0218*

Table 5.4: Comparison of accuracy of 20 *vanilla* and *augmented* models obtained for four different architectures (FCs, CNNs, LSTMs, and self-attention), on three linguistic phenomena (*shallow negation*, *mixed sentiment* and *sarcasm*). All the networks have been trained on the SST-2 dataset. *Augmented* models are *vanilla* models fine-tuned on the linguistic rules of interest. Symbol *, when present, means that the improved performance (from *vanilla* to *augmented*, or the other way round) is statistically significant. Interestingly, *sarcasm* is harder to learn, and models fine-tuned on this phenomenon perform as well as their *vanilla* counterparts (when not worse).

each semantic rule that generates roughly $500 - 1000$ new samples, semantic data augmentation with a multiplicative factor of 1 accounts for additional 1K samples, etc. While for a multiplicative factor of 500, none of the models exhibit any improvement in the semantic tasks, for a multiplicative factor of 750 we observe some improvement in LSTMs and self-attention. While the experiments suggest that FCs and CNNs cannot learn any of the three linguistic phenomena we studied, LSTMs and self-attention networks benefit from data augmentation. Concerning Table 5.4, both LSTMs and self-attention improve considerably on *shallow negation*. On *mixed sentiment*, augmented LSTMs substantially improve over the *vanilla* counterpart. At the same time, self-attention does not seem to exploit the additional information (despite a slight improvement over the *vanilla* case). Finally, data augmentation allows self-attention to improve significantly on *sarcasm*. However, the same regime is detrimental for LSTMs, where the *vanilla* networks consistently outperform those trained on augmented data.[1] Finally, for a multiplicative factor of 1K or superior, we observe a detrimental effect on the robustness of each model that is comparable to the *vanilla* SST-2 training.

5.4.3 Classic Robustness is Linguistically Brittle

We have compared *robust* models trained with IBP (Interval Bound Propagation) (Gowal et al., 2018) with their *vanilla* counterparts. For different values of $\epsilon = (0.001, 0.01)$ in

[1]For FCs and CNNs the average accuracy on the SST-2 test set is 0.8993 ± 0.0029 and 0.9077 ± 0.0038 respectively, while the accuracies of LSTMs and self-attention are 0.9101 ± 0.0033 and 0.8963 ± 0.0015.

	Accuracy (Barnes et al., 2019)	**Accuracy (Our Benchmark)**
Shallow Negation	0.8552	0.7928
Mixed Sentiment	0.6024	0.6974
Sarcasm	0.7111	0.8455

Table 5.5: Summary of BERT *semantic robustness* on different linguistic phenomena, tested on samples from (Barnes et al., 2019) (left column) and from our template-based benchmark (right column). A BERT model has been fine-tuned on the SST-2 dataset for these results.

the ℓ_∞-norm, which makes the model-to-model results easy to explain (La Malfa et al., 2020), and an embedding diameter of approximately 3.17, we assess IBP-induced robustness on semantic rules. Interestingly, their performance is comparable (when not worse) to the brittle counterparts for all the linguistic phenomena we analyze, thus validating our previous observation, i.e., that models robust in the classical sense have a minimal syntax/semantic manipulation capability. Results are reported in Table 5.6.

5.4.4 Accuracy is a Red Herring: the BERT Case

We analyze the relationship between *semantic robustness* and accuracy of a Large Language Model (LLM): while it is known that LLMs have an improved accuracy on out-of-distribution (ood) data (Hendrycks et al., 2020), there is no explicit agreement on the nature of the semantic phenomena, i.e., whether they are *linguistic outliers* or ood. Although, in deep learning, a trade-off has been observed between the classical notions of robustness and accuracy (Tsipras et al., 2019), *semantic robustness* does not seem to exacerbate this phenomenon. We fine-tuned the BERT language model (Devlin et al., 2019b) on the SST-2 dataset and tested its robustness on the linguistic phenomena we introduced in the previous section.

Despite an accuracy of 0.90, which is in line with the accuracy of the (simpler) architectures we tested previously, BERT's *semantic robustness* is considerably higher than the *shallow* counterparts (BERT has 16 hidden layers, the models in our benchmark 3). BERT has an accuracy of 0.7928 on *shallow negation*, 0.6974 on *mixed sentiment*, and 0.8445 on *sarcasm.* The linguistic phenomenon where BERT performs worst is *mixed sentiment*, as (i) a few recent works point out the limitations of LLM models such as BERT when learning complex syntactic/semantic constructs (Sinha et al., 2021); (ii) we have shown in our previous evaluation that self-attention (along with any other model) is especially brittle to that linguistic construct, despite the layer's name suggesting the opposite. In general, we interpret this linguistic performance as a result of the massive amount of unsupervised training (i.e., the masked language prediction) to which BERT is subjected before being fine-tuned on our supervised task: in this sense, the phase of pre-training, which shapes the dynamics of BERT's contextual embeddings, enables it to outperform shallow models on the linguistic phenomena considerably.

We finally validate the results of (Barnes et al., 2019), proving that, on their challenging dataset, which contains texts from other non-movie-review datasets (so certainly out of distribution samples), BERT has an accuracy of 0.8552, 0.6024 and 0.7111 on respectively *shallow negation*, *mixed sentiment* and *sarcasm*. Therefore, this justifies that the task we set up with our synthetic augmentation through templates is a solid alternative benchmark for *semantic robustness*. We summarize the results in Table 5.5.

Ablation Study of BERT. We performed an ablation study of BERT to assess the role of the stacked embeddings for *semantic robustness*. We trained different semantic classifiers on top of a decreasing number of BERT embedding layers. We then measured the *semantic robustness* on *shallow negation*, *mixed sentiment* and *sarcasm* on samples from (Barnes et al., 2019): we found that, despite the accuracy on the task (SST-2) being strongly correlated with the depth of the BERT embedding, *semantic robustness* is not, as depicted in Figure 5.2. While the best performing layer is the penultimate layer (a phenomenon that is already known in the literature (Rogers et al., 2020)), we could not find a layer that performed the best on all the tasks, a result that leads us to conclude that stacked attention embeddings are fundamental but their internal representation w.r.t. linguistic phenomena (i.e., the 'semantics of BERT') is still poorly understood.

To complement the analysis, we tried to disentangle the role of pre-training from that of the embedding depth and attention (which are considered in the design of each BERT hidden layer) by training a very deep LSTM, with 100 input words and an embedding size of 100, which we then tested on the same semantic phenomena as in the previous evaluation. Interestingly, despite an accuracy of 0.9 on the SST-2 test set, the accuracy on *shallow negation* is 0.5789, 0.6684 on *mixed sentiment* and 0.7 on *sarcasm*. Although we cannot conclude anything definite, we suspect that the role played by massive pre-training (next word/sentence prediction) is much more important than that of depth and attention, which is in agreement with observations emerging from other recent studies (Liu et al., 2021).

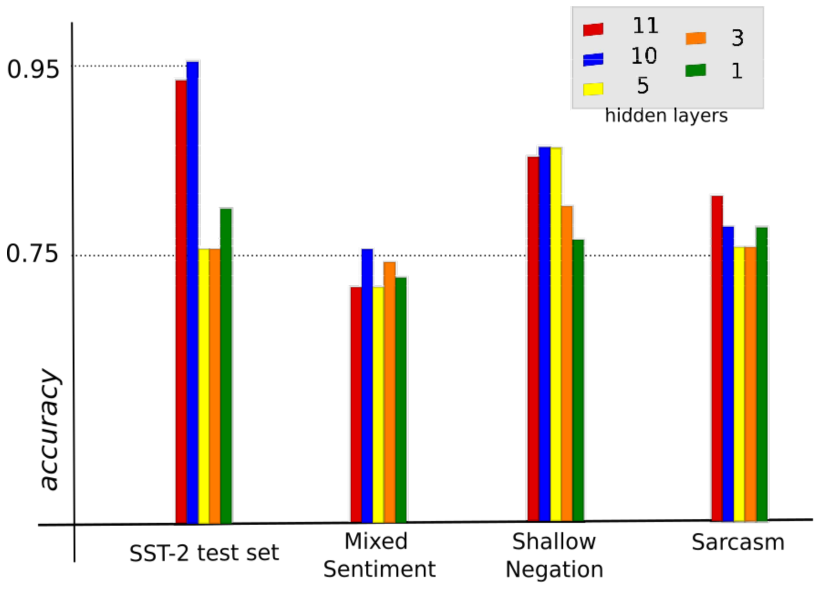


Figure 5.2: Ablation study of BERT on (Barnes et al., 2019), measuring accuracy for 5 different network depths. While depth plays a fundamental role in achieving accuracy on a test set (*SST-2*) and certainly plays a role (albeit minor) on *shallow negation*, it seems not to be correlated to the model performance on *mixed sentiment* and *sarcasm*.

5.4.5 Robustness Induced Biases

In this section, we examine the relationship between common inductive biases that have inspired the design of machine learning algorithms for the past decades (Mitchell, 1980), and recently also neural networks (Kharitonov and Chaabouni, 2021), connecting them to the notions of robustness we dissected in Chapter 4. In particular, we compare local continuous to local *semantic robustness*.

Minimum Cross-validation Error. There is empirical evidence in the literature (Huang et al., 2019; Jia et al., 2019) that continuous robustness does not naturally induce better performance on trained models. Indeed, most of the models that are trained to be robust are less accurate than their brittle counterparts. The margin causes this side-effect to propagate through the network to the output and to induce invariance to the nearest neighbors of a given input. "Shielding" the model with a wide margin of possibly unrelated terms leads to inconsistent treatment of different sentences (human language abounds in edge cases). This is testified by further experiments shown in Figure 5.3 (top). Concerning *semantic robustness*, generalization on cogent linguistic rules does not necessarily benefit a model's performance, as demonstrated by experiments we conducted on 30 networks trained to be semantically

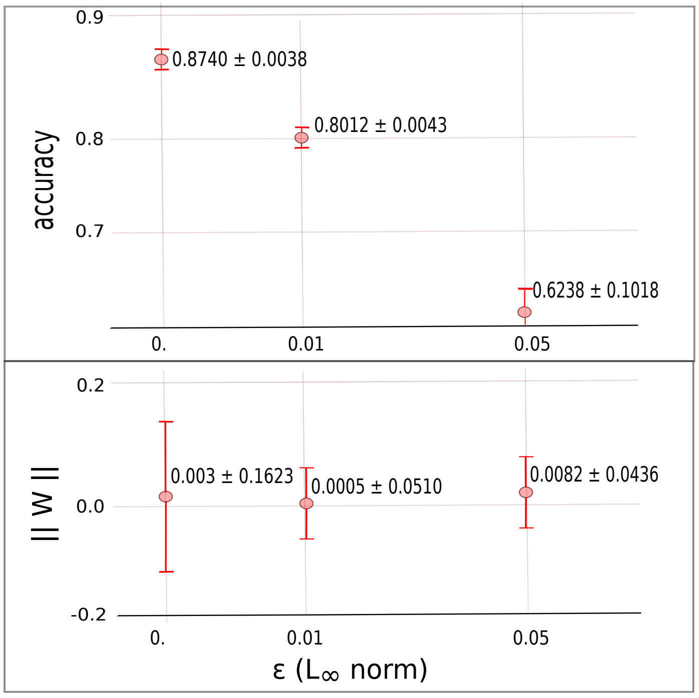


Figure 5.3: On the top plot, we show the average accuracy of 30 trained FC models on the SST-2 dataset, compared for different values of ϵ-robustness. Measurements are taken w.r.t. the ℓ_∞-norm, as it allows us to compare the maximum robustness variation along any dimension. For ϵ equal to 0., a model is not robustly trained; otherwise, it is through IBP (Gowal et al., 2018). There is a clear trade-off between robustness and accuracy. On the bottom plot, the average norm of the models' parameters indicates that robust models tend to have lower variance and hence arguably lower complexity.

robust against *shallow negation* vs. their vanilla counterpart. Both populations have been trained on the SST-2 dataset (Socher et al., 2013a). Robustness is enhanced through simple data augmentation on the dataset provided by Barnes et al. (Barnes et al., 2019), whereas the test is performed on unseen sentences that exhibit the same linguistic phenomenon. While the vanilla networks have an average accuracy of 0.9036 ± 0.0019 on the test set and 0.4916 ± 0.0074 on the *shallow negation* test set, those that have been robustly trained have an accuracy of 0.8838 ± 0.0049 and 0.5491 ± 0.0124, respectively.

Minimum Description Length. Local continuous robustness is a strong regularizer (Gowal et al., 2018). Classical methods used to induce local robustness for NLP (such as IBP), which propagate through all the embedding dimensions and thus amplify the noise, are nonetheless playing an important role as they smooth

	Train	**FCs**		**CNNs**	
Shallow Negation (Our, Barnes et al., 2019)	*Vanilla*	0.4034 ± 0.0214	0.6303 ± 0.0231	0.3753 ± 0.0091	0.4553 ± 0.0719
	IBP ($\epsilon = 0.001$)	0.3852 ± 0.0071	0.6461 ± 0.0039	0.4954 ± 0.0273*	0.5079 ± 0.0822
	IBP ($\epsilon = 0.01$)	0.4249 ± 0.0260	0.6145 ± 0.0263	0.4715 ± 0.0134*	0.4320 ± 0.0501
Mixed Sentiment (Our, Barnes et al., 2019)	*Vanilla*	0.4707 ± 0.0360*	0.6976 ± 0.0126	0.4764 ± 0.0327	0.5506 ± 0.1476
	IBP ($\epsilon = 0.001$)	0.2918 ± 0.0121	0.7205 ± 0.0048	0.5402 ± 0.0961	0.4590 ± 0.1205
	IBP ($\epsilon = 0.01$)	0.2824 ± 0.0169	0.7072 ± 0.0133	0.4485 ± 0.0844	0.5506 ± 0.1476
Sarcasm (Our, Barnes et al., 2019)	*Vanilla*	0.5136 ± 0.0504	0.7133 ± 0.0156*	0.4799 ± 0.0393*	0.3067 ± 0.2883
	IBP ($\epsilon = 0.001$)	0.4333 ± 0.0092	0.5578 ± 0.0185	0.6352 ± 0.3962	0.5778 ± 0.3564
	IBP ($\epsilon = 0.01$)	0.4406 ± 0.0943	0.5222 ± 0.0995	0.1650 ± 0.1866	0.1593 ± 0.1030

Table 5.6: Comparison of 20 *IBP-trained robust* models (Gowal et al., 2018) and their *vanilla* counterparts on samples generated through templates on our benchmark (left subcolumn) and samples exhibiting the same linguistic phenomenon from (Barnes et al., 2019) (right subcolumn): both populations of networks have been trained on the SST-2 dataset. IBP, which we use to train robust models for two different values of ϵ (0.001 and 0.01), cannot ensure robustness to simple semantic rules and, in a few cases, worsens the classifier's performance. Symbol *, when present, means that the improved performance (from *vanilla* to *IBP* or vice-versa) is statistically significant. We consider the two architectures (FCs and CNNs) supported by (Gowal et al., 2018).

out the network's hidden activations. We report the results of experiments that we conducted that support this hypothesis in Figure 5.3 (bottom). Regarding *semantic robustness*, we cannot conclude anything definitively, but the evidence suggests that semantically robust models are not necessarily smoother than the vanilla counterparts. We compared the weights' norm of 30 networks trained to be robust against *shallow negation*, i.e., whose training data was augmented with samples that contain that phenomenon, vs. their vanilla counterparts, i.e., no data augmentation. While the difference between the performance of the two networks on unseen texts that contain that linguistic phenomenon is substantial, there is very little difference in the norm of the two populations, which are respectively 0.0017 ± 0.0019 (vanilla) and 0.0064 ± 0.0032 (robust).

Nearest Neighbors. Local robustness induces a strong bias towards nearest neighbors, as it imposes the same classification label on all the perturbations close to the input point (see Def. 2.18). This assumption is critical as robust training underestimates the effect of making a model robust, treating all the dimensions in the embedding equally important. We hypothesize this causes the the deterioration of the performance of robust models in NLP. The induced invariance along any dimension reduces the effectiveness of the embedding representation on cogent syntactic/semantic tasks such as word-sense-disambiguation, polysemy, etc. *Semantic robustness* takes a different approach and is expected not to be robust to nearest neighbors in the embedding space but rather to perturbations that are generated by the linguistic rules for which they have been robustly trained. For an increasing number of embedding dimensions, *semantic robustness* does not suffer in principle from the trade-off between the performance on linguistic tasks (Chen et al., 2013).

5.5 Conclusions

In this chapter, we have discussed the limitations of standard approaches to robustness. The main concerns of such approaches are to guarantee a model's invariance against word substitution and deletion: two perturbation approaches that are linguistically limited. We thus introduce a notion of robustness encompassing linguistic phenomena and a generative approach to control the generation of test beds. We have performed an extensive experimental evaluation on standard models (FCs, CNNs, LSTMs, and attention-based networks) and LLMs to answer pertinent research questions that lie at the root of the lack of robustness of such models against linguistic phenomena. We have further characterized the notion of *semantic robustness* in terms of inductive biases that induces in a model. The strengths of our approach are the intuitive and interpretable results in terms of the capacity of a model to represent and classify sentences that contain linguistic phenomena that are cogent for humans. Our framework also shows that LLMs are inherently more robust to linguistics phenomena than models trained on standard word embedding representations. On the other hand, for our notion of Oracle to work correctly, it needs to be supported by a human expert or a generative process that almost always produces consistent utterances. While possible in principle, as testified by our simple, template-based generative method or by the examples collected by human experts in (Barnes et al., 2019), these approaches either lack linguistic variability of the augmented dataset or do scale linearly with the number of human experts involved in the generative process.

In the next chapter, we introduce *syntax robustness* as a complementary notion of *semantic robustness*. Such a notion leverages syntax as a cornerstone of post-structuralist linguistics and assesses whether LLMs robustly encode abstract representations such as trees.

Chapter 6

Robustness of Syntactic Structures

Recent works investigated whether linguistic representations, and in particular LLMs, encode syntactic information of a text (Manning et al., 2020). Surprisingly, the results suggest that LLMs' embeddings store structured information sufficient to reconstruct the syntax tree of a sentence alongside further information that has been thought impossible to keep within a continuous representation. In this chapter, we test the validity of this theory through the lens of robustness: we propose the notion of *syntactic robustness* as the consistency of a representation to retrieve coherent syntactic information under perturbations that preserve the syntax of the original text.

We propose a method to evaluate the syntactic consistency and robustness of linguistic representations that leverages probing tasks (Conneau et al., 2018; Manning et al., 2020), namely, neural networks trained directly on the representation embedding to evaluate the representation's ability to encode a specific linguistic phenomenon, such as the syntax tree of a sentence. To this end, we propose an efficient probing method to perturb the input text so that its syntax (or context) is largely preserved. We validate the perturbations to show they can serve as an effective proxy for syntax-preserving perturbations. We focus on syntactic robustness, which informs our selection of four probing tasks, but note that other tasks can be easily incorporated.

To assess robustness, we aim to measure the performance of a language model to probing tasks on the original and perturbed datasets. More specifically, we define a measure of robustness in terms of aggregating (averaging) the worst-case drop of performance of a collection of probing tasks over a given dataset for a given perturbation budget, which then captures the model's ability to encode the linguistic phenomena, and is, therefore, more appropriate for NLP settings.

In summary, in this chapter, we make the following contributions:

- Propose measures to evaluate the robustness of linguistic representations that leverage probing tasks.
- Develop a methodology for analyzing an LLM's ability to capture complex syntactical information underlying its training data robustly.
- Demonstrate how our robustness metrics reveal that context-free representations are equally brittle to manipulations as more sophisticated context-dependent representations.
- Provide empirically insightful observations into feature collapse, training duration, and depth of pre-trained LLM heads from the robustness perspective.

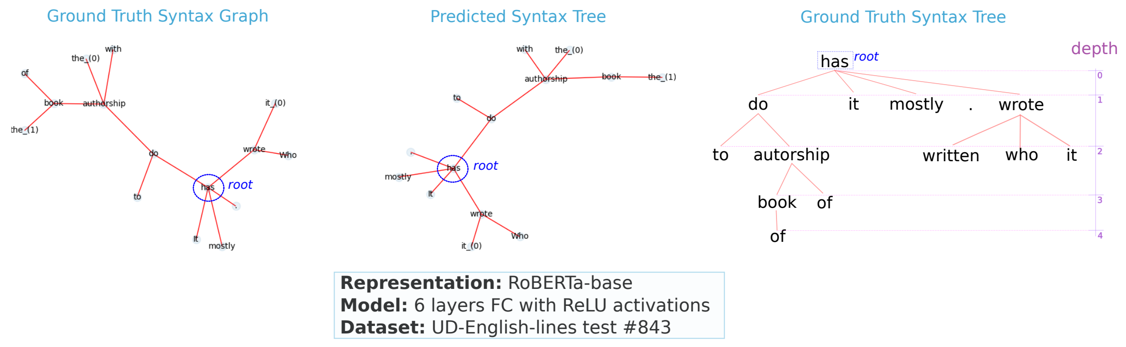
The chapter, based on the preprint (La Malfa et al., 2022), is structured as follows. We introduce some background notation, along with definitions that will be useful throughout the investigation; in particular, we describe how through probes and probing tasks (see Def. 2.13), we inspect a representation's ability to store syntactic information within its embedding. We then define *syntax robustness* and a perturbation scheme that is expected to preserve the syntax correctly, as opposed to one that does not necessarily satisfy this requirement. The chapter proceeds with an extensive empirical evaluation of the robustness of static and dynamic representations over four syntactic probing tasks and six datasets. This serves as a prelude to the conclusion, where we critically analyze the results and their impact on the field.

6.1 Motivation and Setting

The unprecedented and indisputable progress made by NLP techniques is nonetheless counteracted by a series of critiques that show how LLMs are unable to perform basic reasoning (Niven and Kao, 2019), have considerable biases (Bras et al., 2020), are not well aligned to stakeholder values (Bender et al., 2021), and are brittle in the face of adversarial examples (La Malfa and Kwiatkowska, 2022). Such studies clarify that sustainable, long-term advances in NLP must be facilitated by appropriate metrics that capture how LLMs represent the complex linguistic patterns underlying their training data (Bender and Koller, 2020).

In this sense, representing the linguistic and grammatical structure underlying linguistic data plays an influential role in the robust generalization of any NLP system.

There are at least two standard representations of syntax trees in NLP and linguistics, namely dependency and constituency trees. In the former, each word corresponds



Representation: RoBERTa-base
Model: 6 layers FC with ReLU activations
Dataset: UD-English-lines test #843

Figure 6.1: A syntax graph reconstructed via the structural probe task (see Def. 5) from a RoBERTa representation is shown in the middle; for comparison, the ground truth structure is sketched on the left. On the right, the same structure is displayed as a dependency tree (annotated with additional information so that dependencies and hierarchies between words are made clear) so that other supervised tasks can be instantiated, e.g., identifying the root, or computing the depth of the tree.

to a node, and the tree structure reflects the word order, while, in the latter, words themselves are terminal nodes whose order follows the ‘bare phrase structure’ (as per the minimalist program by Noam Chomsky (Chomsky, 2014a)). We work with dependency tree representations in this chapter, but the methodology and results can be extended to the constituency representation standard.

Two of the most influential NLP papers written in recent years directly leverage linguistics: in (Mikolov et al., 2013), the authors build representations as prescribed by the distributional hypothesis (Harris, 1954), while in (Devlin et al., 2019a) as well as (Brown et al., 2020), LLMs are trained to predict masked or following words in context, an activity supported by substantial evidence of its occurrence in the human brain (Schrimpf et al., 2021). Remarkably, linguistic representations such as LLMs are also arguably capable of accurately representing structures such as syntax trees (Manning et al., 2020), which has motivated researchers to investigate their linguistic capabilities (Rogers et al., 2020; Sinha et al., 2021). An example of a tree, rebuilt with a neural network that learns from a sentence via its embedding representation, to approximate the distance between words in the dependency tree (Manning et al., 2020), is reported in Figure 6.1.

Last but not least, the extent to which LLMs represent structured linguistic information can help us better understand the foundations of statistically-based language learning, particularly its relationship with symbolic structures (such as trees).

6.2 Probing Tasks for Model Introspection

6.2.1 Methodology

As introduced in Chapter 2, a sentence $s = \{s_1, .., s_l\}$ is a finite sequence of $l > 0$ symbols (here words) defined over finite vocabulary V. A linguistic rule assesses the violation of a property by a sentence s, and we denote a sentence s satisfying a rule, $\mathbf{R}$, with $\mathbf{R} \models s$. A language, $\mathbf{L}$, is defined by an alphabet V and a (possibly infinite) set of rules $\mathbf{R} = \{R_1, .., R_n\}$. Given a linguistic representation/embedding ψ and a sentence s, this chapter's central question of interest is what information does ψ extract robustly from s? To answer this question, we consider using a perturbation-based analysis. Specifically, given another sentence s' that is semantically similar to s, how does $\psi(s)$ differ from $\psi(s')$? As posed, such perturbation analysis is reminiscent of the problem posed by adversarial examples (Szegedy et al., 2014a), but a naive adaptation to NLP is devoid of the nuance of natural language and inappropriate for this setting (see Chapter 5). We address this shortcoming with a two-phased framework. Firstly, we seek to gain insights into the syntactic properties a language model understands through a probing task. Secondly, we propose an efficient scheme for computing perturbations that preserve the syntax of a given sentence s as much as possible and study how such perturbations affect the model insights from the probing task.

6.2.2 Probing Tasks for Model Introspection

Probing tasks have emerged as a valuable way to assess a language model's grasp of complex linguistic concepts. These tasks are categorized into surface, syntax, and semantic probing tasks, as described in (Conneau et al., 2018). While surface tasks can be solved by looking at information that is immediately accessible to the model, such as the number of tokens in the input sentences, semantic tasks require understanding what a sentence denotes (e.g., guessing the number of the subject of the main clause). Finally, as in the case of this chapter, syntactic tasks check whether a model can infer the hierarchical structure of sentences, for example, reconstructing its dependency tree.

The key idea is to design a probing task that is linguistically specific and easy enough to ensure that solid performance on the task indicates that the language model has accurately captured the linguistic phenomenon. We select four probing tasks, which we design to assess the presence of syntactic structures in linguistic representations, but stress that our framework could be extended to any of the ten probing tasks presented in (Conneau et al., 2018).

We report the definition of a generic probing task, which serves as a basis to describe the four syntactic tasks that our robustness framework will test.

Definition 4. *(Probing Task) Given a set $S = \{s^{(1)}, .., s^{(n)}\}$ of $n > 0$ sentences from a language $\mathbf{L}$, each paired with a label $T = \{t^{(1)}, .., t^{(n)}\}$, a* probing task *consists of finding a mapping f from each sentence representation $\psi^{\theta}(s)$ s.t.* $\mathbb{E}_{(s^{(i)},t^{(i)})\sim(S,T)}\left[\mathcal{L}(f(\psi^{\theta}(s)), t)\right] > p$*, where $\psi^{\theta}(s)$ is a representation of an input sentence returned by an embedding or an LLM representation, and $\mathcal{L}$ is a measure of performance of such a reconstruction and p some positive quantity that certifies a given level of performance.*

The first syntactic probing task we propose to study is the *syntax reconstruction* task. An accurate understanding of the information content of a sentence s depends on the reader's ability to understand the intra-word relationships in s. This applies to natural and programming languages, where parse trees are essential to understand source code.

A syntax tree t is an undirected, acyclic graph $G := (s, A)$, where the words of s are vertices and A is an edge list that contains an edge between two words if they modify each other or are contextually linked, see (Manning and Schutze, 1999) for more details. As stated previously, we work with dependency tree representations in this chapter, but the methodology and the results can be extended to the constituency representation standard. Formally, the syntax reconstruction probing task is given as follows:

Definition 5. *(Syntax Reconstruction) Given a set $S = \{s^{(1)}, .., s^{(n)}\}$ of $n > 0$ sentences from a language $\mathbf{L}$ and their syntax-tree representation $T = \{t^{(1)}, .., t^{(n)}\}$, syntax reconstruction is a probing task f from S to T that guarantees sufficient performance.*

This task, called the 'linear structural probe' (Manning et al., 2020), is commonly used as a proxy of the capabilities of a representation to recognize the mutual relationships between words in a sentence.

In practice, the syntax probing task consists of extracting, from a sentence representation, the distance between each pair of words, as they are arranged in the dependency parse tree of the sentence itself (see Figure 6.2): the task is commonly used as a proxy of the capabilities of a representation to recognize the mutual dependency relationships between words in a sentence, represented as a directed graph. Using probing tasks to assess the capabilities of a model has become a popular approach with the development of increasingly complex linguistic representations. However, some studies have shown that probes can only reveal the correlation between the traces of a symbolic structure in a representation and its performance on a task (Belinkov, 2022; Ravichander et al., 2020). In this thesis, we use probes to provide evidence of the existence of syntactic structures in linguistic representations rather than testing their performance on higher-level NLP tasks.

Another aspect of interest is that probes are usually linear, as one wants to assess how representations encode features that are immediately available to solve the task (Niven and Kao, 2019), despite a few recent works criticizing their excessive simplicity in favor of non-linear ones (Pimentel et al., 2020; White et al., 2021).

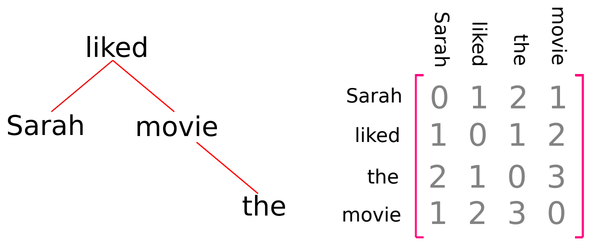


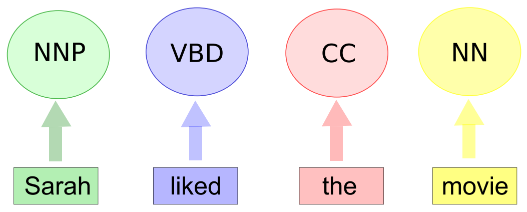
Figure 6.2: The dependency parse tree of a sentence (left), alongside the matrix of distances between pairs of words in the tree (right).

The second probing task disregards intra-word relationships and focuses on a language model's ability to identify the part of speech of a given word. Formally, the part-of-speech (POS) tagging task is given as:

Definition 6. *(POS-tagging) Given a set $S = \{s^{(1)}, .., s^{(n)}\}$ of $n > 0$ sentences from a language $\mathbf{L}$ and the POS-tags for each sentence, $POS = \{pos^{(1)}, .., pos^{(n)}\}$, POS-tag reconstruction is a probing task g from S to POS that guarantees sufficient performance.*

Def. 6 describes a task commonly used as a proxy for a representation's capabilities to represent a word's role in its context: an example is shown in Figure 6.3. These two tasks allow us to inspect how a language model identifies and semantically links entities in a sentence, thus giving us a comprehensive, linguistically informed perspective on what a language model captures.

We complete the benchmark with two further syntactic tasks, namely root identification and the tree-depth estimation, which we present and comment on below.



Index	Word	Relationship	POS-tag
1	Thousands	3	NNS
2	of	3	IN
3	demonstrators	4	NNS
4	have	5	VBP
5	marched	-	VBN

Figure 6.3: A sentence with its POS tags (left). A sentence in CONLL format tests a model on multiple syntactic tasks (right). CONLL is a convenient format as it provides sentences and information that can be used to instantiate syntactic tasks such as tree reconstruction.

Definition 7. *(Root Identification) Given S and T as in Def. 5, and the root of the tree $R = \{r^{(1)}, .., r^{(n)}\}$ where $r^{(i)} \in t^{(i)}$, root identification is a probing task h from S to R that guarantees sufficient performance.*

Definition 8. *(Tree-depth Estimation) Given S and T as in Def. 5, and the depth $D = \{d^{(1)}, .., d^{(n)}\}$ of dependency trees in T, where $d^{(i)} \in \mathbb{N}^+$,* tree-depth estimation *is a probing task u from S to D that guarantees sufficient performance.*

With the tasks in Def. 7 and 8, we assess a representation's capacity to distill single units of information (root and depth), which can be extracted from a tree's sentence representation. We sketch the two tasks in Figure 6.1 (right). When compared to the structural probe task, root identification and tree depth are easier to solve. They are meant to show to what extent high vs. low-order syntax information is encoded in a linguistic representation.

6.3 Measuring Syntactic Robustness

This section introduces the perturbation methods used to define syntactic robustness. We propose a perturbation method, and an algorithm, to compute perturbations that largely preserve the syntax of the original sentence and another method that instead proposes context-based substitutions, which tries to preserve a sentence's original intent while possibly disrupting syntax.

6.3.1 Syntax-preserving Perturbation Analysis

The second phase of our methodology involves perturbation-based analysis. It is widely known and confirmed by neuroscience that human language exhibits very robust linguistic representations (Chomsky, 2009; Gibson et al., 2019). At the same

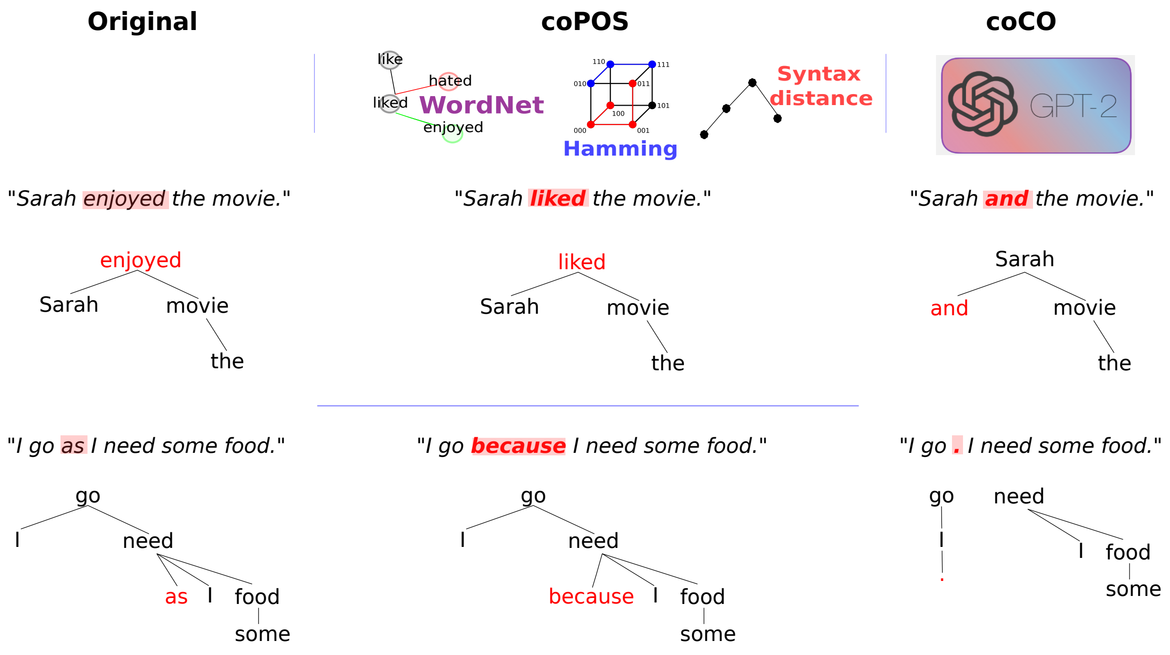


Figure 6.4: Two examples of coPOS and coCO perturbations applied on clean input texts and the resulting syntax trees induced by such alterations. Words perturbed are highlighted in red. coPOS perturbations are designed to minimize the probability of disrupting the syntax of a sentence (such as the substitution of `‘as’` with `‘because’`). In contrast, coCO can disrupt it (e.g., the substitution of `‘and’` with a period).

time, NLP models suffer from brittleness against perturbations, which are often easily transferable across models yet difficult to detect (Kuleshov et al., 2018). Though many works have shown how brittle NLP models are in the presence of bounded attacks on embedding space (see Chapter 4), such attacks do not necessarily preserve human meaning. They are, therefore, arguably of questionable merit (see Chapter 5). We define two types of perturbations: the first aims to preserve syntax (coPOS) and constitutes the backbone of our empirical evaluation. The second exploits context to preserve the semantics (coCO) and is introduced to strengthen our comparison of models' syntactic robustness. As a baseline, we add a perturbation method with words randomly sampled from the English vocabulary. We now introduce the coPOS and the coCO perturbation methods, illustrated in Figure 6.4.

Definition 9. *(Consistent POS Substitution) A* consistent POS substitution (coPOS) *consists of the replacement of one or more words in a sentence s with words that keep unaltered the POS-tag of the perturbed sentence, i.e., if s/s' are the original/perturbed sentence, pos/pos' the ground-truth POS-tag of s/s', and $s' = sub(s)$, the perturbation procedure, then it holds for coPOS that $sub(s) \implies pos \equiv pos'$.*

Ensuring that a perturbation satisfies the coPOS definition enables the interpretability of our results. Specifically, a coPOS perturbation aims to preserve the word's syntactic role in a sentence. Therefore one can impute any probe misclassifications to a lack of robustness of the linguistic representation. Since guaranteeing that a perturbation always preserves its coPOS tag is challenging due to the intrinsic complexity of natural language, we rely on an efficient algorithmic implementation to generate proxy coPOS perturbations, described in Section 6.4, which we carefully validate on the datasets used in our experimental evaluation (see Section 6.5).

Definition 10. *(Context Consistent Substitution) A* context consistent substitution (coCO) *consists of the replacement of one or more words in a sentence s with a generative model that maintains semantic closeness but does not strictly enforce the substitution to be syntactically coherent. While many alternative methods exist in the literature to generate coCo perturbations, we rely on GPT-2 (Radford et al., 2019) next-word predictions, which serve as a benchmark for syntactically-informed methods such as coPOS. In other words, a substitution w' of a word $w \in s$ is generated by a generative model ϕ conditioned on the context where the word appears, i.e., $w' = argmax_{w \in V} \phi(s|s \setminus w)$.*

Initially, our experiments involved using BERT as the generative process for creating coCO perturbations. The robustness results, which can be found in Section 6 of this paper indicated that all models were susceptible and, in fact, brittle to these perturbations. We then realized that using BERT might have biased the results, as BERT itself was being evaluated for robustness. Consequently, we replaced BERT with GPT-2 as the generative model for the coCO perturbations. Section 6.5 of this thesis shows that GPT-2 maintains good generative quality, particularly in terms of syntactic preservation. However, we did not explore any bidirectional models other than BERT, which could be considered a limitation of our approach. This aspect presents an opportunity for future research.

Below, we formally define the conditions under which we consider a linguistic model robust: informally, for a linguistic representation to be robust, we desire it to accurately solve a family of probing tasks and behave consistently on slight syntax-preserving perturbations of an input text. We assume that coPOS substitutions are used as perturbations but note that the concept of linguistic robustness can also be instantiated with Def. 10.

First, we introduce the notion of consistency of representations, termed ϵ-robustness.

Definition 11. *(ϵ-robust Representations) Given a linguistic representation ψ^θ, a set of sentences S, a set of perturbed sentences S' which are coPOS perturbations of S, and a measure of distance dist : $(s, s') \rightarrow \mathbb{R}$ between representations (e.g., ℓ_p-norm, cosine similarity), we say that the representation ψ^θ is* ϵ-robust w.r.t. *dist if* $\forall (s, s') \in (S, S'),\ max(dist(\psi^\theta(s), \psi^\theta(s'))) < \epsilon$.

Despite its simplicity, ϵ-robustness is linguistically informed, as all sentences in S' are coPOS to those in S, and thus we can be confident that the perturbations are syntactically consistent for the given probing task. Moreover, this metric can serve as a valuable tool for developing robust language models in the sense of maximizing ϵ while maintaining good performance on the underlying task.

While ϵ-robust representations are desirable, what is more informative is the ability for a representation, ψ^θ, to be robust concerning a distance metric and a probing task. Formally, we define a language model ψ^θ to be *syntactically robust* if the performance on multiple proxy tasks is not adversely affected by perturbations that are close in some representation space (e.g., Def. 9).

Definition 12. *(Syntactically Robust Representation) Given an input s, its representation $\psi^\theta(s)$, a set of probing tasks $\{\mathbf{T_1}, .., \mathbf{T_m}\}$, a set of mappings $\{f_1(s), .., f_m(s)\}$ that take as input the representation $\psi^\theta(s)$ and solve the respective i-th probing task, a set of strictly positive quantities $\{\tau_1, .., \tau_m\}$ and a small quantity $\epsilon > 0$, a set of measures of performance on each task $\{\mathcal{L}_1, .., \mathcal{L}_m\}$, a consistent perturbation $s' = sub(s)$, and a measure of distance between representations dist : $(s, s') \rightarrow \mathbb{R}$, ψ^θ is* syntactically robust *iff* $\forall (\mathbf{T}_i, f_i, \mathcal{L}_i, \tau_i) \in (\mathbf{T}, f, \mathcal{L}, \tau),\ dist(\psi^\theta(s), \psi^\theta(s')) < \epsilon \implies \mathcal{L}_i(f_i(\psi^\theta(s)), f_i(\psi^\theta(s'))) < \tau_i$.

6.4 Algorithm for Evaluating Syntactic Robustness

In this section, we describe a procedure to assess the robustness of the syntactic structures encoded by a linguistic representation, as formalized in Def. 12. We outline the complete algorithm and give step-by-step comments. Before that, we provide details about the perturbation methods, particularly how candidate perturbations for the coPOS, the coCO, and the baseline methods are extracted.

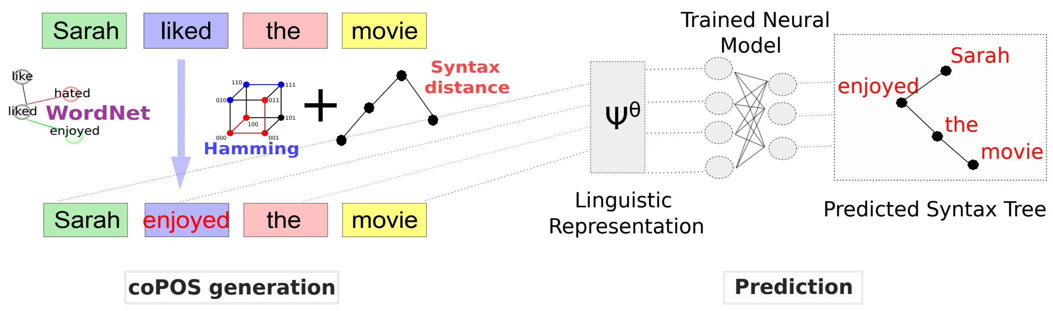


Figure 6.5: An example of a perturbed sentence s' obtained through a coPOS perturbation. Candidate substitutions are sampled from a pool of WordNet synonyms, from which we select the one that maximizes the Hamming distance and minimizes the syntactic disruption w.r.t. the original input; see Section 6.4 for details. The perturbation is then fed, through a linguistic representation ψ^{θ}, to a probe (neural network trained directly on the representation) that predicts its syntax tree.

6.4.1 Computing coPOS Perturbations

Given an $l > 0$ word long sentence s, we formulate a method to obtain a perturbed sentence s', where $\tau \leq l$ words in s are replaced while keeping the syntax of the original input largely preserved. WordNet synonyms (obtained by confining the substitutions to ones that are in the same 'Synset'), or any similar technique, are specifically crafted to maintain the syntactic structure of word replacements. However, it is essential to acknowledge that no technique can offer a guarantee of preserving syntactic integrity when replacing a word in a sentence with a generic alternative. Our procedure is sketched in Algorithm 3.

We replace each candidate word in s with one drawn from the WordNet synonym graph (Miller, 1998). We further ensure that a perturbation is, among the input-perturbation pairs generated by a WordNet replacement, the one that minimizes the syntactic distance of the tree representations while maximizing the Hamming distance between the actual sentences, i.e., the number of words that are actually perturbed. The syntactic distance of each pair of inputs and perturbations is computed via the Stanza dependency parser (Qi et al., 2020). In contrast, the Hamming distance between two sentences is the number of word positions in which two words differ.

In practice, for each input, a constant number of sentences $b > 0$ are generated by perturbing τ words via WordNet (line 3): the syntactic distance between the dependency tree of each input/perturbation pair is computed alongside their Hamming distance (line 6), which could be less than τ if WordNet does not return a viable substitution and only the sentence that minimizes the syntactic distance while maximizing the Hamming is used to test the representation's robustness.

While this procedure is designed to preserve syntax between s and s', semantics, in general, is not: though one may want to introduce further constraints on the replacement procedure to ensure the semantics is preserved, our primary intent is to assess robustness against syntax manipulations. We will show in the experiments that, even for these simple proxy syntax-preserving perturbations, a linguistic representation's performance degrades significantly, and in some cases, it is comparable with random guessing, which indicates that this perturbation scheme is powerful enough to benchmark current language models. Further, this method has a clear advantage in simplicity and computational efficiency as multiple word substitutions can be parallelized. We sketch the procedure above in Figure 6.5 (left).

Algorithm 3 coPOS perturbations.

```
Require: $s, b, \tau, \text{WordNet}(\cdot,\cdot), \text{dist}_{\text{ham}}(\cdot,\cdot), \text{dist}_{\text{syntax}}(\cdot,\cdot)$
Ensure: A coPOS perturbation.
 1: $s^*, d_h^*, d_t^* \leftarrow (s, 0., \text{inf})$
 2: for $j \in [1, .., b]$ do
 3:     $s' \leftarrow \text{WordNet}(s, \tau)$    ▷ Perturb $\tau$ random words in $s$ with synonyms
 4:     $d_h \leftarrow \text{dist}_{\text{ham}}(s, s')$
 5:     $d_t \leftarrow \text{dist}_{\text{syntax}}(s, s')$
 6:     if $d_h > d_h^* \wedge d_t < d_t^*$ then
 7:         $s^*, d_h^*, d_t^* \leftarrow (s', d_h, d_t)$
 8:     end if
 9: end for
10: return $s^*$
```

6.4.2 coCO and Baseline Perturbations

Our results are complemented by experiments with coCo perturbations (Def. 10), which consist of generating τ replacements via a conditioned LLM. While we employ GPT-2 (Radford et al., 2019) for generating a replacement, any generative LLM, thus including masked LLMs such as BERT or RoBERTa (Devlin et al., 2019a), is suitable for this task. The process of coCO perturbation is sketched in Figure 6.4.

Finally, we add a baseline perturbation method that involves substituting $\tau > 0$ words in a sentence with random replacements from the English vocabulary. In this case, the syntactic consistency of a sentence is not guaranteed to be maintained and thus serves as a base case for our analysis.

6.4.3 Average Worst-case Robustness Algorithm

In this section, we describe a procedure to assess the robustness of the syntactic structures encoded by a linguistic representation, as formalized in Def. 12. The general framework requires several inputs, including a language model ψ, a collection of m probing tasks $\{\mathbf{T}_i\}_{i=1}^m$ and performance metrics $\{\mathcal{L}_i\}_{i=1}^m$, a perturbation function *sub*, which w.l.o.g. can be a coPOS, a coCO or any other perturbation method, and three constants, τ and k and b, which control the number of perturbations generated per-input point. For each task $\mathbf{T}$, we collect a number $n > 0$ of input sentences and apply k coPOS or coCO perturbations to each of them, with each perturbation targeting τ words and with an overall budget per-sentence equal to b (see Alg. 3). We then employ the task's corresponding performance function to measure $\mathcal{L}(f(\psi(s)), f(\psi(s')))$, where s' is the sentence that maximizes this quantity and is regarded as the approximate worst-case perturbation. We also record the corresponding drop in performance that s' causes. Finally, we calculate the average performance decrease across all n sentences as an approximation of the worst-case performance of the language model on the given probing task. In line with standard works on adversarial robustness, our framework discovers the cases where a model failure is the most severe, yet our approach allows us to conduct experiments and return the average-case robustness. In this sense, the average-case robustness measure is interpretable as the syntactic robustness of a model to a pool of samples from a not-necessarily adversarial set and reflects a model's sensitivity rather than its robustness.

We propose a technique for generating a modified sentence s' from a sentence s of length $l > 0$, where up to τ words in s are replaced while preserving the original sentence's syntax. We substitute each word in s with a synonym drawn from the WordNet synonym graph (Miller, 1998). Although this approach guarantees that the syntax of s' remains essentially unchanged from s, the semantics of the sentence may not be preserved. While it is possible to introduce additional constraints to maintain semantic coherence, our primary goal is to assess the models' robustness against syntax manipulations. Our experiments indicate that even these basic syntax-preserving substitutions result in a severe drop in performance, comparable to random guessing, demonstrating this perturbation scheme's power as a benchmark for current language models. Additionally, this method is simple and computationally efficient since multiple-word substitutions can be parallelized. The procedure is outlined in Figure 6.5 (left), and we also conduct experiments with coCo perturbations (Def. 10), which involve generating replacements using a conditioned LLM.

Average distance of the farthest representation. As a measure of distance between sentences, we rely on the ℓ_2-norm and the cosine similarity, whose usage is widespread in NLP robustness (Huang et al., 2019; Jia et al., 2019), noting that other measures are also possible (Dong et al., 2021; Kusner et al., 2015; La Malfa et al., 2021). We provide a sketch of the procedure in Alg. 4. It estimates the average distance of the farthest perturbation in the representation space of ψ^θ, w.r.t. a distance measured like an ℓ_p norm. The procedure easily accounts for similarity measures, like the cosine similarity, by substituting *max* with *min* at line 9.

Average worst-case syntactic robustness. Complementary to Algorithm 4, Algorithm 5 permits us to evaluate *syntax robustness* of a linguistic representation ψ^θ. For each pair of a model and a probing task (f_i, $\mathbf{T_i}$), we draw a pool of sentences S_i from a CONLL corpus of choice (lines 2, 4): CONLL is a convenient text format as it provides sentences along with information that can be used to instantiate probing tasks such as tree reconstruction (see Figure 6.3). Then, for each sentence $s \in S_i$, we compute a set of coPOS perturbations (lines 7, 8). The ratio between the number of sentences in S_i and the perturbations depends on the budget parameter k, as well as the number of words per sentence τ that are perturbed; e.g., with $\tau = k = 1$, each sentence in S_i is perturbed once via a single-word substitution. We rely on WordNet (Miller, 1998) and its graph of synonyms to draw, for each sentence s, a substitute s' that is drawn accordingly to Alg. 3 and aims to be syntax-preserving. We exemplify this process in Figure 6.5. We then quantify the drop of performances of f_i on the original vs. perturbed input representations via the performance measure $\mathcal{L}_i$ (line 10). As we aim for a measure of robustness against perturbations, we return for each sentence $s \in S_i$ the worst-case drop induced by any of the s' generated previously, averaged over the number of test cases (lines 11, 13 and 15). We can now pair the measure of robustness with the ϵ-distance between S and the set of worst-case perturbations S' (Def. 11) as the most significant deviation of a pair of input/perturbation w.r.t. the representation ψ^θ.

6.5 Validating the Perturbation Method

In this section, we report the results of the validation process of the coPOS and the coCO perturbation methods. For each perturbation method, and for each dataset that we then employ in the experimental evaluation, we calculate the syntactic distance between the syntax tree of a sentence and a perturbed candidate: the distance between trees is automatically computed as that minimum number of operations of addition and deletion of a node to turn a tree into another, via Stanza (Qi et al., 2020) dependency parses. We report the results regarding the distance between dependency

Algorithm 4 Estimate the average distance of the farthest perturbations w.r.t. a representation ψ.

```
Require: $\psi(\cdot), S, sub(\cdot), \tau, k, b, dist(\cdot,\cdot)$
Ensure: Average distance of the farthest representation of $\psi_{s\sim S}(s)$ against $sub_{s'\sim S'}(s')$
 1: $rob = 0.$
 2: for $s \in S$ do
 3:     $x \leftarrow \psi(s)$
 4:     $worst = 0$
 5:     for $j$ in $[1, .., k]$ do
 6:         $s' \leftarrow sub(s, \tau, b)$
 7:         $x' \leftarrow \psi(s')$          ▷ Obtain the representation of a perturbed input
 8:         $d = dist(x, x')$     ▷ Calculate the distance between input and perturbation
 9:         $worst = max(worst, d)$          ▷ Worst-case as farthest perturbation
10:     end for
11:     $rob \mathrel{+}= worst$
12: end for
13: return $rob/|S|$                          ▷ Average over each worst-case.
```

Algorithm 5 Estimate the average worst-drop of robustness of ψ on probing tasks $\mathbf{T}$.

```
Require: $\psi(\cdot), \{\mathbf{T_1}, .., \mathbf{T_m}\}, \{f_1(s), .., f_m(s)\}, \{\mathcal{L}_1, .., \mathcal{L}_m\}, sub(\cdot), \tau, k, b$
Ensure: Average worst-drop of robustness of $\psi_{s\sim S}(s)$ against $sub_{s'\sim S'}(s')$ on each task $\{\mathbf{T_1}, .., \mathbf{T_m}\}$
 1: $Drop = \{\}$               ▷ Will contain the average worst-case drop per task $\mathbf{T}_i$
 2: for $i \in [1, .., m]$ do
 3:     $drop = 0.$                       ▷ Average worst-case drop of robustness
 4:     $S_i \leftarrow data(\mathbf{T}_i)$                          ▷ Get data from each task
 5:     for $s \in S_i$ do
 6:         $d = 0.$
 7:         for $j$ in $[1, .., k]$ do
 8:             $s' \leftarrow sub(s, \tau, b)$        ▷ $\tau$ words are perturbed to obtain $s'$ from $s$
 9:             $x, x' \leftarrow \psi(s), \psi(s')$                    ▷ Input/perturbation pairs
10:             $\Delta d = \mathcal{L}_i(f_i(x), f_i(x'))$                   ▷ Drop of robustness
11:             $d = max(d, \Delta d)$     ▷ Get the case that minimizes syntax robustness
12:         end for
13:         $drop \mathrel{+}= d$
14:     end for
15:     $Drop \overset{+}{\leftarrow} drop/|S_i|$
16: end for
17: return $Drop$
```

trees, as that is the representation provided by the CoNLL format and that employed in (Manning et al., 2020). However, our methodology also permits us to compute the distance between sentences via their constituency representation.

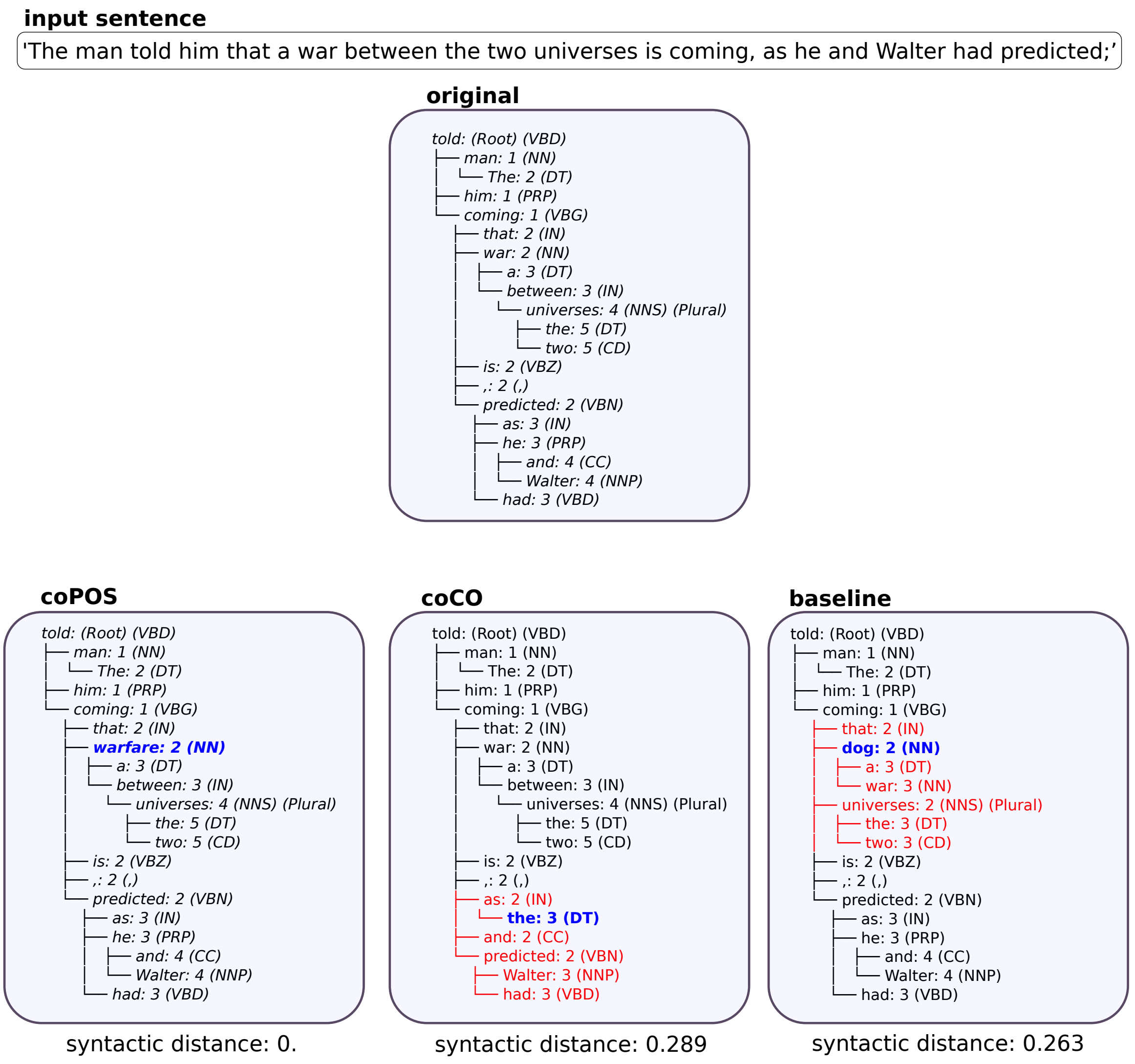
Examples of the perturbed syntax tree of a sentence from the Ud-English-Pud dataset are shown in Figure 6.6 for each perturbation method. In Figure 6.7, we report, for each of the six datasets used in the experimental evaluation, the syntactic distance between trees, for the coPOS, coCO, and baseline method, with varying perturbation budget τ equal to 1, 2, and 3. We further compute the average distance between pairs of sentences randomly sampled from each dataset, for which we expect the distance between trees to be higher than for any other method.

As one can notice from Figure 6.7, the coPOS method induces the slightest changes in a syntax tree, and it is thus expected to disrupt the performance of the probing tasks only if the representations are inherently brittle. On the other hand, both coCO and baseline are expected to challenge a probe's capacity to represent a sentence's syntactic information correctly. We will show with the proxy coPOS method that probes, and their representations, are very brittle to syntax-preserving manipulations.

Finally, we show examples of perturbations that our methods produce. In Figure 6.8, we report example sentences that induce, according to Stanza (Qi et al., 2020), a high degree of disruption in the dependency syntax tree representation, with those produced by the coCO perturbation method the most disrupted for each input/perturbation pair (not counting the baseline, which almost indeed disrupts the syntactic tree of a sentence through random replacements). On the other hand, the coPOS perturbation method can be seen to preserve the structure of each sentence. In Figure 6.9, we present linguistically interesting perturbations that do not induce the maximum syntactic disruption.

6.6 Experimental Evaluation

We implement and empirically validate our framework by demonstrating how it can provide insights into the robustness of language models. We start with details on linguistic representations, datasets, and probing task models. We then discuss the effect of latent feature depth and the duration of fine-tuning. Finally, we summarise the results of applying our framework in these settings.



input sentence

'The man told him that a war between the two universes is coming, as he and Walter had predicted;'

original

```
told: (Root) (VBD)
├── man: 1 (NN)
│   └── The: 2 (DT)
├── him: 1 (PRP)
└── coming: 1 (VBG)
    ├── that: 2 (IN)
    ├── war: 2 (NN)
    │   ├── a: 3 (DT)
    │   └── between: 3 (IN)
    │       └── universes: 4 (NNS) (Plural)
    │           ├── the: 5 (DT)
    │           └── two: 5 (CD)
    ├── is: 2 (VBZ)
    ├── ,: 2 (,)
    └── predicted: 2 (VBN)
        ├── as: 3 (IN)
        ├── he: 3 (PRP)
        │   ├── and: 4 (CC)
        │   └── Walter: 4 (NNP)
        └── had: 3 (VBD)
```

coPOS

```
told: (Root) (VBD)
├── man: 1 (NN)
│   └── The: 2 (DT)
├── him: 1 (PRP)
└── coming: 1 (VBG)
    ├── that: 2 (IN)
    ├── warfare: 2 (NN)
    │   ├── a: 3 (DT)
    │   └── between: 3 (IN)
    │       └── universes: 4 (NNS) (Plural)
    │           ├── the: 5 (DT)
    │           └── two: 5 (CD)
    ├── is: 2 (VBZ)
    ├── ,: 2 (,)
    └── predicted: 2 (VBN)
        ├── as: 3 (IN)
        ├── he: 3 (PRP)
        │   ├── and: 4 (CC)
        │   └── Walter: 4 (NNP)
        └── had: 3 (VBD)
```

syntactic distance: 0.

coCO

```
told: (Root) (VBD)
├── man: 1 (NN)
│   └── The: 2 (DT)
├── him: 1 (PRP)
└── coming: 1 (VBG)
    ├── that: 2 (IN)
    ├── war: 2 (NN)
    │   ├── a: 3 (DT)
    │   └── between: 3 (IN)
    │       └── universes: 4 (NNS) (Plural)
    │           ├── the: 5 (DT)
    │           └── two: 5 (CD)
    ├── is: 2 (VBZ)
    ├── ,: 2 (,)
    ├── as: 2 (IN)
    │   └── the: 3 (DT)
    ├── and: 2 (CC)
    └── predicted: 2 (VBN)
        ├── Walter: 3 (NNP)
        └── had: 3 (VBD)
```

syntactic distance: 0.289

baseline

```
told: (Root) (VBD)
├── man: 1 (NN)
│   └── The: 2 (DT)
├── him: 1 (PRP)
└── coming: 1 (VBG)
    ├── that: 2 (IN)
    ├── dog: 2 (NN)
    │   ├── a: 3 (DT)
    │   └── war: 3 (NN)
    ├── universes: 2 (NNS) (Plural)
    │   ├── the: 3 (DT)
    │   └── two: 3 (CD)
    ├── is: 2 (VBZ)
    ├── ,: 2 (,)
    └── predicted: 2 (VBN)
        ├── as: 3 (IN)
        ├── he: 3 (PRP)
        │   ├── and: 4 (CC)
        │   └── Walter: 4 (NNP)
        └── had: 3 (VBD)
```

syntactic distance: 0.263

Figure 6.6: Comparison of the disruption induced on the dependency syntax tree by different perturbation methods and the syntactic distance between trees. The representation of each dependency syntax tree has been compacted to make the effect of the perturbation methods clear. Yet, it is equivalent to that of Figures 6.2 and 6.4. The example sentence belongs to the Ud-English-Pud dataset, and the perturbations are actual perturbations induced by our methods. In blue, the single word that has been perturbed, while in red, the perturbation induced by such perturbation on the tree.

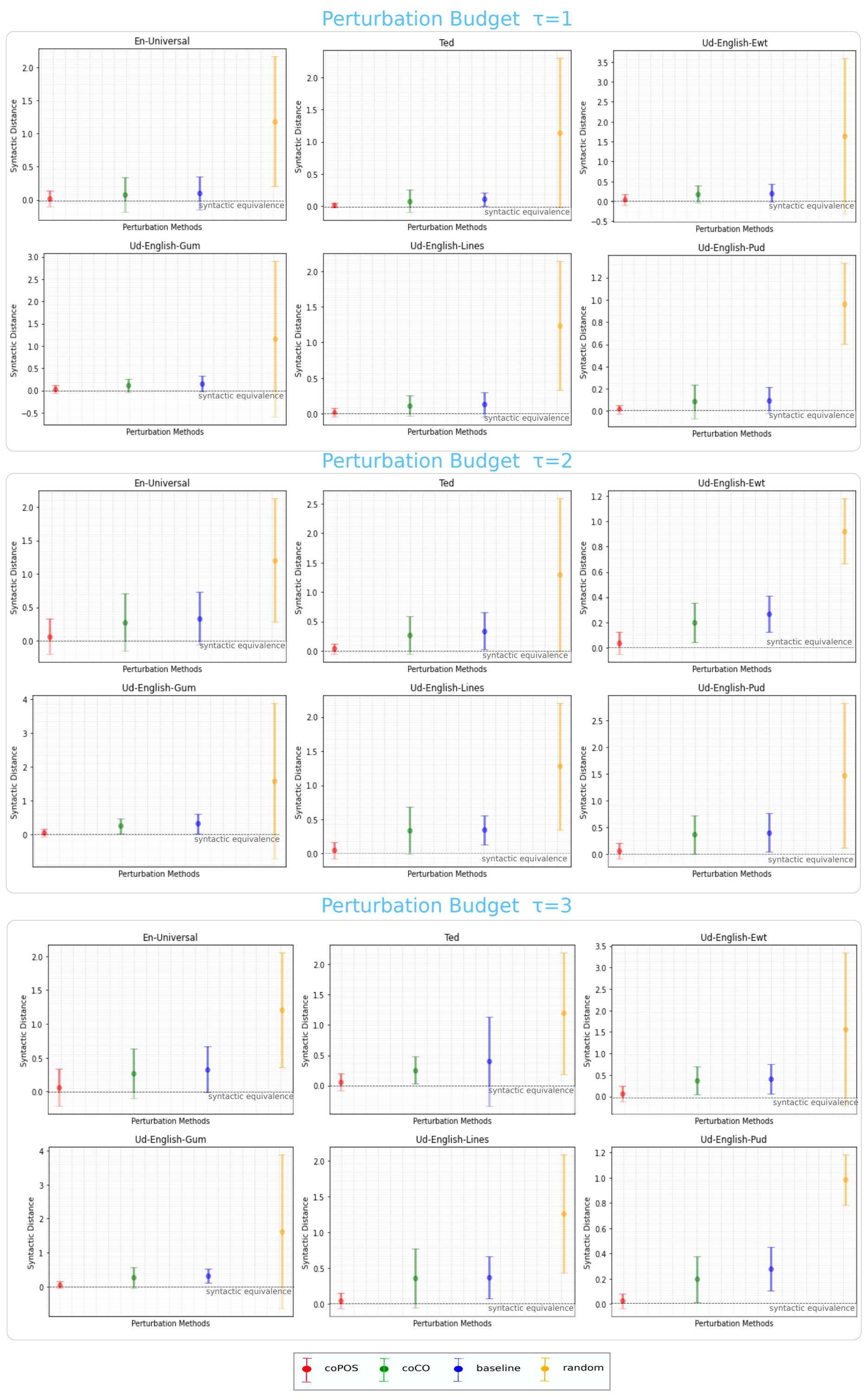


Figure 6.7: Tree distance, measured with Stanza, between an input and its perturbed version, for different datasets and perturbation budgets: results are averaged over the entire dataset. The coPOS perturbation method (red) induces almost no disruption to a perturbation's syntax tree, being always close to the level of syntactic equivalence, while injection of random words (blue) and coPOS perturbations (green) both induce some noticeable disruption. The disruption induced by comparing the syntax tree of two randomly picked-up sentences that belong to the same dataset is reported for further comparison (orange).

Ted

[src]	*And those are usually the basic scientists , The bottom is usually the surgeons .*	
[coCO]	*And those are usually the ones that , The bottom line usually the surgeons .*	(0.875)
[coPOS]	*And those are usually the basic scientist , The bottom is usually the surgeons .*	(0.875)

UD-English-Ewt

[src]	*Drs. Ali work wonders .*	
[coCO]	*Drs. 🟥 work in the*	(2.5)
[coPOS]	*Drs. Ali oeuvre wonders .*	(2.0)

Ud-English-Gum

[src]	*Environment Canada spokeswoman Sujata Raisinghani told CBC News the department will look into the incident .*	
[coCO]	*🟥 Canada 🟥, Sujata Raisinghani told CBC News the agency will be into the incident .*	(1.428)
[coPOS]	*surround Canada spokeswoman Sujata Raisinghani told CBC News the department will look into the incident .*	(0.714)

Un-English-Lines

[src]	*Break with a Banshee by Gilderoy Lockhart*	
[coCO]	*Break with the best by the Lockhart*	(0.428)
[coPOS]	*prison-breaking with a banshie by Gilderoy Lockhart*	(0.714)

Un-English-Pud

[src]	*However , they were intercepted and had to do battle in Freeman , close to the Hudson River .*	
[coCO]	*However , they were able and had to do battle with Freeman , close to the Hudson River .*	(0.266)
[coPOS]	*However , they were intercepted and had to do struggle in Freeman , finis to the Hudson River .*	(0.333)

En-Universal

[src]	*Manufacturers Hanover had a loss due to a big reserve addition .*	
[coCO]	*Manufacturers Hanover , a loss due to a big reserve of .*	(11.0)
[coPOS]	*producer Hanover had a loss due to a big taciturnity addition .*	(10.0)

Figure 6.8: Examples of sentences and worst-case coCO and coPOS perturbations that are reported in our experiments to highly disrupt the dependency syntax tree according to Stanza (Qi et al., 2020) (the syntactic distance between the original and perturbed sentence is shown on the right). We show the original sentence for each of the 6 CoNLL datasets. For coCO, perturbed words are highlighted in red, and replacements with empty words (allowed from the vocabulary) are denoted with a red rectangle 🟥. For coPOS, perturbed words are highlighted in blue. Results refer to the perturbation regime with $\tau = 3$, i.e., where at most three words per sentence are perturbed.

UD-English-Ewt

[src]	*I loved the atmosphere here and the food is good , however the tables are so close together .*	
[coCO]	*I loved the atmosphere here . the food is good . however the tables are so close together .*	(0.684)
[coPOS]	*I loved the aura here and the nutrient is good , however the tables are so close together .*	(0.0)

Ud-English-Gum

[src]	*The purple spheres represent atoms of another element .*	
[coCO]	*The purple and gold atoms are another element .*	(0.666)
[coPOS]	*The purple spheres represent atoms of another constituent .*	(0.0)

Un-English-Lines

[src]	*Can't anyone help you ?*	
[coCO]	*Can't anyone else you know*	(1.0)
[coPOS]	*Can't anyone service you ?*	(0.0)

Un-English-Pud

[src]	*Meanwhile , his place in tribune was occupied by Marco Antonio , who held the position until December .*	
[coCO]	*Meanwhile , his wife in tribune , occupied by Marco Antonio , who held the position until December .*	(0.473)
[coPOS]	*Meanwhile , the place in tribune was occupied by the Antonio , who held the position until dec. .*	(0.157)

En-Universal

[src]	*But the report says : The only way sex is sex between uninfected partners .*	
[coCO]	*But the report says that The only way sex is sex between uninfected partners .*	(2.0)
[coPOS]	*But the reputation says : The only safe sex is sex between uninfected partners .*	(0.0)

Ted

[src]	*A windpipe cell already knows it 's a windpipe cell .*	
[coCO]	*A windpipe is a knows it 's a windpipe cell .*	(0.272)
[coPOS]	*A trachea cubicle already knows it 's a windpipe cubicle .*	(0.0)

Figure 6.9: Examples of linguistically interesting sentences, perturbations, and their syntactic distances (right), as calculated with Stanza (Qi et al., 2020). We report the original sentence on top for each of the six CoNLL datasets. For coCO, perturbed words are highlighted in red, while for coPOS, in blue. Results refer to the perturbation regime with $\tau = 3$, i.e., where at most three words per sentence are perturbed.

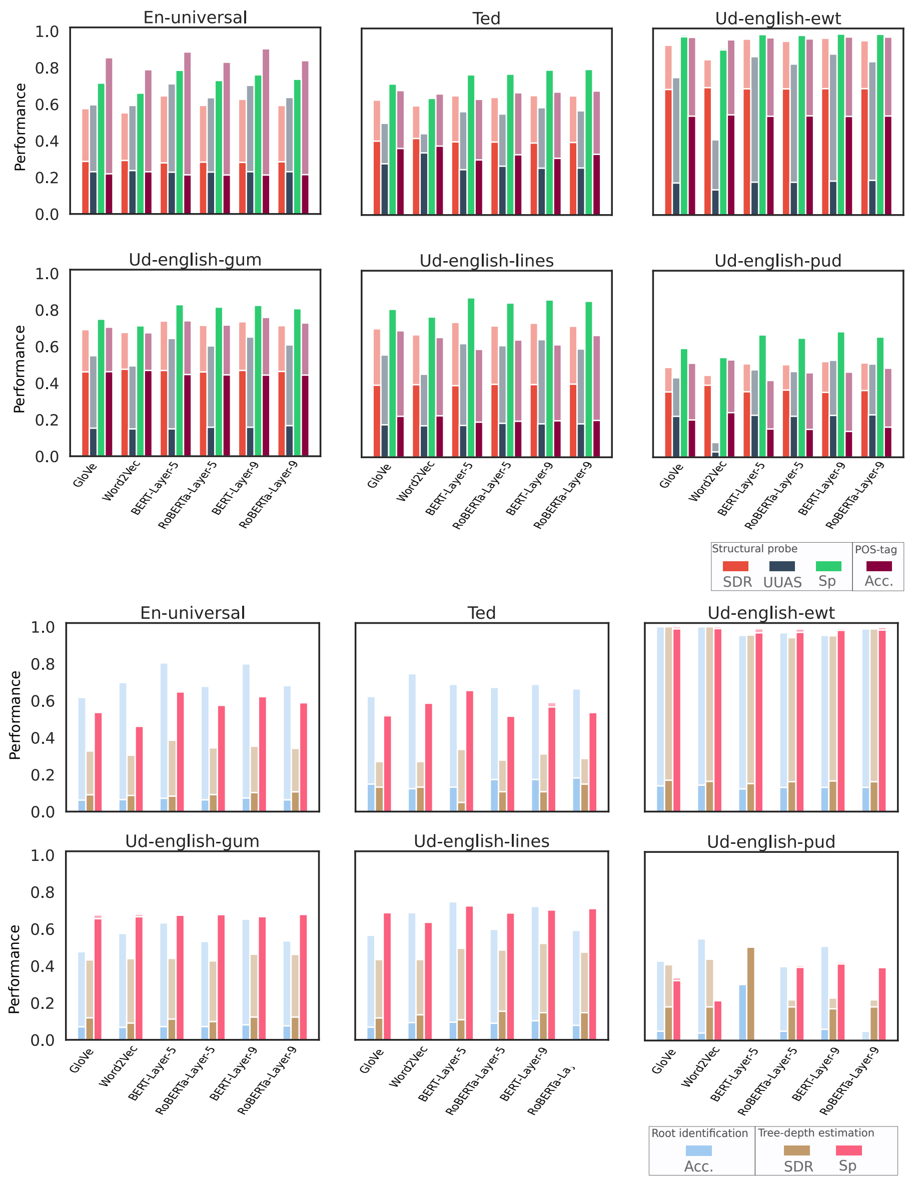


Figure 6.10: Top two rows: performance of different linguistic representations on *syntax reconstruction* and *POS-tagging* probing tasks. Bottom two rows: performance on root identification (accuracy metric) and tree-depth estimation (SDR and Spearman metric) probing tasks. For both plots, the performance of the probing tasks is reported as shaded bars, with the performance for the perturbed representation shown as a solid overlapping bar: the results refer to the case where the coPOS perturbation budget τ is equal to 3. We distilled BERT and RoBERTa's representations from the 5^{th} and 9^{th} hidden layers, resulting in the highest performance on both tasks.

6.6.1 Experimental Setting

Datasets and metrics. We assess the *syntactic robustness* of different language models using *syntax reconstruction*, *POS-tagging*, *root identification* and *tree-depth estimation* on six datasets from the Universal Dependencies collection (Nivre et al., 2016). We chose datasets standardized according to the CONLL format (Hajič et al., 2009), consisting of sentences split into words, each indexed and annotated with multiple syntactic information such as POS-tags and the relationship with other words or tokens. Figure 6.3 reports an example of a sentence in CONLL format: relationship tags between words and part-of-speech tags produce the ground truth labels for our probing tasks.

We measure the performance on syntax reconstruction in terms of the 'undirected unlabeled attachment score' (UUAS), i.e., the fraction of edges in the ground truth syntax tree that is correctly predicted by a model, the 'same distance ratio' (SDR), i.e., the number of times a model correctly guesses the distance between each pair of words in the ground truth syntax tree, and the Spearman correlation (used in (Manning et al., 2020)), which summarizes the strength of the relationship between the matrix representation of the original vs. reconstructed syntax tree. Regarding POS-tagging, we evaluate a model in terms of the accuracy of estimating the correct POS-tag as shown in Figure 6.10 (top). On root identification (see Def. 7) we use the accuracy of correct vs. wrong estimates, while on tree-depth estimation (Def. 8), we use the SDR, along with the Spearman correlation, so as not to penalize too many models that do not infer the label precisely.

Syntactic robustness (Def. 12) is measured in terms of the drop in the performance of a model when targeted with a coPOS or a coCO perturbation, e.g., ΔUUAS represents the drop of the UUAS metric on the syntax robustness task, comparing the performance of the original sentence with its perturbed version. Regarding POS tags, we convert the drop of accuracy into the more intuitive difference between words correctly guessed on the original sentence compared to its perturbed version, denoted '# Words Adv' in Tables 6.1, 6.2, and 6.3.

Models and probing tasks. We perform our analyses on four linguistic representations, of which two are context-free, namely GloVe (Pennington et al., 2014b) and Word2Vec (Mikolov et al., 2013), and two context-dependent, namely, BERT (Devlin et al., 2019a) and RoBERTa (Liu et al., 2019). As LLMs employ deep attention-based architectures, we perform experiments on sentences distilled from the -5^{th} (i.e., the

6.1.1 Robustness

	Syntax Reconstruction			POS-tagging
	ΔSDR	**ΔUUAS**	**ΔSp**	**ΔAcc. (# words)**
GloVe	0.2066 ± 0.1243	0.2444 ± 0.1298	0.0062 ± 0.0032	6.4897 ± 3.0856
Word2Vec	0.1553 ± 0.1161	0.1145 ± 0.1004	0.0052 ± 0.0048	6.831 ± 2.2698
BERT layer -1	0.2011 ± 0.0784	0.1607 ± 0.0835	0.0032 ± 0.0077	3.0803 ± 2.9857
RoBERTa layer -1	0.193 ± 0.0808	0.1752 ± 0.1117	0.0019 ± 0.0039	4.1293 ± 3.3021
BERT layer -5	0.2287 ± 0.0817	0.2451 ± 0.1212	0.005 ± 0.003	3.243 ± 3.0814
RoBERTa layer -5	0.2038 ± 0.0758	0.2086 ± 0.1052	0.0024 ± 0.0379	3.0607 ± 3.0132
BERT layer -9	0.2307 ± 0.0838	0.281 ± 0.1142	0.0035 ± 0.0037	3.544 ± 3.2826
RoBERTa layer -9	0.2045 ± 0.0763	0.2148 ± 0.1116	0.0017 ± 0.0023	3.4723 ± 3.1265

6.1.2 Robustness

	Root Identification	Tree Depth Estimation	
	ΔAcc.	**ΔAcc.**	**ΔSp**
GloVe	0.4387 ± 0.1953	0.2663 ± 0.235	0.0174 ± 0.0189
Word2Vec	0.5251 ± 0.1123	0.2582 ± 0.2712	0.0093 ± 0.0285
BERT layer -1	0.5015 ± 0.2135	0.3495 ± 0.2139	0.0209 ± 0.0159
RoBERTa layer -1	0.5286 ± 0.1661	0.3193 ± 0.2348	0.0261 ± 0.0126
BERT layer -5	0.6039 ± 0.1383	0.3314 ± 0.2401	0.0086 ± 0.0158
RoBERTa layer -5	0.5211 ± 0.168	0.2829 ± 0.2524	-0.0062 ± 0.0379
BERT layer -9	0.612 ± 0.1216	0.3256 ± 0.2423	0.0159 ± 0.018
RoBERTa layer -9	0.5151 ± 0.1773	0.3312 ± 0.2365	-0.0002 ± 0.0222

6.1.3 Distance/Similarity Metrics

	ℓ_2-norm distance	**Cosine similarity**
GloVe	0.023 ± 0.0032	0.9352 ± 0.0132
Word2Vec	0.0038 ± 0.0005	0.9231 ± 0.0161
BERT layer -1	0.0299 ± 0.0036	0.8994 ± 0.0221
RoBERTa layer -1	0.0103 ± 0.0013	0.9835 ± 0.0039
BERT layer -5	0.0388 ± 0.0045	0.926 ± 0.0196
RoBERTa layer -5	0.0259 ± 0.0032	0.9764 ± 0.0059
BERT layer -9	0.0377 ± 0.0045	0.9296 ± 0.0186
RoBERTa layer -9	0.0188 ± 0.0018	0.9843 ± 0.0026

Table 6.1: Relationship between the syntactic robustness metrics for four probing tasks on coPOS perturbations with budget $\tau = 2$ (top and middle row) and the distance between pairs of perturbed and original inputs measured via cosine similarity and ℓ_2-norm distance (bottom row). The accuracy drop of the POS-tag task is reported as the number of words correctly guessed in both cases. The reported standard deviation is measured by averaging over the 6 training corpora. Whilst the distance (similarity) between inputs and perturbations is low (high), we observe that all embeddings/representations are brittle to syntax-preserving perturbations.

6.2.1 Robustness

	Syntax Reconstruction			POS-tagging
	ΔSDR	ΔUUAS	ΔSp	ΔAcc. (# words)
GloVe	0.2098 ± 0.124	0.2616 ± 0.1411	0.139 ± 0.0106	6.743 ± 3.0337
Word2Vec	0.1655 ± 0.1114	0.1232 ± 0.1014	0.0118 ± 0.014	7.071 ± 2.1687
BERT layer -1	0.208 ± 0.0773	0.1782 ± 0.0862	0.0285 ± 0.0177	3.1592 ± 3.0359
RoBERTa layer -1	0.1989 ± 0.0823	0.1951 ± 0.116	0.02 ± 0.0146	4.2887 ± 3.3774
BERT layer -5	0.235 ± 0.082	0.267 ± 0.1293	0.0261 ± 0.0191	3.286 ± 3.1258
RoBERTa layer -5	0.2093 ± 0.0767	0.2319 ± 0.1117	0.0133 ± 0.0126	4.2887 ± 3.3774
BERT layer -9	0.235 ± 0.0859	0.2988 ± 0.154	0.0162 ± 0.0135	3.613 ± 3.3219
RoBERTa layer -9	0.2109 ± 0.0774	0.2378 ± 0.1227	0.0135 ± 0.012	3.561 ± 3.205

6.2.2 Robustness

	Root Identification	Tree Depth Estimation	
	ΔAcc.	ΔAcc.	ΔSp
GloVe	0.4987 ± 0.1827	$0.293, 0.2024$	$0.0417, 0.0134$
Word2Vec	0.5785 ± 0.1462	$0.2915, 0.2444$	$0.0174, 0.0184$
BERT layer -1	0.5526 ± 0.202	$0.3863, 0.2054$	$0.0838, 0.0494$
RoBERTa layer -1	0.5449 ± 0.1804	$0.3567, 0.2166$	$0.0993, 0.0531$
BERT layer -5	0.6374 ± 0.1483	$0.3937, 0.2247$	$0.0633, 0.0374$
RoBERTa layer -5	0.5448 ± 0.1735	$0.3267, 0.2338$	$0.0672, 0.0215$
BERT layer -9	0.6293 ± 0.1408	$0.3726, 0.2217$	$0.054, 0.0512$
RoBERTa layer -9	0.549 ± 0.1786	$0.3613, 0.2317$	$0.0471, 0.0326$

6.2.3 Distance/Similarity Metrics

	ℓ_2-norm distance	Cosine similarity
GloVe	0.0344 ± 0.0018	0.8783 ± 0.0271
Word2Vec	0.0059 ± 0.0002	0.8572 ± 0.0331
BERT layer -1	0.0487 ± 0.0063	0.7951 ± 0.0433
RoBERTa layer -1	0.0195 ± 0.0016	0.9597 ± 0.0064
BERT layer -5	0.0652 ± 0.0058	0.8432 ± 0.0316
RoBERTa layer -5	0.0488 ± 0.0037	0.9416 ± 0.0102
BERT layer -9	0.058 ± 0.004	0.8768 ± 0.0173
RoBERTa layer -9	0.0373 ± 0.0024	0.9557 ± 0.0057

Table 6.2: Relationship between the syntactic robustness metrics for four probing tasks on coCO perturbations with budget $\tau = 2$ (top and middle row) and the distance between pairs of perturbed and original inputs measured via cosine similarity and ℓ_2-norm distance (bottom row). The reported standard deviation is measured by averaging over the 6 training corpora. The accuracy drop of the POS-tag task is reported as the number of words correctly guessed in both cases. Whilst the distance (similarity) between inputs and perturbations is low (high), we observe that all embeddings/representations are brittle to syntax-preserving perturbations.

fifth counting from the most external), the -9^{th} and the last (i.e., -1^{th} or the output) hidden layer of a representation. While in (Manning et al., 2020) researchers observed that the most hidden layers perform the best on syntactic tasks, we also provide results for an intermediate and the last hidden layer.

For each probing task (in our setting, syntax reconstruction, POS-tagging, root identification, and tree-depth estimation, as per Def. 5, 6, 7 and 8), we stack a deep neural network on top of a linguistic representation ψ^{θ}, thus obtaining a set of models $\{f_1(s), .., f_m(s)\}$: we optimize each model f_i via supervised learning on the i-th task $\mathbf{T_i}$, leaving the representation's parameters θ fixed. We note that the measures of performance $\{\mathcal{L}_1, .., \mathcal{L}_m\}$ vary from one probing task to another, as we detail in the experimental evaluation.

When training the probing task models, we searched for a typical architecture that performs well for each of the four language models. We tested FCs, CNNs, and RNNs and found that FCs probing models performed best across the language models. For each combination of datasets, probing tasks, models, perturbation methods (coPOS, coCO), and for a varying perturbation budget τ, we train a 3-layer deep FC network with a varying number of parameters in the order of 10M. In this sense, both the static and dynamic representations are kept fixed (i.e., their parameters are 'frozen' at training time) not to invalidate the scope of the probing tasks and to allow full reproducibility of the results.

6.6.2 Empirical Evaluation of Syntactic Robustness

We now report the results of our robustness evaluation; in particular, we quantify the syntactic robustness of the representation ψ^{θ} according to Def. 12.

Performance on probing tasks. In Figure 6.10, we observe that, across the structural probe task and the POS-tag, all the models have similar performances, with an average POS-tag accuracy of around 0.8 and syntax reconstruction SDR around 0.7. Of the context-dependent language models, RoBERTa and BERT are comparable, although we cannot definitively conclude which is better. Similarly, GloVe slightly outperforms Word2Vec on tree-depth estimation, while Word2Vec is better on the structural probe. Interestingly, word embeddings are only slightly worse than BERT and RoBERTa. The same trends emerge on root identification and tree-depth estimation, as shown in Figure 6.10 (bottom): while BERT generally outperforms the competitors and Word2Vec struggles to compete, GloVe is a competitor of both the language models. We conclude with a final remark on the lack of performance gap

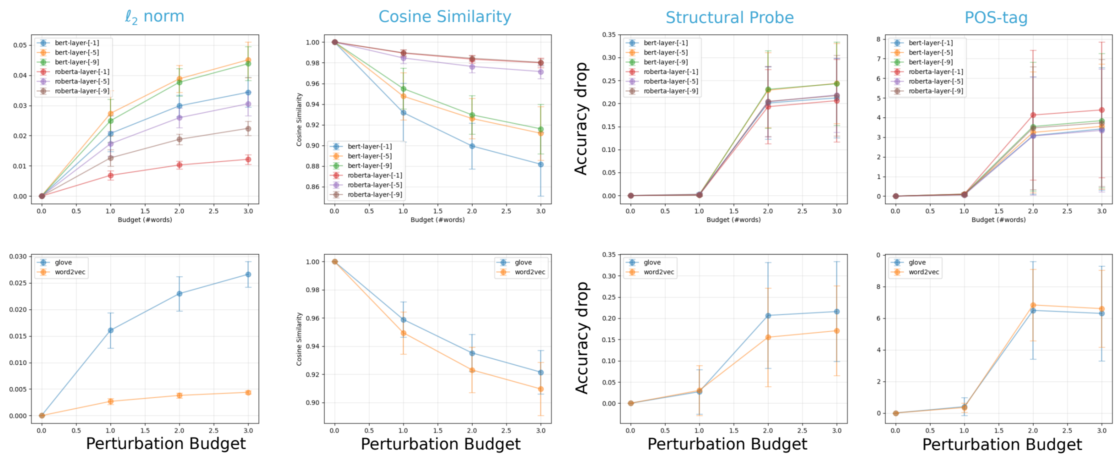


Figure 6.11: Left: For an increasing perturbation budget τ and the coPOS method, cosine similarity between perturbed and original sentences drops while the ℓ_2-norm increases. Right: It is clear that, even with $\tau = 2$ (i.e., at most two words per sentence are perturbed), the models' performance already experiences a significant drop (the higher the curve, the worse the model is on a syntactic task). Increasing the perturbation budget only slightly increases a drop in robustness.

between GloVe and Word2Vec, as it suggests that pre-training a representation on local and global word co-occurrences (Pennington et al., 2014b) does not help syntactic structures to emerge, a controversial yet intriguing discovery.[1]

Robustness on probing tasks. In Figure 6.10, solid bars represent the performance of a trained model subjected to a coPOS perturbation under the perturbation budget of $\tau = 3$: in other words, we generate an approximate worst-case coPOS perturbation given that we can only change at most three words in the given sentence. The drop in performance of a model on a task can be inferred via the gap between the solid and shaded bars (the latter corresponding to the unperturbed sentence). We notice that each metric has a substantial drop in performance, which suggests that each language model represents a brittle understanding of syntactic concepts.

In particular, in the syntax reconstruction task (Figure 6.10, top), we notice a dramatic decrease in UUAS of more than 50% on any task and any model, while the SDR drop is of around $20 - 30\%$. Thus, for each language model and dataset, the syntax reconstruction task is now incorrect more often than it is correct.

For UUAS, this indicates that our coPOS scheme can find syntactically meaningful perturbations which reduce the model's performance to random guessing. Secondly,

[1]`http://languagelog.ldc.upenn.edu/myl/PinkerChomskyMIT.html`

we highlight that, for UUAS and SDR, the most significant decrease in performance comes for the datasets for which the performance was the highest. This indicates that robust representation of syntax may be at odds with performance. The same considerations are valid for the POS-tag probing task, with the accuracy that drops to a random guess with the perturbation budget τ equal to 1. The coPOS perturbation method degrades the performance on root identification and tree-depth estimation as much as in the previous tasks. Performances drop to a random guess on any task and for any representation, providing evidence that these representations are brittle and, thus, not suited to domain shifts.

On the correlation between robustness and sentence similarity. In this batch of experiments, we keep track of the farthest distance between an input and its coPOS perturbation (see Def. 11) using the ℓ_2-norm and the cosine similarity between each pair of input/perturbation. We then measure the performance drop of each model to assess any correlation with the abovementioned measures of distance/similarity. As reported in Table 6.1, we find that high drops in performance can be caused by perturbations with small ℓ_2-norm compared to the unperturbed sentence, and conversely high cosine similarity. This confirms that linguistic representations are remarkably brittle even to local perturbations, i.e., those whose representation lies in the proximity of the original input. We replicate the results using the coCO perturbation method (Table 6.2), which confirms that perturbations extracted via GPT-2 conditioning are farther than the coPOS in the representation space and equally effective at dismantling a model's robustness. Similar observations can be made with the baseline perturbation method, as reported in Table 6.3.

Varying the perturbation budget. While we have already shown that a small perturbation budget τ exposes a representation to effective performance-degrading attacks, we now investigate the relationship between the distance/similarity metrics and robustness concerning various coPOS perturbations. In Figure 6.11, the cosine similarity and the ℓ_2-norm behave monotonically as τ increases. Deeper LLM's layers are less affected by an increased perturbation budget, with BERT less prone than RoBERTa to maintaining the internal consistency of its representation of the original and perturbed sentences (Figure 6.11, top-left). In word embeddings (Figure 6.11, bottom-left), while the trends of cosine similarity between GloVe and Word2Vec are similar, the ℓ_2-norm of Word2Vec does not change as much as for GloVe, a sign that in this representation words lie very close to each other w.r.t. the Euclidean distance.

6.3.1 Robustness

	Syntax Reconstruction			POS-tagging
	ΔSDR	**ΔUUAS**	**ΔSp**	**ΔAcc. (# words)**
GloVe	0.2087 ± 0.123	0.2822 ± 0.1498	0.0545 ± 0.0097	6.583 ± 2.9971
Word2Vec	0.161 ± 0.1137	0.139 ± 0.0996	0.035 ± 0.0079	6.8977 ± 2.2552
BERT layer -1	0.208 ± 0.0803	0.1787 ± 0.0827	0.0289 ± 0.0156	3.1208 ± 3.0138
RoBERTa layer -1	0.2026 ± 0.0805	0.1954 ± 0.1129	0.013 ± 0.0154	4.2408 ± 3.3794
BERT layer -5	0.2395 ± 0.0804	0.2599 ± 0.1287	0.023 ± 0.0196	3.289 ± 3.1118
RoBERTa layer -5	0.2123 ± 0.0764	0.2303 ± 0.1094	0.0137 ± 0.0098	3.1448 ± 3.1161
BERT layer -9	0.2345 ± 0.0938	0.2828 ± 0.162	0.0204 ± 0.0126	3.6504 ± 3.7005
RoBERTa layer -9	0.2151 ± 0.0773	0.2372 ± 0.1189	0.009 ± 0.0106	3.568 ± 3.2097

6.3.2 Robustness

	Root Identification	Tree Depth Estimation	
	ΔAcc.	**ΔAcc.**	**ΔSp**
GloVe	0.4853 ± 0.1776	0.2796 ± 0.2265	-0.0108 ± 0.1222
Word2Vec	0.6118 ± 0.1342	0.3115 ± 0.2495	0.0407 ± 0.0156
BERT layer -1	0.5466 ± 0.2041	0.3883 ± 0.2186	0.103 ± 0.0784
RoBERTa layer -1	0.5557 ± 0.1783	0.3586 ± 0.2312	0.0818 ± 0.0306
BERT layer -5	0.6466 ± 0.1421	0.3896 ± 0.2246	0.0317 ± 0.0575
RoBERTa layer -5	0.5464 ± 0.1778	0.3252 ± 0.2351	0.0225 ± 0.0704
BERT layer -9	0.6462 ± 0.1392	0.3927 ± 0.2461	0.0314 ± 0.0464
RoBERTa layer -9	0.5563 ± 0.1834	0.3485 ± 0.2493	0.0344 ± 0.046

6.3.3 Distance/Similarity Metrics

	ℓ_2-norm distance	**Cosine similarity**
GloVe	0.0427 ± 0.001	0.8156 ± 0.0391
Word2Vec	0.0059 ± 0.0002	0.8606 ± 0.0184
BERT layer -1	0.0538 ± 0.0052	0.7401 ± 0.0418
RoBERTa layer -1	0.0211 ± 0.0014	0.9519 ± 0.006
BERT layer -5	0.0731 ± 0.0041	0.8059 ± 0.0299
RoBERTa layer -5	0.0539 ± 0.004	0.9302 ± 0.0122
BERT layer -9	0.0669 ± 0.0035	0.8382 ± 0.0199
RoBERTa layer -9	0.0375 ± 0.0021	0.9553 ± 0.0051

Table 6.3: Relationship between the syntactic robustness metrics for four linear probing tasks on baseline perturbations with budget $\tau = 2$ (top and middle row) and the distance between pairs of perturbed and original inputs measured via cosine similarity and ℓ_2-norm distance (bottom row). The accuracy drop of the POS-tag task is reported as the number of words correctly guessed in both cases. The reported standard deviation is measured by averaging over the six training corpora. While the distance (similarity) between inputs and perturbations is low (high), all embeddings/representations are brittle to syntax-preserving perturbations.

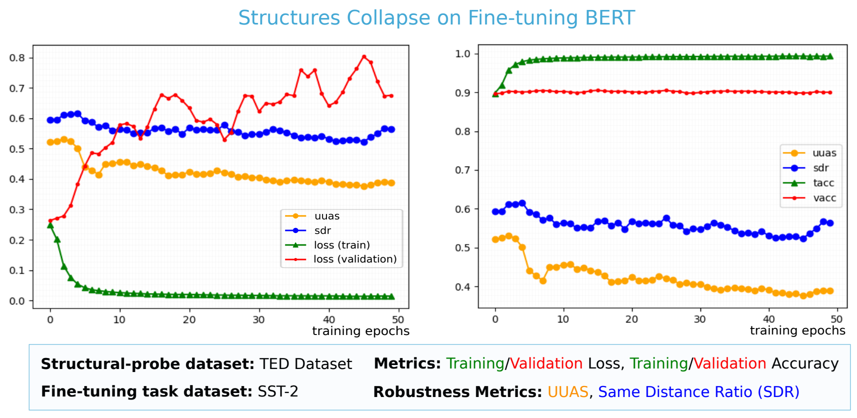


Figure 6.12: BERT model fine-tuned (and finally, overfitted) on the SST dataset, while its representations are used to train a model to solve the structural probe task. Train and validation losses (left) and accuracies (right) pertain to the fine-tuning task (SST), while SDR and UUAS show the performance of the structural probe. The syntactic metrics degrade as the fine-tuning process proceeds, yet severe overfitting does not harm syntactic structures.

Regarding performance drop (Figure 6.11, right), static and dynamic representations are comparable. The distance between representations, the performance drop on the coPOS, and the baseline method are reported in Figures 6.13 and 6.14. Results concerning both the perturbation methods are similar to those obtained with the coPOS method. In this case, we do not expect a model to be syntactically robust to such perturbation schemes.

An interesting phenomenon is the drop in performance that occurs when the number of perturbations is strictly greater than one ($\tau > 1$). Interestingly, such a drop does not correspond to a drop in the linguistic similarity between sentences (i.e., the ℓ_2-norm and the cosine similarity between input-perturbation pairs), a sign that brittleness is not immediately relatable to the distance, or conversely, the lack of similarity, of sentences in their embedding representations.

The effect of fine-tuning and overfitting on syntactic structures. We finally conduct an analysis whose primary intent is to understand the effect of counter-fitting (Mrkšić et al., 2016) and fine-tuning on syntactic structures, respectively, for context-free and context-dependent representations. We train a counter-fitted version of GloVe. We observe and report in Table 6.4 that any metric of a counter-fitted model has inferior performance on any task and any dataset while being equally brittle to coPOS perturbations, thus limiting the utility of counter-fitting on models aimed at

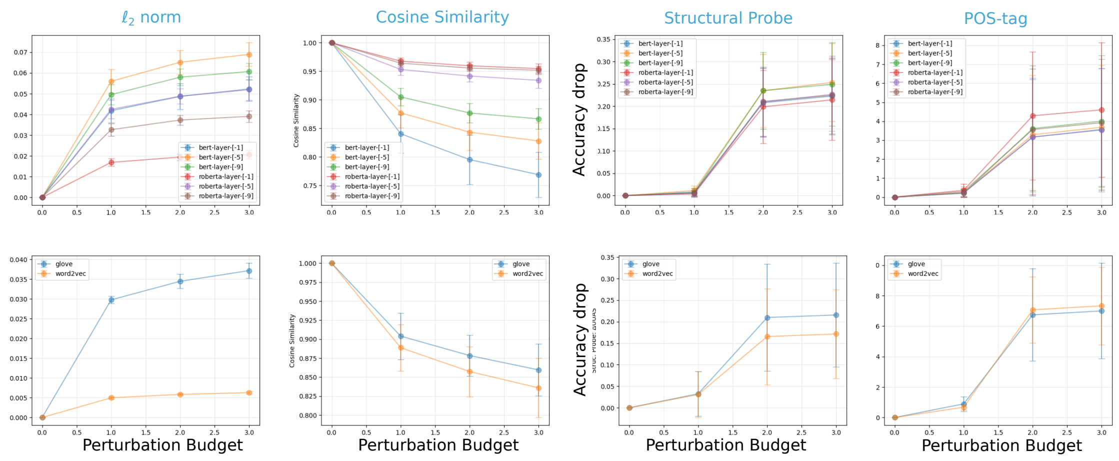


Figure 6.13: Left: for an increasing perturbation budget τ and the coCO method, cosine similarity between perturbed and original sentences drops, while the ℓ_2-norm increases. Right: It is clear that, even with $\tau = 2$ (i.e., at most two words per sentence are perturbed), the models' performance already experiences a significant drop (the higher the curve, the worse the model is on a syntactic task). Increasing the perturbation budget does not lead to a significant drop in robustness.

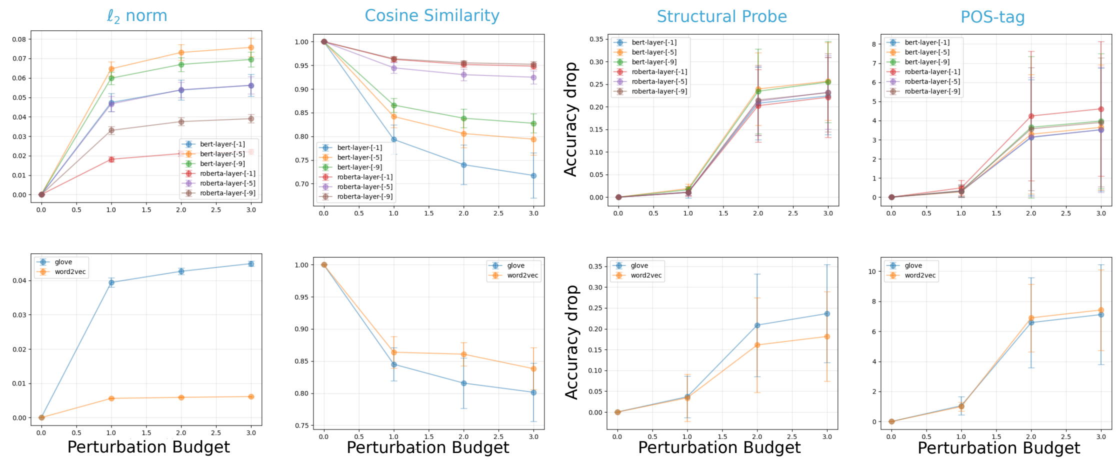


Figure 6.14: Left: For an increasing perturbation budget τ and the baseline method, cosine similarity between perturbed and original sentences drops, while the ℓ_2-norm increases. Right: It is clear that, even with $\tau = 2$ (i.e., at most two words per sentence are perturbed), the models' performance already experiences a significant drop (the higher the curve, the worse the model is on a syntactic task). Increasing the perturbation budget does not lead to a significant drop in robustness.

capturing different aspects of human linguistics: performance and robustness are in line with those of standard GloVe embedding (i.e., Table 6.1 and Figure 6.10).

We also fine-tune, and finally overfit, a BERT-based representation on the SST-2 dataset (Socher et al., 2013a): differently from previous experiments; we update the weights of the language model (i.e., we do not keep them frozen) to investigate the existence of some form of catastrophic forgetting of the syntactic structures encoded in the model. By performing the robustness analysis introduced in this chapter, we observe that fine-tuning negatively affects the performance of both shallow and deep context-dependent representations; despite this, excessive fine-tuning is not significantly harmful as the performance does not collapse even after many epochs the model has overfitted on the training set. The task's validation loss is informative to prevent overfitting on the structural probe task, while accuracy on the fine-tuning task can be misleading and hide syntax collapse. We sketch the training dynamics, along with the accuracy of the model on the classification task and the structural probe metrics, in Figure 6.12.

Justification for the linguistic structures collapse. In light of the empirical evidence in this chapter, we conclude with some hypotheses on why high performances of linguistic representations, whether LLMs or standard word embeddings, come at the cost of brittleness on high-order syntactic tasks. Indeed, the robustness-performance trade-off accounts for the frailty of over-fitted probes (Madry et al., 2018). On the other hand, the absence *stricto sensu* of adversarial attacks, replaced by coPOS and coCO perturbations, forces us to second-guess the existence of such structures. In high-dimensional spaces, vectors (i.e., words) become progressively harder to distinguish. At the same time, the high dimensionality allows one to optimize a decision boundary that is overfitted on the training set but fails poorly on slight input variations. The hypothesis that we put forward in this chapter, to stimulate discussion among NLP researchers as well as linguists, is that linguistic structures emerge as a process of fitting between static sentences and their syntax trees, granted by rich linguistic representations which nonetheless collapse as soon as the input distribution allows for word substitution, a shift against which human linguistic structures are indeed highly robust.

6.4.1 Robustness

	Syntax Reconstruction			POS-tagging
	ΔSDR	ΔUUAS	ΔSp	ΔAcc. (# words)
TED	0.167	0.0872	0.0011	4.470
En-Universal	0.2360	0.29931	0.0056	9.456
Ud-English-ewt	0.1463	0.3124	0.0.106	6.951
Ud-English-gum	0.1678	0.3252	0.0025	3.253
Ud-English-lines	0.2599	0.2981	0.0022	7.402
Ud-English-pud	0.0574	0.0.0612	0.0010	5.746

6.4.2 Robustness

	Root Identification	Tree Depth Estimation	
	ΔAcc.	ΔAcc.	ΔSp
TED	0.3753	0.0727	0.0165
En-Universal	0.5288	0.2215	-0.0011
Ud-English-ewt	0.816	0.7751	0.0102
Ud-English-gum	0.389	0.3065	0.0028
Ud-English-lines	0.4493	0.314	0.0037
Ud-English-pud	0.3894	0.2583	-0.0189

6.4.3 Distance/Similarity Metrics

	Cosine similarity	ℓ_2-norm distance
TED	0.881	0.006
En-Universal	$\approx 1.$	0.0033
Ud-English-ewt	0.9295	0.005
Ud-English-gum	$\approx 1.$	0.005
Ud-English-lines	$\approx 1.$	0.0058
Ud-English-pud	0.881	0.0058

Table 6.4: Robustness of GloVe counter-fitted models, with an analysis, w.r.t. each dataset (one per row), of the relationship between the syntactic robustness metrics for coCO perturbations with budget $\tau = 2$ (top and middle) and the distance between pairs of perturbations and original inputs (bottom). The accuracy drop of the POS-tag task is reported in number of words correctly guessed. Results confirm that counter-fitting does not improve robustness at any level (in this case, syntactic robustness).

6.7 Conclusions

In this chapter, we have examined the concept of syntactic robustness in linguistic representations. Robustness is a highly desirable characteristic for models of this nature, as such models have been reported to represent, with high fidelity, complex structures that researchers believed belong to the realm of linguistics rather than NLP models; however, we have demonstrated the potential risk associated with assuming their inherent robustness without sufficient scrutiny. Through empirical analysis, we have provided evidence of the substantial fragility exhibited by both LLMs and word embeddings when subjected to restrained perturbations using the coPOS and coCO methods. We have also investigated the robustness dynamics concerning overfitting and counter-fitting to complement these findings.

In the next chapter, we describe how robustness can enhance the explainability of an NLP model through the concept of optimal robust explanation (ORE).

Chapter 7

Robust Explainability

In this chapter, which concludes this thesis, we investigate the relationship between robustness and explainability. It is desirable to develop explainability methods that provide sound guarantees on the decision of a neural network model. Standard approaches, such as LIME and Anchors, could produce explanations whose constituent features differ from those that enhance the robustness and can thus be targeted by adversarial manipulations.

We introduce a notion of explanation that comes with robustness guarantees and optimality w.r.t. a cost function, which allows one, for example, to distill the shortest subset of features that logically imply a model's decision. In short, in this chapter, we make the following contributions:

- We define the concept of Optimal Robust Explanation (ORE) for NLP models as an explanation sufficient to preserve the model's decision while being optimal w.r.t. a cost function (e.g., the minimal in length). We provide a solution algorithm to extract such explanations from neural network models. We also show how OREs relate to Anchors (Ribeiro et al., 2018a) in terms of metrics optimized by the explanation.

- We show how OREs can be used to detect biases in a model and how to 'adjust' the explanation of a non-robust classifier by minimally enlarging its output with robust features.

- We conduct an experimental evaluation on different datasets and sentiment analysis models, which shows that OREs are compact and often identify terms in line with human expectation, alongside biases and spurious terms, which help to debug their complex internal.

We begin the study with two motivating examples, which show that standard (NLP) explainers can output explanations that are not locally consistent in the sense that a small perturbation of the features not included in the explanations induces a model's misclassification. We then provide a formal introduction to Optimal Robust Explanations, a notion of explanation meant to be locally robust and, at the same time, optimal w.r.t. a cost function. Such a notion extends abductive explanations (Ignatiev et al., 2018) to NNs. It is computationally challenging to calculate: we thus provide an algorithm to extract an ORE from NNs trained on standard NLP tasks such as sentiment analysis, improving the convergence on challenging instances and long input texts via heuristics that exploit adversarial attacks. OREs can be further used to check whether a model is biased, i.e., it leverages spurious patterns to solve a task and makes existing explainers robust by minimally extending their outputs with robust features.

Finally, after illustrating how ORE explanations compare to those obtained via Anchors in terms of precision and coverage (ref. Chapter 2/Section 2.3.3.1), we conclude the chapter with an extensive experimental evaluation where we extract OREs for FC and CNN models on different sentiment analysis tasks.

This chapter first appeared as (La Malfa et al., 2020), where additional results are reported.

7.1 On the Necessity of Guaranteed Explanations

Explainability in NLP focuses on understanding and interpreting the decision-making processes of complex models, enabling transparency and trustworthiness in their outcomes. One common way to approach explainability is via *post-hoc* explainers, as introduced in Section 3.3, a wide variety of techniques that partition the input features into subsets, guided by their influence on a model's classification.

On the other hand, salient features, i.e., those that influence the classification the most, can differ from those that enable robustness, thus exposing a model to an inconsistency between an explanation and its reliability against (adversarial) attacks. We show that a lower-bound measure of the robustness of a model can be used to perform interpretability analysis on a given text, that is, to quantify a robustness-based importance score of each word in the text.

For each word of a given text, one can compute the lower bound of the MSR as introduced in Chapter 4 and use this as a measure of its *saliency*, where small values of MSR indicate that minor perturbations of that word can significantly influence the

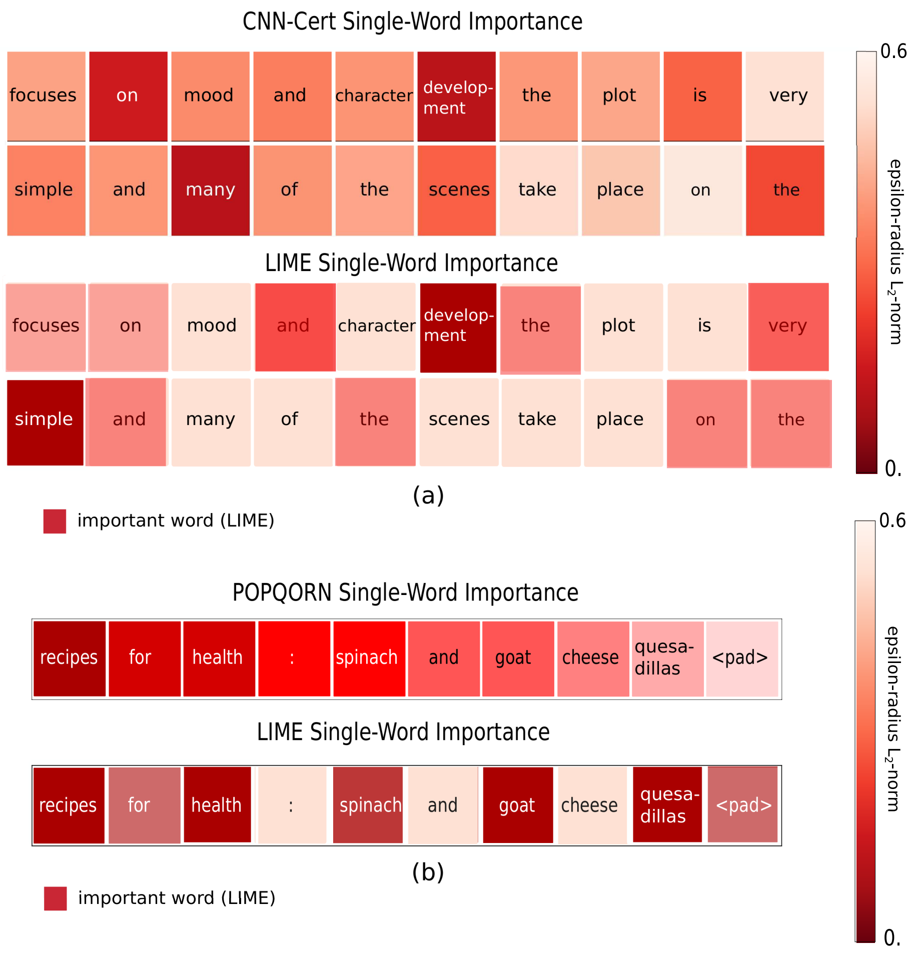


Figure 7.1: Interpretability comparison of our framework with LIME. (a) Saliency map produced with CNN-Cert (top) and LIME (bottom) on IMDB (GloVeTwitter 25d embedding). (b) Saliency map produced with POPQORN (top) and LIME (bottom) on NEWS dataset (GloVe 100d embedding).

classification outcome. We use the above measure to compute saliency maps for both CNNs and LSTMs, and compare our results with those obtained by LIME, which assigns saliency to input features according to the best linear model that locally explains the decision boundary. This method has the advantage of being able to account for non-linearities in the decision boundary that a local approach such as LIME cannot handle, albeit at the cost of higher computational complexity (a similar point was made in (Blaas et al., 2020) for Gaussian processes). As a result, we show words that our framework views as important but LIME does not, and vice versa. In Figure 7.1, we report two examples, one for an IMDB *positive* review (Figure 7.1 (a)) and another from the NEWS dataset classified using an LTSM (Figure 7.1 (b)). In Figure 7.1 (a) our approach finds that the word '`many`' is salient, i.e., slightly perturbing it, can make the NN change the review class to *negative*. In contrast, LIME does not identify `'many'` as significant. In order to verify this result empirically, we run our MCTS algorithm and find that simply substituting `'many'` with `'worst'` changes the classification to 'negative'. Similarly, for Figure 7.1 (b), where the input is assigned to class 5 (`'health'`), perturbing the punctuation mark (`':'`) may alter the classification, whereas LIME does not recognize its saliency.

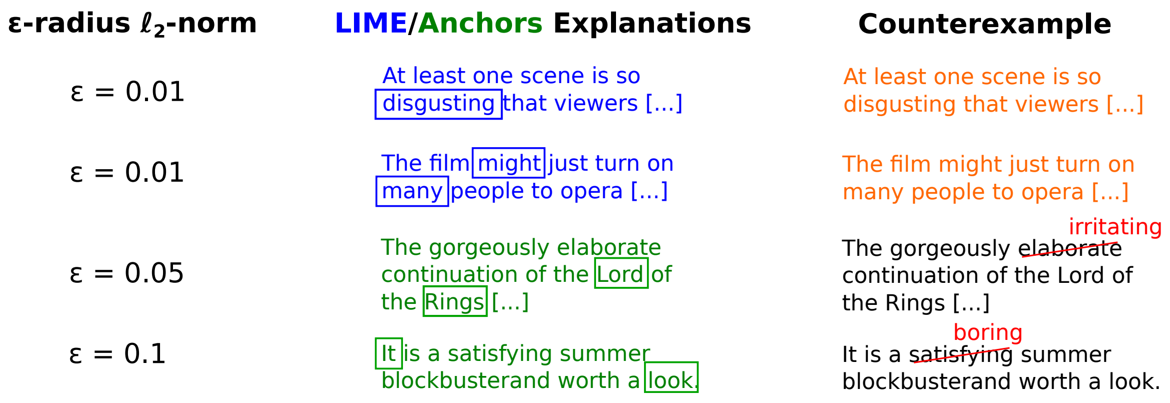
One could argue that LIME has yet to be developed to work with NLP models, as it solves an optimization problem by sampling in the embedding space (as described in Section 2.3.3.1). Unfortunately, even NLP-specific explainers suffer from brittleness to local perturbations of an input. Consider an NN that solves a sentiment analysis task, thus distinguishing between texts with positive and negative semantics (e.g., movie reviews). While explainers such as Anchors can provide helpful introspection of a model's behavior by highlighting positively/negatively polarized features, such explanations lack robustness guarantees. As shown in Figure 7.2, the left-out features of an Anchors explanation (middle) can be manipulated to induce a misclassification of the model. Note that the attacks are conducted in the neighborhood of each input word and are thus local.

Our contribution thus consists of a notion of an explanation that comes with local robustness guarantees, in addition to optimality concerning a cost function that, for example, one can use to distill the shortest explanations (i.e., those that encompass the minimal number of words): examples of shortest robust explanations from the SST dataset are shown in Figure 7.2 (left).

7.2 Optimal Robust Explanations

In this section, we seek to provide a formal definition for *local explanations* for the predictions of a neural network NLP model. For a text embedding $x = \psi(s)$ and a prediction $f(x)$, a local explanation E is a subset of the features of s, i.e., $E \subseteq F$ where $F = \{w_1, \dots, w_l\}$, expressed in terms of word constituents of the input sentence.

Our contribution consists of deriving *robust explanations*, i.e., on extracting a subset E of the text features F, which ensures that the neural network prediction remains invariant for any perturbation of the other features $F \setminus E$. Thus, the features in a robust explanation are *sufficient to imply the prediction* that we aim to explain, a desirable feature for a local explanation. In particular, we focus on robust explanations w.r.t. bounded perturbations in the embedding space of the input text, i.e., locally robust.



ε-radius ℓ_2-norm	LIME/Anchors Explanations	Counterexample
ε = 0.01	At least one scene is so $\boxed{\text{disgusting}}$ that viewers [...]	At least one scene is so disgusting that viewers [...]
ε = 0.01	The film $\boxed{\text{might}}$ just turn on $\boxed{\text{many}}$ people to opera [...]	The film might just turn on many people to opera [...]
ε = 0.05	The gorgeously elaborate continuation of the $\boxed{\text{Lord}}$ of the $\boxed{\text{Rings}}$ [...]	The gorgeously ~~elaborate~~ irritating continuation of the Lord of the Rings [...]
ε = 0.1	$\boxed{\text{It}}$ is a satisfying summer blockbusterand worth a $\boxed{\text{look.}}$	It is a ~~satisfying~~ boring summer blockbusterand worth a look.

Figure 7.2: Input texts are presented, accompanied by explanations generated using the LIME and Anchors methods. The text within the explanations is highlighted in blue and green, respectively, and the corresponding features are enclosed in boxes of matching colors. When the excluded features in the explanations become targets of attacks, both methods are susceptible to bounded perturbations (where the bound is calculated based on the ℓ_2-norm). Attacks against LIME are indicated in orange, as they cannot be traced back to specific words in the embedding dictionary. Attacks against Anchors are shown in red, along with a discrete replacement that leads to misclassification.

Definition 13 (Left-out knn-Ball)**.** *For some word-level perturbations contained in a knn-Ball (see Def. 2.25), a set of features $E \subseteq F$, and input text s with embedding $(x_1, \ldots, x_l)$, we denote with $\mathcal{B}_E(s)$ the set of text-level perturbations obtained from s by keeping constant the features in E and perturbing the others according to $\mathcal{B}$:*

$$\begin{aligned}\mathcal{B}_E(s) = \{(x'_1, \ldots, x'_l) \in \mathbb{R}^{l \cdot d} \mid x'_w = x_w \textit{ if } w \in E;\\ x'_w \in knn\text{-}Ball(w) \textit{ otherwise}\}.\end{aligned} \tag{7.1}$$

A robust explanation $E \subseteq F$ ensures prediction invariance for any point in $\mathcal{B}_E(s)$, i.e., any perturbation (within $\mathcal{B}$) of the features in $F \setminus E$.

Definition 14 (Robust Explanation)**.** *For a text $s = (w_1, \ldots, w_l)$ with linguistic representation $x = \psi(s)$, word-level perturbation $\mathcal{B}$, and classifier f, a subset $E \subseteq F$ of the features of s is a* robust explanation *iff*

$$\forall x' \in \mathcal{B}_E(t).\ f(x') = f(x). \tag{7.2}$$

We denote (7.2) *with predicate* $\mathsf{Rob}_{f,x}(E)$.

While robustness is desirable, stronger characteristics are needed to produce valuable explanations. Indeed, we can see that an explanation E including all the features, i.e., $E = F$, trivially satisfies Definition 14. Typically, one seeks short explanations because these can generalize to several instances beyond the input x and are more accessible for human decision-makers to interpret. We thus introduce Optimal Robust Explanations (OREs), that is, explanations that are both robust and optimal w.r.t. an arbitrary cost function that assigns a penalty to each word.

Definition 15 (Optimal Robust Explanation)**.** *Given a cost function $\mathcal{C} : W \to \mathbb{R}^+$, and for $s = (w_1, \ldots, w_l)$, x, $\mathcal{B}$, and f as in Def. 14, a subset $E^* \subseteq F$ of the features of s is an* ORE *iff*

$$E^* \in \underset{E \subseteq F}{argmin} \sum_{w \in E} \mathcal{C}(w) \ s.t. \ \mathsf{Rob}_{f,x}(E). \tag{7.3}$$

Note that (7.3) is always feasible because its feasible set always includes at least the trivial explanation $E = F$. A particular case of our OREs is when $\mathcal{C}$ is *uniform* (it assigns uniform cost to all words in s), in which case E^* is (one of) the *robust explanations of smallest size*, i.e., with the least number of words. On the other hand, cost functions that weigh each word differently allow ignoring and/or giving importance to specific inputs: one can, for example, assign a uniform cost function to a sentence while forcing the ORE to ignore the padding tokens by assigning them a very high cost.

7.2.1 Relation to Anchors

We now characterize an ORE in terms of the metrics that Anchors optimizes, namely precision and coverage, which we discussed in Section 2.3.3.1.

OREs precision. As discussed previously, precision is the probability that the prediction is invariant for any perturbation x' to which explanation A applies. To discuss the relation between Anchors and OREs, for an input text s, consider an arbitrary distribution $\mathcal{D}$ with support in $\mathcal{B}_{\emptyset}(s)$ (the set of all possible text-level perturbations), see (7.1); and consider anchors A defined as subsets E of the input features F, i.e., $A_E(x) = \bigwedge_{w \in E} x_w = \psi(w)$. Then, our OREs enjoy the following properties.

Proposition. If E is a robust explanation, then $\mathsf{prec}(A_E) = 1$.
Proof. A robust explanation $E \subseteq F$ guarantees prediction invariance for any $x' \in \mathcal{B}_E(s)$, i.e., for any x' (in the support of $\mathcal{D}$) to which anchor A_E applies. $\square$

OREs coverage. $\mathsf{cov}(A)$ is the probability that explanation A applies to a perturbation. With that being said, we note that when $\mathcal{D}$ is continuous, $\mathsf{cov}(A_E)$ is always zero unless $E = \emptyset$, in which case $\mathsf{cov}(A_\emptyset) = 1$ (as $A_\emptyset = \mathsf{true}$). Indeed, for $E \neq \emptyset$, the set $\{x' \mid A_E(x')\}$ has $|E|$ fewer degrees of freedom than the support of $\mathcal{D}$, and thus has both measure and coverage equal to zero. We thus illustrate the next property assuming that $\mathcal{D}$ is discrete (when $\mathcal{D}$ is continuous, the following still applies to any empirical approximation of $\mathcal{D}$).

Proposition. If $E \subseteq E'$, then $\mathsf{cov}(A_E) \geq \mathsf{cov}(A_{E'})$.
Proof. For discrete $\mathcal{D}$ with pmf $p_\mathcal{D}$, we can express $\mathsf{cov}(A_E)$ as

$$\mathsf{cov}(A_E) = \sum_{x' \in supp(\mathcal{D})} p_\mathcal{D}(x') \cdot \mathbf{1}_{A_E(x')} = \sum_{x' \in supp(\mathcal{D})} p_\mathcal{D}(x') \cdot \prod_{w \in E} \mathbf{1}_{x'_w = \psi(w)}.$$

To see that, for $E' \supseteq E$, $\mathsf{cov}(A_{E'}) \leq \mathsf{cov}(A_E)$, observe that $\mathsf{cov}(A_{E'})$ can be expressed as

$$\mathsf{cov}(A_{E'}) = \sum_{x' \in supp(\mathcal{D})} p_\mathcal{D}(x') \cdot \prod_{w \in E'} \mathbf{1}_{x'_w = \psi(w)} =$$
$$\sum_{x' \in supp(\mathcal{D})} p_\mathcal{D}(x') \cdot \prod_{w \in E} \mathbf{1}_{x'_w = \psi(w)} \cdot \prod_{w \in E \setminus E'} \mathbf{1}_{x'_w = \psi(w)},$$

and that for any x', $\prod_{w \in E \setminus E'} \mathbf{1}_{x'_w = \psi(w)} \leq 1$. $\square$

The above proposition suggests that using a uniform $\mathcal{C}$, i.e., minimizing the explanation's length is a sensible strategy to obtain high-coverage OREs.

7.3 Extracting OREs

We present a solution algorithm to derive OREs, based on the hitting-set (HS) paradigm of (Ignatiev et al., 2019c): we will rely on a few standard notions from first-order logic. The algorithm depends on repeated entailment/robustness checks $B \wedge E \wedge Net \models \hat{y}$ for a candidate explanation $E \subset C$, where C is a cube, i.e., the equivalent representation of an explanation as a conjunction of literals in first-order logic, B the first-order logic specification of $\mathcal{B}_E$ as per Eq. 7.1, and Net the logical encoding of the classifier (or equivalently, any classification algorithm). For this check, we employ Marabou (Katz et al., 2019a), a state-of-the-art neural network verification tool, which gives provably correct answers and, when the entailment is not satisfied, produces a counter-example $x' \in \mathcal{B}_E(s)$, i.e., a perturbation that agrees with E and such that $B \wedge C' \wedge Net \not\models \hat{y}$, where C' is the cube representing x'. We now outline the algorithm. We first provide the rationale behind the two procedures, while an in-depth discussion and the novel contributions, including the pseudo-code, are presented in Section 7.3.2.

7.3.1 Minimum Hitting Set

For a counter-example C', i.e., a cube that represents an attack that effectively changes the classification of a model, let I' be the set of feature variables where C' does not agree with C (the cube representing the input). Then, every explanation E that satisfies the entailment must hit all such sets I' built for any counter-examples C' (Ignatiev et al., 2016). Thus, the HS paradigm iteratively checks candidates E built by selecting the subset of C whose variables form a minimum HS (w.r.t. cost $\mathcal{C}$) of said I's.

However, this method does not always converge for our NLP models, especially with large perturbation spaces (i.e., large ϵ or k). We solved this problem by extending the HS approach with a sub-routine that generates batches of *sparse adversarial attacks* for the input C. This has a two-fold benefit: 1) we reduce the number of entailment queries required to produce counter-examples, and 2) sparsity results in small I' sets, further improving convergence.

7.3.2 MHS Pseudocode

In this section, we provide a complete description and the pseudo-code of the MHS algorithm. We report a line-by-line description of the HS procedure (Algorithm 6). We further describe how the adversarial attacks procedure is used to generate candidates that help the HS approach converge on hard instances. An illustrative example of the execution of the algorithm is presented in Figure 7.3

Minimal hitting-sets and explanations. One way to compute optimal explanations against a cost function C is through the hitting set paradigm (Ignatiev et al., 2019c), which exploits the relationship between diagnoses and conflicts (Reiter, 1987): the idea is to collect perturbations and to calculate on their indices a minimum hitting set (MHS), i.e., a minimum-cost explanation whose features are in common with all the others. We extend this framework to find a word-level explanation for non-trivial NLP models. At each iteration of Algorithm 6, a minimum hitting set E is extracted (line 3) from the (initially empty, line 1) set Γ. If function *Entails* evaluates to *False* (i.e., the neural network Net is provably safe against perturbations on the set of features identified by $F \setminus F$) the procedure terminates, and E is returned as an ORE. Otherwise, (at least) one feasible attack is computed on $F \setminus E$ and added to Γ (lines 7-8): the routine then re-starts. Differently from (Ignatiev et al., 2019c), as we have experienced that many OREs whose large perturbation space – i.e., when ϵ or k are large – do not terminate in a reasonable amount of time, we have extended the *vanilla* hitting set approach by introducing *SparseAttacks* function (line 7). At each iteration *SparseAttacks* introduces in the hitting set Γ a large number of sparse adversarial attacks on the set of features $F \setminus E$: it is, in fact, known (Ignatiev et al., 2016) that attacks that use as few features as possible help the convergence on instances that are hard (intuitively, a small set is harder to "hit" hence contributes substantially to the optimal solution compared to a longer one). *SparseAttacks* procedure is based on random search and is inspired by recent works in image recognition and malware detection (Croce et al., 2022): pseudo-code is reported as Algorithm in 7, and a detailed description follows in the next paragraph.

Sparse adversarial attacks. In Algorithm 7, we present a method to generate sparse adversarial attacks against features (i.e., words) of a generic input text. $GeneratePerturbations(k, n, Q)$ (line 2) returns a random population of n perturbations that succeed at changing Net's classification: for each successful attack p, a subset of k out of d features has been perturbed through a Fast Gradient Sign attack (FGSM), while it is ensured that the point lies inside a convex region Q, which in our case will be the ϵ-hyper-cube around the embedded text. Suppose no perturbation is found in this way (i.e., the population size of the attacks is zero, as in line 3). In that case, the budget decreases (line 4), and another trial of $GeneratePerturbations(k, n, Q)$ is performed (e.g., with few features as targets and a different random seed to guide the attacks). The function $AccuracyDrop(Net, P)$ returns the best perturbation a where k is increasingly minimized (line 7). The algorithm terminates when either no attacks are possible (all the combinations of features have been explored) or after a fixed number of iterations has been performed (line 1).

7.3.3 OREs Use Cases

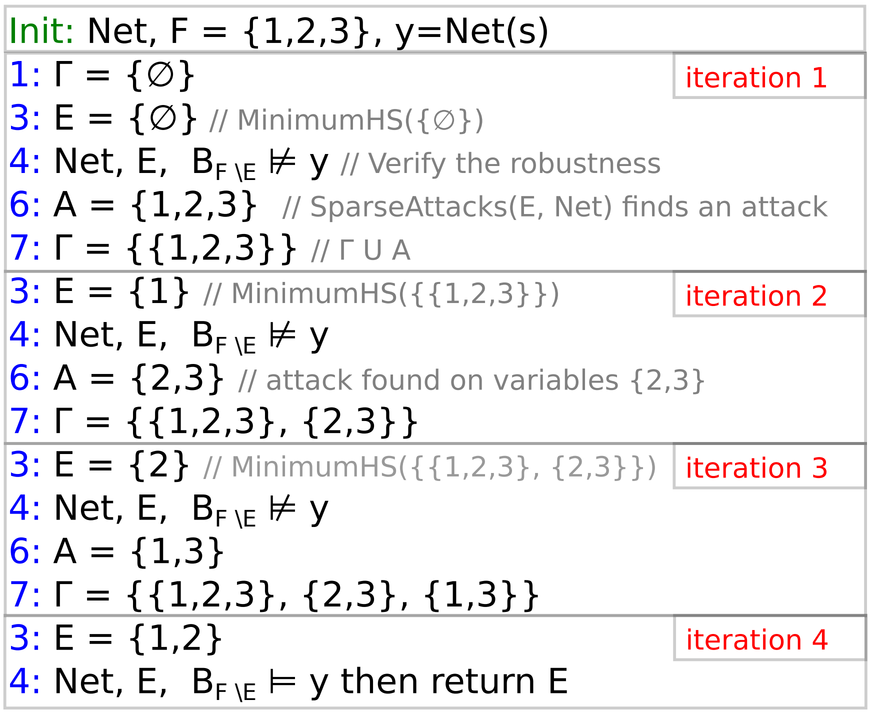
This section introduces how OREs can be leveraged to inspect a model's behavior and eventually find biases. It proceeds with a subsection dedicated to Anchors explanations and how they can be minimally adjusted to become locally robust (as motivated in Figure 7.2). OREs use-cases are discussed before the experimental results, which

Algorithm 6 ORE computation via implicit hitting sets and sparse attacks.

```
Require: a network $Net$, the input text $s$, the initial set of features F, a network
    prediction $\hat{y}$ , a cost function $\mathcal{C}$ against which the explanation is minimised
Ensure: an optimal ORE E
 1: $\Gamma = \emptyset$
 2: while true do
 3:     $E = MinimumHS(\Gamma, \mathcal{C})$
 4:     if $Entails(E, (Net \wedge \mathcal{B}_{F \setminus E}(s)) \rightarrow E)$ then return $E$
 5:     else
 6:         $A = SparseAttacks(E, Net)$
 7:         $\Gamma = \Gamma \cup \{A\}$
 8:     end if
 9: end while
```

Algorithm 7 Computing a perturbation that is successful and minimizes the number of features that are perturbed.

```
Require: $Net$ - neural network model, $F$ - input text from feature space; $k \in \mathbb{N}^{+}_{\setminus\{0\}}$ -
    number of perturbations initially tested; $Q \subseteq F$ - (sub)set of features where per-
    turbations are found; $n \in \mathbb{N}^{+}_{\setminus\{0\}}$ - number of elements generated at each iteration
    of the algorithm; budget - number of iterations allowed before stopping.
Ensure: a perturbation that is optimal w.r.t. the number of perturbed features,
 1: while $k > 0 \ \wedge \ budget > 0$ do
 2:     $P \leftarrow GeneratePerturbations(k, n, Q)$
 3:     if length(P) $==0$ then
 4:         $budget \leftarrow budget - 1$
 5:         continue
 6:     end if
 7:     $a \leftarrow \arg\max_{p \in P} AccuracyDrop(M, P)$
 8:     $k \leftarrow k - 1$, $budget \leftarrow budget - 1$
 9: end while
10: return $a$
```



```
Init: Net, F = {1,2,3}, y=Net(s)
1: Γ = {∅}                                          iteration 1
3: E = {∅} // MinimumHS({∅})
4: Net, E,  B_{F \E} ⊭ y // Verify the robustness
6: A = {1,2,3}  // SparseAttacks(E, Net) finds an attack
7: Γ = {{1,2,3}} // Γ U A
3: E = {1} // MinimumHS({{1,2,3}})                   iteration 2
4: Net, E,  B_{F \E} ⊭ y
6: A = {2,3} // attack found on variables {2,3}
7: Γ = {{1,2,3}, {2,3}}
3: E = {2} // MinimumHS({{1,2,3}, {2,3}})            iteration 3
4: Net, E,  B_{F \E} ⊭ y
6: A = {1,3}
7: Γ = {{1,2,3}, {2,3}, {1,3}}
3: E = {1,2}                                         iteration 4
4: Net, E,  B_{F \E} ⊨ y then return E
```

Figure 7.3: An illustration of the execution of Algorithm 6 for a set of input features $F = \{1, 2, 3\}$ and a generic model Net. Please notice that the lines at each iteration, in blue, correspond to the lines of the pseudo-code of Algorithm 6. After the network has been initialized (Init:), Net is checked to be robust against attacks that encompass all the input features (iteration 1, line 4). An attack is found and added to Γ (iteration 1, lines 6,7). At the second iteration, a valid minimum hitting set for Γ is the set $E = \{1\}$, for which the network is proven again not to be robust. An attack is found and added to Γ (iteration 2, lines 6,7). In the third iteration, the network is still not robust to the left-out features of the newly computed minimum hitting set (iteration 3, lines 3,4). The attack $A = \{1, 3\}$ is found and added to Γ. In the last iteration, the minimum hitting set, which now encompasses two variables (iteration 4, lines 3), is enough to secure the network from any attack on the left-out features, i.e., $\{3\} = F \setminus \{1, 2\}$, hence the ORE $\{1, 2\}$ is returned.

provide empirical evidence of the applicability of robust explanations to the model's introspection and debugging.

7.3.4 OREs can Detect Model/Decision Biases

By adding a simple constraint to the optimization problem formulated in Eq. 7.3, one can enforce specific features F' to be included/excluded from the explanation:

$$E^* \in \underset{E \subseteq F}{argmin} \sum_{w \in E} \mathcal{C}(w) \text{ s.t. } \mathsf{Rob}_{f,x}(E) \wedge \xi(E), \tag{7.4}$$

where $\xi(E)$ is one of $F' \cap E = \emptyset$ (*exclude*) and $F' \subseteq E$ (*include*). Note that adding *include* constraints does not affect the feasibility of our problem, as the feasible region of (7.4) always contains at least the explanation $E^* \cup F'$, where E^* is a solution of (7.3), and F' are the features to include. See Def. 14. Conversely, *exclude* constraints might make the problem infeasible when the features in F' do not admit perturbations, i.e., they are necessary for the prediction and thus cannot be excluded. Any solution algorithm for non-constrained OREs can easily accommodate such constraints: for *include* ones, it is sufficient to restrict the feasible set of explanations to the supersets of F'; for *exclude* constraints, we can manipulate the cost function to make any explanation with features in F' strictly sub-optimal w.r.t. explanations without. That is, we use cost $\mathcal{C}'$ such that $\forall_{w \in F \setminus F'} \mathcal{C}'(w) = \mathcal{C}(w)$ and $\forall_{w' \in F'} \mathcal{C}'(w') > \sum_{w \in F \setminus F'} \mathcal{C}(w)$. The ORE obtained under cost $\mathcal{C}'$ might still include features from F', which implies that (7.4) is infeasible (i.e., no robust explanation without elements of F' exists).

Constrained OREs enable two crucial use cases: *detecting biased decisions*, and *enhancing non-formal explainability frameworks*. As we described at the beginning of the chapter, explainers can be brittle to adversarial perturbations; hence their output is not reliable even for slight variations of the input. We conclude this section by formally defining how an ORE can be used to detect a model bias. At the same time, we reserve the following section to discuss how to improve Anchors explanations by making them robust and then discuss how OREs relate to Anchors in terms of precision and coverage (see Section 2.3.3.1).

Detecting bias. Following (Darwiche and Hirth, 2020), we deem a classifier decision *biased* if it depends on protected features, i.e., a set of input words that should not affect the decision (e.g., a movie review affected by the director's name). In particular, a decision $f(x)$ is biased if we can find, within a given set of text-level perturbations, an input x' that agrees with x on all but protected features and such that $f(x) \neq f(x')$.

Definition 16. *For classifier f, text s with features F, protected features F' and embedding $x = \psi(s)$, decision $f(x)$ is* biased *w.r.t. some word-level perturbation $\mathcal{B}$, if*

$$\exists x' \in \mathcal{B}_{F \setminus F'}(s).f(x) \neq f(x').$$

The proposition below allows us to use 'exclude constraints' to detect bias.

Proposition 3. For f, s, F, F', x and $\mathcal{B}$ as per Def. 16, *decision $f(x)$ is biased iff* (7.4) *is infeasible under $F' \cap E = \emptyset$.*
Proof. Call $A =$ *"$f(x)$ is biased"* and $B =$ "(7.4) *is infeasible under $F' \cap E = \emptyset$"*. Let us prove first that $B \rightarrow A$. Note that B can be equivalently expressed as

$$\forall E \subseteq F.(E \cap F' \neq \emptyset \vee \exists x' \in \mathcal{B}_E(s).f(x) \neq f(x'))$$

If the above holds for all E then it holds also for $E = F \setminus F'$, and so it must be that $\exists x' \in \mathcal{B}_{F \setminus F'}(s)$. $f(x) \neq f(x')$ because the first disjunct is clearly false for $E = F \setminus F'$.

We now prove $A \rightarrow B$ by showing that $\neg B \rightarrow \neg A$. Note that $\neg B$ can be expressed as

$$\exists E \subseteq F.(E \cap F' = \emptyset \wedge \forall x' \in \mathcal{B}_E(s).f(x) = f(x')), \tag{7.5}$$

and $\neg A$ can be expressed as

$$\forall x' \in \mathcal{B}_{F \setminus F'}(s).f(x) = f(x'). \tag{7.6}$$

To see that (7.5) implies (7.6), note that any E that satisfies (7.5) must be such that $E \cap F' = \emptyset$, which implies that $E \subseteq F \setminus F'$, which in turn implies that $\mathcal{B}_{F \setminus F'}(s) \subseteq \mathcal{B}_E(s)$. By (7.5), the prediction is invariant for any x' in $\mathcal{B}_E(s)$, and so is for any x' in $\mathcal{B}_{F \setminus F'}(s)$ $\square$.

7.3.5 Enhancing Anchors Explanations

The local explanations produced by heuristic approaches such as LIME or Anchors do not enjoy the same robustness/invariance guarantees as our OREs. We can use our approach to *minimally extend* (e.g., w.r.t. the uniform cost function) any non-robust local explanation F' in order to make it robust, by solving (7.4) under the *include* constraint $F' \subseteq E$. In particular, with a uniform $\mathcal{C}$, our approach would identify one of the smallest sets of extra words that make F' robust. Such an extension retains to a large extent the original explainability properties.

The experimental section will show how this approach practically applies to an NN classifier.

7.4 Experimental Evaluation

We proceed with an extensive experimental evaluation aimed at addressing the following research inquiries consistently and rigorously. After an introduction to the experimental setup, in terms of datasets and architectures employed, we show some examples of OREs. Secondly, we show how one can detect biases in a model, and use OREs to debug a model when it misclassifies an example. We finally compare our method with Anchors, showing that the latter is not robust, yet can be combined with OREs to minimally extend it and make an Anchors explanation robust to adversarial perturbations.

7.4.1 Experimental Setup

We have trained fully connected (FC) and convolutional neural networks (CNN) models on sentiment analysis datasets that differ in the input length and difficulty of the learning task. We performed our experiments on NNs with up to 6 layers and $20K$ parameters. FCs are constituted by a stack of dense layers. At the same time, CNNs additionally employ convolutional and max-pooling layers: for both CNNs and FCs, the decision is taken through a softmax layer, with dropout that is added after each layer to improve generalization during the training phase. As regards the embeddings that the models are equipped with, we experienced that the best trade-off between the accuracy of the network and the formal guarantees that we need to provide is reached with low-dimensional embeddings; thus, we employed optimized vectors of dimension 5 for each word in the embedding space: this is in line with the experimental evaluations conducted in (Patel and Bhattacharyya, 2017), where for low-order tasks such as sentiment analysis, compact embedding vectors allow one to obtain good performance, as shown in Table 7.1.

Experiments were parallelized on a server with two 24-core Intel Xenon 6252 processors and 256GB of RAM, but each instance is single-threaded and can be executed on a low-end laptop. We considered 3 well-established benchmarks for sentiment analysis, namely SST (Socher et al., 2013b), Twitter (Go et al., 2009) and IMDB (Maas

Training

	TWITTER	**SST**	**IMDB**
Inputs (Train, Test)	$1.55M, 50K$	$117.22K, 1.82K$	$25K, 25K$
Output Classes	2	2	2
Input Length (max, max. used)	88, 50	52, 50	2315, 100
Neural Network Models	FC, CNN	FC, CNN	FC, CNN
Neural Network Layers (min,max)	3,6	3,6	3,6
Accuracy on Test Set (min, max)	0.77, 0.81	0.82, 0.89	0.69, 0.81
Number of Networks Parameters (min,max)	$3K, 18K$	$1.3K, 10K$	$5K, 17K$

Explanations

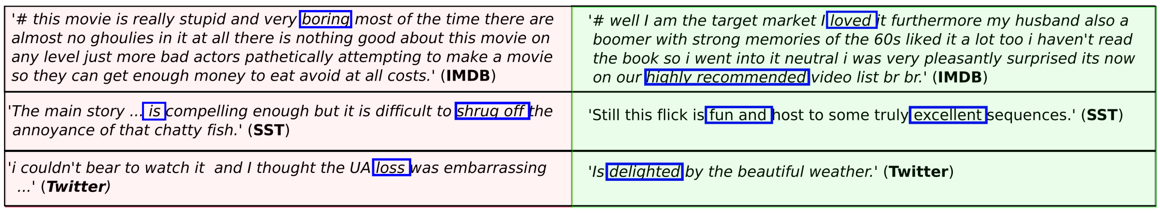
	TWITTER	**SST**	**IMDB**
Sample Size	40	40	40
Review Length (min-max)	10, 50	10, 50	25, 100

Table 7.1: Datasets used for training/testing and extracting explanations. In the top table, we report various metrics concerning the networks and the training phase (including accuracy on the test set), while in the bottom table, we report the number of texts for which we have extracted explanations along with the number of words considered when calculating OREs: samples were chosen to reflect the variety of the original datasets, i.e., a mix of long/short inputs equally divided into positive and negative instances.

et al., 2011a) datasets. The characteristics of both the SST-2 and the IMDB datasets have been discussed extensively in the previous chapters.

We have chosen 40 representative input texts from each dataset, balancing *positive* and *negative* examples. Embeddings are pre-trained on the same datasets used for classification (Chollet et al., 2015). The HS algorithm has been implemented in Python and uses Marabou (Katz et al., 2019a) to answer robustness queries. In the experiments below, we opted for the knn-box perturbation space, as we found that the k parameter was easier to interpret and tune than the ϵ parameter for the ϵ-Ball space and improved verification time.

Effect of classifier's accuracy and robustness. Our approach generally results in meaningful and compact explanations for NLP. In Figure 7.4, we show a few OREs extracted for *negative* and *positive* texts, where the returned OREs are both concise and semantically consistent with the predicted sentiment. However, the quality of our OREs depends on that of the underlying classifier. Indeed, enhanced models with better accuracy and/or trained on longer inputs tend to produce higher-quality OREs. We show this in Figures 7.5 and 7.6, where we observe that enhanced models tend to result in more semantically consistent explanations. For lower-quality models,



'# this movie is really stupid and very boring most of the time there are almost no ghoulies in it at all there is nothing good about this movie on any level just more bad actors pathetically attempting to make a movie so they can get enough money to eat avoid at all costs.' (**IMDB**)

'# well I am the target market I loved it furthermore my husband also a boomer with strong memories of the 60s liked it a lot too i haven't read the book so i went into it neutral i was very pleasantly surprised its now on our highly recommended video list br br.' (**IMDB**)

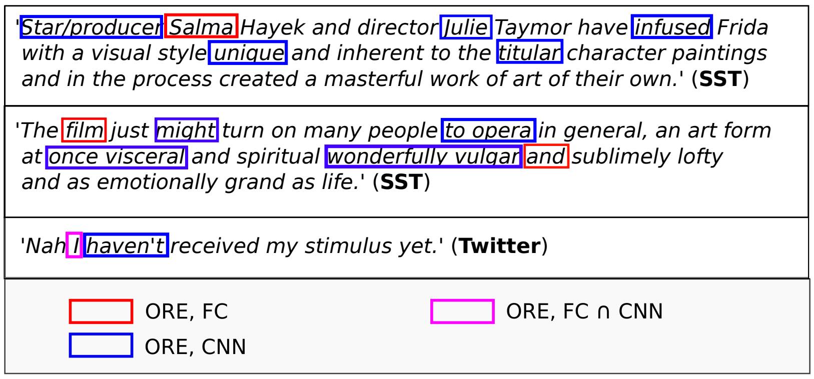
'The main story ... is compelling enough but it is difficult to shrug off the annoyance of that chatty fish.' (**SST**)

'Still this flick is fun and host to some truly excellent sequences.' (**SST**)

'i couldn't bear to watch it and I thought the UA loss was embarrassing ...' (***Twitter***)

'Is delighted by the beautiful weather.' (**Twitter**)

Figure 7.4: OREs for IMDB, SST, and Twitter datasets (all the texts are correctly classified). Models employed are FC with 50 input words, each with accuracies respectively 0.89, 0.77, and 0.75. OREs are highlighted in blue. The technique used is knn boxes with k=15.



'Star/producer Salma Hayek and director Julie Taymor have infused Frida with a visual style unique and inherent to the titular character paintings and in the process created a masterful work of art of their own.' (**SST**)

'The film just might turn on many people to opera in general, an art form at once visceral and spiritual wonderfully vulgar and sublimely lofty and as emotionally grand as life.' (**SST**)

'Nah I haven't received my stimulus yet.' (**Twitter**)

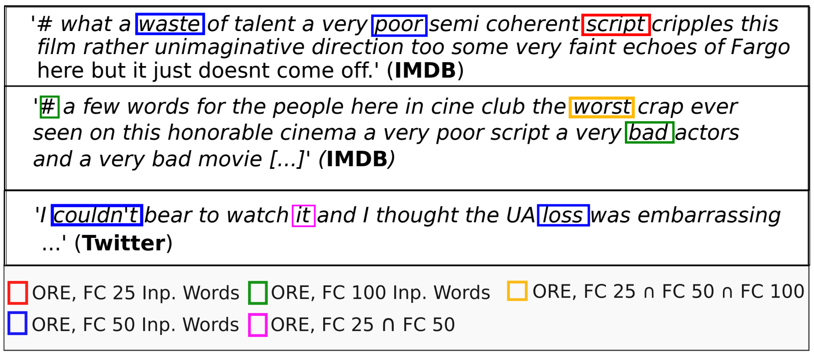
🟥 ORE, FC
🟦 ORE, CNN
🟪 ORE, FC ∩ CNN

Figure 7.5: Comparison of OREs for SST and Twitter texts on FC (red) vs. CNN (blue) models (common words in magenta). The first two are *positive* reviews; the third is *negative* (all correctly classified). Accuracies of FC and CNN models are 0.88 and 0.89 on SST, and 0.77 on Twitter. Models have input lengths of 25 words; OREs are extracted with knn boxes (k=25).

some OREs include seemingly irrelevant terms (e.g., `'film'`, `'and'`), thus exhibiting shortcomings of the classifier.

Detecting biases. As per Prop. 7.3.4, we applied exclude constraints to detect biased decisions. In Figure 7.7, we provide a few example instances exhibiting such a bias, i.e., where *any* robust explanation contains at least one protected feature. These OREs include proper names that should not constitute a sufficient reason for the model's classification. When we try to exclude proper names, no robust explanation exists, indicating a decision bias.

Debugging prediction errors. A relevant use case for OREs is when a model commits a misclassification. Misclassifications in sentiment analysis tasks usually depend on the over-sensitivity of the model to polarized terms. In this sense, knowing a minimal, sufficient reason behind the model's prediction can be helpful in debugging

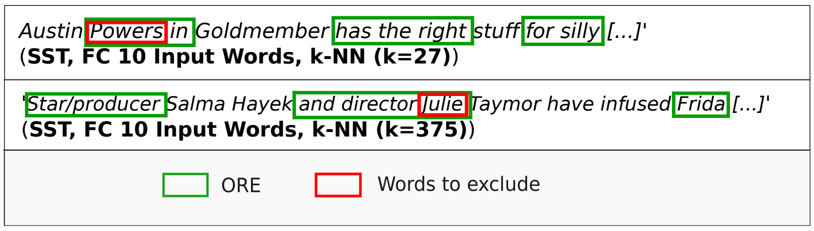

'# what a waste of talent a very poor semi coherent script cripples this film rather unimaginative direction too some very faint echoes of Fargo here but it just doesnt come off.' (**IMDB**)

*'# a few words for the people here in cine club the worst crap ever seen on this honorable cinema a very poor script a very bad actors and a very bad movie [...]' (**IMDB**)*

'I couldn't bear to watch it and I thought the UA loss was embarrassing ...' (**Twitter**)

🟥 ORE, FC 25 Inp. Words 🟩 ORE, FC 100 Inp. Words 🟨 ORE, FC 25 ∩ FC 50 ∩ FC 100
🟦 ORE, FC 50 Inp. Words 🟪 ORE, FC 25 ∩ FC 50

Figure 7.6: Comparison of OREs on *negative* IMDB and Twitter inputs for FC models. The first and third examples are trained with 25 (red) VS 50 (blue) input words (words in common to both OREs are in magenta). The second example uses an FC model trained with 100 input words (words common to all three OREs are in orange). Accuracy is respectively 0.7 and 0.77 and 0.81 for IMDB and 0.77 for both Twitter models. All the examples are classified correctly. OREs are extracted with knn boxes (k=25).


Austin Powers in Goldmember has the right stuff for silly [...]'
(**SST, FC 10 Input Words, k-NN (k=27)**)

'Star/producer Salma Hayek and director Julie Taymor have infused Frida [...]'
(**SST, FC 10 Input Words, k-NN (k=375)**)

🟩 ORE 🟥 Words to exclude

Figure 7.7: Two examples of decision bias from an FC model with an accuracy of 0.80.

'# I've seen Foxy Brown, Coffy Friday Foster Bucktown, and Black Mama White Mama of these this is Pam Griers worst movie poor acting bad script boring action scenes theres just nothing there avoid this and rent Friday Foster Coffy or Foxy Brown instead' (**IMDB, predicted as *negative***)

*'# I gave this a 2 and it only avoided a 1 because of the occasional unintentional laugh the film is excruciatingly. Boring and incredibly cheap its even worse if you know anything at all about the Fantastic Four.', (***IMDB, predicted as negative***)*

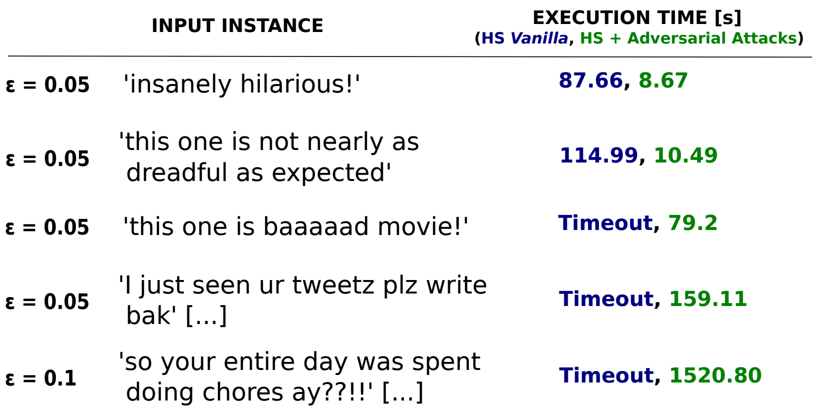
*'# a few words for the people here in cine club the worst crap ever seen on this honorable cinema a very poor script a very bad actors and a very bad movie dont waste your time looking this movie see the very good or any movie have been good commented by me say no more' (***IMDB, predicted as *negative****)*

Figure 7.8: Examples of Optimal Robust Explanations, highlighted in blue. OREs were extracted using kNN boxes with 25 neighbors per word: fixing words in an ORE guarantees the model to be locally robust. The examples come from the IMDB dataset; the model employed is an FC network with 100 input words (accuracy 0.81).

it. As shown in the first example in Figure 7.10, the model cannot recognize the *double negation* constituted by the terms `‘not’` and `‘dreadful’` as a syntax construct, hence it exploits the negation term `‘not’` to classify the review as *negative*.

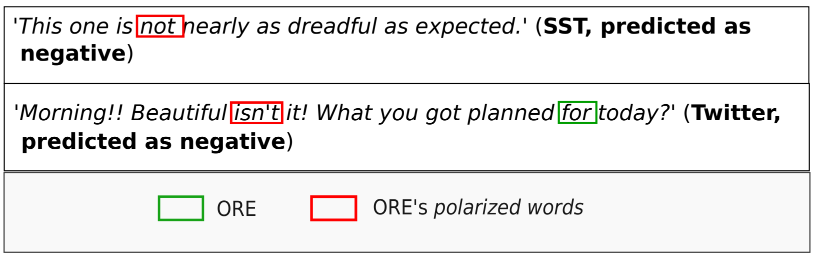
Making OREs scale. We also investigated the relationship between a model's complexity and the execution time of the hitting set algorithm, with and without the improvement brought by the sparse adversarial attacks routine (see Algorithms 6 and 7). We tested a model trained on the IMDB dataset whose explanations are extracted from texts with 100 input words. While the *vanilla* hitting set algorithm cannot solve any instance within a 2-hour timeout, aiding it with sparse attacks allows us to extract OREs, which we report in Figure 7.8. Other examples of the execution time of the hitting set routine, with and without the sparse attacks improvement, and for varying values of ϵ, are reported in Figure 7.9. It is interesting to notice that when the region to certify grows too large, the hitting set paradigm requires sparse attack improvements; otherwise, it does not terminate within a 2-hours timeout.

Comparison to Anchors. We evaluate the robustness of Anchors for FC and CNN models on the SST and Twitter datasets: accuracies are respectively of 0.89 for FC+SST, 0.82 for FC+Twitter, 0.89 for CNN+SST, and 0.77 for CNN+Twitter. We assume a knn-box perturbation space $\mathcal{B}$ with $k = 15$ for FC and $k = 25$ for CNN models to compute robustness. To extract Anchors, we set $\mathcal{D}$ to the standard perturbation distribution of (Ribeiro et al., 2018a), defined by a set of context-wise perturbations generated by a powerful language model. Thus defined $\mathcal{B}$s are small compared to the support of $\mathcal{D}$. Therefore, one would expect high-precision Anchors



	INPUT INSTANCE	EXECUTION TIME [s] (HS *Vanilla*, HS + Adversarial Attacks)
ε = 0.05	'insanely hilarious!'	**87.66, 8.67**
ε = 0.05	'this one is not nearly as dreadful as expected'	**114.99, 10.49**
ε = 0.05	'this one is baaaaad movie!'	**Timeout, 79.2**
ε = 0.05	'I just seen ur tweetz plz write bak' [...]	**Timeout, 159.11**
ε = 0.1	'so your entire day was spent doing chores ay??!!' [...]	**Timeout, 1520.80**

Figure 7.9: Examples of explanations that were enabled by the adversarial attacks routine. Timeout was set to 2 hours.

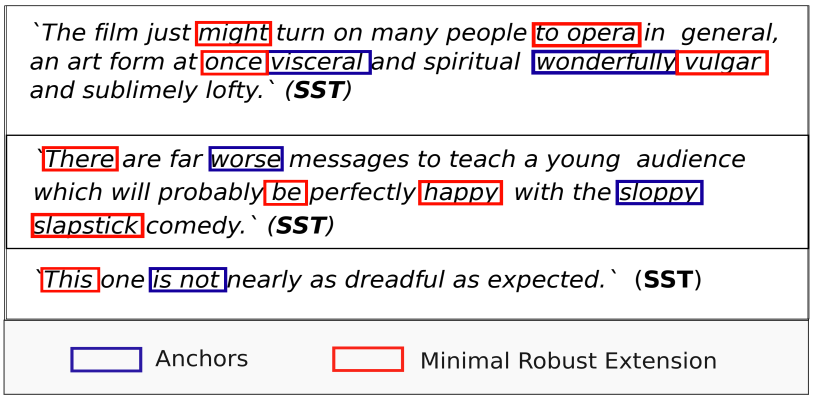


'*This one is not nearly as dreadful as expected.*' (**SST, predicted as negative**)

'*Morning!! Beautiful isn't it! What you got planned for today?*' (**Twitter, predicted as negative**)

🟩 ORE 🟥 ORE's *polarized words*

Figure 7.10: Two examples of over-sensitivity to polarized terms (in red). Other words in the OREs are highlighted in green. Models used are FC with 25 input words (accuracy 0.82 and 0.74). The method used is knn with k respectively equal to 8 and 10.


`The film just might turn on many people to opera in general, an art form at once visceral and spiritual wonderfully vulgar and sublimely lofty.` (***SST***)

`There are far worse messages to teach a young audience which will probably be perfectly happy with the sloppy slapstick comedy.` (***SST***)

`This one is not nearly as dreadful as expected.` (**SST**)

🟦 Anchors 🟥 Minimal Robust Extension

Figure 7.11: Examples of Anchors explanations (in blue) and the minimal extension required to make them robust (in red). Examples are classified (without errors) with a 25-input-word CNN (accuracy 0.89). OREs are extracted for knn boxes and k=25.

to be relatively robust w.r.t. said $\mathcal{B}$s. On the contrary, the Anchors extracted for the FC models attain an average precision of 0.996 on SST and 0.975 on Twitter, but only 12.5% of them are robust for the SST case and 7.5% for Twitter. With CNN models, high-quality Anchors are even more brittle: 0% of Anchors are robust on SST reviews and 5.4% on Twitter, despite an average precision of 0.995 and 0.971, respectively.

We remark, however, that Anchors are not designed to provide such robustness guarantees. Our approach becomes helpful in this context because it can *minimally extend* any local explanation to make it robust by using *include constraints* as explained in Section 3. In Figure 7.11, we show a few examples of how, starting from non-robust Anchors explanations, our algorithm can find the minimum number of words to make them provably robust.

7.5 Conclusions

In the final chapter of this thesis, we proposed Optimal Robust Explanations as a novel approach for enhancing the interpretability of NLP models. The introduction of OREs aims to address the limitations of existing methods by leveraging the concept of robustness. In synthesis, OREs provide a concise and comprehensive justification for specific predictions, incorporating both minimality with respect to a given cost function and robustness. OREs further ensure that a model's prediction remains unaffected by any bounded replacement of the omitted features.

The utilization of OREs facilitates various crucial use cases, including detecting biased decisions, identifying and rectifying misclassifications, and correcting non-robust explanations. By employing OREs, users gain valuable insights into the decision-making process of NLP models while ensuring that the explanations provided are both trustworthy and robust.

This chapter demonstrates the broader significance of robustness as a desirable quality in a model, extending beyond its formal guarantee applicability. Incorporating robustness into the interpretability framework enhances the explanations provided by NLP models, affording users greater confidence in the reliability and consistency of the model's behavior. Ultimately, OREs represent a significant step towards achieving a more transparent and understandable NLP landscape.

In contrast, OREs encounter challenges regarding scalability due to the necessity of solving an NP-hard problem to obtain guaranteed explanations. Our research demonstrated that the convergence of the HS routine can be enhanced through the utilization of sparse adversarial attacks. However, it is important to note that this augmentation does not inherently ensure improved performance outcomes.

Chapter 8

Conclusions

In this thesis, we have examined the interplay between language, robustness, and machine learning.

Through our assessment of NLP model robustness (Chapter 4), we have uncovered issues with the current standard definition inherited from computer vision. Despite being widely used, this definition exhibits limitations when applied to language, and in particular to increasingly complex models, as it tends to yield robustness bounds that diminish rapidly. Such shortcomings undermine the reliability and effectiveness of these models, particularly in scenarios where robustness is crucial, such as adversarial environments or real-world applications with high-stakes implications.

This finding has prompted the formulation of a novel concept, termed *semantic robustness* in this study, which specifically addresses a model's ability to handle linguistic phenomena that hold substantial importance in human language comprehension (refer to Chapter 5). We characterized this newly defined notion of semantic robustness, providing a comprehensive understanding of its underlying principles and properties. We conducted an extensive experimental evaluation to assess the effectiveness and validity of this concept, employing diverse architectural configurations. The results of these experiments demonstrate that language modeling, specifically through unsupervised training techniques, offers superior robustness guarantees when compared to models trained using augmented training approaches.

By adopting a post-structuralist approach to language analysis, our research has comprehensively examined the notion of *syntactic robustness* within NLP models. This investigation encompasses an assessment of the model's capacity to capture and encode syntactic structures within its hidden representations consistently. Through analysis and experimentation, we have uncovered compelling evidence regarding the fragility of diverse linguistic representations when subjected to moderate text manipulations that preserve syntactic integrity, as explicated in Chapter 6. These findings underscore the urgent imperative for enhancing robustness in NLP models to address the vulnerabilities observed in processing syntax effectively.

We conclude by showing that robustness, while being per se a valuable property of a model, can be used to rigorously explain, debug and interpret the decisions of a model. With the concept of ORE (Chapter 7), we distill minimal-length input words sufficient to imply a model's decision while being robust to local manipulations. This final chapter demonstrates that robustness can be integrated into explanations, as our findings pave the way for future research endeavors to extend formal robustness guarantees to the realm of explainable deep neural networks, thereby fortifying the foundations of reliable and accountable AI systems.

There are several directions that future research could take based on the findings and contributions of this thesis. One possible avenue is to explore more advanced methods for incorporating robustness into NLP models.

An important avenue of research lies in developing robust generative models that offer control and guarantee the adherence of the output to human-defined constraints. One illustrative example of such models is the syntactically controlled paraphrase generators, which have the potential to generate previously unseen sentences that are consistent and adhere to linguistic norms while simultaneously exhibiting a high degree of linguistic variability.

We believe expanding the concept of *semantic robustness* to include noisy data generators is also crucial. This extension becomes particularly valuable considering the limitations we have observed in the scalability of our template-based generator and the shortcomings inherent in handcrafted or manually distilled samples, as illustrated in Chapter 5.

While our work proposes Optimal Robust Explanations (OREs) to distill optimal-cost input words sufficient to imply a model's decision, several other potential methods could be explored. For example, researchers could investigate the extension of OREs to complex LMs (especially the emerging trend of models provided as-a-service, such as ChatGPT (OpenAI, 2023)) to provide more transparent decision-making processes.

Another promising direction for future research is to investigate the interplay between robustness and fairness in NLP models. Recent studies have shown that NLP models can exhibit biases that reflect underlying societal biases and prejudices, which can lead to harmful downstream effects. Therefore, there is a need for models that are robust to text manipulations and fair and unbiased in their decision-making processes. While recent works show that it is possible to train models that are provably fair (Benussi et al., 2022), future research could explore the development of methods that scale to the size of NLP models, whose complexity and prominence have grown at an unprecedented pace in the last decade.

Future research could aim to design models that develop a sound understanding of the deep structures of language and its connection to the world while being reliable and robust by design. This would require the development of models that can learn to reason about the world based on natural language input, which would be a significant breakthrough in AI.

In summary, there are several exciting directions for future research based on the contributions of this thesis. The proposed methods and principles could be extended to other domains, and more advanced techniques could be explored to enhance the robustness and interpretability of NLP models. Additionally, the interplay between robustness and fairness in NLP models is a crucial area of research that requires further attention. Ultimately, developing reliable and robust models that can reason about the world based on natural language input is a significant challenge that requires continued efforts from the research community.

Appendix

Reproducibility for Chapter 4

Preliminary

The code used to carry out the experiments has been successfully tested on a Linux server running Ubuntu 18.04. The server boasts 90 CPU cores, 6 NVIDIA GTX 2080 Ti GPUs, and 256GB of RAM.

To run the upper bound experiments' code, one must clone the code stored at the following repository: `https://github.com/EmanueleLM/MCTS`. Once the repository is copied, one will find a folder named `MCTS/` containing the code necessary to replicate the simulations, along with a `README` file containing further instructions.

For the lower bound computation, we provide links to the tools used in our experimental evaluation, namely CNN-Cert and POPQORN Boopathy et al. (2019); Ko et al. (2019a). Where necessary, we also provide additional instructions on replicating the experiments described in this chapter.

Dependencies

The code is written in Python and was tested using version 3.8. Additionally, the following packages are required, and the versions we used are reported below:

- Numpy 1.17.2
- Nltk 3.4.5
- Pandas 0.25.1
- Progress 1.5
- Tensorflow 1.14.0
- Torch 1.3.1

- Torchtext 0.4.0

Further, one needs to download the GloVe and GloVeTwitter pre-trained embeddings respectively from `http://nlp.stanford.edu/data/glove.6B.zip` and `http://nlp.stanford.edu/data/glove.twitter.27B.zip`. Once archives have been unzipped, one must move them inside the `./data/embeddings` folder. Please note that naming should be consistent with variables in `ub_CNN_models.py` and `ub_LSTM_models.py` files, i.e., `glove.6B.<DIMS>d.txt` and `glove.twitter.27B.<DIMS>d.txt`, where <DIMS> is the dimensionality of the embedding, which can be specified as a parameter in the upper bound computation. For example, to test `GloVe50d`, the embedding's name should be `glove.6B.50d.txt`.

Lower and Upper Bound Characterization

Lower Bound Computation

To compute the lower bound of a CNN's robustness, one must install CNN-Cert from the following URL: `https://github.com/AkhilanB/CNN-Cert`. This tool was initially designed to compute a certified robustness lower bound for CNNs in computer vision and requires some adaptations to be used with NLP word embeddings. Specifically, one will need to modify the file `cnn_bounds_full_core.py` as described in this link: `https://github.com/IBM/CNN-Cert/issues/4`.

To compute the lower bound of LSTMs' robustness, one must install POPQORN and follow the instructions provided in this link: `https://github.com/ZhaoyangLyu/POPQORN`.

Upper Bound Computation

To run an MCTS experiment on a CNN model, launch from the console:
`python3 ub_CNN_models.py`, while for an LSTM, run `python3 ub_LSTM_models.py`.
Please note that these scripts can accept multiple arguments:

```
-s, --sims: number of Monte Carlo simulations per-vertex
-m, --max-depth: maximum depth of the tree (i..e, number of perturbations)
-e, --eps: max-distance (in L2 norm) for collecting neighbors
-l, --lrate: UTC learning rate
-d, --discount: UTC discount factor
-ed, --emb-dims: embeddings dimension
-hu, --lstm-hu: LSTM hidden units (applicable only to ‘ub_LSTM_models.py‘)
```

The first time one runs the MCTS algorithm, it will take a while to compute and store the neighbors of each input word in a separate file.

Reproducibility for Chapter 5

Preliminary

The code has been successfully tested on Linux Fedora 32, a low-end machine with 8GB of RAM, an Intel Core i5, and no dedicated GPUs.

In order to run the code of the experiments, one needs to clone the following GitHub folder `https://github.com/EmanueleLM/the-king-is-naked`. Once copied, a folder named `the-king-is-naked` will be created; the folder contains the code necessary to replicate a few simulations, alongside a `README` file that provides further details on how to run the code.

Python Dependencies

The code is written in Python and was tested using version 3.8. Additionally, the following packages are required, and the versions we used are reported below:

- gensim 3.8.3
- Keras 2.4.3
- keras-self-attention 0.49.0
- nltk 3.5
- numpy 1.18.4
- pandas 1.1.4
- tensorflow 2.3.1
- torch 1.7.1
- tqdm 4.53.0

In order to install the dependencies, you can run the command
`pip install` `PACKAGENAME==X.Y.Z --user` where `PACKAGENAME` is the name of the missing dependency and `X.Y.Z` is the version, for example,
`pip install numpy==1.18.4 --user`. If no specific version is specified in the previous list, you can run
`pip install PACKAGENAME --user`.

External Dependencies

Additionally, the code we provide relies on DeepBayesHS for experiments that involve comparing a semantically enhanced model to an IBP-robust model (see Chapter 5/Section 5.4). Please note that DeepBayesHS is an external tool with its specific license and must be installed on the system separately. To run DeepBayesHS, one must clone the code from the following folder: `https://github.com/matthewwicker/deepbayesHF`, and train some augmented models. The repository comes with a `README` file that provides instructions on how to train a simple model.

Preparing the pre-trained embeddings

To run the experiments presented in Section 5.4, you need to download and place a word embedding inside the `the-king-is-naked/data/embeddings/` folder.

Before running any experiments from the paper, please rename the embedding to the following name: `custom-embedding-SST.50d.txt`. Please note that this is just a placeholder name; any embedding can be used.

Measuring Semantic Robustness

Train a Model

To train a model - with or without data augmentation - go to the `the-king-is-naked/train/` folder and run the file `train_sst.py`. The command comes along with many command-line arguments, like `architecture`, which specifies what architecture to choose among {`fc, lstm, cnn1d, cnn2d, and attention`}, and `finetune_on_hard_instances`, which enables fine-tuning on hard instances with samples from (Barnes et al., 2019). One can then select the regime of data augmentation with `data_augmentation` (500, 1000 etc.) and choose which semantic rule you want to learn between `negated` (shallow negation), `mixed` (mixed sentiment) and `sarcasm` (sarcasm). Models will be saved under the folder `the-king-is-naked/models/[ARCHITECTURE]/[RULE]` where `[ARCHITECTURE]` is either {`fc, cnn1d, cnn2d, lstm, attention`} while `[RULE]` is either {`vanilla, shallow_negation, mixed_sentiment, sarcasm`}.

Test a Model

To test the semantic robustness of a trained model, navigate to the `the-king-is-naked/train/` directory and execute the `test_sst.py` script. This script provides several options, such as setting the test set to `linguistic_phenomena` (our benchmark), `sentiment_not_solved` (Barnes et al., 2019), or `sst` dataset using the `test_type` variable. The `test_augmented_networks` variable can be set to either test semantically-augmented networks or *vanilla* networks. The `test_IBP` variable is used to select the networks from the IBP folder, and the `test_rule` variable can be used to select the semantic rule one wants to test.

BERT Experiments

Finally, the `the-king-is-naked\train\train_BERT.py` script allows to train BERT on SST dataset.

The same model can then be tested to measure its semantic robustness: to do so, run
`the-king-is-naked\verify\semantic_robustness_bert.py`, setting `test_type` to either "sentiment_not_solved" (Barnes et al., 2019) or "linguistic_phenomena" (our benchmark), `linguistic_phenomena` to either
texttt{shallow_negation, mixed_sentiment, sarcasm} and `test_rule1`, `test_rule2` to respectively `("negated","negated")`, `("mixed", "mixed")` or `("irony","sarcasm")` depending on the value of `linguistic_phenomena` you chose previously.

Reproducibility for Chapter 6

Preliminary

This code has been successfully tested on a Linux server with Ubuntu 18.04 LTS. The server boasts 90 CPU cores, 6 NVIDIA GTX 2080 Ti GPUs, and 256GB of RAM.

To run the experiments' code, one must first clone the repository at the following address: `https://github.com/EmanueleLM/emergent-linguistic-structures`. Upon cloning, a folder named `robust-linguistic-structures` is created. The code specifically related to Chapter 6 can be executed under the folder path `robust-linguistic-structures\verify \MLM_internals \syntax-integrity`.

Python Dependencies

We coded the framework with Python 3.6.9: dependencies can be installed via the official pip installer (i.e., *package installer for Python*). Since this code requires numerous dependencies to be installed, we suggest installing a virtual environment. It follows the list of Python dependencies alongside the version we used to code our framework:

- numpy 1.18.5
- tensorflow 2.3.1
- torch 1.3.1
- seaborn 0.11.2
- tqdm 4.60.0
- Keras 1.1.2
- scikit-image 0.18.1
- scikit-learn 0.24.2
- scikit-learn-extra 0.2.0
- scipy 1.6.3
- transformers 4.16.2
- matplotlib 3.4.2

- networkx 2.5.1
- pandas 1.3.4
- nltk 3.5
- pickleshare 0.7.5

In order to install these or other packages, you can run the command `pip install PACKAGENAME==X.Y.Z --user` where `PACKAGENAME` is the name of the missing dependency and `X.Y.Z` is the version, for example, `pip install numpy==1.18.4 --user`. If no specific version is specified in the previous list, one can run
`pip install PACKAGENAME --user`.

Robustness Measurement of All the Probing Tasks

One can run the robustness training and measurements for all the probing tasks through the Python script `probing_robustness.py`. In order to do so, cd into `robust-linguistic-structures\verify \MLM_internals \syntax-integrity` folder and launch it.

This script comes with many command-line arguments, whose semantics is explained through the `--help` command. An example of an experiment that involves several command line arguments run the following command:

```
python3 probing_robustness.py --seed 42 --mlm bert --dataset ted
 --layer-p1 -5 --layer-t1 -5 --perturbation-budget 1 --copos True
--perturbation-scenario worst --lp-norm 2 --wordnet-mode True
```
.

Results and details of each of these experiments are collected under the `robust-linguistic-structures\verify \MLM_internals \syntax-integrity \results \robustness` folder.

The folder contains many scripts that allow scheduling the execution for multiple models and datasets.

Reproducibility for Chapter 7

Preliminary

This code has been successfully tested on a Linux server with Ubuntu 18.04 LTS. The server is equipped with 90 CPU cores, 6 NVIDIA GTX 2080 Ti GPUs, and 256GB of RAM.

In order to run the code of the experiments, one first needs to clone the Github repository at the following link `https://github.com/EmanueleLM/OREs`. Having completed this operation, one should find a folder named `OREs_anonym` that contains the code necessary to replicate a few simulations of the experiments presented in Chapter 7.

Python Dependencies

The framework was developed using Python 3.8, and the dependencies can be installed using the official pip installer for Python. Below is the list of Python dependencies we used to code our framework, along with their respective versions:

- numpy 1.18.5
- python-sat
- tensorflow 2.3.0
- Keras 2.3.1

In order to install these or other packages, you can run the command `pip install PACKAGENAME==X.Y.Z --user` where `PACKAGENAME` is the name of the missing dependency and `X.Y.Z` is the version, for example, `pip install numpy==1.18.4 --user`. If no specific version is specified in the previous list, one can run
`pip install PACKAGENAME --user`.

External Dependencies

We use Marabou, an SMT-based tool that can answer queries about neural networks and their properties (Katz et al., 2019b), to assess the robustness of a network against perturbations. Our experiments were conducted using the software that can be downloaded and installed from this address: `https://github.com/NeuralNetworkVerification/Marabou/`. Specifically, all the experiments in the paper were performed using the version with the commit hash `228234ba`, which can be found at this link: `https://github.com/NeuralNetworkVerification/Marabou/tree/228234ba6c8242bc2463f4f0678164ca4c4ede7e`.

One should start by cloning Marabou into the `OREs_anonym` folder, which should be at the same level as the `Explanations` folder. The Marabou repository on GitHub provides detailed instructions on installing and setting up the tool properly. Upon the completion of the previous tasks, one should have the `Marabou` and `Explanations` subdirectories within the `OREs_anonym` folder.

Run Experiments

To launch an experiment that extracts an Optimal Robust Explanation (ORE), go to the
`Explanations\abduction_algorithms\experiments\{DATASET}\` folder, where `{DATASET}` is either `SST Twitter` or `IMDB`, and run one of the python files (`.py` extension). We report a few usage examples.

ORE for a Fully Connected Model on a Sample Review

We now show how to extract an ORE from a sample review and an already trained FC model with 25 input words, using the k Nearest Neighbours bounding box technique with k=10. You first have to navigate to
`Explanations\abduction_algorithms\experiments\SST\` folder and run the following command:

```
python3 smallest_explanation_SST_fc_knn_linf.py -k 10 -w 5 -n 25
 -i 'This is a very bad movie'
```

The algorithm identifies the words `is` and `bad` as an ORE. Additionally, results are logged and stored in a folder inside `results\HS-clear` (the complete path that is reported in the logs of the execution).

Solving a hard instance with Adversarial Attacks

Suppose one wants to extract an ORE from a Twitter text using a CNN model with 25 input words and the k Nearest Neighbours bounding box technique with k=8. Since this instance will not easily converge due to the complexity of the model, one can improve the convergence by employing the sparse adversarial attacks algorithm. It is enough to specify the number of attacks that one wants to launch at each iteration of the HS routine (parameter -a/–adv). One has to navigate to the

`Explanations\abduction_algorithms\experiments\Twitter\` folder and run the following:

```
python3 smallest_explanation_Twitter_cnn2d_knn_linf.py -k 8 -w 5
 -n 25 -a 500 -i 'Spencer is not a good guy'
```

The algorithm identifies the words `Spencer` and `not` as an ORE (alongside a `PAD` token that highlights a problem with the model robustness). Results are stored in a folder inside `results\HS-clear` (the complete path that is reported in the logs of the execution).

Detecting Decision Bias Using Cost Functions

Suppose one wants to check whether an explanation always contains the name of a character, an actor, or a director. To do this, one can run the following command:

```
python3 smallest_HScost_explanation_SST_fc_knn_linf.py -k 27 -w 5
-a 0 -i "Austin Powers in Goldmember has the right stuff for
summer entertainment and has enough laughs to sustain interest to
the end" -u False -x 'austin,powers,goldmember'
```

If the excluded words still appear in the ORE, then the model suffers from a decision bias.

If one wants to exclude a word from the explanation permanently, they should run the following command:

```
python3 smallest_cost_explanation_SST_fc_knn_linf_alternate_cost_exclude.py
-k 27 -w 5 -i "Austin Powers in Goldmember has the right stuff for
summer entertainment and has enough laughs to sustain interest to
the end"
```

Run multiple instances in parallel

One can run several ORE instances in parallel (e.g., on a server) by launching the `run-exp_MODEL_DATASET_knn_linf.sh` script in each `Experiments\DATASET` folder, where DATASET is either ‘IMDB‘, ‘SST‘ or ‘Twitter‘ and MODEL is either ‘fc‘ (FC) or ‘cnn‘ (CNN).[1]

[1] You can use the `screen` command (available on any Linux distribution) to manage the different instances.

Bibliography

Afra Alishahi, Grzegorz Chrupała, and Tal Linzen. Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop. *Natural Language Engineering*, 25(4):543–557, 2019.

Basemah Alshemali and Jugal Kalita. Improving the reliability of deep neural networks in NLP: A review. *Knowledge-Based Systems*, 191:105210, 2020.

Izzat Alsmadi, Kashif Ahmad, Mahmoud Nazzal, Firoj Alam, Ala Al-Fuqaha, Abdallah Khreishah, and Abdulelah Algosaibi. Adversarial attacks and defenses for social network text processing applications: Techniques, challenges and future research directions. *ArXiv preprint*, abs/2110.13980, 2021.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium, 2018. Association for Computational Linguistics.

Kyriakos D Apostolidis and George A Papakostas. A survey on adversarial deep learning robustness in medical image analysis. *Electronics*, 10(17):2132, 2021.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany, 2016. Association for Computational Linguistics.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

S. Bach et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *ArXiv preprint*, abs/1604.00825, 2016.

Jeremy Barnes, Lilja Øvrelid, and Erik Velldal. Sentiment analysis is not solved! assessing and probing sentiment classification. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 12–23, Florence, Italy, 2019. Association for Computational Linguistics.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online, 2020. Association for Computational Linguistics.

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*, pages 610–623, 2021.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

Elias Benussi, Andrea Patane, Matthew Wicker, Luca Laurenti, and Marta Kwiatkowska. Individual fairness guarantees for neural networks. *ArXiv preprint*, abs/2205.05763, 2022.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Arno Blaas, Andrea Patane, Luca Laurenti, Luca Cardelli, Marta Kwiatkowska, and Stephen J. Roberts. Adversarial robustness guarantees for classification with gaussian processes. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3372–3382. PMLR, 2020.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4349–4357, 2016.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *ArXiv preprint*, abs/2108.07258, 2021.

Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3240–3247. AAAI Press, 2019.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. Adversarial filters of dataset biases. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1078–1088. PMLR, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

S. Chakraborty et al. Interpretability of deep learning models: a survey of results. In *SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, pages 1–6. IEEE, 2017.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The expressive power of word embeddings. *arXiv:1301.3226*, 2013.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 251–268, Cham, 2017. Springer International Publishing.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3601–3608. AAAI Press, 2020.

François Chollet and others. Keras: The Python Deep Learning library, 2018.

François Chollet et al. keras, 2015.

Noam Chomsky. *Syntactic structures*. De Gruyter Mouton, 2009.

Noam Chomsky. 153A Minimalist Program for Linguistic Theory. In *The Minimalist Program*. The MIT Press, 12 2014a.

Noam Chomsky. *Aspects of the Theory of Syntax*, volume 11. MIT press, 2014b.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, 2018. Association for Computational Linguistics.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

Ian Covert, Scott M. Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Francesco Croce, Maksym Andriushchenko, Naman D. Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: A versatile framework for query-efficient sparse black-box adversarial attacks. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6437–6445. AAAI Press, 2022.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable AI for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, Suzhou, China, 2020. Association for Computational Linguistics.

Adnan Darwiche. Three modern roles for logic in ai. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 229–243, 2020.

Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. *ArXiv preprint*, abs/2002.09284, 2020.

Congyue Deng and Yi Tian. Towards understanding the trade-off between accuracy and adversarial robustness. *abs/2002.10716*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, 2019a. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, 2019b. Association for Computational Linguistics.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, Vancouver, Canada, 2017. Association for Computational Linguistics.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. Towards robustness against natural language word substitutions. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Susan T Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology (ARIST)*, 38:189–230, 2004.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods*, pages 121–138, 2018.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M. Bender. Towards linguistically generalizable NLP systems: A workshop and shared task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 1–10, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online, 2021. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. Mathematical capabilities of chatgpt. *abs/2301.13867*, 2023.

Edward Gibson, Richard Futrell, Steven P Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in cognitive sciences*, 23(5):389–407, 2019.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

A. Go et al. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 2009.

Yoav Goldberg. Assessing bert's syntactic abilities. *ArXiv preprint*, abs/1901.05287, 2019.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Gregory Goren, Oren Kurland, Moshe Tennenholtz, and Fiana Raiber. Ranking robustness under adversarial document manipulations. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 395–404. ACM, 2018.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *ArXiv preprint*, abs/1810.12715, 2018.

Shreya Goyal, Sumanth Doddapaneni, Mitesh M Khapra, and Balaraman Ravindran. A survey in adversarial defences and robustness in NLP. *ArXiv preprint*, abs/2203.06414, 2022.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5141–5148. AAAI Press, 2018.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, 2009. Association for Computational Linguistics.

Yanfen Hao and Tony Veale. An ironic fist in a velvet glove: Creative misrepresentation in the construction of ironic similes. *Minds and Machines*, 20(4): 635–650, 2010.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online, 2020. Association for Computational Linguistics.

Karl Moritz Hermann, Edward Grefenstette, and Phil Blunsom. "not not bad" is not "bad": A distributional account of negation. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 74–82, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jie Hong, Pengfei Fang, Weihao Li, Tong Zhang, Christian Simon, Mehrtash Harandi, and Lars Petersson. Reinforced attention for few-shot learning and beyond. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 913–923. Computer Vision Foundation / IEEE, 2021.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529, Florence, Italy, 2019. Association for Computational Linguistics.

Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

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. 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)*, pages 4083–4093, Hong Kong, China, 2019. Association for Computational Linguistics.

Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. Reducing sentiment bias in language models via counterfactual evaluation. In *Findings of the Association for*

Computational Linguistics: EMNLP 2020, pages 65–83, Online, 2020. Association for Computational Linguistics.

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

Xuanxiang Huang and Joao Marques-Silva. The inadequacy of shapley values for explainability. *ArXiv preprint*, abs/2302.08160, 2023.

Aminul Huq, Mst Pervin, et al. Adversarial attacks and defense on texts: A survey. *ArXiv preprint*, abs/2005.14108, 2020.

A. Ignatiev et al. On finding minimum satisfying assignments. In *CP*, volume 9892 of *LNCS*, pages 287–297. Springer, 2016.

A. Ignatiev et al. A SAT-based approach to learn explainable decision sets. In *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, pages 627–645. Springer, 2018.

A. Ignatiev et al. On validating, repairing and refining heuristic ml explanations. *ArXiv preprint*, abs/1907.02509, 2019a.

Alexey Ignatiev, António Morgado, Georg Weissenbacher, and João Marques-Silva. Model-based diagnosis with multiple observations. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1108–1115. ijcai.org, 2019b.

Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. Abduction-based explanations for machine learning models. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 1511–1519. AAAI Press, 2019c.

Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. On relating explanations and adversarial examples. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15857–15867, 2019d.

Sarthak Jain and Byron C. Wallace. Attention is not Explanation. 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)*, pages 3543–3556, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, 2019. Association for Computational Linguistics.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 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)*, pages 4129–4142, Hong Kong, China, 2019. Association for Computational Linguistics.

Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez-Moreno, and Yonghui Wu. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4485–4495, 2018.

Steven Johnson and Nikita Iziev. A.i. is mastering language. should we trust what it says?, Apr 2022.

G. Katz et al. The Marabou framework for verification and analysis of deep neural networks. In *CAV*, volume 11561 of *LNCS*, pages 443–452. Springer, 2019a.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017a.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017b.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017c.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pages 443–452. Springer, 2019b.

Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhanderi, Robert Belfer, Nirmal Kanagasabai, Roman Sarrazingendron, Rohit Verma, and Derek Ruths. Sentiment analysis: It's complicated! In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1886–1895, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

Eugene Kharitonov and Rahma Chaabouni. What they do when in doubt: a study of inductive biases in seq2seq learners. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online, 2021. Association for Computational Linguistics.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th*

International Conference on Machine Learning, volume 139 of *Proceedings of Machine Learning Research*, pages 5562–5571. PMLR, 18–24 Jul 2021.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, 2014. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Ching-Yun Ko, Zhaoyang Lyu, Lily Weng, Luca Daniel, Ngai Wong, and Dahua Lin. POPQORN: quantifying robustness of recurrent neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3468–3477. PMLR, 2019a.

Ching-Yun Ko, Zhaoyang Lyu, Lily Weng, Luca Daniel, Ngai Wong, and Dahua Lin. POPQORN: quantifying robustness of recurrent neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3468–3477. PMLR, 2019b.

Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, et al. Chatgpt: Jack of all trades, master of none. *abs/2302.10724*, 2023.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on Machine Learning*, pages 282–293. Springer, 2006.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. Adversarial examples for natural language classification problems. *arxiv pre-print*, 2018.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 957–966. JMLR.org, 2015.

Emanuele La Malfa and Marta Kwiatkowska. The king is naked: On the notion of robustness for natural language processing. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11047–11057. AAAI Press, 2022.

Emanuele La Malfa, Min Wu, Luca Laurenti, Benjie Wang, Anthony Hartshorn, and Marta Kwiatkowska. Assessing robustness of text classification through maximal safe radius computation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2949–2968, Online, 2020. Association for Computational Linguistics.

Emanuele La Malfa, Rhiannon Michelmore, Agnieszka M. Zbrzezny, Nicola Paoletti, and Marta Kwiatkowska. On guaranteed optimal robust explanations for nlp models. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2658–2665. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/366. URL `https://doi.org/10.24963/ijcai.2021/366`. Main Track.

Emanuele La Malfa, Matthew Wicker, and Marta Kiatkowska. Emergent linguistic structures in neural networks are fragile. *ArXiv preprint*, abs/2210.17406, 2022.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, 2016. Association for Computational Linguistics.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online, 2020. Association for Computational Linguistics.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4208–4215. ijcai.org, 2018.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online, 2020. Association for Computational Linguistics.

Seppo Linnainmaa. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.* PhD thesis, Master's Thesis (in Finnish), Univ. Helsinki, 1970.

Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to mlps. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 9204–9215, 2021.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692, 2019.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774, 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, 2011a. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, 2011b. Association for Computational Linguistics.

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13516–13524. AAAI Press, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *ArXiv preprint*, abs/2301.06627, 2023.

Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

Joao Marques-Silva. Logic-based explainability in machine learning. *ArXiv preprint*, abs/2211.00541, 2022.

João Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 12342–12350. AAAI Press, 2022.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

George A Miller. *WordNet: An electronic lexical database.* MIT press, 1998.

Marvin Minsky and Seymour A Papert. *Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry.* MIT press, 2017.

Tom M Mitchell. *The need for biases in learning generalizations.* Department of Computer Science, Laboratory for Computer Science Research . . . , 1980.

Christoph Molnar. *Interpretable machine learning.* Lulu. com, 2020.

John Morris. Second-order NLP adversarial examples. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 228–237, Online, 2020. Association for Computational Linguistics.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online, 2020a. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online, 2020b. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California, 2016. Association for Computational Linguistics.

N. Narodytska et al. Assessing heuristic machine learning explanations with model counting. In *SAT*, pages 267–278. Springer, 2019.

Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy, 2019. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia, 2016. European Language Resources Association (ELRA).

Marwan Omar, Soohyeon Choi, DaeHun Nyang, and David Mohaisen. Robust natural language processing: Recent advances, challenges, and future directions. *ArXiv preprint*, abs/2201.00768, 2022.

OpenAI. GPT-4 technical report. *ArXiv preprint*, abs/2303.08774, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

Kevin Patel and Pushpak Bhattacharyya. Towards lower bounds on number of dimensions for word embeddings. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 31–36, Taipei, Taiwan, 2017. Asian Federation of Natural Language Processing.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014a. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014b. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online, 2020. Association for Computational Linguistics.

Nicolas Pröllochs, Stefan Feuerriegel, and Dirk Neumann. Enhancing sentiment analysis of financial news by detecting negation scopes. In *2015 48th Hawaii International Conference on System Sciences*, pages 959–968. IEEE, 2015.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online, July 2020. Association for Computational Linguistics.

Shilin Qiu, Qihe Liu, Shijie Zhou, and Wen Huang. Adversarial attack and defense technologies in natural language processing: A survey. *Neurocomputing*, 492:278–307, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? *arXiv preprint arXiv:2005.00719*, 2020.

R. Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1): 57–95, 1987.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM, 2016.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1527–1535. AAAI Press, 2018a.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia, 2018b. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online, 2020. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. Token-modification adversarial attacks for natural language processing: A survey. *ArXiv preprint*, abs/2103.00676, 2021.

Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2651–2659. ijcai.org, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Ruslan Salakhutdinov. Deep learning. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM, 2014.

Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

Martin Schrimpf, Idan Asher Blank, Greta Tuckute, Carina Kauf, Eghbal A Hosseini, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative modeling converges on predictive processing. *Proceedings of the National Academy of Sciences*, 118(45):e2105646118, 2021.

Hinrich Schütze. Automatic word sense discrimination. *Computational linguistics*, 24 (1):97–123, 1998.

Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. *ArXiv preprint*, abs/2004.02082, 2020a.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b.

Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. Text data augmentation for deep learning. *Journal of big Data*, 8:1–34, 2021.

Chandan Singh, W. James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, 2013a. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, 2013b. Association for Computational Linguistics.

Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *ArXiv preprint*, abs/2003.04985, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014a.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014b.

Andrea Tocchetti, Lorenzo Corti, Agathe Balayn, Mireia Yurrita, Philip Lippmann, Marco Brambilla, and Jie Yang. Ai robustness: a human-centered perspective on technological challenges and opportunities. *ArXiv preprint*, abs/2210.08906, 2022.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. Classification of short texts by deploying topical annotations. In *European Conference on Information Retrieval*, pages 376–387. Springer, 2012.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018a.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6369–6379, 2018b.

Xuezhi Wang, Haohan Wang, and Diyi Yang. 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*, pages 4569–4586, Seattle, United States, 2022. Association for Computational Linguistics.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. Towards fast computation of certified robustness for relu networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5273–5282. PMLR, 2018a.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018b.

Jennifer C. White, Tiago Pimentel, Naomi Saphra, and Ryan Cotterell. A non-linear structural probe. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 132–138, Online, 2021. Association for Computational Linguistics.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68, Uppsala, Sweden, 2010. University of Antwerp.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5283–5292. PMLR, 2018.

Min Wu and Marta Kwiatkowska. Robustness guarantees for deep neural networks on videos. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 308–317. IEEE, 2020.

Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. A game-based approximate verification of deep neural networks with provable guarantees. *Theoretical Computer Science*, 807:298–329, 2020a.

Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. A game-based approximate verification of deep neural networks with provable guarantees. *Theoretical Computer Science*, 807:298 – 329, 2020b. ISSN 0304-3975.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. Polyjuice: Automated, general-purpose counterfactual generation. *ArXiv preprint*, abs/2101.00288, 2021.

Ying Xu, Xu Zhong, Antonio Jose Jimeno Yepes, and Jey Han Lau. Elephant in the room: An evaluation framework for assessing adversarial examples in NLP. *ArXiv preprint*, abs/2001.07820, 2020.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4944–4953, 2018.

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.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015a.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015b.

George Kingsley Zipf. *The psycho-biology of language: An introduction to dynamic philology*. Routledge, 2013.

Daniel Zoran, Mike Chrzanowski, Po-Sen Huang, Sven Gowal, Alex Mott, and Pushmeet Kohli. Towards robust image classification using sequential attention models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9480–9489. IEEE, 2020.