Certified Robustness in NLP Under Bounded Levenshtein Distance

Elias Abad Rocamora [1] **Grigorios G. Chrysos** [2] **Volkan Cevher** [1]

Abstract

Natural Language Processing (NLP) models suffer from small perturbations, that if chosen adversarially, can dramatically change the output of the model. Verification methods can provide robustness certificates against such adversarial perturbations, by computing a sound lower bound on the robust accuracy. Nevertheless, existing verification methods in NLP incur in prohibitive costs and cannot practically handle Levenshtein distance constraints. We propose the first method for computing the Lipschitz constant of convolutional classifiers with respect to the Levenshtein distance. We use this Lipschitz constant estimation method for training 1-Lipschitz classifiers. This enables computing the certified radius of a classifier in a single forward pass. Our method, `LipsLev`, is able to obtain 38.80% and 13.93% verified accuracy at distance 1 and 2 respectively in the AG-News dataset. We believe our work can open the door to more efficiently training and verifying NLP models.

1. Introduction

Despite the impressive performance of NLP models (Sutskever et al., 2014; Zhang et al., 2015; Devlin et al., 2019), simple corruptions like typos or synonym substitutions are able to dramatically change the prediction of the model (Belinkov and Bisk, 2018; Alzantot et al., 2018). With newer attacks in NLP becoming stronger (Hou et al., 2023), verification methods become relevant for providing future-proof robustness certificates (Liu et al., 2021).

Constraints on the Levenshtein distance (Levenshtein et al., 1966) provide a good description of the perturbations a model should be robust to (Morris et al., 2020), while strong attacks incorporate such constraints (Gao et al., 2018; Ebrahimi et al., 2018; Liu et al., 2022; Abad Rocamora et al., 2024). Despite the success of verification methods in the text domain, existing methods can only certify probabilistically via randomized smoothing (Cohen et al., 2019; Ye et al., 2020; Huang et al., 2023), or can only handle different specifications such as replacements of characters/words, stop-word removal or word duplication (Huang et al., 2019; Jia et al., 2019; Shi et al., 2020; Bonaert et al., 2021; Zhang et al., 2021).

Table 1. **State of the art in Levenstein distance verification and our contributions:** `LipsLev` is the first to verify deterministically against Levenshtein distance constraints in a single forward pass.

Method	Insertions/ deletions	Deterministic	Single forward pass
Huang et al. (2019)	✗	✓	✗
Huang et al. (2023)	✓	✗	✗
`LipsLev` (Ours)	✓	✓	✓

On the performance side, most successful certification methods rely on Interval Bound Propagation (IBP) (Moore et al., 2009), which in the text domain requires multiple forward passes through the first layers of the model (Huang et al., 2019), unlike in the image domain where a single forward pass is enough for verification (Wang et al., 2018). Moreover, IBP has been shown to provide a suboptimal verified accuracy in the image domain (Wang et al., 2021).

In the image domain, a popular approach to get fast robustness certificates is computing upper bounds on the Lipschitz constant of classifiers, and using this information to directly verify with a single forward pass (Hein and Andriushchenko, 2017; Tsuzuku et al., 2018; Latorre et al., 2020; Xu et al., 2022). These methods cannot be trivially applied in NLP because they assume the input to be in an ℓ_p space such $\mathbb{R}^d$, which is not the case of text input, where the input length can vary and inputs are discrete (characters). Therefore, we need to rethink Lipschitz verification for NLP.

In this work, we introduce the first method able to provide deterministic Levenshtein distance certificates. This is achieved by computing the Lipschitz constant of convolutional classifiers with respect to the ERP distance (Chen

[1]LIONS, École Polytechnique Fédérale de Lausanne, Switzerland [2]Department of Electrical and Computer Engineering, University of Wisconsin-Madison, USA. Correspondence to: Elias Abad Rocamora <elias.abadrocamora@epfl.ch>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

and Ng, 2004). This Lipschitz constant estimates allow enforcing 1-Lipschitzness during training in order to achieve a larger verified accuracy. This allows obtaining 38.80% and 18.69% verified accuracy in AG-News and SST-2 at Levenshtein distance 1. Moreover, our method is the only one able to verify for radiuses larger than 1. We set the foundations for Lipschitz verification in NLP and we believe our method can be extended to more complex models.

Notation: We use uppercase bold letters for matrices $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, lowercase bold letters for vectors $\boldsymbol{x} \in \mathbb{R}^m$ and lowercase letters for numbers $x \in \mathbb{R}$. Accordingly, the i^{th} row and the element in the i, j position of a matrix $\boldsymbol{X}$ are given by $\boldsymbol{x}_i$ and x_{ij} respectively. We use the shorthand $[n] = \{0, 1, \cdots, n-1\}$ for any natural number n. Given two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times d}$, the concatenation operator is denoted as $\boldsymbol{A} \oplus \boldsymbol{B} = \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix} \in \mathbb{R}^{(m+n)\times d}$. Concatenating with the empty sequence $\emptyset$ results in the identity $\boldsymbol{A} \oplus \emptyset = \boldsymbol{A}$. We denote as $\boldsymbol{A}_{2:} \in \mathbb{R}^{(m-1)\times d}$ the matrix obtained by removing the first row. We denote the zero vector as $\mathbf{0}$ with dimensions appropriate to context. We use the operator $|\cdot|$ for the size of sets, e.g., $|\mathcal{S}(\Gamma)|$ and the length of sequences, e.g., for $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, we have $|\boldsymbol{X}| = m$.

2. Preliminaries

Let $\mathcal{S}(\Gamma) = \{c_1 c_2 \cdots c_m : c_i \in \Gamma \ \forall m \in \mathbb{N} \setminus 0\}$ be the space of sequences of characters in the alphabet set Γ. We represent sentences $\boldsymbol{S} \in \mathcal{S}(\Gamma)$ as sequences of one got vectors, i.e., $\boldsymbol{S} \in \{0,1\}^{m\times|\Gamma|} : ||\boldsymbol{s}_i||_1 = 1, \ \ \forall i \in [m]$. Given a classification model $\boldsymbol{f} : \mathcal{S}(\Gamma) \to \mathbb{R}^o$ assigning scores to each of the o classes, the predicted class for some $\boldsymbol{S} \in \mathcal{S}(\Gamma)$ is given by $\hat{y} = \arg\max_{i\in[o]} f(\boldsymbol{S})_i$. Our goal is to check whether for a given pair $(\boldsymbol{S}, y) \in (\mathcal{S}(\Gamma) \times [o])$:

$$f(\boldsymbol{S}')_y - \max_{\hat{y}\neq y} f(\boldsymbol{S}')_{\hat{y}} > 0, \ \forall \boldsymbol{S}' \in \mathcal{S}(\Gamma) : d_{\text{Lev}}(\boldsymbol{S}, \boldsymbol{S}') \leq k \,, \tag{1}$$

where d_{Lev} is the Levenshtein distance (Levenshtein et al., 1966). The Levenshtein distance is defined as follows:

$$d_{\text{Lev}}(\boldsymbol{S}, \boldsymbol{S}') := \begin{cases} |\boldsymbol{S}| & \text{if } |\boldsymbol{S}'| = 0 \\ |\boldsymbol{S}'| & \text{if } |\boldsymbol{S}| = 0 \\ d_{\text{Lev}}(\boldsymbol{S}_{2:}, \boldsymbol{S}'_{2:}) & \text{if } \boldsymbol{s}_1 = \boldsymbol{s}'_1 \\ 1 + \min \begin{Bmatrix} d_{\text{Lev}}(\boldsymbol{S}_{2:}, \boldsymbol{S}'_{2:}) \\ d_{\text{Lev}}(\boldsymbol{S}_{2:}, \boldsymbol{S}') \\ d_{\text{Lev}}(\boldsymbol{S}, \boldsymbol{S}'_{2:}) \end{Bmatrix} & \text{otherwise} \,. \end{cases}$$

The Levenshtein distance captures the number of character replacements, insertions or deletions needed in order to transform $\boldsymbol{S}$ into $\boldsymbol{S}'$ and vice-versa. Such constraints are employed in popular NLP attacks in order to enforce the imperceptibility of the attack (Gao et al., 2018; Ebrahimi et al., 2018; Liu et al., 2022; Abad Rocamora et al., 2024) following the findings of Morris et al. (2020).

3. Method

In Section 3.1 we cover the verification procedure once the Lipschitz constant of a classifier is known. In Section 3.2 we cover the convolutional architectures employed in Huang et al. (2019) and our Lipschitz constant estimation for them. Lastly, we introduce our training strategy in order to achieve non-trivial verified accuracy in Section 3.3. We defer our proofs to Appendix D.

3.1. Lipschitz constant based verification

Motivated by the success and efficiency of Lipschitz constant based certification in vision tasks (Huang et al., 2021; Xu et al., 2022), we propose a method of this kind that can handle previously studied models in the character-level classification task (Huang et al., 2019), and provide Levenshtein distance certificates.

Our goal is to compute the local Lipschitz constant. Let $g_{y,\hat{y}}(\boldsymbol{S}) = f(\boldsymbol{S})_y - f(\boldsymbol{S})_{\hat{y}}$ be the margin function for classes y and $\hat{y}$, we would like to have for some $\boldsymbol{S}$:

$$|g_{y,\hat{y}}(\boldsymbol{S}) - g_{y,\hat{y}}(\boldsymbol{S}')| \leq G_{y,\hat{y}} \cdot d_{\text{Lev}}(\boldsymbol{S}, \boldsymbol{S}') \ \ \forall \boldsymbol{S}' \in \mathcal{S}(\Gamma) \,, \tag{2}$$

for some $G_{y,\hat{y}} \in \mathbb{R}^+$. Given Eq. (2) is satisfied, the maximum distance up to which we can verify Eq. (1), is lower bounded by:

$$\left.\begin{matrix} \max_{k\in\{0\}\cup\mathbb{N}} & k \\ \text{s.t.} & g_{y,\hat{y}}(S') > 0 \ \ \forall S' : d_{\text{Lev}}(S, S') \leq k \end{matrix}\right\} \geq \left\lfloor \frac{g_{y,\hat{y}}(S)}{G_{y,\hat{y}}} \right\rfloor . \tag{3}$$

Let $k^{\star}_{y,\hat{y}}(\boldsymbol{S}) := \left\lfloor \frac{g_{y,\hat{y}}(\boldsymbol{S})}{G_{y,\hat{y}}} \right\rfloor$, we denote $k^{\star}_{y}(\boldsymbol{S}) := \min_{\hat{y}\neq y} k^{\star}_{y,\hat{y}}(\boldsymbol{S})$ to be the *certified radius*.

3.2. Lipschitz constant estimation for convolutional classifiers

Let $\boldsymbol{S} \in \mathcal{S}(\Gamma)$ be a sequence of one-hot vectors, our classifier is defined as:

$$\boldsymbol{f}(\boldsymbol{S}) = \left(\textstyle\sum_{i=1}^{m+l\cdot(q-1)} f_i^{(l)}(\boldsymbol{S})\right) \boldsymbol{W}, \text{where}$$
$$\boldsymbol{f}^{(j)}(\boldsymbol{S}) = \begin{cases} \boldsymbol{\sigma}\left(\boldsymbol{C}^{(j)}\left(\boldsymbol{f}^{(j-1)}(\boldsymbol{S})\right)\right) & \forall j = 1, \cdots, l \\ \boldsymbol{S}\boldsymbol{E} & j = 0 \end{cases}, \tag{4}$$

where $\boldsymbol{E} \in \mathbb{R}^{v\times d}$ is the embeddings matrix, $\boldsymbol{C}^{(i)}, \forall i = 1, \cdots, l$ are convolutional layers with kernel size q and hidden dimension k. σ is the ReLU activation function and $\boldsymbol{W} \in \mathbb{R}^{k\times o}$ is the last classification layer.

Our approach to estimate the global Lipschitz constant of such a classifier is to compute the Lipschitz constant of each layer. Then, since the overall function in Eq. (4) is the sequential composition of all of the layers, we can just multiply the Lipschitz constants to obtain the global one. However, in order to be able to do this, we need some metric with respect to which we can compute the Lipschitz constant. The Levenshtein distance cannot be applied, as it can only measure distances between one-hot vectors and the outputs of intermediate layers are sequences of real vectors. For this task, we select the *ERP distance* (Chen and Ng, 2004):

Definition 3.1 (ERP distance (Chen and Ng, 2004))**.** Let $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times d}$ be two sequences of m and n real vectors respectively. The ERP distance is defined as:

$$d_{\text{ERP}}(\boldsymbol{A},\boldsymbol{B}) = \begin{cases} \sum_{i=1}^{m} ||\boldsymbol{a}_i||_p & \text{if } n=0 \ (\boldsymbol{B}=\emptyset) \\ \sum_{i=1}^{n} ||\boldsymbol{b}_i||_p & \text{if } m=0 \ (\boldsymbol{A}=\emptyset) \\ \min \begin{Bmatrix} ||\boldsymbol{a}_1||_p + d_{\text{ERP}}(\boldsymbol{A}_{2:},\boldsymbol{B}), \\ ||\boldsymbol{b}_1||_p + d_{\text{ERP}}(\boldsymbol{A},\boldsymbol{B}_{2:}), \\ ||\boldsymbol{a}_1 - \boldsymbol{b}_1||_p + d_{\text{ERP}}(\boldsymbol{A}_{2:},\boldsymbol{B}_{2:}) \end{Bmatrix} & \text{otherwise} \end{cases}$$

The ERP distance is a natural extension of the Levenshtein distance for sequences of real valued vectors. In fact, in the case we compare sequences of one-hot vectors and we set $p=\infty$, we recover the Levenshtein distance, see Lemma S4.

In the following we define a useful representation of convolutional layers.

Definition 3.2 (1D Convolutional layer with zero padding)**.** Let $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ be a sequence of d-dimensional vectors. Let k be the number of filters and q the kernel size, a convolutional layer $\boldsymbol{C} : \mathbb{R}^{m\times d} \to \mathbb{R}^{(m+q-1)\times k}$ with parameters $\mathcal{K} \in \mathbb{R}^{q\times k\times d}$ can be represented as:

$$\boldsymbol{c}_i(\boldsymbol{A}) = \textstyle\sum_{j=1}^{m+2\cdot(q-1)} \hat{\boldsymbol{K}}_{i,j}\hat{\boldsymbol{a}}_j, \ \text{ where}$$
$$\hat{\boldsymbol{K}}_{i,j} = \begin{cases} \boldsymbol{K}_{j-i+1} & \text{if } 0 \le j-i \le q-1 \\ \boldsymbol{0}\boldsymbol{0}^\top & \text{otherwise} \end{cases}, \quad \forall i \in [m+q-1],$$

and $\hat{\boldsymbol{A}} = \boldsymbol{0}_{(q-1)\times d} \oplus \boldsymbol{A} \oplus \boldsymbol{0}_{(q-1)\times d} \in \mathbb{R}^{(m+q-1)\times d}$ is the zero-padded input.

We denote the parameter tensor corresponding to every layer $\boldsymbol{C}^{(i)}$ as $\mathcal{K}^{(i)}$. In the following we present our main result:

In Theorem 3.3 we present our Lipschitz constant upper bound. In Corollary 3.4 the Lipschitz constant upper bound is employed to compute the certified radius at a sentence $\boldsymbol{P}$. The Lipschitz constant upper bound can be further refined considering the local Lipschitz constant of the embedding layer around sentence $\boldsymbol{P}$, see Remark 3.5.

Theorem 3.3 (Lipschitz constant of margins of convolutional models)**.** *Let $\boldsymbol{f}$ be defined as in Eq.* (4)*. Let $g_{y,\hat{y}}(\boldsymbol{S}) = f(\boldsymbol{S})_y - f(\boldsymbol{S})_{\hat{y}}$ be the margin function for classes y and $\hat{y}$. Let $\boldsymbol{P}$ and $\boldsymbol{Q}$ be sequences of one-hot vectors, we have that for any y and $\hat{y}$:*

$$|g_{y,\hat{y}}(\boldsymbol{P}) - g_{y,\hat{y}}(\boldsymbol{Q})|$$
$$\le ||\boldsymbol{w}_{\hat{y}} - \boldsymbol{w}_y||_r \cdot \left(\textstyle\prod_{i=1}^{l} M\left(\mathcal{K}^{(i)}\right)\right) \cdot M(\boldsymbol{E}) \cdot d_{Lev}(\boldsymbol{P},\boldsymbol{Q}),$$

where $M(\mathcal{K}) = \sum_{i=1}^{q} ||\boldsymbol{K}_i||_p$, $M(\boldsymbol{E}) = \max\{\max_{i\in[|\Gamma|]} ||\boldsymbol{e}_i||_p, \max_{i,j\in[|\Gamma|]} ||\boldsymbol{e}_i - \boldsymbol{e}_j||_p\}$ *and* $\frac{1}{p}+\frac{1}{r}=1$[1].

Proof. See Appendix D □

Corollary 3.4 (Certified radius of convolutional models)**.** *Let $\boldsymbol{f}$ be defined as in Eq.* (4) *and the Lipschitz constant of $g_{y,\hat{y}}$ be:*

$$G_{y,\hat{y}} = ||\boldsymbol{w}_{\hat{y}} - \boldsymbol{w}_y||_r \cdot \left(\prod_{i=1}^{l} M\left(\mathcal{K}^{(i)}\right)\right) \cdot M(\boldsymbol{E}).$$

Then, the certified radius of $\boldsymbol{f}$ at the sentence $\boldsymbol{P}$ is given by: $k^\star_{y,\hat{y}}(S) = \min_{\hat{y}\neq y} \left\lfloor \frac{g_{y,\hat{y}}(\boldsymbol{P})}{G_{y,\hat{y}}} \right\rfloor$.

Remark 3.5 (Local Lipschitz constant of the embedding layer). Let the embeddings of a sentence $\boldsymbol{S}$ be given by $\boldsymbol{SE}$, we have that for any two sentences $\boldsymbol{P}$ and $\boldsymbol{Q}$:

$$d_{\text{ERP}}(\boldsymbol{PE},\boldsymbol{QE}) \le M(\boldsymbol{E},\boldsymbol{P}) \cdot d_{\text{Lev}}(\boldsymbol{P},\boldsymbol{Q}),$$

where

$$M(\boldsymbol{E},\boldsymbol{P}) = \max\{\max_{i\in[|\Gamma|]} ||\boldsymbol{e}_i||_p, \max_{i\in|\boldsymbol{P}|, j\in[d]} ||\boldsymbol{p}_i\boldsymbol{E} - \boldsymbol{e}_j||_p\},$$

satisfying $M(\boldsymbol{E},\boldsymbol{P}) \le M(\boldsymbol{E})$.

3.3. Training 1-Lipschitz classifiers

Models trained with the standard Cross Entropy loss and Stochastic Gradient Descent (SGD) recipe are not amenable to verification methods, resulting in small certified radiuses. This has motivated the use of specialized training methods in the image domain (Mirman et al., 2018; Gowal et al., 2018; Mueller et al., 2023; Palma et al., 2024). Verification methods in the text domain also require tailored training methods to achieve non-zero certified radiuses (Huang et al., 2019; Jia et al., 2019). Motivated by methods enforcing classifiers to be 1-Lipschitz in the image domain (Xu et al., 2022), we enforce this constraint during training in order to improve certification.

In order to achieve a 1-Lipschitz classifier, we enforce 1-Lipschitzness of every layer by dividing the output of each layer by its Lipschitz constant. This results in our modified classifier being:

$$\hat{\boldsymbol{f}}(\boldsymbol{S}) = \left(\textstyle\sum_{i=1}^{m+l\cdot(q-1)} \hat{f}_i^{(l)}(\boldsymbol{S})\right) \frac{\boldsymbol{W}}{M(\boldsymbol{W})}, \text{where}$$
$$\hat{\boldsymbol{f}}^{(j)}(\boldsymbol{S}) = \begin{cases} \frac{\boldsymbol{\sigma}\left(\boldsymbol{C}^{(j)}\left(\hat{\boldsymbol{f}}^{(j-1)}(\boldsymbol{S})\right)\right)}{M(\mathcal{K}^{(j)})} & \forall j=1,\cdots,l, \\ \frac{\boldsymbol{SE}}{M(\boldsymbol{E})} & j=0 \end{cases} \tag{5}$$

where $M(\boldsymbol{W}) = \max_{y,\hat{y}\in[o]} ||\boldsymbol{w}_y - \boldsymbol{w}_{\hat{y}}||_r$. Note that the last layer is made 1-Lipschitz with respect to the worst

[1] In the case $p=1$ and $p=\infty$, we have $r=\infty$ and $r=1$ respectively.

pair of class labels. Incorporating this information and Remark 3.5, we end up with the final Lipschitz constant for the classifier:

Corollary 3.6 (Lipschitz constant of modified classifiers)**.** *Let $\hat{\boldsymbol{f}}$ be defined as in Eq.* (5)*. Let $\hat{g}_{y,\hat{y}}(\boldsymbol{S}) = \hat{f}(\boldsymbol{S})_y - \hat{f}(\boldsymbol{S})_{\hat{y}}$ be the margin function for classes y and $\hat{y}$. Let $\boldsymbol{P}$ and $\boldsymbol{Q}$ be sequences of one-hot vectors, we have that for any y and $\hat{y}$:*

$$|\hat{g}_{y,\hat{y}}(\boldsymbol{P}) - \hat{g}_{y,\hat{y}}(\boldsymbol{Q})| \leq \frac{||\boldsymbol{w}_{\hat{y}} - \boldsymbol{w}_y||_r}{M(\boldsymbol{W})} \cdot \frac{M(\boldsymbol{E},\boldsymbol{P})}{M(\boldsymbol{E})} \cdot d_{lev}\left(\boldsymbol{P},\boldsymbol{Q}\right),$$

where $M(\boldsymbol{E})$ is defined as in Theorem 3.3, $M(\boldsymbol{E},\boldsymbol{P})$ is as in Remark 3.5 and $M(\boldsymbol{W}) = \max_{y,\hat{y}\in[o]} ||\boldsymbol{w}_y - \boldsymbol{w}_{\hat{y}}||_r$.

Note that the Lipschitz constant estimate in Corollary 3.6 is guaranteed to be at most 1 as $||\boldsymbol{w}_{\hat{y}} - \boldsymbol{w}_y||_r \leq M(\boldsymbol{W})$ and $M(\boldsymbol{E},\boldsymbol{P}) \leq M(\boldsymbol{E})$. Given this estimate, we can proceed similarly to Corollary 3.4 in order to obtain the certified radius of the modified model. Note that in the forward pass of Eq. (5), we need to compute $M(\boldsymbol{E}), M(\boldsymbol{\mathcal{K}}^{(j)})$ and $M(\boldsymbol{W})$, which increases the complexity of a forward pass with respect to Eq. (4). Nevertheless, we observe this can be efficiently done during training. Then, the weights of each layer can be divided by its Lipschitz constant, resulting in the same architecture in Eq. (4) with the guarantees of Corollary 3.6.

4. Experiments

In this section, we cover our experimental validation. In Section 4.1 we cover the experimental setup and training mechanisms shared among all experiments. In Section 4.2 we compare performance of our approach with existing IBP approaches and the naive brute force verification baseline.

4.1. Experimental setup

We train and verify our models in the sentence classification datasets AG-News (Gulli, 2005; Zhang et al., 2015) and SST-2 (Wang et al., 2019). We consider all of the characters present in the dataset except for uppercase letters, which we tokenize as lowercase. Each character is tokenized individually and assigned one embedding vector via the matrix $\boldsymbol{E}$. For all our models and datasets, following Huang et al. (2019), we select an embedding size of 150, a hidden size of 100 and a kernel size of 5 and 10 for the SST-2 and AG-News datasets respectively with a single convolutional layer. Following the setup used in Andriushchenko and Flammarion (2020) for adversarial training, we use the SGD optimizer with batch size 128 and a 30-epoch cyclic learning rate scheduler with a maximum value of 50.0, which we select via a grid search in a validation dataset, see Appendix C.1. For every experiment, we report the average results over three random seeds. All of our experiments are conducted in a single machine with an NVIDIA A100 SXM4 40 GB GPU.

4.2. Comparison with IBP and Brute Force approaches

In this section, we compare our verification method against a brute-force approach and a modification of the IBP method in (Huang et al., 2019) to handle insertions and delections of characters.

With the brute-force approach, for every sentence $\boldsymbol{P}$ in the test dataset, we evaluate our model in every setence in the set $\{\boldsymbol{Q} : d_{\text{lev}}(\boldsymbol{P},\boldsymbol{Q}) \leq k\}$ and check if there is any missclassification. Since the size of this set grows exponentially with k, we only evaluate the broute-force accuracy for $k = 1$.

In the case of IBP, we evaluate the classifier up to the pooling layer in every sentence of $\{\boldsymbol{Q} : d_{\text{lev}}(\boldsymbol{P},\boldsymbol{Q}) \leq k\}$ and then build the overapproximation. In (Huang et al., 2019) doing this was enough to build this overapproximation for $k = 1$ and re-scale it to capture larger ks. This is not the case for insertions and deletions, this constrains IBP with Levenshtein distance specifications to work only for $k = 1$. Overall, this results in IBP having the same complexity as the brute-force approach. Because of Huang et al. (2019) only considered perturbations of characters nearby in the English keyboard, the maximum perturbation size at $k = 1$ was very small, e.g., 206 and 722 sentences for SST-2 and AG-News respectively[2]. In our setup, the maximum perturbation sizes are $33,742$ and $85,686$. This makes it impractical to performe IBP verified training.

We train 3 models for each dataset and $p \in \{1, 2, \infty\}$ and verify them with the three methods. We report the average time to verify every sentence and the clean and verified accuracies at $k \in \{1, 2\}$.

In Table 2, we can observe that the p value has a big influence in the clean accuracy of the models and the verification capability of each method. With $p = 2$, we observe the highest clean accuracy, with an average of 74.80% for AG-News and 69.95% for SST-2. In terms of robust accuracy (Brute-force), $p = 2$ also provides the best performance with 62.07% for AG-News and 48.78% for SST-2. When comparing our approach with IBP, we observe that IBP obtains the best ratio between clean and verified accuracy when employing $p = \infty$, with the largest verified accuracy in SST-2 at 33.94% with $k = 1$. Our method, obtains the best performance in AG-News with $p = 2$ and in SST-2 with $p = 1$, with 38.80% and 18.69% verified accuracy respectively at $k = 1$.

In terms of runtime, our method is 4 orders of mag-

[2]See Table 3 in Huang et al. (2019)

Table 2. **Verified accuracy in AG-News and SST-2 under bounded d_{lev}:** We report the Clean accuracy (Acc.), Verified accuracy (Ver.) and the average runtime in seconds (Time) for the brute-force approach (BruteF), IBP (Huang et al., 2019) and `LipsLev`. **OOT** means the experiment was Out Of Time. ✗ means the method does not support $d_{\text{lev}} > 1$. Our method, `LipsLev`, is the only method able to provide non-trivial verified accuracies for any k in a single forward pass.

$p = \infty$

	AG-News					SST-2				
d_{lev}		1		2			1		2	
	Acc. ↑	Ver. ↑	Time ↓	Ver. ↑	Time ↓	Acc. ↑	Ver. ↑	Time ↓	Ver. ↑	Time ↓
BruteF	$65.23_{\pm(0.12)}$	$47.87_{\pm(0.09)}$	17.26	**OOT**	**OOT**	$63.95_{\pm(0.30)}$	$39.68_{\pm(0.99)}$	3.18	**OOT**	**OOT**
IBP		$27.77_{\pm(0.12)}$	17.65	✗	✗		$33.94_{\pm(1.11)}$	3.81	✗	✗
`LipsLev`		$32.33_{\pm(0.31)}$	0.0081	$11.60_{\pm(0.45)}$	0.0081		$14.68_{\pm(0.25)}$	0.0035	$0.99_{\pm(0.05)}$	0.0035

$p = 1$

	AG-News					SST-2				
d_{lev}		1		2			1		2	
	Acc. ↑	Ver. ↑	Time ↓	Ver. ↑	Time ↓	Acc. ↑	Ver. ↑	Time ↓	Ver. ↑	Time ↓
BruteF	$69.63_{\pm(0.19)}$	$54.43_{\pm(0.53)}$	17.34	**OOT**	**OOT**	$69.69_{\pm(0.14)}$	$45.22_{\pm(0.14)}$	3.05	**OOT**	**OOT**
IBP		$18.93_{\pm(0.50)}$	20.06	✗	✗		$19.00_{\pm(1.08)}$	3.88	✗	✗
`LipsLev`		$34.50_{\pm(0.36)}$	0.0014	$12.53_{\pm(0.29)}$	0.0014		$18.69_{\pm(0.80)}$	0.0026	$1.83_{\pm(0.00)}$	0.0026

$p = 2$

	AG-News					SST-2				
d_{lev}		1		2			1		2	
	Acc. ↑	Ver. ↑	Time ↓	Ver. ↑	Time ↓	Acc. ↑	Ver. ↑	Time ↓	Ver. ↑	Time ↓
BruteF	$74.80_{\pm(0.45)}$	$62.07_{\pm(0.82)}$	27.66	**OOT**	**OOT**	$69.95_{\pm(0.32)}$	$48.78_{\pm(0.43)}$	4.41	**OOT**	**OOT**
IBP		$29.10_{\pm(0.45)}$	31.16	✗	✗		$16.06_{\pm(1.17)}$	5.34	✗	✗
`LipsLev`		$38.80_{\pm(0.29)}$	0.0076	$13.93_{\pm(0.21)}$	0.0076		$14.57_{\pm(0.34)}$	0.0073	$0.73_{\pm(0.27)}$	0.0073

nitude faster than brute-force and IBP, which attain similar runtimes. The impossibility of IBP to verify for $k = 1$ and its larger runtime than brute-force, poses it as an impractical tool for Levenstein distance verification. Our method is the only one able to verify for $k > 1$, with 13.93% verified accuracy for AG-News and 1.83% for SST-2 at $k = 2$.

5. Conclusion

In this work, we propose the first approach able to verify NLP classifiers using the Levenshtein distance constraints. Our approach is based on an upper bound of the Lipschitz constant of convolutional classifiers with respect to the Levenshtein distance. Our method, `LipsLev` is able to obtain a verified accuracy of 38% at distance $k = 1$ in the AG-News dataset in a single forward pass per sample. Moreover, our method is the only existing method that can practically verify for Levenshtein distances larger than 1. We expect our work can inspire a new line of works on verifying larger distances and more broadly verifying additional classes of NLP classifiers. We will make the code publicly available upon the publication of this work, our implementation is attached with this submission.

Future work and limitations: A problem shared with verification methods in the image domain is scalability (Wang et al., 2021). In our experimental validation, we show the verified accuracy decreases with the number of layers. Scaling verification methods to production models is a challenge, that becomes more relevant with the deployment of large language models and their recently discovered vulnerabilities (Zou et al., 2023). Even though our method is the first to practically provide Levenshtein distance certificates in NLP, our formulation does not cover modern architectures as transformers (Vaswani et al., 2017) and does not support classifiers working over popular tokenizers such as SentencePiece (Kudo and Richardson, 2018). We believe though, that the generality of the approach will enable certification once Lipschitz constant estimates of such pieces are known. Even though Lipschitz constant upper bounds exist for transformer models (Qi et al., 2023), their extension to our metric is not straightforward, yet it is an interesting avenue.

Broader impact

In this work, we tackle the important problem of verifying the robustness of NLP models against adversarial attacks. By advancing in this area, we can positively impact society by ensuring NLP models deployed in safety critical applications are robust to such perturbations.

Acknowledgments

Authors acknowledge the constructive feedback of reviewers and the work of the ICML 2024 Next Generation of AI Safety Workshop organizers. We thank Zulip[3] for their project organization tool. ARO - Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-24-1-0048. Hasler AI - This work was supported by Hasler Foundation Program: Hasler Responsible AI (project number 21043). SNF project – Deep Optimisation - This work was supported by the Swiss National Science Foundation (SNSF) under grant number 200021_205011. Elias appreciates the great discussions with Adrian Luis Müller and Ioannis Mavrothalassitis.

References

Elias Abad Rocamora, Yongtao Wu, Fanghui Liu, Grigorios G. Chrysos, and Volkan Cevher. Revisiting character-level adversarial attacks. In *International Conference on Machine Learning (ICML)*, 2024.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in neural information processing systems (NeurIPS)*, 2020.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018.

Gregory Bonaert, Dimitar I. Dimitrov, Maximilian Baader, and Martin Vechev. Fast and precise certification of transformers. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021.

Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 792–803. VLDB Endowment, 2004.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International conference on machine learning*, pages 854–863. PMLR, 2017.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

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)*, 2019.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *IEEE Security and Privacy Workshops (SPW)*, 2018.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

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 arXiv:1810.12715*, 2018.

Antonio Gulli. Ag's corpus of news articles, 2005. URL `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems (NeurIPS)*, 30, 2017.

[3] `https://zulip.com`

Bairu Hou, Jinghan Jia, Yihua Zhang, Guanhua Zhang, Yang Zhang, Sijia Liu, and Shiyu Chang. Textgrad: Advancing robustness evaluation in NLP by gradient-driven optimization. In *International Conference on Learning Representations (ICLR)*, 2023.

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 *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in neural information processing systems (NeurIPS)*, 2021.

Zhuoqun Huang, Neil G Marchant, Keane Lucas, Lujo Bauer, Olga Ohrimenko, and Benjamin I. P. Rubinstein. RS-del: Edit distance robustness certificates for sequence classifiers via randomized deletion. In *Advances in neural information processing systems (NeurIPS)*, 2023. URL https://openreview.net/forum?id=ffFcRPpnWx.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2018.

Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rJe4_xSFDB.

Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

Aiwei Liu, Honghai Yu, Xuming Hu, Shu'ang Li, Li Lin, Fukun Ma, Yawen Yang, and Lijie Wen. Character-level white-box adversarial attacks against transformers via attachable subwords substitution. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4, 2021.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning (ICML)*. PMLR, 2018.

Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics (SIAM), 2009.

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*, 2020.

Mark Niklas Mueller, Franziska Eckert, Marc Fischer, and Martin Vechev. Certified training: Small boxes are all you need. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=7oFuxtJtUMH.

Alessandro De Palma, Rudy R Bunel, Krishnamurthy Dj Dvijotham, M. Pawan Kumar, Robert Stanforth, and Alessio Lomuscio. Expressive losses for verified robustness via convex combinations. In *International Conference on Learning Representations (ICLR)*, 2024.

Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. Lipsformer: Introducing lipschitz continuity to vision transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Haifeng Qian and Mark N. Wegman. L2-nonexpansive neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. In *International Conference on Learning Representations (ICLR)*, 2020.

Zhouxing Shi, Yihan Wang, Huan Zhang, J Zico Kolter, and Cho-Jui Hsieh. Efficiently computing local lipschitz constants of neural networks via bound propagation. *Advances in neural information processing systems (NeurIPS)*, 35:2350–2364, 2022.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems (NeurIPS)*, 27, 2014.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in neural information processing systems (NeurIPS)*, 31, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *USENIX Conference on Security Symposium*, 2018.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in neural information processing systems (NeurIPS)*, 34, 2021.

Michael S Waterman, Temple F Smith, and William A Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367–387, 1976.

Xiaojun Xu, Linyi Li, and Bo Li. LOT: Layer-wise orthogonal training on improving l2 certified robustness. In *Advances in neural information processing systems (NeurIPS)*, 2022.

Mao Ye, Chengyue Gong, and Qiang Liu. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems (NeurIPS)*, 28, 2015.

Yuhao Zhang, Aws Albarghouthi, and Loris D'Antoni. Certified robustness to programmable transformations in LSTMs. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A. Related Work

In this section we cover the related works in Lipschitz constant based verification and more generally, verification in NLP.

Lipschitz verification: Hein and Andriushchenko (2017) firstly study the computation of the Lipschitz constant in order to provide formal guarantees of the robustness of support vector machines and two-layer nueral networks. Tsuzuku et al. (2018) compute Lipschitz constant upper bounds for deeper networks and regularize such upper bounds to improve certificates. Since then, tighter upper bounds for the Lipschitz constant have been proposed (Huang et al., 2021; Fazlyab et al., 2019; Latorre et al., 2020; Shi et al., 2022). A variety of works propose constraining the Lipschitz constant to be 1 during training in order to have automatic robustness certificates (Cisse et al., 2017; Qian and Wegman, 2019; Gouk et al., 2021; Xu et al., 2022). All previous works center in the standard ℓ_p norms and cannot be applied to the NLP domain. Our work provides the first 1-Lipschitz training method for the Levenshtein distance.

Verfication in NLP: Jia et al. (2019) propose using Interval Bound Propagation via an over-approximation of the embeddings of the set of synonyms of each word. Concurrently, Huang et al. (2019) incorporate this technique for verifying against replacements of nearby characters in the english keyboard. Bonaert et al. (2021); Shi et al. (2020) propose zonotope abstractions and IBP for verifying against synonym substitutions in transformer models. Zhang et al. (2021) propose a verification procedure that can handle a small number of input perturbations for LSTM classifiers. Deviating from these approaches, Ye et al. (2020) propose using randomized smoothing techniques Cohen et al. (2019) in order to verify probabilistically against character substitutions. Huang et al. (2023) used similar techniques in order to probabilistically verify under Levenshtein distance specifications. In this work, we propose the first non-probabilistic method for verifying under Levenshtein distance specifications. In Table 1 we highlight the differences with existing works in NLP verification.

B. Interval Bound Propagation (IBP)

Existing robustness verification approaches rely on IBP for verifying the robustness of text models (Huang et al., 2019; Jia et al., 2019). IBP relies on the input being constrained in a box. Let $\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^d$, every element of $\boldsymbol{x}$ is assumed to be in an interval given by $\boldsymbol{l}$ and $\boldsymbol{u}$, i.e., $l_i \leq x_i \leq u_i \; \forall i \in [d]$ or $\boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}]$ for short. These constraints arise naturally when studying robustness in the ℓ_∞ norm, as the constraint $\boldsymbol{x} \in \{\boldsymbol{x}^{(0)} + \boldsymbol{\delta} : ||\boldsymbol{\delta}||_\infty \leq \epsilon\}$ can exactly be represented as $\boldsymbol{x} \in [\boldsymbol{x}^{(0)} - \epsilon, \boldsymbol{x}^{(0)} + \epsilon]$. IBP consists in a set of rules to obtain interval constraints of the output of a function, given the interval constraints of the input. In the case of an affine mapping $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, we can easily obtain the interval constraints $\boldsymbol{f}(\boldsymbol{x}) \in [\boldsymbol{l}_{\boldsymbol{f}}(\boldsymbol{x}), \boldsymbol{u}_{\boldsymbol{f}}(\boldsymbol{x})], \; \forall \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}]$ with:

$$\boldsymbol{l}_{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{W}^{+}\boldsymbol{l} + \boldsymbol{W}^{-}\boldsymbol{u} + \boldsymbol{b}, \quad \boldsymbol{u}_{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{W}^{+}\boldsymbol{u} + \boldsymbol{W}^{-}\boldsymbol{l} + \boldsymbol{b}\,, \tag{6}$$

where $\boldsymbol{W}^{+}$ and $\boldsymbol{W}^{-}$ are the positive and negative parts of $\boldsymbol{W}$. In the case of the ReLU activation function $\boldsymbol{\sigma}(\boldsymbol{x}) = \max\{0, \boldsymbol{x}\}$, we have that:

$$\boldsymbol{l}_{\boldsymbol{\sigma}}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{l}), \quad \boldsymbol{u}_{\boldsymbol{\sigma}}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{u})\,. \tag{7}$$

By applying recursively the simple rules in Eqs. (6) and (7), one can easily verify robustness properties of ReLU fully-connected and convolutional networks (Wang et al., 2018).

Nevertheless, IBP has two main limitations:

a) IBP assumes the input space to be of fixed length, e.g., $\mathbb{R}^d$.

b) IBP can only handle interval constrained inputs, e.g., $\boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}]$.

Limitation a) makes it impossible to verify Levenshtein distance constraints as they include insertion and deletion operations, which change the length of the input sequence. In the literature, limitation a) forces existing verification methods to only consider replacements of characters/words (Huang et al., 2019; Jia et al., 2019; Shi et al., 2020; Bonaert et al., 2021; Zhang et al., 2021).

Limitation b) can be circumvented by building an over approximation of the replacement constraints that can be represented with intervals. In the case of text, one can directly build an over approximation of the embeddings. Let $\boldsymbol{Z} = \boldsymbol{S}\boldsymbol{E} \in \mathbb{R}^{m \times d}$, where $\boldsymbol{S} \in \mathcal{S}(\Gamma)$ is the sequence of one-hot vectors representing each character/word, and $\boldsymbol{E} \in \mathbb{R}^{|\Gamma| \times d}$ is the embedding matrix. Let d_{edit} be the edit distance without insertions and deletions, our constraint in the edit distance (Eq. (1)) translates in the embedding space to the set:

$$\mathcal{Z}_k(\boldsymbol{S}) = \{\boldsymbol{S}'\boldsymbol{E} : d_{\text{edit}}(\boldsymbol{S}, \boldsymbol{S}') \leq k, \boldsymbol{S}' \in \mathcal{S}(\Gamma)\}\,.$$

We can overapproximate this set with interval constraints such that $\hat{\boldsymbol{Z}} \in [\boldsymbol{L}, \boldsymbol{U}]$, with $l_{i,j} = \min_{\boldsymbol{Z} \in \mathcal{Z}_k(\boldsymbol{S})} z_{i,j}$ and $u_{i,j} = \max_{\boldsymbol{Z} \in \mathcal{Z}_k(\boldsymbol{S})} z_{i,j}$. But, because we can replace any character/word at any position, we end up with $\boldsymbol{L} = \boldsymbol{l} \oplus \boldsymbol{l} \oplus \cdots \oplus \boldsymbol{l}$ and $\boldsymbol{U} = \boldsymbol{u} \oplus \boldsymbol{u} \oplus \cdots \oplus \boldsymbol{u}$, where:

$$l_i = \min_{k \in [|\Gamma|]} e_{k,i}, \quad u_i = \max_{k \in [|\Gamma|]} e_{k,i}, \quad \forall i \in [d]\,.$$

Therefore, this overapproximation contains the embeddings of any $\boldsymbol{S}' \in \{0,1\}^{m \times |\Gamma|} : ||\boldsymbol{s}'_i||_1 = 1, \ \ \forall i \in [m]$, i.e., *every sentence of length m*, making verification impossible. To circumvent this, existing methods focus on the synonym replacement task, further restricting $\mathcal{Z}_k(\boldsymbol{S})$ to only replace words for a word in a small set of synonyms (Jia et al., 2019; Shi et al., 2020; Bonaert et al., 2021). Alternatively, Huang et al. (2019) compute the over approximation after the pooling layer of the model, circumventing this problem. Nevertheless, their approach requires $|\mathcal{Z}_1(\boldsymbol{S})|$ forward passes. This number of forward passes can be in the order of tenths of thousands for large m and $|\Gamma|$.

Our Lipschitz constant based approach, `LipsLev`, can handle sequences of any length and requires a single forward pass through the model.

C. Additional experimental validation

In Appendix C.1 we present our grid search for selecting the best learning rate for each dataset and p value in the ERP distance Definition 3.1. In Appendix C.2 we analyze the effect of increasing the number of convolutional layers. in Appendix C.3 we cover the hyperparameter selection of our method.

C.1. Hyperparameter selection

In order to select the best learning rate in each dataset and p norm for the ERP distance, we compute the clean and verified accuracy at $k = 1$ in a validation set of $1,000$ samples extracted from each training set. We test the learning rate values $\{0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000\}$. We train convolutional models with 1 convolutional layer and the standard embedding, hidden and kernel sizes in Section 4.1. We notice these large learning rates are needed due to the 1-Lipschitz formulation in Eq. (5).

Based on the results from Fig. S1, we select 50 as our learning rate for the rest of experiments in this work.

C.2. Training deeper models

In this section, we study the performance of models with more than one convolutional layer. We train with $1, 2, 3$ and 4 convolutional layers with a hidden size of 100 and a kernel size of 5 and 10 for SST-2 and AG-News respectively. We train the models with the 1-Lipschitz formulation in Eq. (5) with $p \in \{1, 2, \infty\}$.

In Figs. S2 and S3 we can observe that increasing the number of layers degrades the clean and verified accuracy for every value of p. Nevertheless, for $p = 2$, the effect is diminished. Jointly with the improved peformance when using $p = 2$ in Section 4.2, we advocate for its use in the ERP distance.

C.3. Regularizing the Lipschitz constant

In Section 3.3 we describe how to enforce our convolutional classifier to be 1-Lipschitz. But, is there a better way of improving the final verified accuracy of our models? Because our Lipschitz constant estimate in Theorem 3.3 its differentiable with respect to the parameters of the model, we can regularize this quantity during training in order to achieve a lower Lipschitz constant and hopefully a better verified accuracy. In practice we regularize $G = M(\boldsymbol{W}) \cdot M(\boldsymbol{\mathcal{K}}^{(1)}) \cdot M(\boldsymbol{E})$ as defined in Theorem 3.3 and Corollary 3.6.

We train single-layer models with a regularization paremeter of $\lambda \in \{0, 0.001, 0.01, 0.1\}$, where $\lambda = 0$ is equivalent to standard training. We initialize the weights of each layer so that their Lipschitz constant is 1. We use a learning rate of 0.01. We measure the final Lipschitz constant of each model and their clean and verified accuracies in a validation set of 1000

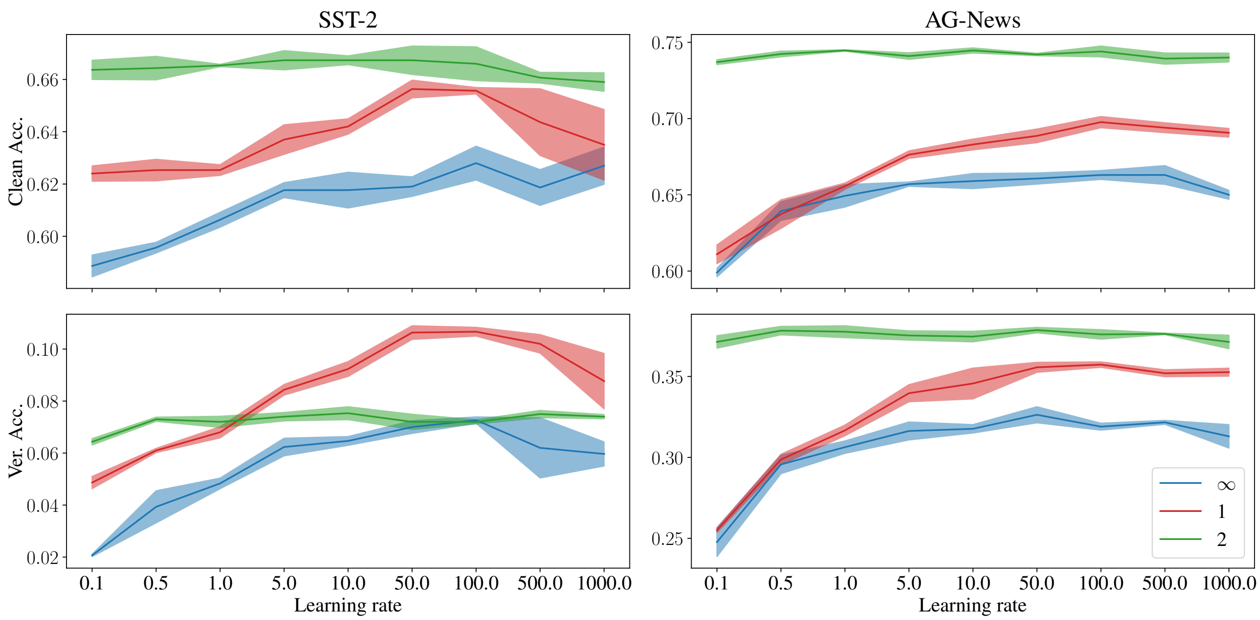


Figure S1. **Learning rate selection for the SST-2 and AG-News datasets:** We report the clean and verified accuracy in a validation set of 1,000 sentences extracted from the training split of each dataset and set aside during training. We set the learning rate equal to 100 in the rest of our experiments as it provides a good trade-off between clean and verified accuracy for all norms and datasets.

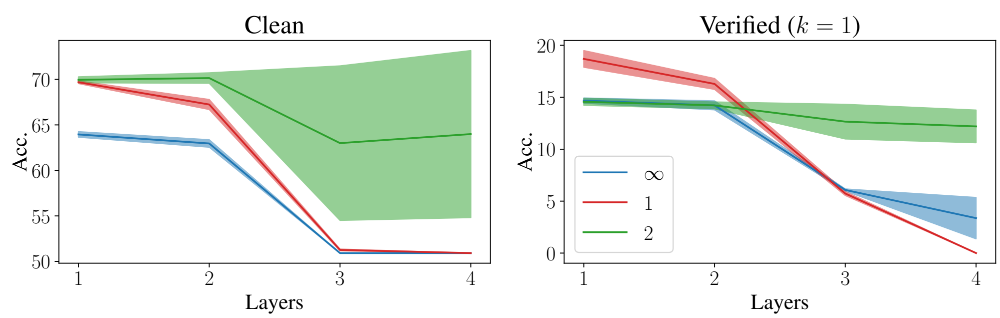


Figure S2. **Training deeper models in SST-2:** We report the clean and verified accuracies with `LipsLev` at $k = 1$ for $p \in \{1, 2, \infty\}$. Clean and verified accuracies decrease with the number of layers. With $p = 2$ the performance is less degraded with the number of layers.

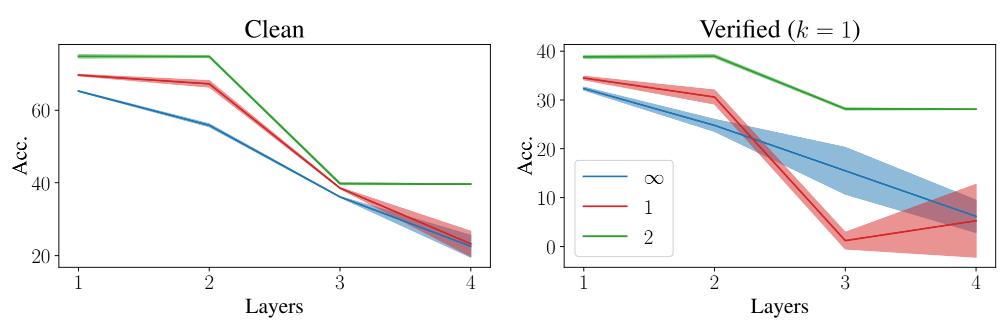
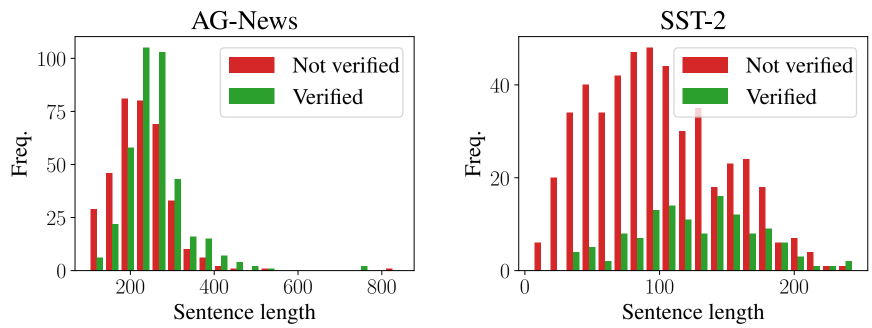


Figure S3. **Training deeper models in AG-News:** We report the clean and verified accuracies with `LipsLev` at $k = 1$ for $p \in \{1, 2, \infty\}$. Clean and verified accuracies decrease with the number of layers. With $p = 2$ the performance is less degraded with the number of layers.

Table S3. **Regularizing v.s. enforcing Lipschitzness in SST-2:** We compare the performance when regularizing the Lipschitz constant (G) during training with $\lambda \in \{0, 0.001, 0.01, 0.1\}$, against enforcing 1-Lipschitzness through Eq. (5). Regularizing G leads to either models with similar performance to a constant classifier (55.7% for SST-2), or more accurate but non-verifiable models than when using the formulation in Eq. (5).

	$p=\infty$			$p=1$			$p=2$		
λ	Clean ↑	Ver. ↑	G	Clean ↑	Ver. ↑	G	Clean ↑	Ver.↑	G
0	$89.0_{\pm(0.5)}$	$0.0_{\pm(0.0)}$	$2850.2_{\pm(80.1)}$	$86.1_{\pm(0.4)}$	$0.0_{\pm(0.0)}$	$449.6_{\pm(3.0)}$	$87.2_{\pm(0.2)}$	$0.0_{\pm(0.0)}$	$129.1_{\pm(2.9)}$
0.001	$80.8_{\pm(0.6)}$	$0.0_{\pm(0.0)}$	$65.0_{\pm(0.9)}$	$84.5_{\pm(0.5)}$	$0.0_{\pm(0.0)}$	$44.1_{\pm(0.7)}$	$86.2_{\pm(0.4)}$	$0.0_{\pm(0.0)}$	$37.7_{\pm(0.3)}$
0.01	$60.1_{\pm(1.1)}$	$1.7_{\pm(0.1)}$	$1.4_{\pm(0.1)}$	$79.7_{\pm(0.5)}$	$0.1_{\pm(0.0)}$	$6.9_{\pm(0.0)}$	$81.6_{\pm(0.4)}$	$0.1_{\pm(0.0)}$	$8.8_{\pm(0.1)}$
0.1	$56.2_{\pm(0.0)}$	$55.7_{\pm(0.3)}$	$0.0_{\pm(0.0)}$	$57.3_{\pm(0.0)}$	$53.2_{\pm(0.8)}$	$0.1_{\pm(0.0)}$	$57.5_{\pm(0.9)}$	$34.1_{\pm(3.0)}$	$0.1_{\pm(0.0)}$
Eq. (5)	$62.8_{\pm(0.6)}$	$7.3_{\pm(0.1)}$	$1.00_{\pm(0.0)}$	$65.6_{\pm(0.1)}$	$10.7_{\pm(0.2)}$	$1.00_{\pm(0.0)}$	$66.6_{\pm(0.6)}$	$7.2_{\pm(0.1)}$	$1.00_{\pm(0.0)}$



Average Length

Dataset	Method	Verified ✗	Verified ✓
AG-News	BruteF	221.4	248.4
	`LipsLev`	228.0	258.9
SST-2	BruteF	93.2	109.5
	`LipsLev`	98.3	126.8

Figure S4. **Sentence length distribution for verified and not verified sentences:** We report the histogram of the lengths for verified and not verified sentences at $k = 1$ with `LipsLev` in the models trained with $p = 2$. Shorter sentences are harder to verify in both SST-2 and AG-News with both `LipsLev` and the brute force approach.

samples left out from the training set. As a baseline, we report these metrics for the models trained with the formulation in Eq. (5).

In Table S3 we observe that for all the studied norms, when regularizing the Lipschitz constant G, we cannot easily match the performance when using Eq. (5). Regularized models converge to either close-to-constant classifiers (55.7% clean accuracy for SST-2) or present a close-to-zero verified accuracy. The formulation in Eq. (5) allows us to obtain verifiable models without the need to tune hyperparameters.

C.4. The influence of sentence length in verification

In this section we study the qualitative characteristics of a sentence leading to a better verification properties, specifically, we study the influence of the sentence length in verification. We compute the sentence lengths for the sentences that were verified and not verified at $k = 1$ for the models in Section 4.2 with $p = 2$ and both brute force and `LipsLev`.

In Fig. S4 we can observe that for both verification methods on both datasets, the verified sentences present a larger average length. We believe this is reasonable as single characters perturbations are likely to introduce a smaller semantic change for longer sequences.

D. Proofs

In this section we introduce the mathematical tools needed to derive our Lipschitz constant upper bounds for each layer in Eq. (4). The section concludes with the proof of our main result in Theorem 3.3.

Definition S1 (Zero-paddings)**.** Let $\boldsymbol{X} \in \mathcal{X}_d$ a sequence of m non-zero vectors. Let $l \geq m$, a zero padding function $\boldsymbol{Z} : \mathcal{X}_d \to \mathbb{R}^{l \times d}$ is some function defined by the tuple:

$$(i_k)_{k=1}^{l} : \begin{cases} m \geq i_k > i_j \;\; \forall 1 < j < k & \text{if } i_k \neq 0 \\ |\{k \in [l] : i_k = 0\}| = l - m & \text{if } i_k = 0 \end{cases}$$

so that:

$$\boldsymbol{z}_k(\boldsymbol{X}) = \begin{cases} \boldsymbol{x}_{i_k} & \text{if } i_k \neq 0 \\ \boldsymbol{0} & \text{if } i_k = 0 \end{cases}$$

Intuitively, a valid zero-padding function inserts $l - m$ zeros in between any vector of the sequence, the beginning or the end. We denote as $\mathcal{Z}_{m,l}$ the set of zero paddings from sequences of length m to sequences of length l.

Remark S2. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ and a zero padding $\boldsymbol{Z} \in \mathcal{Z}_{m,l}$, we denote the column and row-wise padding as $\overline{\boldsymbol{Z}}(\boldsymbol{A}) = \boldsymbol{Z}(\boldsymbol{Z}(\boldsymbol{A}^\top)^\top) \in \mathbb{R}^{l\times l}$.

Proposition S3 (Alternative definition of d_{ERP})**.** *Let d_{ERP} be as in Definition 3.1. Let $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times d}$ be two sequences. Let $\mathcal{Z}_{m,m+n}$ and $\mathcal{Z}_{n,m+n}$ be the zero-padding functions from length m and n respectively to length $m+n$. The ERP distance can be expressed as:*

$$d_{ERP}(\boldsymbol{A}, \boldsymbol{B}) = \min_{\boldsymbol{Z}^a \in \mathcal{Z}_{m,m+n}, \boldsymbol{Z}^b \in \mathcal{Z}_{n,m+n}} \sum_{k=1}^{m+n} \left|\left| \boldsymbol{z}_k^a(\boldsymbol{A}) - \boldsymbol{z}_k^b(\boldsymbol{B}) \right|\right|_p$$

Lemma S4 (Properties of the ERP distance)**.** *Some important properties of the ERP distance are summarized here:*

(a) Generalization of edit distance:
In the case of having sequences of one-hot vectors $\boldsymbol{A} \in \{0,1\}^{m\times d} : ||\boldsymbol{a}_i||_1 = 1$, and using $p = \infty$, the ERP distance is equal to the edit distance (Levenshtein et al., 1966).

(b) Invariance to the concatenation of zeros:

$$d_{ERP}(\boldsymbol{A} \oplus \boldsymbol{0}, \boldsymbol{B}) = d_{ERP}(\boldsymbol{0} \oplus \boldsymbol{A}, \boldsymbol{B}) = d_{ERP}(\boldsymbol{A}, \boldsymbol{B}) \;\; \forall \boldsymbol{A} \in \mathbb{R}^{m\times d}, \boldsymbol{B} \in \mathbb{R}^{n\times d}$$

(c) Distance to the empty set:

$$d_{ERP}(\boldsymbol{A}, \emptyset) = \sum_{i=1}^{m} ||\boldsymbol{a}_i||_p \;\; \forall \boldsymbol{A} \in \mathbb{R}^{m\times d}$$

(d) Symmetry:
$d_{ERP}(\boldsymbol{A}, \boldsymbol{B}) = d_{ERP}(\boldsymbol{B}, \boldsymbol{A}) \;\; \forall \boldsymbol{A} \in \mathbb{R}^{m\times d}, \boldsymbol{B} \in \mathbb{R}^{n\times d}$

(e) Triangular inequality:
For any $\boldsymbol{A} \in \mathbb{R}^{m\times d}, \boldsymbol{B} \in \mathbb{R}^{n\times d}, \boldsymbol{B} \in \mathbb{R}^{l\times d}$, we have:

$$d_{ERP}(\boldsymbol{A}, \boldsymbol{B}) \leq d_{ERP}(\boldsymbol{A}, \boldsymbol{C}) + d_{ERP}(\boldsymbol{C}, \boldsymbol{B}).$$

(f) Subdistance:
The ERP distance is not a distance because of its invariance to the concatenation of zeros:

$$d_{ERP}(\boldsymbol{A}, \boldsymbol{A} \oplus \boldsymbol{0}) = d_{ERP}(\boldsymbol{A}, \boldsymbol{A}) = 0 \;\; \forall \boldsymbol{A} \in \mathbb{R}^{m\times d}$$

proof of Lemma S4. Properties (a), (b), (c), (d) and (f) are straightforward from the deffinition, we will prove the triangular inequality (e). This proof follows similarly to the one of Waterman et al. (1976) for the standard Levenshtein distance. Let $L = m + n + l$, starting from the definition in Proposition S3:

$$d_{\text{ERP}}(\boldsymbol{A}, \boldsymbol{B}) + d_{\text{ERP}}(\boldsymbol{B}, \boldsymbol{C}) = \min_{\substack{\boldsymbol{Z}^a \in \mathcal{Z}_{m,L}, \boldsymbol{Z}^b \in \mathcal{Z}_{n,L} \\ \boldsymbol{Z}^c \in \mathcal{Z}_{n,L}, \boldsymbol{Z}^d \in \mathcal{Z}_{l,L}}} \sum_{k=1}^{L} \left|\left| \boldsymbol{z}_k^a(\boldsymbol{A}) - \boldsymbol{z}_k^b(\boldsymbol{B}) \right|\right|_p + \sum_{j=1}^{L} \left|\left| \boldsymbol{z}_j^c(\boldsymbol{B}) - \boldsymbol{z}_j^d(\boldsymbol{C}) \right|\right|_p .$$

Let $\boldsymbol{Z}^e, \boldsymbol{Z}^f \in \mathcal{Z}_{L,2L}$ be two zero paddings so that $\boldsymbol{Z}^e(\boldsymbol{Z}^b(B)) = \boldsymbol{Z}^f(\boldsymbol{Z}^c(B))$:

$$\begin{aligned}
d_{\text{ERP}}(\boldsymbol{A},\boldsymbol{B}) + d_{\text{ERP}}(\boldsymbol{B},\boldsymbol{C}) &= \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m,L},\boldsymbol{Z}^b\in\mathcal{Z}_{n,L}\\ \boldsymbol{Z}^c\in\mathcal{Z}_{n,L},\boldsymbol{Z}^d\in\mathcal{Z}_{l,L}}} \sum_{k=1}^{2L} \left|\left|\boldsymbol{z}_k^e(\boldsymbol{Z}^a(\boldsymbol{A})) - \boldsymbol{z}_k^e(\boldsymbol{Z}^b(\boldsymbol{B}))\right|\right|_p \\
&\qquad + \left|\left|\boldsymbol{z}_k^f(\boldsymbol{Z}^c(\boldsymbol{B})) - \boldsymbol{z}_k^f(\boldsymbol{Z}^d(\boldsymbol{C}))\right|\right|_p \\
\text{[Triangular ineq. for } ||\cdot||_p\text{]} &\geq \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m,L},\boldsymbol{Z}^b\in\mathcal{Z}_{n,L}\\ \boldsymbol{Z}^c\in\mathcal{Z}_{n,L},\boldsymbol{Z}^d\in\mathcal{Z}_{l,L}}} \sum_{k=1}^{2L} \left|\left|\boldsymbol{z}_k^e(\boldsymbol{Z}^a(\boldsymbol{A})) - \boldsymbol{z}_k^e(\boldsymbol{Z}^b(\boldsymbol{B}))\right.\right. \\
&\qquad \left.\left. + \ \boldsymbol{z}_k^f(\boldsymbol{Z}^c(\boldsymbol{B})) - \boldsymbol{z}_k^f(\boldsymbol{Z}^d(\boldsymbol{C}))\right|\right|_p \\
[\boldsymbol{Z}^e(\boldsymbol{Z}^b(B)) = \boldsymbol{Z}^f(\boldsymbol{Z}^c(B))] &= \min_{\boldsymbol{Z}^a\in\mathcal{Z}_{m,L},\boldsymbol{Z}^d\in\mathcal{Z}_{l,L}} \sum_{k=1}^{2L} \left|\left|\boldsymbol{z}_k^e(\boldsymbol{Z}^a(\boldsymbol{A})) - \boldsymbol{z}_k^f(\boldsymbol{Z}^d(\boldsymbol{C}))\right|\right|_p \\
&= d_{\text{ERP}}(\boldsymbol{A},\boldsymbol{C})\,,
\end{aligned}$$

where the last equality follows from $\boldsymbol{z}_k^e(\boldsymbol{Z}^a)$ and $\boldsymbol{z}_k^f(\boldsymbol{Z}^d)$ being valid zero paddings. □

Lemma S5 (Difference of sums)**.** *Let $\boldsymbol{A},\boldsymbol{B} \in \mathcal{X}_d$, we have that:*

$$\left|\left|\sum_{i=1}^{m} \boldsymbol{a}_i - \sum_{j=1}^{n} \boldsymbol{b}_j\right|\right|_p \leq d_{ERP}(\boldsymbol{A},\boldsymbol{B})$$

proof of Lemma S5.

$$\begin{aligned}
\left|\left|\sum_{i=1}^{m} \boldsymbol{a}_i - \sum_{j=1}^{n} \boldsymbol{b}_j\right|\right|_p &= \left|\left|\min_{\boldsymbol{Z}^a\in\mathcal{Z}_{m,m+n},\boldsymbol{Z}^b\in\mathcal{Z}_{n,m+n}} \sum_{k=1}^{m+n} \boldsymbol{z}_k^a(\boldsymbol{A}) - \boldsymbol{z}_k^b(\boldsymbol{B})\right|\right|_p \\
&\leq \min_{\boldsymbol{Z}^a\in\mathcal{Z}_{m,m+n},\boldsymbol{Z}^b\in\mathcal{Z}_{n,m+n}} \sum_{k=1}^{m+n} \left|\left|\boldsymbol{z}_k^a(\boldsymbol{A}) - \boldsymbol{z}_k^b(\boldsymbol{B})\right|\right|_p \\
&= d_{\text{ERP}}(\boldsymbol{A},\boldsymbol{B})
\end{aligned}$$

□

Lemma S6 (Difference of means)**.** *Let $\boldsymbol{A},\boldsymbol{B} \in \mathcal{X}_d$, we have that:*

$$\left|\left|\frac{1}{m}\cdot\sum_{i=1}^{m} \boldsymbol{a}_i - \frac{1}{n}\cdot\sum_{j=1}^{n} \boldsymbol{b}_j\right|\right|_p \leq \frac{|m-n|}{m\cdot n}\cdot\left|\left|\sum_{i=1}^{m} \boldsymbol{a}_i\right|\right|_p + \frac{1}{n}\cdot d_{ERP}(\boldsymbol{A},\boldsymbol{B})$$

and

$$\left|\left|\frac{1}{m}\cdot\sum_{i=1}^{m} \boldsymbol{a}_i - \frac{1}{n}\cdot\sum_{j=1}^{n} \boldsymbol{b}_j\right|\right|_p \leq \frac{|m-n|}{m\cdot n}\cdot\left|\left|\sum_{j=1}^{m} \boldsymbol{b}_j\right|\right|_p + \frac{1}{m}\cdot d_{ERP}(\boldsymbol{A},\boldsymbol{B})\,.$$

In the case of $\boldsymbol{A}$ and $\boldsymbol{B}$ being sequences of one-hot vectors, we have that:

$$\left|\left|\frac{1}{m}\cdot\sum_{i=1}^{m} \boldsymbol{a}_i - \frac{1}{n}\cdot\sum_{j=1}^{n} \boldsymbol{b}_j\right|\right|_\infty \leq \begin{cases} \frac{1}{m}\cdot d_{lev}\left(\boldsymbol{A},\boldsymbol{B}\right) & \text{if } m=n \\ \frac{2}{m}\cdot d_{lev}\left(\boldsymbol{A},\boldsymbol{B}\right) & \text{if } m\neq n \end{cases}$$

proof of Lemma S6. Starting with the first result:

$$\begin{aligned}\left|\left|\frac{1}{m}\cdot\sum_{i=1}^{m}\boldsymbol{a}_i-\frac{1}{n}\cdot\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_p &= \frac{1}{m\cdot n}\left|\left|(n+m-m)\cdot\sum_{i=1}^{m}\boldsymbol{a}_i-m\cdot\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_p\\ &\leq \frac{1}{m\cdot n}\left(|m-n|\cdot\left|\left|\sum_{i=1}^{m}\boldsymbol{a}_i\right|\right|_p+m\cdot\left|\left|\sum_{i=1}^{m}\boldsymbol{a}_i-\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_p\right)\\ \text{[Lemma S5]} &\leq \frac{|m-n|}{m\cdot n}\cdot\left|\left|\sum_{i=1}^{m}\boldsymbol{a}_i\right|\right|_p+\frac{1}{n}\cdot d_{\text{ERP}}\left(\boldsymbol{A},\boldsymbol{B}\right)\,.\end{aligned}$$

Note that since $\boldsymbol{A}$ and $\boldsymbol{B}$ are interchangeable, we immediately have:

$$\left|\left|\frac{1}{m}\cdot\sum_{i=1}^{m}\boldsymbol{a}_i-\frac{1}{n}\cdot\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_p \leq \frac{|m-n|}{m\cdot n}\cdot\left|\left|\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_p+\frac{1}{m}\cdot d_{\text{ERP}}\left(\boldsymbol{A},\boldsymbol{B}\right)\,. \tag{8}$$

In the case $\boldsymbol{A}$ and $\boldsymbol{B}$ are sequences of one-hot vectors, if $m=n$, we can directly get the $1/m$ factor out of the norm and apply Lemma S5 to get the first case. For the case $m\neq n$, we can manipulate Eq. (8) to get the desired result:

$$\begin{aligned}\left|\left|\frac{1}{m}\cdot\sum_{i=1}^{m}\boldsymbol{a}_i-\frac{1}{n}\cdot\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_\infty &\leq \frac{|m-n|}{m\cdot n}\cdot\left|\left|\sum_{j=1}^{n}\boldsymbol{b}_j\right|\right|_\infty+\frac{1}{m}\cdot d_{\text{lev}}\left(\boldsymbol{A},\boldsymbol{B}\right)\\ [|m-n|\leq d_{\text{lev}}\left(\boldsymbol{A},\boldsymbol{B}\right)+\text{Triang. ineq.}] &\leq \left(\frac{1}{m\cdot n}\cdot\sum_{j=1}^{n}||\boldsymbol{b}_j||_\infty+\frac{1}{m}\right)\cdot d_{\text{lev}}\left(\boldsymbol{A},\boldsymbol{B}\right)\\ [||\boldsymbol{b}_j||_\infty=1\ \ \forall j\in[n]] &= \frac{2}{m}\cdot d_{\text{lev}}\left(\boldsymbol{A},\boldsymbol{B}\right)\,.\end{aligned}$$

□

Lemma S7 (Linear transformations)**.** *Let $\boldsymbol{A},\boldsymbol{B}\in\mathcal{X}_d$ be two sequences and $V\in\mathbb{R}^{d\times k}$. We have that:*

$$d_{ERP}(\boldsymbol{AV},\boldsymbol{BV})\leq d_{ERP}(\boldsymbol{A},\boldsymbol{B})\,||\boldsymbol{V}||_p$$

In the case of sequences of one-hot vectors, we have that:

$$d_{ERP}(\boldsymbol{AV},\boldsymbol{BV})\leq d_{Lev}(\boldsymbol{A},\boldsymbol{B})\cdot M(\boldsymbol{V})\,,$$

where

$$M(\boldsymbol{V})=\max\{\max_{i\in[d]}||\boldsymbol{v}_i||_p\,,\,\max_{i,j\in[d]}||\boldsymbol{v}_i-\boldsymbol{v}_j||_p\}$$

Proof. Follows immediately from Definition 3.1 and the fact that $||\boldsymbol{AB}||\leq||\boldsymbol{A}||\,||\boldsymbol{B}||$ for any matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. The second result for one-hot vectors follows immediately from the fact that the biggest change in the embedding sequence that can be produced from a single-character change, is either given by inserting the character with the largest norm embedding (left side of the $\max$), or replacing a character with the character that has the furthest away embedding in the ℓ_p norm (left side of the $\max$). □

Lemma S8 (Elementwise Lipschitz functions)**.** *Let d_{ERP} be as in Definition 3.1. Let $\boldsymbol{A}\in\mathbb{R}^{m\times d}$ and $\boldsymbol{B}\in\mathbb{R}^{n\times d}$ be two sequences. Let $f:\mathbb{R}^d\to\mathbb{R}^k$ be a Lipschitz function so that:*

$$||\boldsymbol{f}(\boldsymbol{a})-\boldsymbol{f}(\boldsymbol{b})||_p\leq L_f\cdot||\boldsymbol{a}-\boldsymbol{b}||_p\ \ \forall\boldsymbol{a},\boldsymbol{b}\in\mathbb{R}^d\,.$$

Let $\boldsymbol{F}(\boldsymbol{A})\in\mathbb{R}^{m\times k}$ and $\boldsymbol{F}(\boldsymbol{B})\in\mathbb{R}^{n\times k}$ be the application of f to every vector in both sequences, we immediately have that:

$$d_{ERP}(\boldsymbol{F}(\boldsymbol{A}),\boldsymbol{F}(\boldsymbol{B}))\leq L_f\cdot d_{ERP}(\boldsymbol{A},\boldsymbol{B})$$

Lemma S9 (Convolution). *Let d_{ERP} be as in Definition 3.1. Let $\boldsymbol{P} \in \{0,1\}^{m\times d}$ and $\boldsymbol{Q} \in \{0,1\}^{n\times d}$ be two sequences of m and n one hot-vectors respectively. Let the function working with arbitrary sequence length l be $\boldsymbol{F} : \{0,1\}^{l\times d} \to \mathbb{R}^{l\times r}$. Let the convolutional filter $\boldsymbol{C} : \mathbb{R}^{l\times r} \to \mathbb{R}^{(l+q-1)\times k}$ with kernel $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{q\times k\times r}$, where q is the kernel size and k is the number of filters. We have that:*

$$d_{ERP}\left(\boldsymbol{C}(\boldsymbol{F}(\boldsymbol{P})), \boldsymbol{C}(\boldsymbol{F}(\boldsymbol{Q}))\right) \quad \leq M(\boldsymbol{\mathcal{K}}) \cdot d_{ERP}\left(\boldsymbol{F}(\boldsymbol{P}), \boldsymbol{F}(\boldsymbol{Q})\right) \,.$$

where:

$$M(\boldsymbol{\mathcal{K}}) = \sum_{i=1}^{q} ||\boldsymbol{K}_i||_p \,.$$

Proof of Lemma S9. Let $L = m + n + 2q - 2$. Starting from the definition of the ERP distance in Lemma S4 and the definition of the convolutional layer in Definition 3.2:

$$\begin{aligned}
& d_{\text{ERP}}(\boldsymbol{C}(\boldsymbol{F}(\boldsymbol{P})), \boldsymbol{C}(\boldsymbol{F}(\boldsymbol{Q}))) \\
&= \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m+q-1,L},\\ \boldsymbol{Z}^b\in\mathcal{Z}_{n+q-1,L}}} \sum_{k=1}^{L} \left|\left|\boldsymbol{z}_k^a(\boldsymbol{C}(\boldsymbol{F}(\boldsymbol{P}))) - \boldsymbol{z}_k^b(\boldsymbol{C}(\boldsymbol{F}(\boldsymbol{Q})))\right|\right|_p \\
&= \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m+q-1,L},\\ \boldsymbol{Z}^b\in\mathcal{Z}_{n+q-1,L}}} \sum_{k=1}^{L} \left|\left|\boldsymbol{z}_k^a\left(\left[\sum_{j=1}^{m} \hat{\boldsymbol{K}}_{i,j}\hat{\boldsymbol{f}}_j(\boldsymbol{P})\right]_{i=1}^{m+q-1}\right) - \boldsymbol{z}_k^b\left(\left[\sum_{l=1}^{n} \hat{\boldsymbol{K}}_{i,l}\hat{\boldsymbol{f}}_l(\boldsymbol{Q})\right]_{i=1}^{n+q-1}\right)\right|\right|_p \\
&= \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m+q-1,L},\\ \boldsymbol{Z}^b\in\mathcal{Z}_{n+q-1,L}}} \sum_{k=1}^{L} \left|\left|\boldsymbol{z}_k^a\left(\left[\sum_{j=1}^{q} \boldsymbol{K}_j\boldsymbol{f}_{i+j-1}(\boldsymbol{P})\right]_{i=1}^{m+q-1}\right) - \boldsymbol{z}_k^b\left(\left[\sum_{j=1}^{q} \boldsymbol{K}_j\boldsymbol{f}_{i+j-1}(\boldsymbol{Q})\right]_{i=1}^{n+q-1}\right)\right|\right|_p \\
&= \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m+q-1,L},\\ \boldsymbol{Z}^b\in\mathcal{Z}_{n+q-1,L}}} \sum_{k=1}^{L} \left|\left|\sum_{j=1}^{q} \boldsymbol{K}_j\left(\boldsymbol{z}_k^a\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{P})]_{i=1}^{m+q-1}\right) - \boldsymbol{z}_k^b\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{Q})]_{i=1}^{n+q-1}\right)\right)\right|\right|_p \\
&\leq \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m+q-1,L},\\ \boldsymbol{Z}^b\in\mathcal{Z}_{n+q-1,L}}} \sum_{k=1}^{L}\sum_{j=1}^{q} ||\boldsymbol{K}_j||_p \cdot \left|\left|\boldsymbol{z}_k^a\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{P})]_{i=1}^{m+q-1}\right) - \boldsymbol{z}_k^b\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{Q})]_{i=1}^{n+q-1}\right)\right|\right|_p \\
&= \min_{\substack{\boldsymbol{Z}^a\in\mathcal{Z}_{m+q-1,L},\\ \boldsymbol{Z}^b\in\mathcal{Z}_{n+q-1,L}}} \sum_{j=1}^{q} ||\boldsymbol{K}_j||_p \cdot \sum_{k=1}^{L} \left|\left|\boldsymbol{z}_k^a\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{P})]_{i=1}^{m+q-1}\right) - \boldsymbol{z}_k^b\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{Q})]_{i=1}^{n+q-1}\right)\right|\right|_p \\
&= \sum_{j=1}^{q} ||\boldsymbol{K}_j||_p \cdot d_{\text{ERP}}\left(\boldsymbol{F}(\boldsymbol{P}), \boldsymbol{F}(\boldsymbol{Q})\right) \,,
\end{aligned}$$

□

where the last equality follows from the fact that $[\boldsymbol{f}_{i+j-1}(\boldsymbol{P})]_{i=1}^{m+q-1}$ and $[\boldsymbol{f}_{i+j-1}(\boldsymbol{Q})]_{i=1}^{n+q-1}$ are just windows of $\boldsymbol{F}(\boldsymbol{P})$ and $\boldsymbol{F}(\boldsymbol{Q})$ respectively including the complete sequences $\boldsymbol{F}(\boldsymbol{P})$ and $\boldsymbol{F}(\boldsymbol{Q})$, resulting in $d_{\text{ERP}}\left([\boldsymbol{f}_{i+j-1}(\boldsymbol{P})]_{i=1}^{m+q-1}, [\boldsymbol{f}_{i+j-1}(\boldsymbol{Q})]_{i=1}^{n+q-1}\right) = d_{\text{ERP}}\left(\boldsymbol{F}(\boldsymbol{P}), \boldsymbol{F}(\boldsymbol{Q})\right) \;\; \forall j = 1, \cdots, q$.

Proof of Theorem 3.3. We will bound the absolute value of the difference of outputs for two sentences $\boldsymbol{P}, \boldsymbol{Q} \in \mathcal{S}(\Gamma)$ of lengths m and n respectively:

$$
\begin{aligned}
|f(\boldsymbol{P}) - f(\boldsymbol{Q})| &\coloneqq \left(\sum_{i=1}^{m+l\cdot(q-1)} \boldsymbol{\sigma}\left(\boldsymbol{c}_i^{(l)}\left(\boldsymbol{PE}\right)\right) - \sum_{j=1}^{n+l\cdot(q-1)} \boldsymbol{\sigma}\left(\boldsymbol{c}_j^{(l)}\left(\boldsymbol{QE}\right)\right) \right) \boldsymbol{w} \\
\text{[Hölder's inequality]} &\leq ||\boldsymbol{w}||_r \cdot \left|\left| \sum_{i=1}^{m+l\cdot(q-1)} \boldsymbol{\sigma}\left(\boldsymbol{c}_i^{(l)}\left(\boldsymbol{PE}\right)\right) - \sum_{j=1}^{n+l\cdot(q-1)} \boldsymbol{\sigma}\left(\boldsymbol{c}_j^{(l)}\left(\boldsymbol{QE}\right)\right) \right|\right|_p \\
\text{[Lemma S5]} &\leq ||\boldsymbol{w}||_r \cdot d_{\text{ERP}}\left(\boldsymbol{\sigma}\left(\boldsymbol{C}^{(l)}\left(\boldsymbol{PE}\right)\right), \boldsymbol{\sigma}\left(\boldsymbol{C}^{(l)}\left(\boldsymbol{QE}\right)\right)\right) \\
\text{[Lemma S8 and Lemma S9 recursively]} &\leq ||\boldsymbol{w}||_r \cdot \left(\prod_{k=1}^{l} M(\boldsymbol{\mathcal{K}}^{(k)}) \right) \cdot d_{\text{ERP}}\left(\boldsymbol{PE}, \boldsymbol{QE}\right) \\
\text{[Lemma S7]} &\leq ||\boldsymbol{w}||_r \cdot \left(\prod_{k=1}^{l} M(\boldsymbol{\mathcal{K}}^{(k)}) \right) \cdot M(\boldsymbol{E}) \cdot d_{\text{Lev}}\left(\boldsymbol{P}, \boldsymbol{Q}\right)
\end{aligned}
$$

□