Robust Encodings: A Framework for Combating Adversarial Typos

Erik Jones **Robin Jia*** **Aditi Raghunathan*** **Percy Liang**
Computer Science Department, Stanford University
{erjones,robinjia,aditir,pliang}@stanford.edu

Abstract

Despite excellent performance on many tasks, NLP systems are easily fooled by small adversarial perturbations of inputs. Existing procedures to defend against such perturbations are either (i) heuristic in nature and susceptible to stronger attacks or (ii) provide guaranteed robustness to worst-case attacks, but are incompatible with state-of-the-art models like BERT. In this work, we introduce *robust encodings* (RobEn): a simple framework that confers guaranteed robustness, without making compromises on model architecture. The core component of RobEn is an *encoding function*, which maps sentences to a smaller, discrete space of encodings. Systems using these encodings as a bottleneck confer guaranteed robustness with *standard training*, and the same encodings can be used across multiple tasks. We identify two desiderata to construct robust encoding functions: perturbations of a sentence should map to a small set of encodings (stability), and models using encodings should still perform well (fidelity). We instantiate RobEn to defend against a large family of adversarial typos. Across six tasks from GLUE, our instantiation of RobEn paired with BERT achieves an average robust accuracy of 71.3% against *all* adversarial typos in the family considered, while previous work using a typo-corrector achieves only 35.3% accuracy against a simple greedy attack.

1 Introduction

State-of-the-art NLP systems are brittle: small perturbations of inputs, commonly referred to as adversarial examples, can lead to catastrophic model failures (Belinkov and Bisk, 2018; Ebrahimi et al., 2018b; Ribeiro et al., 2018; Alzantot et al., 2018). For example, carefully chosen typos and word substitutions have fooled systems for hate speech detection (Hosseini et al., 2017), machine translation (Ebrahimi et al., 2018a), and spam filtering (Lee and Ng, 2005), among others.

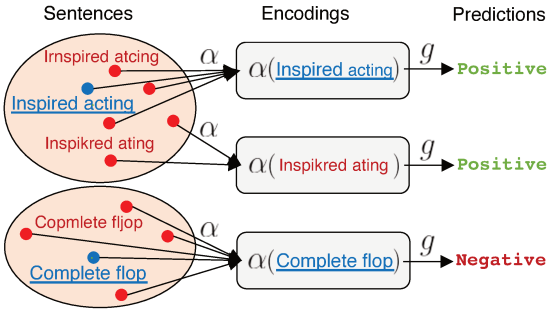


Figure 1: Example of a defense using RobEn. An adversary can perturb sentences (blue, underlined) to many different perturbations (red, not-underlined) within the attack surface (red, ovals). We define an encoding function α such that each perturbation of the input sentences maps to one of a few encodings (grey, rounded rectangles). We can then use any model g to make predictions given the encodings.

We aim to build systems that achieve high *robust accuracy*: accuracy against worst-case attacks. Broadly, existing methods to build robust models fall under one of two categories: (i) adversarial training, which augments the training set with heuristically generated perturbations and (ii) certifiably robust training, which bounds the change in prediction between an input and *any* of its allowable perturbations. Both these approaches have major shortcomings, especially in NLP. Adversarial training, while quite successful in vision (Madry et al., 2018), is challenging in NLP due to the discrete nature of textual inputs (Ebrahimi et al., 2018b); current techniques like projected gradient descent are incompatible with subword tokenization. Further, adversarial training relies on heuristic approximations to the worst-case perturbations, leaving models vulnerable to new, stronger attacks. Certifiably robust training (Jia et al., 2019; Huang et al., 2019; Shi et al., 2020) circumvents

*Authors contributed equally.

the above challenges by optimizing over a convex outer-approximation of the set of perturbations, allowing us to lower bound the true robust accuracy. However, the quality of bounds obtained by these methods scale poorly with the size of the network, and are vacuous for state-of-the-art models like BERT. Moreover, both approaches require separate, expensive training for each task, even when defending against the same type of perturbations.

Ideally we would like a "robustness" module that we can reuse across multiple tasks, allowing us to only worry about robustness once: during its construction. Indeed, *reusable* components have driven recent progress in NLP. For example, word vectors are a universal resource that are constructed once, then used for many different tasks. Can we build a reusable robust defense that can easily work with complex, state-of-the-art architectures like BERT? The recent work of Pruthi et al. (2019), which uses a typo-corrector to defend against adversarial typos, is such a reusable defense: it is trained once, then reused across different tasks. However, we find that current typo-correctors do not perform well against even heuristic attacks, limiting their applicability.

Our primary contribution is *robust encodings* (RobEn), a framework to construct encodings that can make systems using *any* model robust. The core component of RobEn is an *encoding function* that maps sentences to a smaller discrete space of encodings, which are then used to make predictions. We define two desiderata that a robust encoding function should satisfy: stability and fidelity. First, to encourage consistent predictions across perturbations, the encoding function should map all perturbations of a sentence to a small set of encodings (stability). Simultaneously, encodings should remain expressive, so models trained using encodings still perform well on unperturbed inputs (fidelity). Because systems using RobEn are encoding-based we can compute the *exact* robust accuracy tractably, avoiding the lower bounds of certifiably robust training. Moreover, these encodings can make any downstream model robust, including state-of-the-art transformers like BERT, and can be reused across different tasks.

In Section 4, we apply RobEn to combat adversarial typos. In particular, we allow an attacker to add independent edit distance one typos to each word in an input sentence, resulting in exponentially more possible perturbations than previous work (Pruthi et al., 2019; Huang et al., 2019). We consider a natural class of *token-level encodings*, which are obtained by encoding each token in a sentence independently. This structure allows us to express stability and fidelity in terms of a clustering objective, which we optimize.

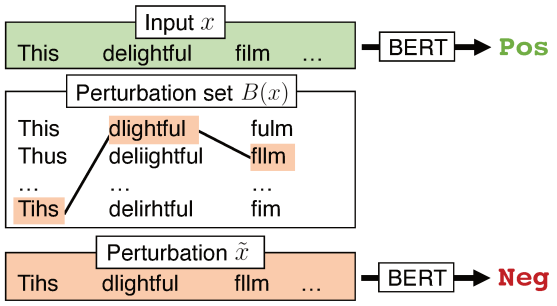


Figure 2: Attack model allowing independent perturbations of each token. The original input, x is classified by the model as positive while the perturbation $\tilde{x}$ =, obtained by choosing perturbations of *"This"*, *"delightful"*, and *"film"* independently, is classified as negative. Independent perturbations of each word results in an exponentially large perturbation space $B(x)$.

Empirically, our instantiation of RobEn achieves state-of-the-art robust accuracy, which we compute exactly, across six classification tasks from the GLUE benchmark (Wang et al., 2019). Our best system, which combines RobEn with a BERT classifier (Devlin et al., 2019), achieves an average robust accuracy of 71.3% across the six tasks. In contrast, a state-of-the-art defense that combines BERT with a typo corrector (Pruthi et al., 2019) gets 35.3% accuracy when adversarial typos are inserted, and a standard data augmentation defense gets only 12.2% accuracy.

2 Setup

Tasks. We consider NLP tasks that require classifying textual input $x \in \mathcal{X}$ to a class $y \in \mathcal{Y}$. For simplicity, we refer to inputs as sentences. Each sentence x consists of tokens $x_1, \ldots, x_L$ from the set of all strings $\mathcal{T}$. Let p_{task} denote the distribution over inputs and labels for a particular task of interest. The goal is to learn a model $f : \mathcal{X} \to \mathcal{Y}$ that maps sentences to labels, given training examples $(x, y) \sim p_{\text{task}}$.

Attack surface. We consider an attack surface in which an adversary can perturb each token x_i of a sentence to some token $\tilde{x}_i \in B(x_i)$, where $B(x_i)$ is the set of valid perturbations of x_i. For example, $B(x_i)$ could be a set of allowed typos of x_i. We define $B(x)$ as the set of all valid perturbations

of the set x, where every possible combination of token-level typos is allowed:

$$B(x) = \{(\tilde{x_1}, \ldots, \tilde{x_L}) \mid \tilde{x_i} \in B(x_i) \, \forall \, i\} \quad (1)$$

The size of the attack surface $|B(x)|$ grows exponentially with respect to number of input tokens, as shown in Figure 2. In general $x_i \in B(x_i)$, so some words could remain unperturbed.

Model evaluation. In this work, we use three evaluation metrics for any given task.

First, we evaluate a model on its *standard accuracy* on the task:

$$\text{acc}_{\text{std}}(f) = \mathbb{E}_{(x,y)\sim p_{\text{task}}} \mathbf{1}[f(x) = y]. \quad (2)$$

Next, we are interested in models that also have high *robust accuracy*, the fraction of examples (x, y) for which the model is correct on all valid perturbations $\tilde{x} \in B(x)$ allowed in the attack model:

$$\text{acc}_{\text{rob}}(f) = \mathbb{E}_{(x,y)\sim p_{\text{task}}} \min_{\tilde{x}\in B(x)} \mathbf{1}\left[f(\tilde{x}) = y\right]. \quad (3)$$

It is common to instead compute accuracy against a heuristic attack a that maps clean sentences x to perturbed sentences $a(x) \in B(x)$.

$$\text{acc}_{\text{attack}}(f; a) = \mathbb{E}_{(x,y)\sim p_{\text{task}}} \mathbf{1}[f(a(x)) = y]. \quad (4)$$

Typically, $a(x)$ is the result of a heuristic search for a perturbation $\tilde{x} \in B(x)$ that f misclassifies. Note that $\text{acc}_{\text{attack}}$ is a (possibly loose) upper bound of acc_{rob} because there could be perturbations that the model misclassifies but are not encountered during the heuristic search (Athalye et al., 2018).

Additionally, since robust accuracy is generally hard to compute, some existing work computes *certified accuracy* (Huang et al., 2019; Jia et al., 2019; Shi et al., 2020), which is a potentially conservative lower bound for the true robust accuracy. In this work, since we use robust encodings, we can tractably compute the exact robust accuracy.

3 Robust Encodings

We introduce *robust encodings* (RobEn), a framework for constructing encodings that are reusable across many tasks, and pair with arbitrary model architectures. In Section 3.1 we describe the key components of RobEn, then in Section 3.2 we highlight desiderata RobEn should satisfy.

3.1 Encoding functions

A RobEn classifier $f_\alpha : \mathcal{X} \to \mathcal{Y}$ using RobEn decomposes into two components: a *fixed* encoding function $\alpha : \mathcal{X} \to \mathcal{Z}$, and a model that accepts encodings $g : \mathcal{Z} \to \mathcal{Y}$.[1] For any sentence x, our system makes the prediction $f_\alpha(x) = g(\alpha(x))$. Given training data $\{(x_i, y_i)\}_{i=1}^n$ and the encoding function α, we learn g by performing standard training on encoded training points $\{(\alpha(x_i), y_i)\}_{i=1}^n$. To compute the robust accuracy of this system, we note that for well-chosen α and an input x from some distribution P_{x}, the set of possible encodings $\alpha(\tilde{x})$ for some perturbation $\tilde{x} \in B(x)$ is both small and tractable to compute quickly. We can thus compute $\text{acc}_{\text{rob}}(f_\alpha)$ quickly by generating this set of possible encodings, and feeding each into g, which can be any architecture.

3.2 Encoding function desiderata

In order to achieve high robust accuracy, a classifier f_α that uses α should make consistent predictions on all $\tilde{x} \in B(x)$, the set of points described by the attack surface, and also have high standard accuracy on unperturbed inputs. We term the former property *stability*, and the latter *fidelity*, give intuition for both in this section, and provide a formal instantiation in Section 4.

Stability. For an encoding function α and some distribution over inputs P_{x}, the stability $\text{Stab}(\alpha)$ measures how often α maps sentences $x \sim P_{\mathsf{x}}$ to the same encoding as all of their perturbations.

Fidelity. An encoding function α has high fidelity if models that use α can still achieve high standard accuracy. Unfortunately, while we want to make task agnostic encoding functions, standard accuracy is inherently *task dependent*: different tasks have different expected distributions over inputs and labels. To emphasize this challenge consider two tasks: for an integer n, predict n mod 2, and n mod 3. The information we need encodings to preserve varies significantly between these tasks: for the former, 2 and 6 can be identically encoded, while for the latter they must encoded separately.

To overcome this challenge, we consider a single distribution over the inputs P_{x} that we believe covers many task-distributions p_{task}. Since it is hard to model the distribution over the labels, we take the more conservative approach of mapping

[1] We can set $\mathcal{Z} \subseteq \mathcal{X}$ when g accepts sentences.

the different sentences sampled from P_{x} to different encodings with high probability. We call this $\mathrm{Fid}(\alpha)$, and give an example in Section 4.5.

Tradeoff. Stability and fidelity are inherently competing goals. An encoding function that maps every sentence to the same encoding trivially maximizes stability, but is useless for any non-trivial classification task. Conversely, fidelity is maximized when every input is mapped to itself, which has very low stability. In the following section, we construct an instantiation of RobEn that balances stability and fidelity when the attack surface consists of typos.

4 Robust Encodings for Typos

In this section, we focus on *adversarial typos*, where an adversary can add typos to each token in a sentence (see Figure 2). Since this attack surface is defined at the level of tokens, we restrict attention to encoding functions that encode each token independently. Such an encoding does not use contextual information; we find that even such robust encodings achieve greater attack accuracy and robust accuracy in practice than previous work.

First, we will reduce the problem of generating token level encodings to assigning vocabulary words to clusters (Section 4.1). Next, we use an example to motivate different clustering approaches (Section 4.2), then describe how we handle out-of-vocabulary tokens (Section 4.3). Finally, we introduce two types of token-level robust encodings: connected component encodings (Section 4.4) and agglomerative cluster encodings (Section 4.5).

4.1 Encodings as clusters

We construct an encoding function α that encodes x token-wise. Formally, α is defined by a token-level encoding function π that maps each token $x_i \in \mathcal{T}$ to some *encoded token* $\pi(x_i) \in \mathcal{Z}_{\text{Tok}}$:

$$\alpha(x) = [\pi(x_1), \pi(x_2), \ldots \pi(x_L)]. \qquad (5)$$

In the RobEn pipeline, a downstream model g is trained on encodings (Section 3.1). If π maps many words and their typos to the same encoded token, they become indistinguishable to g, conferring robustness. In principle, the relationship between different encoded tokens is irrelevant: during training, g learns how to use the encoded tokens to perform a desired task. Thus, the problem of finding a good π is equivalent to deciding which tokens should share the same encoded token.

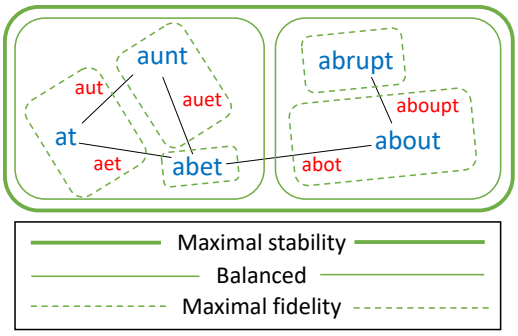


Figure 3: Visualization of three different encodings. Vocabulary words (large font, blue) share an edge if they share a common perturbation (small font, red). The maximal stability cluster (thick solid line) clusters identically, the maximal fidelity clusters (thin dotted line) encodes all words separately, while the balanced clusters (thin solid line) trade off the two.

Since the space of possible tokens $\mathcal{T}$ is innumerable, we focus on a smaller set of words $V = \{w_1, \ldots, w_N\} \subseteq \mathcal{T}$, which contains the N most frequent words over P_{x}. We will call elements of V *words*, and tokens that are perturbations of some word *typos*. We view deciding which words should share an encoded token as assigning words to clusters $C_1, \ldots, C_k \subseteq V$. For all other tokens not in the vocabulary, including typos, we define a separate π_{OOV}. Thus, we decompose π as follows:

$$\pi(x_i) = \begin{cases} \pi_V(x_i) & x_i \in V \\ \pi_{\text{OOV}}(x_i) & x_i \notin V \end{cases}, \qquad (6)$$

Here, π_V is associated with a clustering C of vocabulary words, where each cluster is associated with a unique encoded token.

4.2 Simple example

We use a simple example to illustrate how a token-level encoding function can achieve the RobEn desiderata: stability and fidelity defined in Section 3.2. We will formally define the stability and fidelity of a clustering in Sections 4.3 and 4.5.

Consider the five words (large font, blue) in Figure 3, along with potential typos (small font, red). We illustrate three different clusterings as boxes around tokens in the same cluster. We may put all words in the same cluster (thick box), each word in its own cluster (dashed boxes), or something in between (thin solid boxes). For now, we group each typo with a word it could have been perturbed from (we will discuss this further in Section 4.3).

To maximize stability, we need to place all words in the same cluster. Otherwise, there would be two

words (say *"at"* and *"aunt"*) that could both be perturbed to the same typo (*"aut"*) but are in different clusters. Therefore, *"aut"* cannot map to the same encoded token as both the possible vocab words. At the other extreme, to maximize fidelity, each word should be in its own cluster. Both mappings have weaknesses: the stability-maximizing mapping has low fidelity since all words are identically encoded and thus indistinguishable, while the fidelity-maximizing mapping has low stability since the typos of words *"aunt"*, *"abet"*, and *"abrupt"* could all be mapped to different encoded tokens than that of the original word.

The clustering represented by the thin solid boxes in Figure 3 balances stability and fidelity. Compared to encoding all words identically, it has higher fidelity, since it distinguishes between some of the words (e.g., *"at"* and *"about"* are encoded differently). It also has reasonably high stability, since only the infrequent *"abet"* has typos that are shared across words and hence are mapped to different encoded tokens.

4.3 Encoding out-of-vocab tokens

Given a fixed clustering of V, we now study how to map out-of-vocabulary tokens, including typos, to encoded tokens without compromising stability.

Stability. Stability measures the extent to which typos of words map to different encoded tokens. We formalize this by defining the set of tokens that some typo of a word w could map to, $B_\pi(w)$:

$$B_\pi(w) = \{\pi(\tilde{w}); \tilde{w} \in B(w)\}, \qquad (7)$$

where $B(w)$ is the set of allowable typos of w. Since we care about inputs drawn from P_{x}, we define Stab on the clustering C using $\rho(w)$, the normalized frequency of word w based on P_{x}.

$$\text{Stab}(C) = -\sum_{i=1}^{N} \rho(w_i)|B_\pi(w_i)| \qquad (8)$$

For a fixed clustering, the size of $B_\pi(w)$ depends on where π_{OOV} maps typos that w shares with other words; for example in Figure 3, *"aet"* could be a perturbation of both *"at"* and *"abet"*. If we map the typo the encoded token of *"at"*, we increase the size of B_π("abet") and vice-versa. In order to keep the size of $B_\pi(w)$ smaller for the more frequent words and maximize stability (Equation 8), we map a typo to the same encoded token as its most frequent neighbor word (in this case *"at"*). Finally, when a token is not a typo of any vocab words, we encode it to a special token OOV.

4.4 Connected component encodings

We present two approaches to generate robust token-level encodings. Our first method, connected component encodings, maximizes the stability objective (8). Notice that Stab is maximized when for each word w, $B_\pi(w)$ contains one encoded token. This is possible only when all words that share a typo are assigned to the same cluster.

To maximize Stab, define a graph G with all words in V as vertices, and edges between words that share a typo. Since we must map words that share an edge in G to the same cluster, we define the cluster C_i to be the set of words in the i^{th} connected component of G. While this stability-maximizing clustering encodes many words to the same token (and hence seems to compromise on fidelity), these encodings still perform surprisingly well in practice (see Section 5.4).

4.5 Agglomerative cluster encodings

Connected component encodings focus only stability and can lead to needlessly low fidelity. For example, in Figure 3, *"at"* and *"about"* are in the same connected component even though they don't share a typo. Since both words are generally frequent, mapping them to different encoded tokens can significantly improve fidelity, with only a small drop in stability: recall only the infrequent word *"abet"* can be perturbed to multiple encoded tokens.

To handle such cases, we introduce *agglomerative cluster encodings*, which we construct by trading off Stab with a formal objective we define for fidelity: Fid. We then approximately optimize this combined objective Φ using an agglomerative clustering algorithm.

Fidelity objective. Recall from Section 3.2 that an encoding has high fidelity if it can be used to achieve high standard accuracy on many tasks. This is hard to precisely characterize: we aim to design an objective that could approximate this.

We note that distinct encoded tokens are arbitrarily related: the model g learns how to use different encodings during training. Returning to our example, suppose *"at"* and *"abet"* belong to the same cluster and share an encoded token z. During training, each occurrence of *"at"* and *"abet"* is replaced with z. However, since *"at"* is much more frequent, classifiers treat z similarly to "at"

in order to achieve good overall performance. This leads to mostly uncompromised performance on sentences with *"at"*, at the cost of performance on sentences containing the less frequent *"abet"*.

This motivates the following definition: let $\vec{v_i}$ be a the indicator vector in $\mathbb{R}^{|V|}$ corresponding to word i. In principle $\vec{v_i}$ could be a word embedding; we choose indicator vectors to avoid making additional assumptions. We define the encoded token $\vec{\mu_j}$ associated with words in cluster C_j as follows:

$$\vec{\mu_j} = \frac{\sum_{w_i \in C_j} \rho(w_i)\vec{v_i}}{\sum_{w_i \in C_j} \rho(w_i)} \tag{9}$$

We weight by the frequency ρ to capture the effect of training on the encodings, as described above.

Fidelity is maximized when each word has a distinct encoded token. We capture the drop in standard accuracy due to shared encoded tokens by computing the distance between the original embeddings of the word its encoded token. Formally, let $c(i)$ be the cluster index of word w_i. We define the fidelity objective Fid as follows:

$$\mathrm{Fid}(C) = -\sum_{i=1}^{N} \rho(w_i)\|\vec{v_i} - \vec{\mu}_{c(i)}\|^2. \tag{10}$$

Fid is high if frequent words and rare words are in the same cluster and is low when when multiple frequent words are in the same cluster.

Final objective. We introduce a hyperparameter $\gamma \in [0, 1]$ that balances stability and fidelity. We approximately minimize the following weighted combination of Stab (8) and Fid (10):

$$\Phi(C) = \gamma\,\mathrm{Fid}(C) + (1-\gamma)\,\mathrm{Stab}(C). \tag{11}$$

As γ approaches 0, we get the connected component clusters from our baseline, which maximize stability. As γ approaches 1, we maximize fidelity by assigning each word to its own cluster.

Agglomerative clustering. We approximate the optimal value of Φ using *agglomerative clustering*; we start with each word in its own cluster, then iteratively combine the pair of clusters whose resulting combination increases Φ the most. We repeat until combining any pair of clusters would decrease Φ. Further details are provided in Appendix A.1.

5 Experiments

5.1 Setup

Token-level attacks. The primary attack surface we study is edit distance one (ED1) perturbations. For every word in the input, the adversary is allowed to insert a lowercase letter, delete a character, substitute a character for any letter, or swap two adjacent characters, so long as the first and last characters remain the same as in the original token. The constraint on the outer characters, also used by Pruthi et al. (2019), is motivated by psycholinguistic studies (Rawlinson, 1976; Davis, 2003).

Within our attack surface, *"the movie was miserable"* can be perturbed to *"thae mvie wjs misreable"* but not *"th movie as miserable"*. Since each token can be independently perturbed, the number of perturbations of a sentence grows exponentially with its length; even *"the movie was miserable"* has 431,842,320 possible perturbations. Our attack surface contains the attack surface used by (Pruthi et al., 2019), which allows ED1 perturbations to at most two words per sentence. Reviews from SST-2 have 5 million perturbations per example (PPE) on average under this attack surface, while our attack surface averages 10^{97} PPE. We view the size of the attack surface as a strength of our approach: our attack surface forces a system robust to subtle perturbations (*"the moviie waas misreable"*) that smaller attack surfaces miss.

In Section 5.7, we additionally consider the internal permutation attacks studied in Belinkov and Bisk (2018) and Sakaguchi et al. (2017), where all characters, except the first and the last, may be arbitrarily reordered.

Attack algorithms. We consider two attack algorithms: the worst-case attack (WCA) and a beam-search attack (BSA). WCA exhaustively tests every possible perturbation of an input x to see any change in the prediction. The attack accuracy of WCA is the true robust accuracy since if there exists some perturbation that changes the prediction, WCA finds it. When instances of RobEn have high stability, the number of possible encodings of perturbations of x is often small, allowing us to exhaustively test all possible perturbations in the encoding space.[2] This allows us to tractably run WCA. Using WCA with RobEn, we can obtain computationally tractable *guarantees* on robustness: given a sentence, we can quickly compute whether or not any perturbation of x that changes the prediction.

For systems that don't use RobEn, we cannot tractably run WCA. Instead, we run a beam search

[2]When there are more than 10000 possible encodings, which holds for 0.009% of our test examples, we assume the adversary successfully alters the prediction.

attack (BSA) with beam width 5, perturbing tokens one at a time. For efficiency, we sample at most $\text{len}(x_i)$ perturbations at each step of the search (see Apendix A.2). Even against this very limited attack, we find that baseline models have low accuracy.

Datasets. We use six of the nine tasks from GLUE (Wang et al., 2019): SST-2, MRPC, QQP, MNLI, QNLI, and RTE. We do not use STS-B and CoLA as they are evaluated on correlation, which does not decompose as an example-level loss. We additionally do not use WNLI, as most submitted GLUE models cannot even outperform the majority baseline, and state-of-the-art models are rely on external training data (Kocijan et al., 2019). We evaluate on the test sets for SST-2 and MRPC, and the publicly available dev sets for the remaining tasks. More details are provided in Appendix A.3.

5.2 Baseline models.

We consider three baseline systems. Our first is the standard base uncased BERT model (Devlin et al., 2019) fine-tuned on the training data for each task.[3]

Data augmentation. For our next baseline, we augment the training dataset with four random perturbations of each example, then fine-tune BERT on this augmented data. Data augmentation has been shown to increase robustness to some types of adversarial perturbations (Ribeiro et al., 2018; Liu et al., 2019). Other natural baselines all have severe limitations. Adversarial training with black-box attacks offers limited robustness gains over data augmentation (Cohen et al., 2019; Pruthi et al., 2019). Projected gradient descent (Madry et al., 2017), the only white-box adversarial training method that is robust in practice, cannot currently be applied to BERT since subword tokenization maps different perturbations to different numbers of tokens, making gradient-based search impossible. Certifiably robust training (Huang et al., 2019; Shi et al., 2020) does not work with BERT due to the same tokenization issue and BERT's use of non-monotonic activation functions, which make computing bounds intractable. Moreover the bounds computed with certifiably robust training, which give guarantees, become loose as model depth increases, hurting robust performance (Gowal et al., 2018).

Typo-corrector. For our third baseline, we use the most robust method from Pruthi et al. (2019). In particular, we train a scRNN typo-corrector (Sakaguchi et al., 2017) on random perturbations of each task's training set. At test time inputs are "corrected" using the typo corrector, then fed into a downstream model. We replace any OOV outputted by the typo-corrector with the neutral word *"a"* and use BERT as our downstream model.

5.3 Models with RobEn

We run experiments using our two token-level encodings: connected component encodings (CONNCOMP) and agglomerative cluster encodings (AGGCLUST). To form clusters, we use the $N = 100,000$ most frequent words from the Corpus of Contemporary American English (Davies, 2008) that are also in GloVe (Pennington et al., 2014). For AGGCLUST we use $\gamma = 0.3$, which maximizes robust accuracy on SST-2 dev set.

Form of encodings. Though unnecessary when training from scratch, to leverage the inductive biases of pre-trained models like BERT (Devlin et al., 2019), we define the encoded token of a cluster to be the cluster's most frequent member word. In the special case of the out-of-vocab token, we map OOV to [MASK]. Our final encoding, $\alpha(x)$, is the concatenation of all of these words. For both encodings, we fine-tune BERT on the training data, using $\alpha(x)$ as input. Further details are in Appendix A.4.

5.4 Robustness gains from RobEn

Our main results are shown in Table 1. We show all three baselines, as well as models using our instances of RobEn: CONNCOMP and AGGCLUST.

Even against the heuristic attack, each baseline system suffers dramatic performance drops. The system presented by Pruthi et al. (2019), Typo Corrector + BERT, only achieves 35.3% attack accuracy, compared to its standard accuracy of 78.2%. BERT and Data Augmentation + BERT perform even worse. Moreover, the number of perturbations the heuristic attack explores is a tiny fraction of our attack surface, so the robust accuracy of Typo Corrector + BERT, the quantity we'd like to measure, is likely far lower than the attack accuracy.

In contrast, simple instances of RobEn are much more robust. AGGCLUST + BERT achieves average robust accuracy of 71.3%, 36 points higher than the attack accuracy of Typo Corrector + BERT. AGGCLUST also further improves on CONNCOMP in terms of both robust accuracy (by 1.3 points) and standard accuracy (by 2.8 points).

[3] https://github.com/huggingface/pytorch-transformers

Accuracy	System	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Avg
Standard	**Baselines**							
	BERT	93.8	87.7	91.3	84.6	88.6	71.1	86.2
	Data Aug. + BERT	92.2	84.3	88.7	83.0	87.4	63.5	83.1
	Typo Corr. + BERT	89.6	80.9	87.6	75.9	80.5	54.9	78.2
	RobEn							
	Con. Comp. + BERT	80.6	79.9	84.2	65.7	73.3	52.7	72.7
	Agg. Clust. + BERT	83.1	83.8	85.0	69.1	76.6	59.2	76.1
Attack	**Baselines**							
	BERT	8.7	10.0	17.4	0.7	0.7	1.8	6.6
	Data Aug. + BERT	17.1	1.0	27.6	15.4	10.7	1.4	12.2
	Typo Corr. + BERT	53.2	30.1	52.0	23.0	32.3	21.3	35.3
	RobEn							
	Con. Comp. + BERT	80.3	79.4	82.7	62.6	71.5	47.3	70.6
	Agg. Clust. + BERT	**82.1**	**82.8**	**83.2**	**65.3**	**74.5**	**52.7**	**73.4**
Robust	**RobEn**							
	Con. Comp. + BERT	80.1	79.4	**82.2**	61.4	70.5	46.6	70.0
	Agg. Clust. + BERT	**80.7**	**80.9**	81.4	**62.8**	**71.9**	**49.8**	**71.3**

Table 1: Standard, attack, and robust accuracy on six GLUE tasks against ED1 perturbations. For baseline models we only compute attack accuracy, an upper bound on robust accuracy, since robust accuracy cannot be tractably computed. Using RobEn, we get robustness guarantees by computing robust accuracy, which we find outperforms a the typo corrector in (Pruthi et al., 2019) by *at least* 36 points.

Standard accuracy. Like defenses against adversarial examples in other domains, using RobEn decreases standard accuracy (Madry et al., 2017; Zhang et al., 2019; Jia et al., 2019). Our agglomerative cluster encodings's standard accuracy is 10.1 points lower then that of normally trained BERT. However, to the best of our knowledge, our standard accuracy is state-of-the-art for approaches that guarantee robustness. We attribute this improvement to RobEn's compatibility with any model.

Comparison to smaller attack surfaces. We note that RobEn also outperform existing methods on their original, smaller attack surfaces. On SST-2, Pruthi et al. (2019) achieves an accuracy of 75.0% defending against a *single* ED1 typo, which is 5.7 points lower than AGGCLUST's robust accuracy against perturbations of all tokens: a superset of the original perturbation set. We discuss constrained adversaries further in Appendix A.5. AGGCLUST also outperforms certified training: Huang et al. (2019), which offers robustness guarantees to three character substitution typos (but not insertions or deletions), achieves a robust accuracy of 74.9% on SST-2. Certified training requires strong assumptions on model architecture; even the robust accuracy of AGGCLUST outperforms the standard accuracy of the CNN used in Huang et al. (2019).

5.5 Reusable encodings

Each instance of RobEn achieves consistently high stability across our tasks, despite reusing a single function. Figure 4 plots the distribution of $|B_\alpha(x)|$, across test examples in SST-2 and RTE, where $B_\alpha(x)$ is the set of encodings that are mapped to by some perturbation of x. Over AGGCLUST encodings, $|B_\alpha(x)| = 1$ for 25% of examples in RTE and 66% in SST-2, with the other four datasets falling between these extremes (see Appendix A.6). As expected, these numbers are even higher for the connected component encodings. Note that when $|B_\alpha(x)| = 1$, every perturbation of x maps to the same encoding. When $|B_\alpha(x)|$ is small, robust accuracy can be computed quickly.

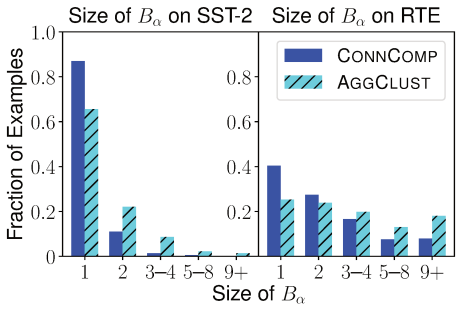


Figure 4: Histogram of $|B_\alpha(x)|$ for SST-2 and RTE. SST-2 has the highest percentage of inputs x where $|B_\alpha(x)| = 1$, while RTE has the least. On both datasets, $|B_\alpha(x)| < 9$ for most x, and $|B_\alpha(x)| = 1$ on a plurality of inputs.

5.6 Agglomerative Clustering Tradeoff

In Figure 5, we plot standard and robust accuracy on SST-2 for AGGCLUST encodings, using differ-

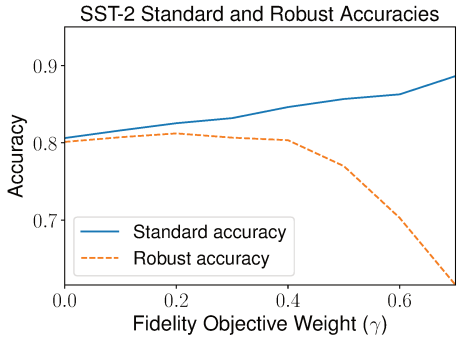


Figure 5: Standard and robust accuracies on SST-2 with AGGCLUST using different values of γ. While the gap between standard and robust accuracy increases monotonically, robust accuracy increases before decreasing.

ent values of γ. Recall that $\gamma = 0$ maximizes stability (CONNCOMP), and $\gamma = 1$ maximizes fidelity. At $\gamma = 0$, the gap between standard and robust accuracy, due to out-of-vocabulary tokens, is negligible. As γ increases, both standard accuracy and the gap between standard and robust accuracy increase. As a result, robust accuracy first increases, then decreases.

5.7 Internal permutation attacks

RobEn can also be used to defend against the internal perturbations described in Section 5.1. For normally trained BERT, a heuristic beam search attack using internal permutations reduces average accuracy from 86.2% to 15.7% across our six tasks. Using CONNCOMP with the internal permutation attack surface, we achieve robust accuracy of 81.4%. See Appendix A.7 for further details.

6 Discussion

Additional related work. In this work, we introduce RobEn, a framework to construct systems that are robust to adversarial perturbations. We then use RobEn to achieve state-of-the-art robust accuracy when defending against adversarial typos. Besides typos, other perturbations can also be applied to text. Prior attacks consider semantic operations, such as replacing a word with a synonym (Alzantot et al., 2018; Ribeiro et al., 2018). Our framework extends easily to these perturbations. Other attack surfaces involving insertion of sentences (Jia and Liang, 2017) or syntactic rearrangements (Iyyer et al., 2018) are harder to pair with RobEn, and are interesting directions for future work.

Other defenses are based on various forms of preprocessing. Gong et al. (2019) apply a spell-corrector to correct typos chosen to create ambiguity as to the original word, but these typos are not adversarially chosen to fool a model. Edizel et al. (2019) attempt to learn typo-resistant word embeddings, but focus on common typos, rather than worst-case typos. In computer vision, Chen et al. (2019) discretizes pixels to compute exact robust accuracy on MNIST, but their approach generalizes poorly to other tasks like CIFAR-10. Garg et al. (2018) generate functions that map to robust features, while enforcing variation in outputs.

Incorporating context. Our token-level robust encodings lead to strong performance, despite ignoring useful contextual information. Using context is not fundamentally at odds with the idea of robust encodings, and making contextual encodings stable is an interesting technical challenge and a promising direction for future work.

In principle, an oracle that maps every word with a typo to the correct unperturbed word seems to have higher fidelity than our encodings, without compromising stability. However, existing typo correctors are far from perfect, and a choosing an incorrect unperturbed word from a perturbed input leads to errors in predictions of the downstream model. This mandates an intractable search over all perturbations to compute the robust accuracy.

Task-agnosticity. Many recent advances in NLP have been fueled by the rise of task-agnostic representations, such as BERT, that facilitate the creation of accurate models for many tasks. Robustness to typos should similarly be achieved in a task-agnostic manner, as it is a shared goal across many NLP tasks. Our work shows that even simple robust encodings generalize across tasks and are more robust than existing defenses. We hope our work inspires new task-agnostic robust encodings that lead to more robust and more accurate models.

Acknowledgments

This work was supported by NSF Award Grant no. 1805310 and the DARPA ASED program under FA8650-18-2-7882. A.R. is supported by a Google PhD Fellowship and the Open Philanthropy Project AI Fellowship. We thank Pang Wei Koh, Reid Pryzant, Ethan Chi, Daniel Kang, and the anonymous reviewers for their helpful comments.

Reproducibility

All code, data, and experiments are available on CodaLab at https://bit.ly/2VSZI2e.

References

M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. Srivastava, and K. Chang. 2018. Generating natural language adversarial examples. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Athalye, N. Carlini, and D. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

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

J. Chen, X. Wu, V. Rastogi, Y. Liang, and S. Jha. 2019. Towards understanding limitations of pixel discretization against adversarial attacks. In *IEEE European Symposium on Security and Privacy (EuroS&P)*.

J. M. Cohen, E. Rosenfeld, and J. Z. Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*.

M. Davies. 2008. The corpus of contemporary american english (coca): 560 million words, 1990-present. https://www.english-corpora.org/faq.asp.

M. Davis. 2003. Psycholinguistic evidence on scrambled letters in reading. https://www.mrc-cbu.cam.ac.uk/.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Association for Computational Linguistics (ACL)*, pages 4171–4186.

W. B. Dolan and C. Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *International Workshop on Paraphrasing (IWP)*.

J. Ebrahimi, D. Lowd, and D. Dou. 2018a. On adversarial examples for character-level neural machine translation. In *International Conference on Computational Linguistics (COLING)*.

J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. 2018b. Hotflip: White-box adversarial examples for text classification. In *Association for Computational Linguistics (ACL)*.

B. Edizel, A. Piktus, P. Bojanowski, R. Ferreira, E. Grave, and F. Silvestri. 2019. Misspelling oblivious word embeddings. In *North American Association for Computational Linguistics (NAACL)*.

S. Garg, V. Sharan, B. H. Zhang, and G. Valiant. 2018. A spectral view of adversarially robust features. In *Advances in Neural Information Processing Systems (NeurIPS)*.

H. Gong, Y. Li, S. Bhat, and P. Viswanath. 2019. Context-sensitive malicious spelling error correction. In *World Wide Web (WWW)*, pages 2771–2777.

S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, T. Mann, and P. Kohli. 2018. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*.

H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran. 2017. Deceiving Google's Perspective API built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

P. Huang, R. Stanforth, J. Welbl, C. Dyer, D. Yogatama, S. Gowal, K. Dvijotham, and P. Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics (NAACL)*.

R. Jia and P. Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. Jia, A. Raghunathan, K. Gksel, and P. Liang. 2019. Certified robustness to adversarial word substitutions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

V. Kocijan, A. Cretu, O. Camburu, Y. Yordanov, and T. Lukasiewicz. 2019. A surprisingly robust trick for the Winograd schema challenge. In *Association for Computational Linguistics (ACL)*.

H. Lee and A. Y. Ng. 2005. Spam deobfuscation using a hidden Markov model. In *Conference on Email and Anti-Spam (CEAS)*.

N. F. Liu, R. Schwartz, and N. A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *North American Association for Computational Linguistics (NAACL)*.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2017. Towards deep learning models resistant to adversarial attacks (published at ICLR 2018). *arXiv*.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*.

J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

D. Pruthi, B. Dhingra, and Z. C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Association for Computational Linguistics (ACL)*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

G. E. Rawlinson. 1976. The significance of letter position in word recognition. Ph.D. thesis, University of Nottingham.

M. T. Ribeiro, S. Singh, and C. Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Association for Computational Linguistics (ACL)*.

K. Sakaguchi, K. Duh, M. Post, and B. V. Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Z. Shi, H. Zhang, K. Chang, M. Huang, and C. Hsieh. 2020. Robustness verification for transformers. In *International Conference on Learning Representations (ICLR)*.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*.

A. Williams, N. Nangia, and S. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Association for Computational Linguistics (ACL)*, pages 1112–1122.

H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*.

A Appendix

A.1 Aggloemrative clustering

Recall that any π_V induces a clustering of V, where each cluster contains a set of words mapped by π_V to the same encoded token. We use an agglomerative clustering algorithm to approximately minimize Φ. We initialize π_V by setting $\pi_V(w) = w$ for each $w \in V$, which corresponds to placing each word in its own cluster. We then examine each pair of clusters C_i, C_j such that there exists an edge between a node in C_i and a node in C_j, in the graph from Section 4.2. For each such pair, we compute the value of Φ if C_i and C_j were replaced by $C_i \cup C_j$. If no merge operation causes Φ to decrease, we return the current π_V. Otherwise, we merge the pair that leads to the greatest reduction in Φ, and repeat. To merge two clusters C_i and C_j, we first compute a new encoded token r as the $w \in C_i \cup C_j$ with largest $\rho(w)$. We then set $\pi_V(w) = r$ for all $w \in C_i \cup C_j$. Our algorithm thus works as follows

Algorithm 1 Objective-minimizing agglomerative clustering

$C \leftarrow V$
for i in range($|V|$) **do**
  $C_{\text{next}} \leftarrow$ Get Best Combination(C)
  if $C = C_{\text{next}}$ **then**
    return C
  end if
  $C \leftarrow C_{\text{next}}$
end for
return C

Now, we simply have to define the procedure we use to get the best combination.

Algorithm 2 Get Best Combination(C)

$C_{\text{opt}} \leftarrow C$
$\Phi_{\text{opt}} \leftarrow \Phi(C)$
for $(C_i, C_j) \in$ Adjacent Pairs(C) **do**
  $C_{\text{comb}} \leftarrow C_i \cup C_j$
  $C_{\text{new}} \leftarrow C \cup C_{\text{comb}} \setminus \{C_i, C_j\}$ {New clusters}
  $\Phi_{\text{new}} \leftarrow \Phi(C_{\text{new}})$
  if $\Phi_{new} < \Phi_{opt}$ **then**
    $\Phi_{opt} \leftarrow \Phi_{new}$
    $C_{\text{opt}} \leftarrow C_{\text{new}}$
  end if
end for
return C_{opt}

Recall our graph $G = (G, E)$ used to define the connected component clusters. We say two clusters C_i and C_j are *adjacent*, and thus returned by Adjacent Pairs, if there exists a $v_i \in C_i$ and a $v_j \in C_j$ such that $(v_i, v_j) \in G_E$. The runtime of our algorithm is $O(N^2E)$ since at each of a possible N total iterations, we compute the objective for one of at most E pairs of clusters. Computation of the objective can be reframed as computing the difference between Φ and Φ_{new}, where the latter is

computed using new clusters, which can be done in $O(N)$ time.

A.2 Attacks

We use two heuristic attacks to compute an upper bound for robust accuracy: one for ED1 perturbations and one for internal permutations. Each heuristic attack is a beam search, with beam width 5. However, because $|B(x_i)|$ is very large for many tokens x_i, even the beam search is intractable. Instead, we run a beam search where the allowable perturbations are $B'(x_i) \subseteq B(x_i)$, where $|B'(x_i)| << B(x_i)$ for sufficiently long x_i. For our ED1 attack, we define $B'(x_i)$ to be four randomly sampled perturbations from $B(x_i)$ when the length of x_i is less than five, and all deletions when x_i is greater than five. Thus, the number of perturbations of each word is bounded above by $\min\{4, \text{len}(x_i) - 2\}$. For our internal permutations, $B'(x_i)$ is obtained by sampling five permutations at random.

A.3 Datasets

We use six out of the nine tasks from GLUE: SST, MRPC, QQP, MNLI, QNLI, and RTE, all of which are classification tasks measured by accuracy. The Stanford Sentiment Treebank (SST-2) (Socher et al., 2013) contains movie reviews that are classified as positive and negative. The Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) and the Quora Question Pairs dataset[4] contain pairs of input which are classified as semantically equivalent or not; QQP contains question pairs from Quora, while MRPC contains pairs from online news sources. MNLI, and RTE are entailment tasks, where the goal is to predict whether or not a premise sentence entails a hypothesis (Williams et al., 2018). MNLI gathers premise sentences from ten different sources, while RTE gathers premises from entailment challenges. QNLI gives pairs of sentences and questions extracted from the Stanford Question Answering Dataset (Rajpurkar et al., 2016), and the task is to predict whether or not the answer to the question is in the sentence.

We use the GLUE splits for the six datasets and evaluate on test labels when available (SST-2, MRPC), and otherwise the publicly released development labels. We tune hyperparameters by training on 80% of the original train set and using the remaining 20% as a validation set. We then retrain using the chosen hyperparameters on the full training set.

A.4 Experimental details

For our methods using transformers, we start with the pretrained uncased BERT (Devlin et al., 2019), using the same hyperparameters as the pytorch-transformers repo.[5]. In particular, we use the base uncased version of BERT. We use a batch size of 8, and learning rate $2e{-}5$. For examples where $|B_\alpha(x)| > 10000$, we assume the prediction is not robust to make computation tractible. Each typo corrector uses the defaults for training from[6]; it is trained on a specific task using perturbations of the training data as input and the true sentence (up to OOV) as output. The vocabulary size of the typo correctors is 10000 including the unknown token, as in (Pruthi et al., 2019). The typo corrector is chosen based on word-error rate on the validation set.

A.5 Constrained adversaries

Using RobEn, since we can tractably compute robust accuracy, it is easy to additionally consider adversaries that cannot perturb every input token. We may assume that an attacker has a budget of $b \leq L$ words that they may perturb as in (Pruthi et al., 2019). Exiting methods for certification (Jia et al., 2019; Huang et al., 2019) require attack to be factorized over tokens, and cannot give tighter guarantees in the budget-constrained case compared to the unconstrained setting explored in previous sections. However, our method lets us easily compute robust accuracy exactly in this situation: we just enumerate the possible perturbations that satisfy the budget constraint, and query the model.

Figure 6 plots average robust accuracy across the six tasks using AGGCLUST as a function of b. Note that $b = 0$ is simply standard accuracy. Interestingly, for each dataset there is an attack only perturbing 4 tokens with attack accuracy equal to robust accuracy.

A.6 Number of representations

We include here histograms for the datasets we did not cover in the main body. The histograms for

[4] data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[5] https://github.com/huggingface/pytorch-transformers

[6] https://github.com/danishpruthi/Adversarial-Misspellings

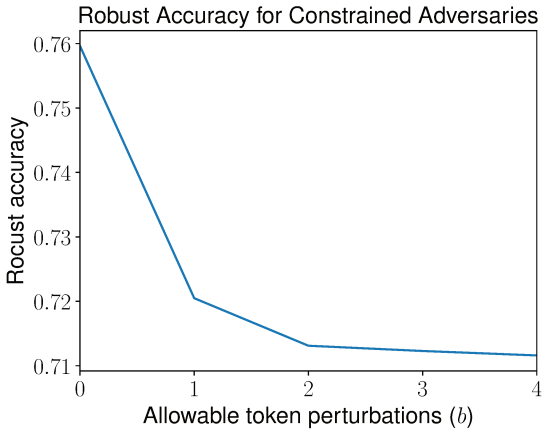
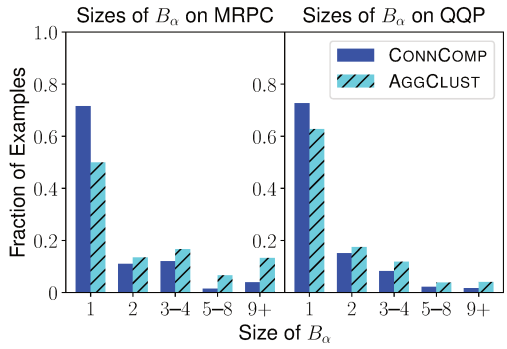
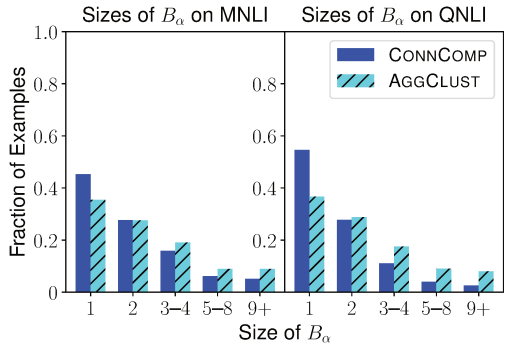


Figure 6: Robust accuracy averaged across all tasks based on different adversarial budgets b. $b = 0$ corresponds to clean performance, and robust performance is reached at $b = 4$



(a) Size of B_α over MRPC and QQP



(b) Size of B_α over MNLI and QNLI

Figure 7: Histograms showing sizes of B_α for MRPC, QQP, MNLI, and QNLI.

MRPC and QQP are shown in Figure 7(a), while the histograms for MNLI and QNLI are shown in Figure 7(b). The fraction of x such that $|B_\alpha(x)| = 1$ for each dataset and each set of encodings is provided in Table 2.

A.7 Internal Permutation Results

We consider the *internal permutation* attack surface, where interior characters in a word can be permuted, assuming the first and last characters are fixed. For example, *"perturbation"* can be permuted to *"peabreuottin"* but not *"repturbation"*. Normally, context helps humans resolve these typos. Interestingly, for internal permutations it is impossible for an adversary to change the cluster assignment of both in-vocab and out of vocab tokens since a cluster can be uniquely represented by the first character, a sorted version of the internal characters, and the last character. Therefore, using ConnComp encodings, robust, attack, and standard accuracy are all equal. We use the attack described in A.2 to attack the clean model. The results are in Table 3.

Encodings	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Avg
Con. Comp.	86.9	71.6	72.7	45.3	54.6	40.4	61.9
Agg. Clust.	65.6	50.0	62.7	35.4	36.6	25.2	45.9

Table 2: Percentage of test examples with $|B_\alpha(x)| = 1$ for each dataset.

Accuracy	System	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Avg
Standard	BERT	93.8	87.7	91.2	84.3	88.9	71.1	86.2
	Con. Comp. + BERT	93.2	87.7	86.9	75.9	83.4	61.4	81.4
Attack	BERT	28.1	15.9	33.0	4.9	6.2	5.8	15.7
	Con. Comp. + BERT	93.2	87.7	86.9	75.9	83.4	61.4	81.4
Robust	Con. Comp. + BERT	93.2	87.7	86.9	75.9	83.4	61.4	81.4

Table 3: Results from internal permutation attacks. Internal permutation attacks bring the average performance for BERT across the six listed tasks from 86.2 to 15.7. Our CONNCOMP encodings, generated using the internal permutation attack surface, achieve a robust accuracy of 81.4, which is only 4.8 points below standard accuracy.