TASK-AGNOSTIC ROBUST ENCODINGS FOR COMBATING ADVERSARIAL TYPOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite achieving excellent benchmark performance, state-of-the-art NLP models can still be easily fooled by adversarial perturbations such as typos. Previous heuristic defenses cannot guard against the exponentially large number of possible perturbations, and previous certified defenses only work with limited model sizes and simple architectures. In this paper, we construct *task-agnostic robust encodings* (TARE): sentence representations that improve the robustness of any model for multiple downstream tasks at once, and enable efficient exact computation of robust accuracy (accuracy on worst-case perturbations) for a fixed family of perturbations. The core idea behind TARE is to map sentences through a discrete bottleneck before feeding them to a downstream model. To create robust encodings, we must optimize for two competing goals: the encoding of a sentence must retain enough information about the sentence, but should also map all perturbations of the sentence to the same encoding to ensure invariance to perturbations. Averaged across six tasks from GLUE, a standard suite of NLP tasks, the same encoding leads to robust accuracy of 71.2% when defending against a large family of typos, while a strong baseline that uses a typo corrector achieves only 38.5% accuracy, and training on random typos achieves only 9.9% accuracy.

1 INTRODUCTION

Recent work has revealed the brittleness of NLP systems to perturbations (Belinkov & Bisk, 2017; Ebrahimi et al., 2017; Ribeiro et al., 2018; Alzantot et al., 2018). In particular, adversarial typos have fooled systems for hate speech detection (Hosseini et al., 2017), machine translation (Ebrahimi et al., 2018), and spam filtering (Lee & Ng, 2005), among others. In order to build more robust NLP systems, we typically define a set of allowable perturbations, such as a list of plausible typos for each word (Jia et al., 2019; Huang et al., 2019). Then, we attempt to train models that have high *robust accuracy*, the worst-case accuracy across all perturbations of each example. An ideal defense against perturbations would produce a single representation that works across different tasks and models, while giving guarantees on the ability of *any* adversary to hurt model performance.

Previous work does not address all of these goals at once. Some prior work tries to recover the original sentence given a perturbed input (Pruthi et al., 2019; Gong et al., 2019). Pruthi et al. (2019) apply a typo corrector to inputs before passing them to a model, but do not achieve high robust accuracy on large families of typos. Other prior work confers robustness by changing the training procedure. Adversarial training, which adds adversarial perturbations to the training set, can be applied to any model. However, it only provides limited robustness, due to heuristics used to compute adversarial attacks for text inputs (Ebrahimi et al., 2017). Certifiably robust training with interval bound propagation (Huang et al., 2019) confers guaranteed robustness, but can only currently be applied to limited model architectures of small size. Neither of these training methods produces a single defense that works across multiple tasks.

Our proposed method simultaneously overcomes the task-specificity or model-specificity of prior approaches, while providing robustness against worst-case perturbations. We construct *task-agnostic robust encodings* (TAREs): discrete representations of sentences that can be used across many tasks to improve robustness of any downstream model. TAREs are model-agnostic, reusable across tasks, and confer guaranteed robustness via *standard training* of downstream models, avoiding the computational overhead typically associated with adversarial training. By introducing a discrete

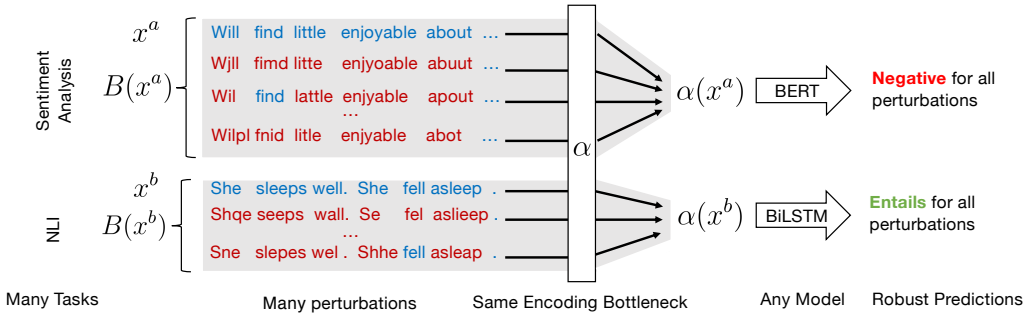


Figure 1: Task-agnostic robust encodings confer robustness to adversarial perturbations. Clean inputs x^a and x^b (blue) can be perturbed to many different sentences, denoted by the sets $B(x^a)$ and $B(x^b)$. We optimize an encoding function α such that for many sentences x, every perturbation of x maps to $\alpha(x)$, providing robustness to perturbations. The same encoding function can be used for multiple tasks, and any model using our encodings as input makes robust predictions.

bottleneck, our robust encodings enable tractable computation of robust accuracy, even when the set of allowed perturbations is combinatorially large. See Figure 1 for an overview of our approach.

The key idea of TARE is to have an encoding function $\alpha(x)$ that maps each token of an input sentence x to an encoding such that each token has the same encoding as all of its possible perturbations. This property, which we term *stability*, ensures that any model that makes predictions using only the encoding will give the same prediction on all perturbed versions of an input. We also need to ensure that the encodings contain enough information about the input to do the task, which we term *fidelity*. We choose α by optimizing a weighted combination of these two objectives. We cast this search over possible encoding functions as a clustering problem over tokens, where each cluster contains tokens that are mapped to the same encoding. We perform this clustering in an unsupervised, task-agnostic fashion, and use the same encoding function for various downstream tasks.

In our experiments, we consider attacks that are allowed to add an edit distance one typo to each word in an input sentence, without changing the first and last character of each word, following prior work (Pruthi et al., 2019). Using TARE, we significantly improve robust accuracy across six classification tasks from the GLUE benchmark (Wang et al., 2019). Our best system, which combines TARE with a BERT classifier (Devlin et al., 2018), achieves robust accuracy of 71.2% on average across the six tasks. In contrast, a state-of-the-art defense that combines BERT with a typo corrector (Pruthi et al., 2019) gets 38.5% accuracy, and a standard defense that trains BERT on random perturbations gets only 9.9% accuracy in the presence of heuristically generated adversarial typos.

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 $x_1, \ldots, x_L$ where each $x_i \in \mathcal{T}$, the set of all possible tokens, which includes valid perturbations (e.g., typos) of vocabulary words. 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 model. We consider an attack model in which the attacker 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 (e.g., tokens that are edit distance one from x_i). We define $B(x)$ as the set of all valid perturbations of sentence x. Each token can be perturbed independently, so

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

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

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

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

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:

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

For many models, computing robust accuracy is intractable. It is customary to instead compute accuracy against a heuristic attack a that maps clean sentences x to perturbed sentences $a(x) \in B(x)$.

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

Typically, we compute $a(x)$ via a heuristic search over $B(x)$ for a sentence that f misclassifies. Note that $\text{acc}_{\text{attack}} \geq \text{acc}_{\text{robust}}$ because there could be perturbations that the model misclassifies but are missed by the heuristic search. In general, a high $\text{acc}_{\text{attack}}$ does not guarantee that the model has high $\text{acc}_{\text{robust}}$, which is the actual quantity of interest (Athalye et al., 2018).

3 TASK-AGNOSTIC ROBUST ENCODINGS

In this work, we construct *task-agnostic robust encodings* (TAREs). Informally, TARE enables arbitrary model architectures to achieve high robust accuracy on many standard tasks. We define an *encoding function* to be a function $\alpha : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{X}$ is the set of sentences and $\mathcal{Z} \subseteq \mathcal{X}$. We refer to $\alpha(x)$ as the encoded sentence, or encoding, corresponding to x. A downstream model $f_{\text{down}} : \mathcal{Z} \to \mathcal{Y}$ makes predictions using the encoded sentence $\alpha(x)$. We denote $f_{\text{down}}(\alpha(x))$ by f_{TARE}. Because $\mathcal{Z} \subseteq \mathcal{X}$, f_{down} can be any model that takes sentences as inputs.

3.1 ROBUSTNESS OF ENCODINGS

We are interested in obtaining models that are robust, i.e., accurate on all valid perturbations of an input. For some model f_{TARE} that uses our robust encodings, we can guarantee high $\text{acc}_{\text{robust}}(f_{\text{TARE}})$ if $\text{acc}_{\text{std}}(f_{\text{TARE}})$ is high and f_{TARE} usually makes the same prediction on all $\tilde{x} \in B(x)$:

$$\text{acc}_{\text{robust}}(f_{\text{TARE}}) \geq \text{acc}_{\text{std}}(f_{\text{TARE}}) - \mathbb{E}_{(x,y)\sim p_{\text{task}}} \max_{\tilde{x}\in B(x)} \mathbf{1}\left[f_{\text{TARE}}(\tilde{x}) \neq f_{\text{TARE}}(x)\right]. \tag{5}$$

Based on Equation 5, we define two desiderata of α that enable f_{TARE} to achieve high robust accuracy. First, high *fidelity* of an encoding ensures that it is possible for f_{TARE} to get high standard accuracy. Second, high *stability* of an encoding upper bounds the second term in (5), ensuring that *any* model that has high standard accuracy also has high robust accuracy.

Fidelity. First, we must ensure that $\alpha(x)$ contains enough information to perform the task. We say that α has high *fidelity* if there exists some $f^\star_{\text{down}}$ for which $f^\star_{\text{TARE}}(x) \stackrel{\text{def}}{=} f^\star_{\text{down}}(\alpha(x))$ has high standard accuracy. Fidelity is maximized by having $\alpha(x) = x$, since this preserves all information about x.

Stability. We now focus on the second term of Equation 5. Consider the simple case where $\alpha(x)$ is constant for all x, i.e. all sentences share the same encoding. Here, trivially $f_{\text{TARE}}(x) = f_{\text{TARE}}(\tilde{x}) \,\forall\, \tilde{x} \in B(x)$ and the second term is 0. For the more general case, we bound the second term of (5) by the *stability* of an encoding, which we define as the probability over x drawn from p_{task} that all perturbations of x are encoded to the same $\alpha(x)$. Formally, we have

$$B_\alpha(x) = \{\alpha(\tilde{x}) \mid \tilde{x} \in B(x)\}, \tag{6}$$

$$\text{Stability}(\alpha) = \mathbb{E}_{x\sim p_{\text{task}}} \mathbf{1}[|B_\alpha(x)| = 1] \tag{7}$$

$$\mathbb{E}_{(x,y)\sim p_{\text{task}}} \max_{\tilde{x}\in B(x)} \mathbf{1}\left[f_{\text{TARE}}(\tilde{x}) \neq f_{\text{TARE}}(x)\right] \leq 1 - \text{Stability}(\alpha) \tag{8}$$

When $|B_\alpha(x)| = 1$, f_{TARE} outputs the same prediction for all perturbations of x. As $|B_\alpha(x)|$ increases, perturbations get mapped to different possible encodings, which may be classified differently by f_{down}. However, even when $|B_\alpha(x)| > 1$, keeping $|B_\alpha(x)|$ small enables efficient computation of $\text{acc}_{\text{robust}}(f_{\text{TARE}})$ by exhaustively enumerating $B_\alpha(x)$. Without using α to collapse $B(x)$ to a small number of encoded sentences, computing robust accuracy is intractable for arbitrary models.

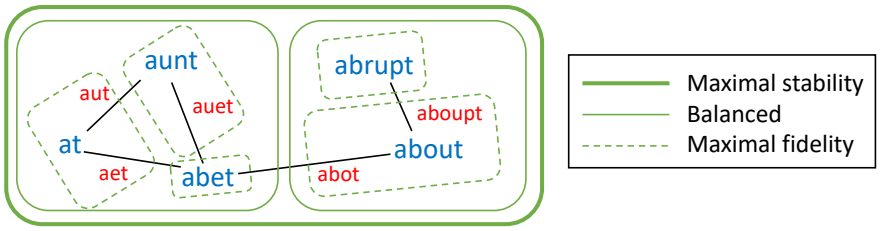


Figure 2: Visualization of three different encodings. Vocabulary words (large font, blue) share an edge if they share a common perturbation (small font, red). We show three different encodings: the bold line corresponds to the encoding with maximal stability (connected component encoding), the dotted lines correspond to maximizing fidelity, and the thin solid lines balance stability and fidelity (agglomerative clusters).

Tradeoff. Note that fidelity and stability are often opposing goals. Fidelity tries to map tokens to themselves while stability tries to map tokens and their perturbations together.

3.2 Task-agnostic robust encodings

Finally, a robust encoding is *task-agnostic* if it can be constructed without knowledge of any downstream task, but is still a robust encoding for many downstream tasks. This is challenging, as stability and fidelity are both defined with respect to a task. Stability only depends on the distribution of inputs x, and hence can be approximated using a large, unlabeled text corpus. Fidelity requires labels, and hence is more difficult to approximate. Nonetheless, in the next section, we show how to construct α in an unsupervised, task-agnostic fashion such that for many natural tasks, we can train a model $\hat{f}_{\text{TARE}}$ that achieves high standard accuracy.

4 Task-agnostic construction of robust token-level encodings

Now, we describe how to construct robust encodings in a task-agnostic fashion. We focus on token-level encodings, where the encoding of a sentence is the concatenation of the encoding of individual tokens. Formally, we extend α to map tokens $w \in \mathcal{T}$ to some set $\mathcal{Z}_{\text{Tok}} \subseteq \mathcal{T}$ of *encoded tokens*. The encoding $\alpha(x)$ for input sentence x can then be written as

$$\alpha(x) = [\alpha(x_1), \alpha(x_2), \ldots, \alpha(x_L)]. \tag{9}$$

We now focus on how to choose $\alpha(w)$ for all $w \in \mathcal{T}$, in a task-agnostic fashion.

4.1 Robustness of token-level encodings

How can we choose a robust token-level encoding α, balancing the competing goals of stability and fidelity? As a running example, consider the five words (large font, blue) in Figure 2, along with typos of those words (small font, red). We illustrate three different choices of α; for each, we draw a box around tokens that α maps to the same encoded token. α could map all five words to the same encoded token (thick box), map each word to a different encoded token (dashed boxes), or do something in between (thin solid boxes).

First, consider trying to maximize stability. Note that for a sentence x, $|B_\alpha(x)| = 1$ if and only if $|B_\alpha(x_i)| = 1$ for all words x_i in x, so maximizing stability requires $|B_\alpha(w)| = 1$ for all words w that might appear in x. Consider two words w_1 and w_2 (e.g., *"aunt"* and *"abet"* for which there exists a token $\tilde{w} \in \mathcal{T}$ that could be a perturbation of both (e.g., *"auet"* is edit distance one from both words). We draw edges between all such word pairs in Figure 2, and place an example of a typo in $B_\alpha(w_1) \cap B_\alpha(w_2)$ next to each edge. For any such w_1 and w_2, maximizing stability requires that $\alpha(w_1) = \alpha(w_2)$, since if they were not equal, α must either map w_1 and $\tilde{w}$ to different encoded tokens (hence, $|B_\alpha(w_1)| > 1$), or map w_2 and $\tilde{w}$ to different encoded tokens (hence, $|B_\alpha(w_2)| > 1$).

In Figure 2, we would therefore have to map all five words to the same encoded token (thick box), as they are in the same connected component in this graph.

However, this mapping clearly has low fidelity, since these five words are all identically encoded. Fidelity is maximized when all information about x is retained, so each word is mapped to a different encoded token (dashed green boxes). Unfortunately, this choice of α has low stability. For the words *"art"*, *"abet"*, and *"abrupt"*, $|B_\alpha(w)| > 1$, since these words and their typos map to at least two different encoded tokens (e.g., *"aut"*, a typo of *"aunt"*, maps to the encoding of *"at"*).

The encoding represented by the thin solid boxes in Figure 2 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 *"abet"* has $|B_\alpha(w)| > 1$. Since the word *"abet"* occurs infrequently in normal English text, the fact that $|B_\alpha(w)| > 1$ for that word will only affect $|B_\alpha(x)|$ for a very small fraction of $x \sim p_{\text{task}}$, for most natural task distributions p_{task}.

With this intuition, we now define two ways to construct α. Our first method will focus on maximizing stability, at the cost of fidelity. Our second method allows us to trade off stability for fidelity, improving both standard and robust accuracy of downstream models. Both methods will construct α in a task-agnostic fashion: we will not reference any task distribution p_{task}, but instead only assume access to a large, unlabeled corpus of English text. Implicitly, we assume that the distribution of sentences $x \sim p_{\text{task}}$ will not differ too much from the distribution of text in our corpus.

4.2 Baseline: Connected component encodings

Our first method, connected component encodings, focuses on maximizing stability. Ideally, we would have $|B_\alpha(w)| = 1$ for all $w \in \mathcal{T}$, but this is very restrictive, and would lead to very low fidelity. Note that in order to have small $B_\alpha(x)$ for most $x \sim p_{\text{task}}$, it suffices to ensure $|B_\alpha(w)| = 1$ for all tokens that are actually likely to appear in x. Therefore, we define a set of frequent words $V \subseteq \mathcal{T}$, and then make sure that $|B_\alpha(w)| = 1$ for all $w \in V$.

We first create a vocabulary $V = \{w_1, \ldots, w_N\} \subseteq \mathcal{T}$ by taking the N most frequent words in a large, unlabeled text corpus. For a word $w \in V$, let $\rho(w)$ denote the fraction of times w occurs in this corpus. Next, we construct the graph $G = (V, E)$ shown in Figure 2, whose vertex set is our vocabulary V, and its edge set E contains all pairs of vertices that share a common perturbation:

$$E = \{(w_i, w_j) \mid w_i, w_j \in V, B(w_i) \cap B(w_j) \neq \emptyset\} \tag{10}$$

Let K be the number of connected components of G, and let $C_1, \ldots, C_K$ denote these connected components. For each C_j, we define its *representative* r_j to be the word $w \in C_j$ with largest $\rho(w)$, and set $\alpha(w) = r_j$ for all $w \in C_j$. Using connected components guarantees that every pair of words $w_i, w_j \in V$ that shares a perturbation will be encoded identically. Mapping to representative words helps us leverage the inductive biases of pre-trained models like BERT (Devlin et al., 2018).

Extending connected component encodings to out-of-vocabulary tokens. We have defined α for all $w \in V$; now we must define it for other $\tilde{w} \in \mathcal{T} \setminus V$, such as typos (e.g., *"aboupt"*). There are two cases. First, suppose there exists $w \in V$ such that $\tilde{w} \in B(w)$. Then, we define $\alpha(\tilde{w}) = \alpha(w)$. Note that this is well-defined, as we constructed the C_j's to ensure if multiple such w exist, they map to the same encoded token. This choice guarantees $|B_\alpha(w)| = 1$ for all $w \in V$. It may also be the case that $\tilde{w} \notin B(w) \, \forall \, w \in V$. In this case, we set $\alpha(\tilde{w}) = \text{OOV}$, a special out-of-vocabulary token.

Connected component encodings and full sentences. We now return to the question of stability at the sentence-level—for which sentences x does $|B_\alpha(x)| = 1$? Connected component encodings ensure that $|B_\alpha(x)| = 1$ as long as $x_i \in V$ for all words x_i in x. However, some task distributions p_{task} may include rare words that are not in our vocabulary V. When x contains a word $w' \notin V$, it is possible that $|B_\alpha(w')| > 1$, and hence $|B_\alpha(x)| > 1$. Note that $|B_\alpha(w')|$ will still be 1 if all perturbations of w' also map to OOV.

4.3 Agglomerative cluster encodings

Connected component encodings focus on stability at the cost of fidelity. When G has large connected components, the resulting α may not be very useful for downstream tasks, since too many tokens

share the same encoding. In particular, two frequent words that do not share a common perturbation will still have the same encoding if there is a path connecting them. In Figure 2, *"at"* and *"about"* are in the same connected component, even though no token is a plausible typo of both words. Since both words are very frequent in general, mapping them to the same encoding creates ambiguity in a large proportion of frequent sentences. Therefore, encoding *"at"* and *"about"* differently would significantly improve fidelity. The thin solid boxes in Figure 2 accomplish this goal, at a small cost to stability: recall that only the infrequent word *"abet"* has $|B_\alpha(w)| > 1$, which is unlikely to affect $|B_\alpha(x)|$ for most sentences x.

To take advantage of cases like this, we formally define two task-agnostic objectives on α: one that encourages stability, and one that encourages fidelity. Recall that stability and fidelity are dependent on a particular task distribution p_{task}; our objectives are task-agnostic approximations of stability and fidelity that empirically generalize to many natural tasks. We then use an agglomerative clustering algorithm to approximately optimize a weighted sum of these objectives, where we view a cluster $C_j \subseteq V$ as a set of vocabulary words that share the same encoding under α.

Stability objective. As mentioned previously, it is more important that $|B_\alpha(w)| = 1$ when w is a frequent word, as opposed to an infrequent word, as frequent words are more likely to occur in sentences $x \sim p_{\text{task}}$. Moreover, even when $|B_\alpha(w)| > 1$, we prefer $|B_\alpha(w)|$ to be small to enable efficient computation of robust accuracy, and to aid robustness of f_{TARE} under some assumptions on f_{down} (see Appendix A.3). While we do not know what words are frequent in p_{task}, we can approximate this with the corpus word frequencies ρ. We define the stability objective $\Phi_{\text{stability}}$ as:

$$\Phi_{\text{stability}}(\alpha) = \sum_{i=1}^{N} \rho(w_i)|B_\alpha(w_i)| \tag{11}$$

$\Phi_{\text{stability}}$ is minimized when $|B_\alpha(w)| = 1$ for every $w \in V$. Therefore, connected component encodings achieve optimal $\Phi_{\text{stability}}$.

Fidelity objective. In addition to stability, we also want to maximize fidelity. Intuitively, α has high fidelity when the ambiguity induced by encoding tokens does not appreciably decrease the accuracy f^*. To formalize this intuition, we approximate the accuracy drop due to ambiguity with a loss that measures the distance between each word and its cluster in some vector space, inspired by k-means clustering. Let $\vec{v_i}$ be the N-dimensional indicator vector that is 1 at index i and 0 at all other indices. We represent the cluster C_j by a cluster centroid $\vec{\mu_j}$ defined as the average of word embeddings in C_j, weighted by word frequency:

$$\vec{\mu_j} = \frac{\sum_{w_i \in C_j} \rho(w_i)\vec{v_i}}{\sum_{w_i \in C_j} \rho(w_i)} \tag{12}$$

This definition is motivated by our expectation that downstream models will roughly use such a weighted average to represent each encoded token, similarly to how multiple word senses are known to be encoded in word vectors (Arora et al., 2018). In general, $\vec{v_i}$ could be any vector, such as a pre-trained word vector for word w_i. We choose $\vec{v_i}$ to be an indicator vector to avoid making assumptions about the downstream task.

Now, let $c(i)$ denote the j for which $w_i \in C_j$. We define the ambiguity objective Φ_{fidelity} as

$$\Phi_{\text{fidelity}}(\alpha) = \sum_{i=1}^{N} \rho(w_i)\|\vec{v_i} - \vec{\mu}_{c(i)}\|^2. \tag{13}$$

Note that if each word is in its own cluster, Φ_{fidelity} is zero. Φ_{fidelity} increases by a small amount if a frequent word and rare word are in the same cluster, and increases more when multiple frequent words are in the same cluster.

Final clustering objective. We introduce a hyperparameter $\gamma \in [0, 1]$ that balances our two objectives. The final objective Φ becomes:

$$\Phi(\alpha) = \gamma\Phi_{\text{fidelity}}(\alpha) + (1 - \gamma)\Phi_{\text{stability}}(\alpha) \tag{14}$$

Note that when $\gamma = 1$, we maximize fidelity, so each vocabulary word is assigned to its own cluster. As γ approaches 0, we get the connected component clusters from our baseline, which were designed to be maximize stability.

Defining agglomerative cluster encodings on out-of-vocabulary words. Since Φ involves $|B_\alpha(w)|$, it not only depends on the action of α on all vocabulary words, but also on all perturbations of vocabulary words. We simplify the optimization problem by imposing some constraints on α. In particular, similarly to the connected component encoding, we define $\alpha(\tilde{w})$ for $\tilde{w} \notin V$ in terms of α's action on words in V. Thus, optimizing Φ reduces to choosing $\alpha(w)$ for all $w \in V$.

First, if $\tilde{w} \notin B(w)$ for all $w \in V$, we set $\alpha(\tilde{w}) = \text{OOV}$, as before. Otherwise, note that unlike for the connected component clusters, $\tilde{w}$ could now be a perturbation of multiple words with different encodings. We use the simple rule of mapping $\tilde{w}$ to the same encoded token as the most frequent word it could be the perturbation of. Given $\tilde{w} \notin V$, we compute

$$w^\star = \arg\max_{w \in V} \{\rho(w) \mid \tilde{w} \in B(w)\}, \tag{15}$$

and define $\alpha(\tilde{w}) = \alpha(w^\star)$. This ensures that our choice of $\alpha(\tilde{w})$ does not cause $|B_\alpha(w^\star)|$ to increase, but it will cause $|B_\alpha(w)| > 1$ for all w where $\tilde{w} \in B(w)$ but $\alpha(w) \neq \alpha(w^\star)$. In Figure 2, we see that for the solid green boxes, we map *"abot"* to the same encoding as the frequent word *"about"* rather than the infrequent word *"abet"*, so that B_α still has size 1 for *"about"*.

Agglomerative clustering. Now, we have reduced the problem of minimizing Φ to choosing $\alpha(w)$ for each $w \in V$. Recall that any α induces a clustering of V, where each cluster contains a set of words mapped by α to the same encoded token. We use an agglomerative clustering algorithm to approximately minimize Φ. We initialize α by setting $\alpha(w) = w$ 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 we were to combine them into a single cluster. If no merge operation causes Φ to decrease, we return the current α. 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 representative r as the $w \in C_i \cup C_j$ with largest $\rho(w)$. We then set $\alpha(w) = r$ for all $w \in C_i \cup C_j$. Our algorithm is formally defined in appendix A.1.

Agglomerative clustering is fairly slow, as each iteration considers merging $O(|E|)$ pairs of clusters, each of which takes $O(N)$ time to evaluate. The algorithm can run for up to N iterations. As an optimization, we note that we can run the algorithm in parallel on each connected component of G, as we only consider merges involving nodes in the same connected component.

5 Experiments

5.1 Setup

Attack surface. The primary attack surface we study is edit distance one (ED1) perturbations. For each token in the input, the adversary is allowed to insert a lowercase letter, delete a character, substitute a character for any lowercase letter, or swap any two adjacent characters, so long as the first and last characters remain the same as the original token. The constraint on the outer characters is motivated by psycholinguistic studies (Rawlinson, 1976; Davis, 2003), and was also used in Pruthi et al. (2019). For example, *"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 encompasses the attack surface used by Pruthi et al. (2019), and we also allow all substitutions instead of only substitutions of adjacent letters on a QWERTY keyboard, and we perturb every input token as opposed to at most two.

Attack algorithms. We consider two attack algorithms: a brute-force attack and a heuristic attack. The brute-force attack exhaustively searches through all perturbations for an $\tilde{x} \in B(x)$ that fools the model, so it exactly computes robust accuracy. Since $|B_\alpha(x)|$ is small for many x, the brute-force attack can efficiently compute robust accuracy for models that use TARE by enumerating $B_\alpha(x)$. When $|B_\alpha(x)| > 10000$, which holds for 0.009% of our test examples, we assume the adversary can successfully perturb x. For other models, however, the brute-force attack is intractable, as $B(x)$ is too large to enumerate, so we cannot compute robust accuracy. Instead, we use a heuristic beam search attack with beam size 5. This is still prohibitively expensive as $B(x_i)$ can be large for each token x_i,

so the attack only tries at most len(x_i) perturbations in each $B(x_i)$. Even against this limited attack, we find that baseline models have low accuracy. Details are provided in Appendix A.6.

Datasets. We use six of the nine tasks from GLUE (Wang et al., 2019): SST-2, MRPC, QQP, MNLI, QNLI, and RTE. SST-2 (Socher et al., 2013) is a sentiment analysis classification task, MRPC and QQP[1] are paraphrase detection tasks (Dolan & Brockett, 2005), and MNLI, QNLI, and RTE are entailment tasks (Williams et al., 2018; Rajpurkar et al., 2016). We do not use STS-B and CoLA as they are evaluated on correlation, which does not decompose as an example-level loss. We do not use WNLI, as most submitted GLUE models cannot 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 development sets for the remaining tasks. For all datasets, we use 20% of the training set as a validation set. More details are provided in Appendix A.5.

Baseline models. We consider three baseline systems. First, we use a standard BERT model (Devlin et al., 2018) fine-tuned on the training data for each task.[2] Second, we augment the training dataset with a single[3] random perturbation of each example, and train another BERT model on this augmented data. Data augmentation has been shown to increase robustness to some types of adversarial perturbations (Ribeiro et al., 2018; Jia et al., 2019; Liu et al., 2019).

Finally, we apply an scRNN (Sakaguchi et al., 2017) typo corrector to the input, as in Pruthi et al. (2019). Following Pruthi et al. (2019), we train a task-specific typo corrector on random perturbations to the training set. At test time, inputs are fed through the typo corrector, then to the downstream model. We map out-of-vocabulary words, as defined by the typo corrector's vocabulary, to a neutral word and use BERT as our downstream model, which was the best method from Pruthi et al. (2019).

Models using TAREs. We run experiments using encoding functions defined using connected components and agglomerative clusters. To form clusters, we use the 100,000 most frequent words from the Corpus of Contemporary American English (Davies, 2008) that are also in GloVe (Pennington et al., 2014). We form agglomerative clusters using $\gamma = 0.3$ which maximizes robust accuracy on SST-2 development set. For both encodings, we train a BERT model on the training data, using $\alpha(x)$ as input. Further details and hyperparameters are provided in Appendix A.2.

5.2 Robustness gains from task-agnostic robust encodings

Our main results are shown in Table 1. We show all three baselines, as well as models using both TAREs: connected component encodings (ConnComp) and agglomerative cluster encodings (AggClust). The attack accuracy we report is an upper bound on robust accuracy obtained by running the heuristic attack, which we report since robust accuracy is only computable with TAREs.

Attacking BERT and BERT + Data Augmentation using the heuristic attack results in dramatic performance drops. Even the best performing baseline, Typo Corrector + BERT, can only achieve 38.5% accuracy on average against the heuristic attack, compared to its standard accuracy of 79.5%. In contrast, TARE provides much greater robustness. AggClust achieve average robust accuracy of 71.2%, 32.7 points higher than the attack accuracy of the best baseline, the typo corrector approach. Moreover, since the heuristic attack covers a tiny fraction of the full perturbation space, the actual gap in robust accuracy is likely higher. AggClust also outperform ConnComp in terms of both robust accuracy (by 1.5 points) and standard accuracy (by 3.7 points).

Even though TAREs were constructed in a task-agnostic manner, they consistently have high stability across our tasks. Figure 3(a) plots the distribution of $|B_\alpha(x)|$ across test examples in SST-2 and RTE. 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.7). As expected, these numbers are even higher for the connected component encodings. Even when $|B_\alpha(x)| > 1$, it is often quite small, which both helps robust accuracy and makes it efficient to compute.

The primary drawback to our representations is lower standard accuracy, which has also been observed for defenses against adversarial examples in other domains (Madry et al., 2017; Zhang et al., 2019;

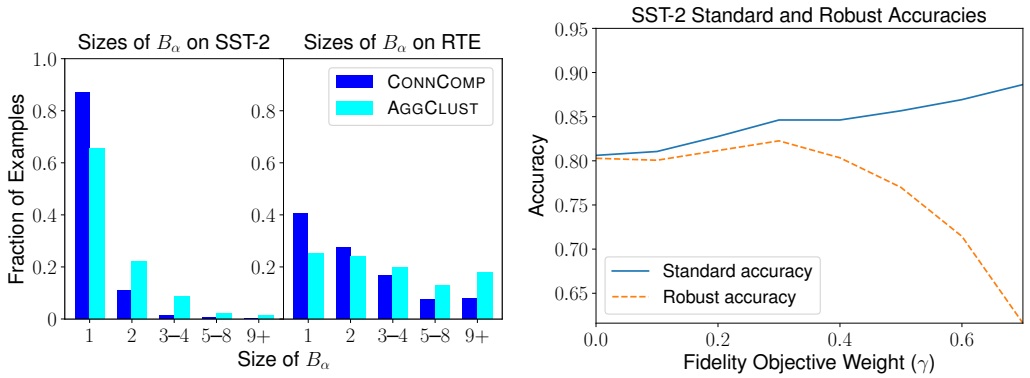
[1] `data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

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

[3] On smaller datasets, augmenting with 4 perturbations per input did not significantly change attack accuracy.

Accuracy	System	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Avg
Standard	**Baselines**							
	BERT	93.8	87.7	91.2	84.3	88.9	71.1	86.2
	Data Aug. + BERT	93.2	87.1	90.9	84.0	88.5	69.3	85.5
	Typo Corr. + BERT	90.6	82.4	87.4	78.5	80.3	59.6	79.8
	TAREs							
	Con. Comp. + BERT	80.6	79.9	84.2	65.7	73.3	52.7	72.7
	Agg. Clust. + BERT	84.6	83.8	85.2	69.2	76.4	59.2	76.4
Attack	**Baselines**							
	BERT	8.4	10.8	18.8	0.3	0.8	1.4	6.9
	Data Aug. + BERT	11.3	4.4	22.7	6.6	6.6	7.6	9.9
	Typo Corr. + BERT	51.3	41.9	58.9	21.5	34.0	23.1	38.5
	TAREs							
	Con. Comp. + BERT	80.2	79.4	82.7	62.3	71.4	47.7	70.3
	Agg. Clust. + BERT	**83.9**	**82.4**	**83.4**	**65.7**	**74.2**	**53.4**	**73.4**
Robust	**TAREs**							
	Con. Comp. + BERT	80.1	79.4	**82.1**	61.4	70.5	46.6	69.7
	Agg. Clust. + BERT	**82.2**	**80.9**	81.6	**62.9**	**71.7**	**49.8**	**71.2**

Table 1: Standard, robust, and attack accuracy on six GLUE tasks against ED1 perturbations. For baseline models, robust accuracy cannot be tractably computed, so we only compute attack accuracy, an upper bound on robust accuracy. Comparing these bounds, our agglomerative cluster encodings outperform the bast baseline, the typo corrector defense proposed by Pruthi et al. (2019) by 34.9 points. Moreover, using TAREs we can compute the robust accuracy against the worst-case adversary, which we find outperforms the typo corrector by *at least* 32.7 points.



(a) Size of B_α over SST-2 and RTE

(b) Agglomerative Clustering Tradeoff

Figure 3: (a) 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 is often 1. ConnComp encodings are more stable than the AggClust encodings, at the cost of fidelity. (b) Standard and robust accuracies on SST-2 with agglomerative cluster encodings using different values of γ. While the gap between standard and robust accuracy increases monotonically, robust accuracy increases before decreasing.

Jia et al., 2019). Compared with normally trained BERT, the agglomerative cluster encodings lead to a 9.8 point reduction in standard accuracy. Huang et al. (2019), which defends against character substitution typos using interval bound propagation (IBP), also saw a drop in standard accuracy when building robust models. In fact, on the SST-2 test set, our agglomerative cluster model achieves standard accuracy of 84.6%, 10.2 points higher than their SST-2 model, while being robust to a larger class of typos. We attribute this improvement to our ability to use a stronger downstream model, namely BERT, while IBP was constrained to shallower, less effective models. Despite our standard accuracy drop from normal BERT, our encodings with BERT exceed the performance of even the *normally trained* model (CNN) used in Huang et al. (2019) by 4.8 points.

5.3 AGGLOMERATIVE CLUSTERING TRADEOFF

In Figure 3(b), we plot standard and robust accuracy on SST-2 for AGGCLUST encodings, using different values of γ . Recall that $\gamma = 0$ corresponds to the stability maximizing CONNCOMP, and $\gamma = 1$ maximizes fidelity by putting each vocabulary word in its own cluster (but still mapping out-of-vocabulary words to OOV). At $\gamma = 0$, the gap between standard and robust accuracy is very small, as the only difference is due to out-of-vocabulary words. As γ increases, the standard accuracy increases, but so does the gap between standard and robust accuracy. As a result, robust accuracy increases for a while, then decreases.

5.4 INTERNAL PERMUTATION ATTACKS

Other work (Belinkov & Bisk, 2017; Sakaguchi et al., 2017) investigates robustness to internal permutations: reorderings of the characters in a word besides the first and last character. TARE can also be used to defend against these perturbations. For the normally trained BERT model, a heuristic beam search attack that uses internal permutations reduces average accuracy from 86.2% to 15.7% across our six tasks. Using connected component clusters defined using the internal permutation attack surface, we achieve robust accuracy of 81.4%. Full details and results are shown in Appendix A.8.

6 DISCUSSION

Adversarial typos can cause large drops in model performance in settings where correct predictions are critical, such as hate-speech detection (Hosseini et al., 2017) and spam classification (Lee & Ng, 2005). Ebrahimi et al. (2017) demonstrates that gradient-based attacks can be used to generate adversarial typos, and evaluates on the AG News dataset.

Besides typos, other perturbations can also be applied to text. Prior work considers semantic operations, such as replacing a word with a synonym (Alzantot et al., 2018; Ribeiro et al., 2018). Our framework can be extended to these perturbations, as they merely define a different set of perturbations $B(x_i)$ for each word x_i. Other perturbations, such as syntactic rearrangements (Iyyer et al., 2018) or insertion of distracting text (Jia & Liang, 2017), are more challenging to incorporate into our framework, and are interesting directions for future work.

Most existing defenses to adversarial perturbations involve specialized training procedures, and cannot be reused for multiple tasks. In computer vision, adversarial training with projected gradient descent yields models that are robust to various types of perturbations(Goodfellow et al., 2015; Madry et al., 2017). However, it is nontrivial to port these methods to NLP, where inputs and perturbations are discrete. Adversarial training has been used to defend against heuristic attacks and random perturbations (Ebrahimi et al., 2017; 2018; Cheng et al., 2019), but these models are often not robust to adversarial perturbations generated by more involved search procedures (Ebrahimi et al., 2017). Using our representations with adversarial training could resolve this issue, since the worst-case attack is readily computable. Huang et al. (2019) and Jia et al. (2019) train NLP models that are robust to worst-case perturbations, but must retrain their entire system for each task and and are restricted to shallow models, as IBP bounds become loose for deep models.

Other defenses are based on various forms of preprocessing. Pruthi et al. (2019) use the typo-corrector from Sakaguchi et al. (2017) to correct perturbed inputs, but only consider cases where one or two tokens in the input are perturbed, and use heuristic search when perturbing two tokens. We consider a much larger attack surface in which every token may be perturbed, and compute exact robust accuracy. 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.

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. Our work shows how to create task-agnostic representations that instead optimize for robustness to perturbations, as many natural tasks require robustness to the same perturbations, such as typos. We hope our work inspires new task-agnostic robust encodings that lead to more robust and more accurate models.

REFERENCES

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

S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics (TACL)*, 6, 2018.

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

Y. Belinkov and Y. Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

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

Y. Cheng, L. Jiang, and W. Macherey. Robust neural machine translation with doubly adversarial inputs. In *Association for Computational Linguistics (ACL)*, 2019.

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

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

J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

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

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

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

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran. Deceiving Googles Perspective API built for detecting toxic comments. *arXiv*, 2017.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A Appendix

A.1 Aggloemrative clustering

Recall the input to our agglomerative clusters is a vocabulary V, and our objective Φ takes in cluster assignments $C = \{C_1, ..., C_K\}$, and $C_i \subseteq V$. Our algorithm this 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 $O(N^2 E)$ 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 Experimental details

For our methods using transformers, we start with the pretrained uncased BERT (Devlin et al., 2018), using the same hyperparameters as the pytorch-transformers repo.[4]. 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[5]; 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.3 Bounding the difference between standard and robust accuracy

In (6), we show how to ensure robust accuracy if $|B_\alpha(x)| = 1$ for most $x \sim p_{\text{task}}$. Intuitively, we should be able to get high robust accuracy even if $|B_\alpha(x)| > 1$ but is relatively small. Here, we give a condition on f_{TARE} under which having small average $|B_\alpha(x)|^2$ suffices. In this section, all expectations are taken over $(x, y) \sim p_{\text{task}}$.

[4] https://github.com/huggingface/pytorch-transformers
[5] https://github.com/danishpruthi/Adversarial-Misspellings

For $(x, y) \sim p_{\text{task}}$ and for encodings e sampled uniformly from $B_\alpha(x)$, suppose that for some f_{down} and $f_{\text{TARE}}(x) \stackrel{\text{def}}{=} f_{\text{down}}(\alpha(x))$, it is unlikely that $f_{\text{TARE}}(x) = y$ but $f_{\text{down}}(e) \neq y$. More specifically,

$$\mathbb{E}\left[\mathbb{P}_{e\sim\text{Unif}(B_\alpha(x))}\left(f_{\text{TARE}}(x) = y \wedge f_{\text{down}}(e) \neq y\right)^2\right] \tag{16}$$

should be small, where $\text{Unif}(B_\alpha(x))$ denotes the uniform distribution over elements in $B_\alpha(x)$. The inclusion of the square helps make our bounds below tighter.

If f_{down} satisfies this, then we can bound the robust accuracy of f_{TARE} in terms of its standard accuracy, its accuracy on random perturbations, and the size of $B_\alpha(x)$:

$$\text{acc}_{\text{robust}} \geq \text{acc}_{\text{std}} - \sqrt{\mathbb{E}[|B_\alpha(x)|^2] \cdot \mathbb{E}\left[\mathbb{P}_{e\sim\text{Unif}(B_\alpha(x))}\left(f_{\text{TARE}}(x) = y \wedge f_{\text{down}}(e) \neq y\right)^2\right]}, \tag{17}$$

Hence, if $\text{acc}_{\text{std}}(f_{\text{TARE}})$ is large, and $|B_\alpha(x)|^2$ to be small on average, then robust accuracy will be high.

Now we prove (17), which we equivalently write as

$$\text{acc}_{\text{std}} - \text{acc}_{\text{robust}} \leq \sqrt{\mathbb{E}[|B_\alpha(x)|^2] \cdot \mathbb{E}\left[\mathbb{P}_{e\sim\text{Unif}(B_\alpha(x))}\left(f_{\text{down}}(\alpha(x)) = y \wedge f_{\text{down}}(e) \neq y\right)^2\right]}.$$

First, note that the difference between standard accuracy and robust accuracy can be written as

$$\text{acc}_{\text{std}} - \text{acc}_{\text{robust}} = \mathbb{E}\left[\mathbf{1}[f_{\text{down}}(\alpha(x)) = y] \cdot \max_{e\in B_\alpha(x)} \mathbf{1}[f_{\text{down}}(e) \neq y]\right]. \tag{18}$$

In other words, standard accuracy and robust accuracy differ exactly on examples where the model is correct on x, but is incorrect on some $e \in B_\alpha(x)$. We can upper-bound this by replacing the max operator with a sum:

$$\text{acc}_{\text{std}} - \text{acc}_{\text{robust}} \leq \mathbb{E}\left[\mathbf{1}[f_{\text{down}}(\alpha(x)) = y] \cdot \sum_{e\in B_\alpha(x)} \mathbf{1}[f_{\text{down}}(e) \neq y]\right]. \tag{19}$$

With some re-arranging, this expression becomes

$$\text{acc}_{\text{std}} - \text{acc}_{\text{robust}} \leq \mathbb{E}\left[|B_\alpha(x)| \cdot \sum_{e\in B_\alpha(x)} \frac{1}{|B_\alpha(x)|} \cdot \mathbf{1}[f_{\text{down}}(\alpha(x)) = y] \cdot \mathbf{1}[f_{\text{down}}(e) \neq y]\right]. \tag{20}$$

Applying the Cauchy-Schwarz Inequality, we can bound the right-hand side as:

$$\leq \sqrt{\mathbb{E}[|B_\alpha(x)|^2] \cdot \mathbb{E}\left[\left(\sum_{e\in B_\alpha(x)} \frac{1}{|B_\alpha(x)|} \cdot \mathbf{1}[f_{\text{down}}(\alpha(x)) = y] \cdot \mathbf{1}[f_{\text{down}}(e) \neq y]\right)^2\right]}. \tag{21}$$

Note that the expression that is squared inside the second expectation is simply

$$\mathbb{P}_{e\sim\text{Unif}(B_\alpha(x))}\left(f_{\text{down}}(\alpha(x)) = y \wedge f_{\text{down}}(e) \neq y\right). \tag{22}$$

This completes the proof.

A.4 Constrained adversaries

Using TARE, we can also 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 4 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.

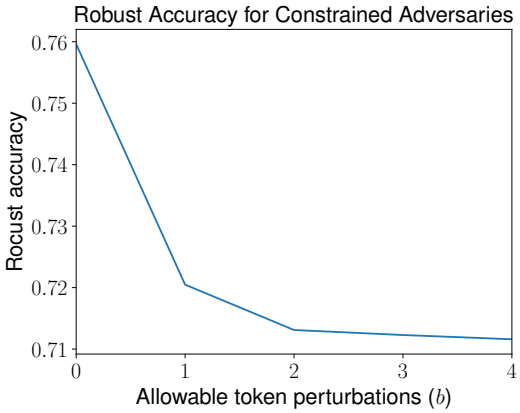


Figure 4: Robust accuracy averaged across all tasks based on different adversarial budgets b. $b = 0$ corresponds to clean performance, and robust performance is reached at $b = 4$

A.5 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 & Brockett, 2005) and the Quora Question Pairs dataset[6] 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 development set. We then retrain using the chosen hyperparameters on the full training set.

A.6 ATTACKS

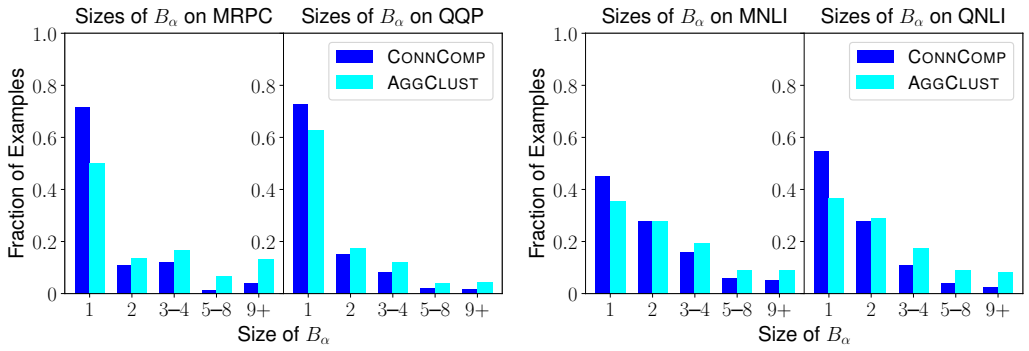
We use two heuristic attacks to compute an upper bound for robust accuracy: one for ED1 perturbations and one for internal permutations. Each heuristic attack is a beam search, with beam width 5. However, because $|B(x_i)|$ is very large for many tokens x_i, even the beam search is intractable. Instead, 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 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.7 NUMBER OF REPRESENTATIONS

We include here histograms for the datasets we did not cover. The histograms for MRPC and QQP are shown in Figure 5(a), while the histograms for MNLI and QNLI are shown in Figure 5(b). The fraction of x such that $|B_\alpha(x)| = 1$ for each dataset and each set of encodings is provided in Table A.7.

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



(a) Size of B_α over MRPC and QQP

(b) Size of B_α over MNLI and QNLI

Figure 5: Histograms showing sizes of B_α for MRPC, QQP, MNLI, and QNLI.

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.

A.8 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. Therefore, using ConnComp encodings, robust, attack, and standard accuracy are all equal. We use the attack described in A.6 to attack the clean model. The results are in .

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	86.5	86.8	75.8	83.5	62.8	81.4
Attack	BERT	28.1	15.9	33.0	4.9	6.2	5.8	15.7
	Con. Comp. + BERT	93.2	86.5	86.8	75.8	83.5	62.8	81.4
Robust	Con. Comp. + BERT	93.2	86.5	86.8	75.8	83.5	62.8	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.