# MockingBERT: A Method for Retroactively Adding Resilience to NLP Models*

**Jan Jezabek**
Hedgefrog Software LLC
jjezabek@hedgefrogsoft.com

**Akash Singh**
Salesforce Inc.
singh.akash@salesforce.com

## Abstract

Protecting NLP models against misspellings whether accidental or adversarial has been the object of research interest for the past few years. Existing remediations have typically either compromised accuracy or required full model re-training with each new class of attacks. We propose a novel method of retroactively adding resilience to misspellings to transformer-based NLP models. This robustness can be achieved without the need for re-training of the original NLP model and with only a minimal loss of language understanding performance on inputs without misspellings. Additionally we propose a new efficient approximate method of generating adversarial misspellings, which considerably reduces the cost needed to evaluate a model's resilience to adversarial attacks.

## 1 Introduction

While artificial neural networks have been able to achieve human level performance on many real-world tasks, they sometimes fail in surprising ways. (Szegedy et al., 2013) showed that state of the art computer vision models can be fooled into misclassifying objects with only limited perturbations imperceptible to human viewers. Along similar lines, it was shown in (Pruthi et al., 2019) that very constrained attacks can successfully trick classification algorithms into making incorrect predictions. In fact such attacks have been used for a long time, chiefly for evading spam classifiers while remaining legible to human readers.

Protecting against such misspellings, whether accidental or intentional, has been a focus of research in the NLP field for many years (Lee and Ng, 2005). Recently, defenses suggested by (Pruthi et al., 2019) and (Jones et al., 2020) can partially remediate adversarial attacks by adding a pre-processing step, at the cost of a drop in classification performance. (Liu et al., 2020) proposed replacing a fixed word embeddings with trained character-based ones and observed improved resilience to adversarial attacks.

In existing systems a tension exists between modularity and accuracy. (Pruthi et al., 2019) and (Jones et al., 2020) propose fully modular systems that are completely oblivious of the downstream language understanding model. This provides explainability (by providing a verbatim sequence of corrected tokens) but comes at a cost of reduced accuracy on unperturbed inputs. Additionally there is an added drawback of not being able to preserve potential ambiguity present in the input, making these systems 'destructive'. Conversely, (Liu et al., 2020) is able to represent ambiguous inputs, however at the cost of losing modularity.

Our central hypothesis is that original accuracy can be preserved while at the same time ensuring modularity. In particular we show that existing classifiers based on BERT and RoBERTa, two widely used pre-trained models, can be retroactively made resilient to perturbations even if only unperturbed data was used during the initial finetuning. This can be done by replacing the heuristic subword tokenizer and token embedding with a machine learned replacement which we call MockingBERT. The MockingBERT tokenizer and embedder learns to mimic a transformer model's tokenization and layer 0 embedding mechanism while providing resilience to input perturbations.

We evaluate the performance of such models when trained on both unperturbed and perturbed training sets to understand their suitability for data augmentation. We perform a comparative analysis with the methods proposed in (Jones et al., 2020), as well as with a regular finetuned BERT model trained with data augmentation.

*MockingBERT is a reference to mockingbirds, a group of birds known for mimicking the sounds of other animals. The code for reproducing our results as well as instructions for obtaining the trained models are available at https://github.com/akash13singh/resilient_nlp/.



*Proceedings of the 29th International Conference on Computational Linguistics*, pages 4640–4650
October 12–17, 2022.

We also propose and evaluate WORDSCOREATTACK, an efficient and effective method for generating adversarial samples without the need for exhaustively considering all possible perturbations. WORDSCOREATTACK works by carefully choosing information bearing words in the input text. This provides a much faster alternative to the exhaustive method proposed by (Pruthi et al., 2019) at the cost of perturbing a larger number of characters in the sentence. Crucially this method requires significantly fewer calls to the underlying NLP model, which more closely approximates real world scenarios that are likely to involve rate limiting and/or a limited query budget. We manually verify that the outputs of this perturbation can still be classified correctly by a human reader with a high probability.

Our central findings are that the MockingBERT tokenizer and embedder model paired with a finetuned BERT or RoBERTa classifier achieves a high level of resilience to character-level adversarial perturbations when pre-trained on perturbed data. This results in a higher accuracy on perturbed inputs on multiple well known datasets than state of the art methods for combating adversarial misspellings such as the one described in (Jones et al., 2020). Crucially, the impact on accuracy for unperturbed inputs, while measurable, is noticeably lower than for comparable methods for protecting against adversarial attacks.

Additionally, WORDSCOREATTACK significantly reduces the number of model queries required to find adversarial inputs. This dramatically speeds up evaluation, while also making the attack more practical in real scenarios.

## 2 Related Work

Previous research has explored both adversarial attacks against NLP systems as well as possible defenses. This section gives a brief overview of related work and concepts.

### 2.1 Adversarial Attacks

(Pruthi et al., 2019) proposes an attack that exhaustively searches for a modification of up to two characters with the intent of causing a wrong prediction. The allowed modifications are from a narrow set: Adding or deleting a single internal character, swapping two neighboring internal characters or replacing an internal character with one of its neighbors in the QWERTY keyboard layout. This choice of perturbations was based on linguistic research which suggests that modifications to internal characters in a word do not significantly hinder legibility for a human reader (Rawlinson, 1976).

Another approach to adversarial attacks works by replacing individual characters with similarly looking symbols or letters from different alphabets. In this scenario the text remains easily understandable to the reader even in the presence of a large number of misspellings (Eger et al., 2019; Sokolov et al., 2020).

### 2.2 Defenses Against Attacks

In addition to the attack described previously, (Pruthi et al., 2019) proposes a remediation in the form of a subcharacter recurrent neural network (ScRNN), which attempts to reverse any perturbations present in the input sentence. (Jones et al., 2020) proposes a system named RobEn that clusters misspellings of vocabulary words with a bounded edit distance and maps them to the most frequent word in the cluster. This approach works even in the case where every word in the sentence is misspelled, at the cost of reduced accuracy on unperturbed inputs.

Both of these approaches are highly modular, i.e. they can be used with any language understanding model as a preprocessing step. In contrast (Liu et al., 2020) proposes joinly training a character-based word embedder and the main NLP model. A similar approach is also considered in (El Boukkouri et al., 2020) where a character-level word embedding is passed to a transformer model with a number of parameters similar to BERT. Finally, (Provilkov et al., 2020) proposes a modification to the BPE tokenization procedure commonly used in transformer models, using a mechanism similar to dropout.

These three approaches allow the representation of ambiguous misspellings and their handling in the NLP model. Even in cases of architectures not specifically aiming for resilience to misspellings, the authors observe such resilience as a side effect of their embedding procedure.

### 2.3 Alternatives to Subword Tokenization

Our research specifically targets NLP models based on the transformer architecture (Vaswani et al., 2017), with a focus on models derived from BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). Such models are pre-trained on large corpora of unlabeled data, and can then be adapted (in a process known as finetuning) to many NLP

tasks. This substantially reduces the amount of data needed for each individual task.

An important aspect of understanding BERT is that it uses the WordPiece subword tokenization (Wu et al., 2016) process. This means that while common words are typically mapped to a single token, uncommon or invented words can still be represented by a sequence of tokens without the need to resort to a catch-all token (typically denoted as UNK). During the operation of the transformer model, tokens corresponding to subwords are first represented as context-free vectors of numbers (that had been learned during training), with subsequent layers incrementally adding contextual information from other tokens using the self-attention mechanism.

There are known alternatives to subword embeddings that are similarly able to avoid emitting out-of-vocabulary tokens. One prominent example is ELMo (Peters et al., 2018), which uses character-level embeddings in addition to a word's surrounding context to come up with an embedding for a particular word in a sentence. Similar techniques have been successfully used with transformer based models (El Boukkouri et al., 2020; Ma et al., 2020).

The tokenization schemes described so far all rely on whitespace tokenization, which means they are likely to be susceptible to adversarial attacks that insert or remove whitespace. In contrast some recent transformer based models avoid using heuristic tokenizers altogether while still using a comparable number of parameters to BERT (Clark et al., 2022), (Tay et al., 2021). This is achieved by using a convolution-like process to map the sequence of input characters to embeddings before passing them to the transformer blocks.

## 3 Data

For training the MockingBERT tokenizer and embedder we use the unlabeled BookCorpus dataset (Zhu et al., 2015), with minimal pre-processing to reverse the existing tokenization present in that dataset. The goal is to utilize textual data that is as close as possible to unprocessed text.

For evaluation purposes we use the Large Movie Review (IMDb) dataset (Maas et al., 2011), the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) and the Large Yelp Review dataset (Zhang et al., 2015). For the latter two datasets we use both the 2-class (binary) and 5-class variants. We subsequently refer to these datasets as IMDb, SST-2, SST-5, Yelp-2 and Yelp-5.

For each task we evaluate each model's accuracy on a randomly chosen subset of 500 sentences from each dataset's test set. The reason for the limited size of the test set is that adversarial attacks require potentially hundreds of model inferences for each sentence to find a successful perturbation.

## 4 Models

We propose a model, MockingBERT, that can be used in place of a transformer model's tokenizer and word embedding. The model consists of a character embedding layer that transforms each input character into a numeric vector of size 768. Characters are converted to lower case before embedding, but no other preprocessing is done (e.g. no special handling for whitespace or punctuation). The layer is followed by three stacked bidirectional LSTM layers, with a hidden size of 768. The final LSTM layer is connected to two parallel dense layers: The subword boundary detection layer and the subword embedding layer (Figure 1). The subword boundary detection layer uses a sigmoid activation function and has an output dimension of 1. The subword embedding layer uses hyperbolic tangent as its activation function and has an output dimension of 768, matching the subword embedding size of both BERT Base and RoBERTa Base.

The model is trained with two objectives. The first objective is a character-level sequence classification task. The subword boundary detection layer is used as a classifier. For every input character it predicts whether the character is a subword boundary, i.e. the last character of a subword. The second objective is a regression task where the subword embedding layer outputs the embedding for each subword. The embeddings are only used for input characters deemed to be subword boundaries. For non-boundary characters the embeddings are discarded by using a mask.

The tokenizer and embedder model is trained on sentences from BookCorpus (Zhu et al., 2015). During training an unperturbed sentence is processed by the transformer model's regular tokenizer and its context free word embedding (trained layer 0 embeddings without the position and segmentation embeddings). This gives us the subword boundaries and embeddings which are used as labels to train the model. The sentence may subsequently be perturbed by adding or deleting a

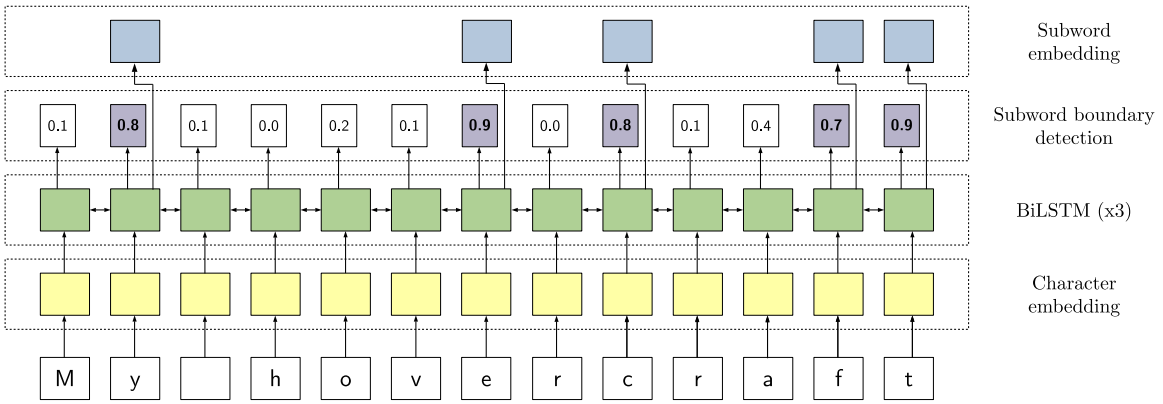


Figure 1: Structure of the MockingBERT tokenizer and embedder. In this example the phrase 'My hovercraft' is split into five tokens. An embeddings is only used if the corresponding subword boundary is detected with confidence of at least 0.5.

character, swapping two neighboring characters, inserting whitespace in the middle of long words, or removing whitespace between two words. The subword boundaries are updated accordingly. In the case of character deletion, special handling is present for characters that are subword boundaries. In this case the immediately preceding character is marked as a subword boundary. In the uncommon case when the preceding character is already a subword boundary, the embedding for the deleted character is simply discarded. In our opinion this situation is sufficiently rare that it does not warrant adding more complexity to the models.

When training, we can choose which context-free subword embedding MockingBERT will approximate:

- We can target the embedding of the generic (that is not finetuned) BERT or RoBERTa model. An advantage of this is that the trained MockingBERT embedder is independent of the finetuned task.
- Alternatively we can target the embedding of a finetuned transformer model. In this case the MockingBERT instance is no longer task agnostic, but can potentially better match the embeddings expected by the transformer model.

In our experiments we evaluate both of these approaches to understand the tradeoffs.

The models are trained on 64000 randomly selected sentences from BookCorpus for 5 epochs. The loss function is a combination of the mean squared error (MSE) losses for the subword boundary detection task and for the embedding prediction task. MSE was chosen since it can be applied both for subword boundary detection (classification) and embedding prediction (regression). The loss values for the two components are scaled to have the same magnitude. This is to prevent one of the subtasks from dominating the other one during training.

When training on perturbed data and targeting generic embeddings for BERT and RoBERTa, the models achieved accuracies for the subword boundary detection task of 99.30% and 99.25% respectively on a held-out evaluation set.

During evaluation and inference, the input text is converted to lower case and the characters are embedded as a 768-dimensional vector representation. Subsequently the MockingBERT model is executed to obtain subword embeddings for the input sequence. The embeddings are then bookended by the fixed representations of the transformer's special tokens, [CLS] and [SEP] in the case of BERT and <s> and </s> for RoBERTa. Finally, the sequences of embeddings are passed to the finetuned transformer model.

The transformer models are based on the pretrained BERT Base and RoBERTa Base models and finetuned using the HuggingFace Transformers package (Wolf et al., 2020).

For finetuning we attach a linear layer on top of the [CLS] (for BERT) or <s> (for RoBERTa) output embedding, and train the entire model using cross-entropy loss. We restrict the sequence length to 128 tokens, and the model is trained with a batch-size of 32, learning rate of 2e-05, for up to 5 epochs. We used an early stopping patience of 10, evaluated every 100 training steps.

## 5 Proposed Attacks

We propose WORDSCOREATTACK , a cost effective way to generate adversarial attacks which can occur in an real-world setting. Typically trained models are hosted and exposed through an API and the users including adversaries can only query the models by sending input and receiving output. Thus we only focus on black-box attacks, where one does not have access to the trained model.

WORDSCOREATTACK intelligently selects input words to perturb in order to maximize the chances of finding an adversarial example with the minimum number of perturbations. This is achieved by computing corpus specific word scores which are based on per-word conditional class probabilities for the corpus. For a binary classification task we compute the log likelihood of each word as shown in equations 1 to 3, where $freq_{pos}, freq_{neg}$ are the frequencies of the word in positive and negative classes respectively. $N_{pos}, N_{neg}$ are the total words in the positive and negative corpus and $V$ is the total vocabulary size. We remove stop words and low frequency words before computing word scores. In the case of multi-class classification, for each class a separate score is computed by considering that class as positive and combining all other classes into the negative class.

$$word\_score = log(\frac{P(word_{pos})}{P(word_{neg})}) \quad (1)$$

$$P(word_{pos}) = \frac{freq_{pos} + 1}{N_{pos} + V} \quad (2)$$

$$P(word_{neg}) = \frac{freq_{neg} + 1}{N_{neg} + V} \quad (3)$$

Given an input text and the original predicted class, WORDSCOREATTACK targets words in the input text which have the highest scores for the given class and perturbs them in the decreasing order of scores, until the model prediction is flipped. Our hypothesis here is that the words with the highest scores are critical to the model's predictions and perturbing them can fool the model to make the wrong classification.

We impose a query budget setting expressed by two parameters: *(max_words_to_perturb, max_tries_per_word)*. The first one denotes the maximum number of words that can be perturbed for each input text. The second parameter denotes the number of perturbation attempts allowed per word. When the *max_tries_per_word* for a word are exhausted and do not yield a successful attack, the attack greedily preserves the perturbation that decreases the model confidence the most (i.e. the one with the lowest score for the original predicted class) and moves to the next word in the order of word scores. The maximum number of model queries per input text are *max_words_to_perturb* * *max_tries_per_word*.

In order to mimic real world misspellings, we only allow one perturbation per input word. The perturbations considered are adding or deleting a character, swapping of adjacent characters, as well as splitting of a word (adding whitespace) and merging of adjacent words (deleting whitespace). Furthermore for the non-whitespace perturbations, only the internal letters of a word are perturbed, and the first and last letters remain unmodified. This ensures the perturbed text can be comprehended by humans (Rawlinson, 1976; Pruthi et al., 2019).

Though we allow one perturbation per word, multiple words per input text can be perturbed until a successful attack is found. This is in contrast to (Pruthi et al., 2019), where the authors do an exhaustive search to find a single character perturbation that flips the model prediction.

The design goal for WORDSCOREATTACK is to find an adversarial perturbation with a significantly lower query budget compared to an exhaustive search. This is done with the objective of constructing a practical framework for both simulating adversarial attacks to analyse model resilience as well as for constructing adversarial samples for data augmentation and adversarial training.

## 6 Experiments

We evaluated each model's performance on an unperturbed version of the test sets, as well as on WORDSCOREATTACK. We also evaluated the fine-tuned BERT model on the IMDb task using the exhaustive adversarial attack as described in (Pruthi et al., 2019). TextAttack (Morris et al., 2020), a Python framework for adversarial attacks, was used to execute the exhaustive attacks. Due to the prohibitive computational cost of this attack we were not able to evaluate it on the other models.

We evaluated variations both including and excluding whitespace modifications. This is because the remediations proposed in (Jones et al., 2020) have not been specifically designed for combating such modifications.

For WORDSCOREATTACK, we allowed the at-

| Model | IMDb | SST-2 | SST-5 | Yelp-2 | Yelp-5 |
|---|---|---|---|---|---|
| **BERT** | | | | | |
| (no remediations) | **88.0**/60.6/58.6 | **91.4**/42.2/38.8 | **56.2**/5.8/5.0 | 95.2/71.2/70.4 | **61.8**/27.2/25.0 |
| with RobEn CONNCOMP | 77.6/69.2/52.6 | 69.4/64.6/27.4 | 33.4/28.2/7.6 | 86.6/80.6/64.8 | 44.8/37.4/23.8 |
| with RobEn AGGCLUST | 78.6/**72.0**/53.8 | 75.8/**72.6**/33.8 | 41.0/**34.6**/6.0 | 90.4/86.4/71.0 | 52.4/**42.6**/26.0 |
| **MockingBERT with BERT** | | | | | |
| targeting generic embedding | 86.8/70.6/**69.0** | 86.2/57.0/**56.8** | 51.6/10.8/**13.2** | 95.2/**88.6**/**89.8** | 60.0/40.8/**41.0** |
| targeting finetuned embedding | 86.4/69.8/68.6 | 86.8/57.4/**56.8** | 49.0/10.8/9.4 | **95.4**/88.4/89.0 | 61.2/40.2/40.2 |
| **RoBERTa** | | | | | |
| (no remediations) | **90.8**/68.4/69.6 | **93.2**/48.6/46.8 | **57.2**/8.2/7.8 | 96.4/76.0/78.0 | **63.8**/36.8/36.0 |
| with RobEn CONNCOMP | 68.4/62.6/56.0 | 75.0/70.0/37.6 | 33.6/30.4/5.8 | 88.0/80.6/68.6 | 50.0/41.6/31.4 |
| with RobEn AGGCLUST | 75.8/71.6/60.8 | 78.2/**74.8**/40.6 | 39.8/**34.8**/7.6 | 91.4/87.2/75.2 | 56.8/**48.8**/36.6 |
| **MockingBERT with RoBERTa** | | | | | |
| targeting generic embedding | 87.2/75.4/**77.0** | 88.8/62.0/**62.2** | 52.4/16.2/15.0 | 96.0/**88.0**/**88.2** | 62.8/47.0/**47.4** |
| targeting finetuned embedding | 87.8/**76.2**/76.0 | 90.0/60.4/**62.2** | 52.8/18.6/**17.0** | **96.6**/87.4/**88.2** | 62.6/46.8/45.8 |

Table 1: For each combination of model and task we provide an accuracy score for the following three variations of the test set: An unperturbed test set; a test set using WORDSCOREATTACK excluding whitespace modifications; and a test set using WORDSCOREATTACK including whitespace modifications. The reason for providing a score when excluding whitespace modifications is that some of the remediations have not been designed to counteract whitespace perturbations.

tack to change up to ten words, with a single modification allowed per word as described in section 5. For each word, up to four attempts were made in order to find the perturbation that decreased the model's confidence the most. As before, we evaluated both variants that allow and disallow whitespace modifications.

To establish baselines, we evaluated finetuned BERT and RoBERTa models with their default tokenizers, both with and without data augmentation. We also evaluated the CONNCOMP and AGGCLUST approaches proposed in (Jones et al., 2020). Accuracy is used as the primary evaluation metric.

A second stream of experiments is focused on evaluating the efficacy and efficiency of WORDSCOREATTACK. We attack the BERT model finetuned for the IMDb task and vary the *max_tokens_to_perturb* from 1 to 40 and *max_tries_per_token* from 1 to 4. For each setting, we calculate the model accuracy on the 500 reviews in the test set. The results are shown in Figure 2. The original accuracy on the test set is 88%. which is reduced to 26.6% in the most adversarial setting of (40, 4).

In order to compare WORDSCOREATTACK to the exhaustive adversarial attack of (Pruthi et al., 2019), we evaluate a forgetful mode for WORDSCOREATTACK, where an unsuccessful perturbation is reset when the attack moves to a new token. In this mode, the attack tries to flip the model's prediction by making only one perturbation to the input text. The results are shown in Table 2, where

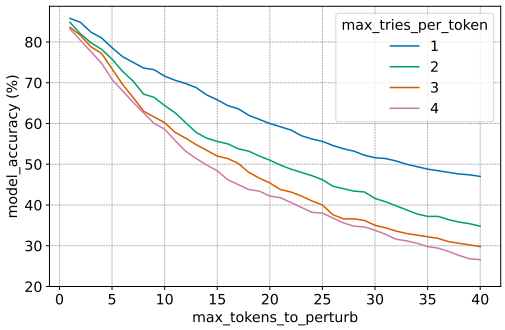


Figure 2: WORDSCOREATTACK analysis with different budget parameter settings.

the normal mode is denoted as WSA and the forgetful model is denoted as WSA-Forgetful. Both modes operate with the query budget setting of (40,4). As expected the efficacy of the attack suffers in forgetful mode, with only a 15% attack success rate and model accuracy dropping to only 72.8% from the original 88%. The exhaustive adversarial attack (Pruthi et al., 2019) on the other hand reduces the accuracy to 66.8%. However this comes at the expense of 1,574 queries on average per attack compared to only 133 queries for WSA-Forgetful. In comparison the normal mode of WSA reduces the model accuracy to 26.6%, with an attack success rate of 70% while only requiring 83 queries on average.

## 7 Analysis

Our experiment results (Table 1) show that MockingBERT consistently achieves the highest accu-

| | Pruthi | WSA-Forgetful | WSA |
|---|---|---|---|
| Orig. Accuracy | 89.6% | 88% | 88% |
| Attack Success % | 25.45% | 18% | 70% |
| Final Accuracy | 66.8% | 72.6% | 26.6% |
| Avg. Queries | 1574 | 133 | 83 |

Table 2: Comparison of exhaustive attack of Pruthi, 2019 with the forgetful (WSA-Forgetful) and normal mode of WORDSCOREATTACK(WSA). Pruthi and WSA- Forgetful are constrained to perturb only one character per input text. The budget setting for both WSA attacks is (40,4)

racy scores for adversarial attacks that allow whitespace modifications by a margin of between 5.6% and 18.8%. When whitespace modifications are disallowed, MockingBERT performs similarly to RobEn, with the exception of the SST datasets where the RobEn models achieve noticeably higher accuracy. It should be noted that pretraining MockingBERT on adversarial data with whitespace modifications results in noticeably lower performance on test sets without such modifications, as noted in subsection 7.1 (Ablation Studies).

Crucially, our model's accuracy on unperturbed data is typically only slightly lower than when using the standard BERT or RoBERTa tokenization/embedding procedure, with the accuracy scores ranging from being 0.2% better to being 4.6% worse than the standard model. This is in contrast with RobEn, where the accuracy on unperturbed data is lower by between 4.8% and 17.4%.

Interestingly the version of the tokenizer that targets default pre-trained transformer embeddings performed slighly better overall than the version targeting embeddings finetuned specifically for individual tasks. This suggests that a universal tokenizer and embedder model can be used for a variety of tasks, with no need to adapt it specifically for each task. This hypothesis requires more research on a wider variety of tasks.

Through our analysis of the WORDSCOREATTACK, we demonstrated that it is a cost effective way of constructing adversarial examples. It provides a flexible framework for evaluating models under attack. As shown in Figure 2 model accuracy drops steadily as the *max_tokens_to_perturb* is increased. Accuracy declines faster with higher values of *max_tries_per_token* as more random perturbations can be tried per token.

The cost effectiveness of the attack is borne by the fact that the most adversarial budget setting of (40,4) requires an average of 83 model queries and reduces the accuracy to 26.6%. On the other the setting of (20,1) will require at most 20 queries on average but still reduces the model accuracy to 60%.

We wanted to construct adversarial samples such that humans can infer the original intent without much difficulty. Thus we considered a limited set of perturbations with constraints such as perturbing only one internal character per token, in addition to adding/deleting whitespace. Two examples of original inputs and their adversarial counterparts are given in Table 4 in the appendix.

### 7.1 Ablation Studies

To understand how data augmentation impacts the performance of MockingBERT, we have trained and evaluated variants of our approach trained on unperturbed training sets and on training sets where no whitespace perturbations were allowed. As with the main experiment we considered MockingBERT variants mimicking both the subword embeddings of generic (that is not finetuned) transformer models, as well as those mimicking finetuned transformer models. For comparison we have also finetuned a standard BERT model using data augmentation using the same perturbations as when training MockingBERT.

For experiments in this section we have used BERT as the underlying transformer architecture and we have evaluated it on the IMDb test set. The results (Table 3) suggest that MockingBERT models require data augmentation to develop robustness to attacks. Indeed without data augmentation the accuracy of MockingBERT based models is strictly lower than that of pure BERT models. However MockingBERT based models trained with data augmentation consistently outperform similarly trained pure BERT models in the presence of test set perturbations.

Interestingly it appears that the presence of whitespace perturbations in the training set negatively affects the MockingBERT models' accuracy on test sets without such perturbations (74.4% vs. 70.6% for the generic embedding variant and 72.8% vs. 69.8% for the finetuned embedding variant). Conversely it appears that including perturbed data when finetuning the pure BERT model improves the model's accuracy on unperturbed test data. A possible explanation is that in the presence of misspellings the transformer model learns to look for multiple redundant signals for its classification, in

| Model | IMDb |
|---|---|
| **BERT** | |
| no augmentation | 88.0/60.6/58.6 |
| augmentation (incl. w/s) | **88.4**/53.6/55.4 |
| **MockingBERT targeting generic embedding** | |
| no augmentation | 86.4/56.4/53.6 |
| augmentation (no w/s) | 87.2/**74.4**/60.2 |
| augmentation (incl. w/s) | 86.8/70.6/**69.0** |
| **MockingBERT targeting finetuned embedding** | |
| no augmentation | 86.4/57.6/55.6 |
| augmentation (no w/s) | 86.4/72.8/58.8 |
| augmentation (incl. w/s) | 86.4/69.8/68.6 |

Table 3: Accuracy scores for models with various data augmentation strategies. The format is the same as in Table 1.

a way similar to the effects of dropout.

## 8 Future Work

In our opinion it would be interesting to see if an approach similar to MockingBERT would work with other practical transformer-based models such as T5, XLNet or ELECTRA. Due to differences in how their tokenizers work some adaptations might be necessary. Nevertheless we see no fundamental issue that would prevent this approach from being applicable for these other model architectures.

Another potentially interesting direction would be to evaluate whether character-based transformer models such as Canine (Clark et al., 2022), CharFormer (Tay et al., 2021) or ByT5 (Xue et al., 2022) are better suited to data augmentation with character-level perturbations than subword based transformer models. Intuitively subword based models are not well equipped for handling misspellings due to the fact that misspelled words might end up being mapped to unrelated tokens and that the number of possible misspellings for each token is very large. It should be noted that MockingBERT has a very clean separation between the tokenization/embedding procedure (which provides resilience to misspellings) and the main transformer-based language understanding layers. This means that it is easy to swap out the tokenizer and embedder if a new attack is devised, which may not be trivial for purely character-based transformer models.

## 9 Conclusion

We have demonstrated that our proposed MockingBERT embedder is able to successfully mimic the operation of a traditional tokenizer and embedder as used in the BERT and RoBERTa models, with only a modest decrease in performance on classification tasks. In the presence of input perturbations, MockingBERT outperforms both a data augmented BERT model and the state of the art RobEn procedure. Furthermore we have provided evidence that a universal embedder can achieve similar results to one that is specifically trained for a particular finetuned embedding, suggesting that embeddings might not need to be trained with specific tasks in mind. We have also proposed an efficient and effective method for constructing adversarial attacks, WORDSCOREATTACK, which allows constructing such attacks at a fraction of the cost of an exhaustive search, at the expense of possibly perturbing more words within a sentence in order to achieve a similar attack success rate.

## Acknowledgements

The authors would like to thank prof. Christopher Potts and the course facilitators and staff of XCS224U for their support and feedback.

Akash Singh would like to thank Salesforce Inc. for funding his participation in XCS224U.

We would like to thank the authors of the HuggingFace and TextAttack packages for making these highly useful tools freely available.

## Authorship Statement

The ideas presented in this article have been jointly developed and refined by all the authors. Akash Singh implemented WORDSCOREATTACK, and performed the finetuning of the BERT model as well as the exhaustive adversarial attacks. Jan Jezabek implemented and trained the joint tokenizer and embedder models, and implemented the perturbers and the evaluation notebook.

## Impact Statement

Our main motivation for this research is to counteract intentional adversarial techniques designed to evade spam or toxic/obnoxious speech detection systems. We think that making such systems more robust results in more efficient moderation systems and more civil online discourse.

That said techniques such as MockingBERT can potentially be abused for surveillance and censorship purposes, by making it harder to fool and evade systems used for monitoring.

Similarly, WORDSCOREATTACK can make it practical to evade text classification systems, re-

gardless of whether such systems' goals can be considered noble or nefarious.

Training of the MockingBERT models described in this article took slightly over 20 hours of compute time on Nvidia Tesla K80 and Nvidia GeForce GTX 1070Ti GPUs. Evaluation of the models on WORDSCOREATTACK took 25 hours, with an additional 50 hours of GPU time spent during development.

## References


Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Honglak Lee and A. Ng. 2005. Spam deobfuscation using a hidden markov model. In *CEAS*.

Hui Liu, Yongzheng Zhang, Yipeng Wang, Zheng Lin, and Yige Chen. 2020. Joint character-level word embedding and adversarial stability training to defend adversarial text. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8384–8391.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Graham Ernest Rawlinson. 1976. *The significance of letter position in word recognition*. Ph.D. thesis, University of Nottingham.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for

semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mark Sokolov, Kehinde Olufowobi, and Nic Herndon. 2020. Visual spoofing in content-based spam detection. In *13th International Conference on Security of Information and Networks*, pages 1–5.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

# A Supplemental Data

| Original Input | Adversarial Input |
|---|---|
| the hand of death most definitely rates a ten on a scale of one to- due, in no small part, to john woo's masterful direction, coupled with kat's superb cinematography: some of the leisurely tracking shots alone are worth the price of a rental; there are moments when this one borders on becoming an art-house film. both james tien and sammo hung make for the kind of villains you can't help but love to hate. tien is particularly good as the baddest of the bad. it's a role reversal the likes of which i don't think i've ever seen before (tien normally played a hero and, in fact, with his, | the hand of death most defini tely rates a ten on a scale of one to- due, in no small part, to john woo's mastreful direction, coupled with kat's supreb cinematography: some of the leisurely tracking shots alone are worth the price of a rental; there are moments when this one borders on becoming an art-house film. both jtames tien and sammo hung make for the kind of villains you can't help but loe to hate. tien is particularly good as the baddest of the bad. it's a rloe reversal the likes of which i don't think i've ever seen before (tien normally played a hero and, in fact, with his |
| i caught this movie right in my eye when i was passing by a hall of posters in the nearby cinema. the tag line was sort of confusing and immediately after reading it, i thought of the possibility of it being similar to national lampoon's dorm daze. i liked that movie, aside from having a huge collection of such genres, i decided to hit it to the cinemas right after my exams for a tension releaser.<br /><br />delightfully, i came out smiling from cheek to cheek and had an equally great amount of laughter at bits and points of the movie. amanda aynes definitely kicked it off better than keira | i caughtthis movie right in my eye when i was passing by a hlal of posters in the nearby cin ema. the tag line was sort of confusing and immediately after reading it, i thought of the possibility of it being similar to national lampoon's dorm daze. i lciked that movie, aside from having a huge collection of such genres, i decided to hit it to the cinemas right after my exams for a tension releaser.<br /><br />delig htfully, i came out smliing from cheek to cheek and had an equallygreat amount of laughter at bits and points of the movie. amanda bynes definitely kicked it off better than keira |

Table 4: Examples of original inputs with adversarial counterparts.