Close or Cloze? Assessing the Robustness of Large Language Models to Adversarial Perturbations via Word Recovery

Luke Moffett & Bhuwan Dhingra
Department of Computer Science
Duke University
Durham, NC 27708, USA
{luke.moffett,bhuwan.dhingra}@duke.edu

Abstract

The current generation of large language models (LLMs) show a surprising degree of robustness to adversarial perturbations, but it is unclear when these models implicitly recover the original text and when they rely on surrounding context. To isolate this recovery faculty of language models, we study a new diagnostic task—*Adversarial Word Recovery*—an extension of spellchecking where the inputs may be adversarial. We collect a new dataset using 9 popular perturbation attack strategies from the literature and organize them using a taxonomy of *phonetic*, *typo*, and *visual* attacks. We use this dataset to study the word recovery performance of the current generation of LLMs, finding that proprietary models (GPT-4, GPT-3.5 and Palm-2) match or surpass human performance. Conversely, open-source models (Llama-2, Mistral, Falcon) demonstrate a material gap between human performance, especially on visual attacks. For these open models, we show that performance of word recovery without context correlates to word recovery with context, and ultimately affects downstream task performance on a hateful, offensive, and toxic classification task. Finally, to show improving word recovery can improve robustness, we mitigate these attacks with a small Byt5 model tuned to recover visually attacked words.[1]

1 Introduction

Intentional, often adversarial, character perturbations are common internet content. For instance, social media users obscure text with similar-looking characters to avoid content moderation, which can be used to propagate harmful content (Rodriguez and Rojas-Galeano, 2018; Le et al., 2023). Despite their simplicity, these attacks can affect a model's ability to perform downstream tasks (Belinkov and Bisk, 2017; Dionysiou and Athanasopoulos, 2021; Pruthi et al., 2019; Boucher et al., 2022; Wang et al., 2023). The attacks are effective when they target words that are difficult to ascertain from context (Li et al., 2020). While large language models (LLMs) are generally quite good at recovering whole tokens, some cannot be reliably predicted from context. Moreover, when evading moderation,

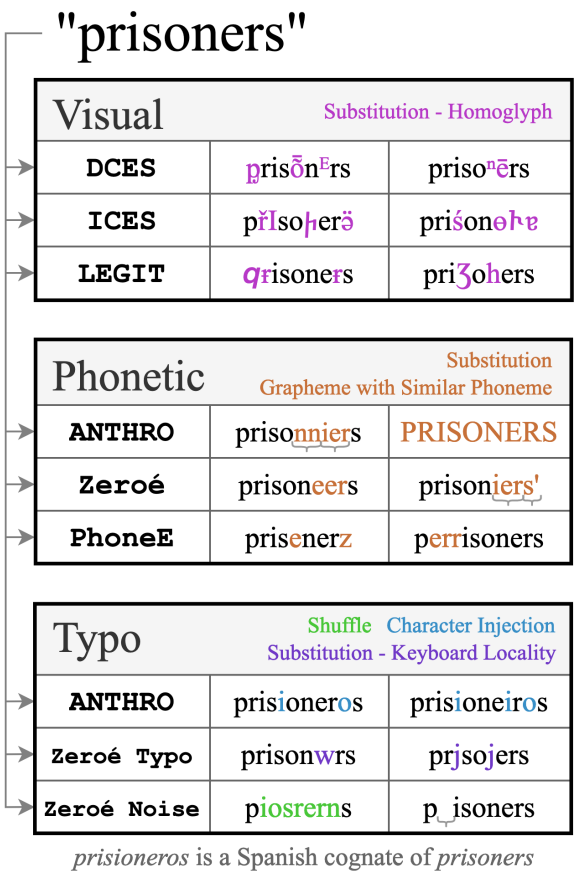


Figure 1: **Perturbation Attack Examples by Class Using the 9 Attack Strategies in Ad-Word.** Each attack strategy is used to perturb each word in Ad-Word multiple times. Phonetic attacks use phonetic similarity between graphemes. Typo attacks mimic accidental typographical errors. Visual attacks use homoglyph character level substitution.

[1]The code is available at https://github.com/lmoffett/cloze-or-close.

Proceedings of the 31st International Conference on Computational Linguistics, pages 6999–7019
January 19–24, 2025.

attackers inject their attacks only into contexts of their choosing. Truly robust models may need to both implicitly recover the perturbations (by finding "close" words) and use surrounding context to identify them (i.e., a cloze task). While the latter is the basis for masked language modeling and has been shown to improve with model size, little attention has been paid to the former.

To remediate this, we begin our study with a new task: **Adversarial Word Recovery**, which requires a model to predict the original *clean* word, like "antivaxxers", from a *perturbed* version, like "anti$^{\vee}$axxers". In the context-free setting, this provides a lower bound on a model's ability to denoise arbitrary text inputs where surrounding context might make the task easier. To enable these experiments, we create a new dataset, `Ad-Word`. `Ad-Word` contains $7,911$ words perturbed multiple times with 9 different attack strategies creating $325,724$ unique perturbations. The attack strategies are organized into *phonetic*, *typo*, and *visual* classes (shown in Figure 1 along with examples). We also establish human recovery accuracies for each attack.

We evaluate several LLMs, under both zero-shot and few-shot settings on `Ad-Word`: (1) GPT-4 (ChatGPT, `gpt-4-1106-preview`), (2) GPT-3.5 (ChatGPT, Models: `gpt-3.5-turbo-0301`, `gpt-3.5-turbo-0613`), (3) Palm-2 (`text-bison-001`) (Anil et al., 2023), (4) Llama-2 Instruct 7B, 13B, and 70B (Touvron et al., 2023), (5) Falcon Instruct 7B and 40B,[2] (6) Mistral Instruct 7B (Jiang et al., 2023).[3][4] We observe a clear gap between the proprietary models (1-3) and the open-source ones (4-6). GPT-4 exceeds the accuracy of our human annotators, while GPT-3.5 and Palm-2 are comparable to them. The open-source models are marginally worse than humans on typo and phonetic attacks, but the gap on visual attacks is multiple times larger than the other classes. A baseline fine-tuned ByT5 model is inferior in recovering typo and phonetic attacks, but surprisingly exceeds even GPT-4 on visual attacks - but only when fine-tuned exclusively on those attacks. We also find that accuracy of these models generally improves with model size, and as few-shot examples are added to the prompt, even when the examples come from attacks not under test.

Are these failures of open-source models in the context-free setting indicative of their robustness in the contextual setting? To answer this, we evaluate Llama-2 and Mistral in two adversarial settings with context. First, we show that even after adding context in the form of a paragraph surrounding the attacked word, their accuracy is less than 50% for words which they are unable recover in the context-free setting. Second, we evaluate these open models on a hateful, offensive, and toxic (HOT) speech classification task, where visual attacks are also more effective at changing model behavior. Together, these results demonstrate a correlation between context-free recovery accuracy and task performance, suggesting that improving this capability is important for improving the robustness of these models in the future. Meanwhile, we show that a simple and efficient defense based on *fine-tuning* the byte-level ByT5 model on the visual attacks in `Ad-Word` (Xue et al., 2022) can mitigate the effects of these attacks.

The main contributions of this work are to show: (1) larger LLMs recover words more accurately, a mechanism by which they spontaneously improve robustness. The largest, proprietary models are human-like; (2) There is a significant performance gap in recovering visual attacks between open and proprietary models; and (3) vulnerability of open models to visual attacks can be exploited by attacking words that are hard to recover from context.

2 Related Work

Adversarial Text Perturbation. Phishing attacks such as "spoofing" have long relied on the human ability to unintentionally match homoglyphs (Kaushik et al., 2021). An early defense was Punycode (Costello, 2003b), encoding `UNICODE` as `ASCII`. It is still used in the DNS protocol (Costello, 2003a).

Early neural NLP models, including pretrained language models, were shown to be susceptible to text perturbation with attack success rates of over 90% (Liang et al., 2018; Belinkov and Bisk, 2017; Eger et al., 2019; Eger and Benz, 2020). Belinkov and Bisk (2017) propose defending against these attacks by including the perturbations directly in the model training data (adversarial training) or by changing the underlying representation of characters in the model to be similar across synonyms and homoglyphs. Rust et al. (2022) proposed na-

[2] `https://huggingface.co/tiiuae/falcon-40b-instruct`

[3] "Instruct" is omitted from here on.

[4] Model Licenses are: ChatGPT, Palm2, Llama2 - Proprietary; Falcon, Mistral - Apache 2.0

tive robustness by using pixels rather than character embedding to construct a language model.

Adversarial attacks in NLP can largely be divided into two categories. The first category involves replacing words or longer phrases with semantically equivalent units (Alzantot et al., 2018; Gao et al., 2018; Garg and Ramakrishnan, 2020; Hofer et al., 2021; Morris et al., 2020). The second category involves perturbing characters within a word in such a way that the symbolic representation changes, but humans can still read the word. This paper focuses on the second category. Belinkov and Bisk (2017) and Pruthi et al. (2019) show that even a single misplaced character in a sentence can reduce the accuracy by up to 50% of BERT-based models. More recent work has proposed homoglyph attacks (Eger et al., 2019; Seth et al., 2023; Dionysiou and Athanasopoulos, 2021). Other papers have scraped perturbations from the internet (Le et al., 2022; Belinkov and Bisk, 2017; Rodriguez and Rojas-Galeano, 2018). Eger and Benz (2020) create a benchmark of low-level adversarial attacks called Zeroé, which we leverage.

There is a long history of spell checkers designed to correct natural misspellings (Choudhury et al., 2007; Whitelaw et al., 2009), but these are not intended to work with adversarial perturbations. Sakaguchi et al. (2017) explore improving spellchecking with an RNN, and Pruthi et al. (2019) and Keller et al. (2021) build shielding models based on RNNs and BERT, respectively. Rodriguez and Rojas-Galeano (2018) shield a toxicity classifier from obfuscations using obfuscations collected from online comments. Jayanthi et al. (2020) developed a neural toolkit, NeuSpell, that treat spell checking as a sequence labeling task. Seth et al. (2023) recover a visual attack using few-shot GPT-3 prompts. Our work leverages existing attacks from many of these works to study general purpose word recovery ability of models.

Content Moderation using NLP. Content moderation is a major evolving concern in the social media age, which has spurred industry and academic efforts to clearly define content moderation policies and automate their implementation (Gillespie, 2020). The result has been the creation of datasets covering a variety of harmful speech types (e.g., hateful or threatening) and an ongoing application of state-of-the-art natural language processing techniques to the detection of such speech (see reviews by Poletto et al. (2021) and Jahan and Ousalah (2023)). The most recent work has leveraged the advent of prompted, transformer-based LLMs. For instance, Poletto et al. (2021) find that GPT-3.5 achieves 0.89 AUC on the HateCheck benchmark (Röttger et al., 2020) and Bauer et al. (2024) find it achieved state-of-the-art (69.9 F1-Macro) on hate speech classification in Tweeteval (Barbieri et al., 2020). Using a Llama-2 based pipeline, Sasidaran and J (2024) achieve state-of-the-art (96.7 F1-Macro) on the dataset from Davidson et al. (2017). Kumarage et al. (2024) and Li et al. (2024) both demonstrate the prompt format, especially context, materially affects ChatGPT's harmful speech classification performance (we leverage prompts from (Li et al., 2024)). Combining both threads of our work, Cooper et al. (2023) combine adversarial attacks and hateful speech classification; they study hateful comment classification of transformer-based language models in the face of homoglyph attacks, but exclusively using smaller, fine-tuned models. Our study extends that work to larger, prompted models and extends previous studies of prompted large language models for harmful speech classification to adversarial inputs.

		Train	Valid	Test
	Unique Words	5,131	1,200	1,580
	Attack	**Train**	**Valid**	**Test**
Unique Perturbations	ANTHRO Phonetic	17,515	4,063	4,771
	Zeroé Phonetic	28,233	6,504	7,443
	PhoneE	24,246	5,545	6,434
	ANTHRO Typo	15,009	3,549	4,106
	Zeroé Typo	19,765	4,697	5,298
	Zeroé Noise	26,811	6,209	7,148
	DCES	28,722	6,624	7,560
	ICES	29,321	6,762	7,713
	LEGIT	27,797	6,481	7,398
	Total	**217,419**	**50,434**	**57,871**

Table 1: **Distribution of perturbations in Ad-Word by attack strategy.** The same words are used for each attack strategy in each split; each clean word appears in only one split.

3 Ad-Word Dataset

3.1 Task Definition

Let w be a word from a dictionary D. D may be multilingual, but in Ad-Word it is English. Let U be the set of all possible strings constructed with characters in UNICODE. A *word-level perturbation*

function maps words $p; \Phi : D \rightarrow U$, where Φ is a set of function-specific parameters, including a random seed. Note that $p(w; \Phi)$ is restricted to changes to the characters in a string, excluding text formatting changes. The Adversarial Word Recovery task is to learn a *recovery* function $r(\cdot)$ such that $r(p(w; \Phi)) = w$ where $p(\cdot)$, Φ and D are not observed. Task performance is measured by recovery accuracy.

3.2 Attack Taxonomy

The `Ad-Word` dataset is created by aggregating 9 attack strategies from the literature and grouping them into three classes: **phonetic**, **typo**, **visual**. Humans have a multifaceted process for decoding words that uses a mix of letter recognition, complete word recognition, phoneme pronunciation, and surrounding context (Castles et al., 2018) (see Appendix A). The attacks in `Ad-Word` add noise to which these mechanisms are robust (excluding context). Since changing letters has the potential to simultaneously affect multiple recovery paths, attacks in `Ad-Word` are organized around the what characteristic the attack is meant to **preserve** (Eger and Benz, 2020; Eger et al., 2019; Seth et al., 2023; Morris et al., 2020). For instance, visual attacks are meant to preserve the visual similarity between the perturbed word and the original word, but they do not attempt to preserve the phonetic similarity. As a result, the different attack classes require a different mechanism to recover. The main design goal of `Ad-Word` is to isolate which of these mechanisms a model has learned (or can approximate). A short description of each attack is presented in § 3.3, and a detailed description in Appendix B.

3.3 Dataset Construction

Following Seth et al. (2023), we use the most frequent 10,000 words from the Trillion Word Corpus (Kaufman, 2012) excluding words of length less than four.[5] To bound performance of models that ignore non-`ASCII` characters or use only common dictionaries, we add 250 uncommon English words to the test set, and 100 common English borrow words that are frequently stylized with accents; 50 to the train set, 25 to the test set, and 25 to the validation set, determined from the Wikitext corpus (`wikitext-103-v1`) (CC) (Merity et al., 2016).[6] Table 1 shows the overall statistics of each attack strategy in `Ad-Word`.

Phonetic Attacks. For phonetic attacks, word-level perturbations are constructed either through grapheme-level perturbations using similarity between their corresponding phonemes (`Zeroé` (Apache 2.0) Phonetic (Eger and Benz, 2020) and `PhoneE`), or through extraction from internet corpora (`ANTHRO` Phonetic (Le et al., 2022)). `PhoneE` is new to this paper and is described in detail in § 3.4. `ANTHRO` Phonetic is a subset of `ANTHRO` preserving phonetic similarity.

Typo Attacks. Two of the typo strategies, `ANTHRO` Typo (Le et al., 2022) and `Zeroé` Typo (Eger and Benz, 2020), are composed largely of perturbations from internet corpora. The `Zeroé` Noise strategy is a subset of `Zeroé` synthetic perturbations that randomly adds, deletes, and rearranges characters. `ANTHRO` Typo is a subset of `ANTHRO` not preserving phonetic similarity.

Visual Attacks. All visual attacks, VIPER (Apache 2.0) `DCES`, VIPER `ICES` (Eger et al., 2019), and `LEGIT` (Seth et al., 2023) use character-level homoglyph substitutions from a nearest-neighbor mapping, with different attack strategies using different embedding spaces to construct the mapping.

3.4 PhoneE Algorithm

`PhoneE` - ***Phone**tic **E**vasion* - occurs in three stages. First, a word is converted into its phonetic representation using the CMU Pronouncing dictionary (BSD 3).[7] Next, the graphemes in the word are mapped to the phonemes produced by the pronouncing library by using an English grapheme-to-phoneme dictionary, described below. We then uniformly sample graphemes g_i from the word to perturb with $p_g = .2$, with a minimum of 1 grapheme. Lastly, for each selected grapheme g_i, we take the phoneme it maps to p_i, and sample from graphemes for the same phoneme excluding g_i: $g_i^{'} \in G_{p_i} - g_i$. $g_i^{'}$ is determined similarly to the visual attacks, where we sample $f = j + 1$, $j \sim Geom(.05)$. However, we do not create a nearest neighbor space of each grapheme, but instead order the graphemes by frequency of use as the phoneme p_i, as observed in the Wikitext corpus (`wikitext-103-v1`) (Merity et al., 2016). Finally, we exclude leading vowels from perturbation, and only replace leading consonants with other common leading graphemes for the phoneme.

The grapheme-phoneme dictionary is a modified version of the English *Sound-to-spelling corre-*

[5] https://huggingface.co/datasets/dvsth/LEGIT

[6] https://huggingface.co/datasets/wikitext

[7] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

spondences dictionary from Wikipedia, where we added grapheme-phoneme correspondences that occur within LEGIT, and remove graphemes that require interjected letters. For consonants, we also subcategorize the graphemes as occurring at the beginning of words or not. The grapheme-for-phoneme frequency is calculated by performing the mapping procedure on each word in the Wikitext corpus. If a word has multiple pronunciations, the grapheme-phoneme is counted once for each pronunciation in which it occurs. Graphemes of words in the Wikitext corpus missing from the CMU Pronouncing dictionary are omitted from frequency calculation.

4 Context-Free Recovery Experiments

We conduct two sets of experiments. In this section, we directly apply Ad-Word to state-of-the-art pretrained models to asses their recovery ability via prompting without context. In the next section, we investigate how this recovery ability correlates with performance against the same attacks with context.

4.1 Setup

We perform both zero-shot and few-shot (50 clean-perturbed pairs) prompting followed by a batch of 20 or 5 new words for the model to recover using ChatGPT, Palm2, Llama2, Mistral, and Falcon as the underlying models. 50 samples are provided to provide greater coverage of the observed attacks since an individual sample contains only one word and few perturbations. In the *All* few-shot setting, the prompt examples are sampled uniformly from perturbations generated using all classes. In the *In-Domain* setting, perturbations are sampled uniformly from all strategies in the class being tested. Finally, in the *Out-of-Domain* setting, few-shot prompts include examples from the two attack classes not being tested. For instance, when testing phonetic recovery, we sample examples from visual and typo attacks. Example prompts for each of the models are in Appendix C.

To save on computation costs, we run the largest models (ChatGPT, Palm2, Llama2 70B, Falcon 40B) on reduced test sets of $1,400$, $1,400$, and $4,500$ for the *In-Domain*, *Out-of-Domain* and *All Classes*, respectively, sampled to match the length distribution of Ad-Word.

4.2 Baselines

Spellcheck. We apply a standard spell checker GNU Aspell (Atkinson, 2019) (GNU).[8]

Human. Five annotators from the authors' institution, proficient in English, provided 500 annotations each via a spreadsheet where perturbed words are provided in one column. Annotators were asked to input their guess for the original word in the adjacent column. Annotators achieved overall accuracy of 63.8%, 62.0%, 68.2%, 65.2%, 68.1%. A full description of the annotation environment, including the word sampling procedure, is in Appendix D. Note that these human accuracies are *not* an upper bound on model performance as humans are prone to spelling mistakes when typing the recovered words.

Finetuning. We finetune tokenization-free ByT5 (Xue et al., 2022) by varying the fine-tuning sets instead of the prompt examples (Appendix E). We use ByT5-Base ($580M$ parameters) for our tests, but we also replicate the *All Class* tests on ByT5-large ($1.2B$) and ByT5-XL ($3.7B$). Tokenization-free byte-level encodings and pretraining on a wide variety of scripts make it ideal for the word recovery task, where we expect the perturbations to consist of arbitrary combinations of UNICODE characters (Xue et al., 2021). We also evaluated Byt5's generalization to unseen attacks by finetuning using a held-out setting (Appendix F).

4.3 Recovery Results

LLM accuracies, relative to the human baseline, over the classes of attacks are in Table 2. Figure 2 shows the best performing models on each specific attack. The annotators and models found the same attacks relatively difficult within a class (Figure 2). For instance, ICES is harder than LEGIT which is harder than DCES.

Zeroé Typo and Zeroé Phonetic were relatively easier for the models than humans (not shown); all of the models except Falcon-7B exceeded the annotators on Zeroé Typo, which is the attack that looks most like the kinds of typos that randomly occur on the internet.

Beyond this, the differences in performance are large. GPT-4 exceeds human annotators by 8.7% in the 50-shot all-class setting, only not exceeding on phonetic attacks (-2.9% zeroshot, -5.2% 50-shot), and Falcon-7B is no better than -30.7%

[8] http://aspell.net/man-html/

	Ann	Asp	Set'g	Falcon 7B	Falcon 40B	Mis'l 7B	Llama2 7B	Llama2 13B	Llama2 70B	Palm2 bison	GPT-3.5 0301	GPT-3.5 0613	GPT-4 1106	Byt5 base
Phone	72.3	65.8	ZS	-37.4	-9.6	-8.1	-14.6	-9.9	-15.7	**1.2**	-8.7	-2.4	-2.9	–
			ID	-19.5	-1.1	-2.5	-6.9	-7.4	-8.9	**3.2**	-8.9	-2.3	-5.2	-6.4
			OoD	-20.6	-3.4	-4.0	-8.9	-9.7	-11.6	**2.6**	-8.9	-2.3	-2.8	-26.2
Typo	56.9	47.2	ZS	-30.7	-3.3	-6.9	-9.1	-6.4	-2.2	-0.9	-0.4	3.0	**9.3**	–
			ID	-14.9	1.1	-5.0	-6.3	-5.1	-2.2	2.5	-0.6	3.4	**10.2**	-11.5
			OoD	-15.7	-0.9	-5.4	-6.2	-5.4	-0.8	1.1	-0.7	3.4	**9.7**	-25.7
Visual	63.2	37.9	ZS	-54.0	-20.9	-30.4	-30.9	-29.3	-22.9	-8.7	-3.7	-1.0	**11.3**	–
			ID	-39.1	-13.9	-28.0	-25.8	-25.1	-18.5	-3.1	-3.6	-0.7	**12.1**	15.7
			OoD	-39.4	-14.3	-26.9	-26.6	-25.9	-18.3	-4.8	-4.0	-0.5	**10.2**	-40.9
All	64.1	50.3	ZS	-40.7	-11.3	-15.1	-18.2	-15.2	-13.6	-2.8	-4.3	-0.1	**5.9**	–
			ID	-24.5	-4.5	-11.8	-13.0	-12.5	-9.9	0.9	-4.4	0.1	**5.7**	-0.7
			All	-22.7	-1.6	-9.9	-13.9	-11.5	-8.0	2.0	-2.0	4.8	**8.7**	-7.0

Table 2: **LLM Performance on `Ad-Word`.** *Ann.* is Annotators. *Asp* is GNU Aspell. *Set'g*, *ZS*, *ID*, and *OoD* are Setting, Zeroshot, In-Domain, and Out-of-Domain, respectively. Model accuracy is in percentages, reported relative to annotator accuracy. *ID* and *OoD* settings use 50-shot prompts. For **All** attacks, *ZS* and *ID* results are the mean of class-level tests. *All* is a separate setting where examples and test samples are uniformly sampled from all classes. Byt5 tests are performed on fine-tuned using the training sets used for prompting. Proprietary models have a clear advantage over open-source ones, which particularly struggle with visual attacks. **bold** - best prompted accuracy. blue - exceeds annotators.

zeroshot. GPT-4's performance on visual attacks is notable given (1) it is the only prompted model that exceeds human performance, (2) it uses the same string tokenizer as GPT-3.5.[9] Note that we only used the text API for GPT-4; a study of GPT-4 vision's capabilities is beyond the scope of this work. The same is true of Palm2, which, like GPT-3.5, has roughly human performance - although, it does not do as well on visual attacks zeroshot (-8.7%) as 50-shot (-3.1%). Palm2 is the best performing model against phonetic attacks ($+1.2\%$ zeroshot, $+3.2\%$ 50-shot). Before these tests, it was unclear if human annotators represented an upper bound on the performance on these tasks. Given that multiple models exceed the annotators on multiple attacks, it is unclear where the upper bound lies.

Among open-source models, larger sizes work better (average zeroshot performance): Llama2 -13.6% (70B) vs. -15.2% (13B) vs. -18.2% (7B), Falcon -11.3% (40B) vs. -40.7% (7B). This effect was previously undocumented, and may explain part of the process by which larger models become more robust (Zhu et al., 2023). It might also explain the superior performance of proprietary models—while we do not know the exact sizes of these models under the hood, prior versions of PaLM and GPT were 540B and 175B parameters, respectively (Chowdhery et al., 2023; Brown et al., 2020).

[9] `https://platform.openai.com/tokenizer`

All models recover shorter words at a lower rate, but word length has a larger effect on smaller models. Specifically, the Pearson Correlation Coefficient between the accuracy of a model and the mean length of the words it fails to recover is -0.869 (p=0.0011, n=10 models) for 0-shot prompting. As model size increases, the performance gains increasingly come from the ability to correctly recover shorter words.

The open models all have difficulty with visual attacks (as does `GNU Aspell`). The best model is Falcon-40B, which is -20.9% in the zeroshot setting and -13.9% in the 50-shot setting. Only one open model does worse than -20.9% on any other attack class, Falcon-7B. Fine-tuned Byt5-base does exceptional on these visual attacks, exceeding both the annotators and GPT-4. However, this only occurs when Byt5-base has been fine-tuned on the attacks. In the out-of-domain settings, it is the worst performing model on all classes. Byt5-large and Byt5-XL results are omitted from Table 2 because they change by less than 2% in most settings from Byt5-base.

Because ChatGPT and Palm2 are proprietary and invoked via their APIs, we do not know if the gap between proprietary and open models is partly the result of preprocessing. Visual attacks, which use a broad set of `UNICODE` characters, may be particularly amenable to preprocessing, as we investigate in § 5.2.1. These results highlight the need for strategies to improve the robustness to visual attacks of open models.

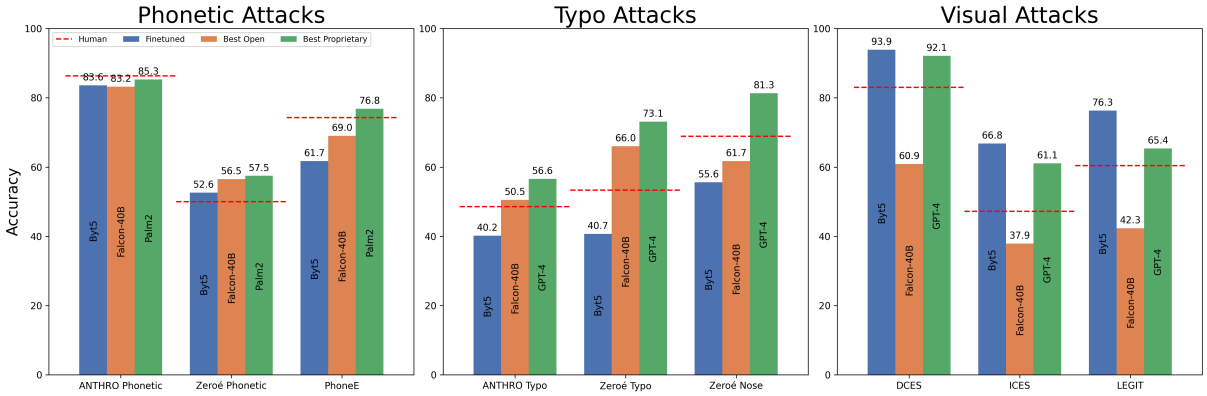


Figure 2: **Best Performing Models by Type and Attack.** Training data for fine-tuning and prompting both come from all attacks within the class. For instance, for visual attacks, they include DCES, ICES, and LEGIT. Human baselines come the annotation environment described in Appendix D. Falcon-40B is the best open model on all attacks, due in large part to improvement under 50-shot prompting.

5 Experiments with Context

We turn to the question of if differences in recovery ability result in vulnerability to these attacks in the presence of context, particularly for visual attacks. We conduct two experiments: First, we attempt word recovery on a single perturbed word within a paragraph. Second, we test the impact of these perturbations on a HOT-classification task.

5.1 With-Context Word Recovery

To test recovery in-context, we perturb a single word from academic paper abstracts at random, and then prompt the models to recover the word. The dataset was constructed by sampling 5,000 abstracts from arXiv, published in October 2023 (after Llama2 7B and 13B and Mistral 7B were released). We then randomly select a single word, which we ensure occurs only once in the abstract using natural language toolkit's (NLTK) porter stemming.[10] We perturb the selected word using visual, phonetic, and typo attacks. To ensure the attacked texts remain largely recoverable by humans and therefore valid attacks, we choose an attack mix that has an expected human word-level recoverability of 65.5% (the average overall annotator accuracy) for each class. The mixes are; phonetic: 63.7% PhoneE, 36.3% Zeroé Phonetic; typo: 78.2% Zeroé Noise, 21.8% Zeroé Typo; visual: 77.4% LEGIT, 22.6% DCES.

We test open models (Mistral-7B, Llama2-7B and -13B) on this dataset in two settings. First, we use the prompts from the Ad-Word experiments to get baseline context-free recovery accuracy for these perturbations. Then, we prompt the model to recover the word using a prompt containing the entire abstract with the perturbed word replacing the original. Both sets of prompts are zero-shot (no perturbation examples) and available in Appendix C.

As expected, Table 3 clearly shows that performance improves in the with-context setting over the context-free setting. However, it is also clear that whether or not a word is recovered successfully without context is strongly predictive of whether it will be recovered with context. We conclude that, for words attacked with Ad-Word's attacks, while context makes recovery easier for models, it does not solve the problem: context-free recovery is highly correlated to with-context recovery.

5.2 Adversarial HOT-Classification

We turn now to the question of context-free word recovery's predictive power of downstream task performance under adversarial perturbations. To do so, we use the HOT Speech dataset (Proprietary License),[11][12] which has 3,841 English social media comments labeled as any mix of hateful, offensive, and toxic, precisely the kind of content where attackers employ evasive perturbation. Of the 3,841 comments, 1,460 are labeled at least one of hateful, offensive, or toxic, with a total of 404 labeled hateful, 862 offensive, and 803 toxic. We filter comments to be less than $2,000$ characters (to fit in the prompt context window) (less 8 comments) and have at least 8 words that

[10] https://www.nltk.org/api/nltk.stem.porter.html

[11] https://socialmediaarchive.org/record/19

[12] https://socialmediaarchive.org/pages/?page=Terms%20of%20Use&ln=en

			Phonetic		*Typo*		*Visual*	
			w/ctx	#	w/ctx	#	w/ctx	#
Llama2	13B	✓	92.2	3,024	93.1	2,643	91.9	2,858
		✗	52.9	1,976	51.8	2,337	49.6	2,142
	7B	✓	82.2	2,742	84.5	2,500	84.1	2,386
		✗	44.5	2,258	42.0	2,500	42.6	2,614
Mstl	7B	✓	91.9	3,067	89.4	2,566	90.4	2,412
		✗	46.7	1,933	34.5	2,434	36.0	2,583

Table 3: **With-Context Word Recovery Performance.** The table shows the with-context word recovery accuracy for words sampled from the paper abstracts as described in §5.1. ✓ rows are words that were successfully recovered without context. ✗ rows are words that were **not** recovered without context. Individual cells in the `w/ctx` column are recovery accuracies **with** context *conditional on* recovery of the word **without** context. Cells in the # columns are number of samples that were or were not recovered without context. For instance, from the first two cells of the first row, Llama2-13B successfully recovered 3,024 of 5,000 phonetically attacked words without context (# column). Of those, 92.2% (2,788) words were also recovered correctly with context (`w/ctx` column). The `w/ctx` cells in the ✗ rows show how much recovery accuracy can be gained from adding context. In our experiment, adding context *at best* fixes about half the errors, but it also introduces some new errors (every ✓-row accuracy is closer to 90% than 100%). In short, while context materially improves word recovery, failure to recover a word without context is strongly predictive of failure to recover with context.

are not usernames (to allow for gradation when perturbing individual words) (less 642 comments) for a final set of 2,830 comments. We use a prompt adapted from Li et al. (2024) to use Chain-of-Thought prompting (Wei et al., 2022). The prompt requests an analysis of the comment against the provided class definition, then a predicted positive class probability (score) in $[0, 1]$ (templates are in Appendix C).

We first prompt each model with unperturbed comments, establishing baseline performance. We then perturb increasing percentages of words (12.5%, 25%, and 50%, excluding usernames and words less than length 3) and measure changes in predictions at these levels. We refer to each combination of a model, HOT-classification task, attack class and perturbation ratio as a *setting*. There are 3 models, 3 classification tasks, 3 attacks, and 3 perturbation ratios, for a total of 81 *attacked* settings, plus 3 baselines, for a total of 84 settings.

Perturbation strategies are applied in the same manner as the with-context experiment. We choose the order in which we perturb the words by word-importance to hateful classification as calculated by a RoBERTa model from Vidgen et al. (2021).[13] For each perturbation percentage, we perturb the same words in each comment for each attack.

Because we are primarily interested in the *relative* effectiveness of the attacks, we analyze the attacks in two ways. First, we use a pairwise difference-in-difference model to measure relative change in positive class predictions (scores) between attacks (i.e., betweeen visual and phonetic attacks). This tells us if the models respond more to visual attacks than other attacks (described below). Second, to see if the responses in scores correlate to changes in task performance, we analyze the model AUC at each perturbation ratio. We report the average AUCs across HOT-classes in Table 4.

The first difference in the pairwise difference-in-difference model is the difference between the baseline model scores on a HOT-class and the scores for that model under each attack. We call this the *perturbation response*. For instance, the difference in scores between Llama2-13B on `toxic` classification using visual attacks at a 50% perturbation ratio and baseline Llama-13B scores on toxic classification. The second difference is the difference

[13] https://huggingface.co/facebook/roberta-hate-speech-dynabench-r4-target

		Attack	0%	12.5%	25%	50%
Llama2	13B	phonetic	0.76	0.74	0.74	0.71
		typo	0.76	0.73	0.72	0.69
		visual	0.76	0.73	0.72	0.68
	7B	phonetic	0.77	0.76	0.74	0.73
		typo	0.77	0.75	0.73	0.71
		visual	0.77	0.75	0.74	0.73
Mistral	7B	phonetic	0.80	0.75	0.74	0.70
		typo	0.80	0.74	0.71	0.67
		visual	0.80	0.73	0.69	0.65

Table 4: **Adversarial HOT Classification Performance.** HOT classification is three different tasks, `hateful`, `offensive` and `toxic` classification. The same comments are used for each task, but the labels depend on the content of the comment. Attacks are applied to to 12.5%, 25%, and 50% of words in each comment (rounded up), respectively. Columns are percentage of words perturbed. AUCs are average AUC for all HOT classification tasks. Prompts do not include examples of perturbed words (zeroshot). Note that average AUCs mask task-level differences for the Llama2 models.

between the perturbation responses for each attack. There are 3 HOT-classes, 3 perturbation ratios, and 2 comparisons (visual-to-phonetic, and visual-to-typo) for a total of 18 per model. Using these comparisons, we performed a one-sided pairwise T-test with the hypothesis that visual attacks result in greater absolute perturbation response than phonetic or typo attacks. We apply a Bonferroni correction to $p = 0.05$ for 18 hypotheses for each model, yielding a significance level of $p < 0.00278$.

For Mistral-7B, all 18 perturbation responses to visual attacks are significantly greater than perturbation responses to phonetic and typo attacks. For Llama 7B, 2 out of 18 is significantly greater - comparing visual to phonetic and typo on offensive classification at 12.5% perturbation. For Llama-13B, 10 out of 18 responses are significantly greater, with 5 out of 6 at the 12.5% (all 3 phonetic, 2 typo), 3 out of 6 at 25% (2 phonetic, 1 typo), and 2 out of 6 at 50% (2 phonetic). Complete statistics are in appendix Table 8. With respect to AUC (Table 4), Mistral is the most affected by the attacks, falling from 0.80 to at best 0.70 AUC on 50% phonetic attacks and at worst 0.65 on 50% visual attacks. Llama2-7B is less affected by all attacks than Llama2-13B. Llama2-13B performance is most impacted by typo and visual attacks.

These attacks are meant to remain recoverable to humans, altering only a small percentage of total characters in the comment. Despite this, of the three models we tested, we observe two (Mistral-7B and Llama2-13B) have a statistically greater response to visual attacks than phonetic or typo attacks. For one of the models, Mistral-7B, this correlates with a relative decrease in task performance at all perturbation ratios.

5.2.1 HOT Classification Shielding

In the HOT Classification setting, Mistral-7B is both the most performant model and most susceptible to perturbation attacks; especially to the visual attacks which it struggled to recover in § 4.3. This raises the question—can we improve the robustness of open-source LLMs aside from increasing their size? Here we show that a simple *shielding* defense built to leverage complementary recovery abilities between models can go a long way.

To do so, we leverage Byt5-base fine-tuned on visual perturbations (where it had a high in-domain accuracy in § 4.1). Our defense consists of running Byt5 recovery on words with non-`ASCII` characters (mostly visual attacks) and replacing these words in the comments with the words Byt5 recovered. The resulting comment is then given to the downstream model - Mistral-7B in this experiment.

We run this experiment for all three HOT-classes with a perturbation ratio of 50%, which was Mistral's worst performing setting we tested; its average AUC dropped from its baseline performance of 0.80 to 0.65. In short, the defense works, restoring the overall performance to 0.77 AUC, which is 80% of the performance lost to the attack.

While more sophisticated shielding defenses have been studied (Keller et al., 2021; Pruthi et al., 2019), this defense relies exclusively on Byt5's perturbation recovery, thereby demonstrating that it is word recovery itself that is improving robustness.

6 Summary

We introduced a new diagnostic task, Adversarial Word Recovery, and a supporting dataset, `Ad-Word`, for assessing language model recovery of perturbed text. Applying `Ad-Word` to the current generation of large language models, we observe that the largest proprietary models already achieve human-like performance, with GPT-4 exceeding human annotators. In this regard, word recovery is yet another example of an emergent capability.

However, a significant gap exists between the proprietary models (GPT-4, GPT-3.5, and Palm2) and open-source models (Llama2, Mistral, Falcon), most prominently on visual attacks. While increasing model size helps, these models are still at least 20% worse than humans in recovering words perturbed with visual attacks. The presence of context does not completely mitigate this vulnerability. Hence, improving the word recovery capabilities of these models is an important area for future work where `Ad-Word` can be useful. In the meantime, we show mixed-model approaches, like shielding with small models like ByT5, can improve robustness by exploiting complementary recovery strengths.

7 Ethics Statement

Intentional text perturbations have been an effective method for escaping content moderation since the early days of the internet (Mitchell, 2005). These kinds of evasions can be used to enable publishing and sharing content online that may be illegal in the users' jurisdiction or disallowed by terms of service. In general, better identification of these evasions through text recovery can lead to more complete and predictable enforcement of relevant

statutes, with the hope of reducing the presence and impact of harmful content.

However, when information control is centralized, these recovery models can aid in identifying and eventually suppressing dissident ideas. As sophistication of model-based text recovery improves, legitimate activists using obfuscation to avoid detection may find it harder to spread their messages. However, in security critical domains, obfuscation via text perturbation is not a replacement for encryption because confidentiality cannot be ensured.

8 Limitations

We know little about the bounds of human word *decoding* (the term used in psycholinguistic literature to describe identification of words), but we do know that humans can compensate for decoding difficulty with extra decoding time (Davis, 2003; Rayner et al., 2006). Our annotation environment did not restrict time taken per perturbation and annotators annotated hundreds of words in immediate succession, which may bias our human baselines in either direction.

The large language model landscape is evolving rapidly. Since the these experiments were conceived and executed, newer versions of some models tested have been released (e.g., the Llama 3 series). Continual benchmarking of new models is beyond the scope of this work, but Ad-Word will be useful as a diagnostic for these newer models as well.

We do not have access to any preprocessing that is done by ChatGPT and Palm2. It is possible that preprocessing contributes significantly to the overall performance of these models, and our tests would not illuminate this effect. Moreover, each of the different model families in our experiments use different tokenizers. Our experiments only control for the role of tokenizers within the same model family. While we know that the differences in model performance within a model family are not the result of tokenization, our work does not provide insights into the impact of tokenization across model families.

The perturbations in Ad-Word are meant to provide enough variety to test the different mechanisms to reverse the visual, phonetic, and typo attacks. However, these examples do not - and could not - represent all possible perturbations within these classes. In particular, the ANTHRO attacks have a larger volume of perturbations for some words from popular discourse than Ad-Word's common English words. For instance, ANTHRO has more than a dozen perturbations for 'republican' and 'democrat'. Some Ad-Word words only have casing changes in ANTHRO (which also retain phonetic similarity). As a result, Ad-Word ANTHRO Phonetic is less diverse than PhoneE or Zeroé Phonetic. In addition, the majority of the attacks in Ad-Word are synthetic. For our experiments, we believe it is necessary to incorporate synthetic attacks to avoid bias from data contamination during model training. However, it may be the case that these LLMs perform better in real word scenarios than on our study's synthetic attacks.

While having a breadth of both attacks and perturbations for the same word in Ad-Word provides a diverse set of conditions for testing recovery, no dataset could cover all perturbations or attacks. Moreover, a general trend of decoding a particular attack well - like decoding phonetic attacks well - does not mean a model must recover all phonetic attacks well. A specifically constructed attack from a given class may still be successful even against a model that is, in general, capable of recovering attacks from that class.

Finally, when testing downstream tasks, attacks may change not only the character representation of the samples but the resulting semantics. It is possible that certain types of attacks have a more significant impact on semantics than others even if they remain recoverable. In our HOT classification experiments, we do not establish new ground truth for every perturbed sentence. Some of the effects we observed may be attributable to genuine ambiguities in semantics, which our experiments would not distinguish.

Acknowledgments

This research was supported by a grant from the Duke Arts & Sciences Council. Thank you to the anonymous reviewers whose feedback improved this manuscript.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak

Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Kevin Atkinson. 2019. Gnu aspell.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Nikolaj Bauer, Moritz Preisig, and Martin Volk. 2024. Offensiveness, hate, emotion and GPT: Benchmarking GPT3.5 and GPT4 as classifiers on Twitter-specific datasets. In *Proceedings of the Fourth Workshop on Threat, Aggression & Cyberbullying @ LREC-COLING-2024*, pages 126–133, Torino, Italia. ELRA and ICCL.

Yonatan Belinkov and Yonatan Bisk. 2017. d. *arXiv preprint arXiv:1711.02173*.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Anne Castles, Kathleen Rastle, and Kate Nation. 2018. Ending the reading wars: Reading acquisition from novice to expert. *Psychological science in the public interest*, 19(1):5–51.

Monojit Choudhury, Markose Thomas, Animesh Mukherjee, Anupam Basu, and Niloy Ganguly. 2007. How difficult is it to develop a perfect spell-checker? a cross-linguistic analysis through complex network approach. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 81–88, Rochester, NY, USA. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Portia Cooper, Mihai Surdeanu, and Eduardo Blanco. 2023. Hiding in plain sight: Tweets with hate speech masked by homoglyphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2922–2929, Singapore. Association for Computational Linguistics.

A Costello. 2003a. Rfc3492: Punycode: A bootstring encoding of unicode for internationalized domain names in applications (idna).

Adam Costello. 2003b. Punycode: A bootstring encoding of unicode for internationalized domain names in applications (idna). Technical report, rfc3492.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515.

Matt Davis. 2003. Aoccdrnig to a rscheearch at cmabrigde uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. the rset can be a toatl mses and you can sitll raed it wouthit porbelm. tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

Antreas Dionysiou and Elias Athanasopoulos. 2021. Unicode evil: Evading nlp systems using visual similarities of text characters. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, pages 1–12.

Steffen Eger and Yannik Benz. 2020. From hero to zéroe: A benchmark of low-level adversarial attacks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 786–803, Suzhou, China. 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.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Tarleton Gillespie. 2020. Content moderation, ai, and the question of scale. *Big Data & Society*, 7(2):2053951720943234.

Nora Hofer, Pascal Schöttle, Alexander Rietzler, and Sebastian Stabinger. 2021. Adversarial examples against a bert absa model–fooling bert with l33t, misspellign, and punctuation. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–6.

Md Saroar Jahan and Mourad Oussalah. 2023. A systematic review of hate speech automatic detection using natural language processing. *Neurocomputing*, 546:126232.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 158–164, Online. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Josh Kaufman. 2012. google-10000-english.

Keshav Kaushik, Sahajpreet Singh, Saksham Garg, Sarthak Singhal, and Shambhavi Pandey. 2021. Exploring the mechanisms of phishing. *Computer Fraud & Security*, 2021(11):14–19.

Yannik Keller, Jan Mackensen, and Steffen Eger. 2021. BERT-defense: A probabilistic model based on BERT to combat cognitively inspired orthographic adversarial attacks. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1616–1629, Online. Association for Computational Linguistics.

Tharindu Kumarage, Amrita Bhattacharjee, and Joshua Garland. 2024. Harnessing artificial intelligence to combat online hate: Exploring the challenges and opportunities of large language models in hate speech detection. *arXiv preprint arXiv:2403.08035*.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2953–2965, Dublin, Ireland. Association for Computational Linguistics.

Thai Le, Yiran Ye, Yifan Hu, and Dongwon Lee. 2023. Cryptext: Database and interactive toolkit of human-written text perturbations in the wild. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 3639–3642. IEEE.

Lingyao Li, Lizhou Fan, Shubham Atreja, and Libby Hemphill. 2024. "hot" chatgpt: The promise of chatgpt in detecting and discriminating hateful, offensive, and toxic comments on social media. *ACM Trans. Web*, 18(2).

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Anthory Mitchell. 2005. A leet primer. Accessed 2022-06-23.

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, Online. Association for Computational Linguistics.

Fabio Poletto, Valerio Basile, Manuela Sanguinetti, Cristina Bosco, and Viviana Patti. 2021. Resources and benchmark corpora for hate speech detection: a systematic review. *Language Resources and Evaluation*, 55:477–523.

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.

Keith Rayner, Sarah J White, Rebecca L Johnson, and Simon P Liversedge. 2006. Raeding wrods with jubmled lettres: there is a cost. *Psychological science*.

Nestor Rodriguez and Sergio Rojas-Galeano. 2018. Shielding google's language toxicity model against adversarial attacks. *arXiv preprint arXiv:1801.01828*.

Paul Röttger, Bertram Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet B Pierrehumbert. 2020. Hatecheck: Functional tests for hate speech detection models. *arXiv preprint arXiv:2012.15606*.

Robert C Russell. 1918. Identification of letters. US Patent US1261167A.

Phillip Rust, Jonas F Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. 2022. Language modelling with pixels. *arXiv preprint arXiv:2207.06991*.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Keerthana Sasidaran and Geetha J. 2024. Multimodal hate speech detection using fine-tuned llama 2 model. In *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, pages 1–6.

Dev Seth, Rickard Stureborg, Danish Pruthi, and Bhuwan Dhingra. 2023. Learning the legibility of visual text perturbations. *arXiv preprint arXiv:2303.05077*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, Online. Association for Computational Linguistics.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, et al. 2023. Are large language models really robust to word-level perturbations? *arXiv preprint arXiv:2309.11166*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Ged Ellis. 2009. Using the Web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899, Singapore. Association for Computational Linguistics.

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.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*.

A Psycholinguistic Basis for `Ad-Word`

In principle, the different attack classes in `Ad-Word` require a different process to recover. Reversing visual attacks requires identifying visually similar characters and replacing them. Decoding phonetic attacks requires identifying graphemes that have similar phonemes. Decoding typo attacks requires identifying how the typo gets created—like a finger slipping to a nearby key on the qwerty keyboard. Here, we provide a theoretical justification for benchmarking recovery with these mechanisms.

We attempted to ground our attack strategies in an understanding of human reading. We think this is one of the key contributions of our work because it highlights that different strategies can reverse different kinds of perturbations. Testing on any one class or strategy does not necessarily imply generalization to others. Our setup allows this to be demonstrated empirically, as we do in the paper.

While the mechanisms of reading are still an active area of research in psycholingistics, there is now high level agreement that there is a bi-modal system for 'word reading' - that is, actually figuring out what word is on the page (as opposed to full reading comprehension). From Castles et al. (2018):

> The important point is that all of the models converge in that they represent two key cognitive processes in word reading: one that involves the translation of a word's spelling into its sound and then to meaning, and one that involves gaining access to meaning directly from the spelling, without the requirement to do so via phonology.

They go on to point out that the 'spelling-to-sound' mechanism is used for decoding novel words, whereas the direct-to-meaning pathway is

used for words with familiar spellings. Importantly, information flow in these models is not unidirectional; the two modalities can influence one another (and be influenced by semantics).

Our attack classes attempt to change words without breaking those mechanisms. The visual attacks add noise to the spelling mechanism at the grapheme level. The typo attacks add noise to the spelling mechanism at the word level. The phonetic attacks add noise to the phonetic mechanism in the first mode. Note that any phonetic attack must necessarily include changes to spelling for a reading task. Also, any attack with sufficient change to the word will obviate the direct mode, which is why inclusion of typo attacks is important. They allow for changes not beholden to integrity of graphemes or phonemes. This is what we mean by 'typo attacks preserve edit distance' above.

Of course, these reading models are theoretical and we could have used any one specific model more directly at the cost of added complexity. We think our 3-part taxonomy strikes a nice balance between generality and complexity for the purpose of benchmarking NLP models.

Within each of these attack classes, we could have chosen other or more attacks. For example, visual attacks that allow 1-to-many or many-to-1 substitutions. What attack strategies are viable (in the sense they allow human decoding) is an empirical question. A full treatment of that topic is an open area of research. However, we believe our current settings have a nice mix of empirically sourced and synthetic attacks.

B Adword Attack Definitions

Phonetic Attacks

For phonetic attacks, word-level perturbations are constructed either through grapheme-level perturbations using similarity between their corresponding phonemes (Zeroé Phonetic and PhoneE), or through extraction from internet corpora (ANTHRO Phonetic).

Strategy 1: PhoneE - ***Phone**tic **E**vasion* works similarly to visual attacks, but entire graphemes are substituted instead of individual characters. A word is first converted into its phonetic representation, then candidate graphemes that represent the same phoneme are sampled, weighted towards graphemes that frequently represent the original phoneme. PhoneE is new to this paper, and is described in detail in subsection 3.4.

Strategy 2: ANTHRO (Phonetically Similar) (Le et al., 2022) - ANTHRO is a database of perturbations meant to replicate or imitate human-generated perturbations sourced from the internet. The perturbations are identified using a combination of phonetic similarity (via a modified version of SOUNDEX (Russell, 1918) called SOUNDEX++) and edit distance. For the ANTHRO Phonetic attack, we produce perturbation candidates using ANTHRO a maximum edit distance 6 and equivalent SOUNDEX++ representations, selecting uniformly from the candidates.

Strategy 3: Zeroé Phonetic (Eger and Benz, 2020) - The Zeroé (Apache 2.0) Phonetic attack is an encoder-decoder model. The encoder transforms a character sequence to a phoneme sequence, and the decoder transforms the phoneme sequence to a (potentially different) character sequence. We make two modifications to the original attack: we use beam search with beam size 10 (instead of greedy decoding) and prune beams that end in the original word.

Typo Attacks

Two of the typo strategies, ANTHRO Typo and Zeroé Typo, are composed largely of perturbations constructed from internet corpora. The Zeroé Noise strategy represents a subset of Zeroé synthetic perturbations that imitate the kinds of typos a human may naturally introduce.

Strategy 1: ANTHRO (Edit Distance) (Le et al., 2022) - ANTHRO Edit Distance is similar to ANTHRO Phonetic except it excludes exact modified SOUNDEX++ matches, which provides candidates that have edit distance similarity but phonetic *dissimilarity*. We also sample the allowed edit distance from a geometric distribution.

Strategy 2: Zeroé-Noise (InnerShuffle, Random Deletion, Intruders (Eger and Benz, 2020) - We group three attack strategies from Zeroé, which we sample from uniformly. InnerShuffle rearranges the letters of the word, but keeps the first and last letter in place. Random deletion randomly deletes a character from the word with probability $p = .1$, deleting at least 1. Intruders interjects randomly selected punctuation characters with $p = .1$ for each pair of characters, adding at least 1.

Strategy 3: Zeroé-Typo (Natural Typos, Keyboard Typos) (Eger and Benz, 2020) - Strategy 2 adds noise to words in a manner that looks similar to human typos. Zeroé includes a dictionary of real typos extracted from the Wikipedia corpus, originally constructed by Belinkov and Bisk (2017). For

words not available in the typo dictionary, we employ the Zeroé Keyboard Typos attack, which uses locality of characters on the QWERTY keyboard to create a nearest neighbor space. We uniformly sample characters to perturb with probability $p = .15$, choosing at least 1.

Visual Attacks

All visual attacks use character-level homoglyph substitutions from a nearest-neighbor space, with different attack strategies using different spaces. For each attack, first, we sample characters to perturb with probability p_c. Then, for each selected character c_i, we sample $k = j + 1, j \sim \text{Geom}(g_k)$, where $\text{Geom}(g_k)$ represents a geometric distribution with parameter g_k, and take the k-th nearest neighbor of c_i in the attack space. To set p_c and g_k, we use the LEGIT dataset released by Seth et al. (2023) which includes an ordinal ranking of legibility for each perturbation as judged by humans, and approximate them by fitting a model to LEGIT's perturbations with positive legibility scores. This results in $p_c = .37$ and $g_k = .05$.

Strategy 1: VIPER DCES (Eger et al., 2019) The *Description-Based Character Embedding Space* (DCES) is constructed by finding characters that have the same case and name in the UNICODE standard description.

Strategy 2: VIPER ICES (Eger et al., 2019) *Image-based Character Embedding Space* (ICES) uses a pixel embedding for each character in a 244×244 space and similarity calculated using cosine distance.

Strategy 3: LEGIT (Seth et al., 2023) - We use the perturbed words in the original LEGIT dataset directly, which was constructed using a combination of 3 different visual embedding spaces. We perturb added words using LEGIT-TrOCR-MT.

C Prompting

C.1 Model Configurations

For all models, max_new_tokens is set to 3 times the length of the perturbed words.

GPT-3.5, GPT-4. Temperature was set to 0 in the request. The prompt is provided as a system prompt. Occasionally, ChatGPT would still respond with a score that expressed a range rather than a single value (ie, $0 - .4$ instead of .2). Those requests were retried.

Palm2. Temperature is set to 0. However, infrequently, Palm2 would not obey the response format. In that case, the temperature was increased by .1 and retried.

Llama2, Falcon, Mistral. Huggingface[14] models are used. Generation is done with beam search with beam size of 3 with sampling disabled.

C.2 Ad-Word Recovery

Below are examples of the prompts provided to the LLMs under tests, with the number of examples and queries truncated from 50 and 20 to 2 and 2, respectively. For zero-shot prompting, we remove the provided examples and replace them with three repetitions of `1. "Perturbed Word" -> "Original Word".`

C.2.1 ChatGPT

```
You are trying to recover the original text from
     words that have been perturbed. The
    Perturbed Words were created by taking the
    Original Word in English and changing some
    of the letters.

Respond with you your best guess about the
    original word that was used to create the
    Perturbed Word. Casing does not matter, but
    if the word has punction (like Amy's or half
    -pipe), please include it. Please do not
    omit any words and do not submit multiple
    answers for the same Perturbed Word. Respond
     just with the Perturbed Word followed by `
    -> ` and then your guess about the Original
    Word.

Below there are examples of some Perturbed Words
     and Original Words labeled `Examples:`.
    Perturbed Words come before `->` and
    Original Words come afterwards.

Examples:

1. "dstridued" -> "distributed"
2. "mtro" -> "metro"

The new words are after `New Perturbed Words:`,
    each followed by `->`. There are 2 words,
    numbered 1 to 2. Please provide 2 responses,
     numbered 1 to 2 in the format

```
1. "Perturbed Word" -> "Original Word".
2. "Perturbed Word" -> "Original Word".
...

```
New Perturbed Words:
1. "h3llo" ->
2. "w0rld" ->

Your first of 1 answers should start with
1. "h3llo" -> "
```

[14] https://huggingface.co/models

C.2.2 Palm2

```
You are trying to recover the original text from
     words that have been perturbed. The
    Perturbed Words were created by taking the
    Original Word in English and changing some
    of the letters.

Respond with you your best guess about the
    original word that was used to create the
    Perturbed Word. Casing does not matter, but
    if the word has punction (like Amy's or half
    -pipe), please include it. Please do not
    omit any words and do not submit multiple
    answers for the same Perturbed Word. Respond
     just with the Perturbed Word followed by `
    -> ` and then your guess about the Original
    Word.
Below there are examples of some Perturbed Words
     and Original Words labeled `Examples:`.
    Perturbed Words come before `->` and
    Original Words come afterwards.

Examples:
1. "dstridued" -> "distributed"
2. "mtro" -> "metro"

The new words are after `New Perturbed Words:`,
    each followed by `->`. There are 2 words,
    numbered 1 to 2.

New Perturbed Words:
0. "h3llo" ->
1. "w0rld" ->

Please provide 2 responses, numbered 1 to 1 in
    the format

```
1. "Perturbed Word" -> "Original Word".
2. "Perturbed Word" -> "Original Word".
...

Your answers should begin with the number (ie,
    1.) and word (ie, "h3llo") in the New
    Perturbed Words list followed by -> (all
    copied exactly from the input). Then, your
    guess for the original word should follow
    the -> in quotes (ie, "word", (not the
    answer for 1.)). ALWAYS INCLUDE AN ORIGINAL
    WORD AFTER THE ->.

You should have answers numbers 1., 2.. Finish
    your response with the word DONE on a single
     line after the 2th Original Word. Like this
    :
```
2. "Perturbed Word" -> "Original Word"
DONE
```
```

### C.2.3 Llama2

```
[INST] <<SYS>>
You are trying to recover the original text from
     words that have been perturbed.
The Perturbed Words were created by taking the
    Original Word in English and changing some
    of the letters.
<</SYS>>

Below there are examples of some Perturbed Words
     and Original Words labeled 'Examples:'.
    Perturbed Words come before '->' and
    Original Words come afterwards.

Respond with you your best guess about the
    original word that was used to create the
    Perturbed Word. Casing does not matter, but
    if the word has punction (like Amy\'s or
    half-pipe), please include it. Please do not
     omit any words and do not submit multiple
    answers for the same Perturbed Word. Respond
     just with the Perturbed Word followed by `
    -> ` and then your guess about the Original
    Word.

Examples:

1. "h3llo" -> "hello"
2. "w0rld" -> "world"

[/INST]

Sure, I'm ready to help! Please provide the new
    perturbed words and I will give you my best
    guesses for the original words in the format
     'n. perturbed_word -> original_word' for
    each word.

[INST]
The new words are after 'New Perturbed Words:',
    each followed by '->'.

New Perturbed Words:
1. haqqy ->
2. berthday ->

[/INST]
Here are my answers for the new perturbed words:

1. "happy" -> "
```

In the zeroshot setting, the Examples clause is replaced with:

```
Below there is an example of the format for
    Perturbed Words and Original Words labeled `
    Format:`. Perturbed Words come before `->`
    and Original Words come afterwards.

Format:
1. "Perturbed" -> "Original"
```

C.2.4 Mistral

The Mistral prompt is the same as the Llama2 prompt, except the «SYS» and «/SYS» markers are removed.

C.2.5 Falcon

Falcon prompt is the same as the Llama2 prompt, except

1. [INST] and <<SYS>> markers are removed
2. >>CONTEXT<< is prepended to the prompt
```

3. >>QUESTION<< is added before New Perturbed Words

4. >>ANSWER<< is added before the starting word.

C.3 HOT Classification

The HOT classification prompt includes instructions to return a rating of $0-1$ for each of `hateful`, `offensive`, and `toxic`. Each class is a separate prompt, with the only change being the name of the class and the definition of the class. The class definitions are:

1. `hateful` - expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group

2. `offensive` - contains hurtful, derogatory, or obscene comments

3. `toxic` - a rude, disrespectful, or unreasonable comment that is likely to make readers want to leave a discussion

Example prompts are given for hateful classification.

Llama2. Prompts for Llama2 are given in two stages using Chain-of-Thought prompting.

First Prompt.

```
[INST]<<SYS>> You are trying to assess whether
    or not a comment is hateful. A comment is `
    hateful' if you perceive that it "expresses
    hatred towards a targeted group or is
    intended to be derogatory, to humiliate, or
    to insult the members of the group".
<</SYS>>

Please analyze this comment and concisely
    describe which, if any, of the criteria from
     the definition of hateful the comment meets
     and why.

Comment:
"thanks."

[/INST]
The main factors in determining whether or not
    this specific comment meets the criteria to
    be considered hateful are:
```

Second Prompt. The second prompt includes the first response and adds:

```
[INST]
Finally, considering the factors that suggest
    the comment is hateful and the factors that
    suggest it is not hateful, provide a score
    on a scale of 0.0, 0.1, 0.2, 0.3, 0.4, 0.5,
    0.6, 0.7, 0.8, 0.9 and 1.0 that the comment
    is hateful.
A score of 1.0 means the comment is certainly
    hateful and a score of 0.0 means it
    certainly is *not* hateful.
[/INST]
SCORE:
```

Mistral. The Mistral prompt is the same as Llama2, except the <<SYS>> and <</SYS>> markers are removed.

C.4 In-Context Word Recovery

For in-context word recovery, we use a single, zero-shot prompt that includes the entire quoted paper abstract, including the perturbed word, and then highlight which word has been perturbed. The prompt is constructed so that the model gives an immediate completion with the correct spelling of the word.

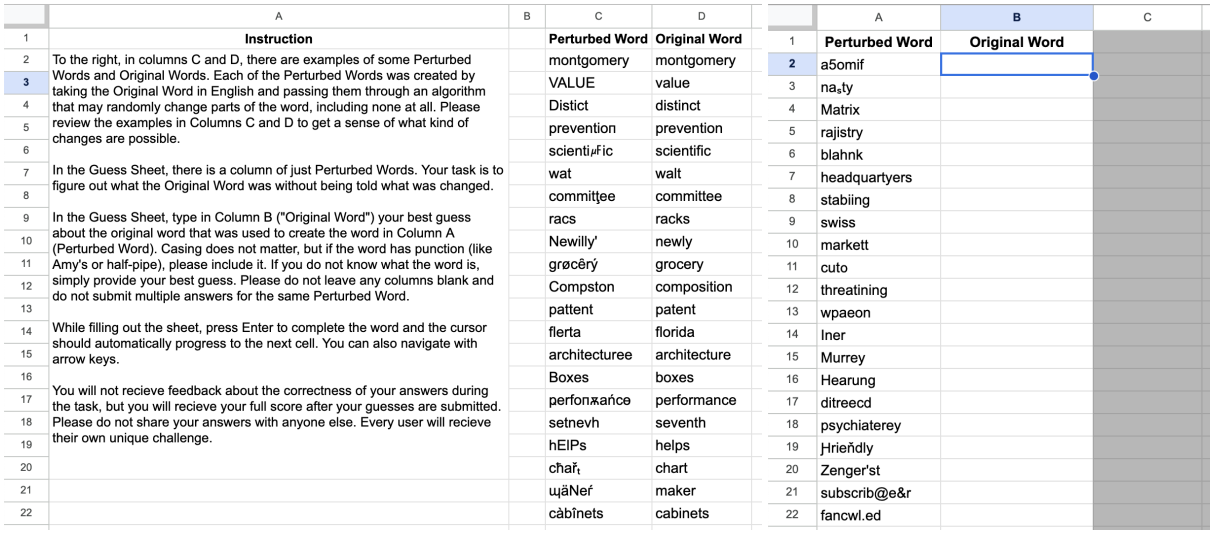
```
[INST]<<SYS>>
You are trying to correct a potential minor
    typographical error in a single word of a
    medical paper's abstract.
<</SYS>>

Please analyze the following abstract and
    determine how to correct this minor error.

The potential minor typographical error is in
    the word "mizbelled" in the following
    abstract:
"This is where the text of the abstract would be.
     It would include a single mizbelled word."

[/INST]
The word "mizbelled" should be written using the
     correct spelling "
```

Mistral. The Mistral prompt is the same as Llama2, except the <<SYS>> and <</SYS>> markers are removed.

D Annotations

Annotators were solicited as volunteers through informal communication with contacts from the author's lab. From our trials of the annotation environment, we expected annotating 500 words to take less than an hour. When soliciting for volunteers, potential volunteers were told the task would take "about an hour". We did not time annotation, as discussed in section 8.

Annotation datasets contain 9 sets of 53 randomly selected words such that word length distribution matches the overall dataset distribution from 4 to 12 characters. We apply each attack to one of the subsets. Each annotator received a different



	A
1	**Instruction**
2	To the right, in columns C and D, there are examples of some Perturbed Words and Original Words. Each of the Perturbed Words was created by taking the Original Word in English and passing them through an algorithm that may randomly change parts of the word, including none at all. Please review the examples in Columns C and D to get a sense of what kind of changes are possible.
3	
4	In the Guess Sheet, there is a column of just Perturbed Words. Your task is to figure out what the Original Word was without being told what was changed.
5	
6	In the Guess Sheet, type in Column B ("Original Word") your best guess about the original word that was used to create the word in Column A (Perturbed Word). Casing does not matter, but if the word has punction (like Amy's or half-pipe), please include it. If you do not know what the word is, simply provide your best guess. Please do not leave any columns blank and do not submit multiple answers for the same Perturbed Word.
7	
8	While filling out the sheet, press Enter to complete the word and the cursor should automatically progress to the next cell. You can also navigate with arrow keys.
9	
10	You will not recieve feedback about the correctness of your answers during the task, but you will recieve your full score after your guesses are submitted. Please do not share your answers with anyone else. Every user will recieve their own unique challenge.

	B	C	D
1		**Perturbed Word**	**Original Word**
2		montgomery	montgomery
3		VALUE	value
4		Distict	distinct
5		prevention	prevention
6		scientiᵢᵣfic	scientific
7		wat	walt
8		committ̰ee	committee
9		racs	racks
10		Newilly'	newly
11		grøcêrý	grocery
12		Compston	composition
13		pattent	patent
14		flerta	florida
15		architecturee	architecture
16		Boxes	boxes
17		perfonɤańce	performance
18		setnevh	seventh
19		hElPs	helps
20		cȟařₜ	chart
21		uɟäNeŕ	maker
22		càbînets	cabinets

	A	B	C
1	**Perturbed Word**	**Original Word**	
2	a5omif		
3	naₛty		
4	Matrix		
5	rajistry		
6	blahnk		
7	headquartyers		
8	stabiing		
9	swiss		
10	markett		
11	cuto		
12	threatining		
13	wpaeon		
14	lner		
15	Murrey		
16	Hearung		
17	ditreecd		
18	psychiaterey		
19	Ɦrieňdly		
20	Zenger'st		
21	subscrib@e&r		
22	fancwl.ed		

Figure 3: **Annotator Interface.** *Left*: The instructions presented to annotators on the first sheet. *Right*: The annotation sheet showing example perturbations with a column to fill out.

subset-strategy combination so that each set has same words with different perturbations.

Finally, we include 23 identity words for a total of 500 words per annotator. Annotator accuracy is 90.2% on the identity set. Annotator accuracies are only comparable to settings where the models are trained/prompted with all attacks since annotators were not told the strategy used.

The annotation environment is a spreadsheet. On the first sheet, annotators were presented with instructions that summarize the task and the environment. Alongside the task definition, 160 perturbations taken from the training sets of all attack strategies (including identity) were provided as examples. Annotators were instructed to "review the examples". The full instructions are on the left-hand size of Figure 3.

Each annotator was provided with their list of 500 perturbed words in a single sheet like the right-hand side of Figure 3.

E Finetuning Byt5

We fine-tune ByT5-base, -large and -XL with the following settings. First, we finetune the model to predict the input word, a setting we call *identity*. In our early experiments we compared this pre-fine-tuning with directly fine-tuning on the perturbed datasets, and saw no material difference in performance the validation set but faster convergence.

When finetuning on perturbed datasets, we use a constant learning rate of 0.0001 and a batch size of 168. but faster convergence For Byt5-large and Byt5-XL, we use gradient accumulation to achieve an effective batch size of 168. We implement early stopping on validation accuracy with a patience of 8 epochs. We tried fine-tuning with a learning rate of 0.00001, 0.00005, and 0.0001 on visual attacks during model development. We observed no difference in accuracy on the validation set (after early stopping), so we we used 0.0001 for all settings.

Training on an NVIDIA A6000, models converge in between 1 and 10 hours, with training time generally increasing in both dataset size and diversity.

F Byt5 Intra-Class (Attack Level) Results

Byt5 and the annotators all exhibit similar relative performance intra-class. For instance, all three find `Zeroé` Phonetic harder than `PhoneE` which is harder than `ANTHRO` Phonetic for phonetic attacks (Table 5). This provides assurance that differences in model performance across strategies is a function of difficulty of the strategy. However, this pattern does not hold for class-level attacks. Byt5 outperforms significantly on visual attacks (possibly by memorizing the character substitutions from its fine-tuning set) (Table 7).

Notably, Byt5's superiority on visual attacks is true of all visual attacks, even though its relative performance on the strategies is the same as the annotators. In-domain, Byt5 achieves 98.0% accuracy on the `DCES` attacks. It does about 20% better than the annotators in-domain on `ICES` and `LEGIT`. Together with the fact that Byt5 does very poorly on all visual attacks when trained on phonetic and typo attacks, this suggests that Byt5 is memorizing

Train Dataset(s)	PhoneE	ANTHRO	Zeroé	All
Baselines				
GNU Aspell	69.9%	84.8%	42.3%	65.8%
Annotators	74.3%	86.3%	50.0%	70.3%
ByT5-Base				
Identity	4.7%	73.8%	8.2%	29.3%
PhoneE	69.6%^{5}	71.7%^{O*}	35.5%^{O}	59.1%
ANTHRO	24.4%^{O}	86.2%^{5}	23%^{O}	45.0%
Zeroé	38.2%^{O}	65.9%	54.4%^{5}	53.0%
PhoneE+ANTHRO	68.0%	82.9%	36.2%^{O*}	62.5%
PhoneE+Zeroé	65.2%	70.1%^{O}	53.4%	63.1%
ANTHRO+Zeroé	42.5%^{O*}	81.7%	52.3%	59.1%
All Phonetic	61.7%	83.6%	52.6%	66.2%^{5}

Table 5: **Byt5 Phonetic Attack Recovery Performance.** Columns show test set accuracies for different strategies of phonetic attacks.

blue^{5} - best accuracy for a Byt5 model. orange^{O} - out of domain test setting. orange^{O*} - best accuracy for an out of domain test setting.

Train Dataset(s)	ANTHRO	Z-Typo	Z-Noise	All
Baselines				
GNU Aspell	38.9%	48.6%	54.2%	47.2%
Annotators	48.6%	53.3%	68.9%	56.9%
ByT5-Base				
Identity	40.3%	53.8%^{5}	39.7%	2.8%
ANTHRO	44.7%^{5}	21.8%^{O}	19.7%^{O}	28.8%
Z-Typo	28.9%^{O}	47.6%	19.1%^{O}	31.8%
Z-Noise	23.2%^{O}	21.7%^{O}	64.6%^{5}	36.4%
ANTHRO+Z-Typo	43.3%	43.3%	20.5%^{O*}	35.6%
ANTHRO+Z-Noise	42.3%	25.5%^{O*}	55.8%	41.2%
Z-Typo+Z-Noise	30.4%^{O*}	45.5%	60.4%	45.5%^{5}
All Typo	40.2%	40.7%	55.6%	45.5%^{5}

Table 6: **Byt5 Typo Attack Recovery Performance.** Columns show test set accuracies for different strategies of phonetic attacks. **Z-Typo** and **Z-Noise** refer to Zeroé-Typo and Zeroé-Noise, respectively.

blue^{5} - best accuracy for a Byt5 model. orange^{O} - out of domain test setting. orange^{O*} - best accuracy for an out of domain test setting.

the homoglyphs in the visual attacks. However, it is not only executing homoglyph replacement; when Byt5 is wrong about a visual attack, it predicts a different English word 22.5% of the time. It is, in effect, combining decoding with a fuzzy dictionary.

We are interested in understanding the degree to which this attack level fine-tuning procedure leads to generalization to attacks unseen during training, so we repeated the class-level In-domain and Out-of-Domain tests from the body of the paper on Byt5 within each attack class. For instance, we train on Zeroé Typo and test on Zeroé Noise. We do this for all combinations of attacks intraclass. See Table 7, Table 6, Table 5.

In this setting, Byt5 is consistent in its lack of generalization. In-domain Byt5 always outperforms out-of-domain, both for single strategy or class and multi-strategy or class training. When only out-of-domain training is available, training Byt5 on multiple out-of-domain strategies within class outperforms training on a single out-of-domain strategy, with two exceptions having similar performance (Train: PhoneE, Test: ANTHRO and Train: LEGIT, Test: ICES). However, at the class level, adding more out-of-domain data is not necessarily advantageous: adding visual attacks to typo attacks drops phonetic recovery accuracy by 5.3%.

Overall, Byt5's performance is promising: 57.7% vs. 63.6% for annotators on our attacks,

Train Dataset(s)	DCES	ICES	LEGIT	All
Baselines				
GNU Aspell	52.7%	26.6%	34.6%	37.9%
Annotators	83.0%	47.2%	60.4%	63.5%
ByT5-Base				
Identity	1.3%	11.7%	6.7%	6.8%
DCES	98.0%^{5}	12.4%^{O}	22.5%^{O}	44.3%
ICES	52.0%^{O}	67.9%^{5}	36.2%^{O}	52.0%
LEGIT	62.7%^{O}	40.6%^{O*}	81.3%^{5}	61.5%
DCES+ICES	95.6%	67.2%	36.5%^{O*}	66.4%
DCES+LEGIT	96.8%	39.1%^{O}	81.0%	72.3%
ICES+LEGIT	64.6%^{O*}	67.6%	77.0%	69.7%
All Visual	93.9%	66.8%	76.4%	78.9%^{5}

Table 7: **Byt5 Visual Attack Recovery Performance.** Columns show test set accuracies for different strategies of visual attacks. For ChatGPT, the prompt includes examples from all the visual attacks.

blue^{5} - best accuracy for a Byt5 model. orange^{O} - out of domain test setting. orange^{O*} - best accuracy for an out of domain test setting.

especially given byt5-base only has 580 million parameters. However, it only achieves that performance when trained on the attacks in `Ad-Word`. Solving this generalization issue, intra-class and inter-class, is an important step to using Byt5 as a lightweight recovery model.

G Hot Statistics

Complete statistical calculations for every hypothesis considered in our difference-in-difference analysis of HOT-classification perturbation response is in Table 8. The test is a one-side T-test with the hypothesis that the perturbation response (see Experiments with Context) to visual attacks is greater than the perturbation response to typo or phonetic attacks. We perform this test for each HOT-class (`hateful`, `offensive`, `toxic`) at each perturbation ratio 12.5%, 25%, and 50%.

Model	Size	Ratio	Baseline	HOT Class	Diff-in-Diff	T-stat	P-value	Sig. 0.05	Sig. Model	Sig. All
Mistral	7B	0.125	phonetic	hateful	-0.04866	-12.01	9.354e-33	True	True	True
				offensive	-0.08806	-17.96	1.030e-68	True	True	True
				toxic	-0.05989	-13.21	5.567e-39	True	True	True
			typo	hateful	-0.02855	-7.17	4.769e-13	True	True	True
				offensive	-0.05035	-10.66	2.325e-26	True	True	True
				toxic	-0.03449	-7.638	1.502e-14	True	True	True
		0.25	phonetic	hateful	-0.04756	-11.24	5.011e-29	True	True	True
				offensive	-0.08565	-17.63	2.187e-66	True	True	True
				toxic	-0.06965	-15.03	1.793e-49	True	True	True
			typo	hateful	-0.02781	-6.704	1.215e-11	True	True	True
				offensive	-0.03721	-8.254	1.160e-16	True	True	True
				toxic	-0.03929	-9.2	3.367e-2	True	True	True
		0.5	phonetic	hateful	-0.04788	-11.04	4.727e-28	True	True	True
				offensive	-0.07102	-15.77	4.726e-54	True	True	True
				toxic	-0.05431	-12.93	1.679e-37	True	True	True
			typo	hateful	-0.02385	-5.679	7.476e-09	True	True	True
				offensive	-0.02141	-5.149	1.396e-07	True	True	True
				toxic	-0.02102	-5.439	2.904e-08	True	True	True
Llama2	7B	0.125	phonetic	hateful	-0.007809	-2.714	3.343e-03	True	False	False
				offensive	-0.007986	-3.271	5.422e-04	True	True	True
				toxic	-0.007032	-2.508	6.104e-03	True	False	False
			typo	hateful	-0.007597	-2.704	3.448e-03	True	False	False
				offensive	-0.007208	-3.035	1.215e-03	True	True	False
				toxic	-0.002191	-0.799	2.122e-01	False	False	False
		0.25	phonetic	hateful	0.000177	0.05522	5.220e-01	False	False	False
				offensive	-0.004629	-1.705	4.418e-02	True	False	False
				toxic	-0.007032	-2.224	1.310e-02	True	False	False
			typo	hateful	0.001731	0.5486	7.083e-01	False	False	False
				offensive	0.003251	1.215	8.878e-01	False	False	False
				toxic	-0.00477	-1.582	5.692e-02	False	False	False
		0.5	phonetic	hateful	0.01198	3.376	9.996e-01	False	False	False
				offensive	0.003852	1.275	8.989e-01	False	False	False
				toxic	0.004912	1.443	9.255e-01	False	False	False
			typo	hateful	0.01099	3.23	9.994e-01	False	False	False
				offensive	0.006113	2.124	9.831e-01	False	False	False
				toxic	0.01134	3.475	9.997e-01	False	False	False
	13B	0.125	phonetic	hateful	-0.01929	-6.398	9.192e-11	True	True	True
				offensive	-0.008057	-3.826	6.648e-05	True	True	True
				toxic	-0.02848	-9.724	2.632e-22	True	True	True
			typo	hateful	-0.01138	-3.772	8.268e-05	True	True	True
				offensive	0.001166	0.55	7.088e-01	False	False	False
				toxic	-0.00841	-2.782	2.716e-03	True	True	False
		0.25	phonetic	hateful	-0.0164	-5.077	2.042e-07	True	True	True
				offensive	-0.005618	-2.582	4.933e-03	True	False	False
				toxic	-0.03604	-10.73	1.211e-26	True	True	True
			typo	hateful	-0.002261	-0.6944	2.438e-01	False	False	False
				offensive	0.006502	3.068	9.989e-01	False	False	False
				toxic	-0.01018	-3.034	1.216e-03	True	True	False
		0.5	phonetic	hateful	-0.002014	-0.5534	2.900e-01	False	False	False
				offensive	-0.008198	-3.473	2.613e-04	True	True	True
				toxic	-0.05074	-13.64	2.381e-41	True	True	True
			typo	hateful	0.00947	2.644	9.959e-01	False	False	False
				offensive	0.002792	1.17	8.790e-01	False	False	False
				toxic	-0.141	-0.03693	4.853e-01	False	False	False

Table 8: **Adversarial Hot Classification Statistical Tests.** Tests are a pairwise one-side T-test that visual attacks cause a greater change in predicted score that phonetic or typo attacks. Phonetic and typo are the *Baselines* in the table, and one test compares to each of them. *Diff-in-Diff* is the mean difference between scores in the visual and non-visual setting. Here, we apply two different Bonferroni Corrections. *Sig 0.05* is significant a $p < 0.05$. *Sig Model* is significant applying Bonferroni Correction to $p < 0.05$ to all hypotheses for one model ($n = 18$), with effective $p < 0.00278$, which is reported in the paper body. *Sig All* is significant applying Bonferroni Correction to all hypotheses for all models ($n = 54$), with effective $p < 0.000926$