Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems

Steffen Eger[†‡], **Gözde Gül Şahin**[†‡], **Andreas Rücklé**[†], **Ji-Ung Lee**[†‡], **Claudia Schulz**[†‡], **Mohsen Mesgar**[†‡], **Krishnkant Swarnkar**[†], **Edwin Simpson**[†], **Iryna Gurevych**[†‡]

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
[‡]Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
[†]www.ukp.tu-darmstadt.de
[‡]www.aiphes.tu-darmstadt.de

Abstract

Visual modifications to text are often used to obfuscate offensive comments in social media (e.g., "!d10t") or as a writing style ("1337" in "leet speak"), among other scenarios. We consider this as a new type of adversarial attack in NLP, a setting to which humans are very robust, as our experiments with both simple and more difficult visual perturbations demonstrate. We investigate the impact of visual adversarial attacks on current NLP systems on character-, word-, and sentence-level tasks, showing that both neural and non-neural models are, in contrast to humans, extremely sensitive to such attacks, suffering performance decreases of up to 82%. We then explore three shielding methods—visual character embeddings, adversarial training, and rule-based recovery—which substantially improve the robustness of the models. However, the shielding methods still fall behind performances achieved in non-attack scenarios, which demonstrates the difficulty of dealing with visual attacks.

1 Introduction

For humans, visual similarity can play a decisive role for assessing the meaning of characters. Some evidence for these are: the frequent swapping of similar looking characters in Internet slang or abusive comments, creative trademark logos, and attack scenarios such as domain name spoofing (see examples in Table 1).

Internet slang writing style	n00b, w!k!p3d!4, 1337
Toxic comments	pi§§, ṇiggers, ḟucking
Trademark logos/artwork	SΛMSUNG
Domain name spoofing	http://wíkipedia.org

Table 1: Examples of text in which characters have been changed to visually similar ones.

Recently, some NLP systems have exploited visual features to capture visual relationships among characters in compositional writing systems such as Chinese or Korean (Liu et al., 2017). However, in more general cases, current neural NLP systems have no built-in notion of visual character similarity. Rather, they either treat characters as discrete units forming a word or they represent characters by randomly initialized embeddings and update them during training—typically in order to generate a character-based word representation that is robust to morphological variation or spelling mistakes (Ma and Hovy, 2016). Intriguingly, this marked distinction between human and machine processing can be exploited as a blind spot of NLP systems. For example, spammers might send malicious emails or post toxic comments to online discussion forums (Hosseini et al., 2017) by visually 'perturbing' the input text in such a way that it is still easily recoverable by humans.

The issue of exposing and addressing the weaknesses of deep learning models to *adversarial inputs*, i.e., perturbed versions of original input samples, has recently received considerable attention. For instance, Goodfellow et al. (2015) showed that small perturbations in the pixels of an image can mislead a neural classifier to predict an incorrect label for the image. In NLP, Jia and Liang (2017) inserted grammatically correct but semantically irrelevant paragraphs to stories to fool neural reading comprehension models. Singh et al. (2018) showed significant drops in the performance of neural models for question answering when using simple paraphrases of the original questions.

Unlike previous NLP attack scenarios, visual attacks, i.e., the exchange of characters in the input with visually similar alternatives, have the following 'advantages': 1) They do not require *any* linguistic knowledge beyond the character level, making the attacks straightforwardly applicable across languages, domains, and tasks. 2) They are allegedly less damaging to human perception and understanding than, e.g., syntax errors or the insertion of negations (Hosseini et al., 2017). 3) They do not require knowledge of the attacked model's parameters or loss function (Ebrahimi et al., 2018).

In this work, we investigate to what extent recent state-of-the-art (SOTA) deep learning models are sensitive to visual attacks and explore various shielding techniques. Our contributions are:

- We introduce VIPER, a Visual Perturber that randomly replaces characters in the input with their visual nearest neighbors in a visual embedding space.
- We show that the performance of SOTA deep learning models substantially drops for various NLP tasks when attacked by VIPER. On individual tasks (e.g., Chunking) and attack scenarios, our observed drops are up to 82%.
- We show that, in contrast to NLP systems, humans are only mildly or not at all affected by visual perturbations.
- We explore three methods to shield from visual attacks, *viz.*, visual character embeddings, adversarial training (Goodfellow et al., 2015), and rule-based recovery. We quantify to which degree and in which circumstances these are helpful.

We point out that integrating visual knowledge with deep learning systems, as our visual character embeddings do, aims to make NLP models behave more like humans by taking cues directly from sensory information such as vision.[1]

2 Related Work

Our work connects to two strands of literature: adversarial attacks and visually informed character embeddings.

Adversarial Attacks are modifications to a classifier's input, that are designed to fool the system into making an incorrect decision, while the original meaning is still understood by a human observer. Different forms of attacks have been studied in NLP and computer vision (CV), including at a character, syntactic, semantic and, in CV, the visual level. Ebrahimi et al. (2018) propose a character flipping algorithm to generate adversarial examples and use it to trick a character-level neural classifier. They show that the accuracy decreases significantly after a few manipulations if certain characters are swapped. Their character flipping approach requires very strong knowledge in the form of the attacked networks' gradients in a so-called white box attack setup. Chen et al. (2018) find that reading comprehension systems often ignore important question terms, thus giving incorrect answers when these terms are replaced. Belinkov and Bisk (2018) show that neural machine translation systems break for all kinds of noise to which humans are robust, such as reordering characters in words, keyboard typos and spelling mistakes. Alzantot et al. (2018) replace words by synonyms to fool text classifiers. Iyyer et al. (2018) reorder sentences syntactically to generate adversarial examples.

In contrast to those related works which perform attacks on the character level, our attacks allow perturbation of any character in a word while potentially minimizing impairment for humans. For example, the strongest attack in Belinkov and Bisk (2018) is random shuffling of all characters, which is much more difficult to restore for humans.

To cope with adversarial attacks, *adversarial training* (Goodfellow et al., 2015) has been proposed as a standard remedy in which training data is augmented with data that is similar to the data used to attack the neural classifiers. Rodriguez and Rojas-Galeano (2018) propose simple rule-based corrections to address a limited number of attacks, including obfuscation (e.g., "idiots" to "!d10ts") and negation (e.g., "idiots" to "NOT idiots"). Most other approaches have been explored in the context of CV, such as adding a stability objective during training (Zheng et al., 2016) and distillation (Papernot et al., 2016). However, methods to increase the robustness in CV have been shown to be less effective against more sophisticated attacks (Carlini and Wagner, 2017).

Visual Character Embeddings were originally proposed to address large character vocabularies in 'compositional' languages like Chinese and Japanese. Shimada et al. (2016) and Dai and Cai (2017) employ a convolutional autoencoder to generate image-based character embeddings (ICE) for Japanese and Chinese text and show improvement on author and publisher identification tasks. Similarly, Liu et al. (2017) create ICEs from a CNN and show that ICEs carry more semantic content and are more suitable for rare characters. However,

[1]Code and data available from `https://github.com/UKPLab/naacl2019-like-humans-visual-attacks`

existing work on visual character embeddings has not used visual information to attack NLP systems or to them.

3 Approach

To investigate the effects of visual attacks and propose methods for shielding, we introduce 1) a visual text perturber, 2) three character embedding spaces, and 3) methods for obtaining word embeddings from character embeddings, used as input representations in some of our experiments.

3.1 Text perturbations

Our visual perturber VIPER disturbs an input text in such a way that (ideally) it is still readable by humans but causes NLP systems to fail blatantly. We parametrize VIPER by a probability p and a character embedding space, CES:[2] For each character c in the input text a flip decision is made (i.i.d. Bernoulli distributed with probability p), and if a replacement takes place, one of up to 20 nearest neighbors in the CES is chosen.[3] Thus, we denote VIPER as taking two arguments:

$$\mathsf{VIPER} = \mathsf{VIPER}(p, \text{CES}).$$

Note that VIPER is a black-box attacker as it does not require any knowledge of the attacked system. It would also be possible to design a more intelligent perturber that only disturbs content words (or "hot" words), similar to Ebrahimi et al. (2018), but this would increase the difficulty for realizing VIPER as a black-box attacker because different types of hot words may be relevant for different tasks.

3.2 Character Embeddings

We consider three different character embedding spaces. The first is continuous, assigning each character a dense 576 dimensional representation, which allows, e.g., for computing cosine similarities between any two characters as well as nearest neighbors for each input character. The other two are discrete and merely used as arguments to VIPER. Thus, they are only required to specify nearest neighbors for standard input characters. For them, each character c in a selected range (e.g., standard English alphabet a-zA-Z) is assigned a set of nearest neighbors, and all nearest neighbors are equidistant to c. All three CES carry visual information, i.e., nearest neighbors are visually similar to the character in question. For practical reasons, we limit all our perturbations to the first 30k Unicode characters throughout.

Image-based character embedding space (ICES) provides a continuous image-based character embedding (ICE) for each Unicode character. We retrieve a 24×24 image representation of the character (using Python's PIL library), then stack the rows of this matrix (with entries between 0 and 255) to form a $24 \cdot 24 = 576$ dimensional embedding vector.

Description-based character embedding space (DCES) is based on the textual descriptions of Unicode characters. We first obtain descriptions of each character from the Unicode 11.0.0 final names list (e.g., LATIN SMALL LETTER A for the character 'a'). Then we determine a set of nearest neighbors by choosing all characters whose descriptions refer to the same letter in the same case, e.g., an alternative to LATIN SMALL LETTER A is LATIN SMALL LETTER A WITH GRAVE as it contains the keywords SMALL and A.

Easy character embedding space (ECES) provides manually selected simple visual perturbations. It contains exactly one nearest neighbor for each of the 52 characters a-zA-Z, chosen as a diacritic below or above a character, such as $\hat{c}$ for the character c.

Differences between the CESs The three embedding spaces play different roles in our experiments. We use ICES as character representations in deep learning systems. DCES and ECES are used as input to VIPER to perturb our test data.[4] ECES models a 'minimal perturbance with maximal impact' scenario: we assume that ECES perturbations do not or only minimally affect human perception but may still have a large impact upon NLP systems. Indeed, we could have chosen an even simpler embedding space, e.g., by considering visually identical characters in different alphabets, such as the Cyrillic 'a' (Unicode 1072) for a Latin 'a' (Unicode 97). DCES is a more difficult

[2]CES may be any 'embedding space' that can be used to identify the nearest neighbors of characters.

[3]The probability of choosing one of the 20 neighbors of c is proportional to its distance to c.

[4]We do not attack with ICES because we also shield with ICES and this would be a (very unrealistic) white box defense scenario. Besides, ICES is also more difficult to restore for humans (see below), making it less desirable for an attacker.

test-bed designed for evaluating our approaches under more realistic conditions with more varied and stronger attacks.

Table 2 exemplifies the differences between ICES, DCES, and ECES by comparing the nearest neighbors of a given character. As expected, ICES contains neighbors of characters which are merely visually similar without representing the same underlying character (such as Λ as a neighbor of A, or l as a neighbor of i). In contrast, DCES sometimes has neighbors with considerable visual dissimilarity to the original character such as Cyrillic small letter i (и) which rather resembles a mirror-inverted n. The overlap between ICES and DCES is modest: out of 20 neighbors, a character has on average only four to five common neighbors in ICES and DCES.

3.3 Word Embeddings

Most neural NLP architectures encode text either on a character or word level. For the latter, word embeddings are needed. In this work, we use the ELMo architecture (Peters et al., 2018) to obtain (contextualized) word embeddings based on characters, i.e., there exists no fixed vocabulary and there will be no (word-level) out-of-vocabulary issues due to perturbation. In the following, we outline our ELMo variant and a visual extension that includes visual signals from the input characters.

SELMo: ELMo as proposed by Peters et al. (2018) first retrieves embeddings for every character in the input, which are learned as part of the network. ELMo then infers non-contextualized word embeddings by applying CNNs over all character embeddings in a word. Two layers of a deep bidirectional language model further process the word embeddings in their local sentential context and output contextualized word embeddings.

We slightly extend ELMo to include character embeddings for the first 30k Unicode characters (instead of the default 256). We call this variant SELMo ("Standard ELMo"). It is worth pointing out that the learned character embeddings of SELMo carry almost no visual information, as illustrated in Table 2. That is, except for a few very standard cases, nearest neighbors of characters do not visually resemble the orginal characters, even when trained on the 1 billion word benchmark (Chelba et al., 2013).[5]

VELMo: To obtain a visually informed variant of ELMo, we replace learned character embeddings with the ICEs and keep the character embeddings fixed during training. This means that during training, the ELMo model learns to utilize visual features of the input, thus potentially being more robust against visual attacks. We call this variant VELMo ("Visually-informed ELMo").

To keep training times of SELMo and VELMo feasible, we use an output dimensionality of 512 instead of the original ELMo's 1024d output. Our detailed hyperparameter setup is given in §A.1.

4 Human annotation experiment

We asked 6 human annotators, university employees and students with native or near-native English language skills, to recover the original underlying English sentences given some perturbed text (data taken from the POS tagging and Chunking tasks, see Table 4). We considered different conditions:

(i) `clean`: $\mathsf{VIPER}(0,\_)$, i.e., no perturbation;
(ii) $\mathsf{VIPER}(p,\text{ICES})$ for $p = 0.2, 0.4, 0.6, 0.8$;
(iii) $\mathsf{VIPER}(p,\text{DCES})$ for $p = 0.2, 0.4, 0.6, 0.8$;
(iv) `easy`: $\mathsf{VIPER}(p,\text{ECES})$ for $p = 0.4, 0.8$.

For each condition, we used 60-120 sentences, where at most 20 sentences of one condition were given to an annotator. Examples of selected conditions are shown in Table 3. Our rationale for including this recovery task is to test robustness of human perception under (our) visual perturbations. We focus on recovery instead of an extrinsic task such as POS because the latter would have required expert/trained annotators.

We evaluate by measuring the normalized edit distance between the recovered sentence and the underlying original, averaged over all sequence pairs and all human annotators. We normalize by the maximum lengths of the two sequences. In our case, this metric can be interpreted as the fraction of characters that have been, on average, wrongly recovered by human annotators. We refer to the metric as "error rate".

Results are shown in Figure 1. In `easy`, there is almost no difference between perturbation levels $p = 0.4$ and $p = 0.8$, so we merge the two conditions.

Humans make copy mistakes even when the input is not perturbed, as evidenced by a positive

[5]We believe SELMo nearest neighbors are more likely to be Chinese/Japanese/Korean (CJK) characters because these nearest neighbors are largely random and there are far more CJK characters in our subset of Unicode.

Input	**ICES**	**DCES**	**ECES**	**SELMo**
e	e ẹ ė ȩ ϵ ē ę ḛ ë ë	ĕ ɝ ə ɜ ē ě ȩ ɛ è ë	ê	é 趽 ㅇ 徛 ꞇ ꞓ ç ↙ ㈎ 坪
i	i ı ị I l l ḷ l į ḷ	ĭ ȉ ȉ i ī ǐ ɪ ȋ í ĩ	î	í ⁝ 嘣 檯 爗 篙 炾 媭 腥 > 窸
A	Α Α Ạ Aι Ḁ Ἀ Ἁ Ἀι Λ Λ	Ā Ă Ǟ A Å Ǎ Ȁ A̸ Á Ä	Â	楞 溷 摮 哋 8. 敁 漸 ̨ 唋 用

Table 2: Ten nearest neighbors in our different character spaces. ‘SELMo’ refers to the nearest neighbors of the trained character embeddings in SELMo.

Condition	**Sentences** (Perturbed / Original)
easy-0.8	Ḿř. Ĉôḟfêê iŝ â ṕřôfêŝsoř âẗ Ĉôḻŭḿb̂îâ Ĺâẘ Ŝcĥôôḻ . Mr. Coffee is a professor at Columbia Law School .
ICES-0.6	Tṇẽ sẖuŧḋown a | fëçț̂ə Э,0ð0 wòřkếrs ḁng ᴎllḷ ҽũt óũṯpuƫ ɓỳ apouț 4ˏ3Ẓ0 câṛṣ : The shutdown affects 3,000 workers and will cut output by about 4,320 cars .
DCES-0.8	Tʰe śţōcƙ ɽeco(v)ɝʶèd ṡøɯəẅẖât ᵗö f̧îṇìṣḣ 1 1/4 ˡòwɇʶ áf 26 1/4 . The stock recovered somewhat to finish 1 1/4 lower at 26 1/4 .

Table 3: Examples of perturbed sentences and underlying originals.

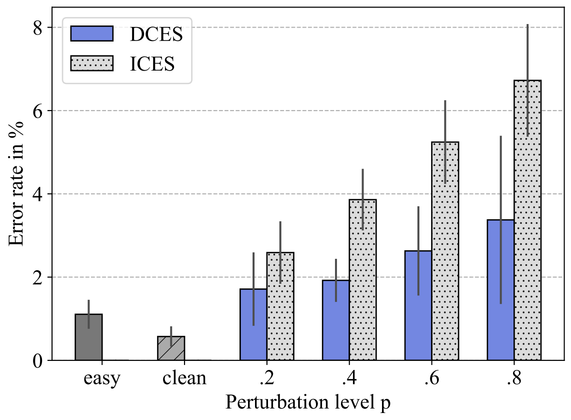

Figure 1: Human annotation experiment. Error bars indicate std. across annotators. For `easy`, we merge the cases $p = 0.4/0.8$.

error rate in `clean`. Such mistakes are typically misspellings or the wrong type of quotation marks (” vs. “). We observe a slightly higher error rate in `easy` than in `clean`. However, on average 75% of all sentences are (exactly) correctly recovered in `easy` while this number is lower (72.5%) in `clean`. By chance, `clean` contains fewer sentences with quotation marks than `easy`, for which a copy mistake was more likely. This may explain `easy`’s higher error rate.

As we increase the perturbation level, the error rate increases consistently for DCES/ICES. It is noteworthy that DCES perturbations are easier to parse for humans than ICES perturbations. We think this is because DCES perturbations always retain a variant of the same character, while ICES may also disturb one character to another character (such as *h* to *b*). Another explanation is that ICES, unlike DCES and ECES, also disturbs numbers and punctuation. Numbers, especially, are more difficult to recover. However, even at 80% disturbance level, humans can, on average, correctly recover at least 93% of all characters in the input text in all conditions.

In summary, humans appear very good at understanding visual perturbations, and are almost perfectly robust to the easy perturbations of ECES. Since adversarial attacks should have minimal impact on humans (Szegedy et al., 2014), the good performance of humans especially on ECES and DCES makes these two spaces ideal candidates for attacks on NLP systems.

5 Computational Experiments

We now evaluate the capabilities of SOTA neural network models to deal with visual attacks in four extrinsic evaluation tasks described in §5.1 and illustrated in Table 4. Hyperparameters of all our models are given in §A.2. We first examine the robustness of all architectures to visual perturbations in §5.2 and then evaluate different shielding approaches in §5.3.

5.1 Tasks

G2P: As our first task, we consider the character-level task of grapheme-to-phoneme (G2P) conversion. It consists of transcribing a character input stream into a phonetic representation. As our dataset, we choose the Combilex pronunciation dataset of American English (Richmond et al.,

Task	**Task Type**	**Input**	**Target / Label(s)**	**Train/Dev/Test**
G2P	char-lvl	ẉṙẹ̰ŧchęđlȳ	r E < @ d 5 i	5K/1K/1K
POS	word-lvl	… exᴛểnḑíng itŝ contraᵋẗ …	… VBG PRP NN …	212K/44K/47K
Chunking	word-lvl	… exᴛểnḑíng itŝ contraᵋẗ …	… B-VP B-NP I-NP …	212K/44K/47K
Toxic Comments	sent-lvl	Ḟŭĉǩ ôḟḟ , yoǔ âňtî - ŝeḿîẗîĉ ĉǔňẗ .	toxic, obscene, insult	149K/10K/64K

Table 4: NLP tasks considered in this work, along with (perturbed) examples and data split statistics.

2009). We frame G2P as a sequence tagging task. To do so, we first hard-align input and output sequences using a 1-0,1-1,1-2 alignment scheme (Schnober et al., 2016) in which an input character is matched with zero, one, or two output characters. Once this preprocessing is done, input and output sequences have equal lengths and we can apply a standard BiLSTM on character-level to the aligned sequences (Reimers and Gurevych, 2017).

POS & Chunking: We consider two word-level tasks. POS tagging associates each token with its corresponding word class (e.g., *noun, adjective, verb*). Chunking groups words into syntactic chunks such as noun and verb phrases (NP and VP), assigning a unique tag to each word, which encodes the position and type of the syntactic constituent, e.g., begin-noun-phrase (B-NP). We use the training, dev and test splits provided by the CoNLL-2000 shared task (Sang and Buchholz, 2000) and use the same BiLSTM architecture as above with SELMo/VELMo embeddings.

Toxic comment (TC) classification: A very realistic use case for adversarial attacks is the toxic comment classification task. One could easily think of a scenario where a person with malicious intent explicitly aims to fool automated methods for detecting toxic comments or insults by obfuscating text with non-standard characters that are still human-readable. We conduct experiments on the TC classification task provided by Kaggle.[6] It is a multi-label sentence classification task with six classes, i.e., *toxic*, *severe toxic*, *obscene*, *threat*, *insult*, *identity hate*. We use average SELMo/VELMo embeddings as input to an MLP.

5.2 VIPER attacks

In Figure 2, we plot how various SOTA systems degrade as we perturb the test data using DCES. We do not only include our own systems, but also existing SOTA models: Marmot (Müller et al., 2013) and Stanford POS tagger (SPT) (Manning et al., 2014). Marmot is a feature-based POS tagger and trained on our data splits. SPT is a bi-directional dependency network tagger that mostly employs lexical features. For SPT, we used the pretrained English model provided by the toolkit. Further, we include a FastText TC classifier which has achieved SOTA performance.[7] We additionally experiment with word level dependency embeddings for POS tagging and TC classification (Komninos and Manandhar, 2016).

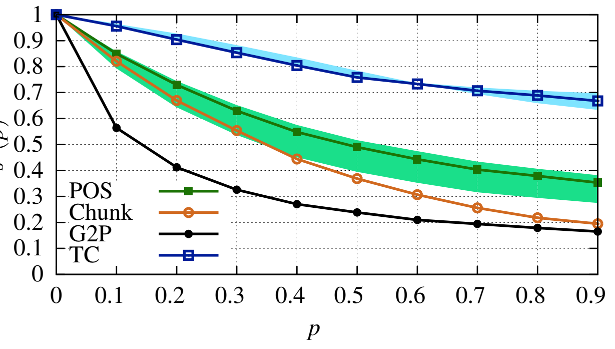

Figure 2: Degradation of SOTA systems for different perturbation levels when attacked by VIPER(p,DCES). The colored regions show how the performance of other SOTA systems relate to ours (i.e., they all suffer from similar degradation).

To compare the performance of different tasks, Figure 2 shows scores computed by:

$$s^*(p) = \frac{s(p)}{s(0)},$$

where p is the perturbation level and $s(p)$ is the score for each task at p, measured in edit distance for G2P, accuracy for POS tagging, micro-F1 for chunking, and AUCROC for TC classification. We invert the scores g of G2P by $1/g$ since lower scores are better for edit distance. Thus, $s^*(0)$ is always 1 and $s^*(p)$ is the relative performance compared to the clean case of no perturbations.

We see that all systems degrade considerably. For example, all three POS taggers have a performance of below 60% of the clean score when 40%

[6]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/

[7]https://www.kaggle.com/yekenot/pooled-gru-fasttext

of the input characters are disturbed. Chunking degrades even more strongly, and G2P has the highest drop: 10% perturbation level causes a 40% performance deterioration. This may be because G2P is a character-level task and the perturbation of a single character is analogous to perturbing a complete word in the word-level tasks. Finally, TC classification degrades least, i.e., only at $p = 0.9$ do we see a degradation of 30% relative to the clean score. These results appear to suggest that character-level tasks suffer the most from our VIPER attacks and sentence-level tasks the least. However, it is worthwhile pointing out that lower-bounds for individual tasks may depend on the evaluation metric (e.g., AUCROC always yields 0.5 for majority class voting) as well as task-specific idiosyncrasies such as the size of the label space.

We note that the degradation curves look virtually identical for both DCES or ECES perturbations (given in §A.3). This is in stark contrast to human performance, where ECES was much easier to parse than DCES, indicating the discrepancies between human and machine text processing.

5.3 Shielding

We study four forms of shielding against VIPER attacks: adversarial training (**AT**), visual character embeddings (**CE**), **AT+CE**, and rule-based recovery (**RBR**). For **AT**, we include visually perturbed data at train time. We do not augment the training data, but replace clean examples using VIPER in the same way as for the test data. Based on preliminary experiments with the G2P task, we apply VIPER to the training data using $p_{\text{train}} = 0.2$. Higher levels of p_{train} did not appear to improve performance. For **CE**, we use fixed ICEs, either fed directly into a model (G2P) or via VELMo (all other tasks). For **AT+CE**, we combine adversarial training with visual embeddings. Finally, for **RBR**, we replace each non-standard character in the input stream with its nearest standard neighbor in ICES, where we define the standard character set as a-zA-Z plus punctuation.

Rather than absolute scores, we report differences between the scores in one of the shielding treatments and original scores:

$$\Delta_\tau := \sigma^*(p) - s^*(p), \quad \sigma^*(p) := \sigma(p)/s(0)$$

where $\sigma(p)$ is the score for each task using a form of shielding. The value Δ_τ denotes the improvement of the scores from shielding method τ over the original scores without shielding. We normalize $\sigma(p)$ by the score $s(0)$ of the systems without shielding on clean data. We also note that our test perturbations are unseen during training for DCES; for ECES this would not make sense, because each character has only one nearest neighbor. In the following, we report results mostly for DCES and show the ECES results in §A.3. We highlight marked differences between the results, however.

All tasks typically profit considerably from **AT** (Figure 3 left). Chunking scores improve most; e.g., at $p = 0.5$, σ^* is 17 percent points (pp) higher than s^*. AT does not help for G2P in the DCES setting but it does help for ECES (see §A.3), where test perturbations may have been seen during training. We conjecture that AT makes systems generally aware that the input can be broken in some way and forces them to shield against such situations, an effect similar to dropout. However, such shielding appears more difficult in character-level tasks, where a missing token is considerably more damaging than in word- or sentence-level tasks.

In Figure 3 (right), we observe that **CE** helps a lot for G2P, but much less particularly for POS and Chunking. We believe that for G2P, the visual character embeddings restore part of the input and thus have considerable effect. It is surprising, however, that visual embeddings have no positive effect for both word-level tasks, and instead lead to small deteriorations. A possible explanation is that, as the character embeddings are fed into the ELMo architecture, their effect is dampened. Indeed, we performed a sanity check (see §A.5) to test how (cosine) similar a word or sentence w is to a perturbed version w' of w under both SELMo and VELMo. We found that VELMo assigns consistently better similarities but the overall gap is small.

We observe that the combined effect of AT and CE (**AT+CE**, Figure 4 left) is always substantially better than either of the two alone. For instance, at $p = 0.5$, POS improves by about 20pp, while AT alone had an effect of only 12pp and the effect of CE was even negative. Thus, it appears that AT is able to kick-start the benefits of CE, especially in the case when they alone are not effective.

RBR is excellent for ECES (see §A.3). It has a small negative effect on clean data, meaning that there is some foreign material in English texts which gets corrupted by RBR, but for any $p > 0$ the performance under RBR is almost on the level of $p = 0$ for ECES. RBR is also consistently better

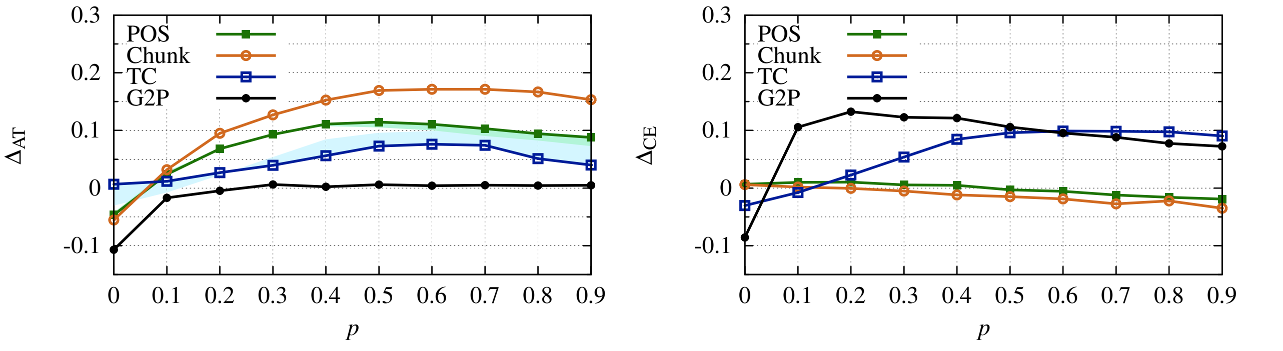

Figure 3: AT (with ICES replacements) and CE tested on DCES perturbed data. The colored regions show AT (with random replacements).

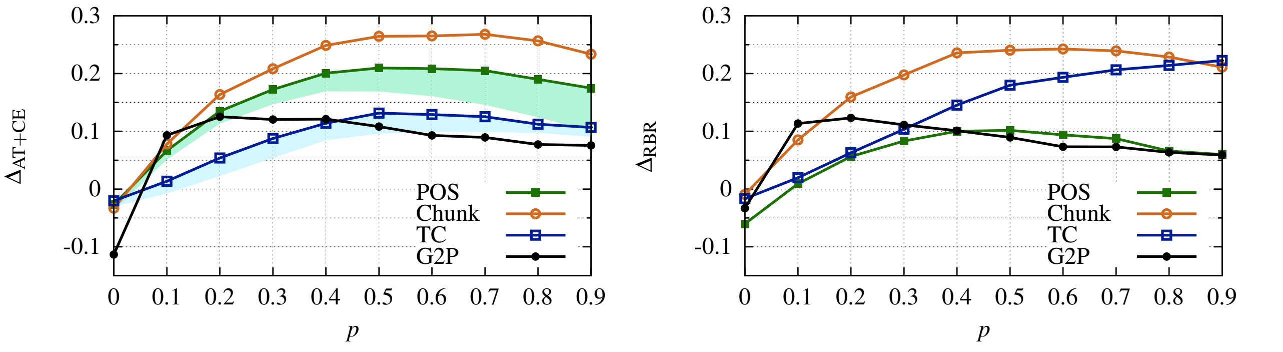

Figure 4: AT+CE (with ICES replacements) and RBR on DCES perturbed data. The colored regions show AT (with random replacements).

than CE, even though both depend on ICES: CE in a 'soft' way and RBR in a 'hard' way. Our best explanation is that RBR is analogous to 'machine translating' a foreign text into English and then applying a trained classifier, while CE is analogous to a direct transfer approach (McDonald et al., 2011) which trains in one domain and is then applied to another. This causes a form of domain shift to which neural nets are quite vulnerable (Ruder and Plank, 2018; Eger et al., 2018a).

For DCES, RBR is outperformed by AT+CE, which better mitigates the domain shift than CE, except for TC.

We note that even with all our shielding approaches, the performance of the shielded systems is still considerably below the performance on clean data at some perturbation levels. E.g., at $p = 0.9$, AT+CE shielded Chunking has a score of less than 60% of the clean performance. While it may be partially due to our character embeddings not being optimal (i.e., they assign low similarity to major and minor variants of the same letter such as ˢ and s, which could be improved by size-invariant CNNs), a main reason for this could be the domain-shift induced by the perturbations, for which even AT cannot always help when attacks are unseen during training. This is another major distinction between human and machine processing.

6 Discussion

Random AT. We discussed that the effect of AT may be similar to dropout. If so, AT with random rather than visual character replacements should be similarly effective. Indeed, the graphs in Figures 3 and 4 show that injecting random noise generally improves the robustness, but not to the level of visually informed AT (exemplarily for POS/TC).

Error analysis. We analyze the cases in which our perturbation with VIPER changes the prediction to a wrong class. We perform our analysis for TC as it represents a real-world attack scenario. We define the *toxicity level* (TL) of a comment x for a class $y \in \{1,\ldots,6\}$ and model θ as $\mathrm{TL}(x) = \sum_y \theta(x,y)$, e.g., a comment x which has been classified as *insult* (probability 0.8) and *obscene* (probability 0.7) gets a TL of 1.5. We consider a successful attack to decrease TL after perturbation. VIPER with DCES and $p = 0.1$ achieves a success rate of 24.1%—i.e., roughly one fourth of the *toxic* comments receive a lower TL. In contrast, the

Perturbation (CES)	Perturbed Text	Toxicity Level	No Shielding	AT	CE	AT+CE	RBR
clean	u are a gigantic faggot	4.00	3.29	3.16	2.93	3.11	3.57
VIPER(p, ECES)	u are â gigantic faggoẗ	4.00	0.11	1.05	1.37	2.05	3.57
VIPER(p, DCES)	u are a ɠigantic faggoŧ	4.00	0.82	1.18	3.36	3.16	1.69
clean	he is also a faggot .	4.00	3.26	3.68	2.81	2.14	3.79
VIPER(p, ECES)	ĥe is also a faggot .	4.00	3.39	3.70	2.64	2.23	3.79
VIPER(p, DCES)	he is alşo a fagᵍot .	4.00	0.48	1.48	1.55	1.78	0.40

Table 5: Two examples of toxic/non-toxic comments that show the effects of the different shielding methods. We report the averaged sum over the six toxicity classes, e.g., 4.00 is equal to a positive example in four classes. $p = 0.1$.

impact on *non-toxic* comments is small—TL increased in only 3.2% of the cases.

Table 5 shows sample comments and their TL for different shielding and perturbation methods. As can be seen, perturbing specific words (*hot words* for TC) substantially reduces the TL score of a non-shielded approach (e.g., from 3.29 to 0.11), while perturbing 'non-hot' words like 'he' has little effect. The shielding approaches help in these show-cased examples to various degrees and the shielding with AT+CE is more robust to stronger attacks (higher visual dissimilarity) than RBR.

This illustrates that a malicious attacker may aim to increase the success rate of an attack by only perturbing offensive words (in the TC task). To test whether VIPER benefits from perturbing such hot words, we manually compiled a list of 20 hand-selected offensive words (see §A.6) which we believe are indicators of toxic comments. We then analyzed how often a perturbation of a word from this list co-occurs with a successful attack. We observe that in 55% of successful attacks, a word from our list was among the perturbed words of the comment. As our list is only a small subset of all possible offensive words, the perturbation of hot words may have an even stronger effect.

7 Conclusion

In this work, we considered visual modifications to text as a new type of adversarial attack in NLP and we showed that humans are able to reliably recover visually perturbed text. In a number of experiments on character-, word-, and sentence-level, we highlighted the fundamental differences between humans and state-of-the-art NLP systems, which sometimes blatantly fail under visual attack, showing that visual adversarial attacks can have maximum impact. This calls for models that have richer biases than current paradigm types do, which would allow them to bridge the gaps in information processing between humans and machines. We have explored one such bias, visual encoding, but our results suggest that further work on such shielding is necessary in the future.

Our work is also important for system builders, such as of toxic comment detection models deployed by, e.g., Facebook and Twitter, who regularly face visual attacks, and who might face even more such attacks once visual character perturbations are easier to insert than via the keyboard. From the opposite viewpoint, VIPER may help users retain privacy in online engagements and when trying to avoid censorship (Hiruncharoenvate et al., 2015) by suggesting visually similar spellings of words.

Finally, our work shows that the 'brittleness' (Belinkov and Bisk, 2018) of NLP extends beyond MT and beyond word reordering or replacements, a recognition that we hope inspires others to investigate more ubiquitous shielding techniques.

Acknowledgments

We thank the reviewers for helpful feedback. This work has been supported by the German Research Foundation (DFG) funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1), the DFG-funded projects QA-EduInf (GU 798/18-1, RI 803/12-1), DIP (GU 798/17-1), the German Federal Ministry of Education and Research (BMBF) under the promotional references 16DHL1040 (FAMULUS) and by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich- Ökonomischer Exzellenz" (LOEWE) as part of the a! - automated language instruction project (No. 521/17-03). We gratefully acknowledge support of NVIDIA Corp. with the donation of the Tesla K40 GPU used for this research. Calculations for this research were also conducted on the Lichtenberg high performance cluster of Technical University Darmstadt.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896. Association for Computational Linguistics.

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

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2587–2597.

Falcon Dai and Zheng Cai. 2017. Glyph-aware embedding of chinese characters. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP, Copenhagen, Denmark, September 7, 2017*, pages 64–69.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. *ICLR Workshop*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 31–36.

Steffen Eger, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2018a. Cross-lingual argumentation mining: Machine translation (and a bit of projection) is all you need! In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Steffen Eger, Paul Youssef, and Iryna Gurevych. 2018b. Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4415–4424. Association for Computational Linguistics.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Chaya Hiruncharoenvate, Zhiyuan Lin, and Eric Gilbert. 2015. Algorithmically bypassing censorship on sina weibo with nondeterministic homophone substitutions. In *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 150–158.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. 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 1875–1885. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2059–2068.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ryan T. McDonald, Slav Petrov, and Keith B. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011*

Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 62–72.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy*, pages 582–597.

Matthew 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. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Korin Richmond, Robert A. J. Clark, and Susan Fitt. 2009. Robust LTS rules with the combilex speech technology lexicon. In *INTERSPEECH*, pages 1295–1298. ISCA.

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

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning, CoNLL 2000, and the Second Learning Language in Logic Workshop, LLL 2000, Held in cooperation with ICGI-2000, Lisbon, Portugal, September 13-14, 2000*, pages 127–132.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714. The COLING 2016 Organizing Committee.

Daiki Shimada, Ryunosuke Kotani, and Hitoshi Iyatomi. 2016. Document classification through image-based character embedding and wildcard training. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 3922–3927.

Sameer Singh, Carlos Guestrin, and Marco Túlio Ribeiro. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 856–865.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 4480–4488.

A Appendices

A.1 SELMo and VELMo Hyperparameters

Differences to the original ELMo as in (Peters et al., 2018) are:

- We exclude CNN filters of size 6 and 7.
- The maximum characters per token is 20 (instead of 50).
- The LSTM dimensionality is 2048 (instead of 4096).
- Our projection dimensionality is 256 (instead of 512).
- We train the models for 5 epochs (instead of training it for 10 epochs).

A.2 Task Settings

G2P: We randomly draw most hyperparameters for the sequence tagging BiLSTM architecture (Reimers and Gurevych, 2017) that we use for G2P, e.g., those concerning the optimizer used, learning and dropout rates. We hand-set the number of hidden recurrent layers to 1, and its size to 50. We use early stopping and set the maximum number of epochs for training to 50. As our dataset, we choose the Combilex pronunciation dataset of American English (Richmond et al., 2009). We randomly draw our train/dev/test splits from the whole corpus. Examples and split sizes are given in Table 4. We report **edit distance** between desired pronunciations and predicted pronunciations as metric. We report the edit distance averaged across all 1k test strings, averaged over 5 random initializations of all weight matrices.

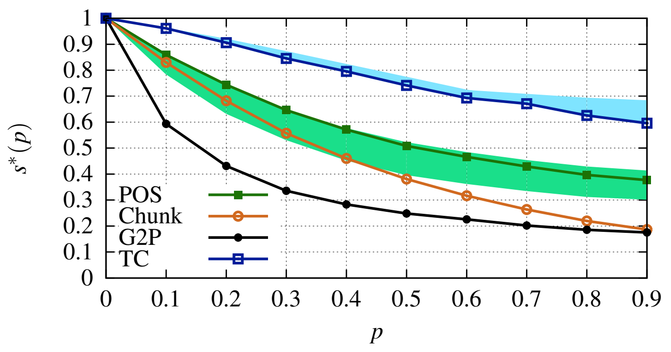

Figure 5: Degradation of SOTA systems for different perturbation levels when attacked by VIPER(p,ECES). The colored regions show how the performance of other SOTA systems relate to ours.

POS & Chunking: We use the training, dev and test splits provided by the CoNLL-2000 shared task (Sang and Buchholz, 2000) for both tasks. We have used a readily available LSTM-CRF sequence tagger (Reimers and Gurevych, 2017) as above, but adapted for ELMo-type input embeddings, with default hyperparameter settings. We run each experimental setting 10 times, and report the average, measured as **accuracy** for POS and **micro-F1** for Chunking. For both tasks, we have used two stacked BiLSTM-layers with 100 recurrent units and dropout probability of 0.5. Mini-batch size is chosen as 32. We used gradient clipping and early stopping to prevent overfitting. Adam is used as the optimizer.

Toxic comment classification: We use the train and test splits provided by the task organizers. For tuning our models, we split off a development set of 10k sentences from the training data. As in POS&Chunking, we train models on clean and perturbed data using SELMo and VELMo representations. We obtain the sentence representation for a single sentence by averaging ELMo word embeddings over all tokens. We then train an MLP which we tune separately for each SELMo and VELMo embedding using random grid search with 100 different configurations. We tune the following hyperparameters separately for each hidden layer: the depth of the neural network, i.e., one, two, or three hidden layers; the size of the hidden layer (128, 256, 512, or 1024); the amount of dropout after each layer (0.1 - 0.5); the activation functions for each hidden layer (*tanh*, *sigmoid*, or *relu*) (Eger et al., 2018b). Both models are trained for 100 epochs with an early stopping after 10 epochs without any substantial improvement and use Nesterov-accelerated Adaptive Moment Estimation (Dozat, 2016) for optimization. Model performance is measured as proposed by the task organizers using the *area under the receiver operating characteristics curve* (**AUCROC**).

A.3 ECES Results

Figure 5 shows how the performances of various SOTA systems degrade on ECES settings. Figures 6 and 7 show our shielding results on ECES perturbed data. As indicated in the main paper, RBR is able to recover ECES data almost perfectly regardless of the perturbation level. This is because ECES only perturbs with a single nearest neighbor, which in addition is visually extremely similar to the underlying original, and thus, RBR can almost completely undo the perturbations.

A.4 AT+CE *vs.* AT or CE

Figure 8 compares AT+CE against either AT or CE. For this, we compute the difference in the performance decrease normalized by the test performance on the clean data. As can be seen, AT+CE almost constantly outperforms either one of both, especially on word- and sentence-level tasks.

A.5 Intrinsic Evaluation

To analyze the differences between VELMo and SELMo, we investigate whether the models learn similar word embeddings for a clean sentence and its visually perturbed counterpart. We compare sentence embeddings which we obtain by averaging over the SELMo or VELMo word embeddings of a sentence (clean or perturbed).

Setup Given a sentence embedding σ of a clean sentence and an embedding σ' of its visually perturbed counterpart, obtained by either averaging over VELMo (indicated by subscript v) or SELMo (indicated by subscript s) word embeddings, we test if the condition

$$\cos(\sigma_v, \sigma'_v) > \cos(\sigma_s, \sigma'_s) \qquad (1)$$

is met (with cos being the cosine similarity).

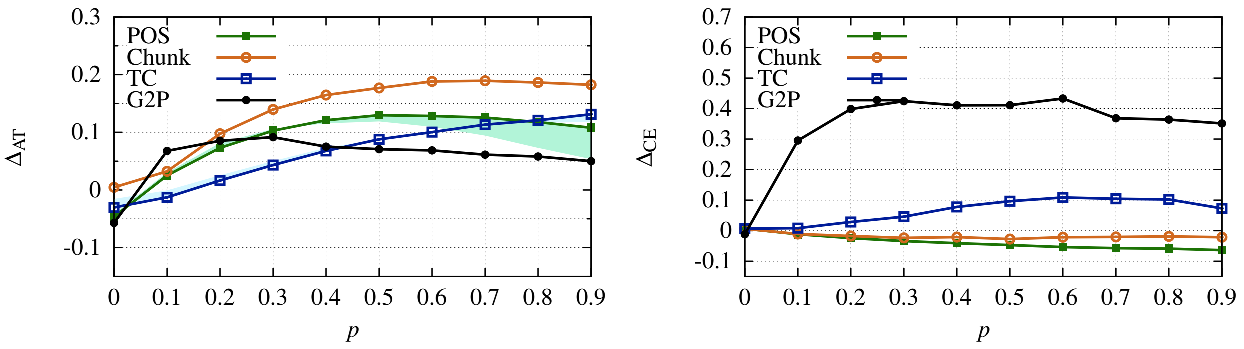

Figure 6: AT (ICES) and CE tested on ECES perturbed data. The colored regions show AT (Random). The y axis of Δ_{AT} spans -0.15-0.3 for better visualization.

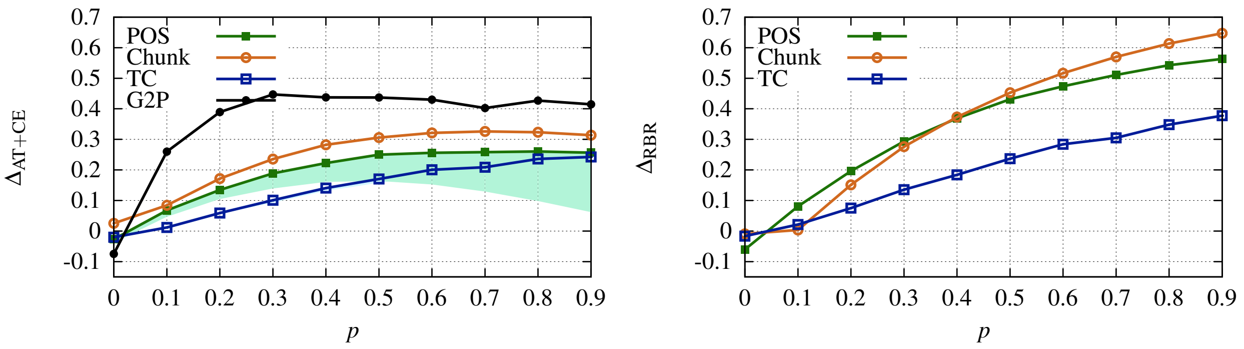

Figure 7: AT+CE (ICES) and RBR on ECES perturbed data. The colored regions show AT (Random).

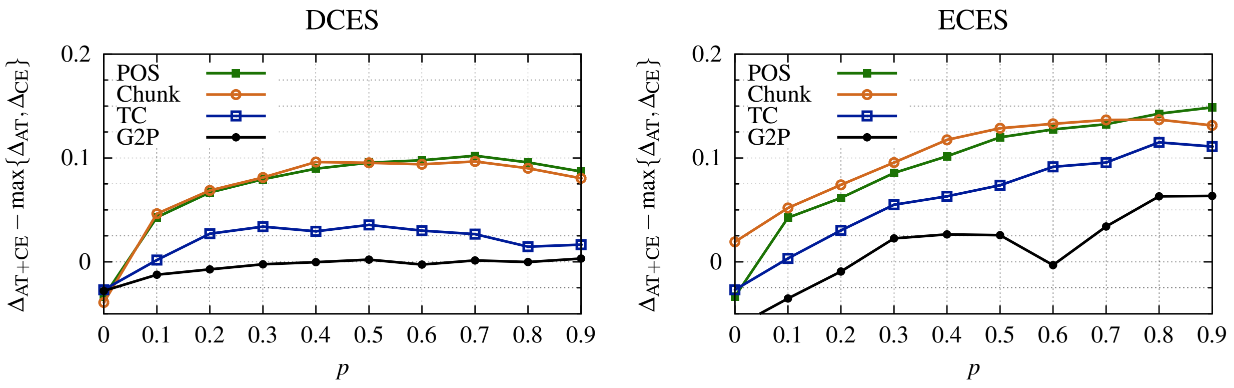

Figure 8: AT+CE (ICES) vs max(AT,CE) for DCES and ECES perturbed test data.

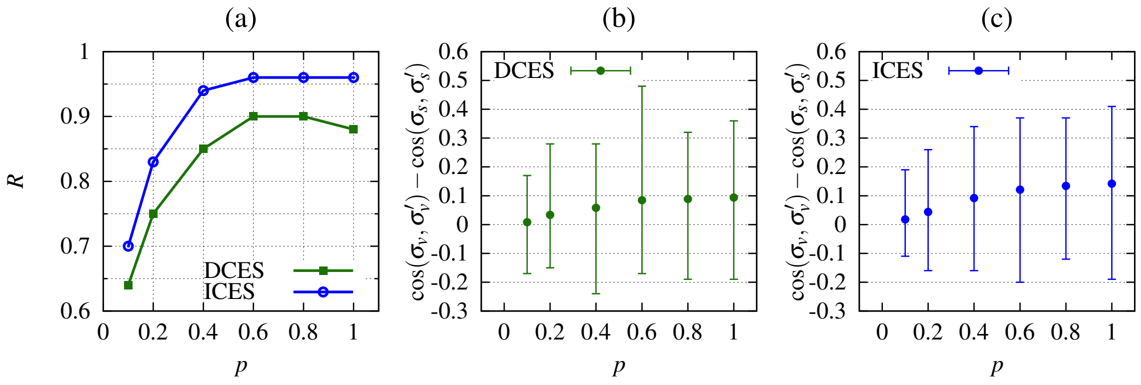

Figure 9: Results of the intrinsic evaluation. (a) shows the ratio of cases in which the condition in Eq. (1) is met ('R'). (b) and (c) show the average difference of the cosine similarities when sentences are perturbed with (b) DCES and (c) ICES.

For our experiments, we randomly sample 1000 sentences from the Toxic Comments dataset (see §5.1) and perturb them with VIPER(p, CES) where CES $\in$ {ICES, DCES}. We then count the number N of cases in which the above condition is met with regards to the chosen CES and the value of p, and report the ratio $R = N/1000$.

Results The results are given in Figure 9. In Figure 9(a) we observe that the VELMo embeddings of a clean sentence and its perturbed counterpart are in many cases more similar than the ones of SELMo. For larger values of p, this ratio substantially increases from 70% to 95% (with ICES), which shows that VELMo is better suited to capture the similarity to the source sentence, especially in cases with a strong perturbation.

In Figures 9(b) and (c) we show the (mean) difference $\cos(\sigma_v, \sigma'_v) - \cos(\sigma_s, \sigma'_s)$. Here, we only observe a small positive effect in favor of VELMo, showing that the advantage of VELMo over SELMo is not substantial. We hypothesize that this is due to the contextual information which is utilized throughout the ELMo architecture, allowing SELMo to infer individual characters from the context of the word and the sentence. However, our results also show that the advantage of VELMo over SELMo is consistent.

Differences in similarities can also be affected by model training or the model architecture—e.g., in an extreme case a model could output the same embedding for every word/sentence. This would result in a 'perfect' cosine similarity, which would be advantageous in the previous experiment. Thus, we perform an additional experiment where we examine if

$$\cos(\sigma_v, \sigma'_v) > \cos(\sigma_v, \rho_v) \qquad (2)$$

	DCES		ICES	
p	**VELMo**	**SELMo**	**VELMo**	**SELMo**
0.1	0.99	0.98	1.00	0.97
0.2	0.98	0.84	0.98	0.77
0.4	0.81	0.38	0.85	0.25
0.6	0.60	0.10	0.63	0.07
0.8	0.42	0.04	0.49	0.03
1.0	0.34	0.01	0.39	0.01

Table 6: Results of the intrinsic evaluation where we compare clean sentences to their perturbed counterparts as well as randomly chosen sentences. The numbers show the ratio of cases where clean sentences are more similar to their perturbed counterparts than the randomly chosen sentences.

holds, where ρ is a randomly sampled sentence from the Toxic Comments dataset (with no perturbation). The same experiment is also performed for SELMo.

The results in Table 6 show that for both models with $p = 0.1$ the original sentence is in 97–100% of the cases more similar to its perturbed counterpart than the randomly chosen sentence. As the perturbation probability increases, VELMo has a clear advantage over SELMo. E.g., if we perturb all characters in a sentence ($p = 1.0$), the SELMo embeddings of the perturbed sentence are in 1% of the cases more similar to the original sentence whereas this is the case for more than 34–39% for VELMo. Thus, VELMo embeddings better capture similarity between visually similar words.

A.6 List of Hand-selected Curse Words

arrogant, ass, bastard, bitch, dick, die, fag, fat, fuck, gay, hate, idiot, jerk, kill, nigg*[8], shit, stupid, suck, troll, ugly

[8]Due to several variations in the data, we match against *nigg** instead of the whole word.