Analyzing the Real Vulnerability of Hate Speech Detection Systems against Targeted Intentional Noise

Piush Aggarwal **Torsten Zesch**
Computational Linguistics
CATALPA - Center of Advanced Technology for Assisted Learning and Predictive Analytics
FernUniversität in Hagen
{piush.aggarwal,torsten.zesch}@fernuni-hagen.de

Abstract

Hate speech detection systems have been shown to be vulnerable against obfuscation attacks, where a potential hater tries to circumvent detection by deliberately introducing noise in their posts. In previous work, noise is often introduced for all words (which is likely overestimating the impact) or single untargeted words (likely underestimating the vulnerability). We perform a user study asking people to select words they would obfuscate in a post. Using this realistic setting, we find that the real vulnerability of hate speech detection systems against deliberately introduced noise is almost as high as when using a whitebox attack and much more severe than when using a non-targeted dictionary. Our results are based on 4 different datasets, 12 different obfuscation strategies, and hate speech detection systems using different paradigms.

1 Introduction

Computer-mediated communication is plagued by toxic and hateful behavior that can cause serious harm (Waldron, 2012; Gelber and McNamara, 2016). Consequently, automatically detecting such behavior has become a major research area (Waseem and Hovy, 2016; Waseem et al., 2017; Kumar et al., 2018; Wiegand et al., 2019; Aggarwal et al., 2019; Kovács et al., 2021).

Whenever there is an automatic system in place to filter out hateful messages, people will try to circumvent it by obfuscating their message. However, there are limits, as the communicative intent has to stay intact. If the intended audience cannot relatively easily understand a message, obfuscation has gone too far. Thus, people will usually only obfuscate a few terms they think will be problematic or could be responsible for filtering out a message (see Table 1). Adding to much noise may render the message unrecognizable even to human readers.

In this paper, we analyze the real vulnerability of state-of-the-art hate speech detection systems against targeted obfuscation. We keep a post recognizable by obfuscating at most one target token per test example. We perform an annotation study to analyze the target selection strategies in real-world setting. We use 12 types of plausible obfuscation strategies and apply them to the targeted tokens. In order to generalize our analysis, we repeat our experiments on multiple hate speech datasets. For future benchmarking, we open-source our code base, trained models, and obfuscated test samples[1].

2 Ethical Considerations

In our research, we are discussing and explaining obfuscation strategies. People could use those strategies to avoid detection and eventually cause even more harm (Prabhumoye et al., 2021). We consider this risk to be small, as people are creative and would have (and almost certainly already have) come up with all of the described strategies.

We decided to release all our code (even the parts obfuscating single words), as we see a clear benefit in researchers reproducing our results and facilitate their own research. This outweighs the risk caused by people using that code to automatically obfuscate their messages.

Law enforcement agencies might use our research to build more robust detection systems. This can be positive, as marginalized groups might not have to deal with being targets of hate speech all the time and might dare again to use their free speech rights without being threatened into silence. However, social media platforms may use deobfuscation to ban words they consider offensive. This might have unintended consequences, e.g. a person called *Richard Gaywood* was not able to use his name as it include the word *gay* and consider to violate community standards (Suzor, 2010).

An even more serious harm are overreaching governments censoring non-hateful expression of

[1] https://github.com/aggarwalpiush/HateSpeechDetection

Post	Obfuscation Level
`A** h*te sp**ch res**rchers sh**ld b* b**ten t* d**th`	High
`All h*te speech res**rchers should be b**ten to d**th`	Medium
`All hate speech researchers should be b**ten to d**th`	Low

Table 1: Trade-off between level of obfuscation and message understandability.

opinions. Over the last years, the web-based censorship as well as surveillance has significantly increased in some parts of the world (Polyakova and Meserole, 2019). Improved detection models will even more empower such authoritarian regimes and give them the opportunity to increase the severity of surveillance towards its citizens (Sherman, 2020; Wright, 2018), as they can no longer circumvent censorship through obfuscation.

3 Obfuscation Strategies

Obfuscation is the deliberate act of obscuring the intended meaning of communication by adding noise to the message. This can happen in many ways. For example, Gröndahl et al. (2018) show that appending a positive word like *love* can already fool a classifier. Kirk et al. (2022) demonstrate the vulnerability of hate detection models by simple replacements of certain tokens with emojis. Another obfuscation strategy is to paraphrase the whole message or use a metaphorical expression to indirectly express the same point. For example, instead of *All those researchers are stupid as hell* one could write *Those Einsteins are not the sharpest knife in the drawer*. However, this also changes the meaning and presupposes that the intended audience is aware of the possible replacement and able to make the connection. Such a replacement might also change the perceived severity of a toxic comment, e.g. it is possible that people would consider the sentence with *Einstein* as more hateful, as it adds a potential connotation of *Jewish researcher*.

In this paper, we only focus on producing simple obfuscation strategies (Röttger et al., 2020) which are generally observed to be implemented in realistic settings. Table 2 gives examples of all strategies that we consider.

Camel Case While camel casing is usually used to improve the readability of a text (e.g. naming conventions in computer programming), it can also be used to add noise. In our camel-casing strategy, we capitalize every alternate letter (starting from the second letter).

Char Drop While keeping first and last character untouched we either drop a randomly chosen character from the selected token or drop all vowels (Cleary, 1976; Baluch, 1992). We call the later obfuscation as **Vowel Drop**. We ensure preservation of token perception (Baba and Suzuki, 2012; ThambiJose, 2014; Pruthi et al., 2019)). Therefore, we only add the noise to tokens with 3 or more characters.

Char Flip Character shuffling within the boundaries of the word (excluding boundary characters) does not have much effect on word semantics. However, it would be quite easy to get this implemented during the message composition. To generate such noise, we only consider tokens having length more than 3. Excluding first and last, we randomly select two characters and flip them.

Diacritics Some non English languages use extra marks or glyph (such as ˆ) above or below (or sometimes next to) a letter for explicit enunciation. We use mapping table to generate diacritic version of the input token.

Kebab Instances are created by adding a *dash* (−) between each letter of the word. This looks like meat on a kebab stick, hence the name.

Leetspeak Visual resemblance of alphabets (Simpson et al., 2012) with numbers and mathematical symbols can also be used to obfuscate token. Therefore we exploit leetspeak where we consider commonly used English alphabets namely *a*, *e*, *l*, *o*, *s* and replace with *4*, *3*, *1*, *0*, *5* respectively.

Masking Deliberate introduction of symbols such as mathematical operators are common practice to obfuscate the disputed tokens. We generate masking based obfuscation examples where we randomly choose and replace one letter with * (tokens with two letters are not considered for this obfuscation). To increase the perception of the token, we do not consider first and last letter of the token for the replacement.

Mathspeak Similar to leetspeak, mathspeak replaces characters with mathematical symbols such

Strategy	Example
	`researcher`
Camel Case	`rEsEaRcHeR`
Char Drop	`researher`
Char Flip	`resaercher`
Diacritics	`résearchêr`
Kebab	`r-e-s-e-a-r-c-h-e-r`
Leetspeak	`re5earc7er`
Masking	`resear**er`
Mathspeak	`ℜesearcher`
Phonetic	ɪsɜːtʃə
Spacing	`r e s e a r c h e r`
Snake	`r_e_s_e_a_r_c_h_e_r`
Vowel Drop	`rsrchr`

Table 2: Overview of Obfuscation Strategies.

as *R* with ℜ.

Phonetic In phonetic obfuscation, the token is replaced with a representation of how it is pronounced. For our example *researcher*, this could be a layperson representation like '*ri-sur-chur*' or blending with aspects of mathspeak if using the international phonetic alphabet (IPA) which would result in 'rɪsɜːtʃə'. In this study, we apply IPA representation for obfuscation generation. Though we do not consider this a very practical obfuscation strategy outside of 'Linguistics Twitter', still keep it in our experiments as an extreme case.

Spacing In this case, examples are created by adding *spaces* between each letter of the word.

Snake Instances are created by adding *underscore* (_) mark between each letter of the word.

4 Target Selection

We propose model independent token selection strategies that range from very broad (all tokens) to very specific (the words conveying the hateful intent). Our intention is to provide every possible cases in order to analyse the model robustness in depth. Table 3 illustrates how tokens are chosen based on different target selection strategies. In this section, we describe each of the strategy in detail.

All We obfuscate all tokens with more than 3 characters (obfuscation of shorter words cannot be reliably performed). This is the most aggressive obfuscation strategy that will probably make it unreadable to humans and machines alike.

Random Any A single word (with more than 3 characters) is randomly selected from the message text. We do not obfuscate the first and last word of the text.

Random Content The same strategy as random word, only that selected words have to be either nouns, verbs, adjectives, or adverb.

Dict Fixed We collect a number of lexicons with hateful words from various sources (including Hatebase[2] and published research on lexicons (Bassignana et al., 2018; Wiegand et al., 2018; Chandrasekharan et al., 2017)). For each language we combine all lexicons and remove duplicates. As token (starting from left side) in the input text found match in the dictionary is selected for the obfuscation and the remaining available tokens are ignored.

Dict Whitebox Following (Papernot et al., 2016) hypothesis, we build an in-house lexical dictionary populated with tokens which are important for an LSTM-based hate-speech classification model for hate-labels predictions. We refer these lexicons as whitebox tokens as they are selected based on model internal parameters. To extract such tokens, first we train this model on the existing hate-speech datasets (see Section 4) and apply a hierarchical based explanation method (Jin et al., 2020). To generate explanations, we use the same training instances on which the model was trained as we are only interested in hateful tokens. The explanations are in the form of scores for all possible n-grams available in the training statements which represent contribution of the n-grams towards hate label predictions. Heuristically, we choose all unigrams having threshold value less than or equal to -0.02 (negative polarity leads to hatefulness). For selection, among all the token matches, we consider token with most negative score.

Dict Domain All target selection methods outlined so far, are trying to simulate the real obfuscation process in a rather crude way. When people want to obfuscate single words, they know which are the most problematic ones and focus only on those. However, for our experiments, we do not know which words this would be. We thus performed an annotation study (described next), which resulted in a domain-specific dictionary. We later use this dictionary (like *Dict Whitebox*) to obfuscate exactly one word in each post that people consider as most problematic. Thus, this target selection strategy is much more realistic than the other

[2] https://hatebase.org/

Target Selection	Post
Clean	`All hate speech researchers should be beaten to death`
Random Content	`All hate speech` **`researchers`** `should be beaten to death`
Random Any	**`All`** `hate speech researchers should be beaten to death`
Dict Fixed	`All hate speech researchers should be beaten to` **`death`**
Dict Whitebox	`All hate speech researchers should be` **`beaten`** `to death`
Dict Domain	`All hate speech researchers should be` **`beaten`** `to death`
All	**`All hate speech researchers should be beaten to death`**

Table 3: Overview of target selection strategies. They differ in which and how many tokens will be selected for obfuscation.

ones and will allow us to better estimate the real vulnerability of hate speech detection systems.

4.1 Annotation Study

To gather a domain specific lexicons, we perform an annotation study. We chose a random sample of 100 hate speech statements from the (Davidson et al., 2017) dataset. We recruited three annotators[3] and asked them to think like potential hater and select three tokens from each statements which are most likely to be chosen for obfuscation (see Figure 1) For the annotation study, we used the inception framework (Klie et al., 2018).

Annotators received the following instructions. First, we provided the scenario:

> Imagine you want to spread hate using social media platforms. Sooner, you realize most of these social media platforms are equipped with hate speech detection systems. Now you want to fool these systems by playing with words you used in message. For example: *Researchers should be banished from holy places*
>
> You can play with words such as *banished*
>
> and make it *ban1shed*

Then, we provided the purpose of annotation study:

> This annotation process is intended to perform sociological analysis. We manually labelled the tokens in the social media posts which potentially be obfuscated during the post-composition to escape from automatic hate-speech detection process.

Finally, we explained the annotation process:

> For each sentence (total: 100), choose three tokens and annotate with their priority levels. For example:
>
> First_Priority: *banished*
>
> Second_Priority: *Researchers*
>
> Third_Priority: *holy*

[3]university graduates and active social media users.

The study resulted in 455 tokens marked by the annotators. Inter-annotator agreement (taking priority into account) was 0.64 gamma (Mathet et al., 2015). It illustrates that annotators are not only in high agreement for token selection but also for priority.

To generate the domain specific dictionary from the annotations, we assign them with a score. This score represents the polarity of hatred carried by the token relative to other tokens available in the list. We use the scores[4] (Equation 1) to prioritize the token selection strategy during obfuscation process.

$$S_{ti} = \sum_{j=1}^{v_{ti}} \vec{P_{tij}} \cdot \vec{W} \qquad (1)$$

To calculate, we create priority vectors (e.g. [1,0,0] for *retard*, [0,0,1] for *stupid*, etc. in Figure 1) for each token (S_{ti}), we take dot product of the priority vector ($\vec{P_{tij}}$) with constant scalar weight vector ($\vec{W}$) $[0.5, 0.33, 0.17]$, summation over token's frequency (v_{ti}) (as single token can have multiple priority vectors depending upon its usage across the statements). The constant weight vector describe the relative amount of preference should be given to each token based on its priority with respect to other token.

5 Experimental Setup

We experiment with all the obfuscation and target selection strategies outlined above. To make sure that our results are not specific to a dataset or detection method, we also use multiple dataset and methods as outlined next.

5.1 Datasets

In order to analyze the vulnerability of available hate speech classifiers, we have used 4 social media hatespeech datasets (see Table 4).

[4]The list of tokens with their scores can be download from the provided github repository

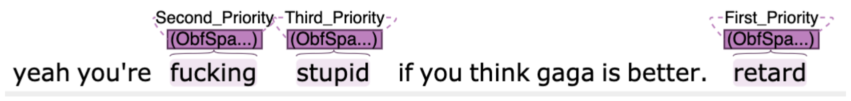

Figure 1: A sample hate statement with top three tokens that are most likely to be obfuscated by a potential hater.

Name	Reference	# posts	tokens	% hate
T1	(Davidson et al., 2017)	24,783	370k	6
T2	(Waseem and Hovy, 2016)	10,588	160k	26
G	(Kennedy et al., 2022)	27,663	590k	12
TF	(Mandl et al., 2019)	7,004	170k	36

Table 4: Specification of datasets use to analyze the real vulnerability against target obfuscations.

- Davidson et al. (2017) (T1) contains Twitter posts labeled with *hate, offensive, not*. It contains a wide range of domains as its collection strategy is based on lexicons provided by *Hatebase.org*.

- Waseem and Hovy (2016) (T2) contains tweets manually tagged with *sexist, racist, not*. Like T1, the tweets were collected but with fewer lexicons.

- Kennedy et al. (2022) (G) contains posts from social media service *gab.ai* with multiple hate-based rhetoric labels.

- Mandl et al. (2019) (TF) contains binary-labeled Tweets and Facebook posts.

For datasets with non-binary labels, we aggregate the labels into two categories namely *hateful* and *not-hateful*. We randomly stratified dataset posts into train, dev and test set in the ratio of 80:10:10 respectively. We apply the proposed obfuscation attacks only on the test set in order to analyse the model robustness against unknown attacks.

5.2 Hate Detection Systems

To generalize our study we train 12 different types of hate detection systems. It include shallow, deep, deep-attention and deep-contextualized based paradigm.

Shallow Models Following Davidson et al. (2017), training shallow machine learning algorithm for hate-speech classification such as support vector machine and logistic regression can be considered as strong baseline. In addition, we use ensemble based classification algorithms such as AdaBoost, Gradient Boosting and Random Forest.

Deep Models Wide range of approaches have been used for hate speech detection. We select a range of reference architectures instead of specific configurations by certain researchers, as we are mainly interested in the relative vulnerability of architectures. Contextualized language model based classification systems such as BERT (Devlin et al., 2019) promise state of the art result in wide domain of downstream tasks. Consequently, for hate-speech classifications, we perform fine-tuning of a variant of BERT called *Distilbert* (Sanh et al., 2019). Textual classification is often considered a time-series problem, where the representation of each token in the text is depends on former and later tokens available in the text. Therefore, we train different variants of LSTM based neural network such as LSTM, BILSTM, CNN with attention networks and CNN-LSTM. Hochreiter and Schmidhuber (1997); Zhou et al. (2016); Brahma (2018); Sainath et al. (2015).

5.3 Model Training

Except *Distilbert*, for training of rest of the systems, we lowercase all postings for each dataset and use the Ark Tokenizer (Gimpel et al., 2011) for word splitting. To extract features, we use word embeddings (Zhang and Luo, 2018; Kshirsagar et al., 2018; Badjatiya et al., 2017). Due to many OOVs in hate speeches, hate speech models adopt character level features (Del Vigna et al., 2017; Warner and Hirschberg, 2012; Lee et al., 2018) where a DNN produces local features around each character of the word and then combines them using a max operation to create a fixed-sized character-level embedding of the word. Char-level embeddings are more likely to encode all variants of a word's morphology closer in the embedded space (Bojanowski et al., 2017). We use n-char fastext embeddings trained on Twitter corpus of 400 Million tweets (Godin, 2019). For shallow models, we apply grid-search algorithm using a dev set on all shallow classifiers (Pedregosa et al., 2011). For all the deep-neural networks, we use the learning rate of 10^{-3} with 16 as batch size. We train each network for 10 epochs with early stopping on dev set accuracy and for 4 patience level. In the case of

Distilbert, we use BERT-base tokenizer and corresponding contextual embeddings for tokenization and feature extraction respectively. We use default hyperparameter settings described in the original *Distilbert* implementation[5]

6 Results

We analyze model vulnerability by looking at the decline in relative performance when they are tested on obfuscated posts. Since we use unbalanced datasets, we estimate the performance using F1 Macro score evaluate on hate-labels.

Target Selection Table 5 shows relative change in F1(Hate) averaged over obfuscation strategies for all systems. We use the broadest as well as the most specific target selections for obfuscations. As expected, model performance is worst when *All* tokens are obfuscated in the test samples. Since this is an unrealistic strategy, we do not consider this effect as real vulnerability. On the other end of the spectrum, random target selection (*Random Content* and *Random Any*) has little effect. It becomes clear that the model are sensitive to specific and meaningful tokens.

Dict Fixed accommodates meaningful lexicons towards hatred and therefore makes systems relatively more vulnerable to it. The T2 dataset is an exception, as it has relatively fewer swear words and hence less number of target tokens are selected. It also indicate that reliance on fixed set of tokens may not be the best solution to generate the obfuscation examples.

An important finding is that a model-dependent dictionary (*Dict Whitebox*) has a large impact on model susceptibility, as does *Dict Domain*. The comparable performance indicate towards the similar ranking order of tokens in both of the dictionaries. To estimate the similarity, we calculate the *Spearmanr Coefficient* (Schober et al., 2018) and find it as 0.421 with $p < 0.005$ which is a moderate correlation. This shows the promising future direction for conducting annotation studies on larger datasets to compile a better manual preferences based dictionary.

Obfuscation Strategies Table 6 shows the average relative change in F1 (Hate) for all our obfuscation strategies, where higher numbers mean that systems are more vulnerable against this strategy. We find that the type of preprocessing might play an important role as e.g. *camelcasing* has almost no effect because lowercasing is performed during preprocessing. In general, models are more vulnerable to strategies that insert characters in a token (like *Kebab* or *Spacing*), than to strategies that remove characters (like *Char Drop* or *Vowel Drop*). This could be linked to the modeling of subwords, but more research is needed in that direction. Strategies that replace characters sometimes have limited applicability, e.g. *Mathspeak* can only be applied for certain characters limiting its effect.

Distilbert To perform deep analysis, we visualize the fine-grained results for our best performing model *Distilbert* (Figure 2 and 3) on T1 dataset. In most cases, we find the model performance in accordance with numbers mention in Table 5 and 6. As expected, the effect of *Random Content* is more prominent compared to *Random Any* which validate the advantages of using limited POS tags. Multiple edit operations make system more vulnerable to *Vowel Drop* than *Char Flip* and *Char Drop* (need only single edit). We also note that for this specific classifier (in contrast to the averaged results discussed above) the *Phonetic* method is even better than *Kebab* and *Spacing*. Please refer Appendix A for fine-grained results of *Distilbert* evaluated on other datasets.

Other Paradigms Other than *Distilbert*, we have evaluated the vulnerability on shallow classifier such as SVM, LogReg, AdaBoost, Gradient Boosting and Random Forest as well as on Deep Networks namely LSTM, BILSTM, CNN with attenion networks, also on CNN-LSTM. We found the order of model's vulnerabilities with respect to obfuscation targets are consistent with *distilbert* which can be interpreted by Tables 5. Table 7 in Appendix B illustrates the performance drop for each system across T1 dataset.

7 Related Work

Although most of the previous studies raise concern about model robustness against obfuscation attacks, the real vulnerability is understudied. Among simple obfuscation strategies, studies (Gröndahl et al., 2018; Röttger et al., 2021; Ebrahimi et al., 2018; Szegedy et al., 2013) introduce syntactic perturbations to validate the robustness of hate detection models. We find overlapping of some of the obfuscation strategies discussed in this paper. Kirk et al.

[5]`https://huggingface.co/docs/transformers/model_doc/distilbert`

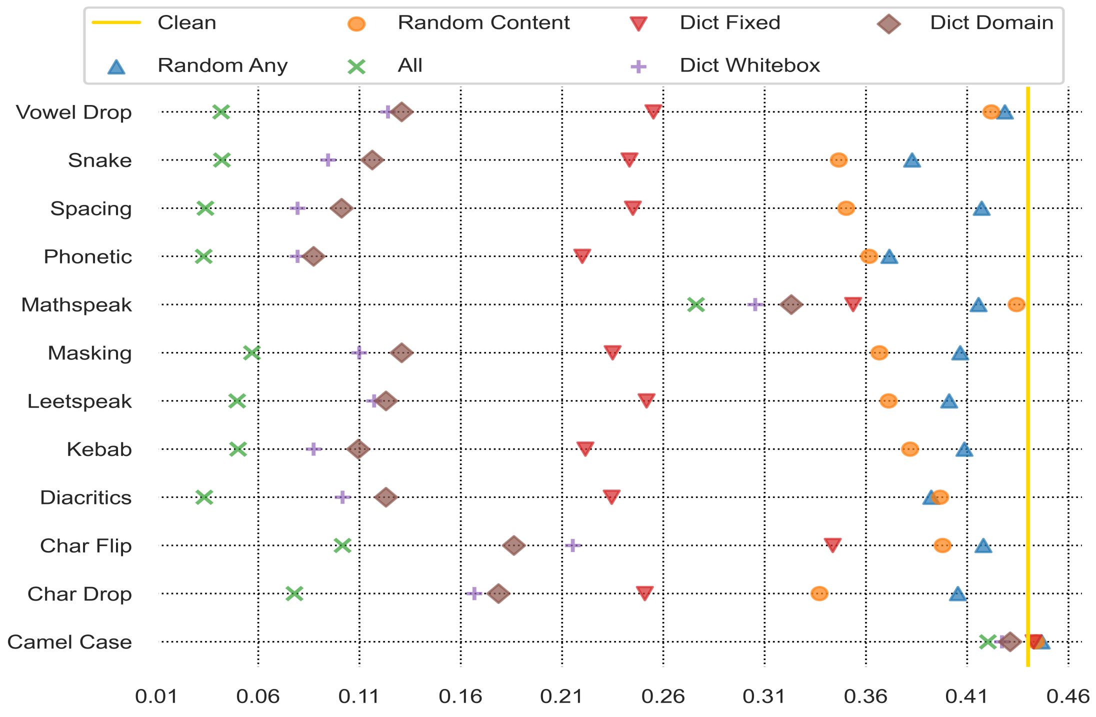

Figure 2: Distilbert's F1 (Hate label) performance on T1 dataset before and after obfuscation on all obfuscation strategies for all target selections. Except *Camel Case*, differences are found to be statistically significant based on McNemar-Test after Bonferroni correction $p < 0.05$.

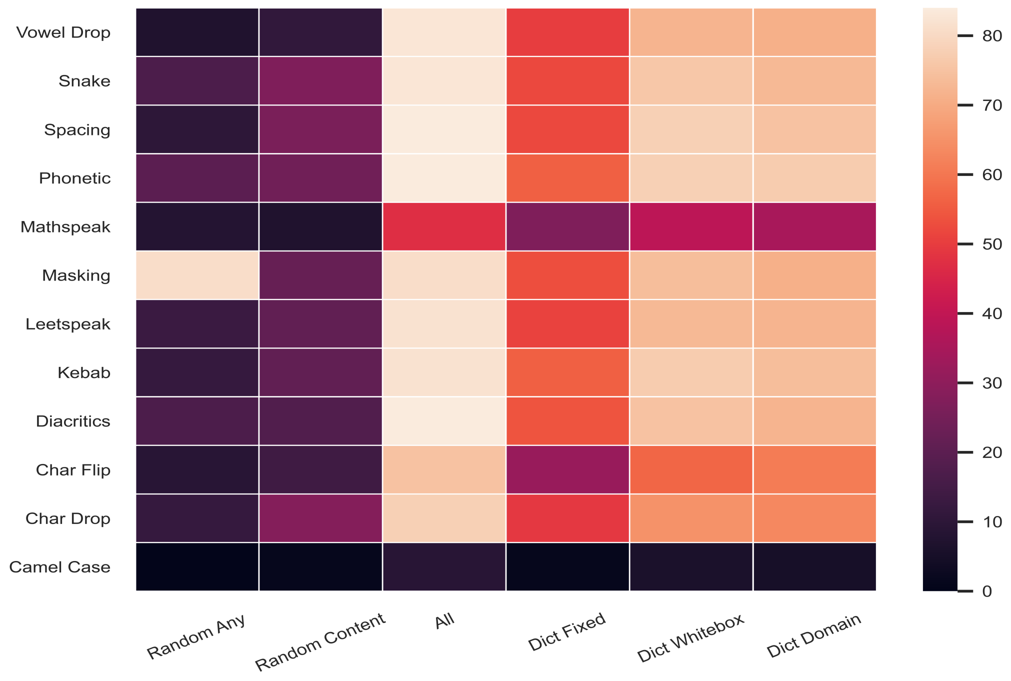

Figure 3: Distilbert's performance decline on T1 dataset based on differences in the True Positives (Hate label) before and after obfuscation on all obfuscation strategies for all target selections.

Target	Datasets T1	T2	TF	G
All	.16	.35	.26	.28
Dict Domain	.13	-	-	-
Dict Whitebox	.13	.13	.16	.10
Dict Fixed	.11	.01	.06	.11
Random Content	.05	.06	.02	.08
Random Any	.03	.05	.02	.07

Table 5: Relative change in F1 (Hate) for different target selection strategies. Results are averaged over obfuscation strategies and detection systems.

Strategy	Datasets T1	T2	TF	G
CamelCase	.01	.02	.03	.04
Char Drop	.12	.11	.09	.13
Char Flip	.09	.11	.13	.13
Diacritics	.12	.11	.14	.14
Kebab	.17	.25	.13	.19
Leetspeak	.12	.11	.06	.11
Masking	.12	.12	.13	.14
Mathspeak	.07	.03	.05	.07
Phonetic	.13	.13	.14	.13
Snake	.12	.12	.14	.14
Spacing	.16	.21	.10	.17
Vowel Drop	.12	.10	.10	.14

Table 6: Relative change in F1 (Hate) for different obfuscation strategies. Results are averaged over target selections and detection systems.

(2022) proposes test-suite contained emoji-based hateful statements and find high vulnerability of text based models. The study also proposes adversarial examples to strengthen the model robustness. Complex obfuscation include *VIPER* (Eger, 2015) which is a probabilistic visual perturber that keep the token recognizable. Target selection has been studied in both model dependent and independent settings. Model dependent targets are either selected by looking at model architecture and its parameters (Goodfellow et al., 2014; Ebrahimi et al., 2018) or solely depend upon model output (Narodytska and Kasiviswanathan, 2017; Papernot et al., 2016; Liu et al., 2017). Among model independent targets, studies consider all the tokens in the message (Gröndahl et al., 2018; Jones et al., 2020; Eger et al., 2019). However, in real time settings, this type of target selection is not observed. In our work, we squeeze target selection to maximum of single token and perform an annotation study, that estimate real vulnerability of the models.

8 Conclusions

Previous work, simulating the obfuscation behavior of haters in a simplified way, is likely to misanalyze the real vulnerability of hate speech detection systems to obfuscation. We have shown that obfuscating all words in a post is a useful lower bound, but a very unrealistic strategy (as the communicative value of the message breaks down). Note that in this work, we deliberately limited ourselves to quite simple lexical modifications. However, detection systems still show a surprising vulnerability against these simple strategies. While it might be possible (and fairly easy) to shield a system against particularly known obfuscation strategies (e.g. by detecting K-e-b-a-b or S_n_a_k_e obfuscation with a regular expression), we need to aim for systems that are also robust against unseen strategies.

As the user study conducted in this paper shows, people have an intuitive understanding of which words are problematic. By obfuscating a single word, someone trying to obfuscate their message can impact the system performance on the same scale as when using a whitebox attack (that has the 'unfair' advantage of having access to the internal workings of the system). We also show that experiments relying on a fixed dictionary of problematic words for obfuscation are likely underestimating the impact of obfuscation on hate speech detection systems.

Acknowledgments

This work was conducted at CATALPA - Center of Advanced Technology for Assisted Learning and Predictive Analytics of the FernUniversität in Hagen, Germany.

References

Piush Aggarwal, Tobias Horsmann, Michael Wojatzki, and Torsten Zesch. 2019. LTL-UDE at SemEval-2019 Task 6: BERT and Two-Vote Classification for Categorizing Offensiveness. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*, Minneapolis USA. Association for Computational Linguistics.

Yukino Baba and Hisami Suzuki. 2012. How Are Spelling Errors Generated and Corrected? A Study of Corrected and Uncorrected Spelling Errors Using Keystroke Logs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 373–377, Jeju Island, Korea. Association for Computational Linguistics.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 759–760, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Bahman Baluch. 1992. Reading with and without vowels: What are the psychological consequences? *Journal of Social and Evolutionary Systems*, 15(1):95–104.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A Multilingual Lexicon of Words to Hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Siddhartha Brahma. 2018. Improved Sentence Modeling using Suffix Bidirectional LSTM. *arXiv preprint arXiv:1805.07340*.

Eshwar Chandrasekharan, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein, and Eric Gilbert. 2017. You Can't Stay Here: The Efficacy of Reddit's 2015 Ban Examined Through Hate Speech. *Proc. ACM Hum.-Comput. Interact.*, 1(CSCW).

Donna McKee Cleary. 1976. Reading Without Vowels: Some Implications. *Journal of Reading*, 20(1):52–56.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate Me, Hate Me Not: Hate Speech Detection on Facebook. In *ITASEC*.

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

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 (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Steffen Eger. 2015. Multiple many-to-many sequence alignment for combining string-valued variables: A G2P experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 909–919, Beijing, 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.

Katharine Gelber and Luke McNamara. 2016. Evidencing the harms of hate speech. *Social Identities*, 22(3):324–341.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics.

Fréderic Godin. 2019. *Improving and Interpreting Neural Networks for Word-Level Prediction Tasks in Natural Language Processing*. Ph.D. thesis, Ghent University, Belgium.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All You Need is "Love": Evading Hate Speech Detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, AISec '18, page 2–12, New York, NY, USA. Association for Computing Machinery.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. 2020. Towards Hierarchical Importance

Attribution: Explaining Compositional Semantics for Neural Sequence Models. In *International Conference on Learning Representations*.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust Encodings: A Framework for Combating Adversarial Typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Brendan Kennedy, Mohammad Atari, Aida Mostafazadeh Davani, Leigh Yeh, Ali Omrani, Yehsong Kim, Kris Coombs, Shreya Havaldar, Gwenyth Portillo-Wightman, Elaine Gonzalez, Joe Hoover, Aida Azatian, Alyzeh Hussain, Austin Lara, Gabriel Cardenas, Adam Omary, Christina Park, Xin Wang, Clarisa Wijaya, Yong Zhang, Beth Meyerowitz, and Morteza Dehghani. 2022. Introducing the Gab Hate Corpus: defining and applying hate-based rhetoric to social media posts at scale, journal=Language Resources and Evaluation. 56(1):79–108.

Hannah Kirk, Bertie Vidgen, Paul Rottger, Tristan Thrush, and Scott Hale. 2022. Hatemoji: A Test Suite and Adversarially-Generated Dataset for Benchmarking and Detecting Emoji-Based Hate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1352–1368, Seattle, United States. Association for Computational Linguistics.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.

György Kovács, Pedro Alonso, and Rajkumar Saini. 2021. Challenges of Hate Speech Detection in Social Media. *SN Computer Science*, 2(2).

Rohan Kshirsagar, Tyrus Cukuvac, Kathy McKeown, and Susan McGregor. 2018. Predictive Embeddings for Hate Speech Detection on Twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 26–32, Brussels, Belgium. Association for Computational Linguistics.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chanhee Lee, Young-Bum Kim, Dongyub Lee, and Heuiseok Lim. 2018. Character-Level Feature Extraction with Densely Connected Networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3228–3239, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *International Conference on Learning Representations*.

Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17, New York, NY, USA. Association for Computing Machinery.

Yann Mathet, Antoine Widlöcher, and Jean-Philippe Métivier. 2015. The Unified and Holistic Method Gamma (γ) for Inter-Annotator Agreement Measure and Alignment. *Computational Linguistics*, 41(3):437–479.

N. Narodytska and S. Kasiviswanathan. 2017. Simple Black-Box Adversarial Attacks on Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318.

Nicolas Papernot, Patrick Mcdaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *ArXiv*, abs/1605.07277.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Alina Polyakova and Chris Meserole. 2019. Exporting digital authoritarianism: The Russian and Chinese models. *Policy Brief, Democracy and Disorder Series*, pages 1–22.

Shrimai Prabhumoye, Brendon Boldt, Ruslan Salakhutdinov, and Alan W Black. 2021. Case Study: Deontological Ethics in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3784–3798, Online. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating Adversarial Misspellings with Robust Word Recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Paul Röttger, Bertie Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet Pierrehumbert. 2021. HateCheck: Functional tests for hate speech detection models. 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 41–58, Online. Association for Computational Linguistics.

Paul Röttger, Bertram Vidgen, Dong Nguyen, Zeerak Waseem, Helen Z. Margetts, and Janet B. Pierrehumbert. 2020. HateCheck: Functional Tests for Hate Speech Detection Models. *CoRR*, abs/2012.15606.

Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. 2015. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Patrick Schober, Christa Boer, and Lothar A. Schwarte. 2018. Correlation coefficients: Appropriate use and interpretation. *Anesthesia & Analgesia*, 126(5).

Justin Sherman. 2020. Digital authoritarianism and the threat to global democracy. *Bulletin of the Atomic Scientists*.

Ian C. Simpson, Petroula Mousikou, Juan Manuel Montoya, and Sylvia Defior. 2012. A letter visual-similarity matrix for Latin-based alphabets. *Behavior Research Methods*, 45(2):431–439.

Nicolas Suzor. 2010. The role of the rule of law in virtual communities. *Berkeley Technology Law Journal*, 25(4):1817–1886.

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

Franklin .S ThambiJose. 2014. Orthographic errors committed by sophomore students: A linguistic analysis. *Mediterranean Journal of Social Sciences*.

Jeremy Waldron. 2012. *The Harm in Hate Speech*. Harvard University Press.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. Inducing a lexicon of abusive words – a feature-based approach. 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 1046–1056, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2019. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018), Vienna, Austria – September 21, 2018, pages 1 – 10. Austrian Academy of Sciences, Vienna, Austria.

Nicholas Wright. 2018. How Artificial Intelligence Will Reshape the Global Order. *Foreign Affairs*.

Ziqi Zhang and Lei Luo. 2018. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter. *Semantic Web*, Accepted.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based LSTM network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256, Austin, Texas. Association for Computational Linguistics.

A Fine-Grained Results

Figure 4, 5 and 6 illustrates fine-grained results for *Distilbert* evaluated on T2, G and TF datasets. On all datasets, models are highly susceptible to *Dict Whitebox* which is considered to be expected behavior. We find large influence of static dictionaries (*Dict Fixed*) on G and TF datasets because of the availability of larger amount of obscene tokens which make more target selection. Also *Spacing* and *Kebab* are most sensitive obfuscation strategies.

B System Performance on T1 Dataset

Figure 7 illustrating the performance drop for each hate speech detection systems on applying obfuscation attacks across the targets. We find the consistency in the order of vulnerabilities. We also find that the drop is proportional to model's performance.

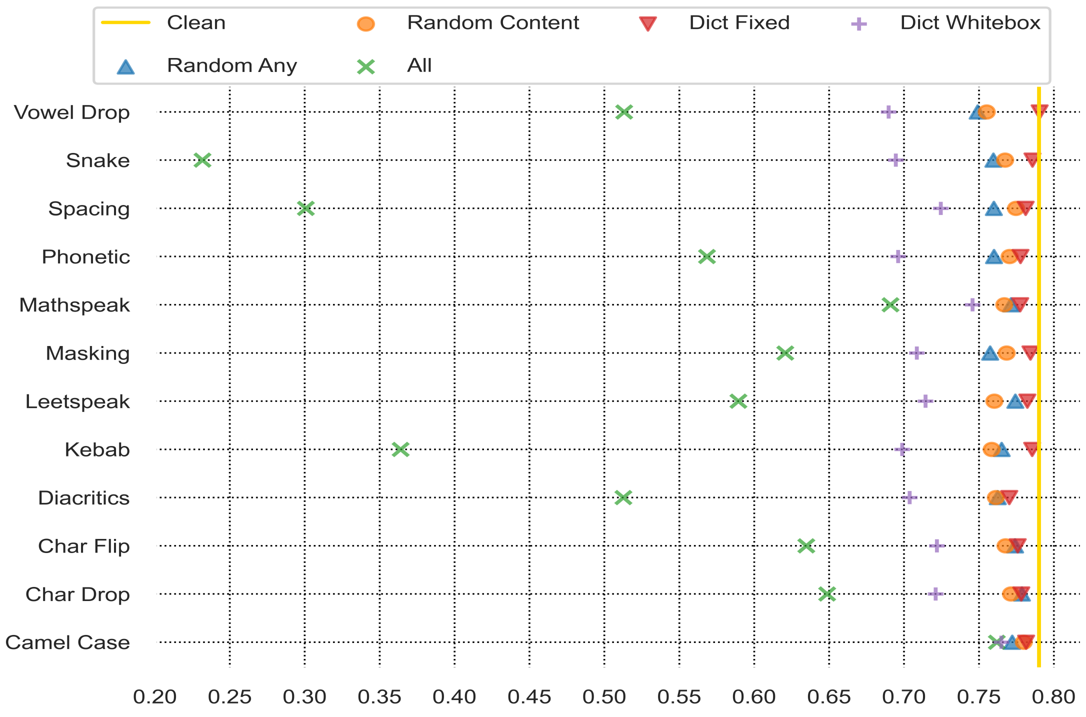

Figure 4: Distilbert's F1 (Hate label) performance on T2 dataset.

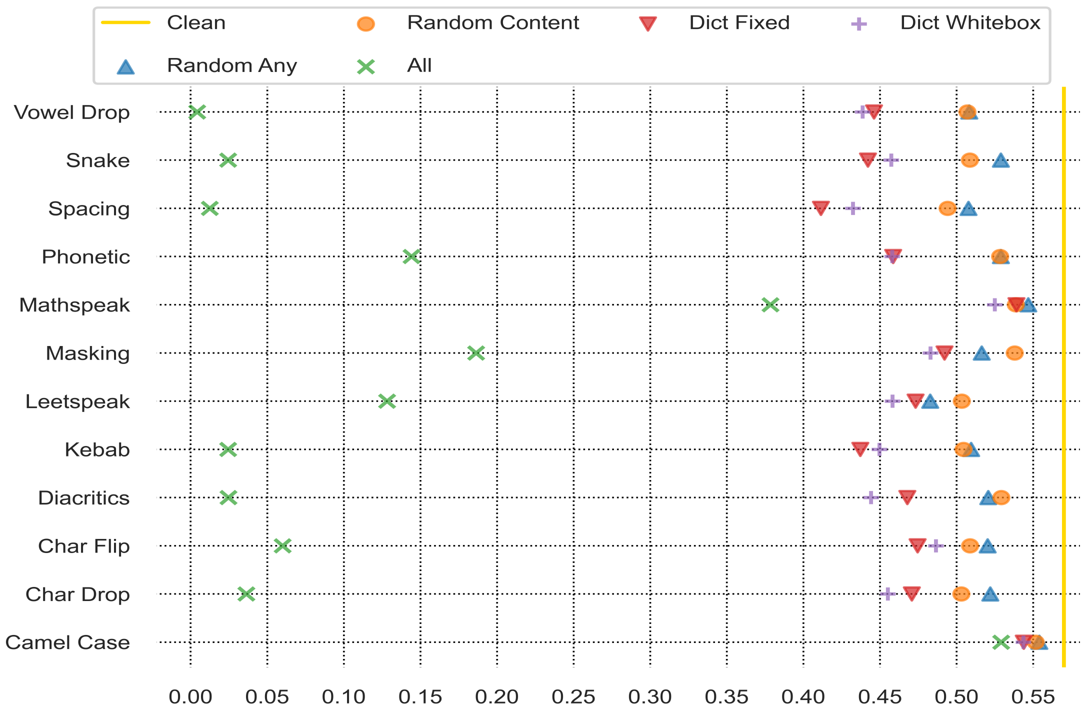

Figure 5: Distilbert's F1 (Hate label) performance on G dataset.

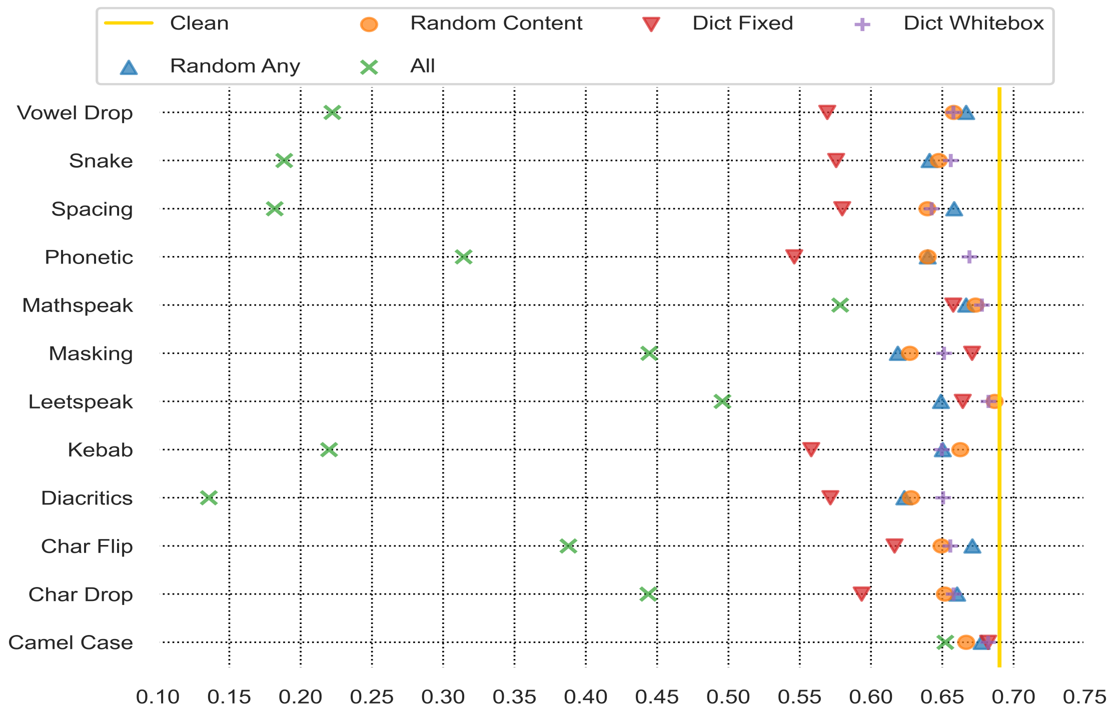

Figure 6: Distilbert's F1 (Hate label) performance on TF dataset.

	Hate Detection Systems									
Target	**Distilbert**	**BILSTM**	**CNN-ATT**	**CNN-LSTM**	**LSTM**	**AdaBoost**	**GradB**	**LogReg**	**RF**	**SVM**
All	.34	.21	.07	.21	.20	.08	.18	.13	.10	.25
Dict Domain	.27	.15	.05	.18	.17	.07	.13	.11	.09	.18
Dict Whitebox	.29	.17	.05	.18	.18	.07	.12	.10	.08	.21
Dict Fixed	.17	.14	.05	.14	.12	.07	.10	.11	.09	.17
Random Content	.06	.06	.03	.07	.04	.03	.05	.05	.04	.08
Random Any	.04	.05	.03	.04	.03	.02	.04	.03	.03	.07

Table 7: Relative change in F1 (Hate) performance for each system estimated on the T1 dataset for different obfuscation strategies. Results are averaged over target selections. GradB, LogReg and RF is abbreviated for *Gradient Boosting*, *Logistic Regression* and *Random Forest* respectively.