

LexObf-12: Measuring and Hardening the Obfuscation Robustness of a Production Lexical Content Filter

Tafadzwa Innocent Tauro, M.S. Cybersecurity, Quinnipiac University

Abstract—Lexical filters remain the first stage of content moderation on many platforms because they are cheap, deterministic, and explainable, yet users evade them with substituted characters, inserted spaces, and look-alike Unicode letters. We introduce LexObf-12, a benchmark that applies twelve obfuscation classes to a filter’s own lexicon in four carrier sentences, and use it, with two public labeled corpora and a list of common English words, to evaluate the pre-publication text filter of Kibhi (kibhi.com), a production short-video and messaging platform. The evaluation exposed two defects in the deployed normalizer: trailing punctuation was read as character substitution, so any listed phrase followed by “!” passed, and repeated-letter collapsing erased one listed term. Look-alike Cyrillic letters and zero-width characters defeated it entirely. After correction and hardening, mean detection across the twelve classes rose from 61.9% to 99.3%, against 17.7% and 28.7% for token and substring baselines, at 65 μ s per check and with no change in false positives on 24,783 labeled tweets (1.95%) or among 10,000 common words. A leave-one-out ablation attributes the gain to each step and shows that only plural matching affects false positives. On 19,229 HateXplain posts, however, the filter flagged 12.0% of posts that annotators labeled normal, mostly through slur matches, and about 90% of the hate posts it missed contained no listed term at all. We conclude that normalization should be tested adversarially, that the limit of a word list is coverage rather than matching, and that slur matches should be held for human review rather than blocked. The benchmark generator is released.

Index Terms—Content moderation, adversarial robustness, text normalization, Unicode security, hate speech, benchmark, trust and safety.

I. Introduction

EVERY platform that hosts user content needs a way to keep the worst of it from being seen. Learned classifiers now do much of that work at large platforms [5], but word lists remain the first stage almost everywhere: a check costs microseconds, its decision can be explained to the person affected, and an operator can add a new slur or scam phrase in minutes. Their weakness is equally well known. Misspellings, inserted spaces, and character substitutions defeat both lexical filters and learned classifiers [6], [7], and Unicode contains characters that look identical to Latin letters [8].

Defenders respond with normalization: before matching, the filter undoes common disguises. Each step is simple,

T. I. Tauro is an independent researcher in Connecticut, USA, and received the M.S. degree in cybersecurity from Quinnipiac University, Hamden, CT, USA, in 2026 (e-mail: tafadzwatauro09@gmail.com).

The author designed and operates the Kibhi platform evaluated in this article.

but the steps interact with each other, with ordinary punctuation, and with the lexicon, and hand-written unit tests rarely exercise those interactions. This article asks how robust a deployed normalizing filter actually is, what each normalization step contributes, and where the remaining errors come from. We make five contributions:

- 1) LexObf-12, a lexicon-relative benchmark of twelve obfuscation classes in four carriers that measures robustness of normalization independently of lexicon coverage (Section IV).
- 2) An evaluation of three versions of a production filter against two baselines, two labeled corpora, and the 10,000 most common English words (Sections V and VI).
- 3) A case study of two defects and two missing defenses that the benchmark found in the deployed filter, and their fixes (Section VII).
- 4) A leave-one-out ablation that attributes detection and false positives to individual normalization steps (Section VI).
- 5) An error analysis showing that misses are dominated by lexicon coverage and false positives by slur matches, with the design implication that such matches should be held for review rather than blocked (Sections VI and VIII).

II. Related Work

A. Evading text classifiers

Hosseini et al. deceived Google’s Perspective API by misspelling toxic words or inserting punctuation between their letters [6]. Gröndahl et al. defeated state-of-the-art hate speech classifiers with inserted whitespace, removed word boundaries, and appended innocuous words [7]. Unicode Technical Standard #39 catalogs confusable characters and defines mixed-script detection as a defense against spoofing [8].

B. Benchmarks and functional tests

Davidson et al. released tweets labeled as hate speech, offensive language, or neither, and showed that lexical methods conflate the first two [9]. HateXplain provides about 20,000 posts from Twitter and Gab with three-annotator labels, targets, and rationales [10]. HateCheck introduced functional tests, 3,728 hand-crafted cases in 29 tests, that probe specific capabilities such as negation,

counter-speech, and reclaimed slurs [11]. LexObf-12 is complementary: it is generated from the filter’s own lexicon, so it isolates robustness to disguise from coverage of vocabulary, and it can be regenerated for any list.

C. Bias and governance

Sap et al. found that hate speech datasets and models disproportionately flag tweets written in African American English [12], and Kurrek et al. showed that many uses of slurs are not derogatory [13], a distinction no word list can make. Gillespie argues that moderation is central to what platforms are [14]; the Santa Clara Principles call for notice and appeal for every action [15]; and Jhaver et al. found that explaining removals on Reddit was associated with fewer future removals [16].

D. Recent developments

The balance between over- and under-enforcement has become a public question. In January 2025 Meta ended its third-party fact-checking program in the United States in favor of community notes and revised its hateful conduct policy, arguing that its moderation systems had made too many mistakes and removed too much legitimate speech [2]. Regulators push in the other direction for some harms and toward procedural fairness for all: Article 17 of the EU Digital Services Act requires platforms to give users a statement of reasons for restrictions on their content [3], and since 25 July 2025 Ofcom’s codes under the U.K. Online Safety Act have required services likely to be accessed by children to moderate content harmful to them [4]. Both pressures make it more important to measure, rather than assume, how often an automated first stage blocks legitimate speech; Section VI does so.

III. The Filter and Its Place in the Pipeline

Kibhi (<https://kibhi.com>) [1] is a short-video and messaging platform for users aged 13 and over, built and operated by the author, with web and Android clients sharing one API. Fig. 1 shows where the text filter sits. It screens posts, captions, comments, and profiles before they are stored, and refuses those that fail with a message naming the category; private messages are held only to the hate and threat categories. Photos and videos pass through an image classifier. Content that passes is subject to user reports, deterministic report triage, and a human review queue that can limit the reach of a post or hide it, with notice and appeal. The security of the moderation console is analyzed separately; here we study the first stage.

The lexicon has 61 entries in three categories: sexual content and solicitation, hate, and threats. Matching is on whole words and phrases, with plurals, after normalization. The fully hardened version (v2) normalizes in seven steps, each of which can be disabled for the ablation: (1) remove zero-width and soft hyphen characters; (2) lowercase, apply Unicode compatibility decomposition, and strip combining marks; (3) map Cyrillic and Greek look-alike letters to Latin, only in words that mix scripts

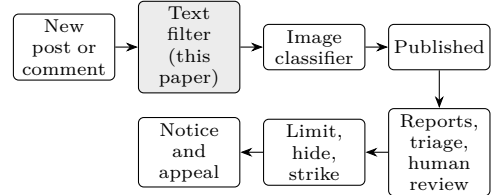


Fig. 1. The text filter is the first, pre-publication stage of a layered pipeline; everything it lets through remains subject to reports and human review.

TABLE I
LexObf-12 obfuscation classes, shown on the neutral word “spam”.

Class	Example	Note
Plain	spam	
Uppercase	SPAM	
Plural	spams	
Leet, all letters	5p4m	a, e, i, o, s, t as 4, 3, 1, 0, 5, 7
Leet, one letter	5pam	first eligible letter
Dotted	s.p.a.m	
Spaced	s p a m	
Repeated letter	spaaam	last vowel tripled
Diacritics	spám	
Cyrillic look-alikes	spam	Cyrillic letters for p, a
Zero-width joiners	s · p · a · m	· marks U+200B
Leet and repeated	5p444m	

or consist entirely of look-alikes, so ordinary Russian or Greek text is untouched; (4) translate digits and symbols used as letters, only inside a run of word characters; (5) join letters separated by dots, dashes, or spaces; (6) reduce runs of three or more identical letters, comparing the text reduced to one letter, to two, and not at all; and (7) accept plural forms. Operators can add terms at run time.

IV. The LexObf-12 Benchmark

LexObf-12 applies each of twelve transformations (Table I) to every lexicon entry and embeds the result in four carriers: the bare term, “omg _ lol”, “hey, _!”, and “_ right now”. A transformation that leaves a term unchanged is skipped, so each class has between 224 and 244 cases for our 61-entry lexicon. Because the cases are derived from the filter’s own list, a perfect normalizer scores 100% however complete the list is; LexObf-12 therefore measures robustness to disguise, not coverage. The generator is a few dozen lines and can be pointed at any lexicon.

V. Evaluation Setup

A. Filter versions and baselines

We evaluate three versions of the production filter, each imported unchanged from the source tree: v0 as deployed before this study; v1 with the two defects of Section VII fixed; and v2, which adds zero-width removal, look-alike mapping, and reduction of repeats to two letters. Two baselines use the same lexicon: Token lowercases the text, splits it on non-letters, and matches whole words and phrases; Substring reports any lexicon entry occurring anywhere in the lowercased text.

TABLE II
Detection rate (%) on LexObf-12.

Class	<i>n</i>	v0	v1	v2	Token	Substr.
Plain	244	74.2	100.0	100.0	100.0	100.0
Uppercase	244	74.2	100.0	100.0	100.0	100.0
Plural	244	73.8	100.0	100.0	1.6	100.0
Leet, all letters	236	75.4	96.6	96.6	1.7	1.7
Leet, one letter	236	75.4	98.3	98.3	6.8	6.8
Dotted	244	75.4	100.0	100.0	0.0	0.0
Spaced	244	75.4	100.0	100.0	0.0	0.0
Repeated letter	232	71.6	94.8	100.0	0.0	34.5
Diacritics	232	75.4	100.0	100.0	0.0	0.0
Cyrillic look-alikes	224	0.0	0.0	100.0	0.0	0.0
Zero-width joiners	244	0.0	0.0	100.0	0.0	0.0
Leet and repeated	240	71.7	91.7	96.7	1.7	1.7
Mean of classes		61.9	81.8	99.3	17.7	28.7

B. Data

Davidson: 24,783 tweets labeled hate speech (1,430), offensive (19,190), or neither (4,163) [9]. HateXplain: 20,148 posts, of which 19,229 have a majority label among three annotators: hate (5,935), offensive (5,480), and normal (7,814); 8,806 come from Twitter and 10,423 from Gab [10]. Common words: the 10,000 most frequent English words from the Google Web Trillion Word Corpus [17], each embedded in “I like _ a lot”, excluding words that are lexicon entries. For the labeled corpora, treating hate and offensive as positive, we report the fraction of each class flagged and precision; the fraction of benign posts flagged serves as the false positive rate.

C. Environment

Timing uses one core of an AMD Ryzen 7 8840HS with Node.js 22.13.1, over all 24,783 Davidson tweets after a warm-up. No text from either corpus is reproduced in this article.

VI. Results

A. Robustness to obfuscation

Table II gives detection rates on LexObf-12. The token baseline catches plain and uppercased terms and almost nothing else; the substring baseline adds plurals and some repeated letters. Version v0 caught about three quarters of most classes and none of the look-alike or zero-width cases. Version v2 catches every case in nine classes and at least 96.6% in the other three: a mean of 99.3%, against 61.9% for v0.

All remaining v2 misses come from two entries: one that becomes a token made only of digits when every letter is substituted, and a phrase whose first word becomes a lone “1”. The filter deliberately leaves tokens without letters as numbers so that text such as “I got 100%” is not rewritten, trading these cases for precision.

B. Davidson corpus

Table III shows that v2 flags 57.3% of hate tweets and 1.95% of tweets labeled neither, with precision of

TABLE III
Fraction flagged on the Davidson corpus (%).

Method	Hate	Offensive	Neither	Precision
v0	55.94	22.42	1.95	98.44
v2	57.34	22.74	1.95	98.46
Token	41.61	19.30	1.56	98.51
Substring	61.26	26.19	3.84	97.36

TABLE IV
Fraction flagged on HateXplain (%), majority labels.

Method	Source	Hate	Offensive	Normal	Precision
v2	All	57.81	21.11	11.98	83.06
v2	Twitter	50.14	21.78	13.73	52.12
v2	Gab	58.85	20.62	7.05	96.28
Token	All	47.40	15.75	8.82	84.22
Substring	All	62.06	32.72	14.78	82.58

98.46%. Hardening raised hate detection from 55.9% to 57.3% without changing the false positive rate. The token baseline flags fewer benign tweets but misses many more hate tweets; the substring baseline catches slightly more hate at about twice the false positive rate.

C. HateXplain

HateXplain tells a less favorable story (Table IV). Detection of hate posts is similar (57.8%), but v2 flags 12.0% of posts labeled normal, and on the Twitter portion, where normal posts dominate, precision falls to 52.1%. Of the 936 normal posts flagged, 795 matched the hate list, 83 the sexual list, and 58 the threat list. The normal class of HateXplain evidently contains many posts that use slurs without being judged hateful by its annotators; a word list cannot tell such in-group, reclaimed, or quoted uses from abuse [13], and its errors fall on the communities that use these words among themselves [12].

D. Common words and cost

The substring baseline flagged 17 of the 10,000 common words, nine of them innocent words containing a listed term (document and four variants, circumstances, cumulative, cocktail, spice). Version v2 flagged seven, all plurals of lexicon entries and therefore intended matches, with one arguable case (escorts), and no innocent word containing a listed term. Each v2 check took 64.9 μ s on average (median 58.1, 99th percentile 135.4), against 8.3 for the token baseline and 3.7 for the substring baseline; the hardened filter normalizes each text three ways, which is negligible next to the network and database cost of the request that triggers it.

E. Ablation

Table V disables one normalization step at a time. Every step contributes to robustness: removing in-word substitution costs the most (99.3% to 76.0%), followed by repeat reduction and joining of spelled-out letters, and

TABLE V
Leave-one-out ablation of v2 (%).

Variant	LexObf-12 mean	Benign flagged Davidson	HX HX	hate flagged
Full v2	99.3	1.95	11.98	57.81
Without zero-width	91.0	1.95	11.98	57.81
Without diacritics	91.0	1.95	11.98	57.81
Without look-alikes	91.0	1.95	11.98	57.81
Without substitution	76.0	1.95	11.98	57.81
Without letter joining	82.6	1.95	11.98	57.81
Without repeats	83.2	1.95	11.98	57.81
Without plurals	91.1	1.56	8.82	47.40
No normalization	17.6	1.56	8.82	47.40

each Unicode step removes exactly the class it targets. No step except plural matching changes the false positive rate on either corpus. Plural matching is a genuine trade-off: on HateXplain it adds 10.4 points of hate detection and 3.2 points of false positives. With every step disabled the filter reduces to the token baseline (17.6%).

F. Error analysis

Why does a hardened filter still miss about 42% of hate posts? For each missed post we asked whether any lexicon entry occurs anywhere in it, even inside a longer word. On Davidson, 554 of 610 missed hate tweets (90.8%) contain no listed term at all; on HateXplain, 2,252 of 2,504 (89.9%). The remaining misses contain a term only inside a longer word, where whole-word matching rejects it by design. Normalization is therefore no longer the bottleneck; what a word list cannot see is hate expressed without its vocabulary, which is the territory of learned classifiers and human review.

VII. Case Study: Defects Found by the Benchmark

Punctuation read as substitution. Version v0 mapped “!” to “i” and “\$” to “s” everywhere, to catch spellings such as “k!ll”. A listed phrase followed by an exclamation mark therefore became an unlisted word ending in “ii”. In the “hey, _!” carrier, v0 detected 1 of 61 entries, which accounts for almost all of its misses in Table II. The fix applies substitutions only inside a run of word characters and treats symbols at the edge of a word as punctuation.

Repeat reduction erased a term. Reducing every run of three identical letters to one turned a three-letter listed term made of one repeated letter into a single letter, so it was never matched. The fix compares the text with and without reduction.

Unicode. Look-alike Cyrillic letters and zero-width characters defeated every method tested, including v0. Mapping look-alikes only in mixed-script or all-look-alike words preserves ordinary Cyrillic and Greek text, which regression tests now check.

All of these passed the hand-written unit tests of the filter. Each normalization step was reasonable on its own; the failures came from their interaction with ordinary punctuation and with the lexicon. Generating cases systematically by class and carrier found them

TABLE VI
Pipeline against the operational Santa Clara Principles.

Principle	Status	Mechanism
Notice	Met	Refusals name the category; limits and removals notify the creator by push and email with the rule
Appeal	Met	Public appeal form; a person decides; the decision is sent to the user
Numbers	Not met	No public transparency report yet

immediately, and each fix is now covered by a regression test.

VIII. Discussion

Hold slur matches for review rather than blocking them. The filter refuses publication outright. That is appropriate for sexual solicitation and explicit threats, where false positives are rare and the harm of delay is high, but the HateXplain results show that slur matches carry a substantial false positive rate concentrated on in-group speech. A better design routes matches in the hate category to the human review queue, holding the post until a moderator sees it, and keeps outright refusal for the other categories. The ablation offers a second lever: plural matching can be restricted to the categories where its false positives are low.

Notice, appeal, and transparency. Table VI maps the surrounding pipeline to the operational Santa Clara Principles [15], whose notice principle corresponds to the statement of reasons required by Article 17 of the Digital Services Act [3]. Refusals name the category, enforcement actions name the rule and the appeal path, and appeals are decided by a person; Kibhi does not yet publish enforcement numbers.

Limitations. The lexicon has 61 English entries, so LexObf-12 measures normalization, not coverage, and results on other lexicons and languages may differ. The look-alike map covers common Cyrillic and Greek letters, not the full confusables table [8], and not confusables within the Latin script. Both corpora reflect the platforms, years, and annotators that produced them [12]. We did not evaluate the image classifier, the triage rules, or moderator decisions.

Ethics. Only public datasets released under the MIT License were used, no user data from Kibhi was analyzed, and no offensive text is reproduced in this article; examples use a neutral word.

IX. Conclusion

A lexical filter is only as strong as its normalization, and normalization steps interact in ways that hand-written tests miss. LexObf-12 found two defects and two missing defenses in a deployed filter; fixing them raised mean detection across twelve obfuscation classes from 61.9% to 99.3% without increasing false positives on labeled tweets or common words. The ablation shows each step earns its place, the error analysis shows the

remaining misses are a matter of vocabulary rather than matching, and HateXplain shows that slur matches should be reviewed by a person rather than refused automatically. The benchmark generator and evaluation code are released so that any platform can run the same test on its own list.

Acknowledgment

The author thanks the creators of the Davidson and HateXplain corpora for releasing them under the MIT License. In accordance with IEEE policy, the author discloses that an AI system (Claude, Anthropic) was used to help draft the text of all sections, to write the evaluation script in docs/papers/eval, to implement the filter changes evaluated as v1 and v2, and to format tables and figures. The author designed the study, reviewed every change, verified every reported number against the result files, and takes full responsibility for the content.

References

- [1] Kibhi, “Kibhi: short video and messaging,” 2026. [Online]. Available: <https://kibhi.com>
- [2] J. Kaplan, “More speech and fewer mistakes,” Meta Newsroom, Jan. 7, 2025. [Online]. Available: <https://about.fb.com/news/2025/01/meta-more-speech-fewer-mistakes/>
- [3] Regulation (EU) 2022/2065 of the European Parliament and of the Council on a Single Market for Digital Services (Digital Services Act), Off. J. Eur. Union, L 277, Oct. 27, 2022.
- [4] Ofcom, “Protection of Children Codes of Practice,” Online Safety Act 2023, in force Jul. 25, 2025. [Online]. Available: <https://www.ofcom.org.uk/online-safety/protecting-children>
- [5] R. Gorwa, R. Binns, and C. Katzenbach, “Algorithmic content moderation: Technical and political challenges in the automation of platform governance,” *Big Data & Society*, vol. 7, no. 1, 2020.
- [6] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, “Deceiving Google’s Perspective API built for detecting toxic comments,” arXiv:1702.08138, 2017.
- [7] T. Gröndahl, L. Pajola, M. Juuti, M. Conti, and N. Asokan, “All you need is ‘love’: Evading hate speech detection,” in *Proc. 11th ACM Workshop Artif. Intell. Secur. (AISec)*, 2018.
- [8] Unicode Consortium, “Unicode Technical Standard #39: Unicode security mechanisms.” [Online]. Available: <https://www.unicode.org/reports/tr39/>
- [9] T. Davidson, D. Warmesley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” in *Proc. Int. AAAI Conf. Web Soc. Media (ICWSM)*, 2017.
- [10] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee, “HateXplain: A benchmark dataset for explainable hate speech detection,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 17, 2021, pp. 14867–14875.
- [11] P. Röttger, B. Vidgen, D. Nguyen, Z. Waseem, H. Margetts, and J. Pierrehumbert, “HateCheck: Functional tests for hate speech detection models,” in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2021.
- [12] M. Sap, D. Card, S. Gabriel, Y. Choi, and N. A. Smith, “The risk of racial bias in hate speech detection,” in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2019.
- [13] J. Kurrek, H. M. Saleem, and D. Ruths, “Towards a comprehensive taxonomy and large-scale annotated corpus for online slur usage,” in *Proc. 4th Workshop Online Abuse Harms (WOAH)*, 2020.
- [14] T. Gillespie, *Custodians of the Internet: Platforms, Content Moderation, and the Hidden Decisions That Shape Social Media*. New Haven, CT, USA: Yale Univ. Press, 2018.
- [15] “The Santa Clara Principles on transparency and accountability in content moderation,” 2nd ed., 2021. [Online]. Available: <https://santaclaraprinciples.org>
- [16] S. Jhaver, A. Bruckman, and E. Gilbert, “Does transparency in moderation really matter? User behavior after content removal explanations on Reddit,” *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, 2019.
- [17] J. Kaufman, “google-10000-english,” derived from T. Brants and A. Franz, *Web 1T 5-gram Version 1*, Linguistic Data Consortium, 2006. [Online]. Available: <https://github.com/first20hours/google-10000-english>

Tafadzwa Innocent Tauro received the B.Comm. degree in information systems from Midlands State University, Gweru, Zimbabwe, in 2022, and the M.S. degree in cybersecurity from Quinnipiac University, Hamden, CT, USA, in 2026. Tauro holds the CompTIA Security+ certification and previously worked in information technology roles at the National Railways of Zimbabwe and at Zimbabwe’s Ministry of Lands, Agriculture, Fisheries, Water and Rural Development. Tauro designed, built, and operates Kibhi, a short-video and messaging platform for users aged 13 and over. Research interests include trust and safety engineering, child online safety, privacy, and application security.