INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Unmasking Hidden Hate: Techniques for Detecting and Normalizing Obfuscated Offensive Language

Luís Miguel dos Santos Pereira

Master in Data Science

Supervisors:
Doctor Diana Elisabeta Aldea Mendes, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Doctor Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

September, 2025

BUSINESS SCHOOL

TECNOLOGIAS E ARQUITETURA

Department of Quantitative Methods for Management and Economics
Department of Information Science and Technology

Unmasking Hidden Hate: Techniques for Detecting and Normalizing Obfuscated Offensive Language

Luís Miguel dos Santos Pereira

Master in Data Science

Supervisors:
Doctor Diana Elisabeta Aldea Mendes, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Doctor Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

September, 2025

Resumo

À medida que mais pessoas recorrem a espaços online para interagir, as preocupações com o surgimento de interações tóxicas tornam-se cada vez mais evidentes. A moderação manual é morosa e psicologicamente desgastante, o que acentua a necessidade de soluções automatizadas. Contudo, é escassa a investigação sobre formas de lidar com linguagem ofensiva ofuscada, frequentemente utilizada para contornar sistemas de deteção. Esta dissertação explora métodos para identificar palavras ofuscadas e normalizá-las. Para tal, os tipos de ofuscação são categorizados, facilitando a deteção e a recuperação da forma original. Foi desenvolvido um *pipeline* de limpeza de palavras ofuscadas. Este lida com ofuscações que usam homóglifos e *leetspeak* através de abordagens baseadas em regras, comparando os termos com um léxico ofensivo por meio de similaridade. Os *Character Architecture with No tokenization In Neural Encoders* (CANINE) *embeddings* constituem o método de representação de palavras, com a similaridade medida através de *cosine distance*, sendo complementadas por similaridade fonética e *edit distance* para atribuição das formas canónicas. Como experiência, o ChatGPT-4o foi utilizado para gerar publicações ofuscadas sintéticas e normalizar os dados originais. A aplicação do pipeline aos dados originais permitiu normalizar 49,4% das palavras, deixando 50,6% por tratar. Já o ChatGPT-4o alcançou 76,9% de normalização, com 23,1% sem resolução. O *pipeline* obteve melhor desempenho nos dados gerados, com 64,9% tratados e 35,1% não tratados. Estes resultados demonstram que, embora os *Large Language Models* apresentem um desempenho superior, melhorias no isolamento de palavras *out-of-vocabulary* e na cobertura do léxico podem aumentar significativamente a eficácia do *pipeline*.

Palavras-chave: Linguagem ofensiva ofuscada, deteção de discurso de ódio, processamento de linguagem natural, normalização de texto, moderação de conteúdo

Abstract

As more people join online spaces to interact with other users, concerns about the emergence of toxic interactions continues to grow. Manual moderation is time-consuming and psychologically taxing, highlighting the need for automated solutions. There is a lack of research regarding ways to deal with obfuscated offensive language, used to indulge in toxicity while avoiding automated detection systems. In this thesis, methods to isolate these words and normalize them are explored. The obfuscation types are categorized to facilitate the process of detecting and recovering the original word. For that end, a word cleaning pipeline, to deal with obfuscated words, was conceived. The pipeline handles homoglyph and leetspeak obfuscations using rule-based approaches, then compares words against an offensive lexicon via similarity scoring. Character Architecture with No tokenization In Neural Encoders (CANINE) embeddings serve as the word representation method, with similarity measured through cosine distance, complemented by phonetic similarity and edit distance for assigning canonical forms. As an experiment, ChatGPT-4o was used to generate synthetic obfuscated posts and to normalize the original data. Applying the pipeline to the original data resulted in 49.4% words being normalized while 50.6% were not. In comparison, ChatGPT-4o normalized 76.9% of the words, with 23.1% remaining unresolved. The pipeline had a better performance applied to the generated data, 64.9% were treated and 35.1% left untreated. These results indicate that, while Large Language Models achieve higher overall performance, improvements in out-of-vocabulary words isolation and lexicon coverage can significantly enhance the pipeline's effectiveness.

Keywords: Obfuscated offensive language, Hate speech detection, Natural Language Processing, Text normalization, Content moderation

Contents

Resumo i

Abstract iii

Glossary vii

List of Acronyms ix

List of Figures xi

List of Tables xiii

Chapter 1. Introduction 1
- 1.1. Objectives and Research Questions 2
- 1.2. Document Structure 2

Chapter 2. Literature Review 5
- 2.1. Methodology 5
- 2.2. Results and Discussion 8
- 2.3. Summary 16

Chapter 3. Methodology and Data 19
- 3.1. Data Sources and Collection 20
- 3.2. Data Description 22
- 3.3. Data Preparation 23
- 3.4. Data Analysis 30

Chapter 4. Modeling 35
- 4.1. Embedding-based Approaches 35
- 4.2. Edit Distance Approaches 36
- 4.3. Phonetic Similarity Approaches 37
- 4.4. Word Cleaning Pipeline 37
- 4.5. RoBERTa Toxicity Classifier Experiment 40
- 4.6. Experiments with Large Language Models 40

Chapter 5. Results and Discussion 43
- 5.1. Models and embeddings 43
- 5.2. Edit Distance 47
- 5.3. Phonetic Similarity 48

5.4. Word Cleaning Pipeline 49
5.5. Experiments 51

Chapter 6. Conclusions 57

Bibliography 61

Appendix 69

Glossary

- **Common Crawl**: open repository of web crawl data
- **Cosine similarity**: measure of similarity between two non-zero vectors defined in an inner product space.
- **Gab**: American alt-tech microblogging and social networking service known for its far-right userbase
- **Reddit**: American proprietary social news aggregation and forum social media platform.
- **Stan communities**: community of extremely or excessively enthusiastic and devoted fans who post opinions on celebrities, music, TV shows, movies, video games, social media, and other topics.
- **Synsets**: group of one or more synonyms that share a common meaning in some context.
- **Wikipedia**: free online encyclopedia

List of Acronyms

- **BERT**: Bidirecional Encoder Representations from Transformers
- **CANINE**: Character Architecture with No tokenization In Neural Encoders.
- **GPT**: Generative Pre-trained Transformer
- **LLM**: Large Language Model
- **NLP**: Natural Language Processing
- **NNP**: Proper noun, singular
- **OCR**: Optical Character Recognition
- **OOV**: Out-of-Vocabulary
- **POS**: Part-of-speech
- **RoBERTa**: Robustly Optimized BERT Pretraining Approach

List of Figures

2.1 Data retrieval processs. 7

3.1 Distribution of obfuscation types. 31

3.2 Distribution of obfuscation types. 32

3.3 Co-occurrence patterns between the different obfuscation types. 33

3.4 Frequency of word-level Levenshtein distances. 33

4.1 Pipeline for the word cleaning process to restore obfuscated words to their canonical form. 38

5.1 Total technique applications. 50

5.2 Total cases final status. 51

5.3 Distribution of obfuscation types (generated data). 52

5.4 Co-occurrence patterns between the different obfuscation types (generated data). 53

5.5 Frequency of word-level Levenshtein distances (generated data). 53

5.6 Total technique applications (generated data). 54

5.7 Total cases final status (generated data). 55

List of Tables

2.2 Literature summarization and mapping to aspects. 8

2.1 Literature characterization by Year, Type and Citations (sorted from oldest to newest). 15

3.1 Results of the assessment and filtering process to retain suitable datasets. 21

3.2 Structure of the data from *A Benchmark Dataset for Learning to Intervene in Online Hate Speech by Jing (2020)* 22

3.3 Structure of the data from *Automated Hate Speech Detection and the Problem of Offensive Language by Davidson (2017).* 22

3.4 Structure of the *Toxicity Dataset* by SurgeAI (2021b) 23

3.5 Structure of the *English Profanity Lexicon* by SurgeAI (2021a) 23

3.6 Structure of the Collected_Obfuscations dataset 24

3.7 Total occurences per unigram subset in the Google Books Ngram Corpus (Version 2). 26

3.8 Structure of the aggregated Google Books Ngram 1-gram dataset. 26

3.9 Distribution of words by frequency categories in the aggregated unigram dataset. 26

3.10 Frequency of numbers, symbols, and homoglyphs among OOV tokens in different datasets, including total count. 28

3.11 Distribution of obfuscated instances across all datasets. 28

3.12 Structure of the final dataset. 29

5.1 Cosine similarity between *faggot* and obfuscated variants using FastText embeddings. 43

5.2 Top-10 most similar words to *faggot* according to the FastText pre-trained word vectors. 44

5.3 Top-10 most similar words to *f4ggot* according to the FastText pre-trained word vectors. 44

5.4 Top-10 most similar words to *faggg00tt* according to the FastText pre-trained word vectors. 44

5.5 Cosine similarity of obfuscated words to *faggot* using CharacterBERT embeddings. 45

5.6 Top-10 most similar words to *faggg00tt* according to the CANINE embeddings. 45
5.7 Top-10 most similar words to *faggg00tt* according to the BPEmb model. 46
5.8 Toxicity classification probabilities of sample texts using RoBERTa Toxicity Classifier. 46
5.9 Top-10 most similar words to *faggg00tt* according to Levenshtein distance. 47
5.10 Top-10 most similar words to *faggg00tt* according to Damerau-Levenshtein distance. 48
5.11 Top-10 most similar words to *faggg00tt* according to Jaro-Winkler distance. 48
5.12 Top-10 most similar words to *phaggg00tt* according to Jaro-Winkler distance. 49
5.13 Top-10 most similar words to *phaggg00tt* according to Damerau-Levenshtein distance. 49
5.14 DoubleMetaphone codes and phonetic similarity of obfuscated words to *fucker*. 49

CHAPTER 1

Introduction

With the rise of the digital age, individuals find themselves interacting with each other online more than ever. Naturally, concerns are raised about behavior and societal norms within online communities, since the lack of supervision could lead to the creation of toxic environments. Moderation of digital spaces plays a key role in ensuring user safety against psychological and emotional abuse, particularly for vulnerable demographics at higher risk (Chan & Li, 2024). Given the time-consuming nature of manual moderation and its detrimental psychological effects on the moderator, automation of this process is highly desirable.

In gaming and social media environments, users often feel inclined to express their feelings and opinions without fear of accountability or judgement. This is largely due to the degree of anonymity granted to users. Social psychological research has shown that people tend to behave more aggressively and violently than they would under identifiable conditions (K. Zhang & Kizilcec, 2014). There are special cases within these environments, such as competitive online games and stan communities, where incidents of hateful behavior and harassment are more frequent. To moderate these communities, service providers rely on reporting systems, which enable users to flag inappropriate behavior for review by a moderator. When a violation is confirmed, it results in a penalty for the abusive user (Adinolf & Turkay, 2018). However, these reports are now commonly reviewed by automated systems/machine learning algorithms. The task performed to handle such cases can be modeled as a supervised task, trained on annotated data where abusive or offensive behavior is present (Zampieri et al., 2019) or an unsupervised task leveraging transformer-based language models like Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019; Liu et al., 2019). Furthermore, directly implementing Natural Language Processing (NLP) models may not be a suitable approach for real-world toxicity detection, since the challenging environment in industries such as gaming and social networks, where harmful content filtering is mandatory, exposes the models to constant adversarial attacks and social engineering, putting their robustness to the test (Bespalov et al., 2023).

Adversarial attacks in NLP consist in modifications to a given textual input, in a way that preserves its original semantics while affecting the model's prediction (Colombo et al., 2023). Although adversarial attacks can occur at the sentence, word, or character level, this research exclusively addresses the type most often overlooked, character-level attacks. Contrary to automated solutions, humans can rely on visual similarity to assess the meaning behind a perturbed input, such as "n00b" and "pi§§" (Eger et al., 2019).

These types of attacks pose a significant challenge to offensive language and hate speech detection models, often allowing abusers to evade moderation entirely and continue their inappropriate conduct. Failure to identify and deal with such offenders can undermine the trust of community members who submit reports, giving the impression that automated solutions are unreliable. Moreover, such failures may also embolden offenders' belief that their behavior cannot be controled, normalizing abusive conduct to the point that engaging in it becomes second nature, without fear of consequences.

1.1. Objectives and Research Questions

Motivated by the concerns previously addressed, this study aims to bridge the gaps in current research related to character-level attacks used to fool NLP systems, thus contributing to the mitigation of their overall prevalence. Given that manual flagging is not a feasible solution in real-life contexts, an important goal of this research is to explore and leverage existing resources to automate the process of retrieving potentially obfuscated words (refer to Appendix A) from different sources of user-generated text. Building on this, after the retrieval stage, it would be valuable to estabilish categories based on obfuscation techniques to efficiently determine the techniques to treat the words, at a later stage. Finally, in the treatment stage, the objective is to reconstruct a final word that closely approximates, and if possible exactly matches, the term intended by the abuser. As a means of comparison, exploring a different automated approach, such as leveraging a Large Language Model (LLM), could offer useful insights into the capabilities of current state-of-the-art NLP systems in detecting and treating character-level obfuscated words.

To address the objectives outlined above and guide the research, the following research questions were formulated:

- RQ1: How can automated methods reliably identify words obfuscated through character-level perturbations in informal online text?
- RQ2: How can the different types of character-level obfuscations addressed in this study be systematically categorized?
- RQ3: What techniques can be used to detect and recover the original form of words obfuscated by character-level adversarial attacks?
- RQ4: How do different automated approaches compare in their ability to detect and treat character-level obfuscated words?

1.2. Document Structure

This thesis follows the phases of the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, and consists of six chapters. Chapters 1 and 2, representing the Business Understanding phase, provide the necessary background, outline the research motivation and objectives, and present a systematic literature review on adversarial NLP in the context of offensive language detection. Chapter 3, representing the Data Understanding and Data Preparation phases, introduces the data, including one

dataset collected manually and additional datasets obtained from online sources, detailing their preprocessing, and exploratory analyses that guide the subsequent modeling phase. Chapter 4, representing the Modeling phase, presents the modeling approaches used and discusses their contributions to obfuscation normalization. Finally, Chapters 5 and 6, representing the Evaluation phasem, evaluate the methods, compare results, draw conclusions, and answer the research questions along with a reflection on the limitations faced and future research. The Deployment phase

CHAPTER 2

Literature Review

The purpose of this systematic literature review, whose structure is influenced by other reviews and guidelines (Fernandes et al., 2023; Kitchenham & Charters, 2007), is to collect and analyze available research on abusive language detection models, primarily focusing on adversarial attacks that compromise their effectiveness and measures taken to mitigate them. First of all, the methodology behind the review will be covered, explaining how the literature was selected and the criteria considered for filtering purposes. After that, the results of the review will be presented followed by their respective discussion. Lastly, the limitations encountered will be exposed and given some suggestions for future research.

2.1. Methodology

This section covers the procedure followed to collect and analyze relevant literature, including the identification of key aspects, search method used in the source, inclusion and exclusion criteria to be applied to the search results and how the screening and categorization was conducted.

Key aspects

In order to organize this review and outline the reasoning behind it, three different aspects were studied:

- **A1** - Models used for abusive language detection and their corresponding performance.

  Addressing this will help gather the available information on the types of models (e.g., recurrent neural networks, BERT) used for abusive language detection. Knowing that and their respective performances, makes it possible to compare benchmarks and comprehend the advantages and disadvantages of each model.
- **A2** - Types of adversarial attacks used against deployed abusive language detection models
  - A2.1 - Prevalence of attacks.
    This will help understand which adversarial attack techniques are most common and the reason why they're so popular, which could be related to ease of implementation, effectiveness and wide applicability across different model types.
  - A2.2 - Impact of attacks.

Addressing this could allow for a selection of attacks to be studied, focusing on their effectiveness, and minimize wasted time and resources exploring attacks that are outdated or ineffective, ensuring the relevance of future studies that will benefit from this review.

  - A2.3 - Defense strategies against attacks.
    This could help understand not only the vulnerabilities of some techniques used to perform attacks, but also potential vulnerabilities on the defense responses, allowing the exposure of flaws in defense methods to improve robustness or enhance the effectiveness of attacks and raise awareness for the defenders.

- **A3** - Robustness of abusive language detection models against adversarial attacks.
  - A3.1 - Methods for improving model robustness
    This will help understand what methods are used and how they contribute to the robustness of the models. Through the analysis of each method, it could be possible to determine their strengths and limitations.
  - A3.2 - Effect of model type on robustness
    Addressing this will provide a better understanding of the effects that each model's architecture can have on their robustness against certain adversarial attacks. Some models may be more susceptible to a certain type of attack than others.

Sources and Queries

The scientific database used to perform the search was Scopus, an abstract and citation database from Elsevier composed of peer-reviewed literature (Ballew, 2009). In order to extract articles relevant to the study at hand, a query was carefully constructed (See Figure 2.1). On the first part of the query there is a clear division of keywords through the AND operator, one group is related to the main focus of the research (NLP, language model, transformer model, hate speech detection, abusive language detection, etc.), the second group captures complementary topics that pair well with the first (adversarial attack, text perturbation, leetspeak, etc.). Regarding the second part of the query, its purpose is to exclude research unrelated to the goals of the review through keywords (suicide, finance, computer vision, face recognition, etc.). It is important to note that some literature was found through references present in the articles extracted from Scopus.

Inclusion and Exclusion Criteria

To narrow the search, the following inclusion criteria were defined:

- Publications from 2015 to 2024;
- Focus on text-related tasks only, excluding those related to speech;
- Within the domain of hate speech or abusive language;
- Only conference papers and journal articles.

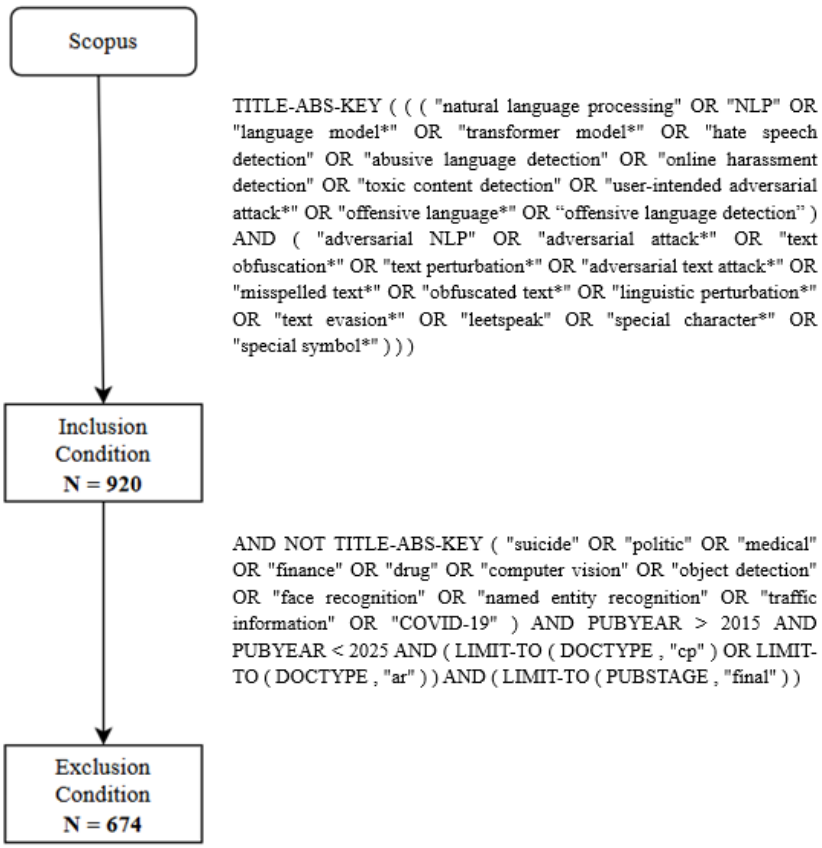


FIGURE 2.1. Data retrieval processs.

In terms of exclusion criteria:

- Research based on a language other than English or with a different writing system;
- Psychological Studies;
- Research utilizing models for multimodal or computer vision tasks.

Literature Screening and Categorization Criteria

Through literature screening, it will be possible to acquire a final sample of articles that are aligned with the review's purpose. Given the presence of unwanted research, which is unavoidable through querying without the possibility of losing valuable research, it was decided to manually screen the scientific papers obtained previously. The screening process involved exporting the query's results from Scopus to a RIS (Research Information Systems) file and uploading it to Zotero, a reference management software, to analyze the entries. During this process, not only the previously set inclusion and exclusion criteria were followed, but it was also taken into account if the research addressed textual adversarial attacks and not only classification of unperturbed offensive language, as this is the main focus of the review.

To categorize the literature, a description of primary studies, quantitative and non-quantitative summaries, and other statistical information will be presented.

2.2. Results and Discussion

This section focuses on presenting and discussing the results gathered from reviewing and characterizing the literature to address the aspects outlined previously.

The data collected to characterize the literature (see Table 2.1) was mostly extracted from Scopus, except for some articles whose bibliographic data had to be searched on the Internet. Through its analysis, it is possible to notice that the paper by (Gao et al., 2018) is the most cited with a total of 486 citations, presenting a novel algorithm, DeepWordBug, that generates small text perturbations in a black-box setting. Among the 28 articles, 24 are conference papers while the remaining 4 are journal articles.

In order to cover the aspects, an overview of the methodology and key findings of each article, from oldest to most recent, will be presented along with indication of the aspects they help cover (see Table 2.2).

Table 2.2: Literature summarization and mapping to aspects.

Literature	**Methodology**	**Key findings**	**A1**	**A2**	**A3**
(Gao et al., 2018)	Develop a black-box adversarial text attack generator (DeepWordBug) capable of evading deep learning classifiers through performing character-level transformations to the highest ranked words.	DeepWordBug performance on Enron spam emails and IMDB movie reviews reveals that it can decrease the performance of uni and bidirectional LSTMs by 40 to 60%.		✓	
(Li et al., 2018)	Develop a general attack framework, TEXT BUGGER, responsible for generating textual adversarial attacks to expose the vulnerability of Deep Learning-based Text Understanding.	TEXT BUGGER outperforms state-of-the-art attack models in terms of success rate. Preserves the legibility of adversarial attacks since 94.9% of the attacks are recognized by humans. Generates adversarial attacks efficiently, with optimized computational cost.		✓	
(Gil et al., 2019)	Perform knowledge distillation from a white-box model (HOTFLIP) to a neural model and whether this neural model can generalize to create black-box attacks.	The process of attacking through the black-box model developed is 39 times faster while maintaining a similar performance to the white-box approach.		✓	

Literature	Methodology	Key findings	A1	A2	A3
(Eger et al., 2019)	Develop a Visual Perturber (VIPER) that randomly replaces characters in the input with a visually similar alternative to fool neural and non-neural models. Explore potential defenses like visual character embeddings, adversarial training and rule-based recovery.	Models attacked by VIPER suffer decreases in performance of up to 82%. Injecting random noise generally improves robustness but not as much as visually informed adversarial training. Humans perform greatly against visual perturbations, which is intended and proves that they're suited to attack NLP systems.		✓	✓
(Zhou et al., 2019)	Develop a framework (DISP) to learn to identify, using a perturbation discriminator to assess the likelihood of a token being perturbed, and correct adversarial attacks through an embedding estimator that learns to restore the embedding of the original word.	Considering BERT as a baseline classifier, DIST reduces overall performance drops resulting from attacking BERT. DIST performs better at blocking character-level attacks than word-level attacks.		✓	✓
(Eger & Benz, 2020)	Propose a large-scale catalogue and benchmark of low-level adversarial attacks, including visual and phonetic examples.	Use of a visual model that is a combination of GAN (Generative Adversarial Network) and VAE (Variational Autoencoder) able to learn embeddings which encode high-level abstract features. Phonetic attacks were the least effective while intrude and visual attacks were the strongest.		✓	
(Morris et al., 2020)	Develop a framework made of existing attacks or new attacks. Perform data augmentation or adversarial training with NLP attacks.	Possibility of creating new attacks based on combinations of previous works and reuse attacks for data augmentation and improvement of the model's robustness to attacks.		✓	✓

Literature	Methodology	Key findings	A1	A2	A3
(Pruthi et al., 2019)	Placement of semi-character based RNN with Backoff for word recognition before the downstream classifier to combat character-level adversarial attacks.	ScRNN involving backing off to the background model yields the best results in terms of word error correction with a 6.9% error rate. Word-piece and character-level models are more susceptible to attacks. Better word recognition doesn't imply higher robustness.			✓
(Mou et al., 2020)	Develop a framework that relies on the text content of a message, identifies most important words and extracts surrounding information, exploiting word-level semantic and sub-word knowledge to classify hate speech.	The most important word contributes most in both sentiment and semantic meaning, therefore it should be focused. SWE2 /w BERT model outperformed all state-of-the-art baselines both under no attack and under 50% attack.	✓		
(Zeng et al., 2021)	Develop an open-source textual adversarial attack toolkit with diverse functionalities.	Provides all-attack-type support, multilinguality and parallel processing. Offers flexibility, extensibility and easy customization of the model that will be attacked for robustness tests.		✓	
(Keller et al., 2021)	Detect the tokens in a perturbed sentence, independently of context. Masked language modeling in BERT. Decision-making using a language model to choose the hypothetically unperturbed version of the sentence.	Model doesn't train on possible error types, outperforms spell-checkers and ScRNN model, considered to be enough against character-level attacks. Rivals human crowd-workers supervised in a 3-shot manner.		✓	
(Hofer et al., 2021)	Perform adversarial attacks like a human would and using leetspeak, common misspellings or misplaced commas with a focus on important words.	Leetspeak has the highest success rate among the attack methods.		✓	

Literature	Methodology	Key findings	A1	A2	A3
(Chen et al., 2021)	Develop a multi-granularity attack model (MAYA) to perform decision-based black-box attacks. Train an attack agent using reinforcement learning through behavior cloning of the MAYA algorithm.	MAYA outperforms other baselines, such as BERT-Attack in attacking (except on SST-2 dataset) and produces adversarial samples that are fluent and grammatically superior.		✓	
(Mary et al., 2022)	Perform adversarial attacks through usage of leetspeak and phonological errors (character "x" substituted by the characters "ks").	Regarding leetspeak attacks, the classifiers perform worse as the perturbation percentage increases. On the other hand, for phonological attacks further increases in perturbation percentage past 20% yields no impact.		✓	
(Rusert et al., 2022)	Perform word selection for replacement following a greedy approach and attention weights. Design a context-aware word embedding for crafty replacements. Implementation of a pre-existing visual adversarial attack, VIPER, and test of all the previous approaches against offensive language classifiers.	Exposed the offensive language classifiers' vulnerability to adversarial attacks. Author's approach of greedy selection outperforms attention weights while FT embeddings and Pre embeddings' success depends on the system attacked. NULI is the most robust BERT-based classifier while vradivchev is the most vulnerable.	✓	✓	✓
(Omar & Mohaisen, 2022)	Adoption of three state-of-the-art pre-trained hate speech detection models, BERT, RoBERTa and CNERG. Test how models perform using Tweet Eval benchmark and expose them to new scenarios using attack frameworks.	Performing adversarial attacks on models not trained with adversarial examples results in a significant drop in accuracy. Upon retraining the models with adversarial examples, accuracy consistently increases. BERT outperforms all other models.	✓		✓

Literature	Methodology	Key findings	A1	A2	A3
(Seth et al., 2023)	Compilation of character-level attacks obtained through the replacement of a subset of characters with a visually similar counterpart that is legible, influenced by human preference. Perform tests for legibility and rank the attacks accordingly. Comparisons are made between two perturbed inputs as well as a perturbed input and its original form.	TrOCR-MT model outperforms all other models in both classification and ranking tasks. Regarding the Jigsaw Challenge, LEGIT produces better results than VIPER-ECES when it comes to impacting model performance while also preserving the legibility of text.		✓	✓
(Shen et al., 2023)	Develop a saliency-based detector to detect if an input sentence is adversarial and a saliency-based corrector to convert the adversarial input to its original form.	TextShield outperforms the best baseline such as Synonym Encoding Method (SEM), in most cases. Additionally, adding adversarial samples generated by BERT-attack does not change the ground-truth.			✓
(Huertas-García et al., 2023)	Produce visually similar inputs through applying leet-speak, inserting punctuation symbols and word inversion. Develop a multilingual NER model for word camouflage detection.	MPNET-ideal, a version of MPNET-base with pre-training in mSTSb, outperforms the remaining models in most datasets, making it the best overall against both multilingual and monolingual models.	✓		
(X. Zhang et al., 2023)	Develop a Masked Language Model-based Detection (MLMD) to detect textual adversarial attacks through exploring changes in manifolds induced by the masked language model.	MLMD consistently shows a similar or better performance compared to FGWS and WDR methods when detecting adversarial attacks. By performing mask and unmask procedures using masked language models, normal and adversarial examples become easily distinguishable. Time and computation complexity are known limitations.		✓	✓

Literature	Methodology	Key findings	A1	A2	A3
(Hu et al., 2023)	Insertion of mask tokens to input sequences during training and inference maximizing how close the convex hulls of the original input and the altered one remain to each other.	Mask insertion for only tokens labeled as tokens to be attacked (Mask_Pol), tokens that have high impact when altered, is the best defense while mask for only normal characters is the worst.			✓
(Ge et al., 2023)	Usage of perplexity to quantify how different clean and adversarial examples are. Develop a defense framework, that leverages masked language models, to recover out-of-vocabulary words and then apply Perplexity-guided Sampling to locate the optimal sample within a convex hull constructed with integrated gradients.	UMPS (Uncover Mask and Perplexity-guided Sampling) generally outperforms the baselines, such as BERTDefense, by granting a higher increase in model accuracy. Proving that perplexity differs between adversarial attacks and their original counterpart leads to new research possibilities.			✓
(Formento et al., 2023)	Perform adversarial attacks through punctuation insertions.	NLP models are more susceptible to punctuation insertions than other character manipulations. DeepWordBug-Punc without random punctuation and random position yields the highest attack success rate.	✓	✓	
(Colombo et al., 2023)	Develop a framework to detect textual adversarial attacks and a benchmark composed of popular attacks, datasets and pre-trained models.	Detection methods perform attacks better on ROB than BERT. LAROUSSE achieves the best results in threshold-free and threshold-based metrics, in comparison to GPT2 and the alternative LAROUSSE using Mahalanobis distance.			✓

Literature	Methodology	Key findings	A1	A2	A3
(Chen et al., 2023)	Establish robustness evaluation dimensions based on model capabilities and assign a suitable adversarial attack. Outline realistic evaluation settings and metrics. Assess sample validity through the perturbation degree of an adversarial sample.	Model robustness scores depend on the robustness dimension and drop as the perturbation degrees increase across different models. Considering average performance estimation, score-based attacks are more effective than rule-based attacks, while in worst performance cases it is the opposite.	✓		✓
(Bespalov et al., 2023)	Perform synonym substitution through predictions made by BERT Masked Language Model or nearest neighbours and character transformations. Goal functions consider multiclass, binary and multilabel toxicity predictors.	Using GloVe for word synonym substitution is better than MLM or WordNet. Character manipulation can generate examples that retain their meaning while fooling the classifier. To combat this, it is proven that adversarial training highly improves models' robustness.		✓	✓
(Jebran Khan, 2023)	Generate textual adversarial attacks using out-of-vocabulary words based on visual and phonetic similarities of characters and use a word saliency-based beam search algorithm to search for the best perturbations.	The proposed attack success rate is slightly higher and more realistic than TextFooler, a word-level attack, and both yield the highest degradation on the classifiers' accuracy.		✓	
(Aggarwal et al., 2024)	Develop a framework "Homograph" to generate character-level adversarial attacks, making use of the Corpus of Linguistic Acceptability (CoLA), in a black-box setting that are visually similar and don't compromise the grammatical acceptability of the original input. The optimal perturbations are found through Particle Swarm Optimization.	Homograph has a high attack success rate outperforming ALBERT, BERT, DistilBERT, RoBERTa and XLNET, fine-tuned on the CoLA dataset. Preserves semantic similarity and maintains grammatical acceptability.		✓	

TABLE 2.1. Literature characterization by Year, Type and Citations (sorted from oldest to newest).

References	**Year**	**Type**	**Citations**
(Gao et al., 2018)	2018	Conference Paper	486
(Li et al., 2018)	2019	Conference Paper	296
(Gil et al., 2019)	2019	Conference Paper	28
(Eger et al., 2019)	2019	Conference Paper	94
(Zhou et al., 2019)	2019	Conference Paper	67
(Eger & Benz, 2020)	2020	Conference Paper	41
(Morris et al., 2020)	2020	Conference Paper	322
(Pruthi et al., 2019)	2020	Conference Paper	149
(Mou et al., 2020)	2020	Conference Paper	17
(Zeng et al., 2021)	2021	Conference Paper	58
(Keller et al., 2021)	2021	Conference Paper	17
(Hofer et al., 2021)	2021	Conference Paper	5
(Chen et al., 2021)	2021	Conference Paper	21
(Mary et al., 2022)	2022	Conference Paper	2
(Rusert et al., 2022)	2022	Conference Paper	6
(Omar & Mohaisen, 2022)	2022	Conference Paper	6
(Seth et al., 2023)	2023	Conference Paper	2
(Shen et al., 2023)	2023	Conference Paper	1
(Huertas-García et al., 2023)	2023	Journal Article	6
(X. Zhang et al., 2023)	2023	Conference Paper	2
(Hu et al., 2023)	2023	Conference Paper	0
(Ge et al., 2023)	2023	Journal Article	0
(Formento et al., 2023)	2023	Conference Paper	18
(Colombo et al., 2023)	2023	Conference Paper	1
(Chen et al., 2023)	2023	Conference Paper	3
(Bespalov et al., 2023)	2023	Conference Paper	2
(Jebran Khan, 2023)	2023	Journal Article	0
(Aggarwal et al., 2024)	2024	Journal Article	0

Analyzing the distribution of the articles between the key aspects reveals that 6 articles address A1, 19 articles address A2 and 14 articles address A3, given that some articles provide insights for more than one aspect. While the quantity of articles that address A1 seems low, it is possible to extract information about abusive language detection models

through the remaining articles, since whether the goal is to perform adversarial attacks or invest in model robustness, it will always end up being tested on different classifiers to evaluate performance.

On the topic of abusive language detection models, in light of **A1**, there is a clear preference for the usage of BERT models to perform this task (Formento et al., 2023; Mou et al., 2020; Omar & Mohaisen, 2022; Rusert et al., 2022). However, there are other transformer-based architectures that are favored, such as a multilingual named-entity recognition (NER) transformer-based model (Huertas-García et al., 2023). Given that this type of architecture achieves state-of-the-art performance in abusive language detection, its adoption is justified despite the possibility to use other architectures such as Recurrent Neural Networks (RNNs) (Rusert et al., 2022).

In terms of adversarial attacks, to cover **A2**, the most common attacks are word-level adversarial attacks, involving word substitution and removal (Bespalov et al., 2023; Chen et al., 2021; Eger et al., 2019; Li et al., 2018; X. Zhang et al., 2023; Zhou et al., 2019). Some frameworks adopt the combination of word-level and character-level attacks (Morris et al., 2020; Zeng et al., 2021) but that does not change the previous claim. It is important to understand that character-level attacks do not fall too behind (Aggarwal et al., 2024; Eger & Benz, 2020; Eger et al., 2019; Formento et al., 2023; Gao et al., 2018; Gil et al., 2019; Hofer et al., 2021; Jebran Khan, 2023; Li et al., 2018; Mary et al., 2022; Seth et al., 2023) and may achieve equal or better performance depending on the scenario, despite the common misconception that character-level attacks are easily tackled by spellcheckers (Keller et al., 2021). Adversarial training is the most recommended defense against adversarial attacks.

2.3. Summary

A systematic literature review was conducted to collect and analyze critically available research in the domain of abusive language detection models. During the collection process it was noticeable that not many articles address adversarial attacks in the same manner as they occur in, for example, social media and gaming communities, in this case as user-intended attacks. It is also worth noting that there was not a standard keyword usage among researchers of this topic, as it often differed greatly from one article to another, leading to the need to broaden the scope of search and thus resulting in the collection of irrelevant research through querying. In terms of research screening and respective analysis, it is not possible to guarantee total absence of bias during the screening process, since it would make the extraction of insights that are aligned with the objective of this research a lot more difficult, due to lack of meaningful literature.

For future research, it is recommended to pay close attention to research on character-level adversarial attacks, as they are the most common attack performed by users in digital spaces, when attempting to evade automated content moderation. Lastly, given the constantly changing nature of user interactions in digital spaces, whether it be new

trends or slang, it is crucial to frequently update abusive language detection models to preserve their effectiveness when facing real-world data in real time.

CHAPTER 3

Methodology and Data

To conduct this research, it was essential to employ a well-estabilished and appropriate framework to guide the data mining process properly. With this in mind, the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology was adopted, as it offers a structured and widely recognized approach to carrying out data-driven projects independent of both the industry sector and the technology used. This process is structured into six main phases, outlining the life cycle of a data mining project: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Although these phases are usually presented sequentially, the actual progression through them is often non-linear and iterative, depending on the outcomes of previous stages (Wirth & Hipp, 2000).

The first phase, Business Understanding, aims to develop an understanding of the project's objectives and requirements from a business perspective. Leveraging the knowledge acquired, a data mining problem is defined, followed by the initial project plan tailored to achieve the objectives (Wirth & Hipp, 2000). In the present work, this phase is covered by Chapters 1 and 2, which provide the necessary background, outline the motivation and goals for the research, and assess the state of the art in the field of adversarial NLP attacks on the context of abusive language detection models.

With clear objectives established, the next step is to collect and explore the data in the Data Understanding phase, to get familiar with the data at hand, extract insights from it, identify potential data quality problems or to find interesting subsets that might lead to the formulation of hypotheses about underlying patterns in the data (Wirth & Hipp, 2000). In the context of this thesis, this phase is addressed in Chapter 3, specifically in sections 3.1 and 3.2, where the selected datasets are introduced, assessed for suitability, and described in detail, providing the foundation for subsequent preparation and modeling steps.

After developing a solid understanding of the data, the Data Preparation phase began with the goal of building the final dataset for modeling. This stage involves a variety of iterative tasks, including the selection of relevant tables, records, and features, the treatment of missing or inconsistent data, the creation of new variables, and the transformation of the dataset into a format suitable for the modeling techniques to be applied. These tasks are often repeated and do not necessarily follow a fixed sequence (Wirth & Hipp, 2000). This phase is covered in Chapter 3, in section 3.3, where the collected material was organized and processed into a final dataset. This included applying Optical Character Recognition (OCR) to extract text from screenshots, annotating obfuscated words

and techniques, implementing preprocessing steps such as tokenization and data cleaning, perform NLP tasks like Part-of-speech (POS) tagging, and isolating out-of-vocabulary tokens using external lexical resources. After iterative filtering and refinement, the final dataset of 501 instances was created, containing structured information on obfuscations to be used in the following phases.

To wrap up Chapter 3, while initial exploration of the data took place during Data Understanding, in this stage the focus shifted toward a more detailed analysis of the prepared dataset, with the aim of identifying patterns and properties relevant to the research objectives. Section 3.4, presents this phase, where the data was systematically explored to understand the nature and distribution of obfuscations, as well as general trends across the datasets. These analyses helped validate the dataset and provided insights that guided the subsequent modeling phase.

Once Data Preparation is complete, the process moves into the Modeling phase, which involves selecting appropriate modeling techniques and applying them to the prepared data. During this stage, model parameters are tuned to achieve optimal performance, and different approaches may be evaluated depending on the nature of the data mining problem. Due to some techniques requiring specific data formats or structures, this phase is often closely linked to Data Preparation. It is common for issues in the data to become apparent during modeling or for new ideas regarding data transformation to emerge, reinforcing the iterative nature of the process (Wirth & Hipp, 2000). Covered by Chapter 4, from sections 4.1 to 4.6, all the explored approaches are introduced and their contribution to the work is explained.

Finally, assuming models have been built and show promising results from a technical standpoint, the Evaluation phase focuses on assessing their effectiveness in meeting the original business objectives. This step goes beyond just analyzing performance metrics, it involves a critical review of the modeling process to ensure that no important aspect of the problem has been overlooked. By the end of this phase, a decision should be made regarding whether the results are ready to move forward into deployment (Wirth & Hipp, 2000). Spanning Chapter 5 and 6, the final results are documented and compared to draw conclusions regarding the differences in performance of the selected approaches.

In this thesis, there is no Deployment phase.

3.1. Data Sources and Collection

To study how obfuscated words are used to bypass abusive language detection models, the first step is to collect examples of this phenomenon. For that purpose, publicly available datasets were assessed based on their potential to contain such occurrences. Some of these datasets are included in a catalogue of resources annotated for hate speech, online abuse and offensive language which was created and is currently maintained by Vidgen and Derczynski (2021) in the context of their work. To assess the data's potential, a lexicon of profanity by SurgeAI (2021a) was used. This lexicon includes profane words and insults along with minimally obfuscated variants, such as misspellings that retain the

TABLE 3.1. Results of the assessment and filtering process to retain suitable datasets.

Dataset/Paper	**Source**	**Suitable**
A Benchmark Dataset for Learning to Intervene in Online Hate Speech - Gab & Reddit	https://github.com/jing-qian/A-Benchmark-Dataset-for-Learning-to-Intervene-in-Online-Hate-Speech	✓
Automated Hate Speech Detection and the Problem of Offensive Language	https://github.com/t-davidson/hate-speech-and-offensive-language	✓
Ethos Hate Speech Dataset	https://github.com/intelligence-csd-auth-gr/Ethos-Hate-Speech-Dataset	×
Ex Machina Personal Attacks Seen at Scale, Personal attacks	https://github.com/ewulczyn/wiki-detox	×
Harassment Corpus	https://github.com/Mrezvan94/Harassment-Corpus	×
Hatecheck Data	https://github.com/paul-rottger/hatecheck-data	×
HateXplain Data	https://github.com/hate-alert/HateXplain	×
Introducing CAD the Contextual Abuse Dataset	https://zenodo.org/records/4881008	×
Measuring Hate Speech	https://huggingface.co/datasets/ucberkeley-dlab/measuring-hate-speech	×
OLIDv1.0	https://sites.google.com/site/offensevalsharedtask/olid	×
OffensEval 2020	https://sites.google.com/site/offensevalsharedtask/offenseval-2020	×
SEMEVAL 2021	https://github.com/ipavlopoulos/toxic_spans/tree/master/data	×
Towards a Comprehensive Taxonomy and Large-Scale Annotated Corpus for Online Slur Usage	https://github.com/networkdynamics/slur-corpus	×
UBWEB Detecting cyberbullying in online communities	https://ub-web.de/research/	×
SurgeAI Toxicity EN	https://github.com/surge-ai/toxicity	✓

original or a similar pronunciation of a word while altering the spelling, as well as minor substitutions of letters with visually similar symbols or numbers.

In addition, more examples were collected, from user interactions on social media and gaming chats, where abusive language was intentionally obfuscated and successfully avoided automatic moderation. The following table, Table 3.1, summarizes the results of the assessment and filtering process to retain datasets suitable for the task (manually collected user interactions are not present as they need to be organized first as a dataset).

3.2. Data Description

In this section, the selected datasets are outlined and described. The first dataset by Jing (2020), which consists of two fully labeled subsets collected from Gab and Reddit with 11,825 and 5,020 posts, respectively. The dataset is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0). It preserves conversational context and includes human-written intervention responses, with posts manually labeled as hate or non-hate to facilitate their use in hate speech detection tasks. The structure of the dataset is as described in Table 3.2.

TABLE 3.2. Structure of the data from *A Benchmark Dataset for Learning to Intervene in Online Hate Speech by Jing (2020)*

Field	**Description**
`id`	The IDs of the posts in a conversation segment.
`text`	The text of the posts in a conversation segment.
`hate_speech_idx`	A list of the indexes of the hateful posts in this conversation.
`response`	A list of human-written responses.

The second dataset (see Table 3.3) used was created in the context of the paper by Davidson (2017). It consists of 24,783 tweets annotated through crowd-sourcing as hate speech, offensive language, or neither. The tweets were initially collected using a lexicon of hate speech keywords, and the annotations aim to distinguish between different forms of offensive content on social media.

TABLE 3.3. Structure of the data from *Automated Hate Speech Detection and the Problem of Offensive Language by Davidson (2017).*

Field	**Description**
`count`	Number of annotations received for the tweet.
`hate_speech`	Number of annotators labeling the tweet as hate speech.
`offensive_language`	Number of annotators labeling the tweet as offensive language.
`neither`	Number of annotators labeling the tweet as neither.
`class`	Final label assigned to the tweet (1 = offensive language, 2 = hate speech, 0 = neither).
`tweet`	The text content of the tweet.

The third dataset used was provided by SurgeAI (2021b). It contains 1,000 human-annotated social media comments, equally balanced between toxic and non-toxic examples. Each comment is annotated by reviewers who judged whether they personally found the content toxic. The dataset includes the comment text and a binary label indicating toxicity. Its structure is detailed in Table 3.4.

In addition to the annotated datasets, this study also utilized the lexicon released by SurgeAI (2021a). The lexicon contains in total 1,598 profanities and some variations. The categories cover a wide range of offensive language, including references to sexual

TABLE 3.4. Structure of the *Toxicity Dataset* by SurgeAI (2021b)

Field	Description
`text`	The text content of the comment.
`is_toxic`	A binary label indicating whether the comment is toxic (1) or non-toxic (0).

anatomy, bodily functions, gender and sexual orientation, race and ethnicity, disabilities, physical attributes, animal references, religion, and politics. Severity is determined on a three-point scale (mild, strong, severe) based on the aggregated judgments of five human annotators. This lexicon was used as a linguistic resource to support the identification and analysis of profane and abusive expressions in online text. The structure is provided in Table 3.5.

TABLE 3.5. Structure of the *English Profanity Lexicon* by SurgeAI (2021a)

Field	Description
`text`	The profanity word or phrase.
`canonical_form_1`	The canonical form of the profanity.
`canonical_form_2`	An additional canonical form, if applicable.
`canonical_form_3`	An additional canonical form, if applicable.
`category_1`	The primary semantic category.
`category_2`	A secondary category, if applicable.
`category_3`	A tertiary category, if applicable.
`severity_rating`	Mean severity score (1-3) from five human annotators.
`severity_description`	Severity level derived from rating (1 = mild, 2 = strong, 3 = severe).

3.3. Data Preparation

This section covers the steps taken to create the final dataset. It describes the extraction and treatment of the manually collected data, the processing approaches applied and the method of isolating obfuscated offensive words, followed by further refinements after a brief exploratory analysis and annotation, to ensure its suitability for a comprehensive analysis and modeling.

3.3.1. Manually collected data

As previously outlined, a portion of the data was manually collected from user interactions on online platforms, both before and during the development of this thesis, and this dataset will be addressed as "Collected_Obfuscations". Given the unstructured nature of the data, an initial stage of organization was required to transform it into a suitable format for further preprocessing and analysis. Since the initial raw material consisted of screenshots containing a considerable amount of text, OCR was applied using PaddleOCR by Cui et al. (2025) to automatically extract textual content. This approach significantly reduced manual transcription effort while ensuring scalability. Post-extraction, all outputs

were systematically validated against the original screenshots, recognition errors were corrected manually. This verification step was feasible due to the limited number of entries and ensured the integrity of the dataset for subsequent preprocessing. Additionally, three columns, "Obfuscated", "Obfuscated_Info", and "Technique" were manually annotated to provide details regarding the obfuscations and the techniques used. As a result of minimal feature engineering and other necessary preparations, the final count of occurrences amounted to 305. The data structure is presented in Table 3.6.

TABLE 3.6. Structure of the Collected_Obfuscations dataset

Field	**Description**
`Image`	Filename of the screenshot from which the text was extracted.
`Date`	Timestamp associated with the interaction, indicating when the message was recorded.
`Text`	Extracted textual content of the message using OCR.
`Regions`	Coordinates of the bounding box in the screenshot that delimit the text region identified by OCR.
`Confidence`	Confidence score output by the OCR system, indicating reliability of the text recognition.
`Obfuscated`	The obfuscated word or token as it appears in the text (e.g., "k1ll").
`Obfuscated_Info`	The corresponding normalized or canonical form of the obfuscated word (e.g., "kill").
`Technique`	The obfuscation technique applied, such as leetspeak, letter removal, word splitting, or transposition.

3.3.2. Text preprocessing and NLP tasks

This subsection describes, by using Python programing language and its modules/libraries, the text preprocessing and natural language processing (NLP) tasks applied to the datasets prior to analysis. After loading the datasets, several preprocessing steps were done before analysis.

The initial preprocessing pipeline consisted of the following steps:

(1) **Tokenization and POS tagging** – The datasets were initially tokenized to enable POS tagging using `nltk` library. Once POS tags were assigned, tokenization was refined using the syntactic information provided by the tags. This two-step process ensured more precise segmentation and improved the reliability of downstream analysis.

(2) **Contraction handling** – Contractions were expanded and normalized using regular expressions from the `regex` library. Simple contractions (e.g., *don't* → *do not*) were directly expanded, while context-dependent ones (e.g., *he'd*) were disambiguated using POS context (*he would* vs. *he had*).

(3) **Punctuation and symbol removal** – Tokens consisting solely of punctuation or symbols were systematically removed using regular expressions to reduce noise.

(4) **Username tagging** – In social media datasets, the token immediately following an "@" was tagged as a proper noun (NNP) to ensure usernames were consistently treated as named entities.

3.3.3. Isolating obfuscated tokens

The isolation of obfuscated tokens from the overall token set represents an ideal scenario for studying their characteristics and developing strategies to prevent them from being overlooked by automated moderation systems. To this end, the objective was to design a filtering method capable of reducing the presence of standard vocabulary items while simultaneously maximizing the retention of out-of-vocabulary (OOV) tokens, the category where obfuscated words belong. Google Books Ngram Corpus Version 2 generated in July 2012 (Lin et al., 2012) was employed as a large-scale lexical resource to filter out common vocabulary items. By leveraging its word frequency information across centuries of published books, it was possible to distinguish standard and widely attested words from OOV tokens more effectively. This approach ensured that the token set under consideration was enriched with less common, potentially obfuscated words, thereby providing a more reliable basis for subsequent analysis of obfuscation strategies in user-generated content.

The unigram subsets vary substantially in size, ranging from as few as 3.9 million tokens ("x" token) to more than 110 million tokens ("s" token). It is worth noting that the largest subsets correspond to more frequent initial letters such as "s", "c", and "a", while the smallest subsets correspond to less common letters such as "x", "y", and "z". The size of each subset is detailed in Table 3.7.

Furthermore, all unigram files were loaded into memory to construct a comprehensive unigram reference. Each file was read line by line, and its contents were parsed into a dataset containing the word form, year of occurrence, and frequency counts. To focus the analysis on more recent language use, entries dated before the year 2000 were excluded. The remaining records were aggregated by word, summing their total frequencies across years. Finally, the resulting subsets were concatenated into a single dataset, providing a unified frequency distribution of unigrams to be used as a filtering resource in later stages of preprocessing. The characteristics of the aggregated unigram dataset are summarized in Table 3.8. However, since the words had their POS tag appended to them, these were separated and new total frequencies were recalculated accordingly. In addition, for visualization purposes, a categorical feature "freq_category" was created to group words by frequency ranges. These frequencies can be seen in Table 3.9. Only words with a total frequency of 9,999 or higher were retained in the dataset to focus on more commonly attested vocabulary.

Once the unigram dataset was ready, an iterative process of preserving as many OOV words as possible started. The final preprocessing approaches that contributed towards our current objective were:

TABLE 3.7. Total occurences per unigram subset in the Google Books Ngram Corpus (Version 2).

Subset	Number of Tokens
a	86,618,506
b	61,551,918
c	97,689,326
d	59,278,474
e	51,648,533
f	47,058,760
g	39,861,296
h	45,597,098
i	52,555,878
j	17,008,270
k	27,279,768
l	47,674,901
m	71,628,465
n	35,668,125
o	35,924,019
p	88,562,874
q	6,733,746
r	54,390,272
s	110,075,425
t	67,720,440
u	22,161,314
v	27,534,069
w	29,936,821
x	3,929,236
y	6,869,308
z	6,828,143

TABLE 3.8. Structure of the aggregated Google Books Ngram 1-gram dataset.

Field	Description
`word`	The observed word form.
`total_freq`	The total number of times the word occurred in that year across the corpus.

TABLE 3.9. Distribution of words by frequency categories in the aggregated unigram dataset.

Frequency Category	Number of Words
0–1,000	6,353,015
1000–5,000	479,205
5000–10,000	94,957
10000–25,000	75,669
25000–50,000	34,676
50000–100,000	23,786
100,000+	51,889

(1) **In-text username tagging and removal of standalone "@" symbols** – As a follow-up to an initial preprocessing step, tokens immediately following an "@" symbol were tagged with the suffix `_USERNAME` to consistently identify user mentions. The standalone "@" symbol was then removed from the token sequence to avoid unnecessary noise.

(2) **Hyperlink removal** – Tokens corresponding to URLs were identified by the presence of the protocols `http` or `https`, along with the subsequent domain/path token, and removed entirely from the dataset to eliminate external references.

(3) **Emoji removal** – Tokens recognized as emojis were filtered out using the `emoji` library, preventing non-linguistic symbols from interfering with textual analysis.

(4) **Punctuation cleaning** – In addition to the previous punctuation treatment, remaining leading and trailing punctuation marks in each token were removed via regular expressions.

(5) **Numeric token removal** – Tokens composed entirely of digits or numeric expressions (including decimal and fractional forms) were removed using pattern matching.

To identify out-of-vocabulary (OOV) words, two complementary approaches were implemented. As previously described, the first leveraged the Google 1-gram corpus as a statistical reference of language usage, marking as OOV those words absent from the unigram set. The second relied on WordNet, flagging words not associated with any synsets as OOV. Both methods excluded common stopwords. Before applying these methods to the data, each dataset was split into train and test subsets. For each dataset, a random sample corresponding to 80% of the total instances was assigned to the training partition, while the remaining 20% was reserved for testing. A preliminary analysis showed that the 1-grams resource captured fewer non-OOV words than WordNet, thereby aligning more closely with the objective of isolating true OOV tokens.

3.3.4. Exploratory Analysis and Iterative Refinement

With the datasets prepared, they were ready for exploratory analysis and further refinement. Functions were developed to check for the presence of numbers, symbols, or homoglyphs in the OOV words for each instance. Originally, the functions count each individual character (e.g., numbers, symbols, or homoglyphs) within the tokens of the OOV list. However, for simplification purposes, if at least one token in a post contained a number, symbol, or homoglyph, it was counted as a single occurrence for that category. A homoglyph map, based on a list from Codebox (2020), with some modifications, was used to detect homoglyphs. The map was created by associating different Unicode characters with their standard counterparts based on visual resemblance. However, the map's coverage is not comprehensive, and some potential variations may be missing. The frequencies of each category across the data are detailed in Table 3.10.

After computing the frequencies for each category, the data was manually reviewed and annotated in Microsoft Excel. The posts flagged for containing numbers, symbols, or

TABLE 3.10. Frequency of numbers, symbols, and homoglyphs among OOV tokens in different datasets, including total count.

Name	Dataset	Numbers	Symbols	Homoglyphs
A Benchmark Dataset for	Train	1,512	2,931	38
Learning to Intervene in	Test	378	738	9
Online Hate Speech - Gab	Total	1,890	3,669	47
A Benchmark Dataset for	Train	823	1,644	8
Learning to Intervene in	Test	212	439	3
Online Hate Speech - Reddit	Total	1,035	2,083	11
Automated Hate Speech	Train	447	612	0
Detection and the Problem	Test	101	144	0
of Offensive Language - Twitter	Total	548	756	0

homoglyphs were checked, and only those in which the OOV words clearly represented obfuscated speech were kept, while false positives were removed. At a later point in this stage, the *SurgeAI Toxicity EN* dataset was reviewed in full, along with additional manually collected obfuscations (*Collected_ Obfuscations*). It is worth noting that the criteria used to filter OOV words were based on whether a word represented an obfuscated variation, as opposed to words denoting, for example, bands, brand names, or other cases ("7-Eleven", "Ty Dolla $ign").

Contrary to expectations, the Reddit and Twitter datasets, despite being social media platforms prone to heated discussions, contained few instances of obfuscated speech. In contrast, the manually collected dataset contains the largest number of instances after including new additions, which was expected, followed by the Gab dataset. The distribution of instances across sources is shown in Table 3.11.

TABLE 3.11. Distribution of obfuscated instances across all datasets.

Dataset	Instances
A Benchmark Dataset for Learning to Intervene in Online Hate Speech - Gab	112
A Benchmark Dataset for Learning to Intervene in Online Hate Speech - Reddit	22
Automated Hate Speech Detection and the Problem of Offensive Language - Twitter	6
Collected_Obfuscations	332
SurgeAI Toxicity EN	29

From this review and annotation process, the structure of the final dataset, containing 501 instances, was defined. Each entry contains both the original text and its processed forms, along with categorical indicators marking the presence of different types of obfuscation. Table 3.12 summarizes the fields included in the dataset.

3.3.5. Post-Annotation Processing

Following the definition and annotation of the final dataset described in the previous subsection, an additional round of preprocessing was carried out to refine and consolidate

TABLE 3.12. Structure of the final dataset.

Field	Description
`Original`	The original text string.
`Original_Tokenized`	The original text string tokenized.
`Flagged`	Words of the original text that are obfuscated.
`Clean`	A cleaned version of the original text without obfuscations.
`Clean_Flagged`	A cleaned version of the flagged words without obfuscations.
`Homoglyphs`	Binary indicator (1 if homoglyph substitutions are present, 0 otherwise).
`Symbols`	Binary indicator (1 if symbol substitutions are present, 0 otherwise).
`Numbers`	Binary indicator (1 if numeric substitutions are present, 0 otherwise).
`Phonetic`	Binary indicator (1 if phonetic variations are present, 0 otherwise).
`Other`	Binary indicator (1 if other obfuscation types are present, 0 otherwise).
`OOV`	Words considered Out-of-Vocabulary present in the original text.
`Dataset_Label`	Dataset where the original text is from.
`Example_Label`	Example label provided for reference.

the data. This stage aimed to address residual noise and inconsistencies that were not fully resolved in the initial processing steps, ensuring that the dataset was ready for subsequent analysis. The operations included:

- **Remove irrelevant parts of a post** – In the Reddit and Gab datasets, entries often included both the original post and replies. With this in mind, segments that did not contain the obfuscated speech that was flagged during the annotation process were removed to ensure the dataset remained focused on relevant content.
- **Refined tokenization and contraction handling** – Building upon the initial tokenization and POS-tagging pipeline, a more robust procedure was implemented to account for insights gained during the manual review and annotation process. This updated approach includes:
  - Normalization of quotes and apostrophes to standard forms to reduce variability across tokens.
  - Separation of leading and trailing quotation marks to ensure accurate token boundaries while preserving meaningful repeated punctuation of obfuscated words (e.g., "ki!!").
  - Protection of obfuscated tokens and URLs during tokenization to prevent unintended splits.
  - Context-aware handling of contractions using POS information to disambiguate forms such as *he'd* (*he would* vs. *he had*).

– Final token cleaning to strip unnecessary punctuation while preserving meaningful patterns relevant for obfuscation detection.

This refinement ensures that tokenization is better aligned with the characteristics of the data and improves the quality of subsequent analyses.

- **Remove tokens representing numbers** – Tokens corresponding to numerical expressions (e.g., percentages, ordinals, abbreviations such as "170k", time formats, fractions, or numbers with appended units) were identified using regular expressions and removed.
- **Strip control characters** – Non-printable control characters were removed from all tokens to prevent noise caused by artifacts such as formatting symbols or hidden characters.
- **Remove URLs** – Tokens matching common URL patterns (e.g., "http://", "https://", or "www.") were excluded from the dataset to attempt to eliminate links unrelated to obfuscated content that were still present after the first removal attempt.
- **Remove emojis** – Emoji characters were filtered out again since new entries were added to the data after the preliminary preprocessing.
- **Remove standalone punctuation and symbols** – Tokens consisting solely of punctuation marks or special symbols (e.g., “.”, “, , ?, $, %, etc.) were removed to reduce noise in the dataset.
- **Remove social mentions** – Tokens representing social media mentions, including usernames and subreddit references (e.g., tokens starting with "@", "!" on Gab, or "u/" and "r/" on Reddit), were excluded to avoid being mistaken as OOV words.

Although OOV words were identified in an earlier preprocessing stage, this step was repeated after the refined cleaning and tokenization to ensure the integrity and consistency of the dataset. For a high-level perspective of the Data Preparation phase refer to Appendix B.

3.4. Data Analysis

Having finalized the dataset, the next step was to conduct a systematic analysis of its contents. This analysis began by examining the distribution of obfuscation types across datasets, providing an overview of their prevalence. From the 501 entries of posts containing obfuscated words, 761 words were flagged during the annotation phase. Since individual entries can exhibit multiple types of obfuscation simultaneously, the distribution of obfuscation types is presented in Figure 3.1.

As shown, the majority of obfuscations are based on phonetic similarities, while substitutions involving numbers are the least frequent. Despite this, the overall number of examples across the different obfuscation types remains relatively balanced.

On the other hand, analyzing the distribution of obfuscation types across the different data sources (see Figure 3.2) reveals that the manually collected dataset is comparatively

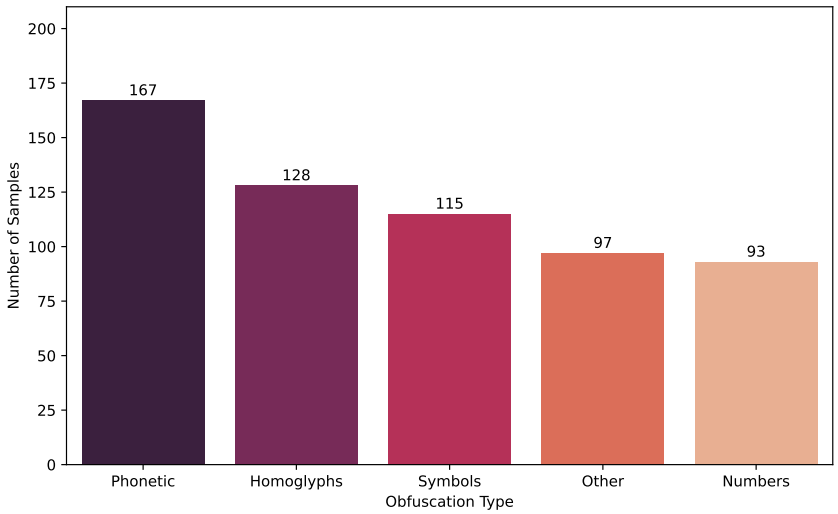


FIGURE 3.1. Distribution of obfuscation types.

diverse, containing substantial examples of multiple obfuscation types. In contrast, the remaining datasets tend to be more skewed, with certain obfuscation types dominating while others are scarcely represented.

Since individual posts often contained multiple obfuscation strategies simultaneously, their intersections were explored using an UpSet plot (see Figure 3.3). The larger portion of obfuscations were represented by a single category, while 83 out of 501 cases involved combinations of multiple obfuscation types. Across all obfuscation categories, the majority of instances were represented as standalone types. Specifically, 61.7% of symbol-based, 75.4% of phonetic variations , 71.1% of homoglyphs, and 86.6% of "Other" obfuscations occurred independently, while the remaining cases involved combinations of multiple obfuscation types. Among these, 47 include numbers (approximately 50.5% of occurences that contained numbers), slightly exceeding their standalone occurrence count of 46. In contrast to other categories, this indicates that numbers are more likely to be used in combination with other obfuscation techniques, rather than appear independently.

Word-level differences between the flagged and cleaned tokens were measured using the Levenshtein distance (see Figure 3.4). For each pair of corresponding tokens that were flagged and their cleaned version (actual word that they represent) columns, the edit distance was computed. Zero distances, representing identical tokens, were excluded to focus exclusively on actual modifications, these cases were often parts of an expression therefore no valuable data is lost. The distribution of non-zero edit distances was analyzed to find patterns in token-level modifications, emphasizing the relative frequency of minor versus major changes. From this analysis, it is possible to observe that minor changes are the most prevalent, with an edit distance of 1 occurring 238 times, followed by distances of 2 (169), 3 (75), 4 (51), and 5 (46), indicating that most obfuscations involve small token-level edits. Edit distances greater than 10 often represented were rare and often corresponded to exceptional cases or blank characters that could not be filtered out by the applied methods.

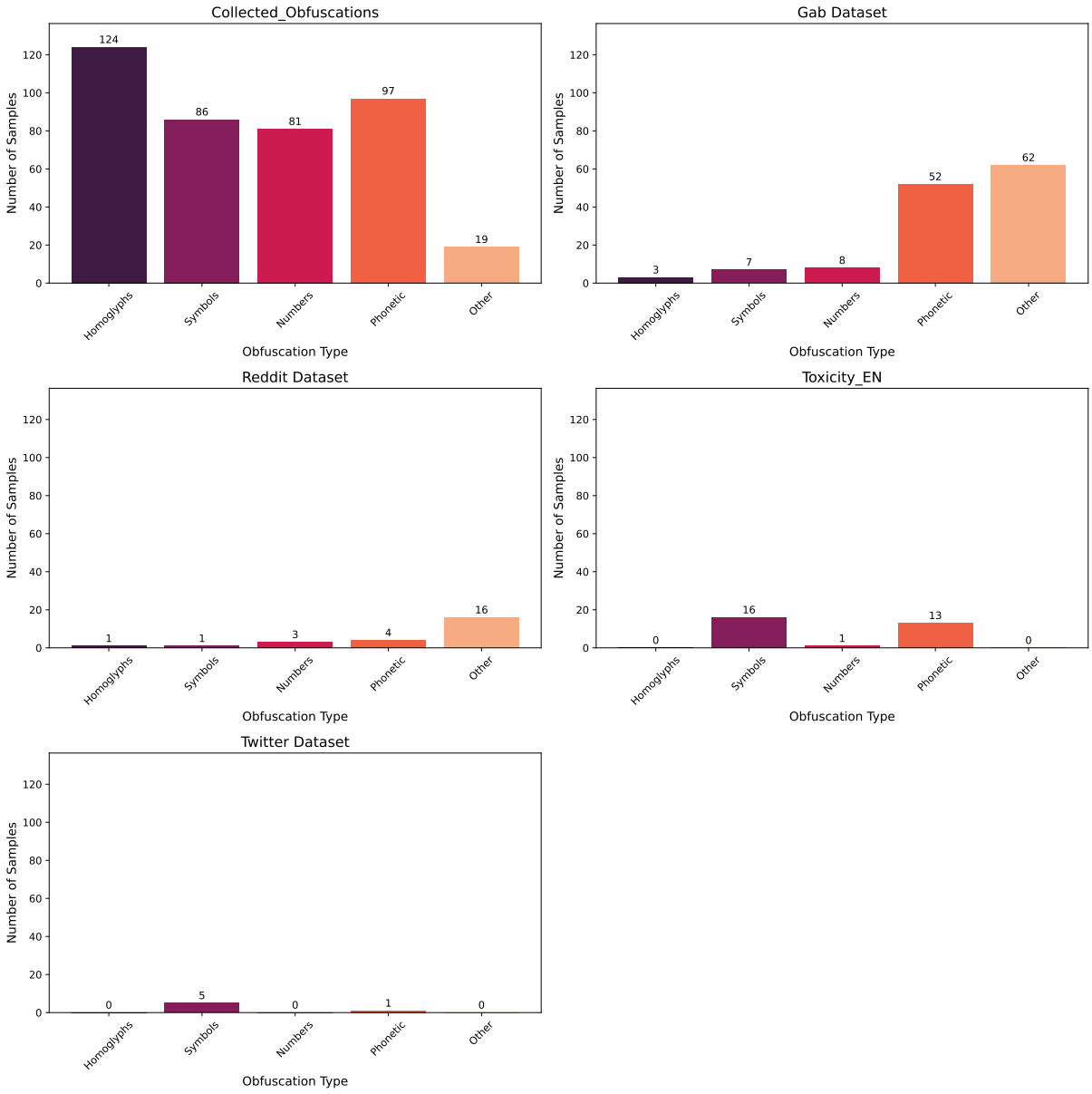


FIGURE 3.2. Distribution of obfuscation types.

Additionally, the analysis of flagged text revealed that a small set of offensive words dominates the dataset. The ten most frequent words were "nigger" (70 occurrences), "faggot" (46), "fuck" (46), "fucking" (45), "niggers" (36), "retard" (24), "bitch" (23), "nigga" (16), "tranny" (16), and "kill" (15), indicating that these terms account for a substantial proportion of the offensive language present.

Overall, these findings highlight both the patterns of obfuscation and the tendency for most cases to involve minor token-level edits alongside the dominance of a small subset of offensive terms, providing an understanding of the data that will support the development of the modeling phase in the next chapter.

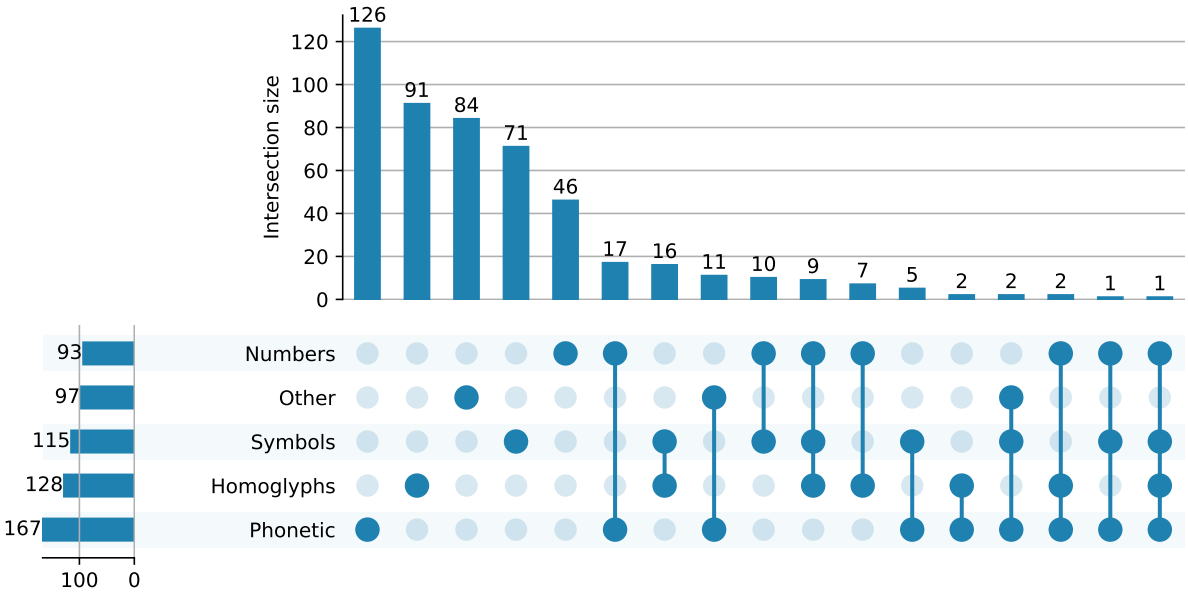


FIGURE 3.3. Co-occurrence patterns between the different obfuscation types.

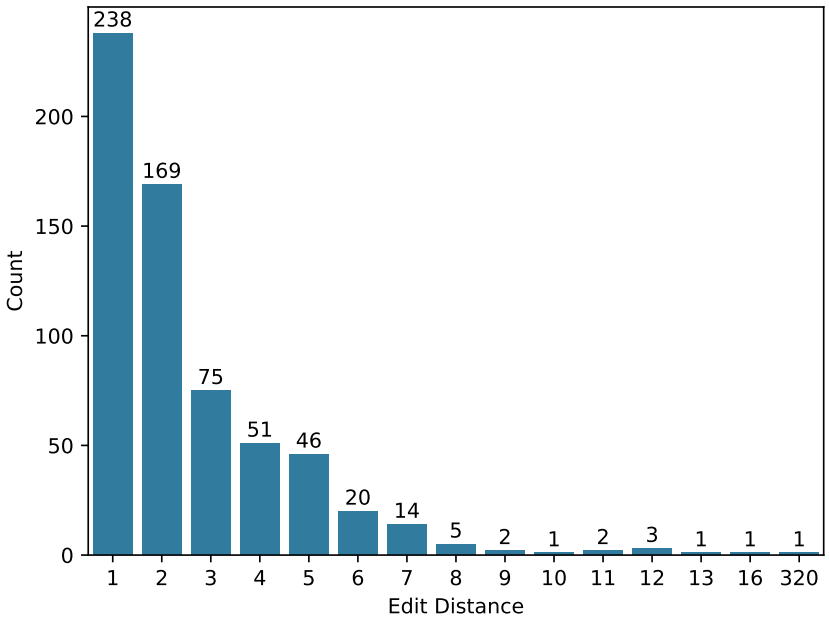


FIGURE 3.4. Frequency of word-level Levenshtein distances.

CHAPTER 4

Modeling

This phase focuses on exploring methods that could act as a "shield" against obfuscated words, aiming to map them as closely as possible to their canonical forms based on similarity before they reach abusive language detection systems. A range of approaches was explored, encompassing embedding-based, edit distance, and phonetic similarity methods, each representing different ways of measuring similarity. These methods were tested to assess how effectively they associate obfuscated inputs with their original words and to evaluate their potential to enhance the robustness of downstream detection. The pipeline developed for word cleaning, as well as the experiments conducted, are also documented in this phase.

4.1. Embedding-based Approaches

This section explores embedding-based methods that represent words as dense vector representations, enabling the detection of similarity between obfuscated and canonical forms.

4.1.1. FastText

FastText builds on the skip-gram model by representing each word as a bag of character n-grams. In this approach, every word is composed of overlapping subword segments, each represented as a vector, and the word's representation is obtained by summing its sub-n-gram vectors. This technique enables the model to generate embeddings for rare or unseen words by leveraging shared subword information, and also allows for fast training on large corpora like Common Crawl and Wikipedia (Bojanowski et al., 2017).

The pretrained `crawl-300d-2M-subword` word vectors were used in two ways. First, they were used directly to retrieve nearest neighbors for canonical and obfuscated terms, checking whether the similarity search naturally aligns variants with their originals. Second, the pretrained embeddings were extracted and compared via cosine similarity to rank candidates flexibly, enabling custom weighting and ranking strategies while still leveraging FastText's sub-n-gram representations.

4.1.2. CharacterBERT

CharacterBERT is a variant of the BERT architecture that replaces the standard WordPiece tokenization with a character-level convolutional network to generate word embeddings (El Boukkouri et al., 2020). Each token is represented as a sequence of characters

processed through convolutional filters, allowing the model to capture morphological information and handle out-of-vocabulary words without relying on fixed subword vocabularies. This property makes CharacterBERT particularly useful for tasks involving misspellings or obfuscated words. In this work, the pretrained CharacterBERT model was employed to generate contextual embeddings for both canonical and obfuscated terms, enabling a similarity-based comparison to assess alignment between altered and original words.

4.1.3. CANINE

CANINE (Character Architecture with No tokenization In Neural Encoders) embeddings derive from a character-level transformer model that processes text without tokenization, representing words directly as sequences of Unicode characters (Clark et al., 2022). This design allows the model to natively handle misspellings and rare words by encoding character sequences instead of relying on a predefined vocabulary. For this work, the CANINE-S (subword) variant was selected, as it is tailored for token-level tasks such as similarity comparison, making it more suitable for evaluating obfuscated words against their canonical forms. CANINE's embeddings were extracted from the pretrained model and used to compute similarity between obfuscated and canonical words, providing a contextual, character-aware measure of closeness that is independent of tokenization artifacts.

4.1.4. BPEmb

BPEmb (Byte-Pair Encoding embeddings) represents words using subword units learned via byte-pair encoding, allowing for robust representations of rare and unseen words (Heinzerling & Strube, 2018). Each word is split into frequent subword segments, which are then embedded and combined to form the final word representation. This approach was used to encode both canonical and obfuscated terms, with cosine similarity computed over the resulting embeddings to evaluate whether subword-based representations effectively align obfuscated variants with their original words. Different parameterizations were tested. For instance, a 300-dimensional embedding space with a 50,000 vocabulary size was selected, offering a balance between coverage and efficiency. This setup was intended to capture obfuscated word forms while maintaining meaningful subword structure in the embedding space.

4.2. Edit Distance Approaches

The Levenshtein distance measures the minimum number of single-character edits, including insertions, deletions, or substitutions, needed to transform one word into another (Levenshtein, 1966). It is commonly used in text processing for spelling correction and fuzzy matching, providing a numeric measure of similarity between words.

The Damerau-Levenshtein distance extends the Levenshtein metric by including transpositions of adjacent characters as a single edit operation (Damerau, 1964). This accounts for common character swaps and provides a measure that can capture slight variations between a canonical word and its altered form.

The Jaro-Winkler distance computes similarity between two strings, giving higher weight to matching prefixes and character order (Jaro, 1989; Winkler, 1990). It produces a score between 0 and 1 and is often used in name matching and record linkage. The prefix adjustment helps detect variations that preserve initial characters.

All three distance measures were used to calculate similarity between obfuscated and canonical words. Candidate words were compared to each target word, and the resulting similarity scores were used to evaluate how well each method could link an altered form to its original. The same procedure was applied across the three metrics, allowing a direct comparison of their performance in capturing character-level differences.

4.3. Phonetic Similarity Approaches

Phonetic similarity assesses how words sound rather than how they are spelled. The Double Metaphone (Philips, 2000) algorithm was employed, which encodes words into primary and secondary phonetic representations, enabling comparisons based on pronunciation patterns. Matches on the primary code were considered exact, while secondary code matches received a slightly lower similarity score, reflecting close phonetic correspondence (Oubiwan, n.d.; Philips, 2000). This approach is suitable for handling obfuscated words that modify their spelling while retaining their pronunciation, as the phonetic encoding enables the assessment of similarity to the canonical form independently of orthographic variation.

4.4. Word Cleaning Pipeline

In order to provide a clear and structured overview of the subsequent steps, a diagram was created to illustrate the modeling pipeline at a high level (see Figure 4.1). This pipeline includes all the stages following the conclusion of the data preparation phase, which are part of the `word cleaning` function, in the `Word_Cleaning` jupyter notebook, that accepts the tokenized posts, their respective OOV words extracted and a cache when available. It follows the process of assessing OOV words that represent obfuscated offensive language and transforming them into forms that are equivalent or close to their canonical counterparts. When this is not possible, those words are flagged accordingly.

4.4.1. Cache

To avoid processing the same OOV word multiple times, a mechanism was implemented to store the result of each transformation. With this in mind, before any assessment, an initial check determines whether the OOV word has already been processed. If it has, no additional processing is performed, and the stored transformed word is considered in place of the obfuscated word.

4.4.2. Rule-based word treatment

Words that pass the previous check and meet basic criteria to avoid unwanted tokens, such as containing at least one alphanumeric character and not being convertible to a number, undergo a series of normalization steps. Firstly, the presence of homoglyphs, which are

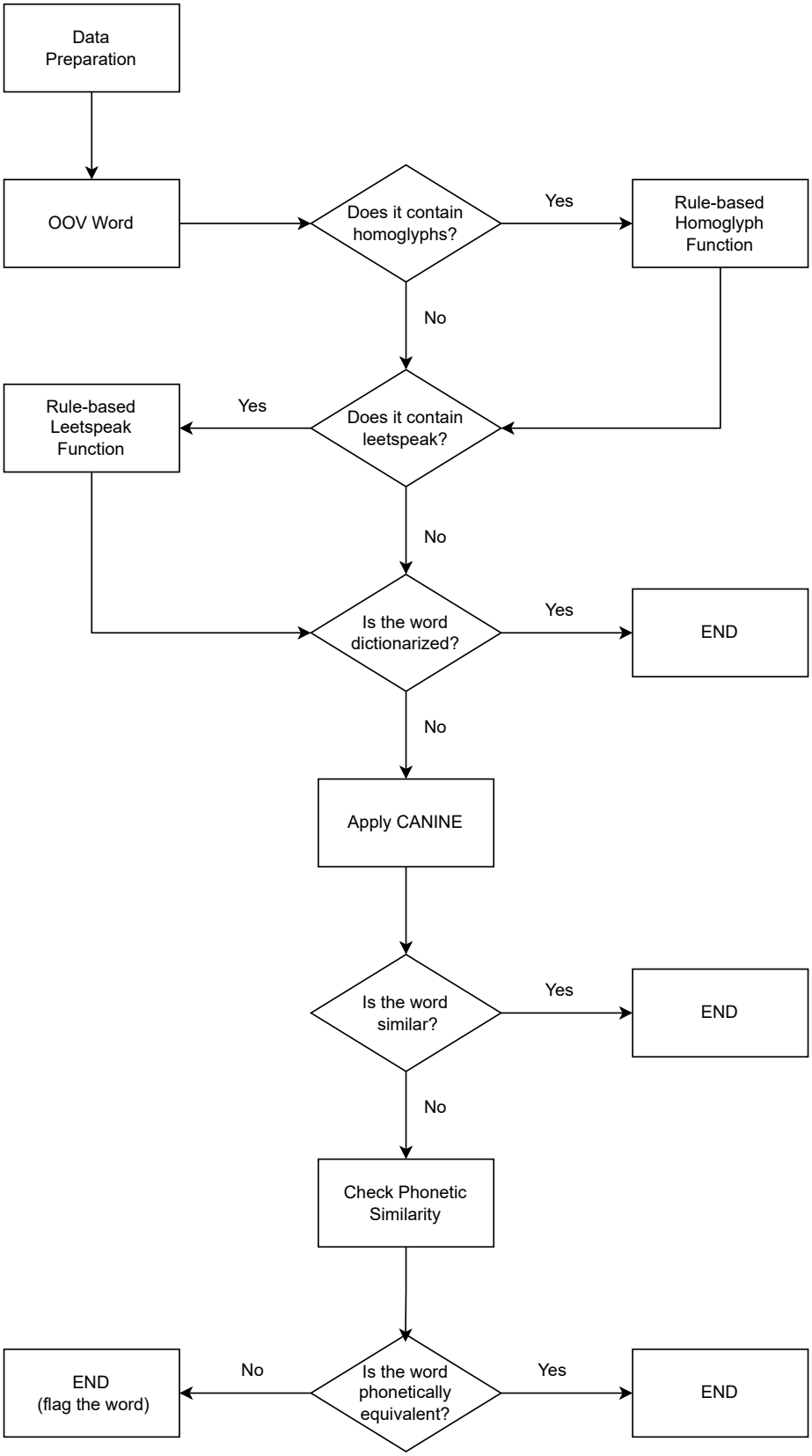


FIGURE 4.1. Pipeline for the word cleaning process to restore obfuscated words to their canonical form.

visually similar characters, is checked. When detected, they are replaced according to a predefined mapping. Following this, the `unicodedata` and `text_unidecode` libraries are used to perform a final normalization step, ensuring that any remaining diacritics are stripped and the token is fully converted to its closest ASCII form.

Secondly, the tokens are checked for characters commonly used in leetspeak and undergo an additional normalization step to convert numeric or symbolic substitutes back to their canonical alphabetic forms. This process identifies common leetspeak substitutions (refer to Appendix C), such as replacing "4" with "A" or "3" with "E", and applies a predefined mapping to restore these characters to their standard equivalents. A context-aware mechanism was added to ensure that ambiguous symbols, such as "1" or "!", are mapped to the appropriate letter (L or I) depending on surrounding characters, which helps accurately restore words where a single symbol could represent multiple possibilities. By doing so, the representation of the word becomes more consistent with its canonical form, allowing for more accurate matching at a later stage.

Upon the conclusion of the rule-based treatment, the word is verified against a dictionary, leveraging enchant library's EN-US dictionary, to determine whether it already corresponds to a canonical form. If the word is now dictionarized, it is considered normalized and recorded in the cache. If not, it moves on to the next processing phase aimed at attempting to identify its canonical form through word similarity.

For words within a sentence that were not identified as OOV, no checks or normalization steps are applied, and they pass through the pipeline unchanged. This ensures that the original structure of the sentence is preserved, with only the obfuscated words being replaced by their normalized counterparts.

4.4.3. Word similarity

At this stage, given that the word is not present in Surge AI's profanity lexicon and is not a standalone character, the process continues. Following this, using CANINE to generate embeddings, the cosine similarity between the word and the remaining candidates is computed. While the five most similar candidates are identified, only the top-ranked candidate is retained, together with its associated similarity score. Additionally, the Damerau-Levenshtein distance is calculated to serve as a complementary criterion alongside the similarity score. The next steps are determined based on the following criteria:

(1) If the word is not exactly the same as the candidate ($edit_distance > 0$) and is not very similar ($similarity < 0.75$).
(2) If the word is not exactly the same as the candidate ($edit_distance > 0$) but is very similar ($similarity >= 0.75$).
(3) If the word is similar enough to be considered identical to the candidate ($similarity >= 0.75$).

In the first case, a last attempt at finding similarity between the words was performed through testing for phonetic simlarity. If the obfuscated and candidate words share the

same primary phonetic representation, they are considered equivalent, and the candidate replaces the obfuscated form. Otherwise, the word is marked as unsolved and added to a list of problematic cases for future inspection.

In the second case, where the word is not identical to the candidate but reaches a sufficiently high similarity score, the candidate is selected as the replacement. These instances are considered partially solved, since the approximation is strong enough to justify substitution but not exact, and are recorded in the cache as such.

In the third case, when the similarity threshold is high enough to treat the words as effectively identical, the candidate is accepted as the canonical form. These words are marked as solved and stored in the cache, as the confidence in the match is deemed sufficient to consider the obfuscated form successfully normalized.

4.5. RoBERTa Toxicity Classifier Experiment

RoBERTa Toxicity Classifier is a contextual language model fine-tuned for toxicity detection. Built on the RoBERTa transformer architecture, it produces sentence-level predictions indicating the presence of toxic content (s-nlp, 2021). The model was applied to sentences containing obfuscated offensive words to evaluate whether the classifier can recognize toxicity despite altered spellings. This allows for assessing the model's robustness against obfuscation in real-world abusive language detection scenarios.

4.6. Experiments with Large Language Models

All three following models were used to attempt the generation of obfuscated offensive examples and perform normalization on the 501 posts that contained obfuscated words. It is worth noting that LLaMA and Deepseek were executed locally through LM Studio while OpenAI's ChatGPT-4o was accessed via the official ChatGPT interface. This evaluation provided insight into how smaller, open-weight models, as well as a large-scale commercial model, handle obfuscation and canonicalization. The outputs were later compared to the results obtained from the structured normalization pipeline, allowing for an assessment of differences in performance between local models, a commercial-grade model, and the pipeline's approach to handling obfuscated offensive language.

4.6.1. LLaMA-3-8B-Instruct_Q4_K_M

LLaMA-3 (Large Language Model Meta AI) is Meta's third-generation family of open-weight transformer models designed for efficiency and high performance across reasoning, text generation, and instruction-following tasks (Touvron et al., 2023). The 8B Instruct variant contains 8 billion parameters and has been fine-tuned for following natural language instructions. For this work, a `Q4_K_M` quantized version was employed, which compresses weights into a 4-bit k-means representation while preserving alignment and usability. This quantization significantly reduces memory requirements, enabling local inference on consumer hardware while maintaining instruction-following fidelity. The model was selected for its balance of efficiency and capacity, allowing controlled experiments.

4.6.2. DeepSeek-R1-Distill-Qwen-7B

DeepSeek-R1-Distill-Qwen-7B is a distilled version of the Qwen-7B model, a large language model released by Tencent (DeepSeek-AI et al., 2025). Distillation transfers knowledge from a larger DeepSeek-R1 reasoning model into a smaller 7-billion-parameter architecture, preserving semantic understanding and contextual reasoning while reducing computational overhead. This approach results in competitive performance across benchmarks, particularly in mathematics, code generation, and reasoning, while being suitable for local deployment. The distilled model was used to test how a compact, efficiency-oriented architecture manages obfuscated offensive text, providing a contrast to both larger commercial systems and structured normalization pipelines.

4.6.3. ChatGPT-4o

ChatGPT-4o (short for "omni") is OpenAI's multimodal flagship model, capable of processing and generating text, images, and audio in real time (OpenAI et al., 2024). Released as the successor to GPT-4 Turbo, GPT-4o integrates perception and reasoning across modalities within a unified transformer framework. Its design emphasizes both alignment and high-capacity reasoning, enabling natural conversation, contextual understanding, and robust handling of complex language phenomena. ChatGPT-4o's inclusion allowed an evaluation of how cutting-edge commercial architectures handle the dual tasks of generating obfuscated offensive examples and normalizing obfuscations.

CHAPTER 5

Results and Discussion

This chapter presents the results obtained from the approaches described in the Modeling phase. Both successful and unsuccessful methods are compared and discussed.

5.1. Models and embeddings

Firstly, using FastText embeddings to compare the canonical form "faggot" to various obfuscated versions of the word, it is understandable that the performance greatly varies depending on the type of obfuscation (see Table 5.1). For character manipulations that involve only letters, such as "fagget" (0.7598), "phaggot" (0.6465), and "faggt" (0.4296), the similarity scores increase compared to variations incorporating symbols or numbers, such as "f4gg0t" (0.1947), "fa66ot" (0.1048), or "f@g0t" (0.2421). These results highlight that FastText struggles when dealing with obfuscations that make use of leetspeak or homoglyphs, yet performs moderately for variants where only the spelling changes, provided that the alteration preserves the phonetic form, as in "fagget" and "phaggot".

TABLE 5.1. Cosine similarity between *faggot* and obfuscated variants using FastText embeddings.

Pair	**Similarity Score**
faggot ↔ f4gg0t	0.1947
faggot ↔ **fagget**	**0.7598**
faggot ↔ phaggot	0.6465
faggot ↔ faggt	0.4296
faggot ↔ fa66ot	0.1048
faggot ↔ f@g0t	0.2421

Additionally, the FastText pre-trained word vectors, crawl-300d-2M-subword, were used to extract the top 10 most similar words to both "faggot" (see Table 5.2) and "f4ggot" (see Table 5.3), in order to assess once more the model's reliability in dealing with obfuscated variants. These experiments revealed that even a simple substitution of "a" with "4" can produce unexpected results, such as "depolarization-evoked" achieving a similarity score of 0.5523, which is completely unrelated. For further testing, it was attempted the same method but now with the word "faggg00tt". As expected, none of the words close to the canonical form were aligned with the canonical form, however the most similar ones now consisted of equally offensive insults (see Table 5.4).

Next, CharacterBERT was evaluated using the same approach as FastText, for its embeddings. In contrast to FastText, CharacterBERT's performance was more balanced, among the top 10, the lowest similarity score was 0.4558 and the highest 0.5180. This

TABLE 5.2. Top-10 most similar words to *faggot* according to the FastText pre-trained word vectors.

Word	**Similarity Score**
fag	0.8209
faggots	0.8065
fagget	0.7598
faggit	0.7572
Faggot	0.7524
faggy	0.7159
homophobe	0.6870
fagg	0.6852
fagot	0.6846
fucktard	0.6828

TABLE 5.3. Top-10 most similar words to *f4ggot* according to the FastText pre-trained word vectors.

Word	**Similarity Score**
depolarization-evoked	0.5523
hemopneumothorax	0.5494
curcumin-loaded	0.5462
re-corded	0.5457
spiral-wrapped	0.5452
pentadecane	0.5332
hydrophobized	0.5301
pan-cooked	0.5233
hypomyelinating	0.5225
dislayed	0.5207

TABLE 5.4. Top-10 most similar words to *faggg00tt* according to the FastText pre-trained word vectors.

Word	**Similarity Score**
coonie	0.7332
ass fucker	0.6856
ape shit	0.6803
assmuncher	0.6600
coonies	0.6580
ass hole	0.6576
asshole fucker	0.6537
sand nigger	0.6513
ass fuck	0.6495
ass-bandit	0.6464

indicates that although the similarity scores were generally low, they were relatively consistent across all types of obfuscations (see Table 5.5). These two models were disregarded for further testing, since their results were not promising, specially in a controlled environment where they were exposed to a diverse selection of obfuscation types and yielded poor similarity scores.

TABLE 5.5. Cosine similarity of obfuscated words to *faggot* using CharacterBERT embeddings.

Canonical ↔ Obfuscated	**Cosine Similarity**
faggot ↔ f4gg0t	0.5180
faggot ↔ fagget	0.5551
faggot ↔ phaggot	0.4638
faggot ↔ faggt	0.4677
faggot ↔ fa66ot	0.3551
faggot ↔ f@g0t	0.4795
faggot ↔ phegget	0.4606
faggot ↔ paggot	0.4595
faggot ↔ ffagggottt	0.4768
faggot ↔ faaaaaggott	0.4558

The following methods will be evaluated using the previously selected offensive language lexicon. With the limitations observed earlier in mind, the approaches must provide satisfactory similarity scores when matching obfuscated terms to their canonical forms within the lexicon. For consistency, the obfuscated word used throughout the testing phases is "faggg00tt" with the exception of one experimental case involving the RoBERTa Toxicity Classifier.

As a next step, CANINE was tested. As shown in Table 5.6, CANINE was able to position well the canonical form "faggot" (0.7843) at the top of the similarity rankings for the heavily obfuscated input "faggg00tt". Other offensive words such as "faggots" (0.7633), "fucktards" (0.5791), and "fuckass" (0.5690) appear within the top 5. Notably, there is a sharp drop in similarity from "faggots" to "fucktards," indicating that CANINE effectively captures character-level similarities. The plural form of the canonical word is much closer than unrelated words, highlighting the model's sensitivity to spelling changes while preserving semantic relevance.

TABLE 5.6. Top-10 most similar words to *faggg00tt* according to the CANINE embeddings.

Word	**Similarity Score**
faggot	0.7843
faggots	0.7633
fucktards	0.5791
fucktard	0.5690
fuckass	0.5590
assfuck	0.5386
buttfuck	0.5368
butt fuck	0.5336
fuckasses	0.5305
jiggas	0.5274

Following CANINE, the BPEmb model was evaluated on the same obfuscated input. By analyzing Table 5.7, BPEmb ranked "faggots" (0.608) as the most similar, followed by

"fag" (0.555) and "faggot" itself (0.508). While the canonical form appears within the top three, the model demonstrates less sensitivity to character-level obfuscations compared to CANINE, as indicated by the relatively lower similarity scores and the presence of unrelated or loosely related offensive terms, such as "kkk" (0.303) and "cums" (0.302), among the top ten. These results suggest that BPEmb captures semantic similarity at a broader subword level but struggles to consistently prioritize the exact canonical word in highly obfuscated cases.

TABLE 5.7. Top-10 most similar words to *faggg00tt* according to the BPEmb model.

Word	**Similarity Score**
faggots	0.608
fag	0.555
faggot	0.508
arse-fucker	0.349
arsewipe	0.349
kkk	0.303
cums	0.302
fucker	0.289
arse	0.288
dicksmoker	0.278

Lastly, for a different purpose but maintaining the goal of dealing with obfuscated words, the RoBERTa Toxicity Classifier was applied to a selection of texts containing obfuscated offensive words (see Table 5.8). The model produced varying toxicity probabilities depending on the context in which the obfuscated words appeared. For example, "You are a ph4ggot" received a high toxicity score (0.9968), whereas "You are a nice ret@rdd and kind ph4ggot" was assigned a much lower score (0.0726). These results highlight that the predicted toxicity is strongly influenced by context, demonstrating a limitation of automated offensive language detection systems, which will not be further explored in this thesis, but is worth noting that may affect the final prediction despite shielding the model against obfuscations. Despite this problem, the successful labeling of "You are a ph4ggot" as highly toxicity demonstrates robustness against character-level adversarial attacks.

TABLE 5.8. Toxicity classification probabilities of sample texts using RoBERTa Toxicity Classifier.

Text	**Toxic**	**Neutral**
Yes I no understanding you retared fucknuts me need more retarded dictionary tanks now I live real American life tanks b to gawd	0.9966	0.0034
You are a nice ret@rdd and kind ph4ggot	0.0726	0.9274
You are a ph4ggot	0.9968	0.0032
I don't know what this f4gget means by that but sure	0.0226	0.9774

5.2. Edit Distance

Regarding edit distance, Levenshtein edit distance was one of the approaches to evaluate similarity (see Table 5.9). As expected, the canonical form "faggot" and its plural "faggots" achieved the highest scores (0.5556), though these values are still mediocre and illustrate its limitations when applied to heavily perturbed words. Shorter variants such as "fag" also appeared in the ranking (0.3333). Beyond these, however, some unrelated terms were ranked with low scores (e.g., ape shit, asshat, bangbros), showing that Levenshtein edit distance, while sensitive to surface-level modifications, is not suitable to be applied as a standalone criteria for ranking.

TABLE 5.9. Top-10 most similar words to *faggg00tt* according to Levenshtein distance.

Word	**Similarity Score**
faggot	0.5556
faggots	0.5556
fag	0.3333
ape shit	0.2222
arseshit	0.2222
asshat	0.2222
bangbros	0.2222
batshit	0.2222
dago	0.2222
darkshit	0.2222

Applying Damerau-Levenshtein distance produced slightly higher similarity scores compared to plain Levenshtein, with "faggot" and "faggots" (0.6923) ranked at the top, followed by "fag" (0.6000) (see Table 5.10). This reflects the algorithm's advantage of considering adjacent transpositions in addition to insertions, deletions, and substitutions. However, as with Levenshtein, a cluster of unrelated terms (e.g., ape shit, asshat, bangbros) also appeared with average scores (0.5625), which also undermines its usefulness as a standalone method. While it shows improved sensitivity over Levenshtein, the presence of this noise confirms its limitations in reliably distinguishing between canonical forms and unrelated words.

The Jaro-Winkler metric yielded the strongest results among the string-matching methods tested, with "faggot" (0.8800), "faggots" (0.8500), and "fag" (0.8400) clearly separated from other candidates (see Table 5.11). This indicates that this approach is particularly well-suited to capturing similarity when obfuscations preserve the beginning of the word, which can also explain why other candidates with similar initial characters sequence have a certain position in the ranking (e.g., fatass, fugly). Nevertheless, beyond the top matches, unrelated words were still ranked relatively higher than expected, showing that while more discriminative than Levenshtein-based approaches, Jaro-Winkler can still be unreliable when applied in isolation when dealing with obfuscated words. Its

TABLE 5.10. Top-10 most similar words to *faggg00tt* according to Damerau-Levenshtein distance.

Word	Similarity Score
faggot	0.6923
faggots	0.6923
fag	0.6000
ape shit	0.5625
arseshit	0.5625
asshat	0.5625
bangbros	0.5625
batshit	0.5625
dago	0.5625
darkshit	0.5625

sharper separation, however, suggests a possibly stronger potential when combined with other methods in hybrid approaches.

To assess whether the potential of Jaro-Winkler distance is consistent, a slight variation was introduced to the obfuscated input by replacing the initial "f" with "ph," resulting in the form "phaggg00tt". As a result, the top-ranked candidates suffered a noticeable drop in similary score. For Jaro-Winkler, both "faggot" and "faggots" decreased by 0.19 (see Table 5.12), while under Damerau-Levenshtein the decrease was more modest, at 0.0673 (see Table 5.13). This difference emphasizes the fact that Jaro-Winkler is particularly sensitive to substitutions at the beginning of a word. In contrast, Damerau-Levenshtein is less affected by such changes, as it treats character edits more uniformly across the word. Consequently, it was decided to proceed with Damerau-Levenshtein distance while disregarding the other alternatives.

TABLE 5.11. Top-10 most similar words to *faggg00tt* according to Jaro-Winkler distance.

Word	Similarity Score
faggot	0.8800
faggots	0.8500
fag	0.8400
fatass	0.6100
fatasses	0.5900
fugly	0.5900
dago	0.5700
fuckass	0.5500
fucking	0.5500
batshit	0.5400

5.3. Phonetic Similarity

Phonetic similarity was evaluated using the Double Metaphone algorithm (see Table 5.14). All tested variants of the word fucker (phukker, fuker, and phuqer) produced identical primary codes to the canonical form, resulting in a similarity score of 1.0. This demonstrates

TABLE 5.12. Top-10 most similar words to *phaggg00tt* according to Jaro-Winkler distance.

Word	**Similarity Score**
faggot	0.6900
faggots	0.6600
fag	0.6200
dipshit	0.5800
dago	0.5700
dipshits	0.5600
ape shit	0.5500
chingchong	0.5300
jiggs	0.5300
beat my meat	0.5200

TABLE 5.13. Top-10 most similar words to *phaggg00tt* according to Damerau-Levenshtein distance.

Word	**Similarity Score**
faggot	0.6250
faggots	0.6250
ape shit	0.5556
arseshit	0.5556
asshat	0.5556
bangbros	0.5556
batshit	0.5556
chingchong	0.5556
dago	0.5556
darkshit	0.5556

TABLE 5.14. DoubleMetaphone codes and phonetic similarity of obfuscated words to *fucker*.

Pair	**Phonetic Codes**	**Similarity Score**
fucker ↔ phukker	FKR ↔ FKR	1.0
fucker ↔ fuker	FKR ↔ FKR	1.0
fucker ↔ phuqer	FKR ↔ FKR	1.0

that phonetic similarity effectively captures obfuscated words that preserve pronunciation despite changes in spelling.

5.4. Word Cleaning Pipeline

After selecting the most effective methods from each similarity strategy, the next step was to evaluate the full pipeline and assess the effectiveness of the ensemble approach. Across the 501 posts, a total of 1337 OOV words were processed, of which 1136 were unique. It is also worth noting that, among the manually flagged words, 761 were identified, with 626 of them being unique. If considering the cleaned version of these flagged obfuscations, the number of unique words drops down to 299. Taking these observations into account, this section presents an analysis of the results obtained from the processing stage.

As previously outlined, the pipeline consists of methods to treat homoglyphs, leetspeak, and subsequently assess similarity against a lexicon. Here, the technique usage is measured by counting the number of posts in which the corresponding treatment was applied at least once, that is, the number of rows with non-zero values for each technique (see Figure 5.1). These technique usage statistics reveal that 219 ($\approx 16.4\%$) OOV words required the treatment of homoglyphs, a number that overwhelms the fact that 128 posts contain obfuscations explicitly identified as homoglyph-based. The difference in values can be explained by the possibility of having more than one obfuscated word on the same post. Another contributing factor is having obfuscated words that were not flagged as offensive language but were still perturbed to mask the context around the offensive word in the post.

Regarding Leetspeak, treatment was applied to 618 ($\approx 46.2\%$) OOV words, covering the originally accounted for 208 posts with obfuscations using symbols and numbers. The additional instances of its usage are, similarly to the homoglyphs case, non-offensive obfuscated words that were not flagged, since no data preparation process was capable fully eliminating undesired tokens, residual noise or obfuscated words part of context. A larger disparity than in the previous case is not unexpected, since obfuscations based on symbols and numbers are generally more common than those involving homoglyphs.

While leetspeak and homoglyph treatments can address a portion of the obfuscations present in the posts, similarity scoring is applied when these methods prove insufficient. For that end, CANINE approach was applied 937 times ($\approx 70\%$) and phonetic similarity was used 871 times ($\approx 65.1\%$). It is worth noting that, along the cleaning process, the cache was used 78 times ($\approx 5.8\%$). Although this is a relatively small proportion, it still contributes to time and computational savings in the long run, demonstrating its potential.

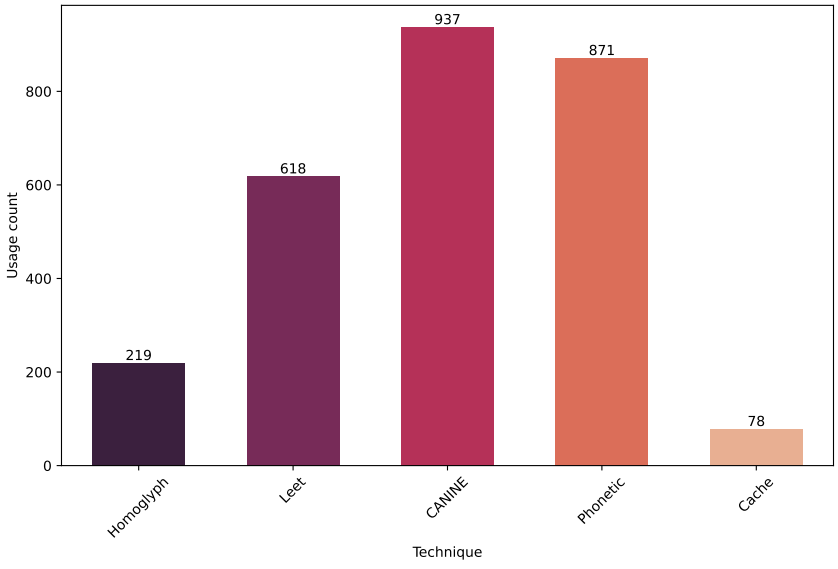


FIGURE 5.1. Total technique applications.

With the conclusion of the cleaning process (see Figure 5.2), 431 occurences were deemed solved (≈ 32.2%), 66 partially solved (≈ 5%) and 831 unsolved (≈ 62.1%). Despite these results, considering that 1337 OOV words were isolated in total, and that the actual words of interest, which are the flagged words, amount to 761, the performance of the process is reasonable considering its simplicity. Additionally, some words, even though they represent offensive language, were missed by the OOV isolation technique, which is expected knowing its coverage is not absolute. In total, 48 flagged words were not captured by this method.

For a fair comparison, excluding as much noise as possible, performance will be assessed based on the flagged words rather than the total OOV words. In this case, 376 out of 761 (≈ 49.4%) flagged words were correctly normalized while the remaining 385 ((≈ 50.6%)) were not. These results illustrate more clearly the effectiveness of the pipeline in a scenario where, hipotetically, the OOV process would have been flawless. This outcome is not completely irrealistic, as more refined and state-of-the-art approaches, like modern Large Language Models (LLMs), would likely achieve improved results while being better at generalizing.

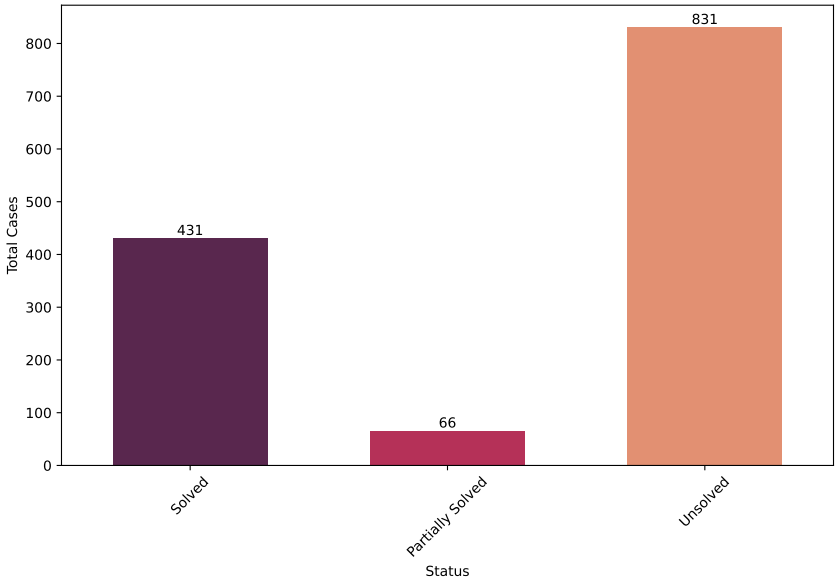


FIGURE 5.2. Total cases final status.

5.5. Experiments

Both local deployments of LLaMA and DeepSeek proved incapable of handling the specific task of obfuscation normalization and generation, although they were able to respond adequately to trivial or general-purpose queries. As a result, the only model successfully tested in this context was ChatGPT-4o. For this experiment, the word cleaning process was carried out by providing the model with a JSON file containing the posts and prompting it to identify and transform obfuscated words back into their canonical forms, which were then stored as lists for each post. Through comparing these results against

the clean version of the flagged words, out of the 761 flagged words it correctly cleaned 585 ($\approx$ 76.9%) and missed 176 ($\approx$ 23.1%). As expected, an LLM has an overall better performance than the rule-based pipeline, given its generalization capabilities, contextual inference and ability to capture semantic relationships. However, all this comes at a higher computational cost.

Regarding the generation of obfuscated examples with Surge AI's offensive language lexicon as reference, after extensive prompting to further refine them to better align towards the thesis goals, the pipeline's performance against this new synthetic data was tested.

An identical analysis to that performed on the finalized dataset was carried out on the generated data. The distribution of obfuscation types among the 397 entries is presented in Figure 5.3. During the annotation phase, 922 words were flagged.

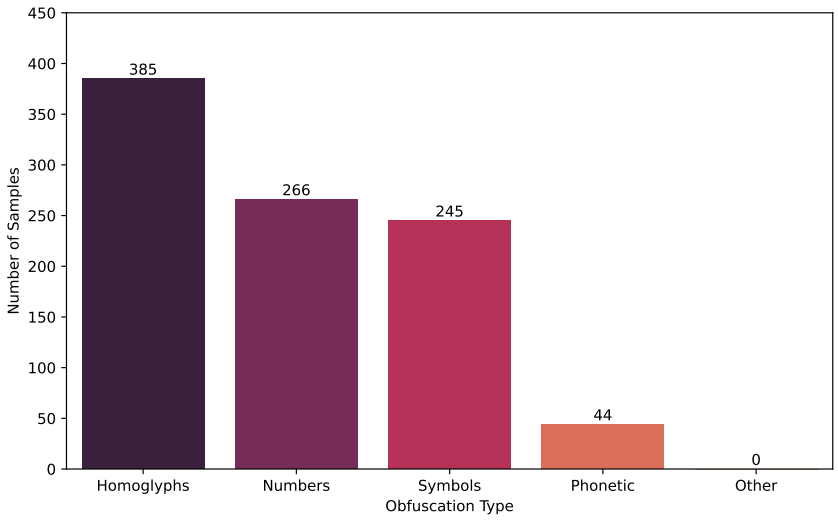


FIGURE 5.3. Distribution of obfuscation types (generated data).

In this case, the majority of obfuscations are based on homoglyphs (385), while phonetic similarities (44) are the least frequent. A noticeable imbalance is present, with numbers and symbols with similar mid-range values, 266 and 245, respectively.

The presence of multiple obfuscation strategies simultaneously also applied to the generated data, therefore their co-occurence was analyzed (see Figure 5.4). Contrary to the original dataset, obfuscations were mainly represented as combinations in the generated data, totaling 343 ($\approx$ 86.4%) out of 397 cases, while standalone types accounted for only 54 ($\approx$ 13.6%). Among the standalone types, the homoglyph-based category was the most frequent, with 44 instances, which is small fraction of the total 385 entries where homoglyphs occurred.

Word-level differences between the flagged and cleaned flagged tokens were measured using the Levenshtein distance (see Figure 5.5). From this analysis, it is possible to observe that distances were more diverse in the generated data. With an edit distance of 2 being the most prevalent, occurring 202 times, followed by distances of 1 (189), 4 (173),

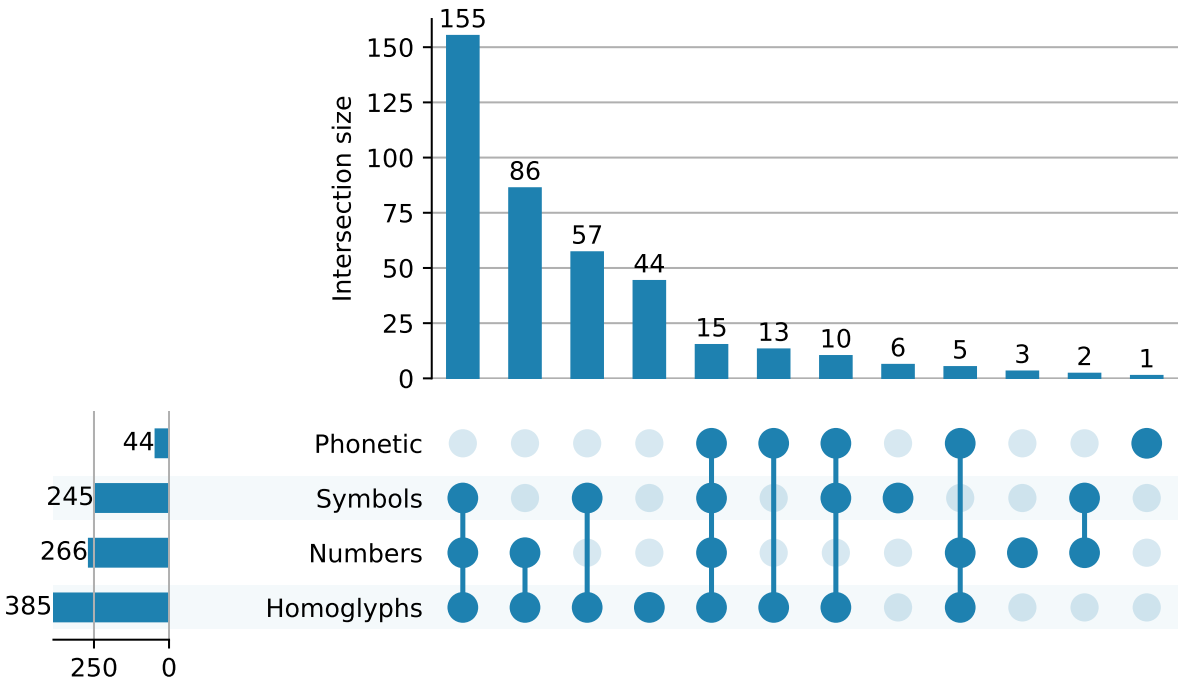


FIGURE 5.4. Co-occurrence patterns between the different obfuscation types (generated data).

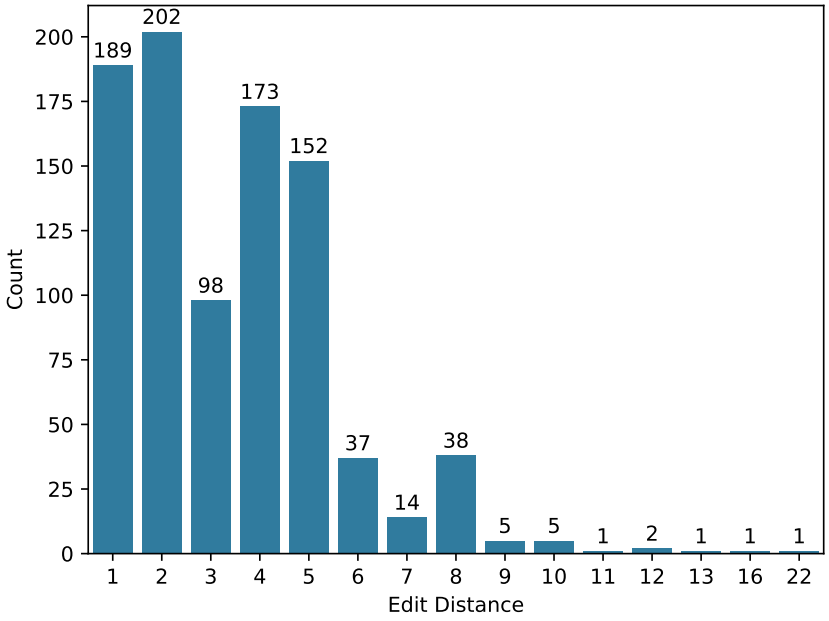


FIGURE 5.5. Frequency of word-level Levenshtein distances (generated data).

5 (152), and 3 (98), indicating that most obfuscations involve fewer than 5 edits. Edit distances greater than 5 often represented excessive obfuscations.

Analysis of the flagged text revealed that a small subset of offensive words dominates the dataset. The five most frequent terms were "nigger" (159 occurrences), "spic" (83), "faggot" (54), "coon" (52), and "retard" (45), collectively representing a substantial proportion of the offensive language present.

Across the 397 generated posts, a total of 1135 OOV words were processed, of which 427 were unique. It is also worth noting that, among the manually flagged words, 922 were identified, with 300 of them being unique. Considering the cleaned version of flagged obfuscations, the number of unique words drops down to 137. These values preemptively indicate that while there is diversity in terms of edit distance and a balanced technique application (see Figure 5.6), it came at the cost of diversity when generating the obfuscated examples. This lack of diversity can be infered from the lower counts of unique words present for OOV words, particularly, flagged and cleaned flagged words.

Having applied the word cleaning process, technique usage statistics reveal that 483 ($\approx$ 42.5%) OOV words required the treatment of homoglyphs, and 466 ($\approx$ 41%) the treatment of leetspeak. As for similarity scoring, CANINE approach was applied 511 times ($\approx$ 45%) and phonetic similarity was used 451 times ($\approx$ 39.7%). The cache was used 424 times ($\approx$ 37.3%). Although this time the distribution of technique usage appears more balanced and the cache proves to be as impactful as the other methods, this further emphasizes that the offensive language obfuscations are less diverse. This observation is reinforced by the noticeable increase in cache utilization with the generated data compared to the original dataset, suggesting that repeated obfuscation patterns are more prevalent in the GPT generated data.

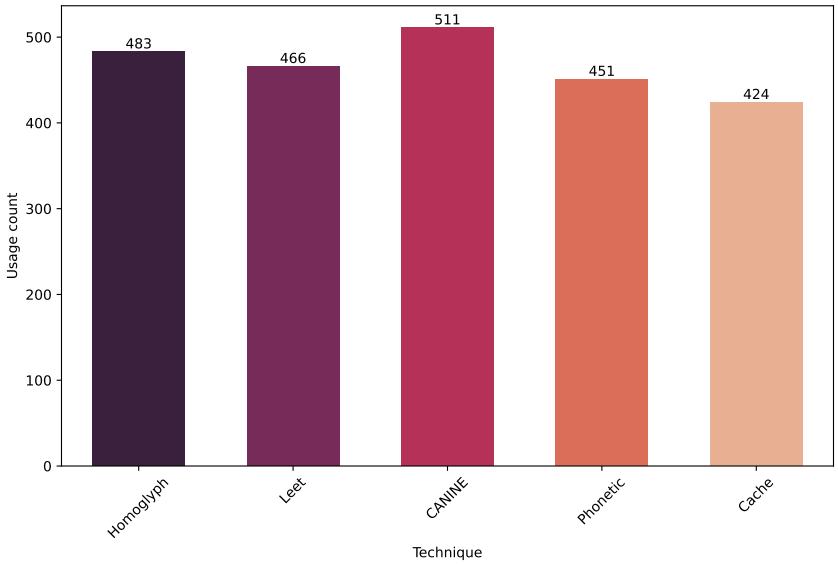


FIGURE 5.6. Total technique applications (generated data).

To sum up the results regarding the generated data and conclude the experimental phase (see Figure 5.7), 631 occurences were deemed solved ($\approx$ 55.5%), 60 partially solved ($\approx$ 5.2%) and 444 unsolved ($\approx$ 39.1%).

Considering the same fair comparison performed for the original data, 598 out of 922 ($\approx$ 64.9%) flagged words were correctly normalized while the remaining 324 (($\approx$ 35.1%)) were not. These results indicate that improvements in the proccesses of isolating OOV

words and refinement of lexicons of offensive language have a strong influence on the outcome of the word cleaning process.

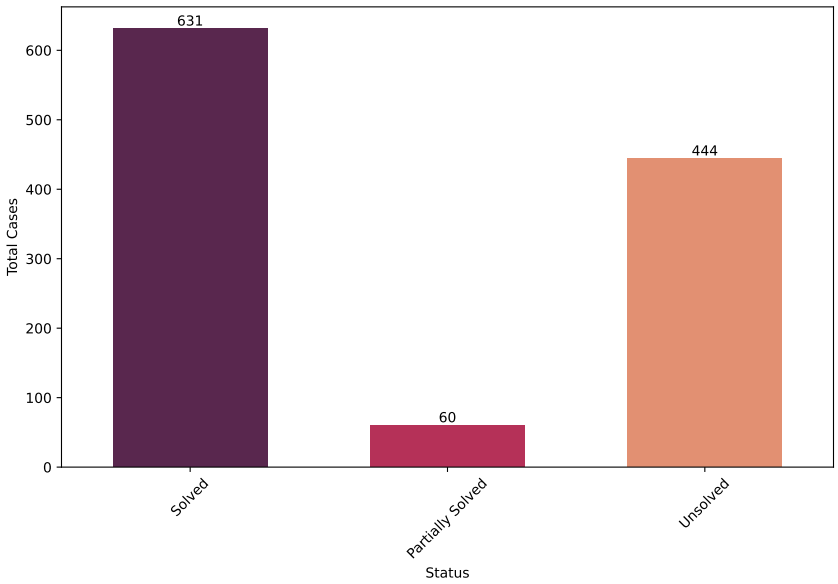


FIGURE 5.7. Total cases final status (generated data).

CHAPTER 6

Conclusions

This thesis addresses the challenges within NLP of dealing with character-level obfuscations deliberately used to fool offensive language detection systems. Following the CRISP-DM methodology, the problem at hand was contextualized and revealed the motivation behind the work. As data was gathered, it became clear that there was a lack of resources of this nature, which limited the availability of suitable data for studying character-level obfuscations in offensive language. To overcome this limitation, and answering the first research question (RQ1), in addition to manual data gathering from occurences in social media and game chats, an approach to attempt to extract valuable examples from the data available online was designed. This approach leveraged Google Books Ngram 1-gram, isolating potential Out of Vocabulary (OOV) words based on their total frequency being higher than 9999, from the year 2000 onward. By doing so, it became possible to retain a set of words likely to contain obfuscated terms while reducing greatly the amount of noise in the data. To further reduce the noise in question, it was necessary to define categories for the different types of obfuscations.

Linked to the first and second research questions (RQ1 and RQ2), 5 categories were defined, "homoglyphs", "symbols", "numbers", "phonetic" and "other". The definition of these categories aided in the creation of ways to check for the presence of possibly obfuscated examples, when applicable, allowing to filter more words out thus enabling the possibility of manual review to annotate the data, as a last resort to increase its quality.

Prior to reaching the point of answering the third research question (RQ3), the data was prepared through the application of the necessary NLP processing and other techniques, which resulted in a final total of 501 posts from different sources ready for the following phases. Through its analysis it was possible to understand that manually collected examples (332) were predominant, further proving the lack of other resources. Regarding the third research question (RQ3), many approaches were tested for their viability to help recover the canonical form of obfuscated words through similarity scoring (cosine similarity) and rule-based modifications. In terms of rule-based approaches, to deal with common leatspeak substitutions after their detection with the previous established checks in the context of the previous research question, a predefined mapping was used, restoring the altered characters to their alphabetic equivalents while taking advantage of a context-aware mechanism to deal with ambiguous cases. A similar mapping process was applied to homoglyphs to normalize them, along with an additional approach

through an existing library. As for similarity scoring, one of the embedding-based approaches was FastText, which had trouble dealing with obfuscation techniques based on the usage of symbols and numbers, but yielded high similarity scores ($similarity \geq 0.75$) when comparing the canonical forms to an obfuscation of the phonetic type. In contrast to this, CharacterBERT and BPEmb embeddings presented more balanced but average results for the closest words ($0.50 \leq similarity \leq 0.60$). On the other hand, leveraging CANINE yielded the best results, with the correct canonical form frequently appearing as the top-ranked candidate, even when the similarity score was not consistently high. To complement the decision-making process made based on similarity, edit distance and phonetic similarity approaches were explored. Among Levenshtein, Damerau-Levenshtein and Jarrow-Winkler distances, the prefered was Damerau-Levenshtein given its consistency. Regarding phonetic similarity, the DoubleMetaphone algorithm was the one and only option, while having satisfactory results.

Addressing the fourth research question (RQ4), the results of the word cleaning pipeline, built from the most effective method of each type mentioned in the previous research question, were compared to the results obtained from the experimental phase, where LLMs were tested. Since both local deployments of LLaMA and DeepSeek were incapable of handling obfuscation normalization and generation, the chosen one for the tasks was ChatGPT-4o. Regarding the word cleaning pipeline applied to the original data, 431 occurences were deemed solved ($\approx 32.2\%$), 66 partially solved ($\approx 5\%$) and 831 unsolved ($\approx 62.1\%$). Despite these results, the performance of the process is acceptable given the circumstances. Under the same conditions, the pipeline was applied to the GPT-generated data, resulting in 631 occurences deemed solved ($\approx 55.5\%$), 60 partially solved ($\approx 5.2\%$) and 444 unsolved ($\approx 39.1\%$).

When these results are based on the flagged words rather than the total OOV words isolated for each dataset, the performance of the pipeline becomes clearer and later comparable to the GPT performance on obfuscation normalization. For the original dataset, 376 out of 761 ($\approx 49.4\%$) flagged words were correctly normalized, while 385 ($\approx 50.6\%$) were not. In comparison, for the GPT-generated dataset, 598 out of 922 ($\approx 64.9\%$) flagged words were correctly normalized, with the remaining 324 ($\approx 35.1\%$) left unprocessed. The differences in the pipeline's performance are highly influenced by the way the data was generated, since it had the SurgeAI offensive language lexicon as reference. This outcome indicates that improvements in the proccesses of isolating OOV words and expanding the coverage of the lexicons can improve overall performance.

Finally, the obfuscation normalization performed by ChatGPT-4o, compared against the clean version of the flagged words, resulted in 585 out of the 761 flagged words being cleaned 585 ($\approx 76.9\%$) and 176 missed ($\approx 23.1\%$).

Although the results seem promising, this work was hindered by several limitations. Among these limitations, the lack of high-quality datasets dedicated to character-level obfuscations on offensive language stands out. On another note, reliance on lexicons and

rule-based mappings introduces an inherent dependency on regular updates and maintenance in order to remain effective against the rapidly evolving landscape of obfuscation techniques. When it comes to the usage of commercial LLMs, tasks involving offensive language can be rejected by safety filters, which can limit their applicability in studies of offensive language obfuscation. With this in mind, future work should attempt to address the limitations discussed above. The creation of larger and more diverse datasets, specifically tailored to obfuscations of offensive language, would significantly enhance the robustness of future studies. To develop such datasets, the collaborative efforts with the platforms where these occurences are most prevalent would be ideal, for example by organizing data challenges that foster a competitive environment for high-quality contributions. Additionally, the study of emerging obfuscation patterns and new offensive slangs would ensure that lexicons remain up to date with evolving obfuscation strategies, maintaining the effectiveness of detection and normalization pipelines. Future work could also address LLM safety filter limitations by using open-source or locally hosted models, and conducting experiments in controlled research environments. These approaches would allow full evaluation of an LLM's obfuscation capabilities without being constrained by safety restrictions, at the cost of more computational resources, and enable the mitigation of obfuscated offensive language occurences.

Bibliography

Adinolf, S., & Turkay, S. (2018). Toxic Behaviors in Esports Games: Player Perceptions and Coping Strategies [event-place: Melbourne, VIC, Australia]. *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, 365–372. https://doi.org/10.1145/3270316.3271545

Aggarwal, S., Bajaj, A., & Vishwakarma, D. K. (2024). HOMOGRAPH: A novel textual adversarial attack architecture to unmask the susceptibility of linguistic acceptability classifiers. *International Journal of Information Security*, *24*(1), 6. https://doi.org/10.1007/s10207-024-00925-w

Ballew, B. S. (2009). Elsevier's Scopus® Database [Publisher: Routledge _eprint: https://doi.org/10.1080/15424060903167252]. *Journal of Electronic Resources in Medical Libraries*, *6*(3), 245–252. https://doi.org/10.1080/15424060903167252

Bespalov, D., Bhabesh, S., Xiang, Y., Zhou, L., & Qi, Y. (2023, July). Towards Building a Robust Toxicity Predictor. In S. Sitaram, B. Beigman Klebanov, & J. D. Williams (Eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)* (pp. 581–598). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-industry.56

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information [_eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00051/1567442/tacl_a_00051.pdf]. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Chan, J., & Li, Y. (2024). Unveiling disguised toxicity: A novel pre-processing module for enhanced content moderation. *MethodsX*, *12*, 102668. https://doi.org/https://doi.org/10.1016/j.mex.2024.102668

Chen, Y., Gao, H., Cui, G., Yuan, L., Kong, D., Wu, H., Shi, N., Yuan, B., Huang, L., Xue, H., Liu, Z., Sun, M., & Ji, H. (2023). From Adversarial Arms Race to Model-centric Evaluation Motivating a Unified Automatic Robustness Evaluation Framework [ISSN: 0736587X Type: Conference paper]. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 9607–9632. https://doi.org/10.18653/v1/2023.findings-acl.611

Chen, Y., Su, J., & Wei, W. (2021, November). Multi-granularity Textual Adversarial Attack with Behavior Cloning. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 4511–4526). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.371

Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). <scp>Canine</scp>: Pre-training an Efficient Tokenization-Free Encoder for Language Representation [Publisher: MIT Press]. *Transactions of the Association for Computational Linguistics*, *10*, 73–91. https://doi.org/10.1162/tacl_a_00448

Codebox. (2020). Homoglyph characters. https://github.com/codebox/homoglyph

Colombo, P., Picot, M., Noiry, N., Staerman, G., & Piantanida, P. (2023, December). Toward Stronger Textual Attack Detectors. In H. Bouamor, J. Pino, & K. Bali (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023* (pp. 484–505). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.35

Cui, C., Sun, T., Lin, M., Gao, T., Zhang, Y., Liu, J., Wang, X., Zhang, Z., Zhou, C., Liu, H., Zhang, Y., Lv, W., Huang, K., Zhang, Y., Zhang, J., Zhang, J., Liu, Y., Yu, D., & Ma, Y. (2025). PaddleOCR 3.0 Technical Report [_eprint: 2507.05595]. https://arxiv.org/abs/2507.05595

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors [Place: New York, NY, USA Publisher: Association for Computing Machinery]. *Commun. ACM*, *7*(3), 171–176. https://doi.org/10.1145/363958.363994

Davidson, T. (2017). Automated Hate Speech Detection and the Problem of Offensive Language [Dataset]. https://github.com/t-davidson/hate-speech-and-offensive-language

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., … Zhang, Z. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning [_eprint: 2501.12948]. https://arxiv.org/abs/2501.12948

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *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)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Eger, S., & Benz, Y. (2020, December). From Hero to Zéroe: A Benchmark of Low-Level Adversarial Attacks. In K.-F. Wong, K. Knight, & H. Wu (Eds.), *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing* (pp. 786–803). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.aacl-main.79

Eger, S., Şahin, G. G., Rücklé, A., Lee, J.-U., Schulz, C., Mesgar, M., Swarnkar, K., Simpson, E., & Gurevych, I. (2019, June). Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In J. Burstein, C. Doran, & T. Solorio (Eds.), *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)* (pp. 1634–1647). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1165

El Boukkouri, H., Ferret, O., Lavergne, T., Noji, H., Zweigenbaum, P., & Tsujii, J. (2020, December). CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters. In D. Scott, N. Bel, & C. Zong (Eds.), *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 6903–6915). International Committee on Computational Linguistics. https://doi.org/10.18653/v1/2020.coling-main.609

Fernandes, E., Moro, S., & Cortez, P. (2023). Data Science, Machine learning and big data in Digital Journalism: A survey of state-of-the-art, challenges and opportunities. *Expert Systems with Applications*, *221*, 119795. https://doi.org/https://doi.org/10.1016/j.eswa.2023.119795

Formento, B., Foo, C. S., Tuan, L. A., & Ng, S. K. (2023). Using Punctuation as an Adversarial Attack on Deep Learning Based NLP Systems: An Empirical Study [Type: Conference paper]. *EACL 2023 - 17th Conference of the European Chapter of the Association for Computational Linguistics, Findings of EACL 2023*, 1–34. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85159858760&partnerID=40&md5=3967cdd3cb075548c29fa21ab8ed98f6

Gao, J., Lanchantin, J., Soffa, M. L., & Qi, Y. (2018). Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. *2018 IEEE Security and Privacy Workshops (SPW)*, 50–56. https://doi.org/10.1109/SPW.2018.00016

Ge, Z., Hu, H., & Zhao, T. (2023, September). Towards Trustworthy NLP: An Adversarial Robustness Enhancement Based on Perplexity Difference. https://doi.org/10.3233/FAIA230347

Gil, Y., Chai, Y., Gorodissky, O., & Berant, J. (2019). White-to-black: Efficient distillation of black-box adversarial attacks [Type: Conference paper]. *NAACL HLT 2019 - 2019*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, *1*, 1373–1379. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083331960&partnerID=40&md5=957bf1df76478a5a2233576df167bb33

Heinzerling, B., & Strube, M. (2018, May). BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, & T. Tokunaga (Eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA). https://aclanthology.org/L18-1473/

Hofer, N., Schöttle, P., Rietzler, A., & Stabinger, S. (2021). Adversarial Examples Against a BERT ABSA Model – Fooling Bert With L33T, Misspellign, and Punctuation, [event-place: Vienna, Austria]. *Proceedings of the 16th International Conference on Availability, Reliability and Security*. https://doi.org/10.1145/3465481.3465770

Hu, X., Xu, C., Ma, J., Huang, Z., Yang, J., Guo, Y., & Barthelemy, J. (2023). [MASK] Insertion for anti-adversarial attacks [Type: Conference paper]. *EACL 2023 - 17th Conference of the European Chapter of the Association for Computational Linguistics, Findings of EACL 2023*, 1028–1040. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85159861349&partnerID=40&md5=edfd59bfa0141320f57bff30f28637f9

Huertas-García, Á., Martín, A., Huertas-Tato, J., & Camacho, D. (2023). Countering malicious content moderation evasion in online social networks: Simulation and detection of word camouflage. *Applied Soft Computing*, *145*, 110552. https://doi.org/https://doi.org/10.1016/j.asoc.2023.110552

Jaro, M. A. (1989). Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida [Publisher: ASA Website _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1989.10478785]. *Journal of the American Statistical Association*, *84*(406), 414–420. https://doi.org/10.1080/01621459.1989.10478785

Jebran Khan, K.-A. S., Kashif Ahmad. (2023). An Efficient Character-Level Adversarial Attack Inspired by Textual Variations in Online Social Media Platforms. *Computer Systems Science and Engineering*, *47*(3), 2869–2894. https://doi.org/10.32604/csse.2023.040159

Jing, Q. (2020). A Benchmark Dataset for Learning to Intervene in Online Hate Speech - Gab & Reddit [Dataset]. https://github.com/jing-qian/A-Benchmark-Dataset-for-Learning-to-Intervene-in-Online-Hate-Speech

Keller, Y., Mackensen, J., & Eger, S. (2021, August). BERT-Defense: A Probabilistic Model Based on BERT to Combat Cognitively Inspired Orthographic Adversarial Attacks. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021* (pp. 1616–1629). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.141

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. *2*.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, *10*(8), 707–710. https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf

Li, J., Ji, S., Du, T., Li, B., & Wang, T. (2018). TextBugger: Generating Adversarial Text Against Real-world Applications [arXiv: 1812.05271]. *CoRR*, *abs/1812.05271*. http://arxiv.org/abs/1812.05271

Lin, Y., Michel, J.-B., Aiden, E. L., Orwant, J., Brockman, W., & Petrov, S. (2012). Syntactic annotations for the Google Books Ngram Corpus [event-place: Jeju Island, Korea]. *Proceedings of the ACL 2012 System Demonstrations*, 169–174.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach [arXiv: 1907.11692]. *CoRR*, *abs/1907.11692*. http://arxiv.org/abs/1907.11692

Mary, K. S. E., Sudhan, S., & Narayanan, A. G. H. (2022). Adversarial Attacks against Machine Learning Classifiers: A Study of Sentiment Classification in Twitter. *2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP)*, 1–6. https://doi.org/10.1109/ICICCSP53532.2022.9862472

Morris, J., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., & Qi, Y. (2020, October). TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In Q. Liu & D. Schlangen (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 119–126). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.16

Mou, G., Ye, P., & Lee, K. (2020). SWE2: SubWord Enriched and Significant Word Emphasized Framework for Hate Speech Detection [event-place: Virtual Event, Ireland]. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 1145–1154. https://doi.org/10.1145/3340531.3411990

Omar, M., & Mohaisen, D. (2022). Making Adversarially-Trained Language Models Forget with Model Retraining: A Case Study on Hate Speech Detection [event-place: Virtual

Event, Lyon, France]. *Companion Proceedings of the Web Conference 2022*, 887–893. https://doi.org/10.1145/3487553.3524667

OpenAI, : Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A. J., Welihinda, A., Hayes, A., Radford, A., Mądry, A., Baker-Whitcomb, A., Beutel, A., Borzunov, A., Carney, A., Chow, A., Kirillov, A., ... Malkov, Y. (2024). GPT-4o System Card [_eprint: 2410.21276]. https://arxiv.org/abs/2410.21276

Oubiwan. (n.d.). Metaphone. https://github.com/oubiwann/metaphone

Philips, L. (2000). The Double Metaphone Search Algorithm. *C/C++ Users Journal*, *18*, 38–43.

Pruthi, D., Dhingra, B., & Lipton, Z. C. (2019, July). Combating Adversarial Misspellings with Robust Word Recognition. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 5582–5591). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1561

Rusert, J., Shafiq, Z., & Srinivasan, P. (2022). On the Robustness of Offensive Language Classifiers [ISSN: 0736587X Type: Conference paper]. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, *1*, 7424–7438. https://doi.org/10.18653/v1/2022.acl-long.513

Seth, D., Stureborg, R., Pruthi, D., & Dhingra, B. (2023). Learning the Legibility of Visual Text Perturbations [Type: Conference paper]. *EACL 2023 - 17th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, 3252–3265. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85159852363&partnerID=40&md5=f4314f3f44a6f4b4b8723523a73bce6f

Shen, L., Zhang, Z., Jiang, H., & Chen, Y. (2023). TEXTSHIELD: BEYOND SUCCESSFULLY DETECTING ADVERSARIAL SENTENCES IN TEXT CLASSIFICATION [Type: Conference paper]. *11th International Conference on Learning Representations, ICLR 2023*. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85193813678&partnerID=40&md5=bebf99e51131fc344eef05887ec71eac

s-nlp. (2021). RoBERTa Toxicity Classifier. https://huggingface.co/snlp/roberta-toxic-classifier

SurgeAI. (2021a). Profanity [Source code]. https://github.com/surge-ai/profanity

SurgeAI. (2021b). SurgeAI Toxicity EN [Dataset]. https://github.com/surge-ai/toxicity

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models [_eprint: 2302.13971]. https://arxiv.org/abs/2302.13971

Vidgen, B., & Derczynski, L. (2021). Directions in abusive language training data, a systematic review: Garbage in, garbage out [Publisher: Public Library of Science]. *PLOS ONE*, *15*(12), 1–32. https://doi.org/10.1371/journal.pone.0243300

Winkler, W. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods*.

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019, June). SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In J. May, E. Shutova, A. Herbelot, X. Zhu, M. Apidianaki, & S. M. Mohammad (Eds.), *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 75–86). Association for Computational Linguistics. https://doi.org/10.18653/v1/S19-2010

Zeng, G., Qi, F., Zhou, Q., Zhang, T., Ma, Z., Hou, B., Zang, Y., Liu, Z., & Sun, M. (2021). OpenAttack: An open-source textual adversarial attack toolkit [Type: Conference paper]. *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the System Demonstrations*, 363–371. https://doi.org/10.18653/v1/2021.acl-demo.43

Zhang, K., & Kizilcec, R. (2014). Anonymity in Social Media: Effects of Content Controversiality and Social Endorsement on Sharing Behavior [Section: Poster Papers]. *Proceedings of the International AAAI Conference on Web and Social Media*, *8*(1), 643–646. https://doi.org/10.1609/icwsm.v8i1.14573

Zhang, X., Zhang, Z., Zhong, Q., Zheng, X., Zhang, Y., Hu, S., & Zhang, L. Y. (2023). Masked Language Model Based Textual Adversarial Example Detection [event-place: Melbourne, VIC, Australia]. *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 925–937. https://doi.org/10.1145/3579856.3590339

Zhou, Y., Jiang, J.-Y., Chang, K.-W., & Wang, W. (2019). Learning to discriminate perturbations for blocking adversarial attacks in text classification [Type: Conference

paper]. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 4904–4913. https://doi.org/10.18653/v1/D19-1496

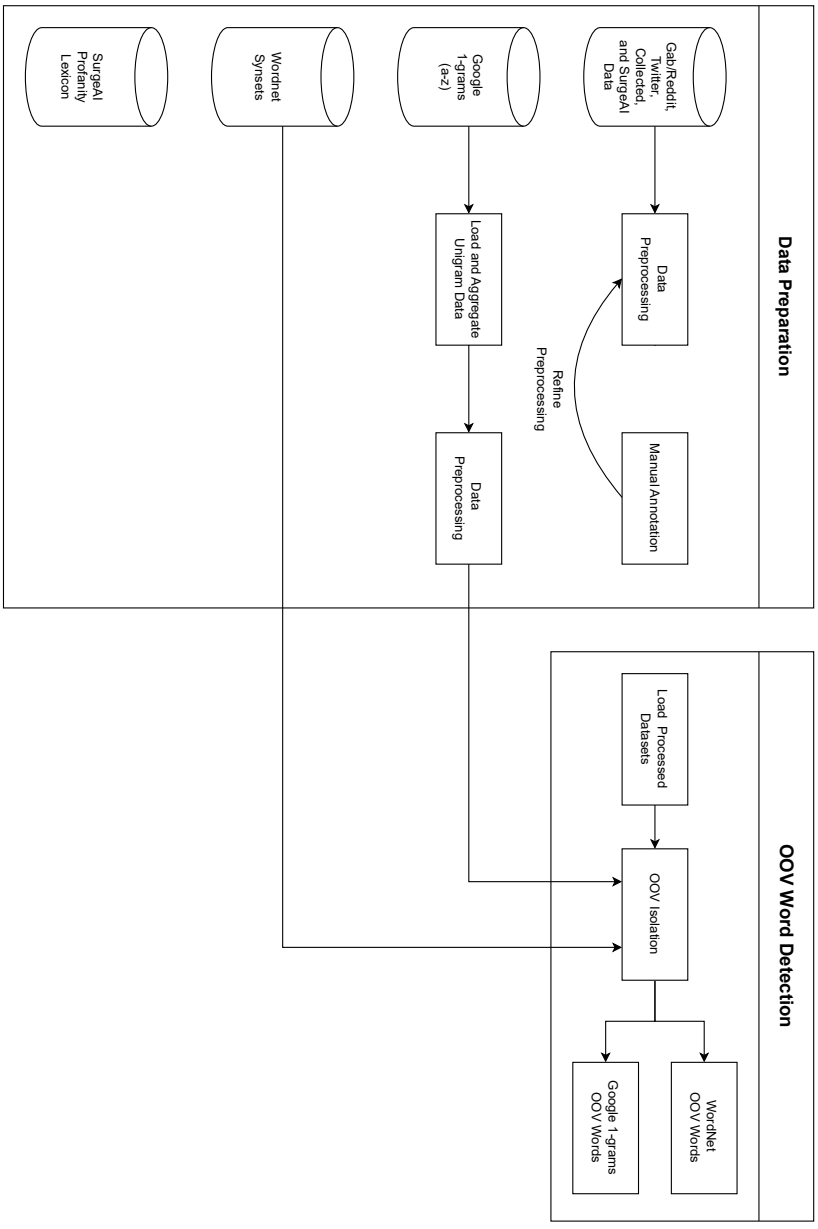
Appendix A

Examples of obfuscated offensive language

Examples
”ki!! y0urself.”
”fiction u are a ch1ng ch0ng KNEEGER adc player (...) ”
”Albanians are fūķīŋğ sūƀhūmāŋṣ (...) ”

Appendix B

High-level perspective of the Data Preparation Phase



Appendix C

Table of leet-speak substitutes for normal letters from Wikipedia

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
4	I3	[	)	3	|=	6	#	1	,_|	>|	1	/\/\	^/	0	|*	(_,)	I2	5	7	(_)	\/	\/\/	><	j	2
/\	8	¢	|)	&	ƒ	&	/-/	|	_|	|<	7	/V\	||	()	|o	()_	9	$	+	|_|	|/	vv	}{	`/	7_
@	13	<	(|	£	|#	(_+	\-\	][	._|	1<	2	[V]	/\/	oh	|º	2	|`	z	-|-	v	|	\N	ecks	|/	-/_
/-	|3	(	[)	€	ph	9	[-]	!	._]	|c	£	|\/|	[\]	[]	?	0_	|~	§	']	L|		'//	×	¥	%
\	ß	©	I>	[-	/=	C-	]-[	eye	_]	|	|_	^^	<\>	p	|^	<|	|?	ehs	['	ɹ		\\'	?	\// `|	>_
^	!3		|>	|=-	v	gee	)-(	3y3	,_]	(7<	|	<\/>	{\}	<>	|>	&	/2	es	†	ʁ		\^/	}{	´	s
(L	(3		?			(?,	(-)		]			{v}	/V	ø	|"	9	|^	2	«|»			\/\/	)(		~/_
Д	/3		T)			[,	:-:					(v)	^		9	¶	1z		~|~			(n)	][		-_
	)3		I7			{,	|~|					(V)	ท		[]D	⁋	7					\V/			-
	|-]		cl			<-	|-|					||\	и		|°	®	2					\X/			|_
	j3		|}			(.	]~[					]\/[					12						|/			
			|]				}{					nn					®						_|_/			
							!-!					11					[z						_:_/			
							1-1										Я						uu			
							\-/										.-						2u			
							I+I										|2						\\//\\//			
							?										|-						ѡ			
																	3						₩			
																	4						ω			