2023

Leveraging keyword identification for abusive language detection in social online domains

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Thesis

LEVERAGING KEYWORD IDENTIFICATION FOR ABUSIVE LANGUAGE DETECTION IN SOCIAL ONLINE DOMAINS

by

DURUVAN SARAVANAN

B.A., Boston University, 2019

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2023

Approved by

First Reader

Bryan Plummer, PhD
Assistant Professor of Computer Science

Second Reader

Derry Wijaya, PhD
Assistant Professor of Computer Science

Acknowledgments

I would like to thank my advisor Professor Bryan Plummer for his continued support and for aiding me in my journey to complete this thesis. His advice and guidance have been invaluable. I would also like to thank Professor Derry Wijaya for her time and support in being a reader for my defense.

To my friends - thank you for always being there. As a student thousands of miles away from home and family, your support has kept me going, and you have my endless gratitude.

I would not be where I am today without the endless love and support of my parents, who were always willing to listen to me talk about my research and encourage me through the toughest of challenges. This would not have been possible without you.

Duruvan Saravanan

Computer Science Department

LEVERAGING KEYWORD IDENTIFICATION FOR ABUSIVE LANGUAGE DETECTION IN SOCIAL ONLINE DOMAINS

DURUVAN SARAVANAN

ABSTRACT

Abusive Language Detection is an important NLP task that can lead to improved content moderation and is particularly crucial for a safer online experience for all. Most online spaces, such as social media or video games, employ some form of content moderation and allow users to report instances of abusive language but fall short when it comes to real-time detection and intervention.

Our work focuses on abusive language detection in social online domains. We begin by exploring the use of pre-trained transformer models for this task and provide a set of baseline results. We introduce an offline filtering step that identifies and removes non-keywords using pointwise mutual information to address linguistic challenges such as domain-specific language and misspelled words. This process emphasizes the remaining words and builds a stronger association between word-category pairs. We also explore an online filtering framework that aims to replicate and remove the offline filtering step by utilizing transformer-produced context-aware embeddings to score words. Identifying and removing non-keywords can significantly improve performance for this task but relies on knowledge of word-category co-occurrences.

Contents

List of Tables

List of Figures

List of Abbreviations

CONDA	....	Contextual Dual-Annotated
NLP	....	Natural Language Processing
OOV	....	Out-of-vocabulary
PMI	....	Pointwise Mutual Information

Chapter 1

Introduction

Abusive language detection is an essential task in NLP that aids in advancing online content moderation and helps create safe and accessible online spaces. It is in the interest of most developers of these online spaces to have systems in place that can detect and intervene when abusive language is used. Ethically, exposure to abusive language can lead to mental health issues like anxiety or depression. Also, existing users are more likely to stop engaging and leave a space if exposed to repeated abusive language. New users are less likely to join if an online space or platform gains a reputation for harboring toxicity. While the end goal is easy to visualize, many nuances to the problem can make it complex and challenging to solve. In this thesis, we focus on social online domains that present linguistic challenges that make abusive language detection complex (Weld et al., 2021; Davidson et al., 2017; Kurrek et al., 2020). Some of the challenges include domain-specific language, grammatical or spelling errors, short sentences, and the use of language devices such as irony and sarcasm (Vidgen et al., 2019).

The consequences of exposure to abusive language can be severe. For example, online games provide text and voice chat capabilities to foster team cooperation but also allow for abusive language to be delivered. The competitive and high-stakes nature of these games can lead to some players engaging in heated confrontations with slurs, derogatory language, and personal insults. A study conducted by the Anti-Defamation League states that 83% of adult gamers experienced some form of harassment while playing online multiplayer games (adl, 2021). The same study shows that the consequences of toxicity in these games are severe for players and developers, with players having reported feeling more depressed and isolated and having their personal relationships disrupted due to the toxicity they experienced while playing. Many players also reported quitting certain games, which directly impacts developers. Being able to detect toxicity is an important consideration for developers, not only to prevent players from quitting but to maintain a safe and healthy environment where players ultimately are only trying to have fun. Naturally, the same concerns apply to other online spaces like mainstream social media, where users consistently interact with each other. Being able to detect toxicity or abusive language use and intervene promptly is important in order to create a safer Internet for everyone.

There is a strong need for detecting abusive language automatically and in real time. For example, most online games provide a reporting feature where players can report other players for various reasons, including toxic communication. However, these reports are verified after the incident, which does not prevent escalation of the situation. Social media platforms also allow reporting of users that engage in abusive language use. There can also be a bystander effect when it comes to reporting, which results in this feature not being utilized even when the situation warrants it. Research has explored this bystander effect in video games, showing that players do not actively report toxic behavior and that factors like team performance can influence if a report is made (Kwak et al., 2015a). Ultimately, even if a report is made, most video games rely on some level of manual verification of the incident, which takes time, and any punishment handed out is well after the fact which does not prevent the offending player from making abusive comments within the same game. Live and automated detection of toxicity is an active area of research and development, with video game developers releasing various features and systems designed to detect and

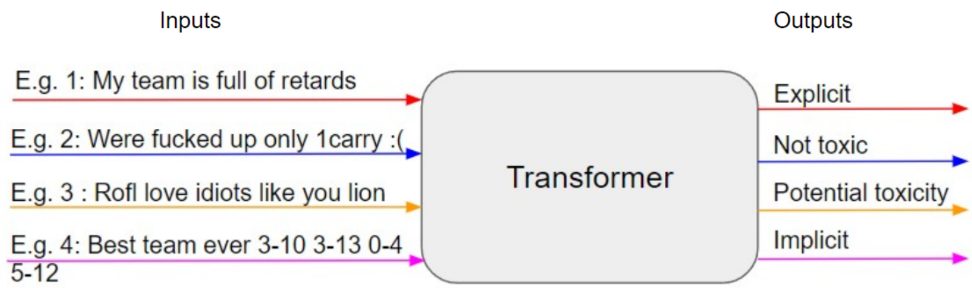


Figure 1·1: Examples of communication between players from the CONDA (Weld et al., 2021) dataset and the ideal outcome for a toxicity detection system in a game. Any such system should be able to determine explicit toxic occurrences as well as determine if communication has toxic intentions behind it using context.

intervene against toxicity. League of Legends developer Riot Games released a feature to detect zero-tolerance language live and in-game (lea, 2022). When detected, the system intervenes and mutes the offending players to prevent further toxic communication. Activision Blizzard introduced audio transcriptions to collect voice chat recordings and convert them into text files to analyze for disruptive communication (Entertainment, 2022). This is also an area of interest in wider social media. For example, Reddit makes a content moderation tool called AutoModerator available to moderators, which is aimed at automating some of the moderation burden and can be customized based on the needs of a subreddit.

Figure 1·1 highlights various examples of communication in the CONDA dataset (Weld et al., 2021). The first sentence is a clear example of toxic communication that should be straightforward for any system to detect, as it contains clear zero-tolerance language and is aimed at insulting other players. The second sentence is an example of communication that could mistakenly be flagged as toxic due to the presence of explicit language. The last two sentences are examples of communication that are difficult for any system to handle, given the sarcastic nature and the need for context and game-specific knowledge to understand the intention behind the communication. These examples highlight areas of focus for toxicity detection systems - detect explicit forms of toxicity with high accuracy and improve detection of implicit toxicity. In this thesis, we explore the abusive language detection task for three different online domains that contain a relatively high volume of domain-specific language - the DOTA 2 video game (CONDA dataset) (Weld et al., 2021), Twitter (Davidson et al., 2017) and Reddit (Kurrek et al., 2020). Since few toxicity detection research papers in the video game domain have explored the use of transformers, focusing instead on exploring n-gram models (Kwak et al., 2015b; Märtens et al., 2015) and Random Forest models (Canossa et al., 2021) or creating annotated datasets (Weld et al., 2021; Märtens et al., 2015), we first explore the use of transformers for abusive language detection in video games using the CONDA dataset. The next step involves addressing the use of domain-specific language in our datasets. These words tend to be OOV for pre-trained language models like BERT, which can be problematic for the final classification. Since these words contain essential domain knowledge and context, we explore the use of offline and online filtering steps that identify and remove non-keywords, emphasizing the contribution of the remaining domain-specific words towards classification.

1.1 Challenges

Abusive Language Detection is a popular area of work in NLP and has received much attention from researchers, but this task presents various challenges. This thesis highlights some commonly recognized linguistic challenges for this task and attempts to address them (Vidgen et al., 2019). The primary focus is dealing with datasets created from social online settings that tend to contain many of these linguistic issues. Chat data from video games typically involve a lot of these linguistic challenges, so we mainly work with the CONDA dataset (Weld et al., 2021) but also introduce and experiment with data gathered from mainstream social media (Davidson et al., 2017; Kurrek et al., 2020).

The video game domain has a particular challenge in that there is a lack of publicly available large-scale annotated datasets created for abusive language detection. Most developers tend to keep chat data from their games private, and while some games like DOTA 2 make chat data available through APIs, it still requires a considerable effort to annotate the data to make it ready for abusive language detection research.

Video games are an interesting space to work with due to the variety of linguistic challenges that can impede abusive language detection. The live and real-time nature of online multiplayer games makes it so that players tend to type out shorter, often incomplete sentences. Naturally, if a player spends much time thinking about what to say, their chances of winning the game decrease. The other consequence is that text data from video games tend to contain many grammatical and spelling errors. The other language-specific challenge is, of course, the use of domain-specific words. Things like character names, ability names, locations, and jargon tend to be words that are typically out-of-vocabulary for pre-trained models. As mentioned earlier, we also work with datasets that use social media as the source. Social media also presents similar challenges, such as emojis, slang, abbreviations, and other forms of jargon, and datasets pulled from sources like Twitter can contain grammatical and spelling errors due to the imposed character limits.

There are, of course, other linguistic challenges, such as the use of implicit forms of abuse. Implicit forms of abuse are shown to be harder to detect, and users of online spaces can be acutely aware that the use of explicit language like swear words and slurs can result in punishment like being banned from the platform very quickly by virtue of being much easier to detect and verify. To avoid this, users can use strategies like intentional misspellings or language devices like sarcasm, back-handed humor, and irony (Vidgen et al., 2019).

1.2 Research Outline

In this thesis, we separate the problem into three parts - exploring the use of transformers for abusive language detection in varied online domains, using an offline word filtering method using pointwise mutual information that identifies and removes non-keywords in order to emphasize important words in a dataset and exploring additional offline and online word filtering methods that aim to learn word scores as opposed to calculating them.

Chapter 2 contains the related work section. We highlight previous research done in the abusive language detection field. We look at past work in the video game domain and the broader social media domain. We introduce previous work done in creating datasets to further research in this field as well as work that explores new techniques to help solve abusive language detection. We also highlight research regarding keyword and stop-word identification for the purpose of text classification.

In Chapter 3, we introduce the different toxicity detection datasets from different sources - DOTA 2 (Weld et al., 2021), Twitter (Davidson et al., 2017), and Reddit (Kurrek et al., 2020). We explore the CONDA dataset and highlight potential areas of concern, such as the semantic similarity between different categories, the relatively short lengths of the examples, the frequent use of domain-specific language, and spelling errors. We also establish a set of baseline results using different transformers to perform classification on these datasets. For the CONDA dataset, we also investigate whether the use of both slot and intent level annotations improves classification performance and compare the throughput and inference time of the transformers.

Chapter 4 introduces an offline filtering step that focuses on removing non-keywords and emphasizing the essential words in a dataset. We use PMI to determine which words are strongly associated with a given category in a dataset and which are irrelevant. We start by calculating PMI scores for each word in every example in the dataset, given its classification. We use these scores to remove words that fall below a certain threshold and compare the performance of BERT (Devlin et al., 2018) on the trimmed datasets to the baseline provided in Chapter 3. We highlight how choosing the right words to remove can positively impact performance for abusive language detection.

In Chapter 5, we explore additional methods to identify non-keywords that involve learning the scores of words as opposed to calculating them using PMI. The first set of experiments explores the use of alternative types of offline filtering methods. We explore the use of FastText (Bojanowski et al., 2017) embeddings passed through a dense network with the PMI scores as the target to produce per-category scores for every word in the dataset. We also explore purely unsupervised techniques of scoring words, such as PCA. The second methodology is an online filtering framework to score words with the help of BERT that is meant to overcome the reliance previously discussed methods have on prior knowledge of word statistics. We devise a two-pass framework where a dense network produces scores after a first pass through BERT, and low-scoring tokens are removed before a second and final pass is used to make the final prediction. Chapter 6 consists of the summarization of our work and highlights future directions for this research.

Chapter 2

Related Work

2.1 Abusive Language Detection in video games

Previous toxicity detection research has been conducted on different games from different genres. The most popular genre for this type of research has been the MOBA (multiplayer online battle arena) genre due to the chat feature being integral to gameplay. Early studies in toxicity detection focused on defining and developing methodologies to annotate chat data while also investigating if game-specific events can lead to toxicity in *Dota 2* (Märtens et al., 2015). The researchers investigated if toxic remarks by players are preceded by a specific game event within a time window and discovered that toxic chats tend to occur shortly after a player has been defeated by the enemy team and that the winning team is more likely to have positive chat interactions. Similarly, research has been conducted in League of Legends to determine if game-related features can be used in addition to chat logs to detect toxicity (Blackburn and Kwak, 2014). The researchers trained separate classifiers using features from three sources - player performance, reports, and chat, while also training a classifier using combined features. They calculated the importance of these features and discovered that most of the top-ranked features came from the game chat. Researchers have also attempted to detect toxicity while conversations are developing in real time (Stoop et al., 2019). This paper details a framework where toxicity estimates for participants in a game are generated and updated separately as they make additional utterances. The classifier in this framework makes use of RNN and bidirectional GRU layers. More recent work has focused on creating a more robust annotated dataset for *Dota 2*, using a dual level (token and utterance) annotation while also providing baseline results using five different Natural Language Understanding models (Weld et al., 2021). The dual-level annotation scheme involves categorizing each utterance in-game chat into one of four categories while also categorizing each token in the utterance into one of six categories. This scheme can be adapted to suit the specific language of a game.

Non-MOBA games such as *For Honor* (Canossa et al., 2021) and *World of Tanks* (Murnion et al., 2019) have also been investigated. Murnion et al. developed a data collection system to scrape and decode data from a website, transform it into JSON files, and persist it in a database. They automatically classified this data using SQL commands and found that it performed well when the results were compared to manual classification. They also used publicly available sentiment analysis tools like Microsoft Azure to identify abusive chats and found that these tools performed poorly. The data collection system and automatic classification detailed in this paper can be adapted for other games to create a robust data pipeline to support toxicity detection models.

2.2 Abusive Language Detection in social media

Researchers have worked on toxicity detection in various social media platforms outside the video game domain. A significant area of focus has been in defining annotation guidelines and creating datasets to aid this area of research. Researchers have worked on creating large-scale datasets from Twitter to aid work in detecting hate speech and use of abusive language (Waseem and Hovy, 2016; Founta et al., 2018; Davidson et al., 2017). Datasets have also been compiled from other online sources such as Reddit (Kurrek et al., 2020) and Wikipedia (Wulczyn et al., 2017). Beyond detecting whether the language used was abusive, research has also been conducted to detect this abuse's target. As such, datasets have been curated to aid in the detection of hate speech or abuse towards specific groups such as immigrants and women (Basile et al., 2019), East Asians (Vidgen et al., 2020a) or Muslims (Chung et al., 2019). Researchers have also explored synthetically generating examples to mimic real-world social media posts by employing a human-and-model-in-the-loop framework (Vidgen et al., 2020b). A significant cataloging effort has also been conducted where researchers have reviewed publicly available datasets and explored the social and ethical concerns of this type of work (Vidgen and Derczynski, 2021).

Researchers have worked on improving abusive language detection in various ways, such as augmenting training with implicit abuse examples (Nejadgholi et al., 2022) using concept activation vectors. They provide a metric (degree of explicitness) that can be used to score examples from abuse detection datasets. This work can be transferred to the video game domain where implicit examples from a dataset built from a game's chat log (for example, from the work of Henry et al. (Weld et al., 2021)) can be picked out to improve the performance of an abusive language detection classifier. Success has also been found in defining a multi-task learning paradigm with abuse and emotion detection (Rajamanickam et al., 2020). The paper proposes a system where embedding representations are passed through an encoder, the results of which are passed to separate BiLSTMs, one for each task. The results support the hypothesis that the auxiliary task (emotion detection) aids in the performance of the primary task (abuse detection), as the multi-task learning frameworks perform better when compared to models that were only trained on abuse detection datasets. Toxicity in video games can be emotionally charged, and it is possible that the use of a game-specific emotion dataset can aid in toxicity. Additionally, the work of (Nobata et al., 2016) highlights some potential issues that can arise in abusive language detection tasks in live service video games like League of Legends. They work with temporal data to show that accumulating more recent data is preferable to having a sizeable initial dataset. The use of transformers has also been explored by researchers who re-trained a BERT model to produce a variation called HateBERT which outperforms a generic BERT on multiple hate detection datasets (Caselli et al., 2021). Other approaches include using syntactic dependency graphs (Narang and Brew, 2020) where the authors produce performance on par with BERT and show their approach to be more scalable.

2.3 Pre-processing and keyword extraction

Researchers have explored the effects of various pre-processing steps for NLP tasks, such as text categorization and sentiment analysis, on the final accuracy result. Simple measures such as tokenization appear to work as well as more complex methods except in domain-specific cases (Camacho-Collados and Pilehvar, 2018). Authors have similarly shown the benefits of pre-processing steps like negation processing, part-of-speech filtering, stop word removal, and lemmatization for affective tasks such as sarcasm detection, emotion classification, and sentiment analysis (Babanejad et al., 2020). Chapters 3 and 4 explore the use of pointwise mutual information (Church and Hanks, 1990) to identify words to remove to improve the separability of categories. Researches have previously explored the use of PMI for feature selection in text categorization and came up with variations of it such as weighted average pointwise mutual information (Schneider, 2005) and discriminative mutual information (Hussain et al., 2020). Similarly, authors have also evaluated various unsupervised methods such as TF-IDF, Part of Speech filtering, and sentence salience score to extract keywords from a meeting corpus (Liu et al., 2009). The work of (Koloski et al., 2022) also showcases the use of TF-IDF (Salton and McGill, 1986) combined with other supervised methods such as TNT-KID and BERT BiLSTM-CRF in order to identify keywords.

Chapter 3

Transformer for Abusive Language Detection in video games

In this chapter, we introduce our datasets, focusing primarily on the DOTA 2 dataset (Weld et al., 2021) which contains high usage of domain specific languages and other linguistic challenges like high volume of grammatical and spelling errors. We also introduce datasets from mainstream social media such as Reddit (Kurrek et al., 2020) and Twitter (Davidson et al., 2017) which present some of the same linguistic challenges. We conduct an exploration of the datasets and compare statistics that highlight these linguistic concerns. We use a variety of transformer models to perform classification on these datasets in order to establish baselines using transformers. Given a dataset containing n examples in a dataset $D = \{e_1,\ e_2,\ \ldots,\ e_n\}$ where each example e belongs to single category and given m possible categories $C = \{c_1,\ldots,c_m\}$, we utilize transformers to predict a category c for each example e and compare to the list of true categories. For the DOTA 2 dataset specifically, we also investigate other factors such as inference time, throughput and whether perform both slot and intent level classification is an improvement over just intent level classification.

3.1 Dataset Exploration

We evaluate the performance of various transformers on three different datasets gathered from the video game DOTA 2 (Weld et al., 2021) and social media platform Twitter (Davidson et al., 2017) and Reddit (Kurrek et al., 2020). Table 3.1 details

Figure 3·1: A basic overview of how classification are made using transformers. Each example from a dataset converted to tokens using a tokenizer which are then passed to a transformer. Each transformer contains a classification head which takes the final output embeddings from the transformer and makes the prediction.

the categories, number of examples, and the average length of examples per category for these datasets. The DOTA 2 dataset consists of four categories - The utterances are annotated at the intent and slot levels. The intent level consists of 4 categories - E(xplicit), I(mplicit), A(ction), and O(ther). The E and I categories make up the toxic utterances in the dataset. The A category consists of utterances that are not toxic in nature and contain language pertaining to game-related actions, such as pausing the game or reporting another player. The O category consists of utterances that do not belong in the other three categories. The Twitter dataset consists of three categories - H(ate) speech, 0(ffensive), and N(either), with the H and O categories making the categories that contain abusive language. The Reddit dataset consists of examples that all contain the use of slurs, and the categories detail the intent behind the slur. The categories are Derogatory (DEG), Non-Derogatory (NDG), Homonym (HOM), and Appriopriative (APR). We observe that the DOTA 2 dataset contains the shortest examples on average. For this dataset in particular, we must be careful when it comes to identifying and removing non-keywords, as removing too many words can have a destructive effect on the classification performance. The real-time nature of online video games makes it so that players do not have much time to type out complete sentences. Examples of the Twitter dataset are longer, averaging approximately 14 words per example, while the Reddit examples are more varied in

Dataset	Category	Number of examples in dataset	Average example length
(a) (Weld et al., 2021)	Action	2299	3.891
	Explicit	4711	5.342
	Implicit	2274	3.648
	Other	26603	2.924
(b) (Davidson et al., 2017)	Hate Speech	1430	13.907
	Offensive	19190	13.959
	Neither	4163	14.919
(c) (Kurrek et al., 2020)	Derogatory	20530	26.644
	Non-derogatory	16727	41.480
	Homonym	1998	49.529
	Appropriative	553	36.669

Table 3.1: Number of examples and average example length for each category in the DOTA 2 (Weld et al., 2021), Twitter (Davidson et al., 2017) and Reddit (Kurrek et al., 2020) datasets.

average length but are the longest of the three datasets. This is due to Twitter enforcing a much shorter character limit than Reddit. The general task for all three datasets is a multi-class classification task.

Table 3.1 also highlights the category imbalance that is present in all three datasets. In the DOTA 2 dataset (Weld et al., 2021), the O category is most prevalent in the dataset by a large margin. In the Twitter dataset (Davidson et al., 2017), the Offensive category makes up for approximately 77% of the dataset, while in the Reddit dataset (Kurrek et al., 2020), the Derogatory and Non-derogatory examples make up approximately 93% of the dataset. The imbalance already gives us a sense of which categories are going to be the most challenging to correctly classify for each dataset.

Table 3.2 shows similarity matrices that were calculated using two paradigms -

	Category	Action	Explicit	Implicit	Other
(a)	Action	1	0.71	0.69	0.66
	Explicit	0.71	1	0.73	0.79
	Implicit	0.69	0.73	1	0.77
	Other	0.66	0.69	0.71	1
(b)	Action	0	11.78	11.92	12.43
	Explicit	11.78	0	11.62	10.24
	Implicit	11.92	11.62	0	10.40
	Other	12.43	11.92	11.78	0

Table 3.2: (a) shows the cosine similarity between the embeddings for the different categories and (b) shows the euclidean distance in the DOTA 2 dataset (Weld et al., 2021).

cosine similarity and Euclidean distance, which we use to compare the categories in the DOTA 2 dataset (Weld et al., 2021). These matrices are calculated by taking the word embeddings for each word in an utterance and averaging them to produce a sentence embedding. These embeddings were produced using BERT (Devlin et al., 2018). The matrices show that all the categories share a degree of similarity with each other, with the lowest cosine similarity score being 0.66 between A and O and the lowest Euclidean distance being 10.40 between I and O, suggesting a significant amount of shared language. We also attempt to visualize the dataset in two-dimensional space by taking these sentence embeddings and reducing them using PCA to determine if this leads to organic and discernible groupings by category. From Figure 3·2 (a), we can see that the O category embeddings dominate the entire space. We filter out the O category embeddings for clarity, shown in Figure 3·2 (b). The I and A category embeddings share a similar grouping due to sharing many keywords. The E category embeddings appear to be the most discernible in this figure, but as shown in Figure 3·2, the O category covers the entire space, which suggests a challenge in separating

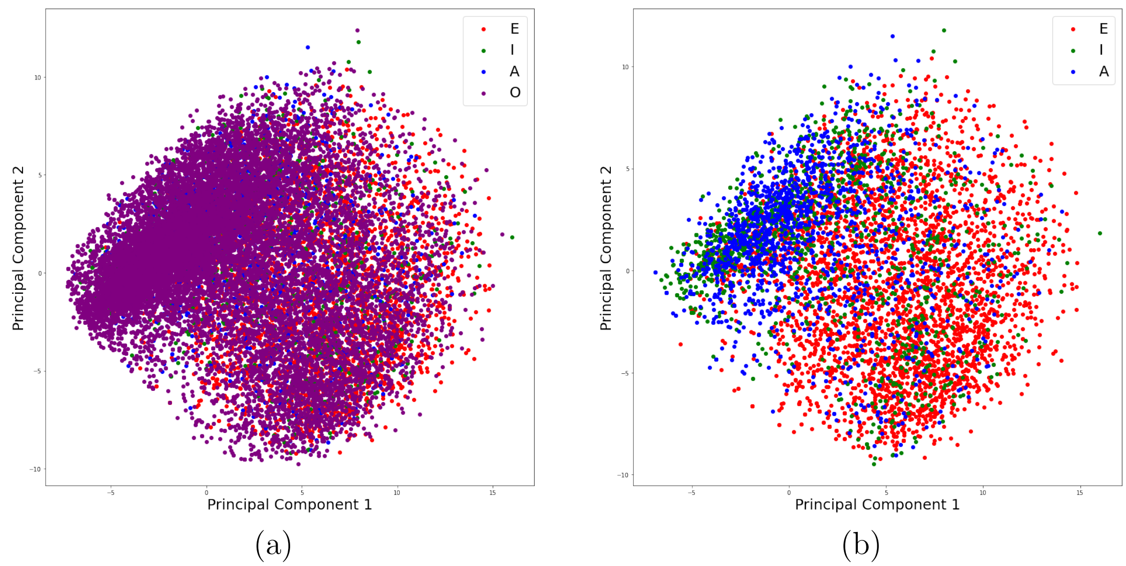


Figure 3·2: Visualization of the average sentence embeddings (produced using BERT (Devlin et al., 2018)) in the DOTA 2 (Weld et al., 2021) dataset. (a) shows that the O category is distributed across the entire visualization space. The higher number of O examples and the sharing of words with the other categories implies any classifier trained on this dataset would be biased. (b) shows that of the other 3 categories, E appears to be the most discernible while A and I appear to share similar words

the other categories from the O category.

3.2 Experiment Setup

We selected four transformers for evaluation in this chapter - BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), GPT-2 (Radford et al., 2019) and ELECTRA (Clark et al., 2020). We use the sequence classification implementation from HuggingFace for each transformer which consists of the base transformer architecture and a classification head. For the ablation experiment, we use an implementation of the model described in Chen et al. (Chen et al., 2019) for Joint BERT and modify this implementation to create a joint version of RoBERTa. All models are trained with a batch size of 32, learning rate of 5e-5, epsilon of 1e-8 and for 2 epochs. The inference time for the models is measured by running a few dummy examples to warm-up

Model	F1(E)	F1(I)	F1(A)	F1(O)
RNN-NLU (Liu and Lane, 2016)	0.813	0.720	0.783	0.944
Slot-Gated (Goo et al., 2018)	0.806	0.694	0.773	0.938
Inter-BiLSTM (Wang et al., 2018)	0.719	0.590	0.728	0.923
Capsule NN (Zhang et al., 2019)	0.735	0.706	0.643	0.926
Joint BERT (Chen et al., 2019)	**0.872**	**0.768**	0.800	0.954
RoBERTa (Liu et al., 2019)	0.852	0.751	0.779	0.951
GPT-2 (Radford et al., 2019)	0.844	0.752	0.754	0.945
ELECTRA (Clark et al., 2020)	0.870	0.761	**0.812**	**0.956**

Table 3.3: (a) Comparison of F1 scores produced by different transformers on the DOTA 2 dataset (Weld et al., 2021) . The transformers shown in bold are the transformers that weren't considered in the original paper. (b) Comparison of F1 scores after word removal using PMI.

the GPU. Then we synchronize the GPU and the CPU to prevent any asynchronous execution. The throughput is measured similarly.

3.3 Results

Table 3.3 compares intent level F1 scores for ELECTRA (Clark et al., 2020), RoBERTa (Liu et al., 2019) and GPT-2 (Radford et al., 2019)to the baseline models provided in the DOTA 2 paper (Weld et al., 2021). The transformers consistently perform well on the O category, given its catch-all nature and its higher prevalence in the dataset compared to the other categories. The lowest F-1 score for the O category is 0.923, and amongst the transformers, the lowest score is 0.945. On the other hand, the I category proves to be the most difficult given the fact that it is the least prevalent in the dataset (along with the A category) and the need for context and domain knowledge to understand the implicit toxicity. As suggested in Figure 3·2, the catch-all nature of the O category and its prevalence in the dataset creates a challenge of distinguishing

Model	F1(H)	F1(O)	F1(N)
BERT (Devlin et al., 2018)	**0.376**	**0.957**	**0.898**
RoBERTa (Liu et al., 2019)	0.230	0.952	0.865
GPT-2 (Radford et al., 2019)	0.231	0.955	0.881
ELECTRA (Liu et al., 2019)	0	0.953	0.838

Table 3.4: Comparisons of F1 scores for the Twitter dataset (Davidson et al., 2017) produced by various transformers.

Model	F1(DEG)	F1(NDG)	F1(HOM)	F1(APR)
BERT (Devlin et al., 2018)	**0.864**	**0.893**	**0.834**	**0.462**
RoBERTa (Liu et al., 2019)	0.856	0.891	0.803	0
GPT-2 (Radford et al., 2019)	0.857	0.886	0.820	0.163
ELECTRA (Clark et al., 2020)	0.857	0.890	0.836	0.359

Table 3.5: Comparisons of F1 scores for the Reddit dataset (Kurrek et al., 2020) produced by various transformers.

the other categories from O. The most common type of error across all experiments is incorrectly predicting other categories as the O category. Figure 3·3 also compares the precision-recall curves for the four transformers. Generally, all transformers can maintain high precision scores as the recall increases for the O and E categories. For the A and I categories, the high precision can only be maintained until recall scores of approximately 0.8 and 0.6, respectively.

We also observe that BERT produces the best results for the E and I categories while ELECTRA performs best with respect to the A and O categories however the difference in performance is not significant as the two models produce F1 scores approximately within 1 points when comparing the respective categories. RoBERTA and GPT-2 produce lower scores across all categories, particularly for the E and A categories where the perform 2-3 and 2-5 points worse than the best performing

Model	Inference Time(ms)	Throughput (instances/s)
GPT-2 (Radford et al., 2019)	13.111	592.769
RoBERTa (Liu et al., 2019)	11.187	718.091
BERT (Devlin et al., 2018)	**10.829**	**725.983**
ELECTRA (Clark et al., 2020)	37.005	212.257

Table 3.6: Comparison of inference time and throughput of transformers when performing classification on the DOTA 2 dataset. (Weld et al., 2021).

model.

Tables 3.4 and 3.5 contain baseline F1 scores for the Twitter (Davidson et al., 2017) and Reddit (Kurrek et al., 2020) datasets. We observe that both datasets consist of low-performing categories regardless of the model used. This is similar to the DOTA 2 dataset, where the Implicit and Action categories consistently had lower F1 scores than the Explicit and Other categories. In the Twitter dataset, the lowest performing category is Hate Speech (H), with F1 scores ranging from 0.231-0.4, while in the Reddit dataset, the lowest performing category is the Appriopriative Usage category (APR). The common denominator here is that these categories make up a small percentage of the total number of examples in their respective datasets. BERT is the best-performing model for both datasets. We observe that RoBERTa (Liu et al., 2019) and ELECTRA (Clark et al., 2020) are unable to correctly classify examples of the lowest volume categories in the Reddit and Twitter datasets, respectively.

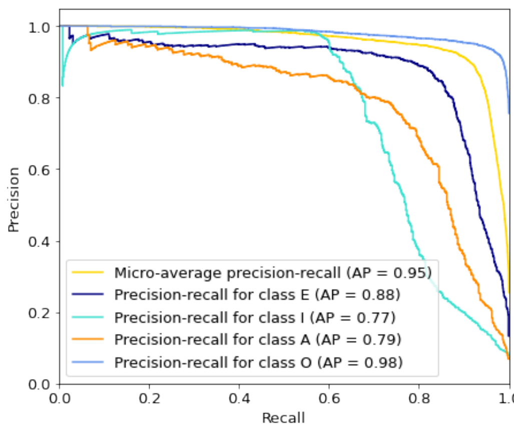
BERT performs well and has the fastest inference time. Table 3.6 compares the inference time and throughput of the various transformers. While ELECTRA (Clark et al., 2020) produces F-1 scores on par with BERT (Devlin et al., 2018) and better than RoBERTa (Liu et al., 2019), it is over three times slower in inference time, and its throughput is also over three times lower. This is an important consideration as a toxicity detection system in online games must make a fast inference

Model	F1(E)	F1(I)	F1(A)	F1(O)
Joint BERT (Chen et al., 2019)	0.872	**0.768**	0.800	0.954
BERT (Devlin et al., 2018)	**0.879**	0.765	**0.803**	**0.957**
Joint RoBERTa (Chen et al., 2019)	0.839	0.713	0.775	0.943
RoBERTa (Liu et al., 2019)	**0.852**	**0.751**	**0.779**	**0.951**

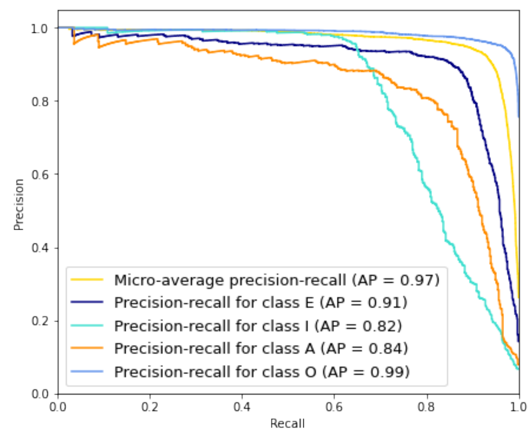
Table 3.7: Comparison of F1 scores for RoBERTa (Liu et al., 2019) and BERT (Devlin et al., 2018) and their joint equivalents (Chen et al., 2019). For this experiment, we modified the joint implementation of BERT for RoBERTa and compare it to the values from 3.3. We also compare BERT to its joint equivalent from the CONDA paper (Weld et al., 2021).

in order to intervene quickly and prevent further toxic communication. Given that BERT produces the best results and has a much faster inference time, we experiment with BERT as the primary transformer model in future experiments.

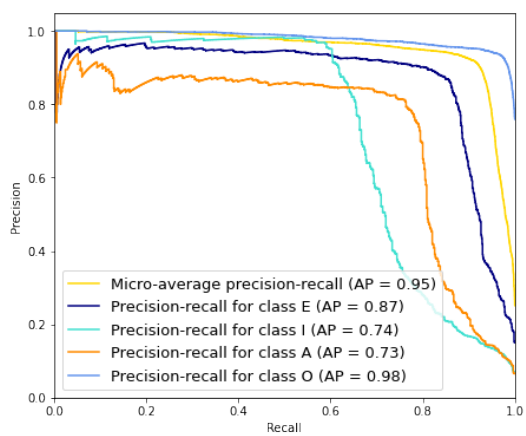
Joint training on slot and intent classification does not significantly improve intent classification. We also conducted an ablation experiment to determine if jointly training to predict intent and slot level categories can lead to better understanding, and therefore performance, of intent level categories. Table 3.7 reports the results of this experiment. The joint versions of BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) perform similarly on all categories with not more than a 1-point difference in the F1 scores for any of the categories. Therefore we elect not to use the joint version of BERT in the coming experiments.



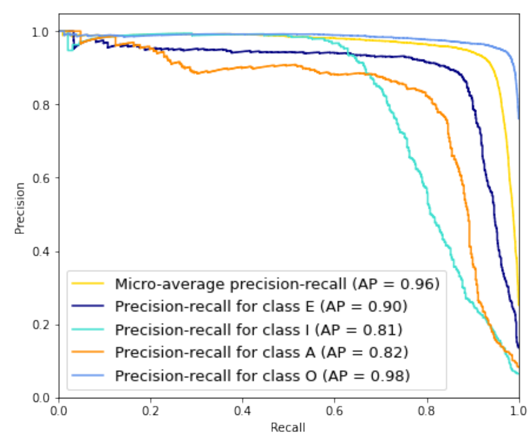
(a)



(b)



(c)



(d)

Figure 3·3: Precision-recall curves for 4 different transformers. All transformers can maintain high precision and recall for E and O. However, for A and I, transformers can only maintain high precision for recall scores of approximately 0.8 and 0.6 respectively.

Chapter 4

Identification and removal of non-keywords to emphasize domain-specific language

The previous chapter established baseline performance using pre-trained transformer models to perform classification on various datasets. This chapter explores an offline word filtering step that identifies non-keywords in a dataset and removes them before fine-tuning. For each unique word-category pair *(w,c)* in a dataset, we assign it a score s where s = *pmi(w,c)* and remove the word w from examples of category c if *pmi(w,c)*<0. We use this step to address linguistic challenges such as domain-specific language and spelling errors that are typically OOV for general-purpose pre-trained language models. Since these words are typically split into sub-words that produce separate embeddings, the information held by the original word can be lost, which can be problematic as these words can contain essential domain knowledge. We attempt to address this by identifying and removing non-keywords beforehand, forcing a transformer's attention on the most significant words and improving the contribution of the remaining words to the final classification. This process yields a dataset containing trimmed versions of the original examples, which we use as input to BERT for classification. We conduct this procedure on datasets from three different domains - the DOTA 2 video game (Weld et al., 2021), Twitter (Davidson et al., 2017), and Reddit (Kurrek et al., 2020) in order to validate this method's applicability to different online spaces.

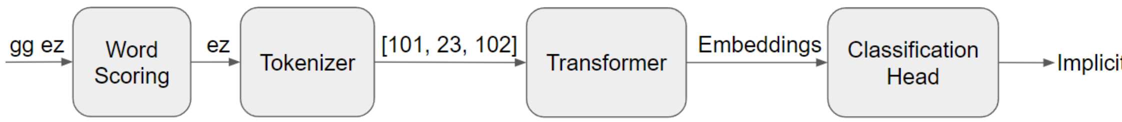


Figure 4·1: The workflow of using a word scoring method to trim examples prior to passing inputs to the transformer. Words are scored using a method like PMI and removed accordingly.

4.1 Pointwise Mutual Information

Pointwise mutual information is a measure of association that compares the probability of two events occurring together to the probability of the events occurring independently. If x and y are outcomes belonging to discrete random variables X and Y, then the PMI of x and y is then defined as follows:

$$pmi(x;y) = log\frac{p(x,y)}{p(x)\cdot p(y)} \tag{4.1}$$

In NLP, PMI is normally used to as a way to measure the co-occurrence of words in a corpus. In the context of scoring how important a word is to a category, we use an adapted version of the above equation (Schneider, 2005). We measure this importance as follows:

$$pmi(w;c) = log\frac{p(w,c)}{p(w)\cdot p(c)} \tag{4.2}$$

Where $p(w,\ c)$ is the probability of an utterance of class c containing the word w, $p(w)$ is the probability of an utterance containing a word w and $p(c)$ is the probability of an utterance belonging to class c. In the following section, we explore using equation 4.2 to assign scores word-category pairs and use this score to trim the dataset before classification.

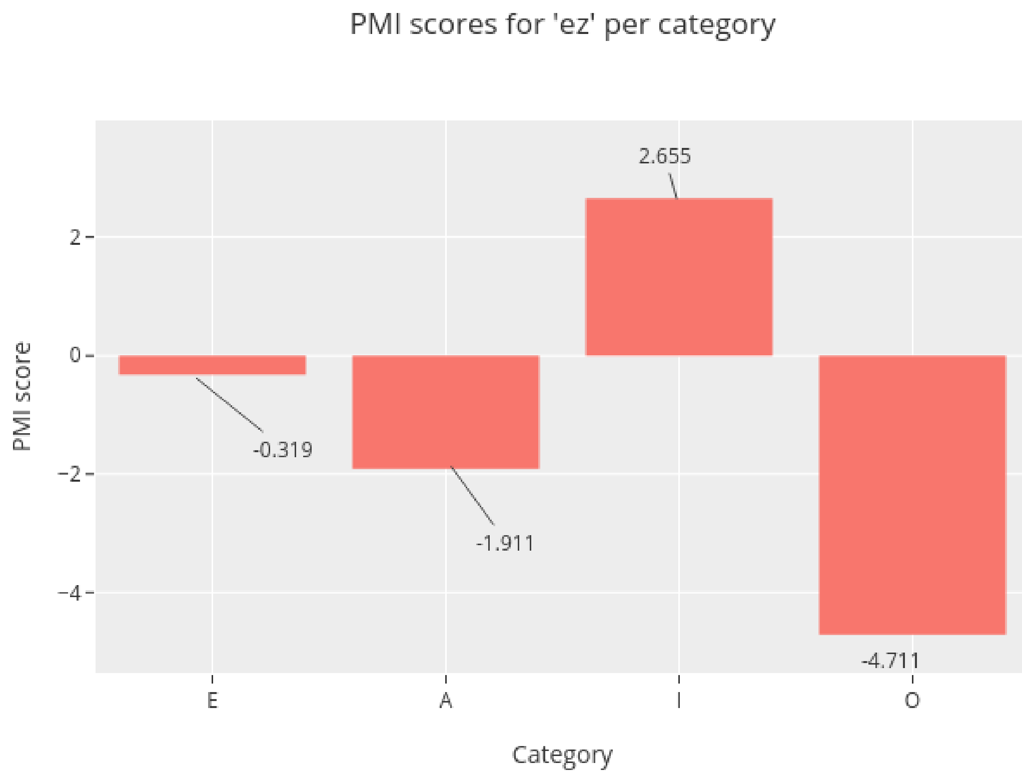


Figure 4·2: Visualization of PMI scores for the word 'ez' for each category in the DOTA 2 dataset (Weld et al., 2021) using equation 4.2.

4.2 Offline Word Filtering using PMI

4.2.1 Word Scoring

As mentioned earlier, we use equation 4.2 to score each word in each category. First, we use the CountVectorizer package from sklearn to vectorize all the examples in a dataset which allows us to build a dictionary to keep track of how many times each word in the vocabulary appears in the utterance of a given category. For example, in the DOTA 2 dataset (Weld et al., 2021), the word 'ez' appears 12 times in the Other category, 145 times in the Explicit category, 14 times in the Action category, and 1351 times in the Implicit category. Building this dictionary already gives us a sense of which words are essential to a category.

This dictionary allows us to calculate all the probabilities required for equation 4.2. *p(w, c)* is obtained by dividing the number of unique examples in a category that a word appears in by the total number of examples. *p(c)* is the ratio of all the examples belonging to a category in the dataset, and *p(w)* is the number of examples the word occurs across all categories divided by the total number of examples. Figure 4·2 shows the results of calculating PMI scores for the word 'ez' in the DOTA 2 dataset (Weld et al., 2021). We observe that this word only scores positively for the Implicit category. Despite the 'ez'-Explicit pair making up for nearly 10% of all unique appearances of the word across examples, the word still scores negatively for the Explicit category. By removing this word from all categories except Implicit, we force a stronger association between the word and Implicit category during the training process.

4.2.2 Word Removal Procedure

Once we have calculated the PMI scores for every word in all categories, we can build a list of words we want to remove per category. For each category, we look at the PMI score for every word in its vocabulary and add it to the list of words we want to remove if its score is negative. Using the same example above would mean that we add 'ez' to the list of words for the Explicit, Action, and Other categories but not the Implicit categories. Once we have the complete list of words for each category, we iterate through all the examples in the dataset and remove any words that appear in the respective category's list, which produces a trimmed version of the dataset which we use to fine-tune BERT. Since this process occurs before fine-tuning, scoring and removal occur at the word level, not the sub-word level.

There are two important cases to consider when removing words. The first is the scenario where the example contains only one word. In these cases, we preserve the

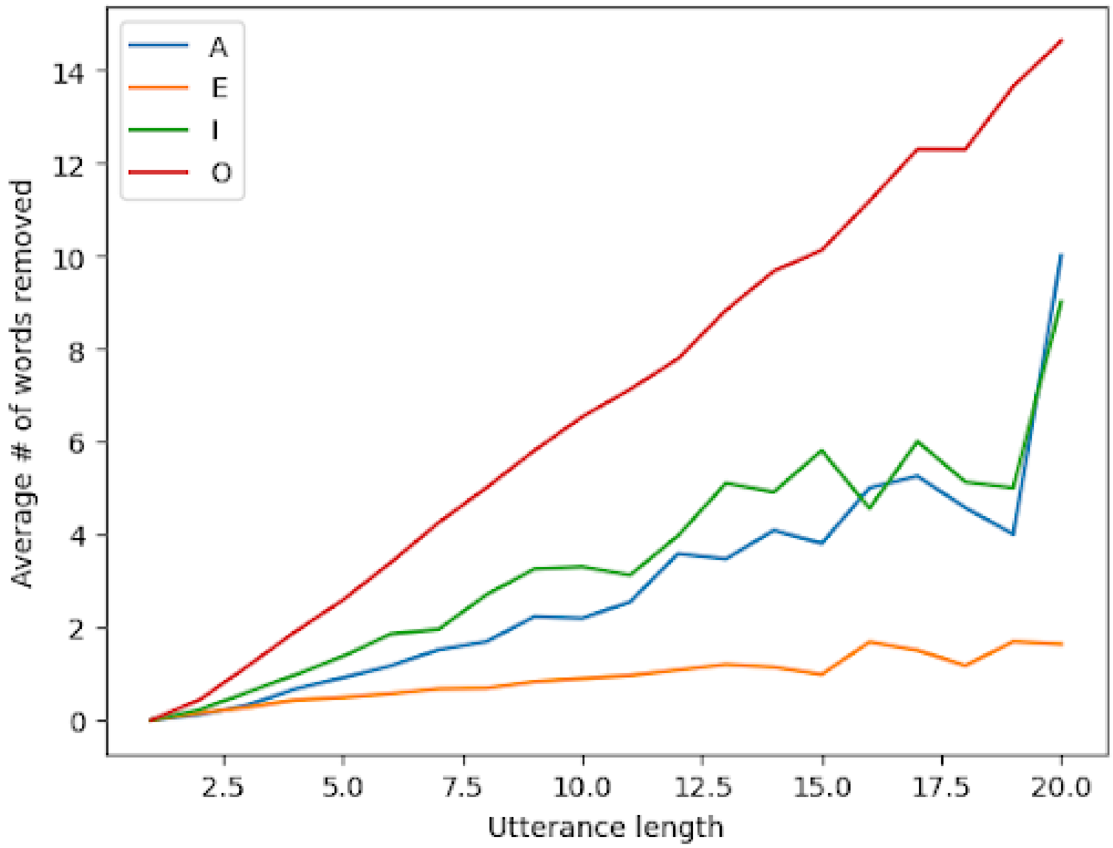


Figure 4·3: Visualization of the average number of words removed per length of the examples for the different categories in the DOTA 2 dataset (Weld et al., 2021) when using PMI to score and remove non-keywords.

example as we want to avoid any empty sentences. The second is when every word in the example has been assigned a negative score. During the process of removing words, if the number of words in the example falls to 0, we preserve the example in its original form without removing any words. An alternative approach would be to remove words until the number of words falls to some k.

4.2.3 Analysis

Figure 4·3 shows how the number of words in an example affects the number of words removed per category. Our word removal procedure impacts the Other category

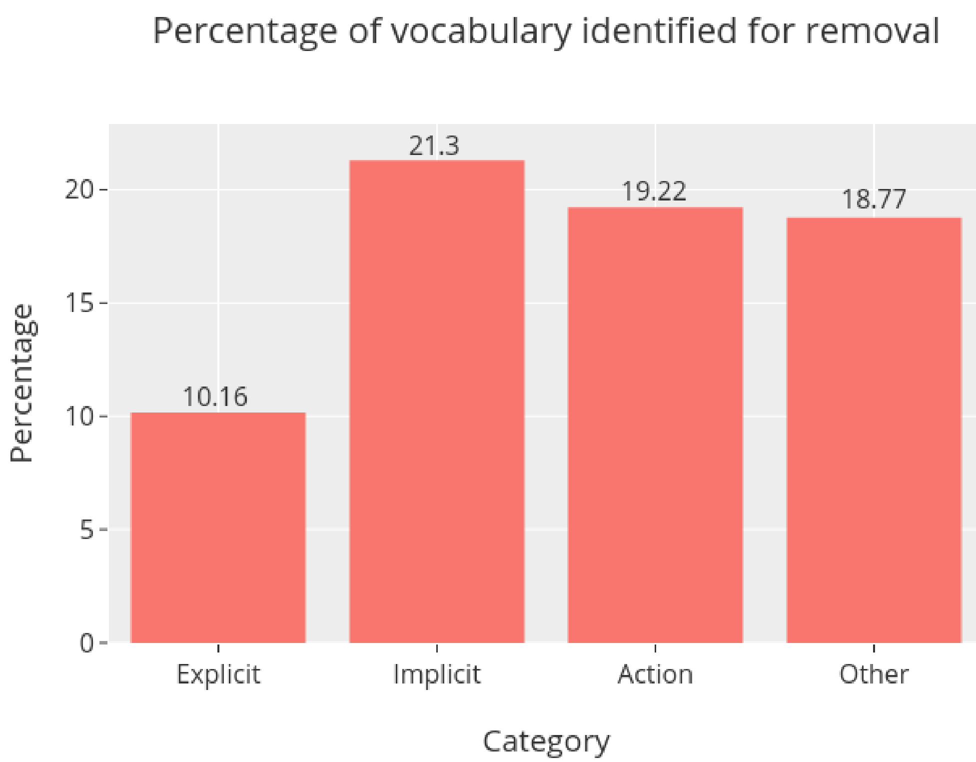


Figure 4·4: Visualization of the percentage of each category's vocabulary identified as non-keywords in the DOTA 2 dataset (Weld et al., 2021) using PMI scores.

heavily. On the other hand, examples in the Explicit category are impacted the least. Generally speaking, Explicit examples are easier to identify since they typically involve using vulgar language and swear words, requiring less trimming. The Other category is a generic, catch-all category that includes all examples that are not toxic and do not involve the use of game-related action vocabulary and therefore require the most trimming to enhance separability. The Action and Implicit categories fall in between Explicit and Other and show a similar trend to each other.

Figure 4·4 visualizes the percentage of a category's vocabulary identified as non-keywords in the DOTA 2 dataset (Weld et al., 2021). The Explicit category was the least impacted, with only 10.2% of its vocabulary marked as non-keywords. The other categories had a more significant percentage of their vocabulary impacted, with the

Implicit category having approximately 21% of its vocabulary identified for removal. There was some overlap of certain non-keywords between some categories, but no words appeared for removal across all categories.

Figure 4·5 visualizes some of the positive-scoring words for each category in the DOTA 2 dataset when using PMI. These words remain in the dataset after trimming and are therefore emphasized in the training process. For example, the Implicit category's word cloud shows words like "mmr," "botlane," and "2-5" score positively. These words are DOTA 2-specific language and provide important context during classification. However, as mentioned earlier, this can be an issue for pre-trained language models because these words are OOV. The figure also highlights words that are misspelled, like "repoirt" and "invoekr" which are also OOV. For a model like BERT, this means the word is split into separate sub-words where each sub-word gets a separate embedding which can result in loss or dilution of information contained by the original word. By removing low-scoring words that appear in the same sentence as the positively-scoring ones, we emphasize and improve the contribution of the remaining words, even if they are OOV, and produce multiple embeddings during the training process.

4.3 Results

Identifying and removing the correct set of words for each category improves performance for abusive language detection. Table 4.1 compares the results of using the offline word filtering step with different transformers to the baseline results established in Chapter 3 for the DOTA 2 dataset (Weld et al., 2021). We observe an increase in F1 scores across all categories regardless of which transformer model is used. BERT (Devlin et al., 2018) performs best across all categories with the trimmed dataset, seeing a 4-point increase in the F1 score for the toxic categories

Model	F1(E)	F1(I)	F1(A)	F1(O)
BERT	0.875	0.770	0.806	0.958
BERT-PMI	**0.918**	**0.839**	**0.887**	**0.974**
GPT-2	0.844	0.752	0.754	0.945
GPT-2-PMI	0.877	0.773	0.803	0.955
RoBERTa	0.852	0.751	0.779	0.951
RoBERTa-PMI	0.888	0.756	0.805	0.959
ELECTRA	0.870	0.761	0.812	0.956
ELECTRA-PMI	0.913	0.799	0.861	0.968

Table 4.1: Comparison of F1 scores produced by different transformers on the DOTA 2 dataset (Weld et al., 2021) to results using a trimmed version of the dataset created using PMI scores.

Model	F1(H)	F1(O)	F1(N)
BERT	0.376	0.957	0.898
BERT-PMI	**0.844**	**0.992**	**0.973**

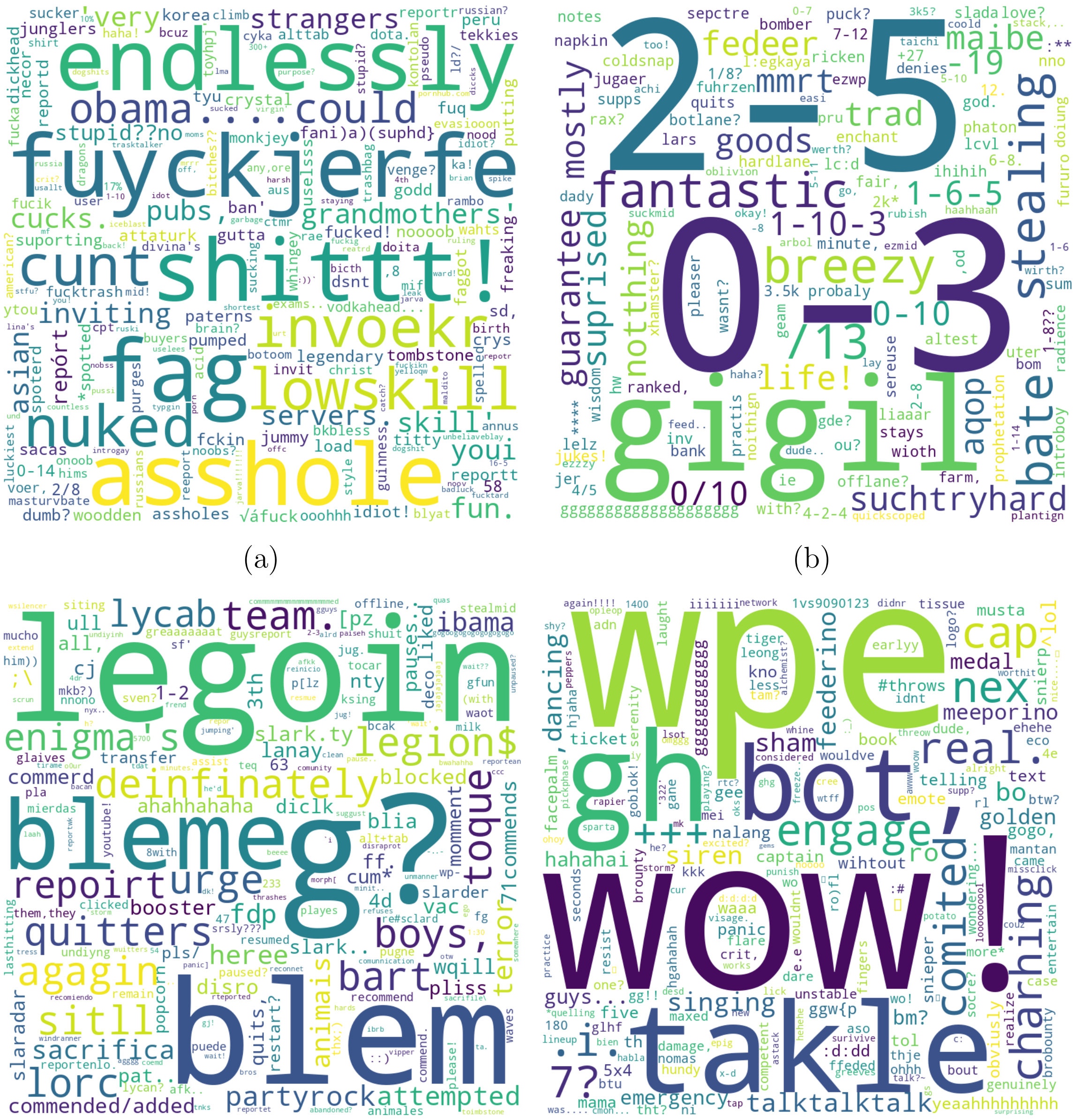
Table 4.2: Comparison of F1 scores produced by the BERT baseline on the Twitter dataset (Davidson et al., 2017) to results using a trimmed version of the dataset created using PMI scores.

Model	F1(DEG)	F1(NDG)	F1(HOM)	F1(APR)
BERT	0.864	0.893	0.834	0.462
BERT-PMI	**0.995**	**0.990**	**0.927**	**0.905**

Table 4.3: Comparison of F1 scores produced by the BERT baseline on the Reddit dataset (Kurrek et al., 2020) to results using a trimmed version of the dataset created using PMI scores.

(Explicit and Implicit) compared to the BERT baseline. We also observe a 6-point increase in the Action category. ELECTRA (Clark et al., 2020) also appears to benefit from using the trimmed dataset with approximately a 4-point increase to the Explicit and Implicit categories and a 5-point increase to the Action category. GPT-2 (Radford et al., 2019) and RoBERTa (Liu et al., 2019) see an increase in F1 scores across the board, but both the delta of improvement and performance are lower than BERT and ELECTRA.

Offline word filtering using PMI also improves abusive language detection in other online domains. Tables 4.2 and 4.3 compare the results of using the offline word filtering step for the Twitter (Davidson et al., 2017) and Reddit (Kurrek et al., 2020) datasets. Similar to the DOTA 2 dataset, we observe an improvement in F1 scores for all categories. A common theme is that the categories that gain the most benefit from PMI-based word removal are the ones that make up a small portion of the dataset. The Implicit, Hate Speech, and Appropriative categories cover approximately 6.3%, 5.7%, and 1.4% of their respective datasets. As seen in the tables above, we observe improvements of up to 7, 47, and 45 points, respectively. Generally, the offline word filtering step is able to identify important words in a category, even if they are OOV by virtue of being domain-specific or misspelled, and emphasize them in the dataset while also improving the separability of categories, by removing words of low importance which helps improves abusive language detection across datasets from different online domains.



(a) (b)

(c) (d)

Figure 4·5: Wordcloud visualizations for words that were identified as keywords for the (a) Explicit (b) Implicit (c) Action and (d) Other categories in the DOTA 2 dataset (Weld et al., 2021) during the offline filtering step.

Chapter 5

Learned Offline and Online Word Filtering

This chapter introduces alternative methods for word scoring to identify non-keywords, focusing on learning word scores. We explore additional offline filtering methods and introduce an online filtering framework to identify non-keywords. The offline filtering methods discussed here are an adaptation of what was introduced in Chapter 4 with the same idea - score word-category pairs to identify which words to retain and produce a trimmed version of the dataset before training a transformer to perform classification. The online framework uses the original version of the datasets and attempts to identify non-keywords within the fine-tuning loop of the transformer without needing to rely on knowledge of word count statistics in the dataset.

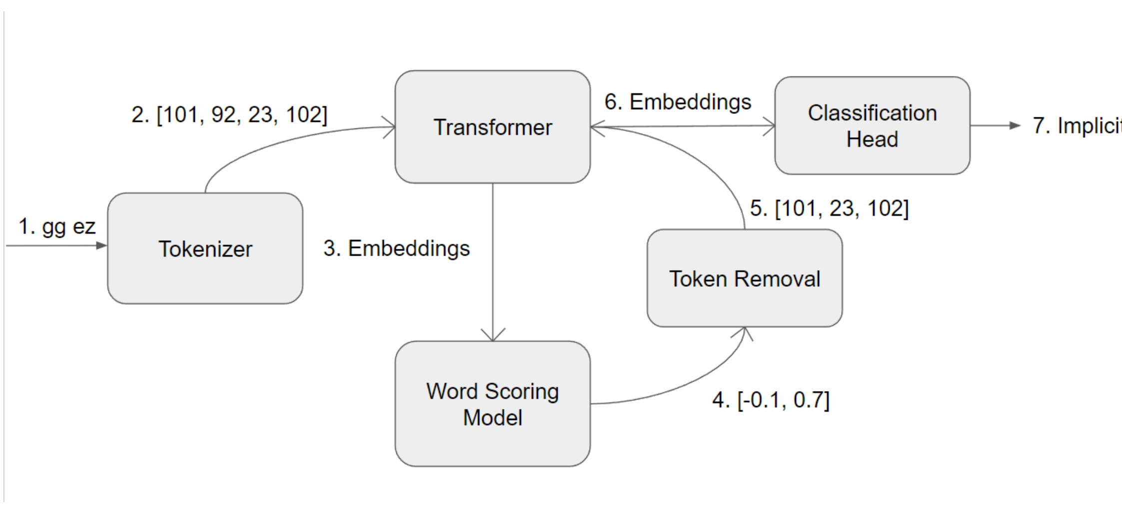


Figure 5·1: The workflow of the two-pass framework. Inputs are tokenized and then passed to BERT. The outputs from BERT are used in the word scoring model and the resulting scores are used to determine which tokens are removed. The reduced input is then passed to BERT one more time to produce the final classification prediction.

5.1 Learned offline word scoring and filtering

5.1.1 Procedure

Our first alternative offline word scoring method is learning PMI scores for words using FastText embeddings (Bojanowski et al., 2017). We train a FastText model on our datasets, which ensures that no words are considered OOV for the FastText model, and use it to produce 100-dimension embeddings for every word in the vocabulary. We also count how many times each word appears across all categories and append these counts to the FastText embedding. For words in the DOTA 2 dataset (Weld et al., 2021), this means we end up with 104 dimension embeddings for each word.

Using the DOTA 2 dataset as an example again, we have a vocabulary of 11,607 words. We generate combined embeddings for these words and split them into a train and test set using an 80/20 split. The train set (9285 embeddings) is passed through a dense network with a tanh activation function to produce scores for each input embedding. We train this model for ten epochs. The tanh activation produces values ranging from -1 to 1. The output corresponds to the word's score for each category, where the order of categories in the output vector matches the order in which the word count vector that was appended to the embeddings was constructed. Taking the same example from Chapter 4, let us assume that passing the embedding for the word 'ez' produces the following output - [-0.6, -0.4, 0.8, -0.9], and the indices correspond to the Explicit, Action, Implicit, and Other categories respectively. Since the first, second, and last values are negative in this scenario, we add the word to the lists for Explicit, Action, and Other. Once we build the lists for each category, the procedure is the same as before. We iterate through all the examples and remove any words that appear in the list for that example's category. An important aspect is that we still rely on using the previously calculated PMI scores as they are the target when training the word scoring model.

We also experiment with an unsupervised offline word filtering method. We use the exact embeddings as in the supervised method - FastText embeddings (Bojanowski et al., 2017) in conjunction with word-category co-occurrences. Then, the combined embeddings have their dimensionality reduced to the number of categories in the dataset, representing the word's score for each category. We experiment with the unsupervised variant as a form of ablation to determine if we need to explicitly calculate PMI scores to identify a suitable set of non-keywords to remove.

5.1.2 Results

PMI-based methods produce the best results but eliminating explicit computation of PMI scores can still result in an improvement over the baseline. Table 5.1 contains the results for the various scoring methods in conjunction with BERT (Devlin et al., 2018) on the DOTA 2 dataset (Weld et al., 2021). Using PMI as our word scoring method (BERT-PMI) produces the best F1 scores for the Action and Other categories, and using supervised word filtering (BERT-FT-PMI) yields the best results on the Explicit and Implicit categories, but the differences are marginal. The unsupervised filtering (BERT-FT-PCA) does not perform as well as the previous methods but produces higher F1 scores than BERT across all categories, which suggests that explicit computation of PMI scores is required for the best performance but not necessary to improve upon the baseline. Generally speaking, we observe that using various offline word filtering methods to choose specific words to remove selectively can improve F1 scores and is particularly impactful for categories with a low volume in the dataset. In the DOTA 2 dataset specifically, the two low-volume classes Implicit and Action see improvements of 7-8 and 3-8 points, depending on the filtering method used.

Tables 5.2 and 5.3 compare the various offline filtering methods to the BERT baseline for the Twitter (Davidson et al., 2017) and Reddit (Kurrek et al., 2020) datasets,

Model	F1(E)	F1(I)	F1(A)	F1(O)
BERT	0.875	0.770	0.806	0.958
BERT-PMI	0.918	0.839	**0.887**	**0.974**
BERT-FT-PMI	**0.922**	**0.850**	0.873	0.971
BERT-FT-PCA	0.882	0.846	0.833	0.962
2 pass BERT	0.876	0.775	0.791	0.956

Table 5.1: Comparison of F1 scores produced by BERT in conjunction with various filtering methods on the DOTA 2 dataset (Weld et al., 2021).

Model	F1(H)	F1(O)	F1(N)
BERT	0.376	0.957	0.898
BERT-PMI	0.844	**0.992**	**0.973**
BERT-FT-PMI	**0.891**	**0.992**	0.969
BERT-FT-PCA	0.416	0.962	0.885
2 pass BERT	0.410	0.955	0.873

Table 5.2: Comparison of F1 scores produced by BERT in conjunction with various filtering methods on the Twitter dataset (Davidson et al., 2017).

respectively. PMI-based filtering methods produce the best results for the Twitter dataset (Davidson et al., 2017). We observe that unsupervised filtering improved the Hateful and Offensive categories but saw a 1-point decrease for the Neither category when compared to the BERT baseline. 5.3 highlights the results when using the Reddit (Kurrek et al., 2020) dataset, which are consistent with the DOTA 2 dataset in that all the offline word filtering methods improve upon the BERT baseline with the PMI-based filtering methods producing the best results.

Model	F1(DEG)	F1(NDG)	F1(HOM)	F1(APR)
BERT	0.864	0.893	0.834	0.462
BERT-PMI	**0.995**	0.990	0.927	**0.905**
BERT-FT-PMI	**0.995**	**0.995**	**0.948**	0.881
BERT-FT-PCA	0.984	0.987	0.942	0.697
2 pass BERT	0.853	0.883	0.799	0.417

Table 5.3: Comparison of F1 scores produced by BERT in conjunction with various filtering methods on the Reddit dataset (Kurrek et al., 2020).

5.2 Online word scoring and filtering

5.2.1 Overview

Our second method is an in-training, online method of scoring words that utilize passing our input through BERT twice with a word scoring model acting as a middle layer, which is highlighted in Figure 5·1. This two-pass framework is based on previous work where the authors utilize a first-pass and second-pass decoder with different objectives (Inaguma et al., 2023). The overall flow follows: We pass our input tokens to BERT and use the output vectors as input to the word scoring model. The model produces a value for each word ranging from 0 to 1. For every value ¡ 0.5, we remove the corresponding input token and reapply padding before passing through BERT again to produce our final classification output. Figure 5·1 highlights the entire flow. BERT's training parameters were the same as in Chapter 2; we trained for two epochs with a learning rate of 5e-5 and an epsilon of 1e-8. The word scoring model is trained in the same loop and with the same optimizer but has a learning rate of 1e-3.

5.2.2 First Pass

The purpose of our first pass through BERT is to obtain embeddings for our input words to pass to the scoring model, similar to how we used FastText embeddings earlier. The key difference here is that we do not expect the embeddings for the same word used in different sentences to be the same due to attention. However, we hope the attention-based embeddings provide additional information to help in word scoring. The other difference here is that the embeddings technically represent tokens, not words, due to how the WordPiece tokenizer handles OOV words. For example, consider the following example from the DOTA 2 dataset: 'gg ez.' The WordPiece tokenizer produces six sub-word tokens - ['CLS', 'g', '##g', 'e', '##z', 'SEP'] which results in 6 embeddings being created after our first pass through BERT. These six embeddings are then used as input to our word-scoring model.

5.2.3 Word Scoring

The second step involves passing the learned token embeddings to a word-scoring model. The scoring model is a dense layer with a tanh activation function that produces outputs in the range [0, 1]. The main difference in this scenario is that we score each word in a binary fashion - to keep or not to keep. Previously when using FastText embeddings, we produced multiple outputs for each word where each output value corresponded to a category. Here we produce a single value per word that determine if we keep the word moving forward. If the value is ¡ 0.5, we remove the token number from our input list and do so until we have considered all the scores. We also remove the corresponding indices in the attention mask input. Once the example has been trimmed, we reapply padding to both the input tokens and the attention mask to maintain the original length. We limit the number of tokens being removed to a fourth of the original length of the input.

5.2.4 Second pass

The last step involves passing our modified inputs to BERT to produce the final classification output. This is the output that is used to calculate the F1 scores that are used for evaluation. An important detail here is that we use the loss of the second pass through BERT in the backpropagation through the scoring model by attaching the BERT loss to the computational graph of the scoring model. This is another critical difference from our first method, where we use PMI scores as the target, which allows us to easily calculate the loss for the backpropagation. The scoring model in this method relies on BERT to supervise the learning of the word-scoring model, where a low loss after the second pass allows the scoring model to know that the words it chose for removal were the correct words.

5.2.5 Results

Table 5.1 also shows the results of using the two-pass framework and compares them to previous results. Generally speaking, the two-pass framework produces F1 scores comparable with the BERT baseline for all categories, except for the Action category, where it performs 1-point worse and performs worse than using offline word filtering. While the two-pass framework produced slightly improved F1 scores for the two toxic categories, Explicit and Implicit, these scores are not statistically significant. Tables 5.2 and 5.3 show results of the two-pass framework for the Twitter (Davidson et al., 2017) and Reddit datasets (Kurrek et al., 2020), respectively. For the Twitter dataset, we observe a gain of 4 points in the Hateful category and a loss of 2 points in the Neither category when compared to BERT. In the Reddit dataset, we observe a decrease in performance across all categories.

Word removal can have diminishing returns. Figure 5·2 compares the percentage of non-keywords identified by BERT-PMI that were also identified by the learned filtering methods explored in this chapter. We use this comparison to determine how well the learned methods can identify non-keywords. Despite BERT-FT-PMI identifying between 50.3% to 88.8% of non-keywords depending on the category, it still performs on par with BERT-PMI on all datasets, which suggests that most

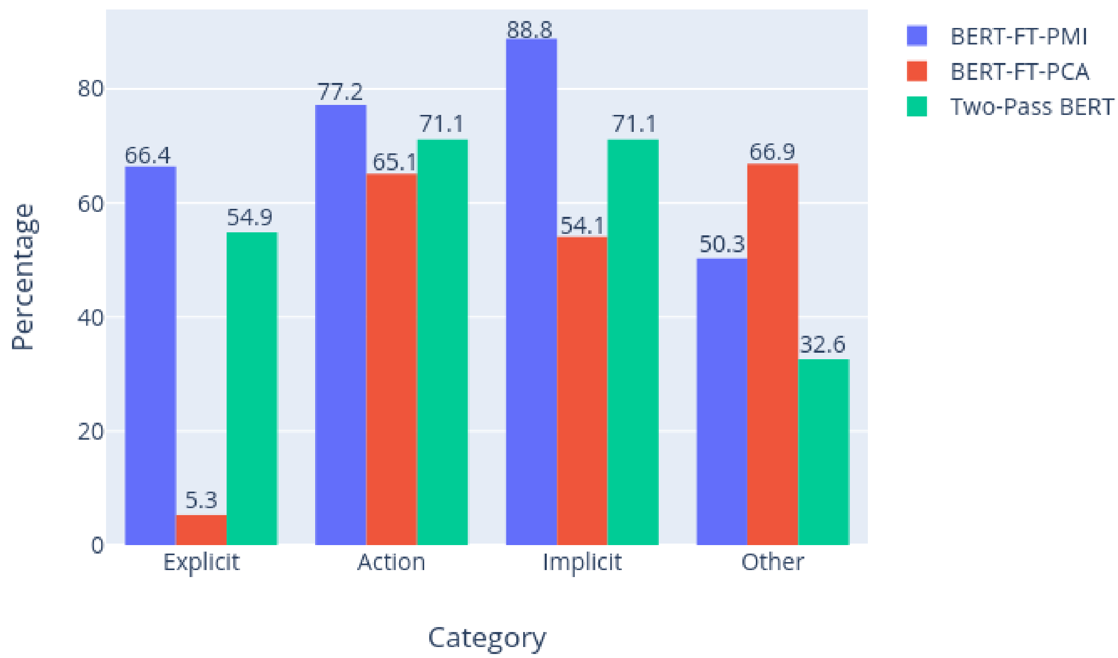


Figure 5·2: Comparison of the percentage of non-keywords identified by BERT-PMI that were also identified by the learned filtering methods.

of the performance gain is contained in removing words that score most negatively. In comparison, BERT-FT-PCA identified a smaller percentage of non-keywords and produced lower F1 scores than BERT-FT-PMI.

Online word filtering does not choose an appropriate set of non-keywords that can improve performance. As mentioned earlier, there are critical differences between the offline filtering methods and the online two-pass framework. PMI relies on calculating probabilities and knowing how often words co-occur with categories, allowing us to use it as a measure of importance. Intuitively, this allows us to choose the correct words to remove and improve performance. The unsupervised offline filtering approach using FastText (Bojanowski et al., 2017), and PCA eliminates the need for

Model	F1(E)	F1(I)	F1(A)	F1(O)
BERT	0.875	0.770	0.806	0.958
BERT-FT-PMI	0.922	**0.850**	**0.873**	**0.971**
BERT-FT-PMI-NWC	0.911	0.741	0.767	0.959
BERT-FT-PCA	0.882	0.846	0.833	0.962
BERT-FT-PCA-NWC	**0.925**	0.635	0.744	0.949

Table 5.4: Ablation experiment with supervised and unsupervised offline word filtering on the CONDA dataset (Weld et al., 2021).

calculating these probabilities. However, we still utilize the knowledge of how often words appear in different categories. In contrast, the two-pass framework does not utilize this feature and therefore does not perform as well. Figure 5·2 highlights the issue of selecting a suitable set of words. We observe that despite identifying a larger percentage of non-keywords than BERT-FT-PCA, it has sub-optimal performance, indicating that it cannot identify the most impactful non-keywords.

Model	Acc
Weld et al.	0.92
BERT-PMI	**0.95**
BERT-FT-PMI	**0.95**
BERT-FT-PCA	0.94
2 pass BERT	0.88

(a)

Model	F1
Zhang et al.	0.94
BERT-PMI	**0.98**
BERT-FT-PMI	**0.98**
BERT-FT-PCA	0.92
2 pass BERT	0.88

(b)

Model	F1
Kurrek et al.	0.89
BERT-PMI	**0.95**
BERT-FT-PMI	**0.95**
BERT-FT-PCA	0.90
2 pass BERT	0.87

(c)

Table 5.5: Comparison of results with past work of (a) (Weld et al., 2021) (b) (Zhang et al., 2018) and (c) (Kurrek et al., 2022) on the (a) DOTA 2 (Weld et al., 2021) (b) Twitter (Davidson et al., 2017) and (c) Reddit (Kurrek et al., 2020) datasets. Each table uses a comparable metric based on what was reported by the authors. (b) uses micro-averaged F1 scores and (c) uses macro-averaged F1 scores.

Knowledge of word-category co-occurrences is crucial to choosing a set of non-keywords that can improve performance. We investigate the importance of word-category co-occurrence using previous word filtering methods by performing an ablation experiment. We perform word filtering using supervised and unsupervised methods without adding a word's category appearance counts to its FastText embedding. Table 5.4 reports the results of this experiment on the DOTA 2 dataset (Weld et al., 2021). In both scenarios (BERT-FT-PMI-NWC and BERT-FT-PCA-NWC), we see an improvement for the Explicit category over the BERT baseline, but this comes at a significant cost to the performance of the Action and Implicit categories. These results highlight why the two-pass framework might not be selecting the correct words to remove. While the need to calculate PMI scores can be eliminated by using unsupervised offline word filtering, knowledge of word-category co-occurrences remains an integral part of being able to choose an accurate set of non-keywords.

We also compare the various filtering methods to previous work using the same datasets as highlighted in Table 5.5 using a comparable metric based on what was reported in the original papers. For the DOTA 2 and Reddit datasets, all word filtering methods show an improvement over the highlighted previous work. For the Twitter dataset, PMI-based filtering produces the best results, while unsupervised filtering performs on par. Generally speaking, offline word filtering methods produce competitive results as long as they utilize word-category co-occurrences during the process.

Chapter 6

Conclusions

6.1 Summary

Abusive language detection is an essential task in NLP that can lead to a better and safer online user experience. In this thesis, we focused on improving abusive language detection in social online domains that present linguistic challenges such as domain-specific language or grammatical and spelling errors. We explored using pre-trained transformers for this task and established baseline performance. We then moved on to identifying non-keywords to improve performance for abusive language detection using PMI, and lastly, we explored alternative methods for identifying keywords.

In Chapter 3, we conducted experiments to provide baseline results for various transformers on chat data from DOTA 2 (Weld et al., 2021). We also set baselines for datasets from Twitter (Davidson et al., 2017), and Reddit (Kurrek et al., 2020) to which we compare further results. We highlight different linguistic challenges that can impede abusive language detection, such as short examples in the datasets and semantic similarity between different categories. We also investigated and compared the throughput and inference time of the different transformers on the CONDA dataset.

Chapter 4 introduced an offline filtering step where we score and identify non-keywords in a given dataset. We use an adapted version of the pointwise mutual information formula to determine how important each word is to the different categories in a dataset. We use these scores to remove any words from examples in the dataset that fall below a certain threshold. This process produces a trimmed version of the dataset, which we then use as input to BERT. We show that offline filtering identifies domain-specific and even misspelled words as essential and preserves these words. We discover that this method helps performance across all categories, especially those comprising a small percentage of the dataset. This method applies well to datasets from different domains. Generally, choosing the correct words to remove is a powerful tool for abusive language detection, as we can force emphasis on the most relevant words to a category before the training process. Removing non-keywords also makes the categories more separable from each other, given that the set of removed words differs between categories.

Chapter 5 introduced alternative methods to score words with a focus on learning scores as opposed to calculating them using the PMI formula. First, we explore additional offline filtering methods to perform the same steps of scoring and removing words as we described in Chapter 4. First, we explore supervised filtering that makes use of FastText embeddings (Bojanowski et al., 2017) in conjunction with word-category co-occurrences passed through a dense network with previously calculated PMI scores as the target. Next, we explore unsupervised filtering to determine if using calculated PMI scores is necessary. The unsupervised method also uses the exact embeddings as in the supervised method as inputs for PCA, where the dimensionality is reduced to the number of categories in the dataset. We determine that using PMI scores, either directly to determine non-keywords or indirectly, as in the supervised method, performs best across our datasets. However, unsupervised filtering is still an improvement over the BERT (Devlin et al., 2018) baseline and eliminates the need to calculate PMI scores for every unique word-category pair in the dataset. We also explore an online filtering method that uses a two-pass framework to leverage BERT-produced context-aware embeddings to score words. The purpose of this method was to overcome the reliance on knowledge of word count statistics needed to calculate

PMI values and for supervised offline filtering and remove the offline filtering step entirely. This method does not perform as well as the offline filtering methods and produces similar results to vanilla BERT. Ultimately, we determine that knowledge of word-category co-occurrences is vital to identify non-keywords. We confirm this by performing an ablation experiment with the learned offline word filtering methods.

6.2 Future Work

Future work in this research will include improving the two-pass framework in order to create a robust, high-performing model that is capable of determining the correct words to remove in a live environment. While we have shown that being able to choose which words to remove using offline scoring methods like PMI can improve performance significantly, the two-pass framework performance is not significantly different from the BERT baseline. We believe that a method that doesn't require prior knowledge of statistics, like word counts per category, and is capable of scoring words in an online manner would suit the live nature of online video games and social media better. Since the methods explored in this thesis explicitly remove words or sub-words from an example, we also want to experiment with using word scores to adjust attention values as opposed to hard removal during training.

Ideally, we would want to test any models or framework in a live environment of an online video game or in moderation tools for social media. The end result should be a system that leverages an abusive language detection model to intervene in real time and prevent further toxicity. We want to test a model's capability of being able to act in a lightweight manner without adversely impacting the performance of the application or video game itself.

Another important consideration for future work is to determine if the offline filtering methods also apply well to other facets of abusive language detection, such as determining the target of abuse or identifying the exact spans of a sentence where abusive language occurred. Another aspect that we want to explore is to investigate if the methods described in this thesis can be extended to identifying additional abusive language that was not previously encountered in the dataset.

References

(2021). Hate is no game: Harassment and positive social experiences in online games 2021.

(2022). Patch 12.20 notes.

Babanejad, N., Agrawal, A., An, A., and Papagelis, M. (2020). A comprehensive analysis of preprocessing for word representation learning in affective tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5799–5810, Online. Association for Computational Linguistics.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F. M., Rosso, P., and Sanguinetti, M. (2019). SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Blackburn, J. and Kwak, H. (2014). STFU noob! predicting crowdsourced decisions on toxic behavior in online games. *CoRR*, abs/1404.5905.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information.

Camacho-Collados, J. and Pilehvar, M. T. (2018). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 40–46, Brussels, Belgium. Association for Computational Linguistics.

Canossa, A., Salimov, D., Azadvar, A., Harteveld, C., and Yannakakis, G. (2021). For honor, for toxicity: Detecting toxic behavior through gameplay. 5(CHI PLAY).

Caselli, T., Basile, V., Mitrović, J., and Granitzer, M. (2021). HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Chen, Q., Zhuo, Z., and Wang, W. (2019). BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909.

Chung, Y.-L., Kuzmenko, E., Tekiroglu, S. S., and Guerini, M. (2019). CONAN - COunter NArratives through nichesourcing: a multilingual dataset of responses to fight online hate speech. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2829, Florence, Italy. Association for Computational Linguistics.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. (2020). ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Entertainment, B. (2022). Defense matrix activated! fortifying gameplay integrity and positivity in overwatch 2.

Founta, A.-M., Djouvas, C., Chatzakou, D., Leontiadis, I., Blackburn, J., Stringhini, G., Vakali, A., Sirivianos, M., and Kourtellis, N. (2018). Large scale crowdsourcing and characterization of twitter abusive behavior.

Goo, C.-W., Gao, G.-L., Hsu, Y.-K., Huo, C.-L., Chen, T.-C., Hsu, K.-W., and Chen, Y.-N. V. (2018). Slot-gated modeling for joint slot filling and intent prediction. In *North American Chapter of the Association for Computational Linguistics*.

Hussain, S. F., Babar, H. Z.-U.-D., Khalil, A. H., Jillani, R. M., Hanif, M., and Khurshid, K. (2020). A fast non-redundant feature selection technique for text data. *IEEE Access*, 8:181763–181781.

Inaguma, H., Popuri, S., Kulikov, I., Chen, P.-J., Wang, C., Chung, Y.-A., Tang, Y., Lee, A., Watanabe, S., and Pino, J. (2023). Unity: Two-pass direct speech-to-speech translation with discrete units.

Koloski, B., Pollak, S., Škrlj, B., and Martinc, M. (2022). Extending neural keyword extraction with tf-idf tagset matching.

Kurrek, J., Saleem, H. M., and Ruths, D. (2020). Towards a comprehensive taxonomy and large-scale annotated corpus for online slur usage. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 138–149, Online. Association for Computational Linguistics.

Kurrek, J., Saleem, H. M., and Ruths, D. (2022). Enriching abusive language detection with community context. *ArXiv*, abs/2206.08445.

Kwak, H., Blackburn, J., and Han, S. (2015a). Exploring cyberbullying and other toxic behavior in team competition online games.

Kwak, H., Blackburn, J., and Han, S. (2015b). Exploring cyberbullying and other toxic behavior in team competition online games. 22.

Liu, B. and Lane, I. R. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *CoRR*, abs/1609.01454.

Liu, F., Pennell, D., Liu, F., and Liu, Y. (2009). Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, page 620–628, USA. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Murnion, S., Buchanan, W. J., Smales, A., and Russell, G. (2019). Machine learning and semantic analysis of in-game chat for cyberbullying. *CoRR*, abs/1907.10855.

Märtens, M., Shen, S., Iosup, A., and Kuipers, F. (2015). Toxicity detection in multiplayer online games.

Narang, K. and Brew, C. (2020). Abusive language detection using syntactic dependency graphs. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 44–53, Online. Association for Computational Linguistics.

Nejadgholi, I., Fraser, K., and Kiritchenko, S. (2022). Improving generalizability in implicitly abusive language detection with concept activation vectors. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5517–5529, Dublin, Ireland. Association for Computational Linguistics.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. (2016). Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 145–153, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Rajamanickam, S., Mishra, P., Yannakoudakis, H., and Shutova, E. (2020). Joint modelling of emotion and abusive language detection.

Salton, G. and McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., USA.

Schneider, K.-M. (2005). Weighted average pointwise mutual information for feature selection in text categorization. In *European Conference on Principles of Data Mining and Knowledge Discovery*.

Stoop, W., Kunneman, F., van den Bosch, A., and Miller, B. (2019). Detecting harassment in real-time as conversations develop. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 19–24, Florence, Italy. Association for Computational Linguistics.

Vidgen, B. and Derczynski, L. (2021). Directions in abusive language training data, a systematic review: Garbage in, garbage out. *PLOS ONE*, 15(12):1–32.

Vidgen, B., Hale, S., Guest, E., Margetts, H., Broniatowski, D., Waseem, Z., Botelho, A., Hall, M., and Tromble, R. (2020a). Detecting East Asian prejudice on social media. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 162–172, Online. Association for Computational Linguistics.

Vidgen, B., Harris, A., Nguyen, D., Tromble, R., Hale, S., and Margetts, H. (2019). Challenges and frontiers in abusive content detection. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 80–93, Florence, Italy. Association for Computational Linguistics.

Vidgen, B., Thrush, T., Waseem, Z., and Kiela, D. (2020b). Learning from the worst: Dynamically generated datasets to improve online hate detection. *CoRR*, abs/2012.15761.

Wang, Y., Shen, Y., and Jin, H. (2018). A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 309–314, New Orleans, Louisiana. Association for Computational Linguistics.

Waseem, Z. and Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Weld, H., Huang, G., Lee, J., Zhang, T., Wang, K., Guo, X., Long, S., Poon, J., and Han, S. C. (2021). Conda: a contextual dual-annotated dataset for in-game toxicity understanding and detection.

Wulczyn, E., Thain, N., and Dixon, L. (2017). Ex machina: Personal attacks seen at scale.

Zhang, C., Li, Y., Du, N., Fan, W., and Yu, P. (2019). Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5259–5267, Florence, Italy. Association for Computational Linguistics.

Zhang, Z., Robinson, D., and Tepper, J. A. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. In *Extended Semantic Web Conference*.

CURRICULUM VITAE

[redacted]

[redacted]