Optimized Large Language Model for Hate Speech Detection

A thesis submitted to the Committee on Graduate Studies

in Partial Fulfillment of the Requirements for the Degree of Master of Science

in the Faculty of Arts and Science

Trent University

Peterborough, Ontario, Canada

Applied Modelling and Quantitative Methods M.Sc. Graduate Program

September 2025

Abstract

Optimized Large Language Model for Hate Speech Detection

Uchechukwu Emmanuel Obinwanne

Recent developments in Artificial Intelligence (AI), particularly Large Language Models (LLMs), have provided powerful tools for Natural Language Processing (NLP) tasks like sentiment analysis. However, their fine-tuning and deployment present challenges, specifically in terms of computational efficiency and high training costs. To address these challenges, this work applies optimization techniques such as Quantized Low-Rank Adaptation (QLoRA) for parameter-efficient fine-tuning, followed by Generalized Post-Training Quantization (GPTQ) on the Llama 3.1 LLM. To evaluate these optimizations, we apply the model to a practical task: hate speech detection, using a curated dataset comprising of X (formerly Twitter) posts. Overall, the optimized model achieved a 67% reduction in size along with significant improvements in classification accuracy and inference speed compared to the base model.

Keywords: Large Language Models, Parameter Efficient Fine-Tuning, Low-Rank Adaptation Quantized Low-Rank Adaptation (QLoRA), Generalized Post-Training Quantization (GPTQ).

Acknowledgements

I would like to give a big thanks to my MSc. supervisor, Dr. Wenying Feng for all her support, encouragement and guidance throughout my 2 years at Trent University. Without her, this project would not have come to fruition. For this, I am eternally grateful!

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

BERT Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

DT Decision Tree

FP Floating-Point Data Type

GPT Generative Pre-trained Transformer

GPTQ Generalized Post-Training Quantization

GPU Graphics Processing Unit

INT Integer Data Type

LLM Large Language Model

LR Logistic Regression

LoRA Low-Rank Adaptation

LSTM Long-Short Term Memory

NLP Natural Language Processing

PEFT Parameter Efficient Fine-Tuning

PTQ Post-Training Quantization

QLoRA Quantized Low-Rank Adaptation

RAG Retrieval Augmented Generation

RF Random Forest

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Problem Overview and Motivation

The importance of social media moderation cannot be overstated. While social media offers individuals the opportunity to express themselves freely, this freedom and right are often misused, providing a platform to incite hate against individuals and communities. Harmful manifestations of this misuse include cyberbullying, racism, sexism, and radicalization. These pervasive issues have been shown to have detrimental psychological and even physical effects on individuals, including anxiety, depression, and even suicide [44, 48, 109, 178]. There have also been instances where hate crime suspects have a history of hateful social media posts [131, 175, 207]. These online posts often show patterns of hostility and intolerance towards specific groups, leading to real acts of violence. According to the U.S. Government Accountability Office (GAO), "Extremist attacks ... illustrate how exposure to hate speech online may have contributed to the attackers' biases against people based on race, national origin, and sexual orientation." [155].

The number of social media users and the online presence of the average individual have grown exponentially in the past two decades. This growth comes with the increased challenge of content moderation due to the volume of multimodal (text, video, audio etc.) content being posted and the evolving nature of language. With each generation ushering in new slang, abbreviations, subtleties, and deliberate obfuscation of profanity, it has become increasingly challenging to keep up with this evolution. Furthermore, the inconsistencies in defining what constitutes hate speech add another layer of complexity to this problem.

While social media platforms have different guardrails in place to prevent the propagation of hate speech, these typically require a significant amount of manpower to review reports of hate speech [131] and consequently take a mental toll on the reviewers [5]. This process can be expedited using automated hate speech detection approaches. However, these approaches are not foolproof. Traditional machine learning moderation systems, which typically include algorithms like Support Vector Machines (SVM), decision trees, logistic regression, or basic neural networks, rely on handcrafted features and supervised learning from labelled datasets. These systems often struggle to detect hateful language, especially when it is intentionally obfuscated or uses slang for which they have not been trained. Additionally, there is also the fact that offensive language often overlaps with hate speech, which can lead to misclassification. This complexity necessitates nuanced classification approaches to distinguish between hate speech and other forms of offensive language effectively.

LLMs offer advanced capabilities that can aid in this nuanced task. Even so, the base versions of these models often perform poorly in domain-specific tasks, including

hate speech detection and classification. Nevertheless, their performance can be significantly improved with domain-adaptive pre-training, where the model's language representations are further adjusted to understand the nuances of a new domain. This has yielded impressive results as seen in the following literature [40,63,85,111,115,118].

Fine-tuning is another method used to enhance the performance of an LLM, and it is one of the key areas this thesis addresses. With fine-tuning, the LLM is tailored to specific objectives, such as classification, by training it on labelled datasets. This enables the LLM to adapt and learn patterns pertinent to the task, such as distinguishing between hate speech and offensive or neutral language. Unlike domain-adaptive pre-training, which enhances the LLM's general language understanding within a defined domain, fine-tuning sharpens its performance on precise tasks by adjusting its parameters to optimize for accuracy.

Overall, this thesis examines how targeted fine-tuning can enhance the detection capabilities of LLMs while also addressing concerns associated with their use, such as computational limitations, through the application of optimization techniques. By contributing to the development of efficient models, this work supports real-world applications in content moderation and the ongoing exploration of refined methods for nuanced language processing in online contexts.

1.2 Research Objectives and Thesis Contributions

Our literature review has found that many previous studies on hate speech detection have focused on using traditional machine learning algorithms, such as Naive Bayes (NB), Support Vector Machines (SVM), and Logistic Regression (LR), among others. More recent work has been found to utilize deep learning methods, such as

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). With the rapid adoption of transformer models, more research has shifted to encoder-only models, such as BERT (Bidirectional Encoder Representations from Transformers), which is widely used for classification tasks, including hate speech detection.

However, relatively less attention has been given to benchmarking the performance of decoder-only transformer models, especially Meta's Llama 3, on this task. Furthermore, existing studies rarely explore the joint benefit of parameter-efficient fine-tuning and quantization for the llama models. The silver lining is that this thesis aims to address these research gaps. Specifically, the objectives and contributions of this work are:

- To fine-tune Meta's pre-trained Llama 3.1 model on a custom hate speech dataset consisting of X social media posts, using Quantized Low-Rank Adaptation (QLoRA). This serves to demonstrate the potential of Parameter-Efficient Fine-Tuning (PEFT) as an LLM fine-tuning technique.

- To optimize the fine-tuned LLM using Post-Training Quantization (PTQ), reducing its memory footprint and improving inference speed while retaining accuracy. This will help showcase the feasibility of deploying a similar model in real-world situations and environments with limited resources.

- To evaluate the optimized model's capability of distinguishing hate speech from offensive and neutral language.

1.3 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 – Background: This chapter focuses on the foundational concepts of machine learning and their relationship to sentiment analysis and hate speech detection. The chapter will also introduce transformer models and LLMs and feature a literature review of related work.

Chapter 3 – Optimization of Large Language Models for Text-Based Applications: This chapter further explores the transformer architecture and LLMs, discussing optimization strategies such as fine-tuning and quantization. The chapter concludes with an overview of the proposed architecture based on Llama 3.1.

Chapter 4 – Model Implementation for Hate Speech Classification: This chapter outlines the methodology employed in fine-tuning and optimizing Llama 3.1, from data acquisition to quantization and model evaluation.

Chapter 5 – Results and Discussion: We analyze the results of the study, comparing the model performance before and after fine-tuning and quantization. We evaluate metrics such as accuracy and F1 scores for assessment.

Chapter 6 – Conclusion: The final chapter summarizes the key contributions of the thesis, discusses the limitations of the proposed approach, and proposes future research directions.

Chapter 2

Background

Sentiment analysis, a prominent application of Artificial Intelligence (AI) within the field of Natural Language Processing (NLP), has transformed the way emotions are interpreted and understood from textual data. Sentiment analysis is an NLP task used to classify the sentiments and opinions expressed within a body of text regarding a topic as positive, negative, or neutral. The popularity of sentiment analysis has increased [203] with the rise of social media platforms, accompanied by the growing volume of data generated from online reviews and discussions. This surge in data provides a unique opportunity for organizations, businesses, and researchers to gain insight into consumer behaviour, public opinion, and market trends. There have also been significant advancements in deep learning models, such as transformer models, which have improved the accuracy of sentiment analysis. These advancements have made sentiment analysis a vital tool for informed decision-making and effective planning.

While sentiment analysis has been shown to have multifaceted applications in finance [110, 203], business [36, 186], healthcare [143] and governance and politics

[79, 230], its utility extends beyond general sentiment classification to addressing specific societal challenges such as hate speech detection. Hate speech, a subset of offensive language characterized by its discriminatory nature, poses significant risks to social harmony and individual well-being. AI-driven solutions, particularly those that leverage sentiment analysis techniques, are crucial for identifying and mitigating such content on online platforms.

The need for precise hate speech detection is evident in our current era of widespread digital communication, where misinformation and harmful discourse spread like wildfires. Building on the methodologies and advancements in sentiment analysis, hate speech detection incorporates additional layers of complexity, such as distinguishing between hate speech, offensive language, and neutral content. This distinction is essential for promoting online safety while preserving freedom of expression.

The integration of Large Language Models (LLMs) has further advanced hate speech detection by enabling nuanced understanding and classification of text. Unlike traditional machine learning models, LLMs such as BERT, GPT, and their fine-tuned variants excel at capturing contextual and semantic subtleties, making them well-suited for this task. This thesis will leverage such models to develop a robust hate speech detection system capable of fine-grained classification, addressing the gaps in current methodologies and enhancing the efficacy of automated moderation systems. However, to fully contextualize the integration of LLMs, it is worthwhile to review their foundations in Artificial Intelligence (AI) and Machine Learning (ML).

AI has become a buzzword in recent times, impacting various aspects of our daily lives. As with sentiment analysis, AI has played a crucial role in healthcare [54, 71, 84], education [41, 176], and engineering [152]. Rooted in disciplines such as philosophy,

mathematics, and neuroscience, AI refers to the creation of machines capable of performing tasks that require human-like cognitive functions—including reasoning, learning, and decision-making [134, 179, 242]. Under the AI umbrella lie various subfields, including NLP, computer vision, ML, and Deep Learning (DL). ML, in particular, serves as the backbone of AI advancements, offering algorithms that can learn from data and improve over time. Its subset, DL, has further pushed the boundaries by introducing neural networks capable of processing large datasets with remarkable efficiency. Among the most significant outcomes of these advancements are LLMs, which build upon ML and DL principles to achieve unprecedented performance in tasks requiring language understanding and generation.

The following sections provide a high-level overview of ML, its principles and applications in sentiment analysis, and its role in the development of LLMs.

2.1 Overview of Machine Learning

Figure 2.1 shows the relationship of LLMs to DL, ML, and AI, all of which will be discussed in this section.

Machine learning can be defined as the process of teaching a machine or program using a set of algorithms and statistical models, thereby enabling it to perform tasks such as making predictions and decisions based on the data it was trained on. In other words, it is the act of training a machine to recognize and extract patterns from past data. Afterwards, the machine is able to apply what has been learned to solve problems in new contexts [76, 196].

(Machine) learning can be divided into three main types [179]:

1. Unsupervised learning

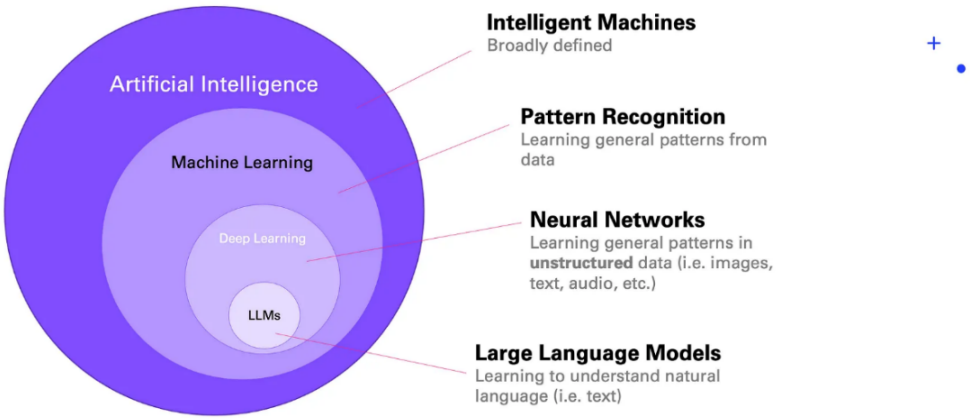

Figure 2.1: Domains in AI [199]

2. Supervised learning

3. Reinforcement learning

2.1.1 Unsupervised Learning

Unsupervised learning uses unlabeled data to train a model without explicit supervision. These algorithms infer structure or patterns in the data, often identifying hidden relationships [76, 216]. Common unsupervised learning techniques include clustering and dimensionality reduction [95].

Applications include anomaly detection in network traffic [193,202], user clustering for targeted advertising [69, 149], and natural language processing for topic modeling and document clustering [105, 114, 214].

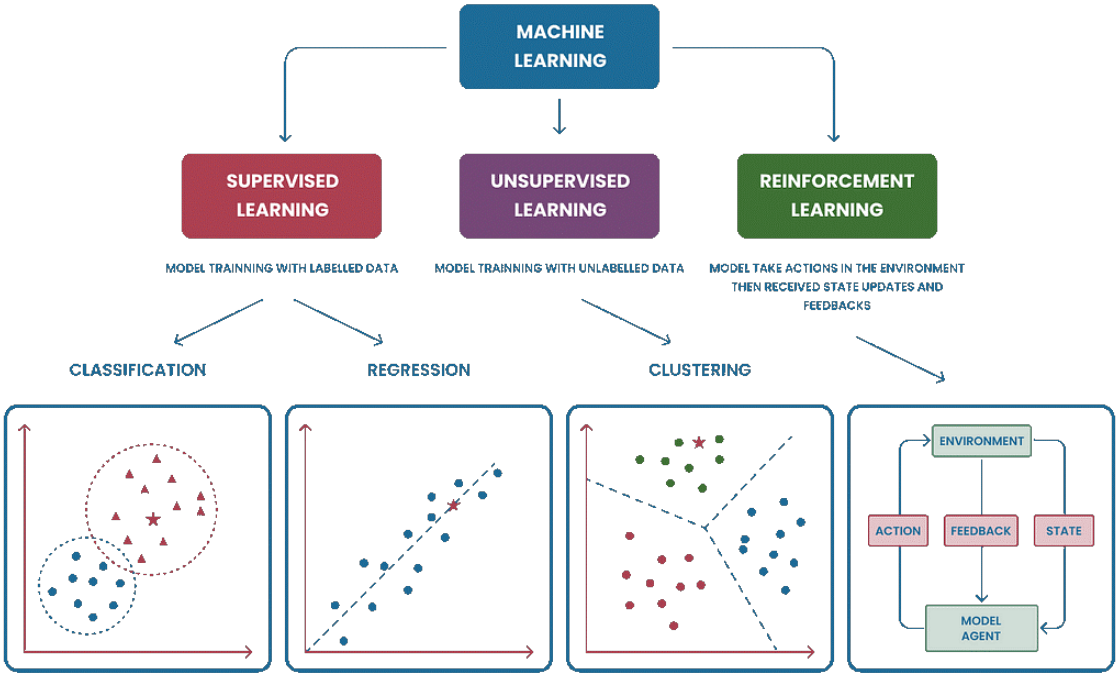

Figure 2.2: Overview of Machine Learning [7]

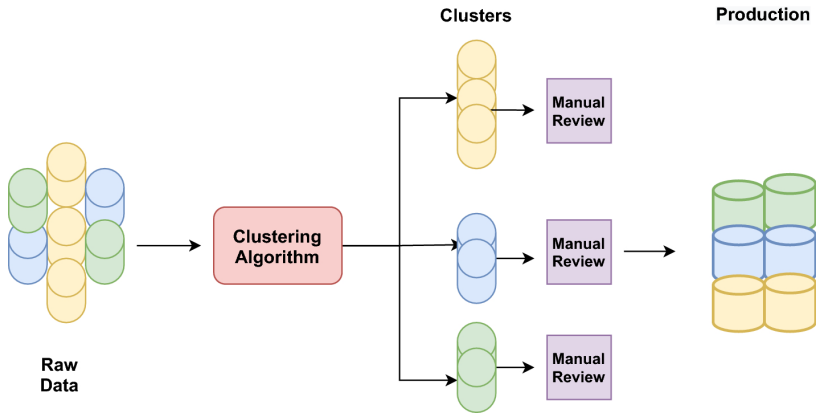

Figure 2.3: Unsupervised Learning Process [218]

2.1.2 Supervised Learning

Supervised learning relies on labeled training data to learn relationships between inputs and outputs [76].

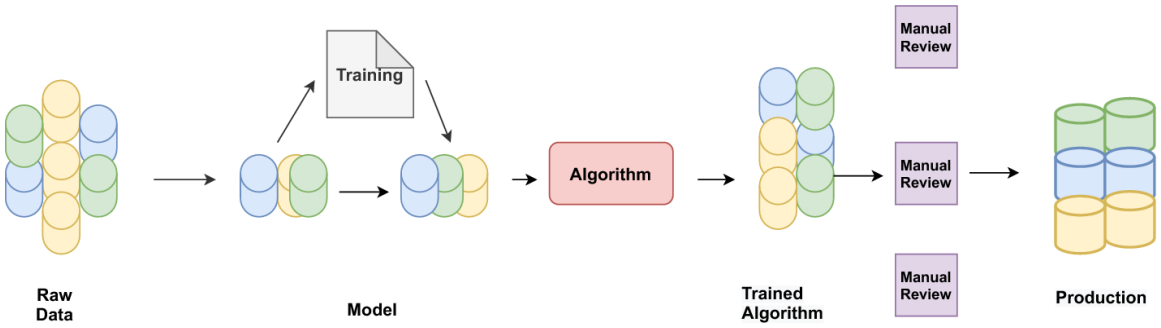

Figure 2.4: Supervised Learning Process [218]

Two common types are:

- **Regression:** Predicts continuous values (e.g., house prices, temperatures). Algorithms include linear regression, polynomial regression, decision trees, and neural networks.
- **Classification:** Assigns categories (e.g., spam detection). Algorithms include logistic regression, SVMs, decision trees, and random forests.

Table 2.1: Key Differences between Regression and Classification

Regression	**Classification**
The dependent variable is continuous	The dependent variable is categorical
Predicts a quantity (continuous value)	Predicts a class label (categorical value)
Evaluation: MSE, RMSE, R^2	Evaluation: F1-score, accuracy, sensitivity
No class boundaries	Decision boundary between classes

Applications include cancer detection [204], predicting academic performance [98], and water quality assessment [9].

2.1.3 Reinforcement Learning

In reinforcement learning, models learn through trial and error by receiving rewards or penalties based on their actions [211]. For example, in a chess game, winning yields rewards while losing yields penalties. Over time, models optimize their actions [179, 216]. A key application is robotics control [198].

2.1.4 Semi-Supervised Learning

Semi-supervised learning combines labeled and unlabeled data, leveraging small labeled datasets to make sense of larger unlabeled ones. It shares many applications with supervised learning [76].

2.2 Applications of Machine Learning in Sentiment Analysis

As seen above, classification aims to develop predictive models that can accurately assign class labels to new, unseen data points. This task is highly applicable to sentiment analysis, which involves categorizing opinions from textual data. In fact, sentiment analysis can be seen as a classification problem. The following section provides an overview of the levels, tasks, and techniques of sentiment analysis, with an emphasis on machine learning approaches.

2.2.1 Levels of Sentiment Analysis

Sentiments can be investigated on three main levels: document level, sentence level, and aspect level [29, 35, 64, 181, 240].

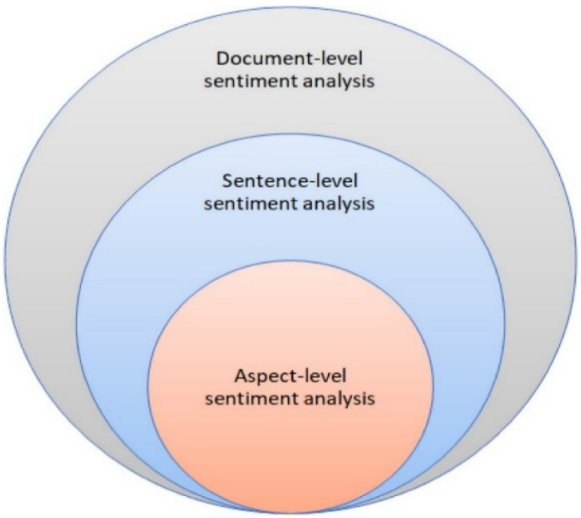

Figure 2.5: Levels of Sentiment Analysis [35]

Document-Level: At this level, the overall sentiment expressed by a single entity in a document is classified, thereby providing a high-level overview of the document.

Sentence-Level: This level involves the evaluation of each sentence within a body of text. This entails two tasks: first, each sentence has to be analyzed for an expressed opinion. If one is detected, then the emotional polarity is examined to determine the conveyed sentiment. This level serves to distinguish between objective and subjective sentences, where objective sentences express factual information and subjective sentences entail personal views and opinions [29, 35].

Aspect-Level: This involves analyzing specific features of a sentence and determining the sentiment expressed through those features [227]. Consider the following

statement: *"My phone's speaker is awful. However, the display's great."* In this example, there are two aspects – the quality of the phone's speaker and the display. The goal of aspect-based sentiment analysis here is to isolate each aspect and predict their sentiment.

Sentiment Analysis Approaches

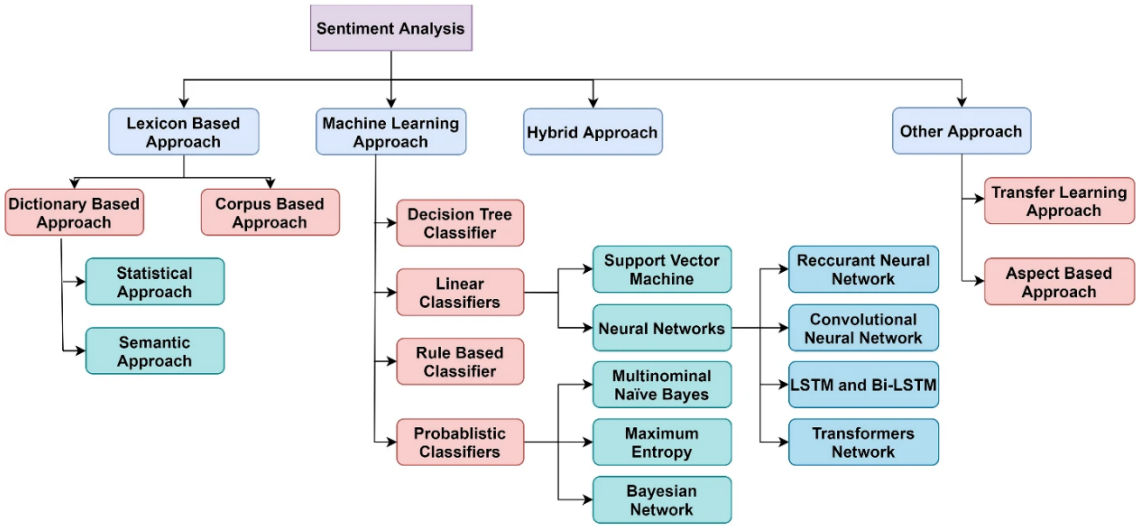

Figure 2.6: Overview of Sentiment Analysis Approaches [218]

Figure 2.6 provides a comprehensive overview of the various approaches for sentiment analysis, categorizing them into four main groups: Lexicon-Based Approach, Machine Learning Approach, Hybrid Approach, and Other Approaches.

i. Lexicon-Based Approach: This technique involves the use of a lexicon, which consists of predefined words assigned a sentiment score based on their polarity. Polarity indicates the sentiment "conveyed by a particular text, phrase or word" [3]. For example, a score ranging from [+1, -1], where +1 is very positive, 0 – neutral, and

-1 – very negative. The text to be analyzed is broken down into tokens, where each token represents a single word, and a sentiment score is assigned to each token from the lexicon. Afterwards, the sentiment scores of each token (or word) are aggregated to give the overall sentiment of the text [35, 106, 218].

"The most commonly used lexical method for sentiment analysis of English data is Valence Aware Dictionary for Sentiment Reasoning (VADER)" [17, 94]. While this method is easy to implement since it does not require any complex training data, it fails to capture the context of a word. Additionally, a word can have different meanings depending on the domain in which it is used [139]. These issues can lead to incorrect sentiment classification and analysis. Lexicon-based approaches can be corpus-based or dictionary-based.

The **corpus-based approach** involves using a predefined set of sentiment words and expanding it by analyzing a large corpus of text to find patterns in how words are used together or in syntax [218]. For example, words joined by "and" usually share the same sentiment. In contrast, words joined by "but" are often opposites [35]. Additionally, modifiers like "very" and "slightly" also affect the sentiment of a word. The corpus-based approach can be divided into two categories: statistical and semantic approaches.

The **dictionary-based approach**, on the other hand, involves the use and storage of a list of words according to their polarity and sentiment. The idea behind this method is that words are manually collected and expanded using their synonyms and antonyms, which are assumed to have the same and opposite polarities, respectively. The dictionary approach can be further divided into statistical and semantic approaches.

The **statistical approach** follows the idea that if a word with an unknown sentiment frequently appears with words whose sentiments are known in similar contexts, then the unknown word is likely to express the same sentiment as the known words [35]. This method does not directly consider the meaning of the words but focuses on their linguistic patterns.

Unlike the statistical approach, which is based on the patterns of occurrence, the **semantic approach** focuses on the relationships between words based on their semantic properties. In this method, words that are semantically close, such as synonyms, are assigned similar values.

ii. Machine Learning Approach: The machine learning approach to sentiment analysis involves the use of machine learning algorithms to sort textual data by polarity. These algorithms learn to recognize patterns in the training data and can classify texts into categories such as positive, negative, neutral, happy, sad, and others. This is a powerful approach to sentiment classification as the algorithms can learn domain-specific patterns from the training data. However, this often requires large amounts of data to achieve.

The different types of machine learning discussed earlier are equally relevant and applicable to this approach. [194] uses a supervised learning approach to classify Arabic tweets by their sentiment polarity (positive, negative, and neutral). [121] uses unsupervised learning to develop a framework for word embeddings using prior sentiment knowledge. [177] uses auto-encoders to address high-dimensionality in textual data. Lastly, [93] develops a novel semi-supervised learning model for emotion recognition.

As previously discussed in the machine learning section of this thesis, this approach

is also divided into unsupervised, supervised, and reinforcement learning approaches. However, to remain aligned with the scope of this thesis, only the supervised approach will be discussed in further detail.

Supervised learning relies on labelled training data. In this case, the training data is labelled with the sentiments expressed by the texts. The supervised learning approach can be performed using linear, decision tree, probabilistic or rule-based classifiers [185, 218, 231].

Probabilistic classifiers predict the probability distribution of all possible classes rather than directly assigning class labels using their input features [35]. Common probabilistic classifiers are Naïve Bayes and Maximum Entropy classifiers used by both [215] and [74] respectively.

Decision trees in text classification work by recursively splitting data into smaller subsets based on specific criteria applied to the features (for instance, the presence of certain words) until the text can be classified. They involve a series of yes/no questions to classify data and can be combined to make up a Random Forest (RF) to avoid overfitting [218]. [28] uses the C5.0 and CART (Classification and Regression Trees) decision tree algorithms for analyzing the Initial Public Offerings (IPOs) of a company.

Lastly, rule-based classifiers use IF-THEN rules to predict the classes of new records. In text classification, a rule can be: IF the word "sad" is found in a text, THEN classify the text as "negative." This approach is similar to decision trees; however, rule-based classifiers are less prone to overfitting but have poor scalability.

Linear classifiers employ straight-line or hyperplane boundaries to categorize text into distinct categories. A straight line is used for binary classification; otherwise –

a hyperplane [35]. Linear classifiers used for sentiment analysis and classification are Support Vector Machines (SVM) [11] and Artificial Neural Networks (ANN) [42].

SVM aims to find the optimal hyperplane that maximizes the margin between classes, reducing classification errors. SVM is particularly effective in text classification tasks, which is demonstrated in various studies. For example, [169] showed that SVM outperformed other methods, such as Naïve Bayes, in analyzing movie reviews. [11] combines SVM and RF, enhancing classification performance.

Neural networks "are the backbone of deep learning algorithms" [53]. As previously mentioned, deep learning is a subset of machine learning that rose to prominence due to the work of Hinton et al. [89] in 2006. It has since become a popular topic in the data-driven industry due to its efficacy, efficiency, and scalability in solving regression and classification problems typically tackled by machine learning algorithms. However, despite its advantages, deep learning suffers in terms of training speed due to the sheer number of training parameters and layers [187].

Inspired by the brain, neural networks aim to emulate the workings of the network of brain cells (neurons). The "workings" of brain cells here refer to how neurons in the brain process data. Neural networks, often used interchangeably with Artificial Neural Networks (ANNs), consist of an input layer, hidden layers, and an output layer, with each node/neuron containing n weights [179].

Deep Neural Networks (ANNs with multiple layers), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) are the most used ANNs in sentiment analysis [35]. RNNs, such as the Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM), are also popular. [119] uses a Bi-LSTM to better

capture important sentiment information in the text and reduce the reliance on manually created sentiment dictionaries or lexicons. [107] uses CNNs for sentence-level classification. Lastly, [125] uses traditional RNNs for improved text classification performance.

iii. Hybrid Approach: The hybrid approach in sentiment analysis combines the lexicon-based and machine-learning approaches discussed above. It leverages the ability of machine learning to adapt to complex patterns and the reliability of lexicon-based techniques in identifying sentiments using predefined sentiment lexicons. In other words, the lexicon-based technique assigns polarity to individual words, while machine learning algorithms handle context and learn from the data. Studies from [11, 86] demonstrated that hybrid models using machine learning algorithms, such as SVM and RF, outperformed individual models in terms of accuracy, highlighting the effectiveness of this method.

iv. Other Approaches: There are other approaches able to handle the complexities of language, one of which is Aspect-Based Sentiment Analysis (ABSA). ABSA is an approach to sentiment analysis that focuses on identifying the sentiment expressed about specific aspects or features of a target, such as products or services, within a text. This method goes beyond overall sentiment classification by breaking down the text into distinct aspects and evaluating the sentiment polarity (positive, negative, or neutral) associated with each one.

It is executed in two stages. First, the aspects of the sentence are identified and extracted. For example, the words, "service" and "bed" could be extracted from a hotel review. Next, the sentiments associated with the identified aspect are

determined. In a review like, "The beds in the hotel were sturdy, however, the customer service was horrendous," "beds" is linked to a positive sentiment, whereas "service" is associated with a negative sentiment. RNNs such as LSTM, Bi-LSTM, and transformer-based models like BERT and GPT are often used to handle the complexities of aspect extraction and sentiment classification [218]. After identifying and classifying sentiments for each aspect, an aggregation step is performed to assess the overall sentiment of the text. [209] uses ABSA to measure the sentiments of hotel reviews.

Another approach is transfer learning. This technique enables a model trained in one domain to apply its knowledge to a different yet related domain. With this approach, the model leverages the similarity between the data and tasks shared by both domains. Consequently, this significantly reduces training time, as the model is not trained from scratch but rather fine-tuned on the new data. [137] trained a CNN model on a source domain dataset, and its learned features were transferred to a target domain with limited or no labelled data. Similarly, [27] applied transfer learning to evaluate sentiment in the Polish language by training a model on one dataset and validating it on another, showcasing its versatility across different languages and domains.

Given its versatility and efficiency, the transfer learning approach is used extensively in this thesis to fine-tune pre-trained models for hate speech detection.

2.3 Role of Sentiment Analysis and Large Language Models in Hate Speech Detection

Sentiment analysis is crucial in detecting hate speech and offensive language, as it aids in capturing the polarity of a statement, which helps identify harmful or abusive communication. Hate speech naturally expresses negative sentiments such as hostility, anger, or contempt, making sentiment analysis a valuable tool in understanding and classifying the emotional content of such language [190].

Building on this, several studies [39, 62, 81, 197] have adopted a multi-step framework, where a sentiment classifier first identifies negative sentiment before a hate speech classifier examines the content for harmful language.

Hate speech detection can be addressed using the Aspect-Based Sentiment Analysis (ABSA) approach [154, 232], which involves a fine-grained level of sentiment analysis that focuses on sentiments towards specific aspects or targets (in this case, individuals, religions, ethnicities, etc.) within texts. Sentiment analysis on its own might not be sufficient to detect hate speech comprehensively; however, it provides contextual information that can be integrated into complex models, such as LLMs, through techniques like transfer learning. With transfer learning, we can fine-tune a pre-trained LLM on datasets annotated for hate speech and offensive language detection.

LLMs have been trained on massive amounts of data and have demonstrated their complex reasoning [223], in-context learning, and decision-making [146] skills with human-level performance. As a result, these models are able to perform a wide range of NLP tasks [229], including summarization, sentiment analysis, classification,

and sentence translation [146]. With the incorporation of sentiment data in the fine-tuning (training) process of LLMs, they can become more proficient at identifying even subtler forms of hate speech that might be hidden under the guise of sarcasm or euphemisms.

Moreover, transfer learning helps integrate ABSA into pre-trained LLMs, allowing it to distinguish between general sentiment and targeted negative sentiment towards specific groups. This integration ensures a more nuanced understanding of the content, improving the accuracy of hate speech detection systems. Therefore, the combination of sentiment analysis and the advanced capabilities of LLMs represents a significant advancement in addressing the challenges associated with identifying and mitigating hate speech.

2.4 Challenges of Hate Speech Detection

Hate speech detection is inherently challenging because of the lack of a concrete and universally accepted definition of what constitutes hate speech [8]. Furthermore, the different interpretations of hate speech often overlap with offensive, abusive, and toxic language, which can be highly subjective, making it difficult for humans and machines to distinguish [162]. While there is a growing need for automated hate speech detection tools at a large scale among governments and companies, the quality of the data required for training these tools also suffers due to the subjective nature of hate speech.

A survey by [162] compared the various definitions of hate speech found in existing literature. In this survey, Schmidt and Wiegand [190] defined hate speech as the "act of offending, insulting or threatening a person or a group of similar people

on the basis of religion, race, caste, sexual orientation, gender or belongingness to a specific stereotyped community." Waseem and Hovy [220] adopt a close definition – "Use of a sexist or racial slur, attack a minority, promotes hate speech or violent crime, blatantly misrepresents truth or seeks to distort views on a minority with unfounded claims, shows support of problematic hashtags (e.g., #BanIslam, #whoriental, #whitegenocide), defends xenophobia or sexism, or contains a screen name that is offensive." Warner and Hirschberg [219] adopt Nockleby's definition [151] of hate speech as "any communication that disparages a person or a group on the basis of some characteristic such as race, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic." Lastly, Davidson et al. [56] define hate speech as "language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group."

Schmidt and Wiegand's definition lays emphasis on offences, threats, and insults to others' religion, race, and gender, but does not concretely state what constitutes an insult or offence, as this varies depending on personal, cultural, and even societal beliefs. It also fails to mention other forms of prejudice, such as those based on nationality and ethnicity. Waseem and Hovy's definition is more specific but adds ambiguity to the definition of hate speech. For instance, the statements "blatantly misrepresents truth" or "problematic hashtags" are subjective and may not necessarily encompass hate speech (like #MAGA), making this definition harder to apply uniformly. Furthermore, the phrase "attack a minority" is context-dependent and varies from country to country. While people of colour are seen as minorities in the United States, they are not minorities in African or Asian countries. If individuals considered as "minorities" make a hateful statement toward other (non)minorities,

are they exempt from being perpetrators of hate speech?

Nockleby's definition is more general and focuses on disparagement based on a person's or group's characteristics, like race or ethnicity. Even so, it does not explain what constitutes "disparagement." Lastly, Davidson et al. add another layer to the definition of hate speech by introducing "intent" to humiliate a group. However, determining intent is a subjective process that heavily depends on the context.

While the definitions above explain what constitutes hate speech, hate speech can sometimes overlap with offensive language, adding to the difficulty of distinguishing them. Even though hate speech and offensive language can manifest similarly, they are not always the same. Consider the following statement: "You're too emotional to do this." This statement could be perceived as a personal attack or interpreted as sexist if directed at a woman, given the stereotype that women are more emotional than men [37]. Despite its offensive or potentially sexist nature, such a statement is typically not categorized as hate speech, as it does not incite hatred against women as a group. However, the statement: "black people are always late" can be classified as hate speech as it makes use of a harmful stereotype to degrade an entire ethnic group. Depending on the annotator, this could also be classified as offensive or even hateful.

Interestingly, the interpretation of potentially offensive or hateful statements can shift significantly depending on context. For instance, consider the statement about Black people being "always late." What if the statement was made by a Black person to another Black person – is this still considered offensive or hate speech? Similarly, the use of the N-word presents another complex example. When used among Black individuals, it may serve as a term of camaraderie or endearment. In contrast, a

non-Black person may use this term ignorantly to show the same endearment or even as a joke, yet the impact and reception can differ drastically. At the other end of the spectrum, it could be used hatefully with the intent to harm. This underscores the significance of intent and context in classifying hate speech and offensive language.

For the purpose of this thesis, we adopt the United Nations' explanation of hate speech. Hate speech is "any kind of communication in speech, writing or behaviour, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, colour, descent, gender or other identity factor" [145]. While the UN's definition of hate speech may not address nuances such as tone and intent, it is comprehensive and globally inclusive, emphasizing human dignity without anchoring itself in jurisdiction-specific legal language. Notably, it avoids potentially limiting terms like "minority," instead focusing on identity-based targeting. Lastly, it recognizes that hate speech extends beyond verbal expression to include written and behavioural forms of communication.

2.5 Related Work

Several sentiment analysis approaches have been employed in hate speech detection, especially machine learning and transfer learning approaches. This is due to their ability to capture subtleties in language. They are also more robust to misclassifications due to word ambiguity faced by word-based approaches [161]. NLP methods, such as Bag-of-Words (BoW), Term Frequency-Inverse Document (TF-IDF), N-grams, and Part-of-Speech (POS) tagging, have difficulties when words are masked (intentionally or unintentionally) with incorrect spellings [161]. BoW, in particular, leads to

many false positives (for hate speech) due to the presence of offensive language [56]. Therefore, reliance on these techniques can lead to the limitation of free speech due to word ambiguity.

Machine learning approaches have yielded mixed results in hate speech detection tasks. [213] found little to no difference in the classification performance of LSTM and SVM on hate speech detection datasets with both methods yielding accuracies of approximately 60%. [56] adopts different algorithms (NB, LR, DT, etc.) to create a multiclass classifier that differentiates hate speech from offensive language and neutral language. However, 40% of texts labelled as hate speech are misclassified, highlighting the challenge of achieving high precision.

Other research has focused on semi-supervised and unsupervised methods. For instance, [225] uses Latent Dirichlet Allocation (LDA) [123], a semi-supervised learning approach, to identify hate speech in tweets with their best-performing model having an F1 score of 71%. In a different approach, [61] proposes Growing Hierarchical Self-Organizing Maps (SOMs), an unsupervised approach, to detect cyberbullying in social networks. On a YouTube dataset, the model achieves an accuracy and F1 score of 0.69 and 0.74, respectively. However, the same model performed worse on a Twitter dataset, achieving an F1 score of 0.4.

Another study [99] addresses hate speech using clustering to categorize tweets into hate speech, offensive content, and clean speech. To achieve this, they apply a weighted pattern-matching technique and a priority-based dictionary tagging system to classify tweets. This results in an impressive average accuracy of 97% on their test data. [221] extracts unigrams and patterns from a training set and employs them for training a machine learning model which performs multi-class classification

(hate, offensive, neither) with an accuracy of 78.4%. Lastly, [174] takes a unique angle, shifting the focus of hate speech detection from the content of posts to the users, noting the distinct social media activity patterns of hateful users. The study uses gradient-boosted trees to differentiate hateful users from normal ones, with an accuracy of 84% and 79% for suspended and active users.

While these machine-learning approaches have their issues, they are still widely used, mainly because they are much less resource-intensive to implement than transformer models [122]. More examples are displayed in Table 2.2 below.

Table 2.2: Additional Studies Incorporating Machine Learning for Hate Speech Detection

Task Description	**NB**	**LR**	**SVM**	**DT**	**RF**
Hate speech and abusive language detection in Albanian [153]	F1: 0.41	-	-	-	-
Multimodal hate speech and offensive language detection in memes [201]	F1: 0.49	F1: 0.48	-	-	-
Multi-class classification – hate speech, offensive language, neither [56]	-	F1: 0.9	-	-	-
Chinese sexism detection [104]	-	F1: 0.74; Acc: 0.79	F1: 0.74; Acc: 0.78	-	-
Abusive language detection towards conversational AI systems [50]	-	-	F1: 0.84	-	-

Continued on next page

Table 2.2 – continued from previous page

Task Description	**NB**	**LR**	**SVM**	**DT**	**RF**
Offensive language detection, categorization and target identification [233]	-	-	F1: 0.45	-	-
Binary classification – hate speech or not [80]	-	-	Acc: 0.74	-	-
Abusive language prediction [157]	-	F1: 0.54; Acc: 0.67	-	-	F1: 0.40; Acc: 0.62
Hostility detection in Hindi [31]	-	F1: 0.84	F1: 0.84	-	F1: 0.8
Fine-grained hate speech and abusive language detection on Indonesian Twitter (X) [96]	-	-	-	Acc: 0.77	-

Continued on next page

Table 2.2 – continued from previous page

Task Description	**NB**	**LR**	**SVM**	**DT**	**RF**
Hate speech detection; Identification of hate targets [184]	-	F1: 0.78	F1: 0.79	F1: 0.77	F1: 0.66

Deep learning techniques have been instrumental in advancing hate speech detection by using large-scale data and sophisticated models. These models have outperformed classical machine learning techniques in several studies [12, 24, 65, 101]. As a result, several works have employed deep learning algorithms such as LSTM, Bi-LSTM, CNN, and RNN [16, 68, 80, 136, 136, 142, 153, 161, 189, 195] in hate speech detection, some of which are seen in Table 2.2. However, the advent of transformer-based models has revolutionized the landscape.

Transformer-based models, such as BERT [59], GPT [165], Llama [208], and RoBERTa [127], have set new benchmarks in various NLP tasks. They can process sentences with attention to all words in the context, enabling a deeper understanding of the language. However, they yield unsatisfactory results in domain-specific use cases, as they have been trained on large amounts of data from multiple domains [118]. To enhance their performance in specialized tasks, domain-adaptive pre-training has been demonstrated to be a viable option.

Despite the novelty of LLMs, several studies have been conducted on their application in sentiment analysis. [49] introduces LlamBERT, a model that combines LLMs with transformer encoders such as Bidirectional Encoder Representations from Transformers (BERT) and RoBERTa. LlamBERT was proven to be a cost-effective Natural Language Processing (NLP) solution without significantly compromising accuracy. Using an efficient labelling strategy and fine-tuning process, the study shows the potential of LlamBERT as an alternative to training domain-specific models. [236] leverages the use of instruction-tuned LLMs to enhance financial sentiment analysis. The proposed model surpasses general-purpose LLMs and supervised learning models with minimal task-specific data. [240] investigates the performance of LLMs on

13 sentiment analysis tasks, including aspect-based sentiment analysis and sentiment classification. The study also introduces a novel and more comprehensive benchmark called SENTIEVAL, which improves the assessment and evaluation of LLMs used for sentiment analysis.

[115] proposes BioBERT, a BERT model pre-trained on biomedical corpora. The study found that BioBERT showed improvements across the board in F1 scores for biomedical named entity recognition, relation extraction, and question answering compared to the base model (BERT). Following the same pretraining approach, [85] pre-trained RoBERTa on biomedical papers, newspapers, computer science papers, and Amazon reviews. Their results also showed significant improvements in comparison to the base model. These same improvements have also been observed when applied to hate speech detection. For example, [40] introduces HateBERT, a BERT model trained on data from banned Reddit communities that contain text with hate speech and offensive language. In a similar vein, [118] builds and evaluates COVID-HateBERT, a language model retrained on hateful tweets related to COVID-19, with the intention of detecting hate speech specific to the COVID-19 pandemic and increasing generalization across different hate speech datasets.

Further on the subject of LLMs applied to hate speech detection, [192] compares the performance of logistic regression with two language models – BERT (1-layer and 2-layer) and GPT-3, using a Twitter (X) hate speech detection dataset. Despite the dataset imbalance, the logistic regression model achieved an 80% accuracy but struggled to classify hate speech. The two BERT models showed comparable performance between each other. Still, they achieved superior performance compared to the logistic regression model through fine-tuning and optimization techniques, such

as hyperparameter tuning and dropout rate experimentation, reaching an accuracy of 86%.

[82] presents approaches to classifying hate speech and offensive content with three tasks: binary classification of harmful English tweets, fine-grained classification of English tweets into hate, offensive, or profane categories, and identification of harmful content in Marathi. Various LLMs were fine-tuned for the binary and multi-class classification tasks, including BERT, BERTweet, Twitter-RoBERTa, and LaBSE (Language-agnostic BERT Sentence Embedding) [72]. Twitter-RoBERTa had significant gains in F1 scores for the English classification tasks, with the integration of additional hate speech datasets marginally improving performance.

A summary of the literature focused on fine-tuning LLMs for hate speech detection is presented in Table 2.3.

F1 = F1 Score, P = Precision, R = Recall, A = Accuracy

Table 2.3: Fine-Tuned Large Language Models for Hate Speech Detection

Year	**Task Summary**	**Dataset Classes**	**Fine-tuned Models**	**F1**	**P**	**R**	**A**
2024	Hate speech detection using a German dataset from online comments [210]	Hate speech or non-hate speech.	GPT 3.5	0.82	-	-	-
2021	Hate speech classification of English tweets [66]	Hate speech, offensive, and neither.	BERT	0.84	-	-	-
2022	Hate speech classification on Urdu Tweets [18]	Hate speech, offensive speech, and neutral.	BERT	0.68	0.73	0.66	0.73
			DistilBERT	0.69	0.70	0.69	0.72
			XML-RoBERTa	0.68	0.69	0.67	0.71
2024	Hate speech detection and fine-grained multi-class target community identification in Nastaliq Urdu [132]	Hate or not hate; political, religious or gender-based hate speech.	Urdu-RoBERTa	0.85	0.85	0.85	0.86
			Urdu-DistilBERT	0.87	0.87	0.86	0.87

Continued on next page

Table 2.3 – continued from previous page

Year	Task Summary	Dataset Classes	Fine-tuned Models	F1	P	R	A
2021	Multilingual hate speech detection in Marathi, English and Hindi [32]	Hate and offensive or not; hate, offensive, profane or none.	XLM-RoBERTa mBERT DistilmBERT	0.8 0.76 0.76	- - -	- - -	- - -
2023	Hate speech detection on Italian tweets [83]	Hate speech or not.	UmBERTo BERT-ita	0.9 0.87	- -	- -	- -
2020	Hate speech detection in multi-modal memes [52]	Hateful or benign.	VisualBERT	0.49	0.74	0.36	0.62
2022	Hate speech detection in under-resourced languages [180]	Hateful or non-hateful.	XLM-T [26], UmBERTo, AraBERT V2 [21], Hindi BERT, RoBERTuito [133]	0.83	-	-	-

Continued on next page

Table 2.3 – continued from previous page

Year	Task Summary	Dataset Classes	Fine-tuned Models	F1	P	R	A
2024	Hate speech detection on Twitter (X) posts and YouTube comments in Brazilian Portuguese [168]	Hate speech or non-hate speech.	BERTimbau, GPT 3.5/4, Gemini-Pro, Mistral 7B	0.77	0.77	0.76	-
2024	Hate speech classification, target identification and determination of stances on climate activism from English Tweets [45]	Hate or non-hate; Individual, organization or community; support, oppose, neutral.	BERT	0.71	-	-	0.90
			DistilBERT	0.67	-	-	0.90
			RoBERTa	0.67	-	-	0.84
			ClimateBERT	0.70	-	-	0.88
			Mistral 7B	0.61	-	-	0.95

Continued on next page

Table 2.3 – continued from previous page

Year	**Task Summary**	**Dataset Classes**	**Fine-tuned Models**	**F1**	**P**	**R**	**A**
2023	Detection of sexual predatory behavior and abusive language in English, Roman Urdu and Urdu datasets [148]	Predatory or non-predatory.	Llama 2 7B	0.98	-	-	1
2024	Hate speech detection in code-switched language from political discussions in Nigeria [97]	Normal, hateful, offensive, abusive and contempt.	Twitter-RoBERTa	0.95	-	-	-
2024	Multimodal hate speech detection using Twitter (X) and YouTube [188]	-	Llama 2	0.94	-	-	-

Continued on next page

Table 2.3 – continued from previous page

Year	Task Summary	Dataset Classes	Fine-tuned Models	F1	P	R	A
2023	Collection of Arabic tweets containing offensive and hateful language using emojis as anchors [140]	Offensive or not; hate speech or not.	AraBERT	0.80	0.81	0.80	0.82
			QARiB	0.82	0.82	0.82	0.84
2024	Detection of anti-semitic hate speech [122]	-	BERT	0.81	0.96	0.73	0.94
			DistilBERT	0.80	0.91	0.71	0.92
			RoBERTa	0.81	0.88	0.73	0.93
			Llama 2 7B	0.83	0.80	0.79	0.92
2024	Implicit hate speech detection using LoRA [30]	Implicit-hate or non-hate.	Mistral 7B	-	-	-	0.92

Continued on next page

Table 2.3 – continued from previous page

Year	**Task Summary**	**Dataset Classes**	**Fine-tuned Models**	**F1**	**P**	**R**	**A**
2023	Generalizability of LLMs to cross-domain hate speech detection and the impact of fine-tuning [144]	Hateful or neutral.	Vicuna 7B	1	-	-	-
			Llama 7B	1	-	-	-
2024	Instruction-tuning for social science tasks including hate speech, irony detection and sentiment [60]	-	Llama 2 7B	0.42	-	-	-

According to a survey of 436 papers by [100], BERT has been found to be the most popular deep learning and transformer model in hate speech detection, with different articles concluding that BERT demonstrated the best performance [15, 65, 150] even in multilingual tasks [13, 33, 163, 217].

Aside from BERT, most other hate-speech-related research using transformer models and LLMs has focused on GPT (or its chatbot – ChatGPT) [19, 20, 43, 47, 78, 112, 222] and Llama 2 [112, 148, 160, 172, 237, 238].

As of the time of writing this thesis, very few works have fine-tuned a Llama model for hate speech classification [90, 144, 148, 158], and to our knowledge, no articles fine-tuning Llama 3 (Meta's latest LLM) for hate speech detection have been published. However, we intend to fill these gaps with this study.

The next chapter will provide a deep dive into the architecture of transformer models and LLMs, with a focus on the Llama models and the various techniques used for training and quantizing them.

Chapter 3

Optimization of Large Language Models for Text-Based Applications

3.1 Overview of Large Language Models

LLMs have been defined by their capabilities to process and generate text through extensive training on vast datasets using the Transformer architecture. Since the development of early foundational models, such as BERT and GPT, and their demonstrated effectiveness, there has been a significant increase in LLM releases, as shown in Figure 3.1.

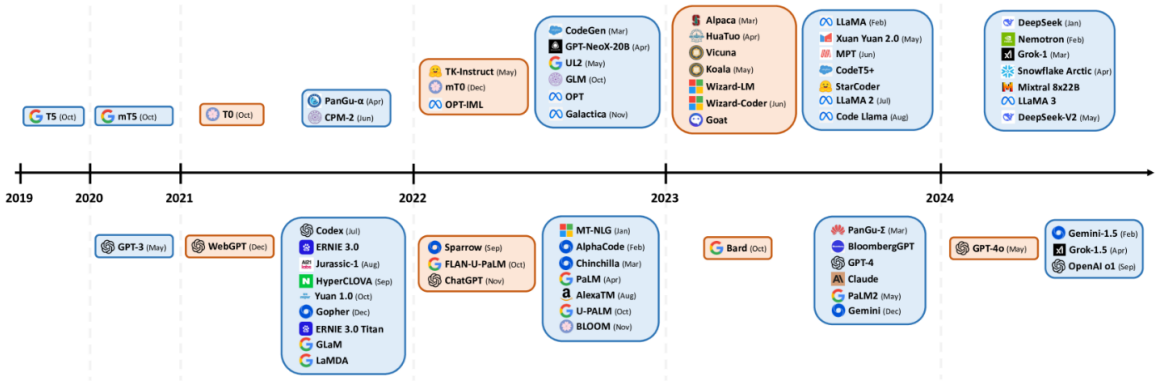

Figure 3.1: LLM Releases Since 2019 [146].

The Transformer architecture [212], introduced in 2017, revolutionized NLP with the use of the attention mechanism [25, 129, 212], which enabled models to comprehend the context of words in relation to others across a sentence. This enables LLMs to learn complex grammatical structures and semantic relationships, resulting in superior text generation performance.

However, this superior performance comes at the cost of slow training, inference, and high running cost, leading to the introduction of Parameter-Efficient Fine-Tuning methods (PEFT) [87, 116, 120], pruning [130, 228], and quantization [205, 226], which have enabled LLMs to be fine-tuned and utilized on fewer resources, making the process more computationally efficient.

Additionally, Retrieval-Augmented Generation (RAG) [77, 183] has also emerged as a solution to LLM limitations in memory and real-time information retrieval. With the use of external knowledge bases, RAG enables LLMs to access and respond with up-to-date, contextually relevant information.

3.2 The Transformer Architecture

The transformer architecture is the fundamental building block of all large language models. A transformer model has two key components: an encoder and a decoder. The encoder processes the input to a transformer model by mapping it to a sequence of vectors. The vectors are then passed on to the decoder, which produces the output [14, 212]. In reality, the encoder and decoder components are stacks of multiple encoders and decoders, respectively.

A high-level visualization of how a transformer works is displayed in Figure 3.2.

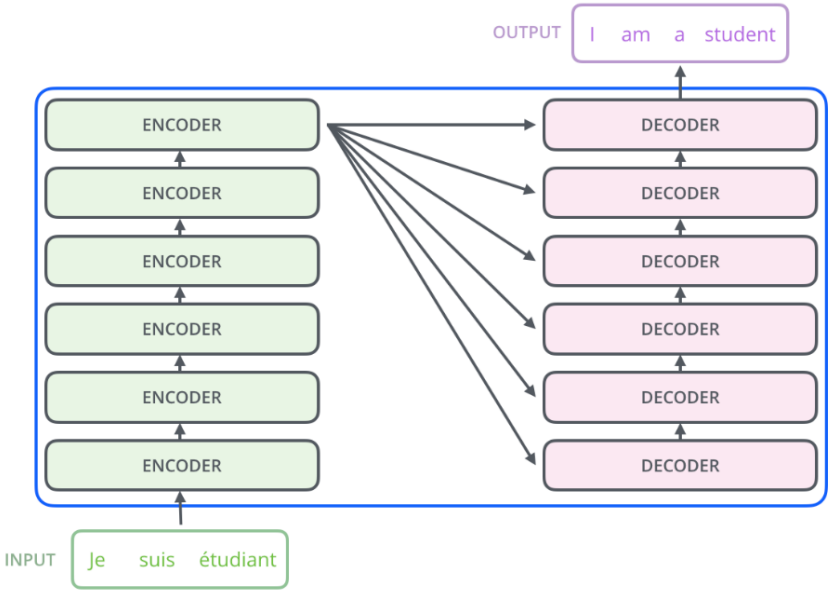

Figure 3.2: High-level Overview of a Transformer Process [14].

3.2.1 The Encoder Side

This section explores the processes of a single encoder in a transformer model. The encoder is a key component of a transformer model that helps understand the meaning of each word in a sentence by examining its relationship to the others. Each encoder block consists of two main components: self-attention and a feed-forward network.

Processes in the Encoder Side

Let's imagine our textual input as a question asked to the model, a text in a foreign language for the model to translate, etc. Firstly, each input word or token in a sentence becomes a list of numerical vectors called embeddings, capturing the meaning and relationships between tokens. Then, positional encodings are added to the token embeddings. These encodings contain information about the order of the words in a sentence.

1. **Self-Attention Mechanism**

These enriched vectors go into a self-attention mechanism, which allows the model to focus on the most relevant words for understanding a particular token. For example, in the sentence *"The animal didn't cross the street because it was too tired,"* the model uses self-attention to figure out that "it" refers to "animal." The self-attention mechanism is essentially a process in transformer models which helps the model understand the meaning of a word by looking at all the other words in the sentence. In other words, each token in the input sequence attends to all other tokens. This is illustrated in Figure 3.3.

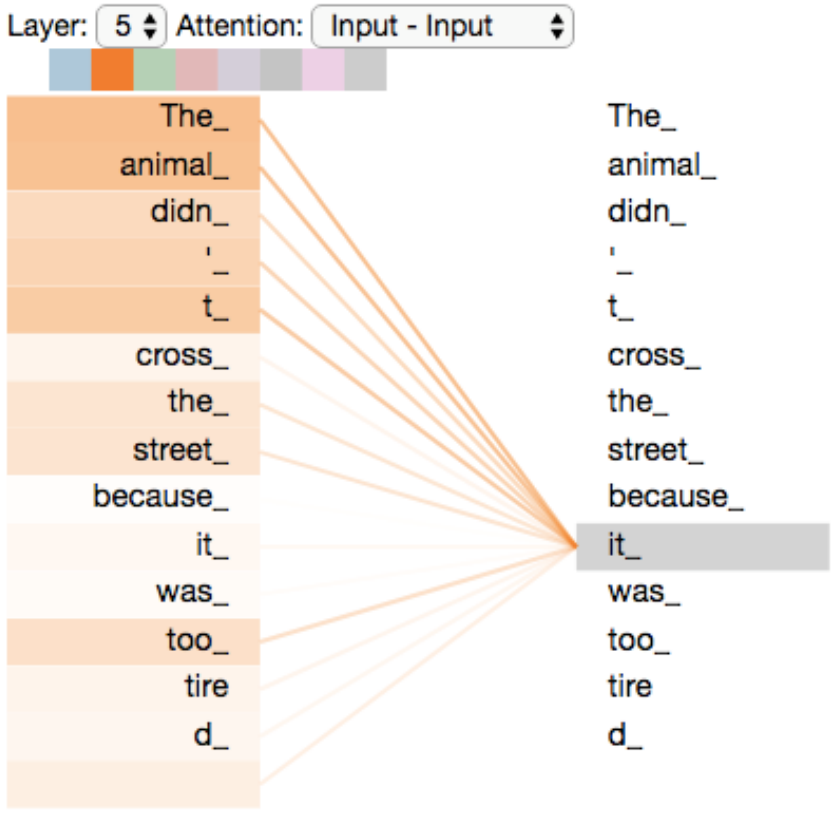

Figure 3.3: Attention Mechanism [14]

2. **Query, Key, and Value Vectors**

To determine the relationship between one token and others in the input sequence as discussed above, the model creates query, key, and value vectors for each token. [14, 73].

- **Query vector:** Represents a token "searching" for relevant information. We can think of a query as asking, "What am I looking for?"
- **Key vector:** Encodes information that is related to the query, in this case, representing the features of all the tokens in the input sentence. The key vector is the same as asking, "What do I have that might match your needs?"
- **Value vector:** Contains the information contributing to the final output. It is used to update the understanding of the token based on attention scores. We can think of the value vector as asking, "What useful information do I carry?"

In a nutshell, the model compares queries and keys to decide which words are important and uses the values of those words to update the understanding of the current token. We can intuitively understand this using an analogy.

Career Fair Analogy: A recruiter (the token) walks into a career fair with a clear job description in mind and needs to know "who meets these qualifications?" (the query). Each candidate (token in the sentence) has a resume describing their skills. This is the key. Each resume also includes specific experiences and projects they've acquired. This is their value and defines their worth to the recruiter. The recruiter reads all resumes, evaluates how relevant each candidate is to the job description (computing an attention score between the

query and key), and then forms an understanding of whom to hire by weighting each candidate's experience and projects (values) according to how well their resume matched the recruiter's needs. The recruiter doesn't just pick one candidate, instead, they form an aggregated view by giving more influence to candidates with higher match scores, but still considering all.

3. **Attention Scores**

   These comparisons use dot products to measure the similarity in direction between two vectors (in this case, the query and key vectors). Suppose we have:

   Query vector, q = [1, 2]

   Key vector, k = [3, 4]

   The dot product of the query and key is calculated by multiplying corresponding elements and then adding them together:

$$\mathbf{q} \cdot \mathbf{k} = (1 \times 3) + (2 \times 4) = 3 + 8 = 11$$

   A larger dot product means greater alignment or similarity between the query and key vectors. In the attention mechanism, this similarity decides how much attention the model should pay to the key when producing the output. The dot product is divided by the square root of the dimensions of the key vector $\sqrt{d_k}$ (where $d_k = 64$ [212]) and passed through a softmax activation function.

   The softmax activation function is given by:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- z: vector of raw attention scores (input vector)
- z_i: i-th element of z
- K: number of elements in z

The softmax emphasizes higher scores and suppresses smaller ones to focus attention. A visual example of the softmax activation function calculations can be seen in 3.4.

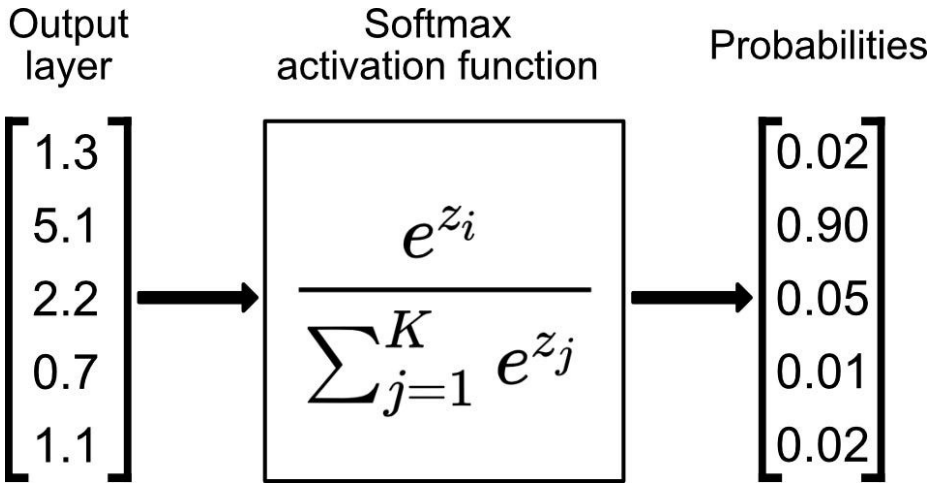

Figure 3.4: Softmax Activation Function [164]

After normalization with softmax, the attention scores (now probabilities) are used to compute a weighted average of the value vectors. Each value vector is multiplied by its corresponding softmax probability, and these weighted vectors are then summed together. The result is a new, context-aware representation for each token, where information from the most relevant tokens is emphasized more strongly.

An outline of the attention steps is outlined in 3.5.

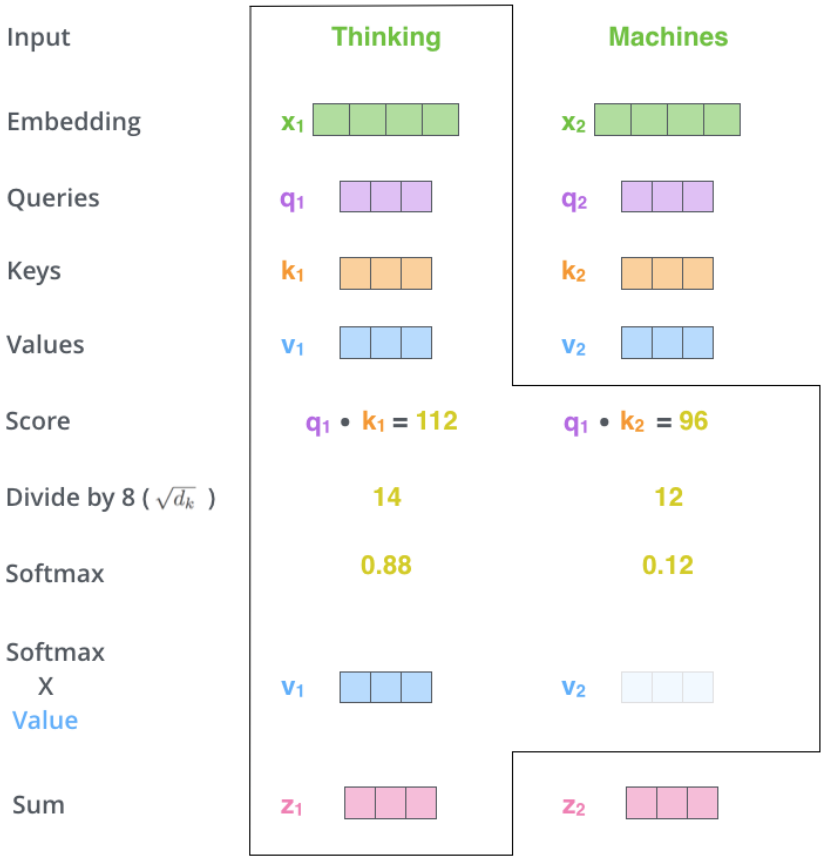

Figure 3.5: Attention Calculation [14]. $d_k = 64$ is the key vector dimension.

4. **Multi-Head Attention**

   In practice, self-attention is performed multiple times in parallel using different learned projections and this is called multi-head attention. It enables the model to capture different types of relationships (e.g., syntactic, semantic, affective) simultaneously.

5. **Feed-Forward Layer**

   After attention, the encoder adds the original input sentence back to the output

(known as a residual connection), normalizes it, and passes it through a feed-forward layer for refinement. This process is repeated across multiple layers. The final output of the encoder now contains rich contextual information, ready to be passed to the decoder for further tasks like translation or text generation.

3.2.2 The Decoder Side

The decoder in a transformer is responsible for generating the output sequence (e.g., in translation from English to French, word by word). The process on the decoder side of the transformer model is similar to that of the encoder, with a few exceptions.

- **Masked Self-Attention** [14, 73]

  The decoder uses a special type of self-attention called "masked" self-attention during training to generate one word at a time. In masked self-attention, each token is influenced only by itself and the token(s) before it. In other words, a mask is applied to prevent the model from attending to (or seeing) future tokens, ensuring that predictions for the current position only depend on tokens up to the current position.

  For example, in the sentence "The book was on the table, and I picked it up," if the model is trying to predict the word "was," the decoder can only attend to "The" and "book." This is called causal masking and is essential for maintaining the correct order during text generation.

- **Encoder-Decoder Cross-Attention**

  Unlike the encoder, the decoder also includes another key step: cross-attention. This mechanism allows the decoder to attend to or look back at the final encoder's output to gather context from the input sentence. It works in the

following way:

- The decoder generates the query vectors based on the outputs it has generated so far.
- The key and value vectors come from the encoder stack based on the input sentence.
- The decoder compares its queries to the key vectors, and based on the attention scores, retrieves the most relevant information from the value vectors.

From the example in Figure 3.6, when translating "Je suis étudiant," the decoder is auto-regressive (predicts future values based on past values) and generates one token at a time. The steps are as follows:

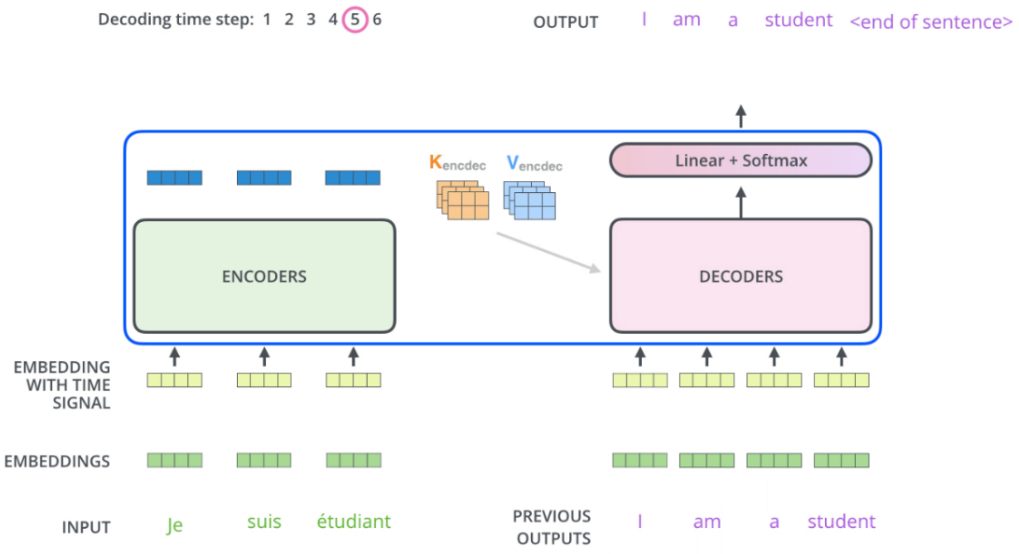

Figure 3.6: Encoder-Decoder Cross-Attention [14]

Decoding time step 1: The encoder has processed the full input sentence: "Je suis étudiant" and produced a set of contextual embeddings which are stored

as key and value vectors (K_{encdec} and V_{encdec}). These are used by the decoder and passed through the linear and softmax layers to produce the first output – "I."

Decoding time step 2: The decoder takes the previously generated token "I" as input, performs masked self-attention and cross-attention, and then applies the linear and softmax layer to produce "am."

Decoding time step 3: The input to the decoder is now "I am." Masked self-attention is applied, and cross-attention is done with the encoder outputs (K_{encdec} and V_{encdec}). Passes through the linear and softmax layers, producing "a."

Decoding time step 4: The same processes are applied to produce "student."

Decoding time step 5: At this step, the model produces an `<end of sentence>` token, signalling the end of text generation.

Decoding time step 6: Generation is finished as the `<end of sentence>` token is produced.

- **Token Prediction: Turning Vectors into Words**

  Once the decoder has gathered all the contextual information above, the result is a vector of numbers which represent meaning. This vector goes through two layers:

  - A linear layer, which is like a classifier, that maps the vector into the size of the vocabulary that the model has learned to recognize and generate.
  - A softmax layer, which turns those numbers into probabilities for each possible word.

The model then chooses the word with the highest probability as the next token in the sequence. This step is repeated until a full sentence is generated.

This mechanism aligns the partially generated output with the input sequence, allowing the model to incorporate information about the input while generating each token. Just as in the encoder, the outputs of the cross-attention mechanism are then passed through a feed-forward neural network for processing.

3.2.3 Variations of Transformer Architectures

There are several variations of the transformer architecture, which will be discussed below.

- **Causal Decoder Architecture**

  This is a decoder-only architecture, where the output tokens are predicted and generated using the preceding tokens (utilizing a unidirectional attention mask) [241, 241]. The most notable LLMs using this architecture are the GPT models [38, 156, 166]. Others are OPT (Open Pretrained Transformers) [239], BLOOM (BigScience Large Open-science Open-access Multilingual Language Model) [206], and the Llama models (Llama 2, Llama 3) [67, 208] which we are using in this study.

  Figure 3.7 shows the architecture of a GPT model. At the bottom of the model, each input token is converted into a vector using an embedding layer. Next, a positional encoding is added to give each token an order in the sequence. The combined vector is then passed into the first transformer block (N), where the same processes described in 3.2.1 (The Encoder Side) & 3.2.2 (The Decoder Side) are followed. There are x number of blocks depending on the model.

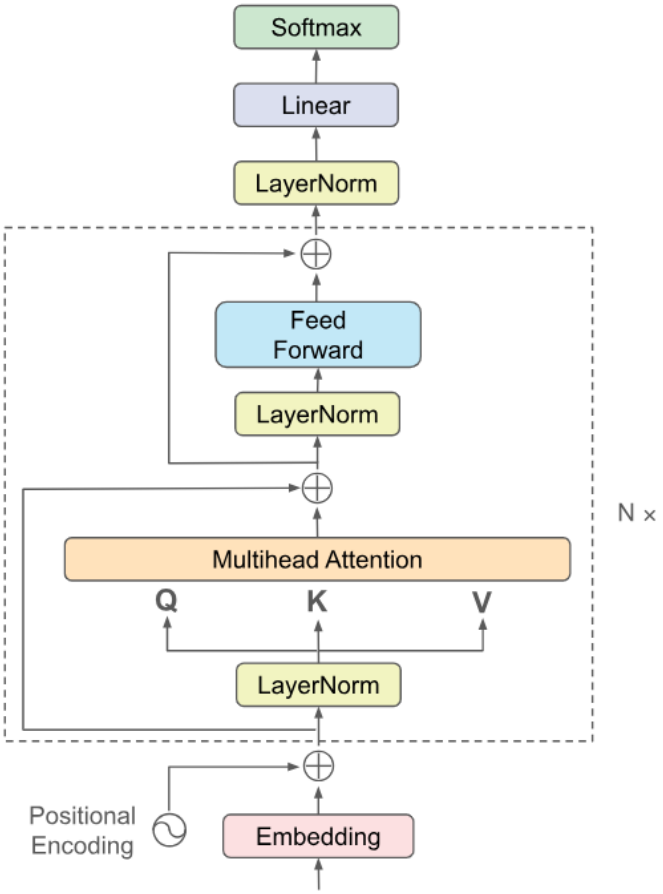

Figure 3.7: Architecture of a Causal-Decoder Model [4]

The transformer blocks contain attention and feed-forward layers, with layer normalization after each one. There is also the presence of residual connections (shown in the diagram as $\oplus$) [88], which are shortcuts in neural networks that allow information to bypass one or more layers. These facilitate effective training in the transformer model by retaining a copy of the original input and adding it back to the output of a layer.

The Llama architecture shown in Figure 3.8 is quite similar in processes to a conventional transformer decoder architecture, but differs with the use of [22]:

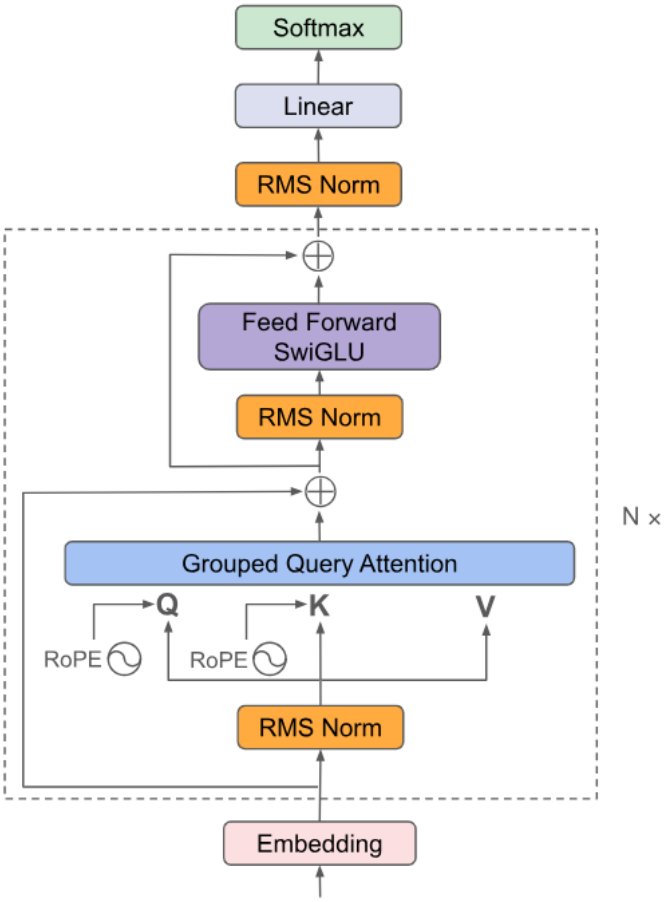

Figure 3.8: Architecture of a Llama Model [4]

- Root Mean Square Normalization (RMSNorm) [235] instead of LayerNorm [23].
- Grouped query attention instead of multi-head attention, resulting in better computational efficiency [10].
- SwiGLU activation [191].
- Rotary Positional Embeddings (RoPE) [200].

- **Encoder-Decoder Architecture**

This architecture consists of encoder and decoder stacks to process input and generate the output sequence respectively. Encoder-decoder models include T5 [159], BART [117] and Flan-T5 [46, 113].

- **Prefix Decoder Architecture**

  Here, the input (i.e. the prefix tokens) is attended to in both the forward and backward directions (using bidirectional attention). At the same time, the output tokens are generated using unidirectional attention (as in the causal decoder architecture), looking at previous tokens [241]. An example is GLM-130B (General Language Model) [234].

- **Mixture-of-Experts (MoE) Architecture**

  Here, only a subset of the model's weights is activated for each input, allowing models to scale up their parameters while keeping computational costs low. Grok-1 [2] and Gemini-1.5 [173] are LLMs using the MoE architecture.

- **Encoder-only Architecture**

  Encoder-only architectures process only the input sequences but do not generate output sequences. They are focused on understanding the input instead of generation and are mostly used for sentiment analysis and text classification. Examples are BERT and RoBERTa [126, 156].

3.3 Model Optimization: Fine-tuning and Quantization

The growing size and complexity of LLMs have presented high memory requirements, computational costs, and latency during inference. Optimization addresses these challenges by making them more efficient and adaptable to different environments. Optimization techniques include pre- and post-training quantization, parameter-efficient fine-tuning, and pruning.

3.3.1 Fine-tuning

After pre-training, the model is more capable of tackling a wide variety of general tasks like question-answering, summarization, and sentiment analysis. However, LLM performance can be further improved by fine-tuning for specific tasks [241]. This involves updating the weights of the pre-trained LLM, thereby optimizing it for the required task.

One of the most direct approaches is full fine-tuning, where all the model's weights are updated. However, given the billions of parameters the latest LLMs have, this process is computationally expensive and time-consuming. This has led to the development of Parameter-Efficient Fine-Tuning (PEFT) techniques, which involve updating fewer LLM parameters through fine-tuning [103]. Common PEFT techniques include prompt tuning [116], prefix tuning [120], adapter tuning [92], and Low-Rank Adaptation (LoRA) [91].

3.3.2 Low-Rank Adaptation (LoRA)

With LoRA, low-rank matrix approximations are used to reduce the number of trainable parameters during fine-tuning [91, 241].

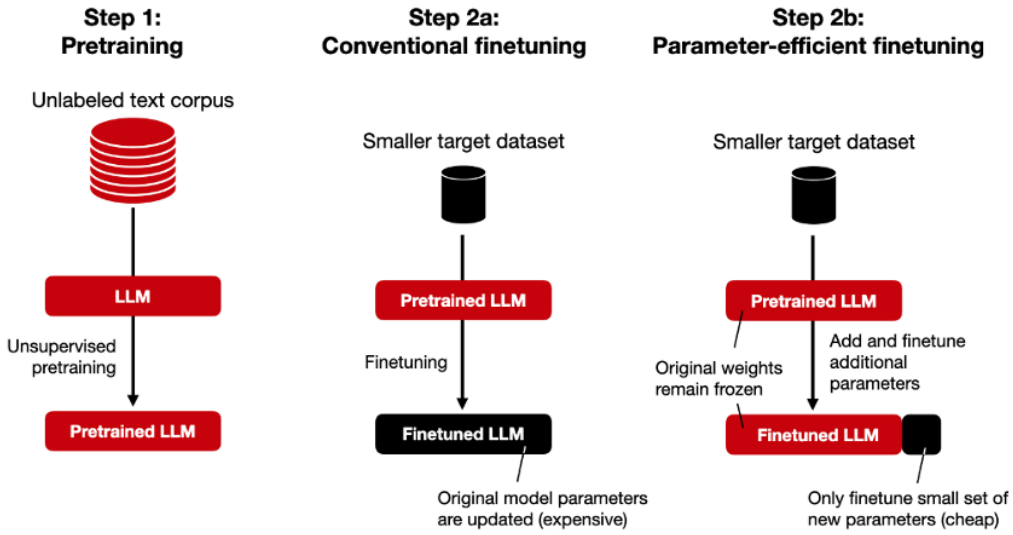

Figure 3.9: Conventional (Full) Fine-Tuning vs Parameter-Efficient Fine-Tuning [103].

In full fine-tuning, the entire set of pre-trained weights of a model is updated during training. Mathematically, this is done with a weight update matrix ΔW, which is added to the original weight matrix W. This results in the new weight matrix, $W' = W + \Delta W$, where W' is the updated weight. While this technique adapts the model fully to a new task, it involves updating millions or billions of parameters.

With LoRA, instead of updating the entire pre-trained model, the original weight matrix W is frozen, and the weight update ΔW is decomposed into the product of two smaller matrices W_A and W_B, which are trained [91]. After training, these two smaller matrices are added to the original weight matrix, giving $W' = W + W_A W_B$ [171].

Another important component of LoRA is the introduction of adapter modules [147] into specific layers (like attention layers) without modifying the original

parameters. These adapters learn W_A and W_B and implement the low-rank decomposition to approximate ΔW. These adapters are trained during fine-tuning and contain fewer trainable parameters, compared to full model fine-tuning. This results in minimal memory usage and faster training times, although with potential trade-offs in accuracy [170].

Quantized Low-Rank Adaptation (QLoRA). The PEFT technique used in this thesis is QLoRA. QLoRA is LoRA with an additional step: prior to adapter insertion and fine-tuning, the original model weights are loaded in 4-bit NormalFloat precision (NF4) [58] from full precision (FP32)/ half precision (FP16). The quantized weights are then dequantized to FP16 during fine-tuning for efficient GPU computation. As in LoRA, the model weights are then frozen and only the small low-rank adapter matrices (W_A and W_B) are trained. With QLoRA, the 4-bit quantized weights are dequantized/decompressed to FP16 precision for computations during fine-tuning allowing for more efficient matrix operations on GPUs. Overall, QLoRA drastically reduces the memory footprint while preserving model performance.

Other PEFT techniques include:

- **Adapter tuning:** In this method, trainable neural networks called adapters are added to each transformer layer, compressing the feature vectors (numerical representations of the important characteristics of data) into lower dimensions, processing them (using nonlinear transformation) and decompressing them back to their original dimensions [241]. Only these adapter layers are updated during training, while the main model's parameters stay frozen, reducing the computational overhead.

- **Prefix tuning:** In place of adapters, small trainable continuous task-specific vectors, known as prefixes, are added to the input of each transformer layer. These prefixes are optimized during fine-tuning using the reparameterization trick (using a smaller matrix that is transformed into the prefix parameter matrix) [120, 224]. In a nutshell, prefix tuning fine-tunes the prefixes while most of the parameters of the transformer remain unchanged.

- **Prompt tuning:** : Trainable prompt tokens (or vectors) are introduced into the model's input layers and act as task-specific instructions to augment the input text. These prompt tokens are combined with the input text vectors and processed by the language model [126,241]. With this method, only the prompt tokens are updated during training.

3.3.3 Quantization

Quantization is a technique in neural network optimization that compresses model parameters and activations from FP32/FP16 to lower-precision formats like INT8 or INT4. This reduces memory use and speeds up inference.

Empirical evidence shows INT8 quantization generally preserves performance, while INT4 requires advanced techniques to minimize model degradation. Notably, LLMs are less sensitive to quantization than smaller models, making them good candidates for aggressive compression [241].

Post-Training Quantization (PTQ). PTQ is an approach for compressing (pre-)trained models without retraining, reducing the computational resources, model size, and latency during model inference while maintaining accuracy. This makes PTQ an

invaluable technique for LLM deployment [124].

There are several libraries supporting PTQ. For our proposed model, we will use the `bitsandbytes` [1] and `GPTQ` [6] libraries for PTQ. `bitsandbytes` supports `NF4` and `LLM.int8()` [57] quantization, while GPTQ implements the GPTQ algorithm [75].

3.4 Proposed Model Overview

There are several ways to apply model for classification. The first method is to directly use the base model without fine-tuning as seen in Figure 3.10.

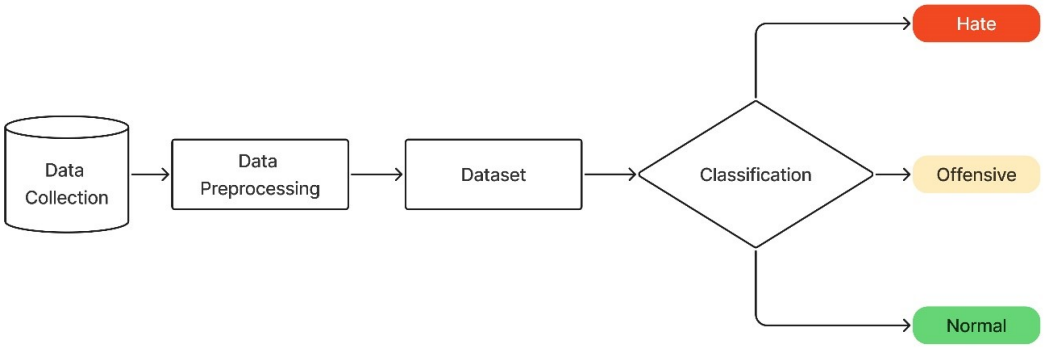

Figure 3.10: Direct Classification with Base Model

Figure 3.11 shows an alternative approach using the model for classification tasks after fine-tuning, but without the extra step of quantization as an optimization method.

However, our proposed model will incorporate both fine-tuning and quantization steps to build a faster, more efficient and compact model as outlined in Figure 3.12.

- **Data Collection:** The initial step involves collecting and combining the Measuring Hate Speech dataset [51] and the Automated Hate Speech Detection and

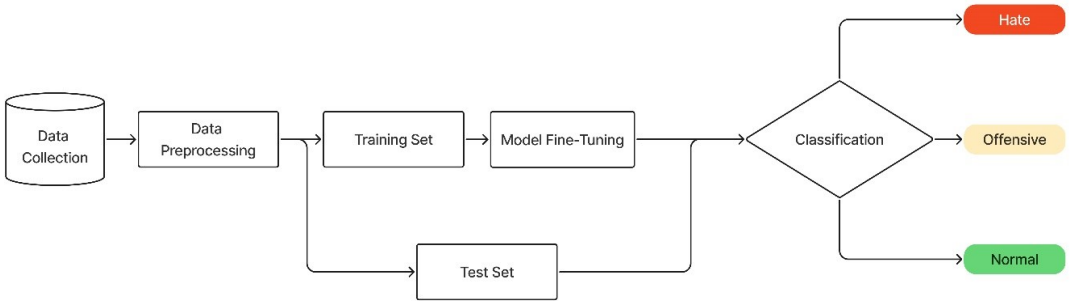

Figure 3.11: Fine-tuned Model without Optimization

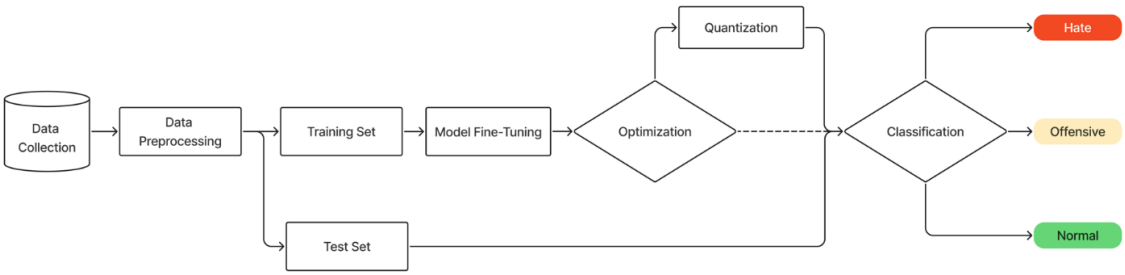

Figure 3.12: Proposed Model Overview

the Problem of Offensive Language dataset [55].

- **Data Preprocessing:** The dataset is cleaned and preprocessed, removing unnecessary columns, rows, and text features (such as usernames, URLs, symbols, etc.) that constitute noise. Afterwards, Exploratory Data Analysis (EDA) is performed to obtain more information about the combined dataset.

- **Dataset Split:** The dataset is split into 6000 training, 900 validation, and 900 test samples, and the entries are formatted as prompts for the LLM. The test data is stored for future use during inference.

- **Fine-tuning:** The Llama 3.1 model is fine-tuned on the training data using

QLoRA and saved. Afterwards, the model can be used as is or further optimized with quantization.

- **Quantization:** The model is quantized using the GPTQ library [6] and saved for inference.
- **Classification:** After optimization, the model is used as a classifier for the dataset and sorts the test set into hate speech, offensive language or neutral classes. Afterwards, the output is evaluated based on the F1, precision, recall, and accuracy scores.

Chapter 4

Model Implementation for Hate Speech Classification

4.1 Data Preparation

Due to varying interpretations of what constitutes hate speech, I sourced datasets closely aligned to the United Nation's definition, which describes hate speech as:

> "Any kind of communication in speech, writing or behaviour, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, colour, descent, gender or other identity factor" [145].

The second criterion was for the datasets to include classes differentiating between hate speech, offensive speech, and normal/neutral language. To ensure broader applicability, datasets focusing on a specific region or cultural context were not considered, as perceptions of hate speech can vary significantly across cultures and geographies.

Two datasets meeting these criteria were chosen and combined. The first dataset

was obtained from the Automated Hate Speech Detection and the Problem of Offensive Language study by Thomas Davidson et al [55].

The automated hate speech detection dataset was created using a hate speech lexicon compiled by Hatebase, containing terms identified by internet users as hate speech. Tweets (posts) containing these terms were collected from 33,458 users, resulting in 85.4 million tweets. A random sample of 25,000 tweets was manually labelled by CrowdFlower (CF) workers into three categories: hate speech, offensive (but not hate speech), or neither. Coders were instructed to consider the context, not just the presence of specific words.

Each tweet was reviewed by at least three people, achieving an intercoder agreement (to what extent two or more people agree when they are labelling or categorizing the same set of data) of 92%. The final labels were assigned based on a majority decision, resulting in 24,802 labeled tweets.

The dataset is summarized in Table 4.1.

Table 4.1: Description of the Automated Hate Speech Detection Dataset

Column Name	**Description**
count	Number of CrowdFlower (CF) users who coded each tweet (minimum is 3, sometimes more users coded a tweet when judgments were determined to be unreliable by CF).
hate_speech	Number of CF users who judged the tweet to be hate speech.
offensive_language	Number of CF users who judged the tweet to be offensive.
neither	Number of CF users who judged the tweet to be neither offensive nor non-offensive.
class	Class label for majority of CF users. 0 - hate speech, 1 - offensive language, 2 – neither.
tweet	The examined tweets.

The `class` and `tweet` columns are used for our combined dataset. The `class` column contains 1,430, 19,190, and 4,163 tweets classified as hate speech, offensive language, and neither respectively. A visual representation of the dataset can be found in Figure 4.1.

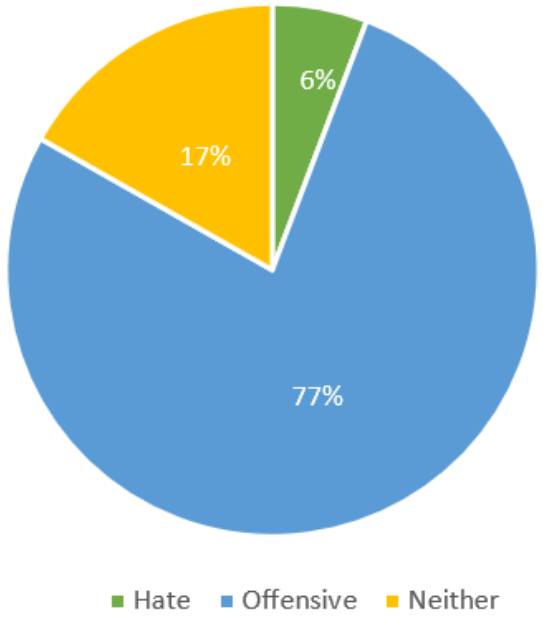

Figure 4.1: Percentage Distribution of Automated Hate Speech Detection Dataset

The second dataset, the measuring hate speech dataset, was sourced from the UC Berkeley D-lab [51]. According to this paper [182], this dataset adopts the legal definition of hate speech in the US and uses this for classifying hate speech. The dataset of 39,565 comments was annotated by 8,472 annotators from Amazon Mechanical Turk (a crowdsourcing marketplace), with the annotators rating each comment on 10 different features (e.g., respect, violence, hate), and identifying targeted identity groups (e.g., race, gender, religion). Statistical checks like the Rasch model [135] identified unreliable annotators (e.g., those who were too random, too predictable, or finished their assigned tasks too fast), and they were removed [182]. This process ensured that the dataset was diverse, high-quality, and statistically validated.

Table 4.2 describes the key columns of the dataset.

Table 4.2: Description of the Measuring Hate Speech Dataset [51]

Column Name	**Description**
hate_speech_score	Continuous hate speech measure, where higher = more hateful and lower = less hateful. > 0.5 is approximately hate speech, < -1 is counter or supportive speech, and -1 to $+0.5$ is neutral or ambiguous.
text	Lightly processed text of a social media post.
comment_id	Unique ID for each comment.
annotator_id	Unique ID for each annotator.
sentiment	Ordinal label measuring the overall positive or negative tone of a comment.
(dis)respect	Ordinal label that captures whether a comment is respectful or disrespectful towards an individual or group.
insult	Ordinal label identifying whether a comment contains direct personal attacks, name-calling, or derogatory language.
humiliate	Ordinal label that shows if a comment aims to shame, embarrass, or degrade a person or group.
status	Ordinal label measuring whether a comment suggests or implies a group is lesser than others.
dehumanize	Ordinal label measuring whether a comment compares people to non-human entities.

Continued on next page

Table 4.2 – continued from previous page

Column Name	**Description**
violence	Ordinal label measuring the level of violence of a comment.
genocide	Ordinal label that measures whether a comment advocates for extermination of a group.
attack_defend	Ordinal label that indicates if a comment is an attack, defense, or neutral in a conflict or debate.

However, our focus is on the *hate speech* column, containing 3 classes – 0, 1, and 2. Class 2 is assigned to the most hateful tweets/comments and 0 is neutral (but not necessarily harmless/inoffensive).

The hate speech class from the automated hate speech detection dataset is augmented with more tweets from the second dataset to make up for the underrepresentation of the class. For this, we select the comments labelled as '2' in the second dataset containing hateful comments.

The final dataset is of the structure below:

Table 4.3: Structure of Final Combined Dataset

Column Name	**Description**
Sentiment	Contains 3 sentiment labels: hate, offensive & normal.
Text	Contains the tweets/comments.

After dropping duplicates, the final combined dataset contains 19,190 posts labelled as "offensive," 16,342 labelled as "hate," and 4,163 – "normal."

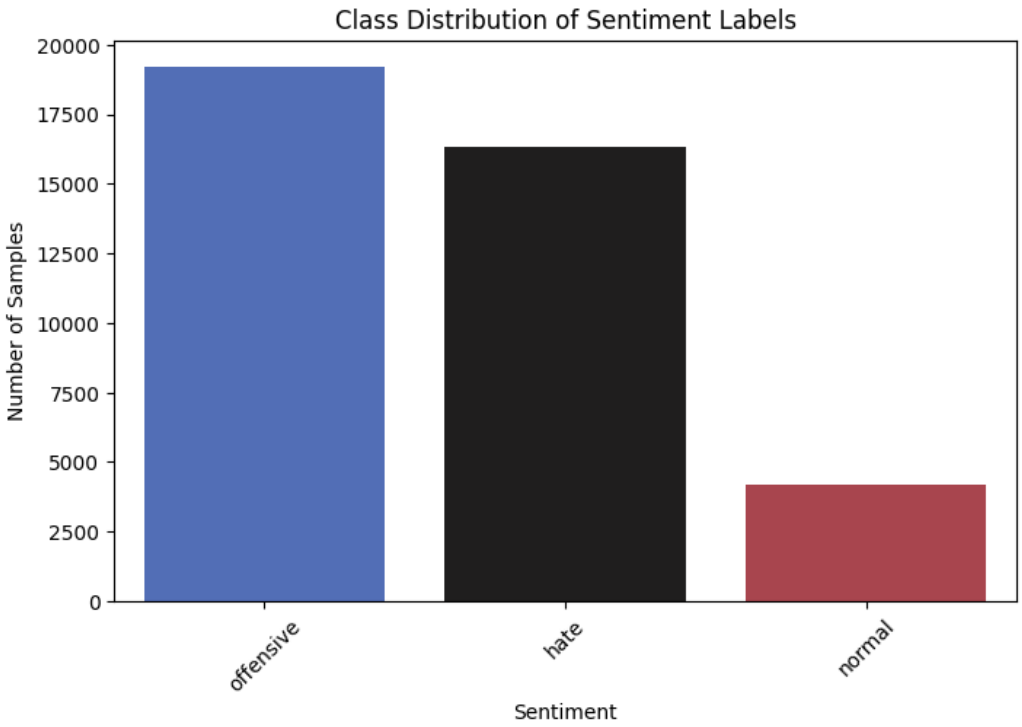

Figure 4.2: Class Distribution

4.2 Data Preprocessing

The first stage of data preprocessing was to convert the numerical class labels (0, 1, 2) into textual representations (hate, offensive, and normal), which allowed for a more intuitive understanding of the dataset.

Text Cleaning

The `text` column contained extraneous elements that could introduce noise to the model. To address this, a text-cleaning function was implemented using regular expressions (RegEx). Consequently, the following were eliminated from the column:

1. HTML Entities: The placeholder &amp

2. Mentions of Twitter handles (e.g., @username) to prevent user-specific bias.
3. Retweet indicators (RT).
4. URLs.
5. Numbers and certain HTML character codes (e.g., {).

Handling Missing Data

An additional validation step was conducted to identify empty text entries. This involved implementing a filtering operation to detect rows where the text column contained an empty string.

4.3 Exploratory Data Analysis

To understand the linguistic characteristics and word usage trends of tweets in each sentiment class, word clouds were generated. This created visual representations of the most common words in each category. To augment the word clouds, bar plots revealing the top 20 most frequently used words in each category were also plotted. However, prior to creating these visualizations, punctuation and non-informative/stop words such as articles (a, an, the), forms of the verb "to be" (were/was, is/are), and conjunctions (and, but), etc. were filtered out using the Natural Language Toolkit (NLTK) library [34]. The visualizations can be seen in the word clouds below.

Common Words Used in Hate Speech

Figure 4.3: Word Cloud of Hate Tweets

From the word cloud in Figure 4.3, the most common words in hate tweets are explicit slurs ("nigga," "fuck," and "bitch"). Words associated with violence and aggression also appear frequently ("kill," "hate," "racist," "slut"). Furthermore, the presence of identity-based terms ("jew," "Muslim," "white," "American," "women," and "gay") suggests that much of hate speech is targeted at specific groups. These words are not inherently offensive but appear in hateful contexts emphasizing the need to consider sentence structure and surrounding words, not just individual words.

Interestingly, "like" is the most common word in the bar graph in Figure 4.4. This is due to "like" being treated as a stopword in the WordCloud library [141]. Furthermore, upon inspection of the dataset, this was mostly used as a filler word and as a synonym of "as if."

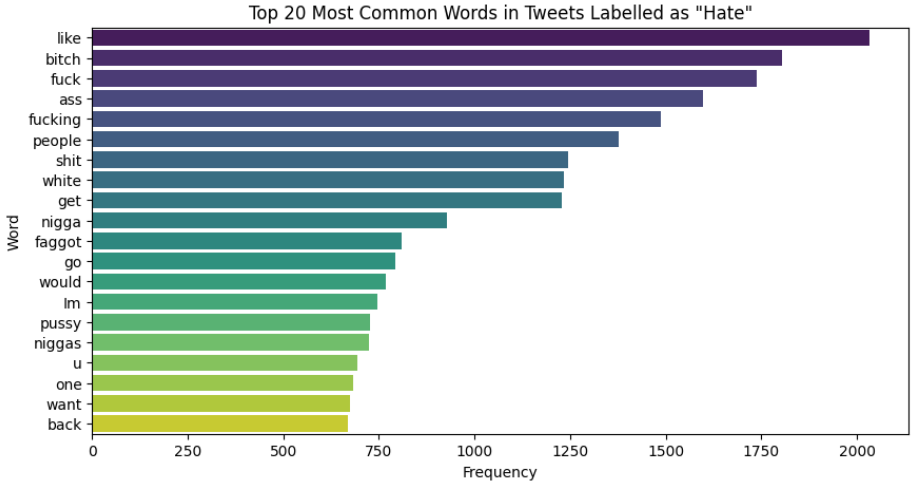

Figure 4.4: 20 Most Commonly Used Words in Hate Tweets

Common Words Used in Offensive Speech

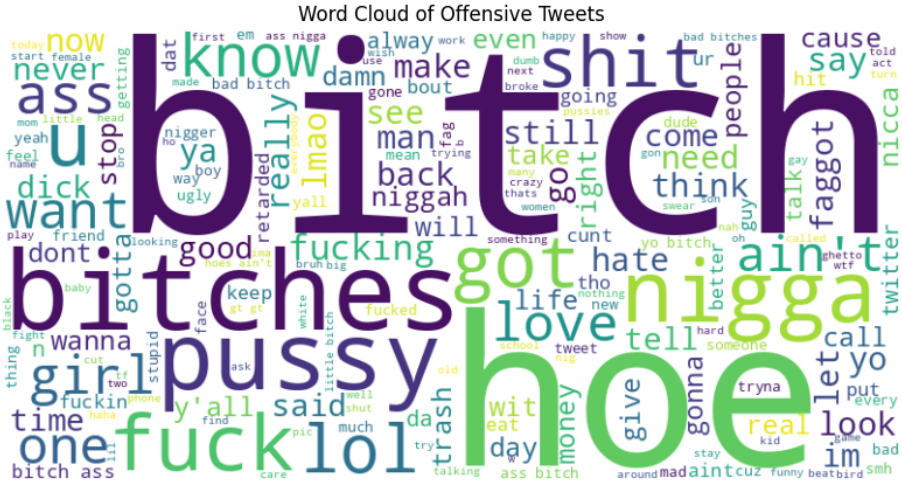

Figure 4.5: Word Cloud of Offensive Tweets

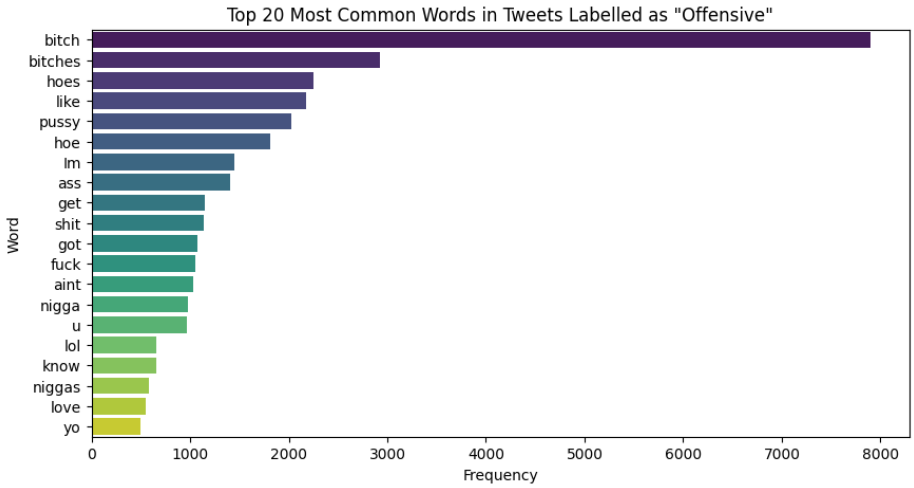

Figure 4.6: 20 Most Commonly Used Words in Offensive Tweets

There is a lexical overlap between the word clouds of offensive tweets and hateful tweets, including derogatory terms and explicit language. However, the intent is different. Hate speech tweets include more words related to identity-based attacks, for instance, racial, religious, and gender-based slurs. In contrast, offensive tweets seem to contain strong profanity but for the most part, lack clear targeted attacks against certain groups. This further reinforces the fact that profanity does not necessarily equal hate speech, as stated earlier in this thesis. Some words appear in both word clouds (e.g., “bitch,” “nigga,” “fuck”), showing the importance of context in distinguishing hate speech from general offensive language. Lastly, words such as “love” and “lol,” commonly associated with positive sentiments, appear frequently, showing that offensive speech can contain casual or non-hateful elements.

Common Words Used in Normal Speech

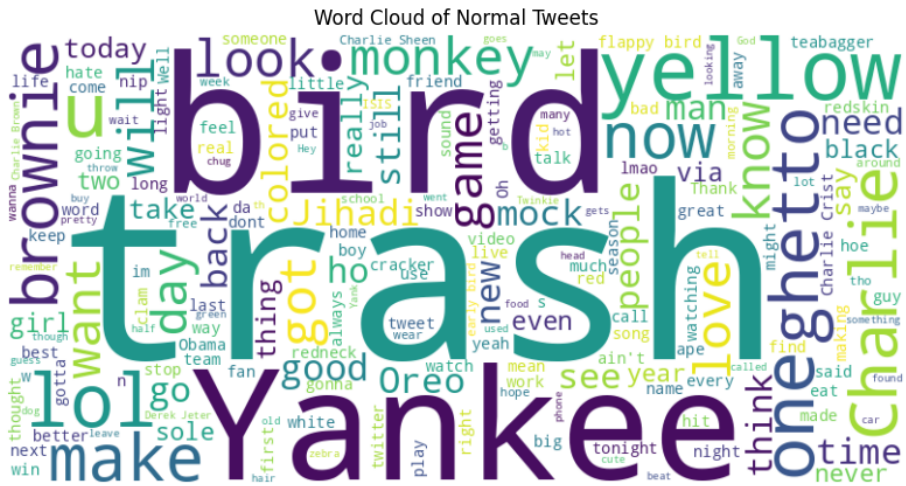

Figure 4.7: Word Cloud of Normal Tweets

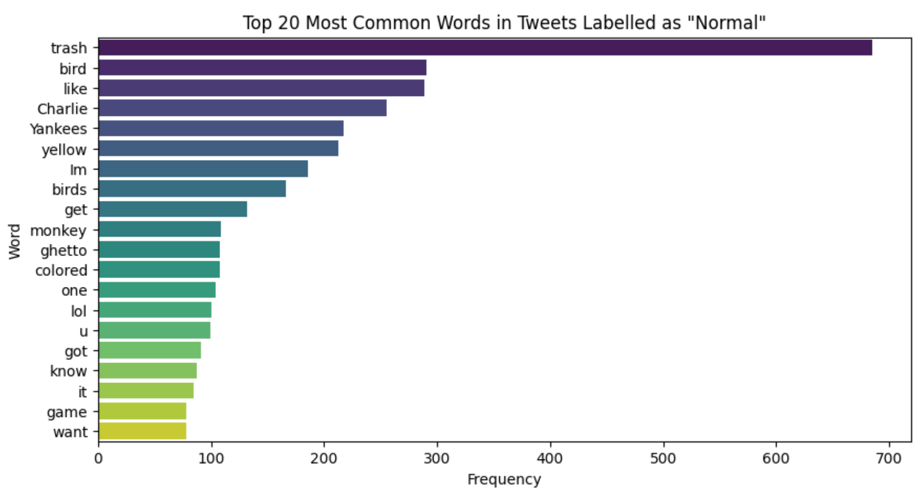

Figure 4.8: 20 Most Commonly Used Words in Normal Tweets

Words such as “trash,” “bird,” “Yankee,” “ghetto,” “monkey,” and “jihad,” appear prominently. This suggests that certain words, often associated with hate speech, also appear frequently in non-hateful contexts. For instance, “monkey” and “ghetto”

may be used in non-offensive ways, which can create challenges for automatic hate speech detection models to distinguish.

While the graph of normal tweets does not contain any profanity as they are, however, the words "monkey" and "coloured" were both used in several offensive contexts. The name "Charlie" was usually used to reference celebrities and people bearing the name (i.e. Charlie Puth, Charlie Sheen etc.), the cartoon character "Charlie Brown", and as a British slang for a "silly person."

4.4 Fine-Tuning Process

4.4.1 Data Preparation

Prior to fine-tuning the model, the dataset was split into training, testing, and evaluation sets employing stratified sampling. 2000 random entries from each sentiment class (offensive, normal, and hate) were selected for training the model, while 300 random examples from each class were used for both the test and validation sets. The final training set was shuffled to eliminate any unintended patterns that could influence the model's learning.

4.4.2 Prompt Engineering for Training and Evaluation

Given that the Llama 3.1 model is a causal-decoder transformer and thus based on causal language modelling (CLM), the classification task is phrased as a text-generation problem using prompt engineering. We designed prompts that will be fed into the model to classify text into one of the three sentiment categories. The prompt is as follows:

> "Examine the text enclosed in square brackets, determine if its sentiment is offensive, hate speech, or normal, and return the answer as the corresponding sentiment label 'offensive', 'hate', or 'normal'. [text] = sentiment"

To further illustrate this, an example from the training set is shown below:

> "Examine the text enclosed in square brackets, determine if its sentiment is offensive, hate speech, or normal, and return the answer as the corresponding sentiment label 'offensive', 'hate', or 'normal'. [This music is trash] = normal"

During training, each row of the training set is replaced by the prompt, with each example in the square brackets and its sentiment labels on the other side of the equation. During inference, the model was provided with similar prompts but without the sentiment label, requiring it to generate the correct classification based on learned patterns.

4.4.3 Model Selection and Quantization

For this stage, we used Meta-Llama-3.1-8B-Instruct from Hugging Face [138]. However, because of the large size of the model, we applied 4-bit NormalFloat (NF4) quantization to load the weights of the pre-trained model, instead of the usual 16-bit or 32-bit, using the `bitsandbytes` library [1]. This quantization approach was implemented based on the QLoRA paper [58] and is the extra step required to implement QLoRA (in place of LoRA). To further reduce memory footprint, a second quantization is applied, saving an extra 0.4 bits per parameter [58].

However, even though the model weights are stored in 4-bit precision, computations (such as matrix multiplications during forward or backward passes) are done in float16 precision. This dequantization step during computations allows for more stable and accurate operations. To implement the steps outlined above, we used the `BitsAndBytesConfig` function of the `bitsandbytes` library with the hyperparameters in Figure 4.9.

```
model_name = "meta-llama/Meta-Llama-3.1-8B-instruct" # from hf

# Setting compute data type to 16-bit fp to maintain a reasonable precision in computations
compute_dtype = getattr(torch, "float16")

# Quantizing model weights to 4-bit precision using 4-bit NormalFloat
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=True,
)

# Loading the pretrained model for causal language modeling with the 4-bit quantization configuration
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map=device,
    torch_dtype=compute_dtype,
    quantization_config=bnb_config,
)
```

Figure 4.9: Model Loading with bitsandbytes

4.4.4 Fine-Tuning with LoRA

Before delving deeper into fine-tuning, we will introduce some vocabulary related to deep learning models which are used in the rest of this chapter.

- **Loss function:** This is the difference between the predicted and actual values of a single training example.
- **Cost function:** This is an indicator of how well a model performs. It is the difference between the predictions of the model and the actual values of the entire training dataset.

- **Parameters:** These are the adjustable parameters of a model during training. They are estimated during training and are dependent on the chosen dataset. Examples of parameters are the weights and biases of a neural network (a machine learning process consisting of nodes and units that mimics the way the neurons in the human brain process data).

- **Hyperparameters:** These are the values of a model manually set before training and are external to the model. Examples of hyperparameters are the number of layers in a neural network and the learning rate.

- **Learning rate:** This decides the speed at which a model converges on a solution. It is a parameter that determines how much the weights of a model are updated with each iteration in response to the error during training. A smaller learning rate leads to slower convergence and will require a lot of iterations before a solution is found. On the other hand, too large a learning rate will increase the speed of convergence but equally increase the chances of overshooting or missing the optimal solution.

- **Gradient descent:** Gradient descent is an optimization algorithm used to minimize the cost function by iteratively updating the model parameters. It calculates the gradient of the cost function for each parameter and adjusts the parameters in the direction that reduces the cost. In this process, it is crucial to select an appropriate learning rate, which controls the size of the steps taken during each iteration and determines the number of iterations needed for convergence.

To adapt the pre-trained Llama 3.1 model to our classification task, we employed

Low-Rank Adaptation (LoRA). The original 4-bit quantized weights are not modified during this stage. Instead, adapters (small trainable matrices) are added on top of the frozen weights (as explained in Chapter 3). The adapters are then trained while the main model remains unchanged.

LoRA was configured with the following hyperparameters, as shown in Table 4.4.

Table 4.4: LoRA Hyperparameters

Hyperparameter	**Value**
Rank	64
Alpha	16
Dropout	0.1
Bias	None
Target Modules	All-linear
Task Type	Causal_LM

Details of Hyperparameters

- **Rank:** The rank (r) determines the degree of parameter adaptation. In other words, it is the rank of the LoRA matrices and determines the number of trainable parameters introduced by LoRA. With 64, the model can learn task-specific adaptations, while being smaller than full finetuning.

- **Alpha:** Alpha is a scaling factor for the LoRA layers that controls the influence of LoRA updates/strength of adaptation. Essentially, it controls how much the adapted weights influence the original model. A higher lora_alpha value could lead to instability, while a lower one might underfit.

- **Dropout:** Dropout is used to prevent overfitting during training by randomly deactivating some neurons during training.
- **Bias:** This controls which bias terms are updated during fine-tuning of the model. The bias parameter can take the values of "none," "all," or "lora only" (for the biases related to the LoRA layers). With "none," the bias terms of the model remain unchanged, further reducing memory overhead as a result of a reduced number of updated parameters.
- **Target Modules:** This variable targets all linear layers of the model except the output layer, which is responsible for predictions.
- **Task Type:** This hyperparameter indicates that the chosen model is for causal language modeling (CLM), which is a primary function of decoder-based autoregressive models, like Llama.

4.4.5 Training and Evaluation

The training process for the model was done with Hugging Face's Trainer API, with key parameters shown in Table 4.5.

Table 4.5: Training Hyperparameters

Parameter	**Value**
Batch Size	8
Gradient Accumulation Steps	8
Gradient Optimizer	AdamW
Learning Rate	5e-4
Weight Decay	0.001
Epochs	1

Hyperparameter Definitions

- **Batch Size:** This is the number of samples processed by the model before updates are made to its weights. The smaller the batch size, the lower the memory required for training.

- **Gradient Accumulation Steps:** Gradients are indicators of how a model's parameters should change to reduce loss/error. The gradients are typically calculated using backpropagation and updated after each batch of data (determined by the "batch size" parameter). Afterward, they are used to update the weights of the model. However, with gradient accumulation, the gradients are computed and stored at each step (determined by the "gradient accumulation steps"), and the model's parameters are updated after the chosen number of batches is processed. In other words, the gradient accumulation steps hyperparameter decides the number of batches over which gradients are accumulated before performing a weight update. The gradients of each batch are summed,

an optimizer averages them, and the model parameters (weights) are updated, effectively improving memory efficiency.

- **Gradient Optimizer:** This is an optimization algorithm for updating a model's weights during training to minimize the loss function. The chosen method, AdamW [128], is an alternative to gradient descent and builds upon the original Adam (Adaptive Moment Estimation) [108]. It is preferred over stochastic gradient descent because of its superior convergence speed, stability, and ease of implementation.

- **Learning Rate:** This determines how much the model's weights are updated during training. It determines the step size taken in the direction of the gradients. Put differently, it controls how much the model updates its weight in response to the gradients during training.

- **Weight Decay:** This regularization technique penalizes the model for large weight updates, effectively preventing overfitting. Large weight updates could imply that the model is adapting too quickly to the training data and learning insignificant patterns instead of the underlying relationships.

- **Epoch:** This is the number of times the entire dataset is passed through the model. Since LoRA fine-tuning typically requires fewer updates, a single epoch may be enough to adjust the model.

After several iterations, we found the best accuracy and training speed with the parameters in Table 4.5. With an NVIDIA Tesla 4 Graphics Processing Unit (GPU), the model was trained in approximately 1 hour and 20 minutes.

Following the fine-tuning and training process, the trained LoRA adapters are merged with the quantized weights of the model, after which the model is used for inference on the test set. The predictions were made using a text-generation pipeline, where the model completed the prompt with the sentiment. Afterward, post-processing steps were taken to extract the answers from the prompt. The prompt in the test set is as follows:

> "Examine the text enclosed in square brackets, determine if its sentiment is offensive, hate speech, or normal, and return the answer as the corresponding sentiment label 'offensive', 'hate', or 'normal'. [text] = "

To further illustrate this, an example from the test set is shown below:

> "Examine the text enclosed in square brackets, determine if its sentiment is offensive, hate speech, or normal, and return the answer as the corresponding sentiment label 'offensive', 'hate', or 'normal'. [lets not forget the fact that everyone and their mother has yellow fever] = "

In contrast to the training set, the test set is blank on the right-hand side of the enclosed text. For evaluation, we generated a classification report to measure precision, recall, and F1 scores across sentiment classes. This will be discussed further in the next chapter.

4.5 Post-Training Quantization

In this section, we apply Generalized Post-Training Quantization (GPTQ) to the fine-tuned model to compress it fully into 4-bit precision using the Transformers

library [70]. Unlike the bitsandbytes quantization method used to load the model earlier, GPTQ precomputes the 4-bit weights, eliminating the need for dequantization during computations and significantly speeding up inference time.

Following the recommendations of the GPTQ paper [75], we calibrated the fine-tuned model with a sample of the C4 (Colossal Clean Crawled Corpus) dataset [167], which is widely used for language model calibration after PTQ. The C4 dataset contains a diverse set of about 750 GB worth of generic English text samples that can help approximate a model's input distribution. With the C4 dataset, the model's weights remain aligned with real-world language patterns.

The quantization process takes approximately 23 minutes to complete on a T4 GPU and is saved. Afterwards, the quantized model is loaded and used to make predictions on the test set. As done previously, we generated a classification report to measure precision, recall, and F1 scores across sentiment classes. This will be discussed in more detail in the next chapter.

4.6 Hardware

Fine-tuning was run on a Google Colaboratory notebook with an NVIDIA Tesla T4 15GB RAM GPU. Post-training quantization was performed on the same platform with an NVIDIA A100 40GB RAM GPU.

Chapter 5

Results and Discussions

5.1 Performance Metrics

Table 5.1: Common Performance Metrics Used for Classification Tasks

Metrics	**Definition**	**Formula**
Accuracy (A)	The ratio of correct predictions to the total number of predictions	Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
Precision (P)	The ratio of correctly predicted positive instances to all instances predicted as positive	Precision = $\frac{TP}{TP+FP}$
Recall (R)	The ratio of correctly predicted positive instances to the actual positive instances	Recall = $\frac{TP}{TP+FN}$
F1 Score (F1)	Harmonic mean of precision and recall	F1-score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Abbreviations: TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

Table 5.1 shows the performance metrics used in the study for measuring model performance.

5.2 Performance Before Fine-Tuning

To recap, the test set consists of 300 samples from each of the three classes: hate speech, offensive language, and neutral (normal) speech. Overall, the base Llama 3.1 model achieves a poor overall accuracy of 52% as seen in Table 5.2.

Surprisingly, the model performs decently in classifying normal speech with an F1 – score of 71%, but significantly worse in the other categories. This alludes to the fact that normal speech is easier to identify than hateful and offensive speech mainly due to the absence of vulgarity. Pertinently, the offensive speech class has a high recall of 95%, capturing most of the actual offensive speech texts. However, this comes at the cost of misclassifying any text with a hint of vulgarity or aggressiveness as offensive – including hate speech as seen in Figure 5.1. As a result, most of the hate speech samples are classified as offensive language resulting in abysmally low F1 and recall scores further underscoring the difficulty in distinguishing hate speech from offensive speech.

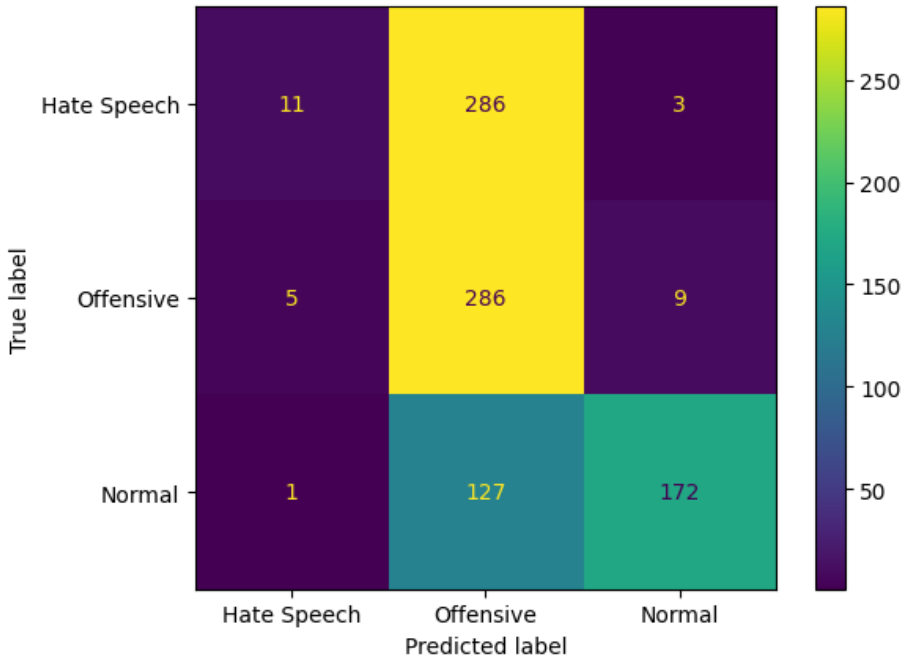

Figure 5.1: Confusion Matrix of Base Llama 3.1 Before Fine-Tuning

Table 5.2: Classification Report Before Fine-Tuning

	Precision	**Recall**	**F1-Score**	**Sample Size**
Hate	0.65	0.04	0.07	300
Offensive	0.41	0.95	0.57	300
Normal	0.93	0.57	0.71	300
Macro Average	0.66	0.52	0.45	900
Accuracy: 0.52				900

Table 5.3: Examples of Misclassified Texts

No.	**Text**	**True Label**	**Predicted Label**	**My Opinion**
1	This song may be for basic bitches but it's such a mood boost	Offensive	Normal	Offensive
2	#Allah can KISS MY ASS! And yes, MUHAMMED was a bitch! Just like OSAMA BINLADEN!	Offensive	Hate	Offensive
3	All Babies are born hipsters because they come out of the hospital in beanies.	Offensive	Normal	Normal
4	trump should nuke pakistan instead of iran pakiland dirtyland britishslaveland punjabiland	Hate	Offensive	Hate
5	Cuke, melon, or squash. They all look the same at this point	Hate	Normal	Normal

Continued on next page

Table 5.3 – continued from previous page

No.	Text	True Label	Predicted Label	My Opinion
6	stfu hoe "I'm proud to be African American" - Proud African American "I'm proud to be white" - Racist	Hate	Normal	Offensive
7	You are without a doubt the most vile pathetic disgusting disgusting racist radical. Go back to Somalia and leave our country alone. Screw your jihadist loving attitude. You can leave anytime you want and we hope it's real soon. Bye bye.	Hate	Offensive	Offensive
8	Niggas may talk more than bitches these days but hunny booboo bitches is sneakier!	Hate	Offensive	Offensive

Continued on next page

Table 5.3 – continued from previous page

No.	**Text**	**True Label**	**Predicted Label**	**My Opinion**
9	Black Americans Are Worse Off Under BHO Well researched article...bound to elicit cries of Uncle Tom	Normal	Offensive	Normal
10	I likes da Yankees	Normal	Hate	Normal
11	Charlie Strong WILL have Texas back as a power better power of big is anybody's now	Normal	Offensive	Normal

Insights from Table 5.3:

Table 5.3 highlights some examples of misclassified text and includes my opinion as a Black male in his mid-twenties.

In the first example, the word "bitch" which is generally considered offensive is used humorously and in a subjectively harmless context leading the model to classify it as normal. The fourth example further demonstrates the model's prioritization of recall for offensive speech over hate speech. Examples 9–11 appear to have been misclassified by the base model because of the presence of politically sensitive terms. Additionally, the model seems to be misinterpreting context and informal language as an indicator of offensive or hate speech due to syntactical or stylistic biases.

5.3 Fine-Tuning to Improve Performance

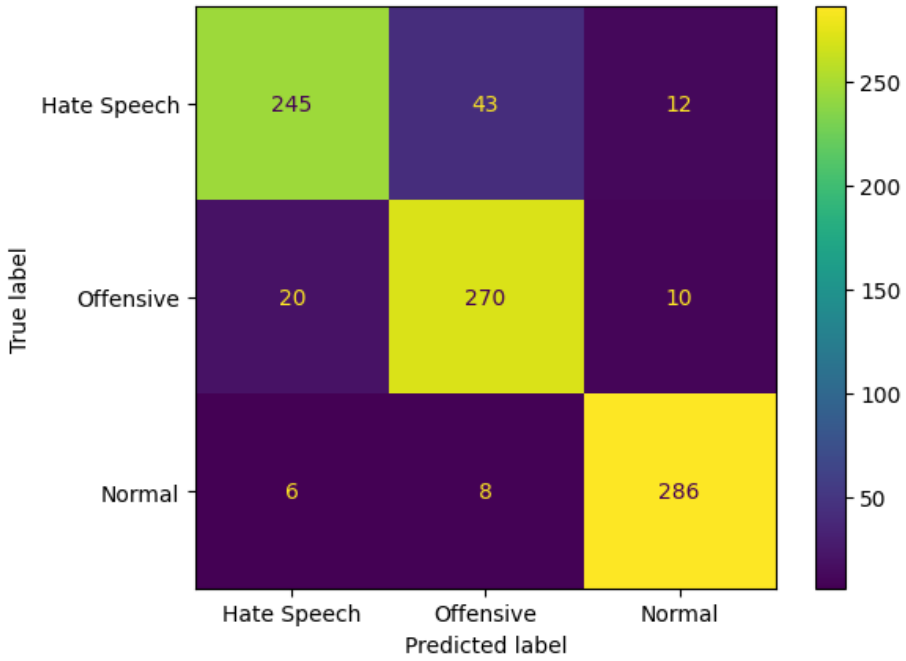

Figure 5.2: Confusion Matrix After Fine-Tuning

Table 5.4: Classification Report of Fine-Tuned Model

	Precision	**Recall**	**F1-Score**	**Sample Size**
Hate	0.90	0.82	0.86	300
Offensive	0.84	0.90	0.87	300
Normal	0.93	0.95	0.94	300
Macro Average	0.89	0.89	0.89	900
Accuracy: 0.89				900

As expected, after the fine-tuning, the trained model achieved strong classification performance as seen in Figure 5.2, and Table 5.4. The precision of the "normal" class remains the same as the base model, while the recall of the "offensive" class drops 5%. Nevertheless, there is a significant bump in every other metric and class, including the overall accuracy of the model which jumps from 52% to 89%.

It is also worth noting that most of the misclassified examples are concentrated in the "hate speech" and "offensive" classes. While the fine-tuned model showed a significant improvement in classification of these classes over the base model, it might be easy to assume that there is still some difficulty in distinguishing these two classes. However, upon exploration of the misclassified examples, it was observed that many would be difficult for even humans to distinguish. Some of these misclassified examples are shown below:

Table 5.5: Some Misclassified Texts

No.	**Text**	**True Label**	**Predicted Label**	**My Opinion**
1	lmaooo roosters is hot trash bro u know it	Offensive	Normal	Normal
2	I'm late as hell....Niggas really hating cuz these hoes want that lightskin dude in jail y'all crazy	Offensive	Hate	Offensive
3	oh....I know dude. I just went OFF ON HIS ASS I'm done with that faggot	Offensive	Hate	Offensive
4	if ive ever been a faggot to you let me know and ill change :)	Offensive	Hate	Offensive
5	I know you bitches is hating on the low. Why me tho? Its way more niggaz in this world that can afford it.	Offensive	Hate	Offensive

Continued on next page

Table 5.5 – continued from previous page

No.	Text	True Label	Predicted Label	My Opinion
6	Dat bitch Katy Perry Can Suck Skin Off Of my Dick	Hate	Offensive	Offensive
7	Girl and yea man niggas be blew you got da bitch errybody tryna fuck and just start sayin anything want and we hope it's real soon. Bye bye.	Hate	Offensive	Offensive
8	Built for eternity - "You adulterous people! Don't you know that friendship with the world is hostility toward God?" James 4:4 #NotOfThisWorld	Hate	Normal	Normal
9	Yo fuk that bitch	Hate	Offensive	Offensive

Continued on next page

Table 5.5 – continued from previous page

No.	Text	True Label	Predicted Label	My Opinion
10	anti American Republicans and teabaggers stop forcing ur religion on the rest of us. Stockman u dont pray.liar	Normal	Hate	Offensive
11	I hate when females try to sound like bird heads Charlie Strong WILL have Texas back as a power better power of big is anybody's now	Normal	Hate	Offensive
12	Tilt my chin back, slit my throat, take a bath in my blood, get to know me	Normal	Hate	Offensive
13	My nicca Drake gear game is all bad lol	Normal	Offensive	Offensive

A common trend observed in Table 5.5 is the model's difficulty in distinguishing between offensive and hateful posts, particularly those containing the words "nigga" and "faggot." This challenge largely stems from contextual nuances. For example, I believe that posts 2, 4, 5, and 7 are just offensive, despite the model and annotators labeling them as hate speech. According to the criteria established in this thesis, hate speech explicitly targets individuals or groups based on their identity. In contrast, these posts appear to be made by individuals from the referenced communities, potentially as a form of humorous self-deprecation or reclaiming the language. Within these communities, such words are often used in a non-offensive manner.

5.4 Performance After Generalized Post-Training Quantization (GPTQ)

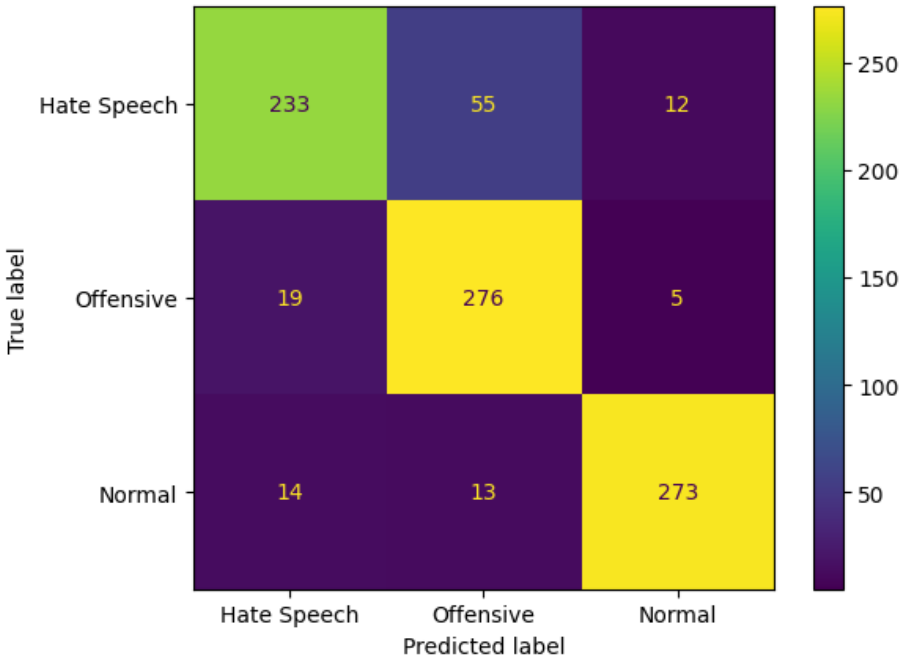

Figure 5.3: Confusion Matrix After Quantization

Table 5.6: Classification Report of Quantized Model

	Precision	**Recall**	**F1-Score**	**Sample Size**
Hate	0.88	0.78	0.82	300
Offensive	0.80	0.92	0.86	300
Normal	0.94	0.91	0.93	300
Macro Average	0.87	0.87	0.87	900
Accuracy: 0.87				900

Table 5.7: Classification Report Comparison of Fine-Tuned and Quantized Models

	Precision		**Recall**		**F1-Score**	
	Pre-GPTQ	**GPTQ**	**Pre-GPTQ**	**GPTQ**	**Pre-GPTQ**	**GPTQ**
Hate	0.90	0.88	0.82	0.78	0.86	0.82
Offensive	0.84	0.80	0.90	0.92	0.87	0.86
Normal	0.93	0.94	0.95	0.91	0.94	0.93
Macro Average	0.89	0.87	0.89	0.87	0.89	0.87
Accuracy:	0.89	0.87				

As seen in Table 5.7, there is a slight downgrade in the overall accuracy of the model from 89% to 87%. Likewise, a similar percentage decrease in performance is observed across the other metrics except for the precision of the normal class. This decrease across the board will lead to an increase in false positives and negatives during the classification of hate speech which could be detrimental depending on how sensitive the application is.

5.5 Statistical Analysis of Results

We run experiments two more times on the base, quantized, and fine-tuned-only models using the same dataset and hardware to observe the differences in accuracy and speed. To compare the results of the tests, we use One-factor designs [102], displayed in Table 5.8.

Table 5.8: One-factor Design Method

	Base Model	**Fine-tuned-only Model**	**Quantized Model**	**Mean**
Test 1	y_{11}	y_{12}	y_{13}	
Test 2	y_{21}	y_{22}	y_{23}	
Test 3	y_{31}	y_{32}	y_{33}	
Column sum	$\sum y_{\cdot 1}$	$\sum y_{\cdot 2}$	$\sum y_{\cdot 3}$	$\sum y_{\cdot\cdot}$
Column mean	$\bar{y}_{\cdot 1}$	$\bar{y}_{\cdot 2}$	$\bar{y}_{\cdot 3}$	$\mu = \bar{y}_{\cdot\cdot}$
Column effect	$\alpha_1 = \bar{y}_{\cdot 1} - \bar{y}_{\cdot\cdot}$	$\alpha_2 = \bar{y}_{\cdot 2} - \bar{y}_{\cdot\cdot}$	$\alpha_3 = \bar{y}_{\cdot 3} - \bar{y}_{\cdot\cdot}$	

The results of this test are shown in Table 5.9.

Table 5.9: Comparative Analysis of Inference Time (900 posts)

	Base Model (Mins)	**Fine-tuned-only Model (Mins)**	**Quantized Model (Mins)**	**Mean**
Test 1	7.53	6.78	4.65	

Continued on next page

Table 5.9 – continued from previous page

	Base Model (Mins)	**Fine-tuned-only Model (Mins)**	**Quantized Model (Mins)**	**Mean**
Test 2	6.30	6.70	4.30	
Test 3	6.68	6.72	4.72	
Column Sum	20.51	20.20	13.67	
Column Mean	6.84	6.73	4.56	6.04
Column Effect	0.80	0.69	-1.48	

The standard deviation of errors is computed as:

$$S_e = \sqrt{\frac{\sum_{i=1}^{a}\sum_{j=1}^{r} y_{ij}^2 - ar\mu^2 - r\sum_{j=1}^{a}\alpha_j^2}{a(r-1)}} \tag{5.1}$$

Where:

- a = number of algorithms (alternatives)
- r = number of tests (observations)

To check for significant differences in performance between any two models, we compute confidence intervals for the differences in effect values:

$$\text{Mean difference } \alpha_i - \alpha_j = \bar{y}_{.i} - \bar{y}_{.j} \tag{5.2}$$

$$\text{Std. dev. } \alpha_i - \alpha_j = S_e \cdot \sqrt{\frac{2}{ar}} \tag{5.3}$$

Given $a(r-1) = 6$ degrees of freedom, and a 90% confidence interval, the t-value is 1.943. Using this, we calculate:

- $S_e = 0.45$
- $\alpha_2 - \alpha_3 = 2.17$, standard deviation $= 0.212$
- $\alpha_1 - \alpha_3 = 2.28$, standard deviation $= 0.212$

$$\text{CI for } \alpha_2 - \alpha_3 = 2.17 \pm 1.943 \times 0.212 = (1.7580,\ 2.5819)$$

$$\text{CI for } \alpha_1 - \alpha_3 = 2.28 \pm 1.943 \times 0.212 = (1.8681,\ 2.6919)$$

Since neither confidence interval contains 0, the quantized model is significantly faster than both the base and fine-tuned-only models.

Table 5.10: Comparative Analysis of Accuracy

	Base Model (%)	**Fine-tuned-only Model (%)**	**Quantized Model (%)**	**Mean**
Test 1	52.1	89.0	86.9	
Test 2	51.2	88.1	87.4	
Test 3	51.0	88.6	85.4	
Column Sum	154.3	265.7	259.7	
Column Mean	51.4	88.6	86.6	75.5

Continued on next page

Table 5.10 – continued from previous page

	Base Model (%)	**Fine-tuned-only Model (%)**	**Quantized Model (%)**	**Mean**
Column Effect	-24.1	13.1	11.1	

Using the same methodology:

- $S_e = 1.99$
- $\alpha_2 - \alpha_3 = 2$, standard deviation = 0.938
- $\alpha_1 - \alpha_3 = -35.1$, standard deviation = 0.938

$$\text{CI for } \alpha_2 - \alpha_3 = 2 \pm 1.943 \times 0.938 = (0.1775,\ 3.8225)$$

$$\text{CI for } \alpha_1 - \alpha_3 = -35.1 \pm 1.943 \times 0.938 = (-36.9225,\ -33.2775)$$

Since neither interval contains 0, we conclude there are statistically significant differences. The quantized model has significantly higher accuracy than the base model but slightly lower than the fine-tuned-only model.

5.6 Model Size and Time Consumption

Table 5.11: Performance Summary of the Models

Metrics	**Model without Quantization**			**Quantized Model**		
	Test 1	**Test 2**	**Test 3**	**Test 1**	**Test 2**	**Test 3**
Inference Time for 900 posts (mins)	6.78	6.70	6.72	4.65	4.30	4.72
Accuracy (%)	89.0	88.1	88.6	86.9	87.4	85.4
Total Training Time (mins)	154	148	153	177	171	176
Model Size (GB)	16.9			5.34		

Table 5.12: Performance Summary of the Models (Averages)

Metrics (Average)	**Model without Quantization**	**Quantized Model**	**Change (%)**
Inference Time (mins)	6.73	4.56	-32.24
Accuracy (%)	88.57	86.57	-2.26
Total Training Time (mins)	151.67	174.67	+13.17
Model Size (GB)	16.9	5.34	-66.95

The performance of both models is summarized in Table 5.12. As shown, several trade-offs exist. Starting with the positives, there is a drastic reduction in model size which is one of the main advantages of quantization. This makes the model more lightweight, efficient, and suitable for use on hardware with limited storage. Another

positive was the significant reduction in inference time. With quantization, the model achieves a significant boost in speed and is able to classify 900 posts about 1.5 times faster. This speed boost can be attributed to the model's reduced size, allowing for faster and quicker response times.

On the other hand, the model's accuracy dropped by approx. 2%. While this might seem like a slight drop depending on the application, it could be critical in hate speech detection. Missing hate speech (False Negatives) could allow harmful content to spread, potentially impacting targeted communities. From another point of view, false positives can lead to unnecessary censorship and suppression of free speech. Thus, this drop in accuracy could impact the overall fairness and reliability of a model.

It is also worth considering the extra time it takes to quantize a model. In our scenario, this added approx. 23 minutes to the training time, which could vary depending on the hardware used and the model's architecture. Another factor to consider is the need for a more powerful GPU to quantize the model. These add more layers of complexity to the whole setup but could be justified when the inference speed gains are significant as in our case. Overall, there are several trade-offs to consider in selecting the right model and the right choice will depend on the domain and the sensitivity of the subject matter.

Chapter 6

Conclusion

6.1 Summary

Hate speech detection is a nuanced and complex topic. Even more challenging is distinguishing hate speech from offensive speech, yet critical for building responsible AI systems. This thesis showcased the ability of LLMs to differentiate the two classes from each other. Furthermore, we explored post-training quantization as a way to optimize our chosen Llama 3.1 LLM, successfully reducing its size by 67% and inference time by 32%. However, this came with a 2% drop in model accuracy accompanied by an additional 23 minutes in training time to optimize the model. This highlights the inevitable trade-offs between model efficiency and predictive performance.

6.2 Limitations

There were several limitations to the approach taken in this study.

1. **Dataset Representativeness and Bias:** The datasets we used in this thesis are well-known but limited in scope. They are in English and mostly based on

social media posts originating from the United States and thus heavily influenced by American culture and contexts. A glaring example of this bias is the fact that one of the most used words in the normal speech class was "Yankee," referring to the New York Yankees baseball team. A potential problem with this is that the model may incorrectly generalize such culturally specific terms as universally neutral or positive, which could lead to misclassification.

2. **Performance on Other Datasets:** Another problem with the poor representativeness of the dataset is that the model will inevitably perform poorly on datasets from countries with different languages, cultures and perceptions of hate speech. Even within English-speaking communities, there are substantial linguistic variations and a multitude of English dialects and derivatives (like Jamaican Patois and Nigerian Pidgin) that the model has not been exposed to. Furthermore, phenomena like code-switching (mixing languages) and the frequent use of evolving slang add additional complexity that the model, trained on a narrow cultural and linguistic base, may not handle robustly.

3. **Fine-Grained Label Ambiguity:** As seen in the discussions in Chapter 5, even human annotators struggle to agree on what constitutes hate speech and offensive speech. This misalignment is also reflected in the model and limits its performance.

4. **Loss in Accuracy from Quantization:** While the drop in accuracy after quantization seems minimal (2%), this could lead to ethical risks in sensitive applications like hate speech detection. Having an increased false negative rate could lead to missed hate speech, while an increase in false positives could lead

to over-censorship.

5. **Computational and Time Costs for Optimization:** While quantization reduces inference costs, it introduces additional computational overhead. Depending on how big the dataset is or how frequently the model needs to be updated, this might ramp up monetary and time costs significantly. Furthermore, despite implementing optimization techniques, fine-tuning still requires significant computational power which can limit accessibility for smaller organizations and researchers.

6.3 Future Research Directions

The current work focused on English Datasets. Future work could extend to multilingual social media contexts and the proposed models evaluated on adversarial datasets to test robustness and limitations. Future research could also explore the implementation of Retrieval Augmented Generation (RAG) or a combination of RAG + fine-tuning for hate speech detection.

Bibliography

[1] bitsandbytes. https://huggingface.co/docs/bitsandbytes/main/en/index.

[2] Open release of grok-1. https://x.ai/blog/grok-os.

[3] Sentiment analysis — comprehensive beginners guide — thematic. https://getthematic.com/sentiment-analysis.

[4] Accelerating a hugging face llama 2 and llama 3 models with transformer engine — transformer engine 2.4.0 documentation, 2017. https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/te_llama/tutorial_accelerate_hf_llama_with_te.html.

[5] Ai saving humans from the emotional toll of monitoring hate speech — waterloo news, 05 2024. https://uwaterloo.ca/news/media/ai-saving-humans-emotional-toll-monitoring-hate-speech.

[6] Gptq, 2025. https://huggingface.co/docs/transformers/en/quantization/gptq.

[7] Hyperskill, 2025. https://hyperskill.org/learn/step/10403.

[8] S. Abro, S. Shaikh, Z. Hussain, Z. Ali, S. Khan, and G. Mujtaba. Automatic hate speech detection using machine learning: A comparative study. *International Journal of Advanced Computer Science and Applications*, 11, 2020.

[9] U. Ahmed, R. Mumtaz, H. Anwar, A. A. Shah, R. Irfan, and J. García-Nieto. Efficient water quality prediction using supervised machine learning. *Water*, 11:2210, 11 2019.

[10] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv:2305.13245*, 10 2023.

[11] Y. Al Amrani, M. Lazaar, and K. E. El Kadiri. Random forest and support vector machine based hybrid approach to sentiment analysis. *Procedia Computer Science*, 127:511–520, 2018.

[12] A. Al-Hassan and H. Al-Dossari. Detection of hate speech in arabic tweets using deep learning. *Multimedia Systems*, 01 2021.

[13] H. Alami, Said, A. Benlahbib, and N. En-Nahnahi. Lisac fsdm-usmba team at semeval-2020 task 12: Overcoming arabert's pretrain-finetune discrepancy for arabic offensive language identification. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, 01 2020.

[14] J. Alammar and M. Grootendorst. *Hands-On Large Language Models*. O'Reilly, 12 2024.

[15] H. S. Alatawi, A. M. Alhothali, and K. M. Moria. Detecting white supremacist hate speech using domain specific word embedding with deep learning and bert. *IEEE Access*, 9:106363–106374, 2021.

[16] N. Albadi, M. Kurdi, and S. Mishra. Are they our brothers? analysis and detection of religious hate speech in the arabic twittersphere, 08 2018.

[17] M. Z. Ali, Ehsan-Ul-Haq, S. Rauf, K. Javed, and S. Hussain. Improving hate speech detection of urdu tweets using sentiment analysis. *IEEE Access*, 9:84296–84305, 2021.

[18] R. A. R. Ali, U. Farooq, U. Arshad, W. Shahzad, and M. O. Beg. Hate speech detection on twitter using transfer learning. *Computer Speech and Language*, page 101365, 02 2022.

[19] S. Almohaimeed, S. Almohaimeed, and L. Bölöni. Transfer learning and lexicon-based approaches for implicit hate speech detection: A comparative study of human and gpt-4 annotation. pages 142–147, 02 2024.

[20] Anjum and R. Katarya. Hate speech, toxicity detection in online social media: a recent survey of state of the art and opportunities. *International Journal of Information Security*, 23, 09 2023.

[21] W. Antoun, F. Baly, and H. Hajj. Arabert: Transformer-based model for arabic language understanding. *ACL Anthology*, pages 9–15, 05 2020.

[22] R. Arun. Gpt vs. llama: Comparing the inner workings of two transformer titans, 03 2025.

[23] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 07 2016.

[24] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma. Deep learning for hate speech detection in tweets. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, 2017.

[25] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 09 2014.

[26] F. Barbieri, L. E. Anke, and J. Camacho-Collados. Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond. *ACL Anthology*, pages 258–266, 06 2022.

[27] R. Bartusiak, L. Augustyniak, T. Kajdanowicz, and P. Kazienko. Sentiment analysis for polish using transfer learning approach. In *2015 Second European Network Intelligence Conference*, 09 2015.

[28] E. Bastı, C. Kuzey, and D. Delen. Analyzing initial public offerings' short-term performance using decision trees and svms. *Decision Support Systems*, 73:15–27, 05 2015.

[29] S. Behdenna, F. Barigou, and G. Belalem. Sentiment analysis at document level. *Communications in Computer and Information Science*, pages 159–168, 2016.

[30] M. Behzadi and I. G. Harris. Implicit hate speech detection using mistral with low rank adaptation, 09 2024.

[31] M. Bhardwaj, M. S. Akhtar, A. Ekbal, A. Das, and T. Chakraborty. Hostility detection dataset in hindi. *arXiv:2011.03588*, 01 2020.

[32] M. Bhatia, T. S. Bhotia, A. Agarwal, P. Ramesh, S. Gupta, K. Shridhar, F. Laumann, and A. Dash. One to rule them all: Towards joint indic language hate speech detection. *arXiv:2109.13711*, 09 2021.

[33] M. Bilal, A. Khan, S. Jan, S. Musa, and S. Ali. Roman urdu hate speech detection using transformer-based model for cyber security applications. *Sensors*, 23:3909–3909, 04 2023.

[34] S. Bird, E. Klein, and E. Loper. Natural language toolkit (nltk), 2009. https://www.nltk.org/.

[35] M. Birjali, M. Kasri, and A. Beni-Hssane. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, 226:107134, 08 2021.

[36] R. Bose, R. K. Dey, S. Roy, and D. Sarddar. Sentiment analysis on online product reviews. *Information and Communication Technology for Sustainable Development*, pages 559–569, 06 2019.

[37] V. L. Brescoll. Leading with their hearts? how gender stereotypes of emotion lead to biased evaluations of female leaders. *The Leadership Quarterly*, 27:415–428, 06 2016.

[38] T. Brown, B. F. Mann, N. Ryder, M. Subbiah, J. Kaplan, and P. Dhariwal *et al.* Language models are few-shot learners. *arXiv:2005.14165)*, 4, 05 2020.

[39] P. Burnap, O. F. Rana, N. Avis, M. Williams, W. Housley, A. Edwards, J. Morgan, and L. Sloan. Detecting tension in online communities with computational twitter analysis. *Technological Forecasting and Social Change*, 95:96–108, 06 2015.

[40] T. Caselli, V. Basile, J. Mitrović, and M. Granitzer. Hatebert: Retraining bert for abusive language detection in english. *arXiv.2010.12472*, 10 2020.

[41] L. Chen, P. Chen, and Z. Lin. Artificial intelligence in education: a review. *IEEE Access*, 8:75264–75278, 04 2020.

[42] L.-S. Chen, C.-H. Liu, and H.-J. Chiu. A neural network based approach for sentiment classification in the blogosphere. *Journal of Informetrics*, 5:313–322, 04 2011.

[43] K.-L. Chiu, A. Collins, and R. Alexander. Detecting hate speech with gpt-3. *arXiv:2103.12407*, 03 2021.

[44] K.-H. Choi, L. Bowleg, and T. B. Neilands. The effects of sexism, psychological distress, and difficult sexual situations on u.s. women's sexual risk behaviors. *AIDS Education and Prevention*, 23:397–411, 10 2011.

[45] C. Christodoulou. Nlpdame at climateactivism 2024: Mistral sequence classification with peft for hate speech, targets and stance event detection. In *Proceedings of the 7th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2024)*, pages 96–104. Association for Computational Linguistics, 03 2024.

[46] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, and W. Fedus *et al.* Scaling instruction-finetuned language models. *arXiv:2210.11416*, 10 2022.

[47] S. Cohen, D. Presil, O. Katz, O. Arbili, S. Messica, and L. Rokach. Enhancing social network hate detection using back translation and gpt-3 augmentations during training and test-time. *Information Fusion*, 99:101887, 11 2023.

[48] B. M. Coimbra, C. M. Hoeboer, J. Yik, A. F. Mello, M. F. Mello, and M. Olff. Meta-analysis of the effect of racial discrimination on suicidality. *SSM - Population Health*, 20:101283, 12 2022.

[49] B. Csanády, L. Muzsai, P. Vedres, Z. Nádasdy, and A. Lukács. Llambert: Large-scale low-cost data annotation in nlp. *arXiv:2403.15938v1*.

[50] A. C. Curry, G. Abercrombie, and V. Rieser. Convabuse: Data, analysis, and benchmarks for nuanced detection in conversational ai. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 01 2021.

[51] D-Lab. ucberkeley-dlab/measuring-hate-speech · datasets at hugging face, 06 2023. https://huggingface.co/datasets/ucberkeley-dlab/measuring-hate-speech.

[52] A. Das, J. S. Wahi, and S. Li. Detecting hate speech in multi-modal memes. *arXiv:2012.14891*, 01 2020.

[53] I. Data and A. Team. Ai vs. machine learning vs. deep learning vs. neural networks — ibm, 05 2024. https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

[54] T. Davenport and R. Kalakota. The potential for artificial intelligence in healthcare. *Future Healthcare Journal*, 6:94–98, 2019.

[55] T. Davidson. Automated hate speech detection and the problem of offensive language, 10 2021. https://github.com/t-davidson/hate-speech-and-offensive-language.

[56] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11:512–515, 05 2017.

[57] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv:2208.07339*, 11 2022.

[58] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv:2305.14314*, 2023.

[59] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:/1810.04805*, 05 2019.

[60] G. Dey, A. V. Ganesan, Y. K. Lal, M. Shah, S. Sinha, M. Matero, S. Giorgi, V. Kulkarni, and S. H. Andrew. Socialite-llama: An instruction-tuned model for social scientific tasks. *arXiv:2402.01980*, 02 2024.

[61] M. Di Capua, E. Di Nardo, and A. Petrosino. Unsupervised cyber bullying detection in social networks. *2016 23rd International Conference on Pattern Recognition (ICPR)*, 12 2016.

[62] K. Dinakar, B. Jones, C. Havasi, H. Lieberman, and R. Picard. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Transactions on Interactive Intelligent Systems*, 2:1–30, 09 2012.

[63] J. Ding, B. Li, C. Xu, Y. Qiao, and L. Zhang. Diagnosing crop diseases based on domain-adaptive pre-training bert of electronic medical records. *Applied Intelligence*, 53:15979–15992, 12 2022.

[64] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications*, 118:272–299, 03 2019.

[65] S. Dowlagar and R. Mamidi. Hasocone@fire-hasoc2020: Using bert and multilingual bert models for hate speech detection. *arXiv:2101.09007*, 01 2021.

[66] A. G. d'Sa, I. Illina, and D. Fohr. Classification of hate speech using deep neural networks. *Revue d'Information Scientifique and Technique*, 25, 12 2020.

[67] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, and A. Letman *et al.* The llama 3 herd of models. *arXiv:2407.21783*, 07 2024.

[68] R. Duwairi, A. Hayajneh, and M. Quwaider. A deep learning framework for automatic detection of hate speech embedded in arabic tweets. *Arabian Journal for Science and Engineering*, 02 2021.

[69] P. Ebrahimi, M. Basirat, A. Yousefi, M. Nekmahmud, A. Gholampour, and M. Fekete-Farkas. Social networks marketing and consumer purchase behavior: The combination of sem and unsupervised machine learning approaches. *Big Data and Cognitive Computing*, 6:35, 03 2022.

[70] H. Face. Quantize transformers models, 2018. https://huggingface.co/docs/transformers/v4.35.0/main_classes/quantization.

[71] R. Fakoor, F. Ladhak, A. Nazi, and M. Huber. Using deep learning to enhance cancer diagnosis and classification. 06 2013.

[72] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang. Language-agnostic bert sentence embedding. *arXiv:2007.01852*, 01 2020.

[73] J. Ferrer. How transformers work: A detailed exploration of transformer architecture, 01 2024.

[74] P. Ficamos, Y. Liu, and W. Chen. A naive bayes and maximum entropy approach to sentiment analysis: Capturing domain-specific data in weibo. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 02 2017.

[75] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv:2210.17323*, 03 2023.

[76] R. P. França, A. C. Borges Monteiro, R. Arthur, and Y. Iano. An overview of deep learning in big data, image, and signal processing in the modern digital age. *Trends in Deep Learning Methodologies*, pages 63–87, 2021.

[77] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv:2312.10997*, 12 2023.

[78] T. Garg, S. Masud, T. Suresh, and T. Chakraborty. Handling bias in toxic speech detection: A survey. *ACM Computing Surveys*, 01 2023.

[79] E. Georgiadou, S. Angelopoulos, and H. Drake. Big data analytics and international negotiations: Sentiment analysis of brexit negotiating outcomes. *International Journal of Information Management*, 51:102048, 12 2019.

[80] d. Gibert, N. Perez, A. García-Pablos, and M. Cuadros. Hate speech dataset from a white supremacy forum. *arXiv:1809.04444*, 01 2018.

[81] N. D. Gitari, Z. Zhang, H. Damien, and J. Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10:215–230, 04 2015.

[82] A. Glazkova, M. Kadantsev, and M. Glazkov. Fine-tuning of pre-trained transformers for hate, offensive, and profane content detection in english and marathi. *arXiv:2110.12687*, 10 2021.

[83] L. Grotti and P. Quick. Berticelli at haspeede 3: Fine-tuning and cross-validating large language models for hate speech detection. *EVALITA 2023*, 09 2023.

[84] R. Gupta, D. Srivastava, M. Sahu, S. Tiwari, R. K. Ambasta, and P. Kumar. Artificial intelligence to deep learning: machine intelligence approach for drug discovery. *Molecular Diversity*, 25:1–46, 04 2021.

[85] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv:2004.10964*, 04 2020.

[86] M. A. Hassonah, R. Al-Sayyed, A. Rodan, A. M. Al-Zoubi, I. Aljarah, and H. Faris. An efficient hybrid filter and evolutionary wrapper approach for sentiment analysis of various topics on twitter. *Knowledge-Based Systems*, 192:105353, 03 2020.

[87] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv:2110.04366*, 10 2021.

[88] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 12 2015.

[89] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 07 2006.

[90] L. Hong, P. Luo, E. Blanco, and X. Song. Outcome-constrained large language models for countering hate speech. *arXiv:2403.17146*, 03 2024.

[91] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv:2106.09685*, 01 2021.

[92] Z. Hu, Y. Lan, L. Wang, W. Xu, E.-P. Lim, R. K.-W. Lee, L. Bing, and S. Poria. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv:2304.01933*, 04 2023.

[93] A. Hussain and E. Cambria. Semi-supervised learning for big social data analysis. *Neurocomputing*, 275:1662–1673, 01 2018.

[94] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8:216–225, 05 2014.

[95] IBM. What is unsupervised learning? — ibm, 2023. https://www.ibm.com/topics/unsupervised-learning.

[96] M. O. Ibrohim and I. Budi. Multi-label hate speech and abusive language detection in indonesian twitter. In *Proceedings of the Third Workshop on Abusive Language Online*, page 46–57. ACLWeb, Association for Computational Linguistics, 08 2019.

[97] C. E. Ilevbare, J. O. Alabi, D. I. Adelani, F. D. Bakare, O. B. Abiola, and O. A. Adeyemo. Ekohate: Abusive language and hate speech detection for code-switched political discussions on nigerian twitter. *arXiv:2404.18180*, 04 2024.

[98] M. Imran, S. Latif, D. Mehmood, and M. S. Shah. Student academic performance prediction using supervised learning techniques. *International Journal of Emerging Technologies in Learning (iJET)*, 14:92, 07 2019.

[99] S. S. Jacob and R. Vijayakumar. Sentimental analysis over twitter data using clustering based machine learning algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 01 2021.

[100] M. S. Jahan and M. Oussalah. A systematic review of hate speech automatic detection using natural language processing. *Neurocomputing*, 546:126232, 08 2023.

[101] S. Jahan. Team oulu at semeval-2020 task 12: Multilingual identification of offensive language, type and target of twitter post using translated datasets. 01 2020.

[102] R. Jain. *The art of computer systems performance analysis : techniques for experimental design, measurement, simulation, and modeling*. Wiley, 1991.

[103] C. Jeong. Fine-tuning and utilization methods of domain-specific llms. *arXiv:2401.02981*, 01 2024.

[104] A. Jiang, X. Yang, Y. Liu, and A. Zubiaga. Swsr: A chinese dataset and lexicon for online sexism detection. *Online Social Networks and Media*, 27:100182, 01 2022.

[105] N. Jung and G. Lee. Automated classification of building information modeling (bim) case studies by bim use based on natural language processing (nlp) and unsupervised learning. *Advanced Engineering Informatics*, 41:100917, 08 2019.

[106] A. Jurek, M. D. Mulvenna, and Y. Bi. Improved lexicon-based sentiment analysis for social media analytics. *Security Informatics*, 4, 12 2015.

[107] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1746–1751. Association for Computational Linguistics, 2014.

[108] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[109] R. M. Kowalski and S. P. Limber. Psychological, physical, and academic correlates of cyberbullying and traditional bullying. *Journal of Adolescent Health*, 53:S13–S20, 07 2013.

[110] O. Kraaijeveld and J. De Smedt. The predictive power of public twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, 65:101188, 03 2020.

[111] J.-D. Krieger, T. Spinde, T. Ruas, J. Kulshrestha, and B. Gipp. A domain-adaptive pre-training approach for language bias detection in news. *arXiv (Cornell University)*, 06 2022.

[112] T. Kumarage, A. Bhattacharjee, and J. Garland. Harnessing artificial intelligence to combat online hate: Exploring the challenges and opportunities of large language models in hate speech detection. *arXiv:2403.08035*, 03 2024.

[113] M. Lamott and M. A. Shakir. Leveraging distillation techniques for document understanding: A case study with flan-t5. *arXiv:2409.11282*, 09 2024.

[114] A. Le Glaz, Y. Haralambous, D.-H. Kim-Dufor, P. Lenca, R. Billot, T. C. Ryan, J. Marsh, J. DeVylder, M. Walter, S. Berrouiguet, and C. Lemey. Machine learning and natural language processing in mental health: Systematic review. *Journal of Medical Internet Research*, 23:e15708, 05 2021.

[115] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36, 09 2019.

[116] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *arXiv:2104.08691*, 04 2021.

[117] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence

pre-training for natural language generation, translation, and comprehension. *arXiv:1910.13461*, 10 2019.

[118] M. Li, S. Liao, E. Okpala, M. Tong, M. Costello, L. Cheng, H. Hu, and F. Luo. Covid-hatebert: a pre-trained language model for covid-19 related hate speech detection. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, page 233–238. IEEE Xplore, 12 2021.

[119] W. Li, F. Qi, M. Tang, and Z. Yu. Bidirectional lstm with self-attention mechanism and multi-channel features for sentiment classification. *Neurocomputing*, 387:63–77, 04 2020.

[120] X. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv:2101.00190*, 01 2021.

[121] Y. Li, Q. Pan, T. Yang, S. Wang, J. Tang, and E. Cambria. Learning word representations for sentiment analysis. *Cognitive Computation*, 9:843–851, 08 2017.

[122] D. Liu, M. Wang, and A. G. Catlin. Detecting anti-semitic hate speech using transformer-based large language models. *arXiv:2405.03794*, 05 2024.

[123] H. Liu, P. Burnap, W. Alorainy, and M. L. Williams. A fuzzy approach to text classification with two-stage training for ambiguous instances. *IEEE Transactions on Computational Social Systems*, 6:227–240, 03 2019.

[124] J. Liu, L. Niu, Z. Yuan, D. Yang, X. Wang, and W. Liu. Pd-quant: Post-training quantization based on prediction difference metric. *arXiv:2212.07048*, 01 2022.

[125] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning, 05 2016.

[126] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too. *arXiv:2103.10385*, 03 2021.

[127] Y. Liu, M. Ott, N. Goyal, J. Du, M. S. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arxiv:1907.11692*, 1, 07 2019.

[128] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 01 2019.

[129] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv:1508.04025*, 08 2015.

[130] X. Ma, G. Fang, and X. Wang. Llm-pruner: On the structural pruning of large language models. *arXiv:2305.11627*, 01 2023.

[131] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder. Hate speech detection: Challenges and solutions. *PLOS ONE*, 14:e0221152, 08 2019.

[132] M. S. I. Malik, A. Nawaz, and M. M. Jamjoom. Hate speech and target community detection in nastaliq urdu using transfer learning techniques. *IEEE Access*, 12:116875–116890, 2024.

[133] J. Manuel, D. A. Furman, L. A. Alemany, and F. M. Luque. Robertuito: a pre-trained language model for social media text in spanish. *arXiv (Cornell University)*, 11 2021.

[134] R. O. Mason. Ethical issues in artificial intelligence. *Encyclopedia of Information Systems*, pages 239–258, 2003.

[135] G. N. Masters. Partial credit model. *Elsevier eBooks*, pages 7–17, 01 2005.

[136] P. Mathur, R. Sawhney, M. Ayyar, and R. Shah. Did you offend me? classification of offensive tweets in hinglish language. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, 2018.

[137] J. Meng, Y. Long, Y. Yu, D. Zhao, and S. Liu. Cross-domain text sentiment analysis based on cnn ft method. *Information*, 10:162, 05 2019.

[138] Meta. meta-llama/llama-3.1-8b-instruct · hugging face, 09 2024. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

[139] A. Moreo, M. Romero, J. Castro, and J. Zurita. Lexicon-based comments-oriented news sentiment analyzer system. *Expert Systems with Applications*, 39:9166–9180, 08 2012.

[140] H. Mubarak, S. Hassan, and S. A. Chowdhury. Emojis as anchors to detect arabic offensive language and hate speech. *Natural Language Engineering*, pages 1–22, 08 2023.

[141] A. Mueller. Wordcloud for python documentation — wordcloud 1.8.1 documentation. https://amueller.github.io/word_cloud/.

[142] H. Mulki, H. Haddad, C. Bechikh Ali, and H. Alshabani. L-hsab: A levantine twitter dataset for hate speech and abusive language. *Proceedings of the Third Workshop on Abusive Language Online*, 2019.

[143] U. Naseem, I. Razzak, M. Khushi, P. W. Eklund, and J. Kim. Covidsenti: A large-scale benchmark twitter data set for covid-19 sentiment analysis. *IEEE Transactions on Computational Social Systems*, pages 1–13, 2021.

[144] A. Nasir, A. Sharma, and K. Jaidka. Llms and finetuning: Benchmarking cross-domain performance for hate speech detection. *arXiv:2310.18964*, 01 2023.

[145] U. Nations. What is hate speech?, 2024. https://www.un.org/en/hate-speech/understanding-hate-speech/what-is-hate-speech.

[146] H. Naveed, A. Ullah Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian. A comprehensive overview of large language models. *arXiv:2307.06435*, 04 2024.

[147] B. Newman, C. P. Kumar, and N. Rajani. P-adapters: Robustly extracting factual information from language models with diverse prompts. *arXiv:2110.07280*, 01 2021.

[148] T. T. Nguyen, C. Wilson, and J. Dalins. Fine-tuning llama 2 large language models for detecting online sexual predatory chats and abusive texts. *arXiv:2308.14683*, 08 2023.

[149] A. Nikolajeva and A. Teilans. Machine learning technology overview in terms of digital marketing and personalization. *Semantic Scholar*, page 125–130, 06 2021.

[150] A. Nikolov and V. Radivchev. Nikolov-radivchev at semeval-2019 task 6: Offensive tweet classification with bert and ensembles. In *Proceedings of the 13th*

International Workshop on Semantic Evaluation, page 691–695. ACLWeb, Association for Computational Linguistics, 06 2019.

[151] D. Nozza, C. Volpetti, and E. Fersini. Unintended bias in misogyny detection. *Web Intelligence*, 10 2019.

[152] I. K. Nti, A. F. Adekoya, B. A. Weyori, and O. Nyarko-Boateng. Applications of artificial intelligence in engineering and manufacturing: a systematic review. *Journal of Intelligent Manufacturing*, 33, 04 2021.

[153] E. Nurce, J. Keci, and L. Derczynski. Detecting abusive albanian. *arXiv:2107.13592*, 01 2021.

[154] Z. Nurulhuda, S. Ali, and I. Roliana. Hate crime on twitter: Aspect-based sentiment analysis approach. *Frontiers in Artificial Intelligence and Applications*, 2019.

[155] U. G. A. Office. Online extremism is a growing problem, but what's being done about it? — u.s. gao, 02 2024. https://www.gao.gov/blog/online-extremism-growing-problem-whats-being-done-about-it.

[156] OpenAI. Gpt-4 technical report. *arXiv:2303.08774*, 03 2023.

[157] E. W. Pamungkas, V. Basile, and V. Patti. Do you really want to hurt me? predicting abusive swearing in social media. *ACL Anthology*, pages 6237–6246, 05 2020.

[158] R. Pan, J. A. García-Díaz, and R. Valencia-García. Comparing fine-tuning, zero and few-shot strategies with large language models in hate speech detection in english. *Computer Modeling in Engineering and Sciences*, 140:2849–2868, 2024.

[159] P. Pezik, A. Mikolajczyk, A. Wawrzynski, B. Niton, and M. Ogrodniczuk. Keyword extraction from short texts with a text-to-text transfer transformer. *Communications in computer and information science*, pages 530–542, 01 2022.

[160] P. Piot and J. Parapar. Decoding hate: Exploring language models' reactions to hate speech. *arXiv:2410.00775*, 10 2024.

[161] G. K. Pitsilis, H. Ramampiaro, and H. Langseth. Effective hate-speech detection in twitter data using recurrent neural networks. *Applied Intelligence*, 48:4730–4742, 07 2018.

[162] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, and V. Patti. Resources and benchmark corpora for hate speech detection: a systematic review. *Language Resources and Evaluation*, 09 2020.

[163] M. Polignano, V. Basile, P. Basile, M. de Gemmis, and G. Semeraro. Alberto: Modeling italian social media language with bert. *Italian Journal of Computational Linguistics*, 5:11–31, 12 2019.

[164] D. Radečić. Softmax activation function explained — towards data science, 06 2020.

[165] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.

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

[167] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv:1910.10683*, 10 2019.

[168] G. Ramos, F. Batista, R. Ribeiro, P. Fialho, S. Moro, A. Fonseca, R. Guerra, P. Carvalho, C. Marques, and C. Silva. Leveraging transfer learning for hate speech detection in portuguese social media posts. *IEEE Access*, 12:101374–101389, 2024.

[169] S. Rana and A. Singh. Comparative analysis of sentiment orientation using svm and naive bayes techniques. *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 10 2016.

[170] S. Raschka. Finetuning llms with lora and qlora: Insights from hundreds of experiments - lightning ai, 10 2023.

[171] S. Raschka. Parameter-efficient llm finetuning with low-rank adaptation (lora), 04 2023.

[172] S. Raza and V. Chatrath. Harmonynet: Navigating hate speech detection. *Natural Language Processing Journal*, 8:100098, 09 2024.

[173] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, and J.-b. Alayrac *et al.* Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv:2403.05530*, 03 2024.

[174] M. Ribeiro, P. Calais, Y. Santos, V. Almeida, and W. Meira Jr. Characterizing and detecting hateful users on twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 12, 06 2018.

[175] C. Robertson, C. Mele, and S. Tavernise. 11 killed in synagogue massacre; suspect charged with 29 counts. *The New York Times*, 10 2018.

[176] I. Roll and R. Wylie. Evolution and revolution in artificial intelligence in education. *International Journal of Artificial Intelligence in Education*, 26:582–599, 02 2016.

[177] W. Rong, Y. Nie, Y. Ouyang, B. Peng, and Z. Xiong. Auto-encoder based bagging architecture for sentiment analysis. *Journal of visual languages and computing*, 25:840–849, 12 2014.

[178] G. Rudes and C. Fantuzzi. The association between racism and suicidality among young minority groups: A systematic review. *Journal of Transcultural Nursing*, 33:104365962110469, 09 2021.

[179] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3 edition, 2010.

[180] P. Röttger, D. Nozza, F. Bianchi, and D. Hovy. Data-efficient strategies for expanding hate speech detection into under-resourced languages. *arXiv (Cornell University)*, 01 2022.

[181] S. Saad and B. Saberi. Sentiment analysis or opinion mining: A review. *International Journal on Advanced Science, Engineering and Information Technology*, 7:1660, 10 2017.

[182] P. Sachdeva, R. Barreto, G. Bacon, A. Sahn, C. von Vacano, and C. Kennedy. The measuring hate speech corpus: Leveraging rasch measurement theory for data perspectivism, 06 2022.

[183] A. Salemi and H. Zamani. Evaluating retrieval quality in retrieval-augmented generation. *The 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 47:2395 – 2400, 07 2024.

[184] J. Salminen, H. Almerekhi, M. Milenković, S.-g. Jung, J. An, H. Kwak, and B. Jansen. Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media. *Proceedings of the International AAAI Conference on Web and Social Media*, 12, 06 2018.

[185] H. Sankar and V. Subramaniyaswamy. Investigating sentiment analysis using machine learning approach. *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 12 2017.

[186] R. Sann and P.-C. Lai. Understanding homophily of service failure within the hotel guest cycle: Applying nlp-aspect-based sentiment analysis to the hospitality industry. *International Journal of Hospitality Management*, 91:102678, 10 2020.

[187] I. H. Sarker. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2, 08 2021.

[188] K. Sasidaran and G. J. Multimodal hate speech detection using fine-tuned llama 2 model. *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, pages 1–6, 08 2024.

[189] K. Satyajit and A. Joshi. Hate speech detection from code-mixed hindi-english tweets using deep learning models. *arXiv:1811.05145*, 01 2018.

[190] A. Schmidt and M. Wiegand. A survey on hate speech detection using natural language processing, 04 2017.

[191] N. Shazeer. Glu variants improve transformer. *arXiv:2002.05202v1*, 02 2020.

[192] X. Shi, J. Liu, and Y. Song. Bert and llm-based multivariate hate speech detection on twitter: Comparative analysis and superior performance. *Communications in computer and information science*, pages 85–97, 01 2024.

[193] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177:3799–3821, 09 2007.

[194] A. Shoukry and A. Rafea. Sentence-level arabic sentiment analysis. *2012 International Conference on Collaboration Technologies and Systems (CTS)*, 05 2012.

[195] G. I. Sigurbergsson and L. Derczynski. Offensive language and hate speech detection for danish. *arXiv:1908.04531*, 08 2019.

[196] S. K. Singh, A. K. Tiwari, and H. Paliwal. A state-of-the-art review on the utilization of machine learning in nanofluids, solar energy generation, and the prognosis of solar power. *Engineering Analysis with Boundary Elements*, 155:62–86, 10 2023.

[197] S. O. Sood, E. F. Churchill, and J. Antin. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, 63:270–285, 10 2011.

[198] M. Soori, B. Arezoo, and R. Dastres. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cognitive Robotics*, 3:54–70, 2023.

[199] A. Stöffelbauer. How large language models work, 10 2023. https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f.

[200] J. Su, Y. Lu, S.-F. Pan, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv:2104.09864*, 04 2021.

[201] S. Suryawanshi, B. R. Chakravarthi, M. Arcan, and P. Buitelaar. Multimodal meme dataset (multioff) for identifying offensive content in image and text, 05 2020.

[202] I. Syarif, A. Prugel-Bennett, and G. Wills. Unsupervised clustering approach for network anomaly detection. *Networked Digital Technologies*, pages 135–145, 2012.

[203] J. F. Sánchez-Rada and C. A. Iglesias. Social context in sentiment analysis: Formal definition, overview of current trends and framework for comparison. *Information Fusion*, 52:344–356, 12 2019.

[204] M. A. Talukder, M. M. Islam, M. A. Uddin, A. Akhter, K. F. Hasan, and M. A. Moni. Machine learning-based lung and colon cancer detection using deep feature extraction and ensemble learning. *Expert Systems with Applications*, 205:117695, 11 2022.

[205] C. Tao, L. Hou, W. Zhang, L. Shang, X. Jiang, Q. Liu, P. Luo, and N. Wong. Compression of generative pre-trained language models via quantization. *arXiv:2203.10705*, 03 2022.

[206] L. S. Teven, A. Fan, C. Akiki, E. Pavlick, I. Suzana, and D. Hesslow *et al.* Bloom: A 176b-parameter open-access multilingual language model. *arXiv:2211.05100*, 11 2022.

[207] T. N. Y. Times. Christchurch shooting live updates: 49 are dead after 2 mosques are hit. *The New York Times*, 03 2019.

[208] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *arXiv:2302.13971*, 02 2023.

[209] T. Tran, H. Ba, and V.-N. Huynh. Measuring hotel review sentiment: An aspect-based sentiment analysis approach. *Lecture Notes in Computer Science*, pages 393–405, 2019.

[210] N. Umansky, M. Kubli, K. Donnay, F. Gilardi, D. Hangartner, A. Kotarcic, L. Bronner, S. Kurer, and P. Grech. Enhancing hate speech detection with fine-tuned large language models requires high-quality data. *OSF Preprints*, 03 2024.

[211] R. Vaish, U. Dwivedi, S. Tewari, and S. Tripathi. Machine learning applications in power system fault diagnosis: Research advancements and perspectives. *Engineering Applications of Artificial Intelligence*, 106:104504, 11 2021.

[212] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv:1706.03762*, 12 2017.

[213] F. D. Vigna, A. Cimino, F. Dell'Orletta, M. Petrocchi, and M. Tesconi. Hate me, hate me not: Hate speech detection on facebook. *ITA-SEC 17*, 01 2017.

[214] A. Vlachos. Evaluating unsupervised learning for natural language processing tasks. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 35–42. Association for Computational Linguistics, 2011.

[215] Y. Wan and Q. Gao. An ensemble sentiment classification system of twitter data for airline services analysis. *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 11 2015.

[216] L. Wang. Discovering phase transitions with unsupervised learning. *Physical Review B*, 94, 11 2016.

[217] S. Wang, J. Liu, X. Ouyang, and Y. Sun. Galileo at semeval-2020 task 12: Multi-lingual learning for offensive language identification using pre-trained language models. *arXiv:2010.03542*, 01 2020.

[218] M. Wankhade, A. C. S. Rao, and C. Kulkarni. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55, 02 2022.

[219] W. Warner and J. Hirschberg. Detecting hate speech on the world wide web, 06 2012.

[220] Z. Waseem and D. Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. *Proceedings of the NAACL Student Research Workshop*, 2016.

[221] H. Watanabe, M. Bouazizi, and T. Ohtsuki. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*, 6:13825–13835, 2018.

[222] B. Wei, J. Li, A. Gupta, H. Umair, A. Vovor, and N. Durzynski. Offensive language and hate speech detection with deep learning and transfer learning. *arXiv:2108.03305*, 01 2021.

[223] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi Quoc, V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models chain-of-thought prompting. *arXiv:2201.11903*, 01.

[224] J. T. Wilson, R. Moriconi, F. Hutter, and M. P. Deisenroth. The reparameterization trick for acquisition functions. *arXiv:1712.00424*, 01 2017.

[225] B. Xiang and L. Zhou. Improving twitter sentiment analysis with topic-based mixture modeling and semi-supervised training. 01 2014.

[226] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv:2211.10438*, 11 2022.

[227] E. Xu, J. Zhu, L. Zhang, Y. Wang, and W. Lin. Research on aspect-level sentiment analysis based on adversarial training and dependency parsing. *Electronics*, 13:1993–1993, 05 2024.

[228] R. Xu, F. Luo, C. Wang, B. Chang, J. Huang, S. Huang, and F. Huang. From dense to sparse: Contrastive pruning for better pre-trained language model compression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36:11547–11555, 06 2022.

[229] B. Yao, Y. Zhang, Q. Li, and J. Qin. Is sarcasm detection a step-by-step reasoning process in large language models? *arXiv:2407.12725*, 2024.

[230] A. Yavari, H. Hassanpour, B. Rahimpour Cami, and M. Mahdavi. Election prediction based on sentiment analysis using twitter data. *International Journal of Engineering*, 35:372–379, 2022.

[231] N. N. Yusof, A. Mohamed, and S. Abdul-Rahman. Reviewing classification approaches in sentiment analysis. *Communications in computer and information science*, pages 43–53, 01 2015.

[232] N. Zainuddin, A. Selamat, and R. Ibrahim. Discovering hate sentiment within twitter data through aspect-based sentiment analysis. *Journal of Physics: Conference Series*, 1447:012056, 01 2020.

[233] M. Zampieri, S. Malmasi, P. Nakov, S. B. Rosenthal, N. Farra, and R. Kumar. Predicting the type and target of offensive posts in social media. *arXiv:1902.09666*, 02 2019.

[234] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xiao, W. L. Tam, Z. Ma, Y. Xue, J. Zhai, W. Chen, P. Zhang, Y. Dong, and J. Tang. Glm-130b: An open bilingual pre-trained model. *arXiv:2210.02414*, 10 2022.

[235] B. Zhang and R. Sennrich. Root mean square layer normalization. 10 2019.

[236] B. Zhang, H. Yang, and X.-Y. Liu. Instruct-fingpt: Financial sentiment analysis by instruction tuning of general-purpose large language models. *arXiv:2306.12659*, 2022.

[237] H. Zhang, L. Harris, and N. S. Moosavi. Beyond hate speech: Nlp's challenges and opportunities in uncovering dehumanizing language. *arXiv:2402.13818*, 02 2024.

[238] M. Zhang, J. He, T. Ji, and C.-T. Lu. Don't go to extremes: Revealing the excessive sensitivity and calibration limitations of llms in implicit hate speech detection. *arXiv:2402.11406*, 02 2024.

[239] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, and S. Chen *et al.* Opt: Open pre-trained transformer language models. *arXiv:2205.01068*, 05 2022.

[240] W. Zhang, Y. Deng, B. Liu, S. J. Pan, and L. Bing. Sentiment analysis in the era of large language models: A reality check. *Association for Computational Linguistics*, Findings of the Association for Computational Linguistics: NAACL 2024:3881–3906, 05 2023.

[241] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, and Y. Hou *et al.* A survey of large language models. *arXiv:2303.18223*, 03 2023.

[242] B. Zohuri and F. Behgounia. Application of artificial intelligence driving nano-based drug delivery system. *Elsevier eBooks*, pages 145–212, 01 2023.