Major Project
on

"HATEFUL MEME DETECTION"

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree In
Computer Science & Engineering

by

Raj Shekhar Yadav (201128)

Ghanshyam Yadav (201135)

Chetna (201014)

Under the Supervision of
Ms. Poonam
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE And ENGINEERING
GLOBAL INSTITUE OF TECHNOLOGY AND MANAGEMENT, GURUGRAM

Affiliated to
MAHARISHI DAYANAND UNIVERSITY, ROHTAK

Academic Year 2023-24

DECLARATION

I Raj Shekhar Yadav, Ghanshyam Yadav and Chetna, students of B. Tech (CSE) declare that the project report titled "HATEFUL MEME DETECTION" which is submitted by us to Department of Computer Science and Engineering, Global Institute of Technology and Management, Gurugram affiliated to Maharishi Dayanand University, Rohtak, Haryana, India. This is the original work submitted by us in the partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering.

RAJ SHEKHAR YADAV - 201128

GHANSHYAM YADAV - 201135

CHETNA - 201014

CERTIFICATE

We hereby certify that the work which is being presented in project entitled "Hateful Meme Detection" by "Raj Shekhar Yadav 201128, Ghanshyam Yadav 201135, chetna 201014" has been successfully completed in partial fulfillment of requirement for the award of B. Tech (Computer Science and Engineering) degree, and is submitted in department of Computer Science and Engineering at Global Institute of Technology and Management, Farrukhnagar, Gurgaon under MDU University Rohtak. Their work progress has been supervised by "Ms. Poonam" at the institute itself.

Ms. Poonam
(Assistant Professor)

Professor (Dr.) Jaspal Kumar
HOD, CSE Department

……..

……
External Examiner

ACKNOWLEDGEMENT

We express my sincere thanks to my Project Guide Ms. Poonam for guiding me right from the inception till the successful completion of the project. I sincerely acknowledge her for extending their valuable guidance, support for literature, critical reviews of project and the report and above all, the oral support she had provided to me in all the stages of this project.

Our special thanks to Honourable Chairman Shri Ravinder Tokas, Director Dr. Parul Gupta and Ms Aruna Yadav, Director admissions providing us the best platform.

RAJ SHEKHAR YADAV - 201128

GHANSHYAM YADAV - 201135

CHETNA - 201014

ABSTRACT

Social media in the recent times has become the perpetrator of hateful content,and detecting these elementshas never been more relevant than it is now. When we humans look at a meme, we do not separate the image from the text on it. We understand the combined meaning together.

An AI tool built to detect hateful memes require that it understands the content and the context like we humans do. The project work proposes to solve the problem of automatically classifying memes as to being hateful or not by combining text, image feature information and additional source of information from web entity detection. The work done makes use of the Multimodal dataset from Hateful Meme Detection Challenge 2020. The dataset is introduced with confounder examples that contain memes that are benign, contrastive or counterfactual, making any State-Of-The-Art (SOTA) visual linguistic models perform less precisely in comparison to non-expert humans, showing difficulty of the task. It is essential for the models to have a deeper knowledge about the language, image, current events, and to understand the interaction among different modalities. The proposed approach uses the text, image and the information collected from the web entity detection process. The report discusses the shortfall of the approach and future directions to improve upon the proposed methodology. The work presented sometimes suffers from the lack of real-world knowledge. Detecting people's characteristics is hard and it is not fully capable of identifying racial/religious groups. It does not perform well on memes suggestive of disability, injury and abuse. The models find it hard to understand the Cultural, Political, Societal References, Traditional attires, and Religious practices.The proposed architecture uses two parallel streams for processing text and image with cross attention training. Both the streams are based on the bidirectionl multi-head attention model. The work discusses the preprocessing pipeline required for the proposed architecture. The project was done in two phases, the first phase resulted in an Area Under The Receiver Operating Characteristic (AUROC) of 0.71 and accuracy of 0.74 on the hateful memes dataset. The model resulted in an AUROC of 0.8108 on the test unseen data and 0.7555 on dev unseen data of the extended Hateful Meme Detection Dataset with an accuracy of 0.7352 for the test unseen and 0.7650 for the dev unseen data. The Hateful Meme detection dataset was extended to include more memes in Phase-II.

Table of contents

CHAPTER 1

INTRODUCTION

A meme by definition is “an image, video, piece of text, etc., typically funny, that are shared by netizens rapidly, more often with slight modifications” [1]. A meme as shown in Figure 1.1 can be put together by an image and a bunch of texts, and are often intended to be humorous, harmless. But, certain categories of images, text, or the combination of the two modalities could turn into a hateful meme.

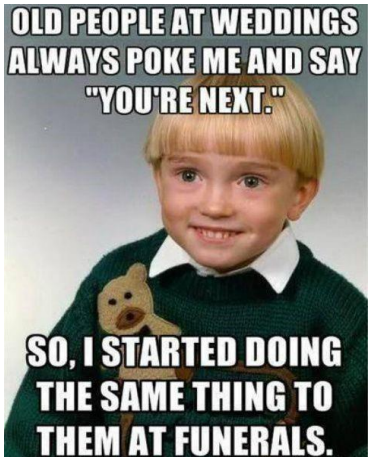


Figure 1.1: An example of a Meme

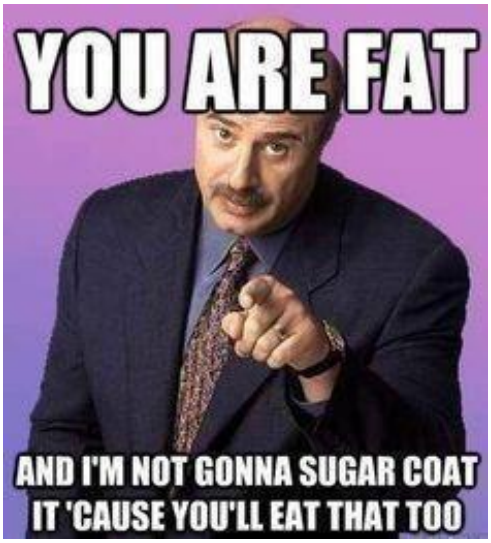
Hate is the direct or an indirect attack on a person or group on the basis of characteristics, including ethnicity, race, nationality, immigration status, religion, caste, sex, gender identity, sexual orientation, and disability or disease [2]. In this era of social media, memes can spread rapidly with slight variations, it might result in widespread of misinformation, hatred and exploitation of our fallibility. It can be unhealthy and destructive when target towards individuals or group. Attack in the online context can be violent or dehumanizing utterances, statements of inferiority, and speech for exclusion or segregation. Mocking hate crime is also considered hate speech. It is also hard to agree upon the definition of hate speech as it could range from cyber bullying, harassment, online aggression, abusive language, and hate speeches. The Hateful Meme Detection [2] presents a challenge due to its unique characteristics. The unimodal models fails to learn and only multimodal models have shown to succeed and it counters the possibility of models exploiting unimodal priors. For every hateful meme image in the dataset, there is an alternative image or

captions that make the meme to be not-hateful. Similarly, for every hateful meme image, there is an image with flipped label by retaining the original images but altering the text. The baseline performance of unimodal, multimodal models and performance of nonexpert humans [2] for the Hateful Meme Detection Dataset is shown in Table .The existing pretrained computer vision models such as VGG, ResNet etc. are trained on ImageNet. The dataset contains image labels and this results in the CNN model learning the visual features, relations and the descriptions more than the content and the context of the image. The models are not suitable for cross-modal tasks on their own. Multi-layer transformer model with multi-head attention is the hallmark for Natural Language Processing tasks. The models such as BERT, XLNET, ROBERTa, GPT are trained using a massive corpus of textual data. They learn textual representation by prediction word tokens using the context and are used for the downstream tasks with further fine tuning. Here, the text is sequential unlike the image. Images can be included in using the autoencoding but deciding how to do is crucial. These State-Of-The-Art(SOTA) multimodal models are shown to perform poorly compared to humans on this dataset. Non-expert human trained annotators have an accuracy of 84.7%. This is a proof that the best of the multimodal models are nowhere near the human performance. These models were trained on datasets that were in context unlike the hateful memes.

1.1 Problem definition and Objective

Figure 1.2: Benign Meme, Contrapositive Meme, Counterfactual Meme

The size of the hateful meme detection dataset is less when compared to the massive corpus of COCO (330K), Visual Genome (108K) and Conceptual Captions (3.3M). Hateful Meme dataset is not meant to be trained from scratch, but to fine tune and perform testing on the large pretrained model. The task is to perform binary classification of meme images as hateful or not. The SOTA Visual-Linguistic models are trained with datasets where the image and the textual content are correlated. The tasks for Visual Commonsense Reasoning (VCR) and Visual Question Answer (VQA) use multimodal dataset that are used in the pre-training and downstream tasks. Unlike existing multimodal dataset, the Hateful Meme Detection dataset [2] used in this work has a unique characteristic. The dataset is introduced with confounder examples that are either benign, contrastive or counterfactual as shown in Figure 1.2 It is diverse and contains objects that the existing classifiers cannot identify. Any visual hints such as injury to the self, physically challenged conditions, historical references, Political references, and traditional attires are difficult to recognize and therefore hard to find by off-the-shelf-classifiers. The placement of the text in the memes is also importance. Hence, doing just multimodal classification is not sufficient. It is essential to incorporate additional information from different sources to improve the accuracy of the classification problem.



(a) Example of a Hateful Meme

(b) Example of a Non-Hateful Meme

The objective is to extend the pre-existing SOTA multimodal models for this VisualLinguistic task that combines text, image feature information and leverage additional source of information to perform classification of memes as hateful or not.

Even with the recent advancements in multimodal classification, the SOTA models achieve an AUROC of 0.71 on the dataset. The non-expert human performance is far superior to this. The project work establishes the difference between the meme classification and the multimodal reasoning task.

$$AUROC = \int_{x=0}^{x=1} TPR(FPR^{-1}(x))\, dx$$

The key metric used is the AUROC and this will penalize the models bad at ordereing the hatefulness. The probabilities itself do not matter, and how they are ordered matters.

1.2 Contribution

Humans understand the meaning of memes by combining the text and image together, along with the understanding of the context and underlying meaning. We do not separate the image from the text on it. We want the AI tools to automatically predict the memes as hateful-or-not like we humans do. But it is hard to make the tool learn the context with a few examples of memes without context. Therefore, in this project work we try to add context to the memes by using additional information gathered from the Web.

CHAPTER 2

LITERATURE SURVEY AND ITS REVIEW

Literature Review

In the recent times, there has been several works in finding hateful speech in the networking sciences and Natural Language Processing . The datasets were text-only based. It is also hard to agree upon the definition of hate speech as it could range from cyber bullying, harassment, online aggression, abusive language, and hate speeches, etc..

2.1 Survery of work related to Hatred detection

Homa Hosseinmardi et al. address the issue of finding cyberbullying using instagram images and corresponding comments . They improved the accuracy of their classifer to 87.5% by including media information apart from images and text. The work done by Z. Waseem et al. propose topology for the task of detecting abuse in language that can be used while collection and annotation of data, as well as feature engineering and modelling. Ribeiro et al. followed a graph based approach in which they collected and annotated the hateful users, also the users that were banned from twitter. The approach was user centric and their node embedding algorithm outperformed the content-based hateful speech detection. A similar experiment was conducted by Zhong et al. using the dataset from the Instagram posts and comments to classify as bulling and not bullying Table 5.1 shows the baseline performances of unimodal, multimodal models and performance of nonexpert humans. The SOTA models do not perform better than humans, indicating the complexity of the classification problem. The non-expert trained annotators have an accuracy of 84.7%. This is a proof that the best of the multimodal models are no where near the human performance. This is where the scope of improvement comes into play.

2.2 Survey of early stage Visual-Linguistic Tasks

A million captioned photo dataset from Flickr and visually relevant captions was collected by Vicente Ordonez et al. . P. Young et al. created a large corpus of images with multiple

captions to create a denotational graph. MSR-VTT (Video-To-Text) is another such dataset which is a massive video description dataset to bridge gap between video and language. Microsoft COCO captions is a dataset with over 1.5 million captions describing 330K images . Sahar Kazemzadeh et al. came up with a two player game to gat her and verify expressions for real world natural scenes , which resulted in 130K expressions, referring to 97K distinct objects, in 19K images. Harm de Vries et al. created a two-player game to locate objects in a image with sequence of visual-linguistic questions . This work resulted in a large dataset with 800K Visual Question Answer(VQA) pairs on 66K images. There has been a broad exploration in multimodal field , however, are there are no standard dataset or benchmark task. Oleksii Sidorov et al., in their work collected 145K captions for 28K images, where they have tied the text to the objects in the image to come up with image captioning and a reading comprehension . Danna Gurari et al. introduced Vizwiz dataset in which the blind have recorded a visual question close to 31K for the images captured on their mobile phones. The quality of the images in the dataset is poor and the visual questions are more conversation unlike the other VQA datasets. In the works of Danna . Gurari et al., have collected over 39K images from blind people each providing 5 captions. A. Suhr et al. introduced a new dataset with 100K English sentences paired with images from the web for the task of visual reasoning using the multimodal data . To overcome the shortcomings of the previous VQA and compositional questioning tasks, Drew A et al. came up with a robust corpus of data that uses graph structure of the scene to come up with 22 million reasoning questions . N. Xie et al. created the SNLI-VE using the Stanford Natural Language Inference corpus and Flickr30 dataset, and have performed baselining for VQA task and built a model for Explainable Visual Entailment (EVE) .Vision and language issues have acquired a great deal of footing in ongoing years (see Mogadala et al.). Cesc C. Park and Gunhee Kim, in their paper generated captions based on the image streams, instead of just working with a single image and one caption .With extraordinary advancement on significant issues, for example, visual inquiry responding to and picture subtitle age and recovery, diagonostic test dataset with minimal bias for the task of visual reasoning, visual narrating ,visual exchange, multimodal machine interpretation, visual thinking, visual sound judgment thinking. F. Alam et al. released a multimodal dataset collected from twitter during natural disasters which includes the images and corresponding tweets to

aide in crisis management. UPMC-Food101 is a large dataset containing 100K recipes in 101 food catergories used in the works of Xin Wang et al. to automatically identify the food receipes based on the images. In the paper , the authors have made use of the image and review of the customer to improve and understand the experience of the future customers. Yiyi Li and Ying Xie in their paper understand the interaction on social media posts based on the images and the placement of the text data. Ignazio Gallo, Alessandro Calefati and Shah Nawaz train a visual classifier and a language classifier using the visual features and linguistic features respectively . The final classification is performed based on the fusion of the results. J. Yu and J. Jiang in their paper , use images and textual data capturing the sentiments to perform a task oriented multimodal classification. They have made use of the BERT model and incorporated the self attention mechanisms between to the two modalities to perform sentimental analysis. J. Kruk et al. introduced a dataset containing instagram images and its corresponding captions to form a mulitmodal dataset. Each of the 1.3K instagram posts are annotated with three orthogonal taxonomies to understand the intent, context and semantics of the instagram posts. John Arevalo et al. in their paper proposed a gated mulitmodal unit to find the representation of multiple modalities and trained on MM-IMDB dataset to predict the types from the banners and plots information. Z. Hussain et al. find a more profound comprehension of multimodal promotions which requires comparably unobtrusive thinking. Some of the datasets utilize web information incorporate Food101 , here the objective is to predict the dish of plans and pictures; different forms of Yelp audits Walmart and Ferramenta item grouping ; online social media name labeling web-based campaign target-situated feeling webbased media emergency taking care of different multimodal new arrangement datasets multimodal archive expectation in Instagram posts and anticipating labels for Flickr pictures. Few datasets incorporate grounded entailment, that takes advantage of the way that one of the huge scope normal language derivation datasets was built utilizing subtitles as premises, giving a picture, premise, theory trio and related entailment mark just as MMIMDB, where the task is to predict types using the banners and plots and getting a more profound comprehension of multimodal promotions, that needs comparably unobtrusive thinking . Sabat et al. as of late found in a fundamental report that the visual methodology can be more instructive for recognizing disdain discourse in images than the text. The nature often fluctuates generously, the information is not easily accessible.

2.3 Survery of SOTA Visual-Linguistic Tasks and Models

Related works on the Hateful Meme Detection Dataset use of these SOTA models that are pretrained on generic Visual-Linguistic(VL) tasks. The image and the textual content are correlated in the datasets used by the SOTA VL models. The tasks for Visual Commonsense Reasoning (VCR) and Visual Question Answer (VQA) have a multimodal dataset that are used in the pre-train datasets and downstream tasks. These datasets also have a direct relationship. The problem setting and the dataset used in the project work have unique characteristics, unlike the existing multimodal dataset. Some of the popular models used for the Visual Linguistic tasks are ViLBERT, VisualBERT, VLBERT and UNITER. All these models use the pretrained Faster RCNN model for extracting visual features and BERT model for Language Modelling. They have produced SOTA generic pretrainable models for Visual Linguistic tasks.

The comparisons of the models are as follows:

1. ViLBERT [40] uses two streams for the two modalities - text and the images, followed by a Transformer training to combine the two streams. They propose a unique way of training this crossmodality called the co-attentional transformer layer (Co-TRM). A third transformer is used to combine. They claim that their architecture has the ability to understand the interaction between the visual and linguistic contents better. VisualBERT [24], Unicoder-VL [41], VL-BERT [42] and UNITER [43] use early fusion with single stream model on both the modalities and use early fusion

2. The above single stream models use the masked language model pre-trainin task. VisualBERT does not have the object detection task or the visual-linguistic matching task. UNITER predicts the objects with the labels and perform masking of one of the modalities at time to avoid misalignment. They make use of the KL divergence loss between the input and the output distributions

3. VisualBERT used MS COCO Captions dataset for its pretraining, ViLBERT and VLBERT used Conceptual Captions dataset. UNITER added one million imagecaptions along with the conceptual captions, MS COCO and Genome Dense Captions data. UNICODER produced SOTA for imageto-text and text-to-image retrieval and VCR.

2.3.1 Visual Question Answer

Open-ended questions include fine-grained recognition ("What kind of sauce is used in the pasta?"), object detection ("How many balloons are in the scene?"), activity recognition ("Are the kids playing football?"), knowledge based reasoning ("Is this a Chinese cuisine?"), and common sense reasoning ("Is the female pregnant?"). The task in VCR is to take in an input image along with free form and openended question about the image. The model is expected to provide an accurate visual description of an image and the question in natural language. The questions target various regions in the image.

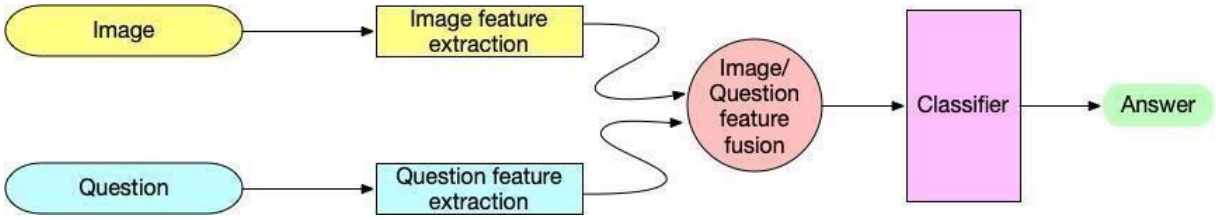


Figure 2.1: Visual Question Answer Task Pipeline

The pipeline in VQA involves three parts as shown in Figure 2.1:

1.Feature Extraction -This module extracts the features from images and the questions separately. The last convolutional layer of a CNN is used to extract image features. The final hidden layers of Long Short Term Memory(LSTM) or Gated Recurrent Units(GRU) is used to extract the features from the questions.

2.Multi Modal Fusion -Fusion is a vital aspect as it outputs the multimodal features by combining the image and question features. Some of the fusion techniques include element wise multiplication, bilinear pooling, concatenation, attention-based pooling, compositional approach, Bayesian-based methods, Elementwise multiplication is used when both image and question features are of the same dimension, bilinear pooling is used when the dimensions are different.

3.Classification This block takes the fused feature vector to perform classification. Questions with one word answers can be thought of as a classification problem, where all answers are converted to labels. For open-ended questions that multiple choice questions, the task becomes a multi-label classification problem. The author of the paper [44], extract the image features from the last hidden layer and l_2 norm of the activations of the last hidden layer of VGG[45]. The question features are extracted using three methods to provide an embedding for the questions. First, Bagof-Words Question (BoW Q), in which the top 1000 words in the questions are used to create a bag-of-words. This gives a 1030 embedding for the question. Second, LSTM Q, in which an LSTM with single hidden layer

gives an embedding of length 1024 and it is the concatenation of the last hidden state and the cell state. Third, deep LSTM Q, in which an LSTM with two hidden layers gives an embedding of length 2048 and it is the concatenation of the last hidden state and the cell state of the last two hidden layers. This is followed by a fully connected layer and tanh activation to reduce the dimension from 2048 to 1024. The question words embeddings for both the LSTMs are of length 300. They are encoded using a fully connected layer and a tanh layer. The input vocabulary is all the words from the questions in the training dataset. BoW Q and the image embeddings are concatenated, the image embeddings is converted to 1024 dimension using a fully connected layer and a tanh activation. This transformed image embedding is fused with LSTM Q embedding and deep LSTM Q embedding using element wise multiplication. The fused embeddings are passed to a fully connected neural network with two hidden layers of 1000 neurons, with tanh activation. The last layer is followed by a softmax layer to get the probabilities over K classes. Cross Entropy loss is the objective function used. Here, K = 1000, which is the top frequent answers and it covers 82% of the answers in training and validation set. The dataset consists of 200K images from Microsoft Common Objects in Context (MS COCO) with 50K abstract scenes. This dataset has over 760K questions with 10M answers.

2.3.2 Visual Commonsense Reasoning

VCR uses a massive dataset for the new task of understanding the images at cognitionlevel. We human beings can explain by looking at the photograph of a person placing orders to the waiter at a restaurant as shown in Figure 2.2a, it is a challenging task for the vision models. It requires that the vision models have a higher level of cognition and reasoning of the world like a human. The authors of the paper have produced SOTA model that answers the visual questions in Figure 2.2b expressed in natural language along with the reasoning in Figure 2.2c why the model's answer is true. VCR dataset has 110K images, 290K multiple choice questions, each with correct rationale and answers. The answers have an average length of 7.5 words and the rationals have typically 16 words. Questions are formed such that they are quite challenging and diverse. The task in VCR requires recognition and cognition level understanding, along with the meaning of the visual data in natural language and answer for the image under analysis. The reasoning for the image

CHAPTER 3

REQUIREMENT COLLECTION & ANALYSIS

Requirement collection and analysis for a hateful meme detection project involves gathering and understanding the needs, expectations, and constraints of stakeholders to define the project's functional and non-functional requirements. Here's a structured approach to requirement collection and analysis for such a project:

Identify Stakeholders:

Identify all stakeholders involved in or affected by the hateful meme detection project, including project sponsors, developers, platform administrators, moderators, content creators, and end-users.

Conduct Stakeholder Interviews and Surveys:

Engage with stakeholders through interviews, surveys, or focus groups to understand their needs, concerns, and expectations regarding hateful meme detection. Gather insights into the types of hateful content they encounter, the platforms or sources where it occurs, and their desired outcomes for the detection system.

Define User Personas:

Create user personas representing different user roles and their characteristics, goals, and pain points related to hateful meme detection. Use user personas to prioritize requirements and design features that address the needs of different user groups effectively.

Gather Functional Requirements:

Identify the functional requirements of the hateful meme detection system, including features and capabilities that stakeholders expect the system to have.

Examples of functional requirements may include:

Memes classification into hateful and non-hateful categories. Real-time monitoring and detection of hateful content. Integration with existing platforms for content moderation. Reporting and analytics capabilities for tracking moderation activities and trends.

Capture Non-Functional Requirements:

Capture non-functional requirements that define the quality attributes and constraints of the system. Consider aspects such as performance, scalability, reliability, usability, security, and compliance. Examples of non-functional requirements may include: System response time for detecting hateful content. Scalability to handle large volumes of meme data. Accuracy and precision of the detection models.

Compliance with data privacy regulations and platform policies.

Prioritize Requirements:

Prioritize requirements based on their importance and urgency, considering stakeholder needs, project goals, and constraints. Use techniques such as MoSCoW prioritization (Must-have, Should-have, Could-have, Won'thave) or Kano model analysis to categorize requirements.

Validate Requirements:

Validate requirements with stakeholders to ensure they accurately capture their needs and expectations. Use techniques such as requirement workshops, prototypes, or user acceptance testing to gather feedback and refine the requirements.

Document Requirements:

Document the gathered requirements in a requirements specification document, outlining the functional and non-functional requirements, user stories, acceptance criteria, and prioritization.

Ensure the requirements document is clear, comprehensive, and accessible to all stakeholders.

CHAPTER 4

SYSTEM FEASIBILITY STUDY, ANALYSIS & DESIGN

4.1 FEASIBILITY STUDY

A feasibility study and analysis design for a hateful meme detection project involves assessing various aspects to determine the project's viability and plan its implementation. Here's a structured approach to conducting a feasibility study and designing the analysis for such a project:

Define Project Objectives and Scope:

Clearly define the objectives of the hateful meme detection project, including the problem statement, goals, and expected outcomes.

Define the scope of the project, specifying the types of hateful content to be detected (e.g., hate speech, harassment, discrimination) and the platforms or sources from which memes will be collected.

Identify Stakeholders:

Identify the key stakeholders involved in or affected by the project, including project sponsors, developers, platform administrators, moderators, and end-users.

Understand their needs, concerns, and expectations regarding hateful meme detection.

Assess Technical Feasibility:

Evaluate the technical feasibility of implementing a hateful meme detection system, considering factors such as available technologies, data availability, computational resources, and expertise.

Assess the feasibility of collecting and preprocessing meme data, training machine learning models, deploying the system, and integrating it with existing platforms.

Evaluate Data Availability and Quality:

Assess the availability and quality of meme data for training and testing the detection models.

Evaluate the diversity, size, and representativeness of the dataset, ensuring it covers a wide range of hateful and non-hateful content.

Analyze Legal and Ethical Considerations:

Consider legal and ethical implications related to hateful meme detection, including compliance with privacy laws, terms of service of online platforms, and ethical guidelines for content moderation.

Assess the potential risks and challenges associated with detecting and moderating hateful content, including unintended biases, censorship, and free speech concerns.

Evaluate Economic Feasibility:

Assess the economic feasibility of the project, including the costs associated with data collection, preprocessing, model training, deployment, maintenance, and ongoing monitoring.

Estimate the potential benefits and returns on investment (ROI) of implementing a hateful meme detection system, considering factors such as reduced harm, improved user experience, and enhanced platform reputation.

Risk Analysis:

Identify potential risks and challenges that could impact the success of the project, such as technical limitations, data biases, legal issues, or user resistance.

Develop risk mitigation strategies to address and minimize these risks throughout the project lifecycle.

Feasibility Report:

Compile the findings of the feasibility study into a comprehensive report, summarizing the project objectives, scope, stakeholders, technical, legal, ethical, and economic considerations, as well as risk analysis.

Present recommendations for proceeding with the project, including any adjustments or precautions needed to ensure its success**.**

4.2 System Analysis & Design

Requirement Analysis:

- Gather requirements from stakeholders, including moderators, users, and platform administrators.
- Define the scope of the system, including supported platforms (e.g., social media, messaging apps), types of memes to be detected, and performance metrics.

Data Collection and Annotation:

- Gather a diverse dataset of memes, including both hateful and non-hateful examples.
- Annotate the dataset with labels indicating whether each meme contains hateful content or not.

System Architecture:

- Design the overall architecture of the system, including frontend, backend, and database components.
- Decide on the technologies and frameworks to be used, considering factors such as scalability, performance, and maintainability.

User Interface Design:

- Design a user-friendly interface for users to interact with the system.
- Include features such as image upload, analysis results display, and feedback mechanisms.

Backend Development:

- Implement the backend server responsible for receiving image data, running the detection algorithm, and returning results.
- Develop APIs for communication between frontend and backend components.
- Implement image preprocessing and feature extraction algorithms as needed.

Detection Algorithm Development:

- Select and implement machine learning models for meme classification.
- Experiment with different architectures, such as convolutional neural networks (CNNs) or transformer-based models, to find the most effective approach.
- Train the models using the annotated dataset and optimize hyperparameters for performance.

Integration with External Services:

- Integrate with external services, such as social media APIs, for seamless meme detection within different platforms.
- Implement mechanisms for real-time or batch processing of memes depending on the requirements.

Testing and Quality Assurance:

- Develop comprehensive test cases to validate the functionality and performance of the system.
- Conduct unit tests, integration tests, and end-to-end tests to ensure reliability and accuracy.
- Perform stress testing to evaluate the system's performance under heavy loads.

Deployment and Maintenance:

- Deploy the system to production environments, ensuring scalability, availability, and security.
- Implement monitoring and logging to track system performance and user interactions.
- Establish procedures for regular maintenance, including updating models, handling feedback, and addressing issues as they arise.

Ethical Considerations:

- Incorporate mechanisms to mitigate biases and ensure fairness in meme detection.
- Implement privacy controls to protect user data and sensitive information.
- Establish policies for handling false positives/negatives and providing transparency to users about the detection process.

CHAPTER 5

METHODOLOGY/ APPROACH OF WORK

Methodology

The approach is to use a single stream of processing for both the Image and Text with BERT as the backbone transformer model. The meme has text embedded on it. The first step in the pipeline in Figure 4.1, is to locate the text within the image and extract the textual information. Image Masks are created for the image region containing text using bounded boxes from the coordinates obtained from Optical Character Recognition (OCR) module. The image masks are then used for filling the region of text using a process called Inpainting [52]. If the images are made up of multiple patches, the inpainted image is passed through Object Detection Model to extract those patches. The next step in the pipeline is to leverage from the additional source of information. For this, the image patches are passed through Web Entity Detection Module to extract key words from it as shown in Figure 4.12. The multimodal data in now a combination of image, text and the additional key words pertaining to the image.

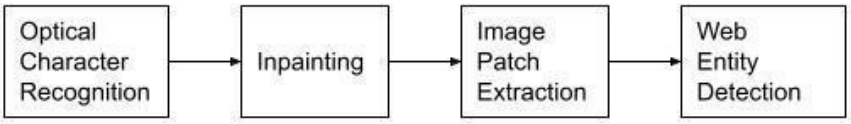


Figure 4.1: Proposed Preprocessing Pipeline

Optical Character Recognition

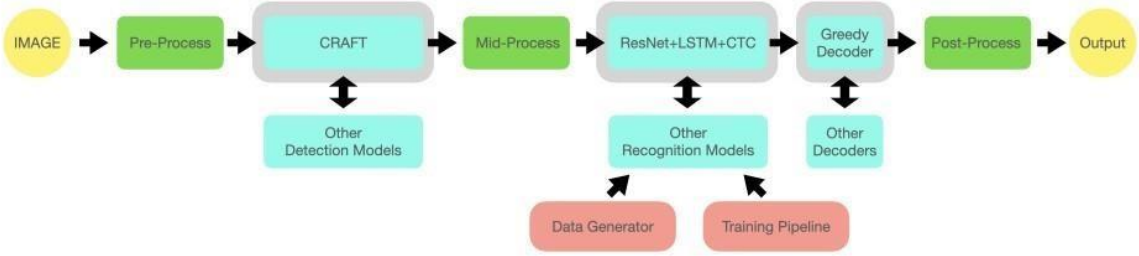


Figure 4.2: Easy OCR Framework

Optical Character Recognition(OCR) is a program used to convert the textual information on images into machine encoded text. The textual information can be a handwritten text, digits, signboards, traffic symbols, bills, invoices, bank documents etc.. Text extraction is achieved with two steps - text detection and text recognition or using a single deep neural network model to

achieve both the text detection and text recognition. EasyOCR is a python program used to achieve OCR and it has access to over 50 languages. EasyOCR has two phases as shown in Figure 4.2.

1.Text Detection

EasyOCR can use any pluggable detection model, in our case we use CRAFT detection model. It will help find the region where the textual information is present on the image. Image is the input and bounding boxes is the output.

2.Text Recognition

EasyOCR has provision for using pluggable recognition models. Here, the text is extracted from the image using the bounding boxes from the previous step. It takes the image and the bounding box as the input and produces raw text as the output. The phases has three components - Feature Extraction, Sequence Labelling and Decoding. Feature extraction is performed using Resnet, for Sequence Labelling LSTM is used and for decoding Connectionist Temporal Classification (CTC) is used. The text recognition pipeline is as shown below in the Figure 4.3

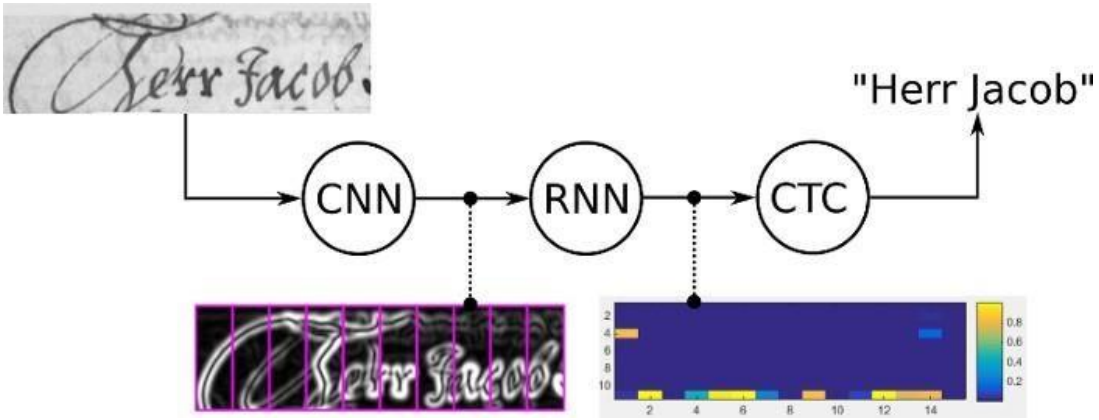


Figure 4.3: Text Recognition Pipeline

Text Recognition Pipeline

The text detection module will output the regions of the image where the text is present as a bounding box, this is passed as the input to CNN for feature extraction as seen in Figure 4.4, these features are sent as input to many-to-many LSTM models which then outputs a softmax probabilities over the vocabulary as shown in Figure 4.5. The outputs from different timesteps are sent to the CTC decoder to fetch the raw text.

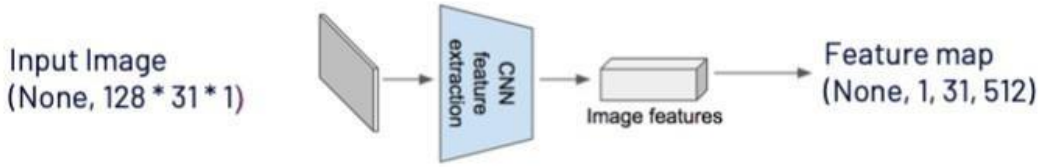


Figure 4.4: CNN based Feature extraction

The images are grayscale, width equal to 128 and height equal to 32, they are passed through a sequence of convolutional layers, pooling layers and activation layers as shown in Figure 4.11. The output is a feature map of size (1, 31, 512). Here, 31 indicates the timestep and 512 is the embedding size for each timestep. The image features are modelled using sequence labelling with first timestep for the left most part of the image and the 31st timestep for the right most part of the image. Each time step of 512 embedding size is sent as the input to LSTM which outputs the softmax probabilities for each timestep as shown in Figure 4.5 Choosing loss function for this task is

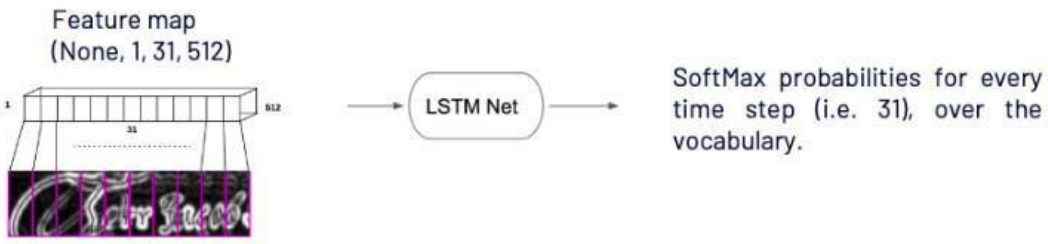


Figure 4.5: Sequence Labelling

quite tricky as the length of the ground truth need not be same as that of the number of timesteps, 31 in this case. Cross entropy loss is not a suitable choice of loss function. It is time consuming to align the ground truth with the timestep count for the training dataset. Annotating character level is time consuming and not fruitful to get just the character scores from a neural network. If the character width is too long, it will be duplicated in the output and we have to drop the duplicates. This will be problematic if the characters repeat consecutively. CTC takes all possible alignments of the ground truth text in the image and finds the sum of the probabilities. The score of the ground truth text is high only if the sum over the particular alignment has high probability. The duplicate characters in the input text are handled by inserting arbitrary number of special characters such as a blank space.
Decoding is done using beam search algorithm to extract the raw text.

Inpainting Images

Image inpainting is essentially completion of missing pixels of an image. The missing regions are filled with visually realistic and semantically probable pixels such that they are coherent with the surrounding pixels in the image. The previous works involved copying the pixels from the background regions and pasting them to holes starting from low to high resolution. This solution fails if the missing regions are complex and do not contain repetitive structures or objects. Inpainting with deep Generative Adverserial Network(GAN) is formulated as conditional image generation from high level recognition [52]. The lowlevel pixel generation are formulated using an encoder-decoder convolutional network. This is jointly trained with adversarial networks to adopt to the coherency of the synthesized and existing pixels. Using CNNs have shown to produce artifacts, distorted structures, and blurry textures that are not consistent with the nearby areas. There are two stages in inpainting - first stage involves using dilated convolutional neural network trained using the reconstruction loss and the second stage uses contextual attention layer as shown in Figure .The contextual attention mechanism makes use of the features of the known patches as the filters for convolution in order to process the generated regions. The contextual attention uses convolutions to match the generated pixel regions with the existing patches, and takes in softmax probabilities per

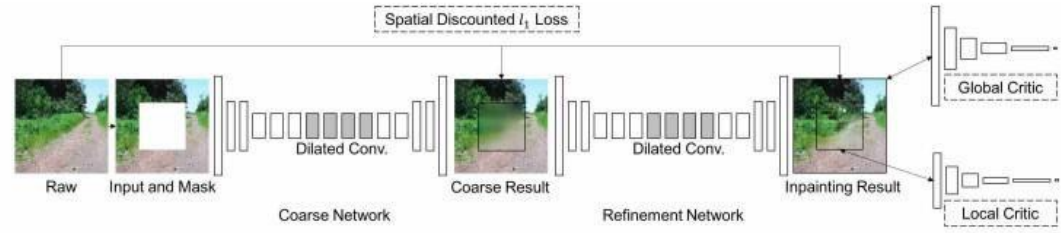


Figure 4.6: Pipeline for Inpaiting

channel to find the weightage of the relevant patches. Deconvolution is performed to reconstruct the generated patch using the contextual regions.

Figure 4.7: Results of Inpainting on Hateful Memes Dataset

It also has a spatial propagation layer to encourage spatial coherency of attention. There is an auxillary path of convolutional layers running in parallel to the contextual attention path inorder to generate novel contents. The output of two paths are aggregated and given as input to a decoder for the final output. An end-to-end training takes place with reconstruction loss and Wasserstein GAN(WGAN) losses. There are two WGAN losses one for the global image inpainting and the other for the local region inpainting.

Image Patch Extraction

Image Patch Extraction has the same pipeline as that of object detection and localization. We use of Faster R-CNN to perform image patch extraction for first approach and Detectron2 model for approach.

Meme Image Feature Extraction with Faster R-CNN

Faster R-CNN [53] is a successor of the R-CNN and Fast R-CNN models. It is fully differentiable with several moving parts. The architecture of the Faster R-CNN model is shown in Figure 4.8.

The input to the model is an image and the output is a list of bounding boxes, with labels and probabilities assigned to each of the bounding box. The images are passed to a pretrained CNN model where the feature maps are extracted from intermediate layer. Following the CNN is the Region Proposal Network (RPN) which is used to find the predefined number of regions that may

include objects. Here, the length of the list of bounding box is variable and it depends on the number of objects present in the image. To solve this problem RPN uses anchors that are fixed size bounding boxes positioned uniformly throughout the image. The new problem statement is to ask if the anchor contains any object of relevance and if yes, how to adjust the anchor to fit the object accordingly. RPN is a fully convolutional network, it uses a fully convolutional layer of 512 channels and 3 × 3 sized kernels, followed by two parallel convolutional layers that use 1 × 1 whose depth depends on the number of anchors at that point.

The classification layer outputs two values per anchor indicating the score of it being a background and the score of it being a foreground object. The regression layer outputs four values that represents the deltas, these deltas are later applied to the anchors to obtain the bounding boxes.

The foreground is determined as follows, the anchors that overlap with the ground truth object with an Intersection over Union (IoU) of more than 0.5 are considered as the foreground. For the background, the anchors that have an IoU of less than 0.1 with the ground truth or does not even overlap are considered as the background. The RPN randomly samples a mini batch of size 256 which includes both the foreground and the background to find the binary cross entropy loss for classification. The anchors that are marked as foreground from this mini batch are used to find the regression loss - Smooth L1 loss. Delta to fit the object within the bounded box is calculated using the foreground anchor and the ground truth.

The anchors overlap resulting the proposed regions to also overlap for the same objects, in order to avoid this duplicacy Non-Maximum Suppression (NMS) is used. The NMS algorithm takes in a list of scores for the proposed regions in a sorted order and discards those regions having an IoU larger than some predefined threshold belonging to a proposal with a higher score. Threshold selection is crucial, as a lower value can result in missing the regions and a higher value can result in having too many proposal regions. After NMS, proposal selection is made where the top-N proposals sorted by score is retained.

The output of RPN are bounding box proposals with no labels assigned to them. It is computationally infeasible to send in N proposals for each image to determine the objects present in the image.

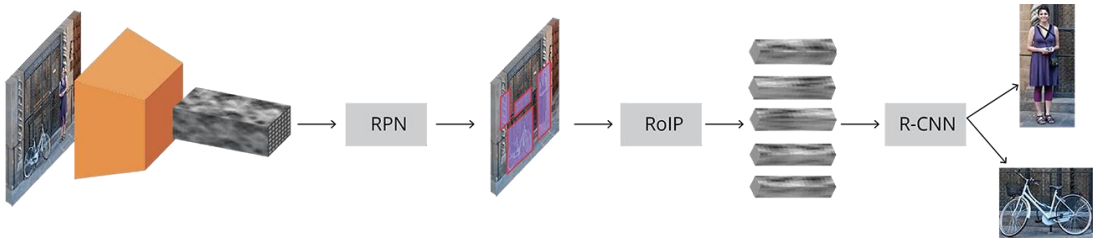


Figure 4.8: Architecture of Faster R-CNN model

each proposal using the region of interest pooling. The feature maps are of fixed size in order to predict fixed number of classes. The feature maps are cropped using the region proposals and resized to fixed sizes using bilinear interpolation.

The last part of the Faster R-CNN architecture is the R-CNN module which is used to classify the objects in the bounding box or discard by using the “background” label. The bounding box coordinates could be further adjusted to fit the objects properly.

The R-CNN module as shown in Figure 4.9 flattens the proposals and sends it two Fully Connected(FC) layers with ReLU activation. One FC layer has N+1 units where N is the number of classes and 1 is for the background class. The other FC layer has 4N units to get the regression output for the bounding boxes.

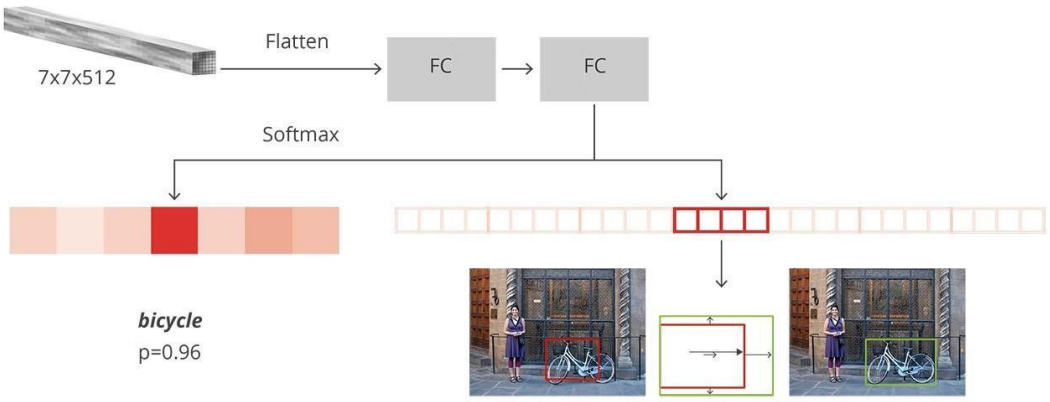


Figure 4.9: R-CNN module for Classification and Regression

The training of the R-CNN is similar to that of RPN, but in R-CNN we need to consider the classes for the objects. The proposals having IoU greater than 0.5 with the ground truth gets assigned with the ground truth and the proposals have IoU between 0.1 and 0.5 are assigned as background. The proposals with no intersection are ignored.

A mini batch of size 64 is randomly sampled to contain over 25% of foreground with objects assigned to a class and the rest as background. The classification loss is a cross entropy loss. The target for the regression FC is the offset of the proposals and the ground truth box. Smooth L1 loss is used for the proposed regions marked as foreground.

The post processing included applying NMS class wise by grouping the objects of the class based on the softmax probabilities in sorted order. We can use this to limit the number of objects for each class.

Mean Average Precision (mAP) at a certain threshold of IoU is used as the metric for evaluation.

Meme Image Feature Extraction with Detectron2

Detectron is an object detection algorithm produced by Facebook Artificial Intelligence Research (FAIR) [54]. It implements the SOTA object detection using Mask R-CNN benchmark, written with Python and on Caffe, a facebook deep learning framework. The base Faster R-CNN with Feature Pyramid Network is used for bounding box detection and it is the defacto standard. The backbone network is an FPN as shown in Figure 4.10 is used to extract the features at varied scales

- $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$. The res4 block uses $\frac{1}{16}$scaled features. FPN is capable of extracting features at multiple scales with different fields of reception as shown in Figure 4.11. The RPN network similar to the one in Faster R-CNN produces bounding box proposals with probability scores assigned them indicating as foregroun or background(no object) region. 1000 proposal are used in this case. The box-head network is used to crop the region proposals and ensure the object fit within the proposal regions. They output the fine-tuned locations for the bounding boxes and the classification result. NMS is used to obtain only those boxes whose IoU is greater than the threshold value.

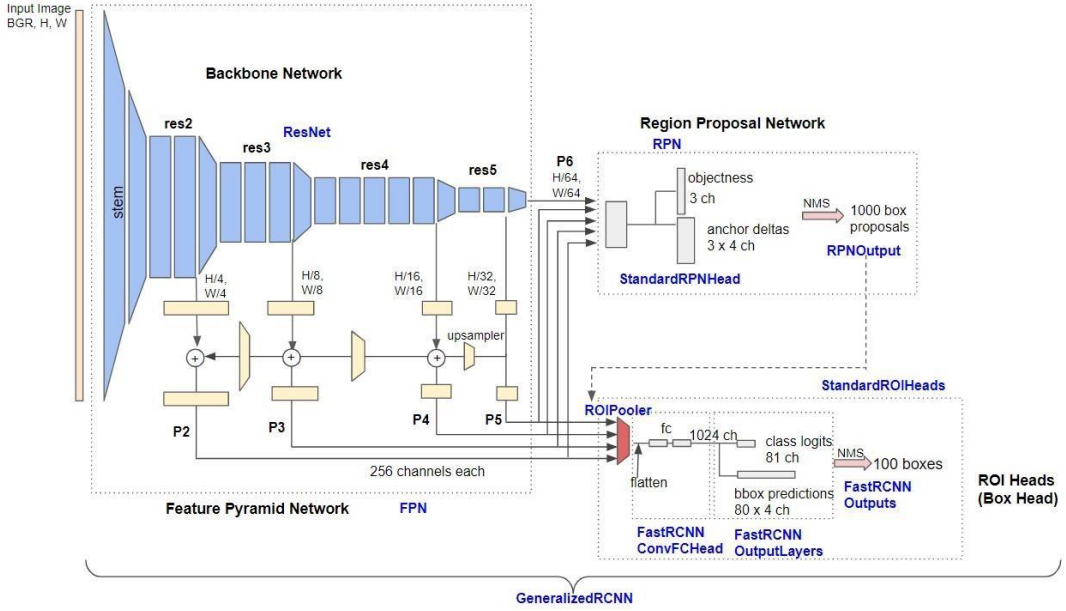


Figure 4.10: Architecture of Detectron2 model

Web Entity Detection

The Google vision API performs feature detection and entity labelling. It finds the relevant information on the web and outputs the web links and other related pages pertaining to the image.

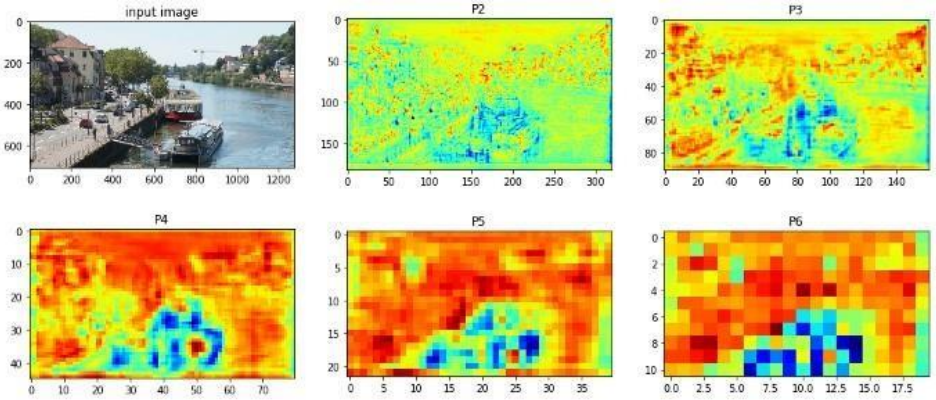


Figure 4.11: Feature maps with different receptive field

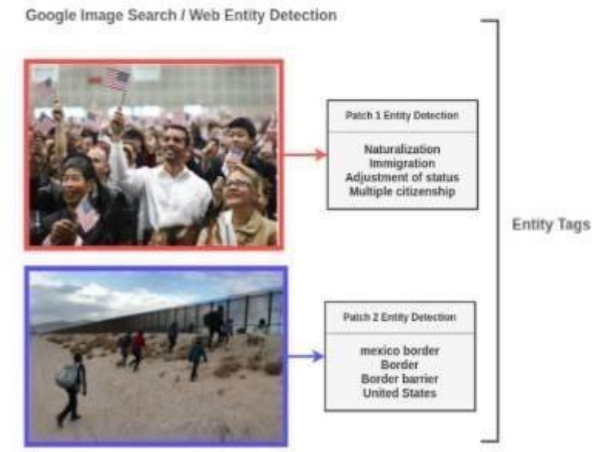


Figure 4.12: Example of Google Web Detection On Hateful Meme Dataset

Single Stream Hateful Meme Classification

Our first approach follows the preprocessing pipeline as shown in Figure 4.1. EasyOCR Model is used to extract text and the bounded boxes from the OCR Module. We create masks in the region of text using the bounded boxes. DeepFillV2 model is used for inpainting the memes. The model takes in the image mask and the image with masked regions of text. A modified Faster-RCNN model is used to extract the image patches from the inpainted image. If the image is made up of multiple image patches, the model splits them into multiple samples. The additional source of information for the multimodal data is obtained from the Google's web entity detection API. The API provides information on the image patches extracted, the key words are used in the classifier module along with the text and inpainted image. The essence is to capture the context of the image. After the data preprocessing pipeline, we use the multiple types of additional information at hand for the input image and also the text. Using this additional source of information of different sources and formats it is essential to combine them seamlessly.

-

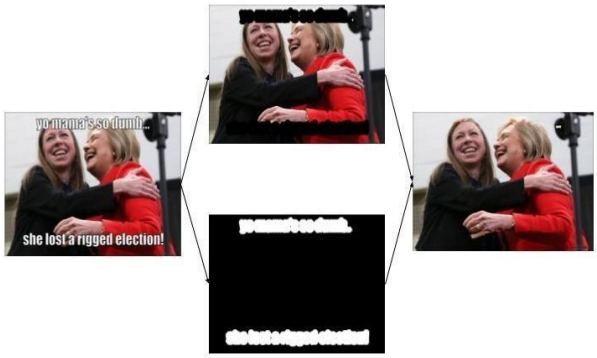


Figure 4.13: OCR and Inpainting Performed on Meme

CHAPTER 6

SYSTEM DESIGN

SYSTEM DESIGN

1.User Interface:

- Upload Interface: Users can upload memes for analysis through a simple and intuitive interface.
- Analysis Results: Display the results of meme analysis, indicating whether the meme is classified as hateful or not.
- Feedback Mechanism: Include a feedback option for users to report misclassifications or provide additional context.

2.Backend Server:

- API Endpoints: Develop RESTful API endpoints to handle requests from the user interfac and other external services.
- Image Processing: Implement image preprocessing techniques such as resizing,.
- Model Inference: Use machine learning models to classify memes based on extracted features and return the results.
- Feedback Handling: Process user feedback to improve the detection algorithm over time.

3.Detection Algorithm:

- Feature Extraction: Extract relevant features from memes, such as text overlays, visual elements, and context.
- Machine Learning Models: Train and deploy machine learning models (e.g., CNNs, transformers) to classify memes as hateful or non-hateful.
- Ensemble Methods: Combine multiple models to improve overall performance and robustness.

4.Data Storage:

- Dataset Storage: Store annotated datasets of memes for model training and evaluation.
- Model Storage: Store trained machine learning models for inference.
- User Feedback Storage: Store user feedback data for analysis and model improvement.

5.Integration with External Services:

- Social Media APIs: Integrate with social media platforms to analyze memes shared in posts or comments.

- Messaging Platform Integration: Extend support for meme analysis within messaging apps through APIs.

6.Scalability and Performance:

- Load Balancing: Distribute incoming requests across multiple servers to handle varying loads efficiently.
- Caching: Cache frequently acessed data to reduce response times and improve scalabilite.
- Asynchronous Processing: Use asynchronous processing for long-running tasks such as model inference to avoid blocking the server.

7.Security:

- Data Encryption: Encrypt sensitive data such as user uploads and feedback to protect pri vacy.
- Access Control: Implement rolebased access control to restrict access tosensitive function and data.
- Input Validation: Validate user inputs to prevent common security vulnerabilities such as injection attacks.

8.Monitoring and Logging:

- Logging: Log system events, errors, and user interactions for troubleshooting and auditin g purposes.
- Performance Monitoring: Monitor system performance metrics such as response time, th, oughout and resource utilization.
- Alerting: Set up alerts for abnormal system behavior or performance degradation.

9.Deployment:

- Containerization: Package the application components into containers for easy deploy.
- Continuous Integration/Continuous Deployment (CI/CD): Automate the deployment process to streamline updates and releases.

10.Documentation and Maintenance:

- Technical Documentation: Provide comprehensive documentation for developers,.
- User Documentation: Create user guides and tutorials to help users navigate the system effectively.
- Regular Maintenance: Schedule regular maintenance tasks such as updates, patches, and performance tuning to ensure system reli

CHAPTER 7

CODING AND TESTING

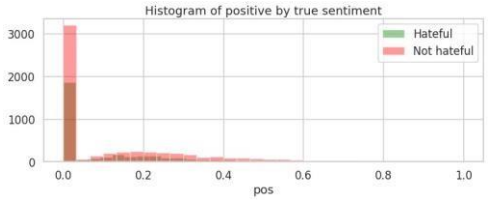
Coding

```
# Required Libraries import numpy as np import pandas as pd import torch import
torchvision import transformers import nltk import cv2 from
sklearn.feature_extraction.text import TfidfVectorizer from sklearn.model_selection
import train_test_split from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score from torch.utils.data import Dataset, DataLoader from
transformers import BertTokenizer, BertModel, BertForSequenceClassification from
torchvision import models, transforms from nltk.tokenize import word_tokenize

# Data Loading and Preprocessing
# Load meme data meme_data =
pd.read_csv("meme_data.csv")

# Preprocess text data tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
meme_data['text_tokens'] = meme_data['text'].apply(lambda x:
    tokenizer.encode(x,
add_special_tokens=True))

# Preprocess image data
# Implement image preprocessing (e.g., resize, normalization)
# Implement face detection and object detection (optional)

# Model Training
# Split data into train and test sets train_data, test_data = train_test_split(meme_data,
test_size=0.2, random_state=42)
```

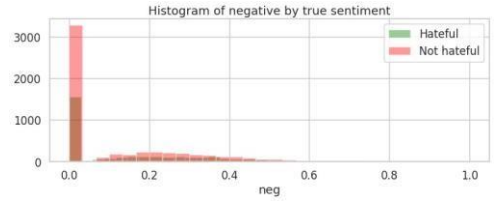
```
# Define and train text-based model (e.g., TF-IDF + SVM) tfidf_vectorizer
= TfidfVectorizer()
X_train_text = tfidf_vectorizer.fit_transform(train_data['text'])
X_test_text = tfidf_vectorizer.transform(test_data['text'])
# Train and evaluate text-based model

# Define and train image-based model (e.g., ResNet)
# Define data loaders for image data
# Train and evaluate image-based model

# Define and train multimodal model (e.g., Visual-BERT)
# Define dataset class for multimodal data
# Define data loaders for multimodal data
# Fine-tune Visual-BERT model
# Train and evaluate multimodal model

# Model Evaluation
# Evaluate models on test data using metrics such as accuracy, precision, recall, and F1 score #
Print or plot evaluation results

# Inference
# Use trained model for inference on new meme data
# Display classification results and probabilities

# Further Optimization and Deployment
# Further optimize models (e.g., hyperparameter tuning, model ensemble)
# Deploy models in production environment (e.g., web application, API) Dataset
```

Analysis of VADER on Hateful Meme Dataset Textual Data

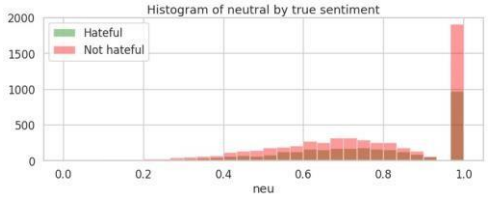
It can be observed that the text fall on the non-hateful side. But this does not indicate the meme in infact non-hateful as the image could be hateful. We cannot rely entirely on the text data for classification. The histogram in Figure 5.1a corresponds to texts that VADER sees as positive sentiment. The dataset is imbalanced when taking the text alone. Out of the texts predicted as positive, there are more non-hateful memes than hateful memes. And, in the Figure 5.1b for the text with negative sentiments, there are more non-hateful memes. This indicates that the same positive text on meme has be duplicated onto a hateful image.



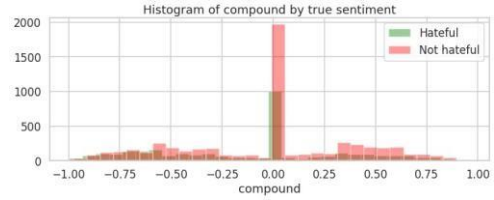
(a) Text Predicted as Positive Sentiment



(b) Text Predicted as Negative Sentiment

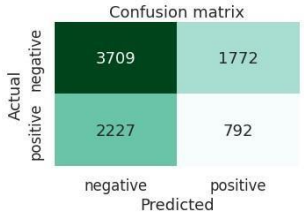


(c) Text Predicted as Neutral Sentiment

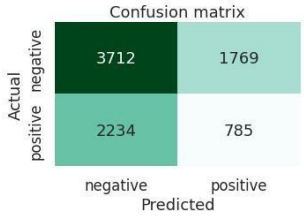


(d) Histogram for the compound score

Figure 5.1: Results of VADER on Meme Text



(a)



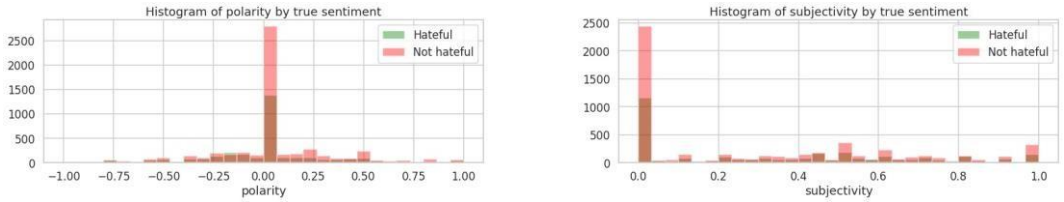
(b)

Figure 5.2: (a) Confusion matrix for Rule based classification of VADER Polarity scores and (b) Confusion matrix for Rule based classification of VADER compound scores

The results of Rule based classification using the positive and negative polarity is shown in 5.2a, this has an accuracy of 0.55 on the test result. The results of Rule based classification using the compound score of VADER is shown in 5.1d and resulted in an accuracy of 0.53 on the test data. In both the cases, precision and recall on the positive class (non-hateful) is low. This is a result of duplicating positive texts on negative images.

Analysis of TextBlob on Hateful Dataset Textual Data

The rule based classification on the Polarity scores from Textblob resulted in an accuracy on 0.55 on the test data and the confusion matrix is presented in 5.4. The algorithm has very lowprecision on the positive class/non-hateful memes and a very high precision on hateful memes.



(a) Histogram of Texblob Polarity Scores (b) Histogram of Texblob Subjectivity Scores

Figure 5.3: Results of TextBlob on Meme Text

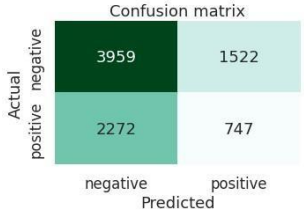


Figure 5.4: Confusion Matrix for Rule Based Classification on TextBlob Polarity Scores

Analysis of SGD on Hateful Meme Dataset Textual Data

Term Frequency Inverse Document Frequency (TF-IDF) tranforms textual information or raw information into matrix of TF-IDF features. This can be used as a preprocessing step in the NLP pipeline. For this experiment, TfidfVectorizer is followed by Randomized Hyperparameter search with cross validation. The contribution of each of the scores on the prediction in terms of coefficients is shown in 5.5. The SGD classifier resulted in an accuracy of 0.64 on the test data. The hateful meme had very high precision and where as the non-hateful meme had very low precision and recall. Most of the textual information is neutral but with negative background image, where as the hateful meme has mostly hateful text.

Table 5.1 shows the Baseline Performances of various Unimodal, Multimodal models with unimodal pretraining and Multimodal models with multimodal pretraining. A non-expert human has an accuracy of 84.7% on the hateful meme dataset. SOTA multimodal models pretrained on COCO and CC have a test accuracy of 75.44%.

The key metric used is the AUROC and this will penalize the models bad at ordereing the hatefulness. The probabilities itself do not matter, and how they are ordered matters. Our first approach with single stream model resulted an AUROC of 0.71 and an accuracy of 74% on the unseen test data. The plot for

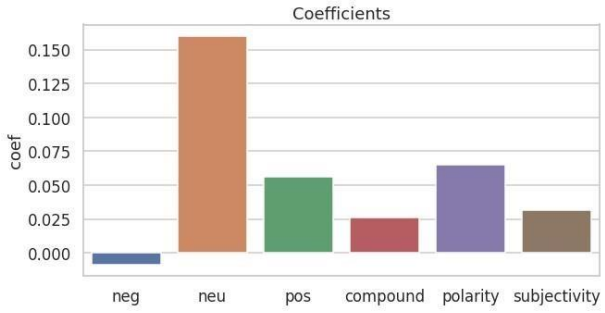


Figure 5.5: Coefficients for Sentiment scores, Polarity and Subjectivity

Table 5.1: Baseline Performance on Hateful Meme Detection Dataset

Type	Model	Validatio		Test	
		Accuracy	AUROC	Accuracy	AUROC
	Human	–	–	84.70	–
Unimodal	*ImageGrid*	50.67	52.33	52.73±0.72	53.71±2.04
	Image-Region	52.53	57.24	52.36±0.23	57.74±0.73
	Text BERT	58.27	65.05	62.80±1.42	69.00±0.11
Multimodal (Unimodal Pretraining)	*Late Fusion*	59.39	65.07	63.23±1.09	69.30±0.33
	Concat BERT	59.32	65.88	61.53±0.96	67.77±0.87
	MMBT-Grid	58.27	66.73	62.83±2.04	69.49±0.59
	MMBT-Region	64.75	72.62	67.66±1.39	73.82±0.20
	ViLBERT	63.16	72.17	65.27±2.40	73.32±1.09
	Visual BERT	65.01	74.14	66.67±1.68	74.42±1.34
Multimodal (Multimodal Pretraining)	*ViLBERT CC*	66.10	73.02	65.90±1.20	74.52±0.06
	Visual BERT COCO	65.93	74.14	69.47±2.06	75.44±1.86

training accuracy, training loss, validation accuracy and validation ROC is shown in 5.6. The training was done for 3000 epochs.

Benign Input and Text Confounder	20%
Random non-hateful Images	10%

as COCO (330K), Visual Genome (108K) and Conceptual Captions (3.3M). The dataset is not meant to be trained from scratch, but to fine tune and perform testing on the large pretrained model. Cultural references date back to hundreds and thousands of years ago, whose photos are rarely available or documented. The existing datasets contain very few examples of such references. The dresses worn or any for of makeup or adornments as shown in Figure 6.2a are not documented using photos, although their practices and customs are in written from of documentation, it is hard to find it in images.

It is the similar case with historical references Figure 6.2b, such as slavery, apartheid, world wars, when there are such a reference to historical events or memes based on them, it is hard for the AI tool to infer because of the lack of examples in the form of images. In Figure 6.2c, the funny aspect of the meme is understood by the facial expression. An AI tool untrained on facial expression and inferring will have a hard time to infer the meaning of the meme based on the image and the text alone. In Figure

Hyperparameter Sweeps									Phase 2-Validation (dev_unseen)		
id	lr_ratio	use_warmup	warmup_factor	warmup_iterations	lr	batch_size	scheduler.num_warmup_steps	scheduler.type	ROC-AUC	Acc.	best iteration
18	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7757	0.7315	3150
19	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7739	0.7241	900
16	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7695	0.7167	2100
8	0.3	TRUE	0.7	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7695	0.7241	700
7	0.3	TRUE	0.6	2.00E+03	5.00E-05	80	250	warmup_cosine	0.7678	0.7259	2100
21	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7677	0.7389	1950
6	0.3	TRUE	0.6	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7675	0.7352	2350
15	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7671	0.713	2650
0	0.6	TRUE	0.1	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7663	0.7241	1000
2	0.6	TRUE	0.1	2.00E+03	5.00E-05	80	500	warmup_cosine	0.7654	0.7259	1800
20	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7652	0.7259	3400
9	0.3	TRUE	0.3	5.00E+02	5.00E-05	80	250	warmup_cosine	0.7652	0.7204	700
13	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7651	0.7315	
23	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.765	0.7222	2350
17	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_linear	0.7649	0.7185	1400
22	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7647	0.7278	1450
1	0.6	TRUE	0.1	1.50E+03	5.00E-05	80	500	warmup_cosine	0.7641	0.6981	750
11	0.6	FALSE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7635	0.7222	
24	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7635	0.7111	750
5	0.3	TRUE	0.5	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7631	0.713	950
25	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7626	0.7204	700
26	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	1000	warmup_linear	0.7625	0.7278	1300
14	1	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7624	0.7241	2300
12	0.3	FALSE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7622	0.7204	
4	0.3	TRUE	0.3	2.00E+03	5.00E-05	80	250	warmup_cosine	0.7619	0.7093	550
3	0.3	TRUE	0.3	1.50E+03	5.00E-05	80	250	warmup_cosine	0.7608	0.7333	2700
10	0.3	TRUE	0.2	1.00E+03	5.00E-05	64	2000	warmup_linear	0.7605	0.7426	

Figure 5.8: Results of Hyperparameter Training

6.2d, the meme can be understood by humans easily as it is rhetorical inside joke. It depicts a social behaviour which is hard for a machine to experience. There are no fixed templates for memes, it

can come in various size, shapes. The text can be placed and in most of the cases the placement of the text on the meme makes a difference. The images may or may not contain text, a meme could be embedded within another meme as shown in Figure 6.2e. One of the hardest part is to training a meme detector to accommodate all the languages in the world, most of the existing datasets in text corpus are English only datasets. If the memes are in different language as shown in Figure 6.2f, the AI tool will fail to accurately classify the memes. The existing datasets do not contain images portraying injury, accidents, self harm, suicide, harm to others. The tool might fail to understand memes containing such images or references to such activities.

Table 6.2: Data Distribution in the Hateful Meme Dataset

Type	Unimodal	Multimodal	Benign	Random	Adverserial	Total
Train	1750	1300	3200	2250	-	8500
Dev-seen	50	200	200	50	-	500
Testseen	100	400	400	100	-	1000
Devunseen	-	200	200	-	140	540
Testunseen	-	729	597	-	674	2000

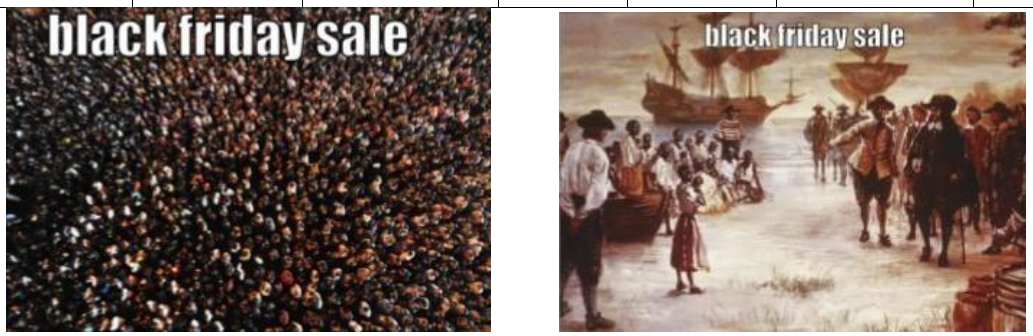


Figure 6.1: Interchanging Background Image

(a) Cultural Reference (b) Historical Reference (c) Facial Inference

```
# Preprocess the test data
test_sequences = tokenizer.texts_to_sequences(test_data['text'])
X_test = pad_sequences(test_sequences, maxlen=maxlen) y_test
= test_data['label']

# Make predictions using SVM classifier
svm_predictions = svm_classifier.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_precision = precision_score(y_test, svm_predictions)
svm_recall = recall_score(y_test, svm_predictions)
svm_f1 = f1_score(y_test, svm_predictions) print("SVM
Accuracy:", svm_accuracy) print("SVM Precision:",
svm_precision) print("SVM Recall:", svm_recall)
print("SVM F1 Score:", svm_f1)

# Evaluate the deep learning model loss, accuracy
= model.evaluate(X_test, y_test) print("Deep
Learning Model Accuracy:", accuracy)
```

CHAPTER 8

IMPLEMENTATION & MAINTENANCE

8.1 Implementation

He implementation of hateful meme detection systems involve several key steps and considerations to ensure effectiveness, accuracy, and ongoing reliability. Here's a general overview of the implementation and maintenance process:

Data Collection:

The first step is to gather a diverse dataset of memes annotated with labels indicating whether they contain hateful or offensive content. This dataset should be sufficiently large and varied to train robust machine learning models.

Algorithm Development:

Next, machine learning algorithms, such as deep neural networks, can be developed and trained using the annotated dataset. These algorithms should be capable of analyzing both text and visual elements of memes to accurately detect hateful content.

Feature Engineering:

Feature engineering involves selecting and extracting relevant features from memes, such as text embeddings, image features, and metadata. These features help the algorithm learn patterns associated with hateful content.

Model Training:

The algorithm is trained using the annotated dataset to learn the patterns and characteristics of hateful memes. Training involves iteratively adjusting the model's parameters to minimize prediction errors and improve performance.

Validation and Evaluation:

Once trained, the model is validated and evaluated using separate datasets to assess its performance. Metrics such as precision, recall, and F1 score are used to measure the model's accuracy and effectiveness in detecting hateful memes.

Deployment:

After validation, the model is deployed into production environments where it can be used to analyze memes in real-time. Integration with existing platforms, such as social media networks or content moderation tools, may be necessary at this stage. Once the model is trained and evaluated,.

Monitoring and Maintenance:

Continuous monitoring is essential to ensure the ongoing performance and reliability of the hateful meme detection system. This involves regularly evaluating the model's performance, detecting drift or degradation in accuracy, and making necessary updates or improvements.

Feedback Loop:

User feedback and human moderation play a crucial role in improving the accuracy of hateful meme detection systems. Feedback from users, moderators, and manual review processes can be used to refine the algorithms and update the training data.

Adaptation to Emerging Trends :

Hateful speech and offensive content evolve over time, so the detection system must be able to adapt to new trends, tactics, and cultural nuances. Regular updates and retraining with fresh data are necessary to keep the system effective.

Ethical Considerations:

Throughout the implementation and maintenance process, ethical considerations should be prioritized. This includes ensuring fairness, transparency, and accountability in the detection system, as well as minimizing unintended biases and risks.

Database:

The system might utilize a database to store data related to memes, such as training data, labeled datasets, and metadata associated with analyzed images.

APIs:

The system could leverage APIs for additional functionality, such as integrating with social media platforms to analyze memes shared in posts or comments.

Monitoring and Logging:

Implementing monitoring and logging mechanisms allows tracking system performance, errors, and user interactions, facilitating troubleshooting and improvement efforts.

8.2 Maintenance

Maintaining a hateful meme detection involves several ongoing tasks to ensure its effectiveness and relevance

1. Data Collection and Annotation:

Continuously collect and update datasets of memes, ensuring diversity in content and contexts. Annotation efforts should involve labeling memes for hateful or offensive content to improve the accuracy of machine learning models.

2. Model Retraining:

Regularly update and retrain machine learning models using the latest annotated data to adapt to evolving trends and new forms of hateful content.

3. Performance Monitoring:

Monitor the performance of the detection system, including metrics such as precision, recall, and F1 score, to assess its effectiveness and identify areas for improvement.

4.Algorithm Optimization:

Conduct research and experiments to optimize detection algorithms, considering factors such as feature selection, model architecture, and hyperparameter tuning.

5.False Positive Analysis:

Investigate false positive cases to understand why they occurance andimprove the model's ability to distinguish between harmless and hateful content. .

6.Scalability and Efficiency:

Ensure that the detection system can handle increasing volumes of memes efficiently by optimizing resource usage and scaling infrastructure as needed.

7.Ethical Considerations:

Continuously assess the ethical implications of the project, including potential biases in data or algorithms, and take steps to mitigate harm and promote fairness in content moderation.

8.Regulatory Compliance: Stay informed about relevant laws and regulations governing online content moderation, ensuring compliance with legal requirements and industry standards.

9.Documentation and Reporting:

Maintain comprehensive documentation of the project's processes, methodologies, and outcomes, and provide regular reports to stakeholders on its performance and impact.

CHAPTER 9

RESULTS & CONCLUSIONS

9.1 Results

Detecting hateful memes typically involves using machine learning algorithms trained on datasets labeled for hate speech or offensive content. These algorithms analyze various aspects of an image, such as text overlays, visual elements, and context, to determine whether the meme contains hateful or offensive content.

The result of hateful meme detection would usually be a binary classification indicating whether the meme is considered hateful or not. If the algorithm detects hateful content, it may also provide insights into the specific elements contributing to the classification, such as the presence of derogatory language, discriminatory imagery, or violent themes.

The results of a hateful meme detection can vary depending on various factors such as the quality of data, the effectiveness of algorithms, and the specific goals of the project. Here are some potential outcomes and metrics used to evaluate the results:

Accuracy:

Accuracy is a fundamental metric that measures the proportion of correctly identified hateful memes among all memes analyzed. A higher accuracy indicates a more reliable detection system.

Precision and Recall:

Precision measures the proportion of correctly identified hateful memes among all memes classified as hateful by the system, while recall measures the proportion of correctly identified hateful memes among all actual hateful memes. Balancing precision and recall is crucial to minimize false positives and false negatives.

F1 Score:

The F1 score is the harmonic mean of precision and recall and provides a single metric to evaluate the overall performance of the detection system.

False Positive Rate:

The false positive rate measures the proportion of non-hateful memes incorrectly classified as hateful by the system. Minimizing false positives helps prevent the wrongful removal of nonoffensive content.

False Negative Rate:

Minimizing false negatives is crucial to effectively identify and mitigate hateful content.

Speed and Scalability:

The speed and scalability of the detection system are essential for real-time analysis of memes on large-scale platforms. Faster processing times and the ability to handle increasing volumes of data indicate better scalability.

User Feedback and Satisfaction:

Gathering feedback from users, moderators, and stakeholders can provide insights into the effectiveness and usability of the detection system. Positive feedback and increased user satisfaction indicate successful implementation.

Reduction in Harmful Content:

Ultimately, the primary goal of a hateful meme detection project is to reduce the prevalence and impact of hateful content online. A successful project will lead to a noticeable decrease in the dissemination of hateful memes and the associated harm to individuals and communities.

Ethical Considerations:

Evaluating the results of a hateful meme detection project also involves assessing its compliance with ethical standards, including fairness, transparency, and accountability. Ensuring that the detection system operates ethically and respects users' rights is essential for long-term success and trust

9.2 Conclusion

The conclusion of hateful meme detection typically involves determining whether a meme contains content that is considered hateful, offensive, or harmful based on predefined criteria and machine learning algorithms. This conclusion is usually presented as a binary outcome: either the meme is classified as hateful/offensive or it is not. However, it's essential to recognize that meme detection algorithms are not infallible. They rely on training data and predefined features to make classifications, which can lead to errors or misinterpretations, particularly in cases involving sarcasm, irony, or satire.

Therefore, while hateful meme detection algorithms can provide valuable insights and assist in to moderating online content, they should be used as tools rather than definitive judgments. Human review and contextual understanding are crucial for accurately interpreting and addressing the co mplex nuances present in memes.

CHAPTER 10

APPLICATION AREA & FUTURE SCOPE

The application areas of hateful meme detection are diverse and encompass various sectors where online hate speech poses a threat to individuals, communities, and society as a whole. Here are some key application areas:

10.1 Application Area

1.Social Media Platforms:

Social media platforms such as Facebook, Twitter, Instagram, and TikTok can utilize hateful meme detection systems to identify and remove offensive content from their platforms. By doing so, they can maintain a safer and more inclusive online environment for their users.

2. Content Moderation:

Online forums, discussion boards, and community-driven websites can benefit from hateful meme detection to moderate user-generated content effectively. This helps prevent there dissemination of hate speech and harmful content within online communities.

3. Educational Institutions:

Schools, colleges, and universities can employ hateful meme detection systems to monitor online platforms frequented by students and faculty..

4. Government Agencies:

Government agencies responsible for regulating online content and combating extremism can use hateful meme detection technology to monitor social media platforms and online forums for signs of radicalization, hate speech, and extremist propaganda.

4. Research and Analysis:

Researchers studying online hate speech and its effects on society can utilize hateful meme detection systems to analyze large datasets of memes and i patterns, trends characteristics associated with hateful content.

5. Media and News Organizations:

Media outlets and news organizations can employ hateful meme detection to monitor social media platforms for instances of online harassment.

6 Corporate Reputation Management:

Companies and brands can use hateful meme detection technology to monitor social media to platforms for instances of brand-related hate speech and negative sentiment. This allows them to identify potential reputational risks and take appropriate action to mitigate them

7.Online Gaming Communities:

Online gaming platforms and communities can benefit from hateful meme detection to addres toxic behavior, harassment, and hate speech among players.

8.Nonprofit Organizations:

Nonprofit organizations dedicated to combating speech, discrimination, and online harassment can leverage hateful meme detection technology to monitor social media platforms for instant hate

10.2 Future Scope

The future scope of hateful meme detection is significant, considering increasing prevalence of online hate speech and theevolving nature of internet communication. Are some potential avenues for advancement and growth in this field:

1. Advanced AI Algorithms:

Continued advancements in artificial intelligence and machine learning algorithms will enhance the ability to detect hateful memes accurately. These algorithms can be train on large dataset datasets of annotated memes to improve their understanding of context, sarcasm, and subtleties in language**.**

2. Multimodal Analysis:

Future research may focus on integrating multiple modalities, such as text, images, and videos, for more robust meme detection. Combining different types of data can provide a more compr ehansive understanding of the content and context of memes, leading to more accurate detect

3. Contextual Understanding:

Developing algorithms that can grasp the context surrounding memes is crucial. Memes often rely on cultural references, current events, and internet trends, which can be challenging for tr ditional algorithms to interpret accurately. Advancements in natural language processing and computer vision can help algorithms better understand the context in which memes are used

4. Dynamic Adaptation:

Hate speech and offensive content evolve rapidly, often using coded language or imagery to evade detection. Future systems may need to dynamically adapt and update their detection capabilities to keep pace with emerging trends and tactics used by malicious actors.

REFERENCES

[1] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, and D. Testuggine, "The hateful memes challenge: Detecting hate speech in multimodal memes," *arXiv preprint arXiv:2005.04790*, 2020.

[2] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the fifth international workshop on natural language processing for social media*, 2017, pp. 1–10.

[3] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–30, 2018.

[4] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Detection of cyberbullying incidents on the instagram social network," *arXiv preprint arXiv:1503.03909*, 2015.

[5] Z. Waseem, T. Davidson, D. Warmsley, and I. Weber, "Understanding abuse: A typology of abusive language detection subtasks," *arXiv preprint arXiv:1705.09899*, 2017.

[6] M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. Almeida, and W. Meira Jr, "Characterizing and detecting hateful users on twitter," in *Twelfth international AAAI conference on web and social media*, 2018.

[7] H. Zhong, H. Li, A. C. Squicciarini, S. M. Rajtmajer, C. Griffin, D. J. Miller, and C. Caragea, "Content-driven detection of cyberbullying on the instagram social network." in *IJCAI*, vol. 16, 2016, pp. 3952–3958.

[8] V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2text: Describing images using 1 million captioned photographs," in *Neural Information Processing Systems (NIPS)*, 2011.

[9] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.