

National University of Computer and Emerging Sciences



Memotion Analysis

Duraze Saquib 16L-4215

Shahiryar Khalid 16L-4469

Ahmad Moatasim 16L-4332

Mehar Fatima Khan 16L-4388

Supervisor:Maryam Bashir

B.S. Computer Science

Final Year Project

December 2019

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

**Title: \_\_\_\_Memotion Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: December 4, 2019

Student 1

Name: Duraze Saquib

Signature: 

Student 2

Name: Shahiryar Khalid

Signature: 

Student 3

Name: Ahmad Moatasim

Signature: 

Student 4

Name: Mehar Fatima Khan

Signature: 

Table of Contents

[Table of Contents i](#_Toc26362194)

[List of Tables iii](#_Toc26362195)

[List of Figures iv](#_Toc26362196)

[Abstract 1](#_Toc26362197)

[Chapter 1: Introduction 2](#_Toc26362198)

[1.1 Goals and Objectives 2](#_Toc26362199)

[Scope of the Project 2](#_Toc26362200)

[Chapter 2: Literature Survey / Related Work 4](#_Toc26362201)

[2.1 Detailed Literature Review 4](#_Toc26362203)

[2.1.1 Analyzing Textual Features 4](#_Toc26362204)

[2.1.2 Analyzing Visual Features 4](#_Toc26362205)

[2.1.3 Integrating Textual and Visual Features 5](#_Toc26362206)

[Literature Review Summary Table 6](#_Toc26362207)

[Chapter 3: Requirements and Design 9](#_Toc26362208)

[Functional Requirements 9](#_Toc26362210)

[3.1.1 Sentiment Classification of Memes 9](#_Toc26362211)

[3.1.2 Humor Classification of Memes 9](#_Toc26362212)

[3.1.3 Identifying Scales of Semantic Classes 9](#_Toc26362213)

[3.1.4 Display Accuracies 9](#_Toc26362214)

[Non-Functional Requirements 9](#_Toc26362215)

[3.1.5 Scalability 9](#_Toc26362216)

[3.1.6 Portability 9](#_Toc26362217)

[3.1.7 Performance 9](#_Toc26362218)

[3.1.8 Usability 9](#_Toc26362219)

[3.1.9 Reliability 9](#_Toc26362220)

[3.1.10 Manageability 10](#_Toc26362221)

[3.1.11 Reusability 10](#_Toc26362222)

[3.1.12 Availability 10](#_Toc26362223)

[Hardware and Software Requirements 10](#_Toc26362224)

[System Architecture 11](#_Toc26362225)

[3.1.13 External System Architecture 11](#_Toc26362226)

[3.1.14 Internal System Architecture 12](#_Toc26362227)

[3.1.15 Assumptions 12](#_Toc26362228)

[Chapter 4: Implementation 13](#_Toc26362229)

[Text Preprocessing 13](#_Toc26362231)

[4.1.1 Duplicates and Null Values 13](#_Toc26362232)

[4.1.2 Punctuation Removal 13](#_Toc26362233)

[4.1.3 Lower Case Alphabets 13](#_Toc26362234)

[4.1.4 Removal of Contractions 13](#_Toc26362235)

[4.1.5 Spelling Correction 13](#_Toc26362236)

[4.1.6 Removal of Special Characters 13](#_Toc26362237)

[Textual Feature Engineering (Word Embeddings) 13](#_Toc26362238)

[4.1.7 Result for Textual Classification 13](#_Toc26362239)

[Image Preprocessing 13](#_Toc26362240)

[Visual Feature Engineering (GLCM Texture Features) 14](#_Toc26362241)

[Emotion Detection 14](#_Toc26362242)

[Concatenation of Feature Attributes 14](#_Toc26362243)

[Model Fitting 14](#_Toc26362244)

[Application Development 14](#_Toc26362245)

[Chapter 5: Conclusion 15](#_Toc26362246)

[References 16](#_Toc26362247)

[Appendix 17](#_Toc26362248)

[Appendix A: Sample Memes 17](#_Toc26362249)

[Appendix B: Glossary 18](#_Toc26362250)

[Definitions 18](#_Toc26362251)

[Acronyms 19](#_Toc26362252)

[Abbreviations 20](#_Toc26362253)

[Appendix C: Dataset 21](#_Toc26362254)

[Appendix D: Code 22](#_Toc26362255)

[Code for Visual Feature Extraction 22](#_Toc26362256)

[Code for Textual Classification 23](#_Toc26362257)

List of Tables

[Table 1: Research Work on Text Sentiment and Image Sentiment Analysis 6](#_Toc26354111)

List of Figures

Figure 1: External System Architecture 11

Figure 2: Internal System Architecture 12

Figure 3: Sample Memes [5] 17

Figure 4: Memotion Analysis Dataset 21

# Abstract

In recent times, the internet has evolved its way of social contact by embracing meme culture. However, when memes start exhibiting offensive connotations, the duty falls on social media sites to remove them. This project serves to propose a hybrid technique which uses text and image processing to decipher the sentiment of memes. This interpretation of memes is further categorized in two subtasks: sentiment classification and humor classification. A dataset of over 6600 memes is used to separately preprocess the text and image of a meme. After text preprocessing, the data is fed into a CNN which uses pre-trained Glove embedding. This yields a score of 74% for humor, 65% for sarcasm, and 56% for offense. Through past research work, it is found that visual features are of paramount significance in sentiment analysis of images. Thus, visual features (homogeneity, contrast, energy, dissimilarity) are extracted using GLCM matrix. To yield better results for visual feature extraction, images are to be preprocessed. In the later phases, both visual and textual features are to be combined to produce results. It is anticipated that a multi-model which yields the highest possible accuracy is delivered. In the subsequent days, the model is to be deployed on a desktop application for ease of use for a layperson.

# Chapter 1: Introduction

Internet memes have revolutionized the way we incorporate humor into communication over the recent years. A meme can be aptly described as a virally-transmitted image which contains witty or humorous catchphrases carrying sarcastic, ironic or dark undertones (see Figure 3 in Appendix A). Memes may relate to cultural experiences that are gained through movies or TV shows, and they more often than not resonate powerfully with people’s opinions and emotions [4].

As they have so deeply penetrated into the social media world, it has become imperative for social media companies to understand what they actually mean. Memes, being the latest contender for the proliferation of hate speech, put these social media companies to test. Whenever a derogatory or objectionable text is uploaded on the internet, it is almost always reported by someone which in turn is investigated by the site’s team. However, classifying a meme as offensive or not is a more complicated procedure than classifying text. Meme analysis involves a multimodal approach; it consolidates text and vision to interpret memes [1, 3].

In the past, work has been done on automatic meme generation and extraction of sentiments [2]. In spite of that, unexplored avenues like humor, sarcasm, and offense are yet to be broached. For this reason, this research, ‘Memotion Analysis’, needs to be put into practice to classify memes based on their sentiments and humor by the extraction of textual and visual features which would pinpoint hate speech over the internet [5].

## Goals and Objectives

The goal of this project is to channel research into the processing of internet memes through automated ways using deep learning. The objectives that will be covered in this project are as follows:

* Sentiment Classification: Inferring whether the meme presented is categorized as positive or negative.
* Humor Classification: Gathering from the text and context of the meme whether it falls into offensive, humorous, sarcastic or ‘others’ category. There can be more than one category for a meme.

## Scope of the Project

In order to streamline the entire process of analyzing memes for sentiment and humor classification, the task is decomposed into two major subdomains:

* Textual Sentiment Analysis: Subjective information is drawn from the pieces of text in the memes.
* Visual Sentiment Analysis: Subjective information is drawn from an image (meme) with respect to facial expressions and context recognition.

The techniques of computer vision and natural language processing are to be employed for the completion of the above stated methods. Through this, a model is to be designed which deals with the challenges associated with the semantic nature of text and image (the two components of a meme), and performs classification on the emotional responses gained. Moreover, any patterns or trends in the dataset are to be identified which would eventually help in determining further potential of research in previously overlooked areas.

This document serves to propose a solution to counter issues arising due to the propagation of offensive memes, on social media platforms, through ‘Memotion Analysis.’ All aspects of the work achieved (and the work to be achieved), in terms of literature survey, system requirements, and functional/non-functional requirements, are covered by this document to provide the reader with relevant information in this subject matter. Chapter 2 presents the related work and provides a concise description of all the findings of the literature survey for this subject. Chapter 3 lists down the functional, non-functional, hardware, and software requirements, and also describes the system architecture of the model proposed by this research. Chapter 4 elucidates the complete working of the proposed model and gives an explanation of the methodology adopted for the execution of this model. Chapter 5 concludes the report as this chapter provides an overall summary of the research.

# Chapter 2: Literature Survey / Related Work

For this project, various research papers have been studied and examined. From the literature survey below, it can be observed that most of them follow disparate approaches in analyzing the textual and visual sentiments depicted by memes.



## Detailed Literature Review

A detailed description of the findings of the literature survey is as follows.

### Analyzing Textual Features

Text sentiment analysis is pivotal in understanding the implication a meme carries. Research work in [4] focuses on generating meme descriptions through nonparanormal approaches which incorporate lexical features, part-of-speech features, named-entity features, frame-semantics features and dependency triples. For images that have text written on them, OCR technique is applied to segregate text from an image [9]. French [3], uncovers the multimodal nature of social media content by discovering a correlation between the semantic meaning of memes and the textual content of discussions in social media. The discussion threads are anatomized using Word and Phrase Analysis and an overall score is assigned to each of them; memes are assigned categorical tags with respect to the emotions depicted by their textual description. Through this research, it is realized that Phrase Analysis is better, at predicting and classifying sentiments in text, than Word Analysis. Another approach in [13], a novel Unsupervised Sentiment Analysis (USEA) framework, is developed which uses textual information of social media images for semantic information. The algorithm uses non-negative matrix factorization to assign a polarity (negative or positive) measure to the word with respect to the emotion it exudes and achieves an accuracy of 59.94%. This research is further supported by the model suggested by Subbaraj [11] which examines memes on Twitter. The model employs an amalgamation of Naïve Bayes and KNN to classify emotional connotations in text. Like USEA framework, it assigns a polarity of negative, positive, and neutral to the overall text of the word. The technique when compared to baselines such as SVM and Logistic Regression yields far better results (accuracy of 66.29% measured with KNN and Naïve Bayes).

### Analyzing Visual Features

Since the penetration of memes on the internet has been more through images than just text, thus visual sentiment analysis is just as critical as textual analysis. The work of Gajarla and Gupta [2] use emotional categories on images, and then perform classification on those categories. The categories are fear, sadness, love, happiness, and violence. About 1900 images on each category are collected from Flickr for training. The behavior of each category of image is tested on Places205-VGGNet-16, VGG-ImageNet, and RESNET and an accuracy of 73% is achieved. Chen, Li, and Cao [15] demonstrate that sentiments can also be extracted from images through processing of low-level features. The data gathered is fed into AdaBoost and Backpropagation Neural Network which attain a precision of 91.5% and 86.7% respectively. Similarly, the work of You, Lue, and Yang [1] exploits low-level features of an image, but it also takes the scene of an image into account (mid-level features). Another research implements Progressively Trained and Domain Transferred Deep Neural Networks in which half a million images are trained through 300,000 iterations on CNN and 100,000 iterations on PCNN reaching an accuracy up to 78.3% [7]. So, it can be seen that the use of deep learning techniques can greatly improve the process of image sentiment analysis. Another technique, Region Convolutional Neural Network searches for 2000 selective regions to obtain sentiments and produces excellent results [8].

Most of the approaches related to image sentiment analysis focus on extracting the sentiments of an image primarily through the analysis of facial expressions or textual descriptions. However, recognizing the context of the image is equally important. This situation occurs when an ambiguity arises in detecting emotions in an image [6]. Therefore, a method to overcome this limitation needs to be devised. CAER-Net is one such strategy which encompasses facial expressions and context information: it integrates a face encoding stream (only the faces in the images are selected) and a context encoding stream (the faces in the images are cropped out and attention is directed only to the scene). Both of these streams make use of CNNs with multiple max pooling, BN, and ReLU layers, and at the end, both networks are fused together through an additional number of convolution layers. This entire process yields an accuracy of 73.51% [6]. Wang and Li [12] further validate the importance of context recognition in image sentiment analysis through the employment of a supervised method, based on an attention model, to discern low-level and mid-level visual features. These attributes are then merged and mapped to a dictionary. After obtaining the emotional responses from an image, it is imperative to perceive the degree of attention they stimulate in human beings. The work in [10] makes use of a DNN which suggests that images involving humans capture more attention than others (greater stimulus intensity).

### Integrating Textual and Visual Features

Unmistakably, memotion analysis requires the inspection of textual and visual features [3,4,14]. For this, both of these features are derived from a meme to inter-relate their attributes using NLP and CV techniques. In addition to the textual features, the research also considers dense PHOW features for visual information and accordingly uses copula distribution to model its prediction architecture [4]. One more research work proposes a method where characteristics from 650 memes are acquired disjointedly and then combined for an overall classification. The text is passed through a series of processes including tokenization, stemming, stop word removal, part-of-speech tagging and sentiment classification, whereas, the image’s GLCM features are mined and sentiment labels are assigned to it through Adaboost Facial Recognition Stream. After this isolated preprocessing, both the features are fused together to be given as input to WEKA tool: J48 and Naïve Bayes are applied to the data. Consequently, it is observed that J48 and Naïve Bayes Classifier achieve an accuracy of 81% and 71% respectively [14]. While the concerned image and its text, both, are important in meme classification, but them alone are not sufficient for sentiment analysis; the context of the image needs to be known as well [12]. Moreover, a large training dataset is essential in developing a robust memotion analysis model [14].

## Literature Review Summary Table

A summary of the related work is listed in the table below.

Table 1: Research Work on Text Sentiment and Image Sentiment Analysis

The summary of research papers related to text and image sentiment analysis from 2013-2019 is presented here.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Name, reference | Author(s) | Year | Input | Output | Description |
| 1. | Robust Image Sentiment Analysis using Progressively Trained and Domain Transferred Deep Networks, [1] | Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang | 2015 | Dataset: Half million Images from Flickr | Accuracy for CNN: 78.3%;  Accuracy for PCNN: 77.3% | A pre-trained model is fine-tuned on CNN and PCNN on 300,000 and 100,000 iterations respectively. |
| 2. | Emotion Detection and Sentiment Analysis of Images, [2] | Vasavi Gajarla and Aditi Gupta | 2015 | Dataset: 9854 images from Flickr | Accuracy: 73% | Divides image sentiments into five categories and assigns weight to each sentiment. |
| 3. | Image-Based Memes as Sentiment Predictors, [3] | Jean H. French | 2017 | Dataset: Memes and discussion threads | Strong correlation discovered between the semantics of memes and discussion threads | Highlights impact and trends of memes in social media discussion threads. |
| 4. | A Nonparanormal Approach to Combining Textual and Visual Information for Predicting and Generating Popular Meme Descriptions, [4] | William Yang Wang and Miaomiao Wen | 2015 | Dataset: Memes and textual descriptions of memes | Textual descriptions for memes | Generates meme descriptions using different language modelling techniques and PHOW visual features. |
| 5. | Context-Aware Emotion Recognition Networks, [6] | Jiyoung Lee, Seungryong Kim, Sunok Kim, Jungin Park, and Kwanghoon Sohn | 2019 | Dataset: SFEW and FER-Wild | Accuracy for Caer-Net (Static): 73.51%  Accuracy for Caer-Net (Dynamic): 77.04% | Hides faces through attention mechanism and uses this to deduce emotions from context. |
| 6. | Emotional Attention: A Study of Image Sentiment and Visual Attention, [7] | Shaojing Fan, Zhiqi Shen, Ming Jiang, Bryan L. Koenig, Juan Xu, Mohan S. Kankanhalli, and Qi Zhao | 2018 | Dataset: EMOd (Emotional Content Images) | Accuracy: 83% | Emotional properties of images and visual attention given by people is realized. |
| 7. | Image Sentiment Analysis Using Deep Learning, [8] | Namita Mittal, Divya Sharma, and Manju Lata Joshi | 2018 | Dataset: Images from Flickr and Twitter | Comparison of techniques to recognize the best possible technique for a problem | Tells us which technique is better for what kind of problem. |
| 8. | Meme Opinion Categorization by Using Optical Character Recognition and Naïve Bayes Algorithms, [9] | Amalia Amalia, Amer Sharif, Fikri Haisar, Dani Gunawan, and Benny B. Nasution | 2018 | Dataset: 100 memes | Accuracy: 75% | Meme sentiment analysis is successfully implemented by using OCR and Naïve Bayes. |
| 9. | Image Sentiment Analysis Using Latent Correlations Among Visual, Textual, and Sentiment Views, [10] | Marie Katsurai and Shinichi Satoh | 2013 | Dataset: 105,587 images from Flickr and 120,000 images from Instagram | Accuracy for Flickr dataset: 74.77%  Accuracy for Instagram dataset: 73.60% | Uses latent correlations for analysis of visual and textual sentiment views in images. |
| 10. | Meme’tic Engineering to Classify Twitter Lingo, [11] | Priyashree S., Shivani N., Vigneshwar K., and Karthika S. | 2018 | Dataset: 1524 tweets and dynamic dataset using Twitter API | Accuracy for Naïve Bayes: 65.3% and Accuracy for KNN Classifier: 66.32% | Pattern characteristics of tweets are analyzed and their sentiments are predicted. |
| 11. | Sentiment Analysis for Social Media Images, [12] | Yilin Wang, and Baoxin Li | 2015 | Dataset: 120,221 from Flickr images and 130,230 images from Instagram | Accuracy for RSAI-Flickr: 0.57%  Accuracy for RSAI-Instagram: 0.62%  Accuracy for USEA-Flickr: 56.18% | Both textual and visual features are analyzed for examination of social media images. |
| 12. | Unsupervised Sentiment Analysis for Social Media Images, [13] | Yilin Wang, Suhang Wang, Jiliang Tang, Huan Liu, and Baoxin Li | 2015 | Dataset: 350,4192 images from Flickr and 131224 images from Instagram | Accuracy: 59.94% | Highlights the importance of how textual information helps in minimizing the semantic gap between image sentiments and visual characteristics. |
| 13. | Meme Classification Using Textual and Visual Features, [14] | E. Smitha, S. Sendhilkumar, and G. Mahalaksmi | 2018 | Dataset: 650 memes | Classification Accuracy in Naïve Bayes: 71% and J48: 81% | Features from dataset are extracted for classification based on sentiments. |
| 14. | An Adaboost-Backpropagation Neural Network for Automated Image Sentiment Classification, [15] | Jianfang Cao, Junjie Chen, and Haifang Li | 2014 | Dataset: 500 images with different styles | Recall: 91.5%; Precision: 86.7% | Achieves high level feature retrieval. |

Thus, through this research work, it can be seen that through the integration of textual and visual feature extraction, a successful and robust model for meme analysis can be constructed. It was also found that different people have different responses to memes. Therefore, it is of paramount importance to take ‘emotional response’ and context into consideration. It is anticipated that the inclusion of the findings of the past researches will yield better results.

# Chapter 3: Requirements and Design

This chapter gives an explanation of all modules of requirements and design for this project. The modules include descriptions of functional requirements, non-functional requirements, system requirements, and the system architecture.



## Functional Requirements

The functional requirements for this project are listed as follows.

### Sentiment Classification of Memes

The system shall accurately predict (and display) the sentiment of a meme as positive and negative when user inputs a meme.

### Humor Classification of Memes

The system shall accurately categorize (and display) a meme as sarcastic, humorous, or offensive when it is inputted by the user. If a meme does not fall under any of the above-mentioned categories, then it should fall in the ‘others’ category.

### Identifying Scales of Semantic Classes

The system shall be able to assign (and display) a degree to the sentiment and humor being expressed in the meme that is inputted by the user.

### Display Accuracies

The system shall display reasonable accuracies after completing training for sentiment, sarcasm, humor, and offense.

## Non-Functional Requirements

The non-functional requirements for this project are listed as follows.

### Scalability

The system shall be able to increase its operational activities i.e., increase workload if required in the future.

### Portability

The system shall be deployable on other platforms such as android, iOS, or web platforms.

### Performance

The system shall be efficient with minimum response time once it is trained on a vast dataset.

### Usability

The system shall have a consistent and user-friendly interface. The interface shall be provided after the deployment of the multi-model.

### Reliability

The system shall be reliable and will be able to recover easily in case of failure. This shall be achieved by maintaining multiple online backups.

### Manageability

The system shall be manageable i.e., easy to maintain, manage, and update, after its deployment.

### Reusability

The system shall be reusable as it would classify different elements in memes such as sentiment, humor, sarcasm, and offense, and these elements could be separately made use of.

### Availability

The system shall be in running state for all its users, after deployment.

## Hardware and Software Requirements

This section lists down the system requirements which are decomposed into hardware requirements and software requirements.

The hardware requirements are listed as follows:

* **Processor:** Core i5, 8th Generation hexa-core processor. It is required to carry out dataset pre-processing and computations.
* **Graphics Card:** Nvidia RTX 2060. It is required to train the neural network models of the pre-processed datasets.
* **Memory:** 16GB DDR4 RAM. It is required to load and provide runtime access to the dataset.
* **Hard Disk Drive:** 1TB 7200 RPM. It is required to load the datasets and store the results of different pre-processed tasks, computations, and machine learning models.
* **Power Supply Unit:** 550 Watts PSU. It is required to power all the aforementioned hardware specifications.

The software requirements are listed as follows:

* **Integrated Development Environment:** Jupyter Notebook, Pycharm
* **Drivers:** Nvidia CUDA, Nvidia cuDNN. These drivers (by Nvidia Corporation) are required to power the graphic card, and provide necessary utilities for the computations related to training of the model.
* **Python Libraries:** Numpy, Pandas, Sklearn, NLTK, Matplotlib, TensorFlow, TensorFlow-GPU, and PyTorch. These libraries are required to perform pre-processing, matrix computations, training of deep neural networks, and graph representations of models and results.

## System Architecture

This section provides the external and internal system architecture that defines the structure and behavior of the system.

### External System Architecture

The diagram below illustrates a generalized overview of the external architecture of this project. A dataset of over 6600 memes is used (see Appendix C) which contains OCR extracted text from the memes, and the meme image. The memes in the dataset are then preprocessed through segmentation, resizing, noise removal, and gray level conversion and the corresponding text is preprocessed through punctuation correction, stop word removal, spelling corrections, and stemming by pre-trained models.

After the preprocessing stages, feature extraction is performed separately for an image and its text. Visual features are extracted from an image using techniques of computer vision. The extraction of textual features, on the other hand, predict the sentiment of the OCR extracted text using pre-trained models of text classification and a text corpus of movies. Emotion is also extracted from an image as a feature attribute using pre-trained computer vision models.

All these extracted feature attributes are then combined and forwarded for the overall classification of images using deep learning techniques, the parameters of which are to be fine-tuned to achieve decent accuracies. Moreover, the techniques studied in past researches on image sentiment analysis and text classification are to be combined and served as baseline models.

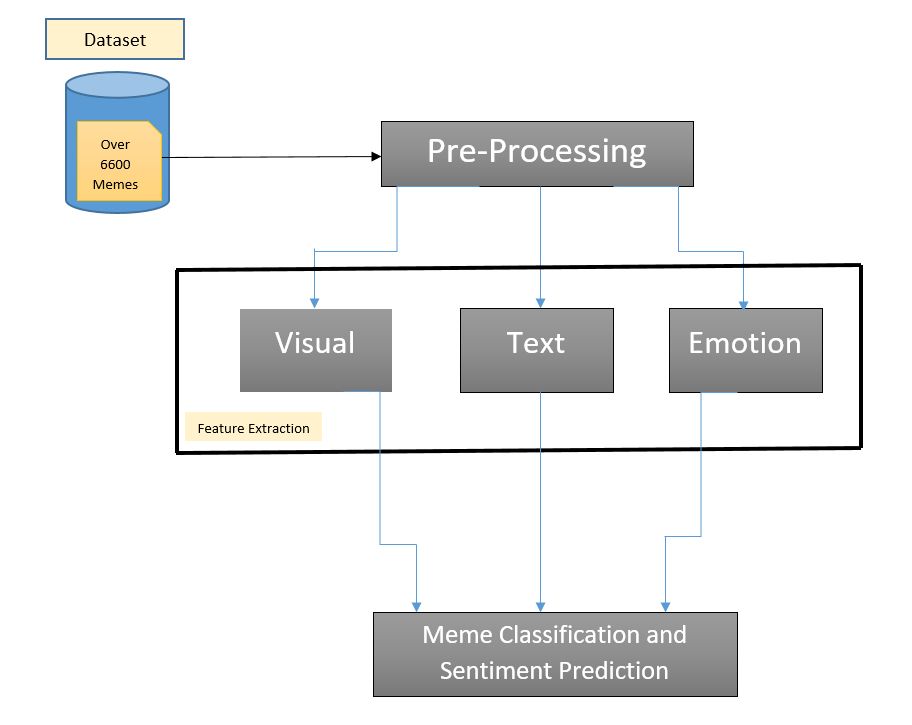


Figure 1: External System Architecture

The diagram presents a generalized overview of the main modules of the system.

### Internal System Architecture

The diagram below illustrates the detailed system architecture for this project. Refer to Chapter 4 to get a better understanding of the internal working of the system.

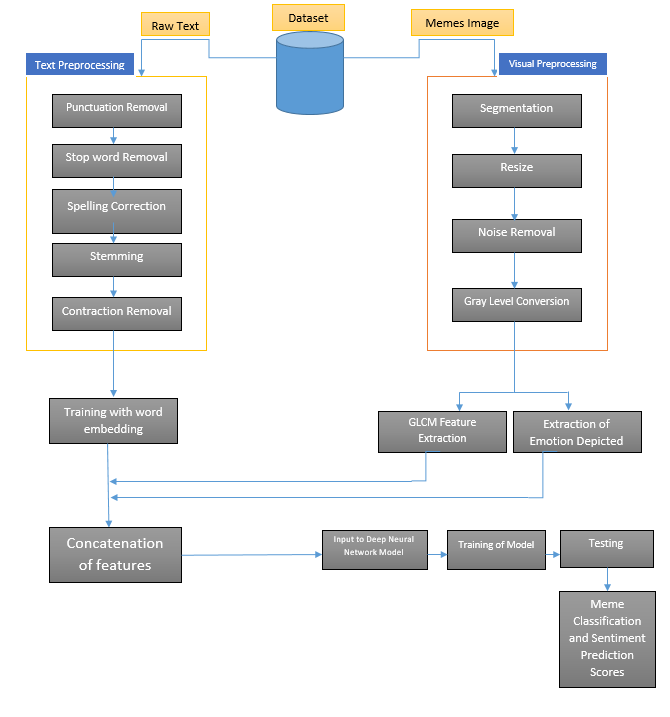


Figure 2: Internal System Architecture

The diagram presents a detailed overview of the main modules of the system.

### Assumptions

The system shall be constructed based on the following assumptions:

1. The dataset is in English language.
2. The users of the model are desktop and mobile literate, and educated enough to understand and operate the system.
3. The users of the model can identify which meme classifies as humorous, offensive, or sarcastic.

# Chapter 4: Implementation

This chapter provides a detailed description of the implementation of the prototype. The methodologies used for each stage of implementation are listed and described below.



## Text Preprocessing

To make the text of the dataset predictable and analyzable for this task, it is preprocessed through the below-mentioned techniques.

### Duplicates and Null Values

Null values are removed and duplicates are dropped.

### Punctuation Removal

The textual content of the memes in the dataset is cleaned i.e., unnecessary punctuations and numbers are removed since they do not carry any significant meaning.

### Lower Case Alphabets

The letter casing of the text in the dataset is changed to lower case since most words with uppercasing are not understood by pre-trained word embeddings.

### Removal of Contractions

Unnecessary contractions are changed back to the spoken form of the word to preserve the meaning of the sentences.

### Spelling Correction

Words having high number of occurrences and incorrect spellings are corrected.

### Removal of Special Characters

All unknown special characters are removed to clean the dataset for further computations.

## Textual Feature Engineering (Word Embeddings)

At this stage, the vocabulary and sentence of each meme fed into text-to-sequence models. Textual content of each meme is represented as a sequence of words but in numerical form. Different pre-trained word embedding models are used (as input) for this task; Glove, fastText, and Word2Vec are used to train the dataset vocabulary and sentences with respect to their intended semantic meaning. Meta-Embeddings (combination of two embeddings) may also be considered to improve the prediction of context in vocabulary.

### Result for Textual Classification

Accuracies for humor, sarcasm, and offense are 74%, 65%, and 56% respectively and they are achieved by using Glove embedding and convolutional neural network. Now, the dataset is to be trained on ELMo and BERT and the results are to be compared.

## Image Preprocessing

All textual content is to be cropped out of the images and the entire dataset is to be converted to a native resolution. After that, the image shall be preprocessed for noise removal using Gaussian filter.

## Visual Feature Engineering (GLCM Texture Features)

The preprocessed images are to be converted into their gray-level co-occurrence matrices (GLCMs) in order to compute their visual features such as homogeneity, contrast, dissimilarity, energy, entropy etc. Homogeneity and contrast are employed to achieve image segmentation.

## Emotion Detection

The meme dataset is to be processed by computer vision libraries to predict the emotion depicted in the images. The emotions are to be extracted as a feature attribute and this feature attribute shall be used as one of the inputs for the classifier model.

## Concatenation of Feature Attributes

All textual feature attributes, visual feature attributes, and emotion attributes are to be concatenated at different hierarchies of different neural network models.

## Model Fitting

Multiple deep neural network architectures are to be considered for the training and testing of the dataset. Among these deep neural networks, Convolution Neural Networks, Long-Short Term Memory Network, Bi-Directional Recurrent Neural Network are used. The output values pass through a sigmoid function to yield binary classifications based on the weights learned. The iterations of training are set to 100 to achieve a decent training accuracy and run the trained model on test data to compare the accuracy of predicted labels with actual labels.

## Application Development

The trained model will be saved to be deployed in the future; it will be incorporated in a graphical user interface such as desktop application, web application or android application. When given an input meme, the application will use the model to predict its sentiments and display a graded score to the end-user.

Hence it can be deduced that text preprocessing and image preprocessing are integral to the development of the model, and through the employment of the above-stated techniques, it can be successfully created.

# Chapter 5: Conclusion

The immense popularity of internet memes has sparked an interest in the field of their analysis and research. Their increased usage has opened newer avenues for the infiltration of hate speech over social media platforms and owing to this fact, precautionary measures are required by the respective organizations of such platforms. Hence, to cater this issue, this research proposes a learning model which efficiently categorizes the given input of memes with respect to their intended semantic meaning. The end-state vision of this research is to propose a model which is able to classify memes based on their intended sentiments with the highest accuracy possible and to identify the potential research gap.

Through past research work, it was inferred that, in comparison with visual features, textual features of a meme carry a greater significance in generating adequate results. It was also found that text embeddings yield better results than conventional machine learning feature engineering methods. So far, for text classification, combination of different embeddings in neural networks have been tried. The best results were yielded by Glove. This embedding was used in combination with a convolutional neural network which yielded 74% accuracy for humor, 65% accuracy for sarcasm, and 56% accuracy for offense.

Even though textual features are of substantial importance, meme analysis would not be accurate if only the text of the memes is considered. For this very reason, emotional responses and visual features are to be extracted and combined with textual features to improve results. Since, the responses of a meme are very subjective for each individual, and a positive score for text does not necessarily mean that the meme is positive, so visual features are taken into account to build a context-aware model. As of yet, visual features (homogeneity, contrast, energy, dissimilarity) of the memes have been extracted using GLCM matrix.

A challenge faced with visual feature extraction is that all images in the dataset are not of the same size, and cropping them all using the same measures results in a loss of pertinent information. Another challenge is that memes have text written over them and segregating the text from them might yield results that are not be as up to the mark. Though, there are techniques that allow image repainting but those techniques also use DNN which makes the entire process very expensive.

Given the constraints and limitations of the difficult interpretation of memes, adequate hardware resources were required to implement a model, which efficiently perform the classification of the input images. Due to this, a GPU is arranged which performs parallel execution of operations and training of deep neural learning model.

Up to the present time, the scope of the project has been substantially covered and text classification of memes has been achieved. Image sentiment analysis is yet to be broached because of complications in image resizing and segmentation. Time constraints and the availability of only one GPU were major obstacles in the development of the multimodal for meme analysis.

For FYP-2, state-of-the-art text classification models are to be implemented, and the text is to be classified and trained using ELMo, BERT, and ExcelNet for comparison. Moreover, recurrent neural networks are to be tried and tested. Once this is done, the visual and textual features are to be concatenated to generate results. Furthermore, a desktop application is to be developed for the proper deployment of this model.

# References

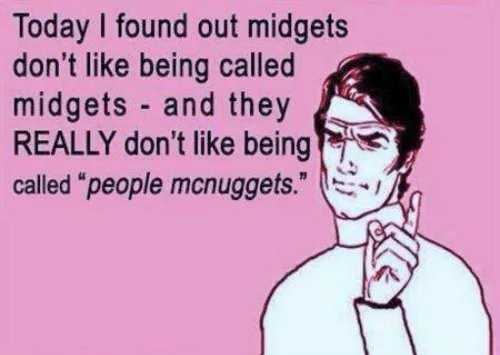
|  |  |  |
| --- | --- | --- |
| [1] |  | Q. You, J. Luo, H. Jin and J. Yang, "Robust Image Sentiment Analysis Using Progressively Trained and Domain Transferred Deep Networks," *Association for the Advancement of Artificial Intelligence (www.aaai.org),* pp. 1-7, 2015. |
| [2] |  | V. Gajarla and A. Gupta, "Emotion Detection and Sentiment Analysis of Images," *Georgia Institute of Technology,,* pp. 1-4, 2015. |
| [3] |  | J. French, "Image-Based Memes as Sentiment Predictors," *International Conference on Information Society (i-Society 2017),* pp. 1-5, 2017. |
| [4] |  | W. Wang and M. Wen, "I Can Has Cheezburger? A Nonparanormal Approach to Combining Textual and Visual Information for Predicting and Generating Popular Meme Descriptions," *Association for Computational Linguistics,,* pp. 1-10, 2015. |
| [5] |  | Semeval, "CodaLab - Memotion Analysis," competitions.codalab.org, 2019. [Online]. Available: https://competitions.codalab.org/competitions/20629#learn\_the\_details-overview. [Accessed 1 September 2019]. |
| [6] |  | J. Lee, S. Kim, J. Park and K. Sohn, "Context-Aware Emotion Recognition Networks," *Yonsei University, Ecole Polytechnique Federale de Lausanne (EPFL),* pp. 1-8, 2019. |
| [7] |  | S. Fan, Z. Shen, M. Jiang, B. Koenig, J. Xu, M. Kankanhalli and Q. Zhao, "Emotional Attention: A Study of Image Sentiment and Visual Attention," in *IEEE 2018/CVF Conference on Computer Vision and Pattern Recognition*, 2018. |
| [8] |  | N. Mittal, D. Sharma and M. L. Joshi, "Image Sentiment Analysis Using Deep Learning," in *IEEE 2018/WIC/ACM International Conference on Web Intelligence (WI)*, 2018. |
| [9] |  | A. Amalia, A. Sharif, F. Haisar, D. Gunawan and B. B. Nasution, "Meme Opinion Categorization by Using Optical Character Recognition (OCR) and Naïve Bayes Algorithm," in *Third International Conference on Informatics and Computing (ICIC)*, 2018. |
| [10] |  | M. Katsurai and S. Satoh, "Image Sentiment Analysis Using Latent Correlations Among Visual, Textual, and Sentiment Views," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016. |
| [11] |  | K. Subbaraj, "Meme'tic Engineering to Classify Twitter Lingo," in *International Conference on Computational Intelligence in Data Science (ICCIDS)*, 2018. |
| [12] |  | Y. Wang and B. Li, "Sentiment Analysis for Social Media Images," in *IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015. |
| [13] |  | Y. Wang, S. Wang, J. Tang and H. Liu, "Unsupervised Sentiment Analysis for Social Media Images," in *International Joint Conferences on Artificial Intelligence*, 2015. |
| [14] |  | E. S. Smitha, S. Sendhilkumar and G. S. Mahalaksmi, "Meme Classification Using Textual and Visual Features," *Computational Vision and Bio Inspired Computing,* pp. 1-5, 2018. |
| [15] |  | J. Cao, J. Chen and H. Li, "An Adaboost-Backpropagation Neural Network for Automated Image Sentiment Classification," *The Scientiﬁc World Journal,* vol. 2014, pp. 1-8, 2014. |

# Appendix

## Appendix A: Sample Memes

Figure 3: Sample Memes [5]

The above figures depict different kinds of memes circulating the internet.









## 

## Appendix B: Glossary

The definitions, acronyms, and abbreviations used in this document are arranged in alphabetical order below.

### Definitions

The definitions used in this document are listed as follows:

* **Batch Normalization** –Batch Normalization is performed on output layers of neural networks.
* **Computer Vison** – Computer vision deals with the manipulation of images and videos.
* **Context Aware Emotion Recognition Network** –A combination of different neural networks used for understanding the context of an image.
* **Convolutional Neural Network** – A neural network used in image processing related tasks.
* **Deep Neural Networks** – A complex neural network involving more than one layer and diverse mathematical techniques for data processing.
* **Dependency Triples** –It is an arc showing relationship between entities.
* **Facial Expression Recognition in the Wild** –Dataset used in [6].
* **Frame-Semantics Features** –This means that the consideration of all facts and knowledge regarding a word will help in understanding its meaning.
* **Gray Level Co-occurrence Matrix Features** –Different combinations of gray levels in an image.
* **Gray Level Co-occurrence Matrix Features** – Different combinations of gray levels in an image.
* **K-Nearest Neighbour** –The machine learning technique, used for classification problem, which works on the principle of similarity measure between entities involved.
* **Lexical Features** – Words which have an individualistic meaning such as nouns, verbs etc.
* **Logistic Regression** – It is used to predict the data and explain the relation of one object with another.
* **Meme** – An image that is virally transmitted over the internet and carries witty, sarcastic, and humorous undertones.
* **Memotion** – It is derived from the two words, ‘meme’ and ‘emotion’, and it essentially means the emotions of a meme.
* **Multiple Max Pooling** –Reduction in dimensionality (the hidden layer or output matrix).
* **Naïve Bayes** –The classification technique used in machine learning based on probabilistic methods.
* **Named-Entity Features** –Identification of specific and uniquely named entities.
* **Natural Language Processing** – The study of the language we speak.
* **Nonparanormal Approach** – An approach which is within the bounds of scientific understanding and is achievable.
* **Optical Character Recognition** – The process which involves the scanning of an image to extract text.
* **Part-Of-Speech Features** –Words used for naming purposes.
* **Pulse-Coupled Neural Network** –A neural network in image processing related tasks; it is a two-dimensional neural network.
* **Pyramid of Histogram of Visual Words Features** –This is used to extract low level visual features in an image.
* **Rectified Linear Unit** –It is used as an activation function for CNN.
* **Residual Network** –A typical neural network used for Computer Vison tasks.
* **Semantic** – Explanation of meaning of words and sentences.
* **Sentiment Analysis** – The process used to identify feelings/ opinions associated with certain text pieces and images.
* **Static Facial Expressions in the Wild** – Dataset used in [6].
* **Support Vector Machine** –The classifier used in supervised learning that categorizes data in a plane.
* **Tokenization** –If a sentence is parsed (broken into chunks), it is tokenized.
* **Unsupervised Sentiment Analysis** –The analysis of sentiments using techniques which are unsupervised (information is not labelled).

### Acronyms

The acronyms used in this document are listed as follows:

* **BERT** – Bidirectional Encoder Representations from Transformer
* **CAER-Net** – Context Aware Emotion Recognition Network
* **CUDA** – Compute Unified Device Architecture
* **ELMo** – Embeddings from Language Models
* **FER-Wild** – Facial Expression Recognition in the Wild
* **PHOW** – Pyramid of Histogram of Visual Words
* **RAM** – Random Access Memory
* **ReLU** – Rectified Linear Unit
* **RESNET** – Residual Network
* **USEA** – Unsupervised Sentiment Analysis

### Abbreviations

The abbreviations used in this document are listed as follows:

* **BN** – Batch Normalization
* **CNN** – Convolutional Neural Network
* **CV** – Computer Vision
* **DDR** – Double Data Rate
* **DNN** – Deep Neural Network
* **GLCM** – Gray Level Co-occurrence Matrix
* **GPU** – Graphical Processing Unit
* **IDE** – Integrated Development Environment
* **NLP** – Natural Language Processing
* **NLTK** – Natural Language Toolkit
* **OCR** – Optical Character Recognition
* **PCNN** – Pulse-Coupled Neural Network
* **PSU** – Power Supply Unit
* **RNN** – Recurrent Neural Network
* **RPM** – Revolutions Per Minute
* **SFEW** – Static Facial Expressions in the Wild
* **SVM** – Support Vector Machine
* **TB** – Terabyte

## Appendix C: Dataset

The dataset for memotion analysis is shown below. As it can be seen, it contains the OCR extracted text and its correction, the meme’s link, and the categories of sentiments it depicts.



Figure 4: Memotion Analysis Dataset

The dataset shown above contains OCR extracted text, the links to memes, and their semantic categories for over 6600 memes.

## Appendix D: Code

### Code for Visual Feature Extraction

The code for visual feature extraction is as follows:

import skimage

import os

from skimage.io import imread

from skimage.feature import greycomatrix

from skimage.feature import greycoprops

from datetime import datetime

import numpy as np

import cv2 as comp

import pandas as pd

start = datetime.now()

df = pd.read\_csv('./shuffled.csv', encoding='latin-1').drop\_duplicates()

df.dropna(inplace=True)

df2 = pd.DataFrame(index=range(df.shape[0]),columns=range(7))

df2.columns=['name', 'contrast', 'energy', 'homogeneity', 'correlation', 'dissimilarity', 'ASM']

print(df.shape[0])

i=0

directory=r'D:\FYP - Final Year Project\Task 8\data\_7000'

for filename in df.name:

    path = os.path.join(directory, filename)

    if(filename != 'got\_GOT-Meme-9.png'):

        im = imread(path, as\_gray=True)

        #im = imread('D:/FYP - Final Year Project/Task 8/data\_7000/got\_GOT-Meme-9.png', as\_gray=True)

        im = skimage.img\_as\_ubyte(im)

        im = im / 32

        im = im.astype(int)

        g = greycomatrix(im, [1], [0], levels=8, symmetric=False, normed=True)

        df2.loc[i, 'name'] = filename

        contrast = greycoprops(g, 'contrast')[0][0]

        df2.loc[i, 'contrast'] = round(contrast,2)

        energy = greycoprops(g, 'energy')[0][0]

        df2.loc[i, 'energy'] = round(energy,2)

        homogeneity = greycoprops(g, 'homogeneity')[0][0]

        df2.loc[i, 'homogeneity'] = round(homogeneity, 2)

        correlation = greycoprops(g, 'correlation')[0][0]

        df2.loc[i, 'correlation'] = round(correlation, 2)

        dissimilarity = greycoprops(g, 'dissimilarity')[0][0]

        df2.loc[i, 'dissimilarity'] = round(dissimilarity, 2)

        ASM = greycoprops(g, 'ASM')[0][0]

        df2.loc[i, 'ASM'] = round(ASM, 2)

        print("IMAGE", i,": ", filename)

        print('contrast is: ', contrast)

        print('energy is: ', energy)

        print('homogeneity is: ', homogeneity)

        print('correlation is: ', correlation)

        print('dissimilarity is: ', dissimilarity)

        print('ASM is: ', ASM)

        print()

    i += 1

df2.to\_csv('glcm.csv', index=False)

print()

print("time taken by this file:")

print(datetime.now() - start)

### Code for Textual Classification

The following is the code for textual classification:

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 177,

"metadata": {},

"outputs": [],

"source": [

"import numpy as np\n",

"import pandas as pd\n",

"import gensim.downloader as api\n",

"from keras.preprocessing.text import Tokenizer\n",

"from keras.preprocessing.sequence import pad\_sequences\n",

"from keras.layers import Dense, Input, GlobalMaxPooling1D, Conv1D, Embedding, MaxPooling1D\n",

"from keras.models import Model\n",

"from keras.models import load\_model\n",

"from keras.initializers import Constant\n",

"import matplotlib.pyplot as plt\n",

"from sklearn.metrics import classification\_report\n",

"from sklearn.metrics import accuracy\_score\n",

"\n",

"headers = [\"name\", \"link\", \"extraxted\", \"corrected\", \"Humor\",\n",

" \"Sarcasm\", \"offense\",\"Motivation\",\"sentiment\" ]\n",

"df = pd.read\_csv(\"./data.csv\", encoding='latin-1',\n",

" header=None, names=headers).sample(frac=1).drop\_duplicates()\n",

"df=df.drop([ \"Motivation\",\"sentiment\", \"name\", \"link\", \"extraxted\"], axis=1)\n",

"df.Sarcasm=df.Sarcasm.replace({\"twisted\_meaning\":1,\"very\_twisted\":1,\"general\": 0,\"not\_sarcastic\": 0})\n",

"df.Humor=df.Humor.replace({\"funny\":1,\"very\_funny\":1,\"hilarious\": 1,\"not\_funny\": 0})\n",

"df.offense=df.offense.replace({\"very\_offensive\":1,\"slight\":1,\"not\_offensive\": 0,\"hateful\_offensive\": 1})\n",

"df=df.dropna()\n",

"df=df.drop(df[(df['Humor'] !=0) & (df['Humor'] != 1)].index,axis=0)\n",

"df=df.drop(df[(df['offense'] != 0) & (df['offense'] != 1)].index,axis=0)\n",

"df=df.drop(df[(df['Sarcasm'] != 0) & (df['Sarcasm'] != 1)].index,axis=0)\n",

"df1=df.iloc[:496, :]\n",

"df=df.iloc[496:, :]"

]

},

{

"cell\_type": "code",

"execution\_count": 178,

"metadata": {},

"outputs": [],

"source": [

"embedding\_model = api.load(\"glove-wiki-gigaword-300\")"

]

},

{

"cell\_type": "code",

"execution\_count": 179,

"metadata": {},

"outputs": [],

"source": [

"prediction\_labels = [\"Humor\",\"Sarcasm\",\"offense\"]\n",

"labels = df[prediction\_labels].values\n",

"tokenizer = Tokenizer(num\_words=20000)\n",

"tokenizer.fit\_on\_texts(df['corrected'])\n",

"word\_index = tokenizer.word\_index\n",

"sequences = tokenizer.texts\_to\_sequences(df['corrected'])\n",

"\n",

"data = pad\_sequences(sequences, maxlen=100)\n",

"\n",

"num\_words = min(20000, len(word\_index)) + 1\n",

"embedding\_matrix = np.zeros((num\_words, 300))\n",

"for word, i in word\_index.items():\n",

" try:\n",

" embedding\_vector = embedding\_model.get\_vector(word)\n",

" if embedding\_vector is not None:\n",

" embedding\_matrix[i] = embedding\_vector\n",

" except:\n",

" continue\n",

"\n",

"embedding\_layer = Embedding(num\_words,\n",

" 300,\n",

" embeddings\_initializer=Constant(embedding\_matrix),\n",

" input\_length=100,\n",

" trainable=False)\n",

"\n",

"nb\_validation\_samples = int(0.2 \* data.shape[0])\n",

"x\_train = data[:-nb\_validation\_samples]\n",

"y\_train = labels[:-nb\_validation\_samples]\n",

"x\_val = data[-nb\_validation\_samples:]\n",

"y\_val = labels[-nb\_validation\_samples:]\n",

"input\_ = Input(shape=(100,))\n",

"x = embedding\_layer(input\_)\n",

"x = Conv1D(128, 5, activation='relu')(x)\n",

"x = MaxPooling1D(5)(x)\n",

"x = Conv1D(128, 5, activation='relu')(x)\n",

"x = MaxPooling1D(5)(x)\n",

"x = Conv1D(128, 3, activation='relu')(x)\n",

"x = GlobalMaxPooling1D()(x)\n",

"x = Dense(128, activation='relu')(x)\n",

"output = Dense(len(prediction\_labels), activation='sigmoid')(x)\n",

"model = Model(input\_, output)\n",

"model.compile(loss='binary\_crossentropy',\n",

"optimizer='rmsprop',\n",

" metrics=['acc'])"

]

},

{

"cell\_type": "code",

"execution\_count": 180,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Train on 4812 samples, validate on 1203 samples\n",

"Epoch 1/50\n",

"4812/4812 [==============================] - 2s 489us/step - loss: 0.6325 - acc: 0.6556 - val\_loss: 0.5993 - val\_acc: 0.7041\n",

"Epoch 2/50\n",

"4812/4812 [==============================] - 2s 453us/step - loss: 0.6080 - acc: 0.6971 - val\_loss: 0.6020 - val\_acc: 0.7041\n",

"Epoch 3/50\n",

"4812/4812 [==============================] - 2s 454us/step - loss: 0.5974 - acc: 0.6971 - val\_loss: 0.6199 - val\_acc: 0.7024\n",

"Epoch 4/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.5943 - acc: 0.7016 - val\_loss: 0.5992 - val\_acc: 0.7032\n",

"Epoch 5/50\n",

"4812/4812 [==============================] - 2s 459us/step - loss: 0.5790 - acc: 0.6996 - val\_loss: 0.6381 - val\_acc: 0.6902\n",

"Epoch 6/50\n",

"4812/4812 [==============================] - 2s 459us/step - loss: 0.5707 - acc: 0.7070 - val\_loss: 0.6066 - val\_acc: 0.6974\n",

"Epoch 7/50\n",

"4812/4812 [==============================] - 2s 466us/step - loss: 0.5532 - acc: 0.7206 - val\_loss: 0.6120 - val\_acc: 0.6880\n",

"Epoch 8/50\n",

"4812/4812 [==============================] - 2s 458us/step - loss: 0.5285 - acc: 0.7261 - val\_loss: 0.7014 - val\_acc: 0.5810\n",

"Epoch 9/50\n",

"4812/4812 [==============================] - 2s 461us/step - loss: 0.5137 - acc: 0.7519 - val\_loss: 0.7164 - val\_acc: 0.6412\n",

"Epoch 10/50\n",

"4812/4812 [==============================] - 2s 465us/step - loss: 0.4984 - acc: 0.7519 - val\_loss: 0.6445 - val\_acc: 0.6528\n",

"Epoch 11/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.4663 - acc: 0.7850 - val\_loss: 0.6549 - val\_acc: 0.6678\n",

"Epoch 12/50\n",

"4812/4812 [==============================] - 2s 478us/step - loss: 0.4521 - acc: 0.8084 - val\_loss: 0.6563 - val\_acc: 0.6838\n",

"Epoch 13/50\n",

"4812/4812 [==============================] - 2s 467us/step - loss: 0.4006 - acc: 0.8231 - val\_loss: 0.7258 - val\_acc: 0.5930\n",

"Epoch 14/50\n",

"4812/4812 [==============================] - 2s 457us/step - loss: 0.4046 - acc: 0.8168 - val\_loss: 0.6896 - val\_acc: 0.6730\n",

"Epoch 15/50\n",

"4812/4812 [==============================] - 2s 463us/step - loss: 0.3482 - acc: 0.8479 - val\_loss: 0.7672 - val\_acc: 0.5630\n",

"Epoch 16/50\n",

"4812/4812 [==============================] - 2s 451us/step - loss: 0.3666 - acc: 0.8428 - val\_loss: 0.8023 - val\_acc: 0.6536\n",

"Epoch 17/50\n",

"4812/4812 [==============================] - 2s 456us/step - loss: 0.3199 - acc: 0.8604 - val\_loss: 0.8408 - val\_acc: 0.6930\n",

"Epoch 18/50\n",

"4812/4812 [==============================] - 2s 462us/step - loss: 0.3008 - acc: 0.8751 - val\_loss: 1.1277 - val\_acc: 0.5151\n",

"Epoch 19/50\n",

"4812/4812 [==============================] - 2s 461us/step - loss: 0.2964 - acc: 0.8836 - val\_loss: 0.9072 - val\_acc: 0.5683\n",

"Epoch 20/50\n",

"4812/4812 [==============================] - 2s 458us/step - loss: 0.2782 - acc: 0.8843 - val\_loss: 0.7908 - val\_acc: 0.6207\n",

"Epoch 21/50\n",

"4812/4812 [==============================] - 2s 460us/step - loss: 0.2466 - acc: 0.9136 - val\_loss: 0.8290 - val\_acc: 0.6572\n",

"Epoch 22/50\n",

"4812/4812 [==============================] - 2s 453us/step - loss: 0.2187 - acc: 0.9327 - val\_loss: 0.9638 - val\_acc: 0.6736\n",

"Epoch 23/50\n",

"4812/4812 [==============================] - 2s 452us/step - loss: 0.2333 - acc: 0.9216 - val\_loss: 1.1213 - val\_acc: 0.6536\n",

"Epoch 24/50\n",

"4812/4812 [==============================] - 2s 451us/step - loss: 0.1819 - acc: 0.9326 - val\_loss: 0.9848 - val\_acc: 0.6437\n",

"Epoch 25/50\n",

"4812/4812 [==============================] - 2s 451us/step - loss: 0.2653 - acc: 0.9187 - val\_loss: 0.9455 - val\_acc: 0.6534\n",

"Epoch 26/50\n",

"4812/4812 [==============================] - 2s 452us/step - loss: 0.2148 - acc: 0.9249 - val\_loss: 1.1803 - val\_acc: 0.6761\n",

"Epoch 27/50\n",

"4812/4812 [==============================] - 2s 452us/step - loss: 0.1080 - acc: 0.9685 - val\_loss: 1.3219 - val\_acc: 0.6293\n",

"Epoch 28/50\n",

"4812/4812 [==============================] - 2s 462us/step - loss: 0.2072 - acc: 0.9186 - val\_loss: 1.0502 - val\_acc: 0.5957\n",

"Epoch 29/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.0728 - acc: 0.9841 - val\_loss: 1.1473 - val\_acc: 0.6104\n",

"Epoch 30/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.2418 - acc: 0.9401 - val\_loss: 1.1870 - val\_acc: 0.6542\n",

"Epoch 31/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.1781 - acc: 0.9451 - val\_loss: 1.1217 - val\_acc: 0.6423\n",

"Epoch 32/50\n",

"4812/4812 [==============================] - 2s 463us/step - loss: 0.0703 - acc: 0.9821 - val\_loss: 2.1656 - val\_acc: 0.6559\n",

"Epoch 33/50\n",

"4812/4812 [==============================] - 2s 462us/step - loss: 0.1825 - acc: 0.9756 - val\_loss: 1.2772 - val\_acc: 0.6570\n",

"Epoch 34/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.1348 - acc: 0.9700 - val\_loss: 1.3342 - val\_acc: 0.5666\n",

"Epoch 35/50\n",

"4812/4812 [==============================] - 2s 465us/step - loss: 0.0748 - acc: 0.9818 - val\_loss: 1.2415 - val\_acc: 0.6348\n",

"Epoch 36/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.0411 - acc: 0.9908 - val\_loss: 1.5036 - val\_acc: 0.5666\n",

"Epoch 37/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.2350 - acc: 0.9528 - val\_loss: 1.2608 - val\_acc: 0.6318\n",

"Epoch 38/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.0402 - acc: 0.9912 - val\_loss: 1.3958 - val\_acc: 0.6176\n",

"Epoch 39/50\n",

"4812/4812 [==============================] - 2s 466us/step - loss: 0.2508 - acc: 0.9627 - val\_loss: 1.2840 - val\_acc: 0.6365\n",

"Epoch 40/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.0400 - acc: 0.9915 - val\_loss: 1.5163 - val\_acc: 0.6392\n",

"Epoch 41/50\n",

"4812/4812 [==============================] - 2s 463us/step - loss: 0.0338 - acc: 0.9920 - val\_loss: 1.5914 - val\_acc: 0.6107\n",

"Epoch 42/50\n",

"4812/4812 [==============================] - 2s 463us/step - loss: 0.2955 - acc: 0.9386 - val\_loss: 1.2922 - val\_acc: 0.6290\n",

"Epoch 43/50\n",

"4812/4812 [==============================] - 2s 462us/step - loss: 0.0351 - acc: 0.9936 - val\_loss: 1.4689 - val\_acc: 0.6362\n",

"Epoch 44/50\n",

"4812/4812 [==============================] - 2s 463us/step - loss: 0.0282 - acc: 0.9927 - val\_loss: 1.6133 - val\_acc: 0.6451\n",

"Epoch 45/50\n",

"4812/4812 [==============================] - 2s 464us/step - loss: 0.1659 - acc: 0.9737 - val\_loss: 1.3719 - val\_acc: 0.5658\n",

"Epoch 46/50\n",

"4812/4812 [==============================] - 2s 465us/step - loss: 0.0482 - acc: 0.9909 - val\_loss: 1.4162 - val\_acc: 0.6312\n",

"Epoch 47/50\n",

"4812/4812 [==============================] - 2s 468us/step - loss: 0.0267 - acc: 0.9938 - val\_loss: 1.6613 - val\_acc: 0.6359\n",

"Epoch 48/50\n",

"4812/4812 [==============================] - 2s 465us/step - loss: 0.2498 - acc: 0.9569 - val\_loss: 1.3436 - val\_acc: 0.6323\n",

"Epoch 49/50\n",

"4812/4812 [==============================] - 2s 472us/step - loss: 0.0304 - acc: 0.9939 - val\_loss: 1.4923 - val\_acc: 0.6301\n",

"Epoch 50/50\n",

"4812/4812 [==============================] - 2s 469us/step - loss: 0.0255 - acc: 0.9940 - val\_loss: 1.6384 - val\_acc: 0.6345\n"

]

}

],

"source": [

"md = model.fit(x\_train, y\_train, epochs=50, batch\_size=512, validation\_data=(x\_val, y\_val))"

]

},

{

"cell\_type": "code",

"execution\_count": 181,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "meme1",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

},

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Sarcasmania Model Accuracy: 99.39733743667603 %.\n"

]

}

],

"source": [

"plt.plot(md.history['acc'], label='Accuracy Plot')\n",

"plt.legend()\n",

"plt.show()\n",

"print(\"Sarcasmania Model Accuracy: \", md.history['acc'][-1]\*100, \"%.\")"

]

},

{

"cell\_type": "code",

"execution\_count": 182,

"metadata": {},

"outputs": [],

"source": [

"model.save('sarcasmania\_model.h5')"

]

},

{

"cell\_type": "code",

"execution\_count": 183,

"metadata": {},

"outputs": [],

"source": [

"def create\_prediction(model, sequence, tokenizer, max\_length, prediction\_labels):\n",

" # Convert the sequence to tokens and pad it.\n",

" sequence = tokenizer.texts\_to\_sequences(sequence)\n",

" sequence = pad\_sequences(sequence, maxlen=max\_length)\n",

" \n",

" # Make a prediction\n",

" sequence\_prediction = model.predict(sequence, verbose=1)\n",

" \n",

" # Take only the first of the batch of predictions\n",

" sequence\_prediction = pd.DataFrame(sequence\_prediction).round(3)\n",

" \n",

" # Label the predictions\n",

" sequence\_prediction = np.where(sequence\_prediction>=0.5, 1, 0)\n",

" \n",

" return sequence\_prediction"

]

},

{

"cell\_type": "code",

"execution\_count": 184,

"metadata": {

"scrolled": true

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"496/496 [==============================] - 0s 222us/step\n",

"Humour Accuracy: 0.6895161290322581\n",

"Sarcasm Accuracy: 0.6451612903225806\n",

"Offense Accuracy: 0.5\n"

]

}

],

"source": [

"sequence = tokenizer.texts\_to\_sequences(df1['corrected'])\n",

"sequence = pad\_sequences(sequence, maxlen=100)\n",

"sequence\_prediction = model.predict(sequence, verbose=1)\n",

"sequence\_prediction = pd.DataFrame(sequence\_prediction).round(3)\n",

"sequence\_prediction = np.where(sequence\_prediction>=0.5, 1, 0)\n",

"sequence\_prediction = pd.DataFrame.from\_records(sequence\_prediction)\n",

"\n",

"df2=df1.iloc[:,-3:].astype('int64')\n",

"\n",

"y\_true=sequence\_prediction.iloc[:,0]\n",

"y\_pred=df2.iloc[:,0].astype('int64')\n",

"y\_pred.reset\_index(drop=True, inplace=True)\n",

"acc=accuracy\_score(y\_true, y\_pred)\n",

"print(\"Humour Accuracy: \", acc)\n",

"\n",

"y\_true=sequence\_prediction.iloc[:,1]\n",

"y\_pred=df2.iloc[:,1].astype('int64')\n",

"y\_pred.reset\_index(drop=True, inplace=True)\n",

"acc=accuracy\_score(y\_true, y\_pred)\n",

"print(\"Sarcasm Accuracy: \", acc)\n",

"\n",

"y\_true=sequence\_prediction.iloc[:,2]\n",

"y\_pred=df2.iloc[:,2].astype('int64')\n",

"y\_pred.reset\_index(drop=True, inplace=True)\n",

"acc=accuracy\_score(y\_true, y\_pred)\n",

"print(\"Offense Accuracy: \", acc)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.2"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}