Precog recruitment task

Date:-18/07/24

Task-2 Analyzing hateful memes

Name:-B.Surya Kranthi Vardhan
Affiliation:-B-tech,Computer and Communication Engineering,Amrita vishwa vidyapeetham,Chennai
E-mail:-kranthiballa633@gmail.com
Phone no.:-7989665757

- I have attempted all the three compulsory sub-tasks including bonus task associated with the task.
- I obtained the dataset from the provided link in the tasks PDF. Due to the large size of the dataset, I selected a subset to reduce time complexity.
- For classification,prediction task I took the labeled dataset from online and included the provided dataset in that as a part.
- The dataset which have been taken from online is uploaded in the github repo.
- For bonus task I used the dataset which has only ocr text coloumn and rest of the unwanted coloumns have been removed.

1. Object detection and Frequency of the objects detected

This task has been done in google colab and dataset is taken from here.As the dataset is huge which contains10000+ images i took subset of the dataset with 500(hateful&not hateful) images.The dataset has been zipped and uploaded to google drive and, used to read the dataset in google colab from google drive.

1.1 Approach for object detection:

The main aim of this task is to detect the objects in memes using computer vision techniques and my approach unfolds as follows:

i. **Importing Required Libraries:-**The libraries used for this task are

- os:-used for file path operations.
- Torch and Torchvision:-These are the core libraries for pytorch and include models and utilities
- Transformers form Torchvision:-It is used to preprocess images.
- PIL(Python Imaging Library):-It is used for opening images.
- Matplotlib and cv2(OpenCV):-For visualizing and saving the results.

ii. **Loading the pre-existing model:-**I have used faster R-CNN model which is one of the most accurate object detection algorithms but a challenging thing for me while using this model is, it require lot of power at inference time.There are other choices based upon specific needs.

iii. **Transformation(Convert-images to tensors)**:-By converting these tensors we can train our model with this data.We can mirror image,resize,crop and manipulate in other ways that are useful to us.

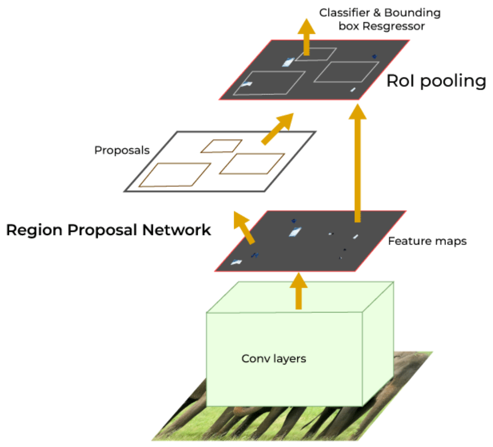
iv. **Performing inference:-**Here it checks whether the image is in RGB or not and also adds the dimensions to the specific image.

v.**Visualizing:-**After performing inference ‘cv2.imread’ loads the image using opencv.The output contains detected bounding boxes and saved in the results directory.

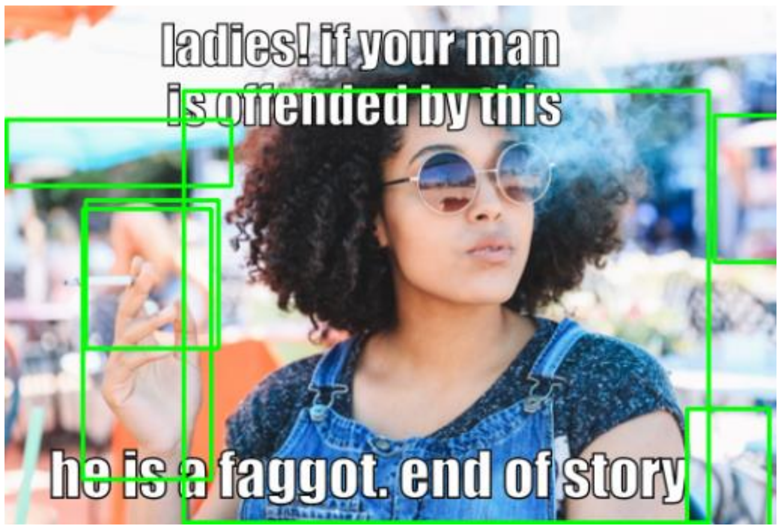
1.2 Inference and Results:-

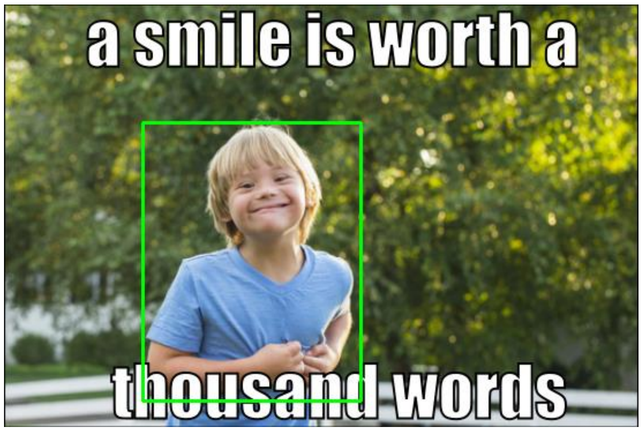
Therefore by using opencv and pytorch I have performed object detection for the dataset provided.The output will have green coloured rectangular bounding boxes for the detected objects. Faster R-CNN is preferred for object detection due to its speed. It replaces slow, external region proposal methods in R-CNN and Fast R-CNN with a built-in Region Proposal Network (RPN) for efficient proposal generation. This makes Faster R-CNN faster and more accurate.which is one of the good choice in pre existing models.The output will get saved in a ‘results’ directory.

Block diagram for Faster R-CNN:-



Here are the examples of result:-





1.3 Approach for frequency of the objects detected:

The main aim of this task is to evaluate the frequency of objects detected in memes using computer vision techniques and my approach unfolds as follows:

i. **COCO Class Labels:-**I defined a list of COCO class labels to map class indices to human-readable class names.

ii. **Model loading and Image transformation**:-As explained in 1.1 i have use pre-existing model i.e faster R-CNN and transformed images to tensors and performed object detection.

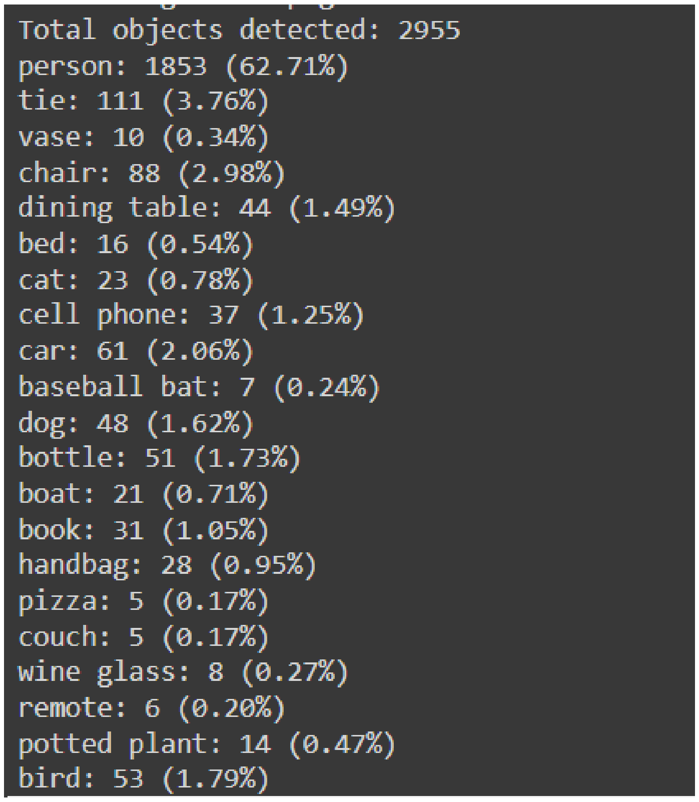
iii. **Process the data and cataloging objects:-**It is the process of creating a detailed record for each item in a collection. This record serves as a way to identify, describe, and organize the objects for easy retrieval, research, and management.

iv. **Frequency:-** Each image in our dataset directory will be processed, and identifies the frequency of each object in that image.The output will be printed as text format in the terminal.

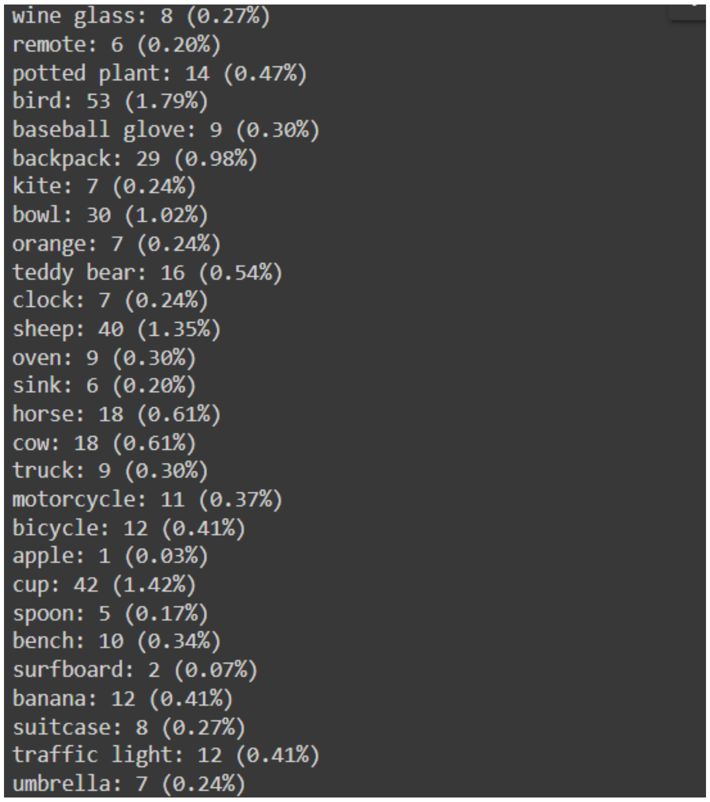
1.4 Inference and Results:-

Therefore, I have evaluated the frequency of objects for the dataset provided.The output will have a printed summary in the terminal that includes the total number of objects detected across the dataset and a breakdown of each object type detected, along with its count and percentage of the total detections.

Here are the examples of result:-



```
Total objects detected: 2955
person: 1853 (62.71%)
tie: 111 (3.76%)
vase: 10 (0.34%)
chair: 88 (2.98%)
dining table: 44 (1.49%)
bed: 16 (0.54%)
cat: 23 (0.78%)
cell phone: 37 (1.25%)
car: 61 (2.06%)
baseball bat: 7 (0.24%)
dog: 48 (1.62%)
bottle: 51 (1.73%)
boat: 21 (0.71%)
book: 31 (1.05%)
handbag: 28 (0.95%)
pizza: 5 (0.17%)
couch: 5 (0.17%)
wine glass: 8 (0.27%)
remote: 6 (0.20%)
potted plant: 14 (0.47%)
bird: 53 (1.79%)
```



```
wine glass: 8 (0.27%)
remote: 6 (0.20%)
potted plant: 14 (0.47%)
bird: 53 (1.79%)
baseball glove: 9 (0.30%)
backpack: 29 (0.98%)
kite: 7 (0.24%)
bowl: 30 (1.02%)
orange: 7 (0.24%)
teddy bear: 16 (0.54%)
clock: 7 (0.24%)
sheep: 40 (1.35%)
oven: 9 (0.30%)
sink: 6 (0.20%)
horse: 18 (0.61%)
cow: 18 (0.61%)
truck: 9 (0.30%)
motorcycle: 11 (0.37%)
bicycle: 12 (0.41%)
apple: 1 (0.03%)
cup: 42 (1.42%)
spoon: 5 (0.17%)
bench: 10 (0.34%)
surfboard: 2 (0.07%)
banana: 12 (0.41%)
suitcase: 8 (0.27%)
traffic light: 12 (0.41%)
umbrella: 7 (0.24%)
```

Frequency of the objects printed in the terminal

2. Caption Impact Assessment

This task has been done in google colab and dataset is taken from here.As the dataset is huge which contains10000+ images i took subset of the dataset with 500(hateful&not hateful) images.The dataset has been zipped and uploaded to google drive and, used to read the dataset in google colab from google drive.

2.1 Approach for Caption impact:

The main aim of this task is to asses the effect of captions on the image using OCR techniques and my approach unfolds as follows:

i. **Extract Text Using OCR:-** pytesseract is the library which is used to extract the text from meme images and PIL for image processing.

ii. **Overlay Extracted Text:-** The add_text function overlays the extracted text back onto the images.

iii. **Process the dataset:-**After extracting the text and overlaying the text it performs object detection and computes the detection ration and average IOU.

- ❖ Detection ratio:-It is a measure of how many objects are detected in the captioned images compared to the original images.

$$\text{Detection Ratio} = \frac{\text{Number of objects detected in captioned image}}{\text{Number of objects detected in original image}}$$

If the average detection ratio is 1, it means that the presence of captions does not affect the number of detected objects. If it is less than 1, it means fewer objects are detected in the captioned images.

- ❖ IOU(Intersection Over Union):-It is a metric used to evaluate the overlap between two bounding boxes.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

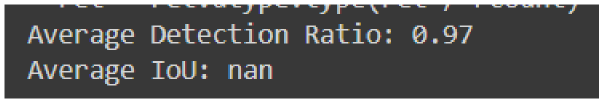
While performing IOU I got I as NaN value because few images in dataset contains only one object where there is no overlapping between bounding boxes,at this time intersection area becomes zero. In such cases, the IoU formula division by zero would result in NaN. It became a challenging task for me but we can try to overcome this using Instead of NaN, you can assign a default value (e.g., 0) when the IoU calculation results in NaN. This approach helps in maintaining consistent metrics calculations across all images.

iv.**Output Metrics:-** After processing all images, the code prints the average detection ratio and average IoU, providing insights into the impact of the pre-existing text on object detection performance.

2.2 Inference and Results:-

Therefore by using opencv and pytesseract(OCR) I have performed the impact of overlaid captions on the accuracy and effectiveness of object detection.The output will be printed as detection ratio and metrics IOU in text format.I got average detection ratio as 0.97 which is near to 1.It means the impact is less in the give dataset.

Here is the output image:-



```
Average Detection Ratio: 0.97
Average IoU: nan
```

3. Classification of hateful and non-hateful memes

This task has been done in google colab and to perform this task we need to label the dataset,as the dataset is huge the process will be complex to label the provided dataset,so I took the labeled dataset from kaggle and included 100 images which I labelled them from our given dataset. The dataset has been zipped and uploaded to google drive and, used to read the dataset in google colab from google drive.

3.1 Approach for Classification:

The main aim of this task is to classify the images based on something non-trivial.I classified the memes by taking 5 labels i.e positive, very_positive, neutral, negative, very_negative and my approach unfolds as follows:

i**. Extract text using OCR:-**I used easy OCR to extract the text from images and this extract text is saved in the csv file.

ii. **Labeling the dataset**:-I labeled 100 to 200 images of dataset and include that in the existing dataset which is taken from the online.

- **Image classification:**

i. **Reading the data:-** I've read the data given in the CSV (.csv) file, which contains the extracted text (corrected) from each image along with its sentiment, and stored it in a pandas. I've assigned numerical values to each of the sentiments corresponding to the images, ranging from '0' for a sentiment of "neutral" to '4' for a sentiment of "very_negative"

ii. **Determining if there's an imbalance between classes**:-Imbalanced classes are one of the most common problems in machine learning, which make the machine learning classification models significantly inaccurate.

iii. **Balancing the imbalanced classes**:- For this, I've used a common balancing method called up-sampling, which is the process of randomly duplicating observations from the minority class in order to reinforce its signal.

iv. **Extracting the images from the specified directory:-**In the specified directory is a collection of images (memes) that need to be read individually for their image data and storing them in a list. The scikit-image module accomplishes this with the skimage.io.imread function.

v. **Pre-processing the data:-** I've split the image data into training and testing sets using the sklearn.model_selection.train_test_split function. Both the training and testing images are then converted to greyscale to ensure uniformity, and resized to the same smaller resolution for more consistent and efficient processing.

vi. **Random Forest Classifier:-**I have used random forest which is an ensemble learning method for classification after that predicting the labels, Calculating the accuracy and Displaying the confusion matrix for the classification model.

vii. **K-Nearest Neighbours Classifier:-** k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions after that predicting the labels, Calculating the accuracy and Displaying the confusion matrix for the classification model.

viii. **Extra Trees Classifier:-** The extra trees algorithm, like the random forests algorithm, creates many decision trees, but the sampling for each tree is random, without replacement. after that predicting the labels, Calculating the accuracy and Displaying the confusion matrix for the classification model.

ix. **Testing the most accurate classification:-** Now that I've implemented and trained all three models successfully,the most accurate one out of all, which is the ExtraTreesClassifier model, against a custom image (meme) to see if the predicted result is correct or not.

▪ **<u>Text Classification:</u>**

i. Same as in image classification i read the data again, Determining if there's an imbalance between classes,if there is any imbalanced classes I done balancing the imbalancing classes.

ii. **One-hot encoding to the image sentiments:-** One-hot encoding is used for converting categorical data into a numerical format that can be used by machine learning algorithms. Specifically, it transforms each category value into a new binary (0 or 1) column, with a single high bit (1) and all other bits low (0).

iii. **Pre-processing the data:-** I've used sklearn.feature_extraction.text.Tfidf Vectorizer to convert the raw text into a matrix of word counts. Additionally, it can also filter out any stop-words, which can make the machine learning classification models more accurate.

x. **Stochastic Gradient Descent Classifier:-** Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable),after that predicting the labels, Calculating the accuracy and Displaying the confusion matrix for the classification model.

xi. **Multinomial Naïve Bayes Classifier:-** The Multinomial Naïve Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem, after that predicting the labels, Calculating the accuracy and Displaying the confusion matrix for the classification model.

iv. **Logistic Regression:-** Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables.

v. **Testing the most accurate classification:-** Now that I've implemented and trained all three text classification models successfully, I can test the most accurate one out of all, which is the Multinomial Naïve Bayes model, against text extracted from a custom image (meme) to see if the predicted result is correct or not.

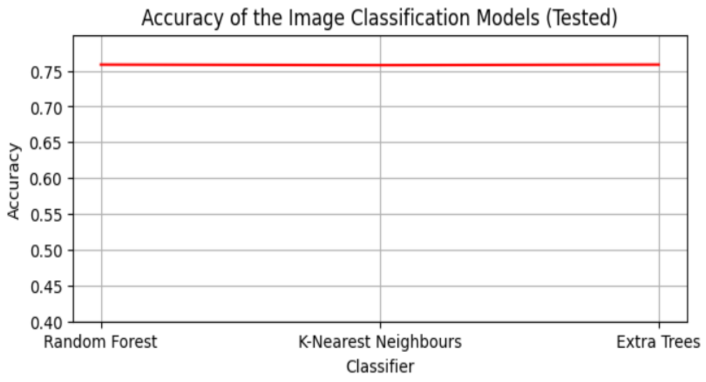
vi. **Comparison (Accuracy):-**I plotted the accuracies of all six analysed machine learning classification models.

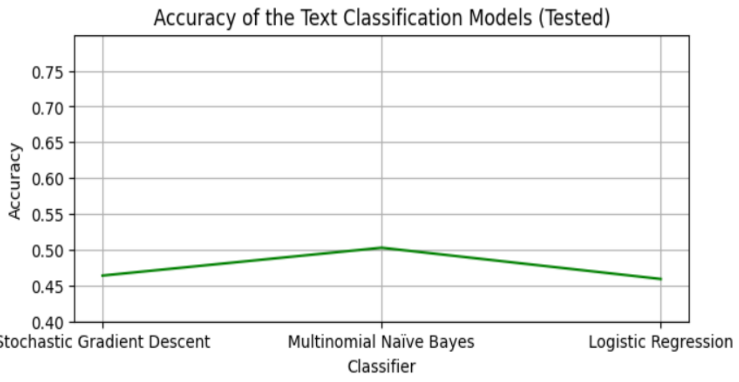
3.2 Inference and Results:-

Therefore, Based on the accuracies all six analysed machine learning classification models, it's clear that the ExtraTreesClassifier is the most accurate one when it comes to analysing the sentiment(s) behind images (memes), with an accuracy of approximately 75.85%. Of course, this isn't highly accurate either, since machine learning algorithms generally perform poorly.

It's also worth noting that, compared to the image classification models, the text classification models for significantly more inaccurate, which could be due to the imbalance in the classes of the pandas.DataFrame. While balancing the classes increased the accuracy significantly in the case of image classification, there are still some discrepancies in the analysed text classification models that are causing some classes to remain unaccounted for. However, I've still managed to achieve relatively decent accuracies with the text classification models, with the most accurate being the Multinomial Naïve Bayes model, having an accuracy of approximately 50.27%.

Here is the output comparision image:-





4. <u>Bonus task-Predicting whether meme is toxic or not</u>

This task has been done in google colab and i used the dataset which has only ocr text coloumn and rest of the unwanted coloumns have been removed.The dataset has been uploaded to google drive and, used to read the dataset in google colab from google drive.

4.1 Approach for Prediction:

The main aim of this task is to predict whether or not a meme is toxic, simply based on the text in the image.I used NLTK sentiment analysis for performing this task and my approach unfolds as follows:

i. **OCR extraction:-**I have extracted the text from meme images and remove ever coloumn including path of image which are not necessary and kept only ocr_text coloumn sentiment analysis.

ii. **Import necessary libraries**:- we need to import the necessary libraries for text analysis and sentiment analysis, such as pandas for data handling, nltk for natural language processing, and SentimentIntensityAnalyzer for sentiment analysis and read the dataset.

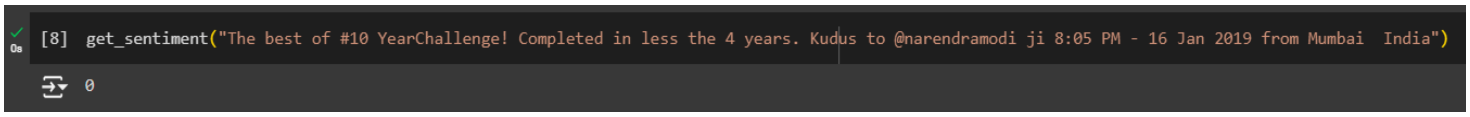
iii. **Preprocess text**:- Here I created a function preprocess_text in which we first tokenize the documents using word_tokenize function from NLTK, remove step words using stepwords module from NLTK and finally,lemmatize the filtered_tokens using WordNetLemmatizer from NLTK.

iv. **Initialize NLTK sentiment analyzer**:-I used function called get_sentiment that takes a text string as its input. The function calls the polarity_scores method of the analyzer object to obtain a dictionary of sentiment scores for the text, which includes a score for positive, negative, and neutral sentiment.The function will then check whether the positive score is greater than 0 and returns a sentiment score of 1 if it is, and a 0 otherwise. This means that any text with a positive score will be classified as having a non-toxic, and any text with a non-positive score will be classified as having a toxic sentiment.

4.2 Inference and Results:-

Therefore, get_sentiment function is used where I given the input as text for predicting toxic i.e 1 or non toxic i.e 0.I used NLTK for sentiment analysis because it has extensive collection of corpora, which includes text data from various sources such as books, news articles, and social media platforms. These corpora provide a rich data source for training and testing NLP models.

Here are the output results:-



```
[8] get_sentiment("The best of #10 YearChallenge! Completed in less the 4 years. Kudus to @narendramodi ji 8:05 PM - 16 Jan 2019 from Mumbai  India")
0
```