IIT Delhi

Indian Institute of Technology Delhi

Meme Classification

submitted by

Bhupender Dhaka, 2018MT60779
Ramneek Singh, 2018MT60788

Project report describing our submissions and attempts in the meme classification kaggle competetion.

Supervisor:
Prof. Niladri Chatarjee

April, 2021

Contents

Chapter 1

Introduction

1.1 Problem Statement

Classification of Memes into **Troll, Humorous or None.**

1.2 Dataset

The dataset contains the text extracted from the meme image along with the meme image. **1991 labelled samples** are provided for training. **599 unlabelled samples** are given as test samples whose predicted labelling we have to submit for grading.

1.3 Overview of our approaches and submissions

We mainly work on the text data

- Text analysis and cleaning, SVM models on 1/2/3 gram split.
- Training BERT model from scratch on our data.
- Using final scores in various categories of pre-trained BERT models as features
- Testing a BERT pre-trained **humor detection** model for humor classification of our data and extracting features from it.
- Testing a BERT pre-trained **sentiment detection** model for troll classification of our data and extracting features from it.

- Training various ML-models on the humor and sentiment features/embeddings extracted previously.
- Extracting other feature/embeddings based on irony, political stance and sarcasm and testing simple models on them.
- Building a multi-modal neural network architecture on the sentiment, humor, irony, political stance and sarcasm features/embeddings extracted so far.

Chapter 2

Data Cleaning and Initial Attempts

2.1 Code

COLAB-NOTEBOOK

The notebook contains our code partitioned into sections with brief descriptions.

2.2 Data-Cleaning

We first clean the text data into space separated base form words and filter out some spam words like meme, and some frequently occurring URLs.
Next we remove stop words from the text and make 3 copies of our data based on the stop word removal done.

1. Text with all stop words
2. Text without all stop words.
3. Text with only certain stop words like negations and quantifiers.

2.3 Models

We tried 4 models on our 3 types cleaned data and compiled the results.

2.3.1 NLTk sentiment analyzer

We used the compound score of the sentiment analyzer to make a simple classification of data.

$$\begin{cases} \text{None} & \text{-}0.1 \leq \text{Score} \leq 0.1 \\ \text{Humorous} & 0.1 < \text{Score} \\ \text{Troll} & \text{-}0.1 > \text{Score} \end{cases}$$

2.3.2 SVM models

We trained a SVM model with linear kernel on 3 different tokenization of texts.

- Uni-gram Tokenization
- Bi-gram Tokenization
- Tri-gram Tokenization

2.4 Results

We compile the F1-score on the test set(from train_test_split) of the models in a table.

'f1-score'

	model	w all stop words	w/o custom stop words	w/o all stop words
0	NLTK-Sentiment-Analyzer	0.349238	0.342325	0.348703
1	SVM-1gram	0.517917	0.517917	0.517917
2	SVM-2gram	0.438116	0.406238	0.416492
3	SVM-3gram	0.427224	0.514291	0.517917

*w/o custom stop words is the 3rd class of cleaned data described above.

Chapter 3

BERT Model for Classification

3.1 Code

COLAB-NOTEBOOK

The notebook contains our code partitioned into sections with brief descriptions.

3.2 BERT base model (uncased)

Pretrained model on English language using a masked language modeling (MLM) objective. It was introduced in **this paper** and first released in **this repository**. We train this model on our data for the classification task.

3.3 Results

We compile the Precision, accuracy and F1-score on the test set(from train_test_split)

	precision	recall	f1-score	support
0	0.31	0.46	0.37	116
1	0.44	0.26	0.33	135
2	0.43	0.43	0.43	148
accuracy			0.38	399
macro avg	0.39	0.38	0.38	399
weighted avg	0.40	0.38	0.38	399

Chapter 4

BERT-TweetEval-Scores

4.1 Code

COLAB-NOTEBOOK

The notebook contains our code partitioned into sections with brief descriptions.

4.2 Using TweetEval models

We first clean our text, keeping all the stop words and then use TweetEvals models[LINK] to get final scores of the text in the 5 categories:
Emotion, Irony, Hate, Offense, Sentiment
We finally get the following 13 features of our text data:-
anger, joy, optimism, sadness,
non_irony, irony,
not-hate, hate,
not-offensive, offensive,
negative, neutral and positive

4.3 Models

4.3.1 Neural Network

We experimented various architectures and finally settled upon the model shown below:

```
model = Sequential()
model.add(Dense(50, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
history = model.fit(X_train, y_train, epochs=60, batch_size=64)
```

4.3.2 SVM and XGB models

We trained a SVM models with with rbf, linear and polynomial kernals. The polynomial model performed the best.
Along with this we trained the XGB model which gave similar results.

4.4 Results

We tabulate the F1-scores of the models on the test set(from train_test_split)

Table 4.1: Results

Model Name	Weighted Average	Class	F-1 Score
Neural Network	0.37	0	0.30
		1	0.36
		2	0.44
SVM	0.38	0	0.35
		1	0.40
		2	0.38
XGB Classifier	0.38	0	0.27
		1	0.43
		2	0.43

Chapter 5

Sentiment, Humor Features and models

5.1 Code

The following colab-notebooks contain our code partitioned into sections with brief descriptions.

1. **Analysis of humor features**
2. **Analysis of sentiment features**
3. **Models on the combined extracted features**

5.2 Humor Features

We used a pre-trained BERT model for humor detection. [LINK]
We first see how the model performs on humor classification of our data. We remove the troll samples and use the model for classification of humorous vs none samples.
Next we extract the final dense layer(i.e. the layer before the output layer) from the model and save the layer outputs as features. The size of the layer is 256. We also train some models: NN, SVM, XGB on the extracted features for humor classification only(as before). The results can be seen in the humor features notebook.

5.3 Sentiment Features

We use a pre-trained BERT model for sentiment detection. [LINK]
We do a similar thing as before and label the troll samples as negative sentiment and none/humorous as positive sentiment and test the performance of model in classification.
Next we extract the embedding layer of the text from the model which is of size 768. We use these as sentiment features and save them. We also train some models: NN, XGM, Decision Tree, Random Forest because of unbalanced data. The results can be seen the sentiment features notebook.

5.4 Models on combined features

Finally we combine(concatenate) the extracted features to get a 1024 sized feature vector on which we train a NN and other classifiers: SVM, XGB and tree based classifiers.
We try PCA to reduce the dimension of features and try our models again.

5.5 Results

We compile the weighted average F1-scores of the models used.

Table 5.1: Results

Model/Features	F1-Score
Neural Network	0.41
SVM-rbf	0.39
SVM-linear	0.38
Random-Forest	0.40
XGB	0.38
Neural Network with PCA	0.42
SVM-rbf with PCA	0.40
SVM-linear with PCA	0.42

Chapter 6

Multi Modal Network

6.1 Code

The following colab-notebooks contain our code partitioned into sections with brief descriptions.

1. **Extracting features for the model**
2. **Model architecture and training**

6.2 Feature Extraction

Along with our previously extracted humor and sentiment features, we extract features/embeddings for irony[LINK], political stance[LINK] and sarcasm [LINK].The size of the extracted features/embeddings is Humor-**256**, Sentiment-**768**, Irony-**768**, Political-**768** and Sarcasm-**1**. Before moving to the multi-modal model, we try the previously used classifiers after dimensionality reduction by PCA. However most of them were over-fitting. The results could be seen in "Extracting features for the model" notebook.

6.3 The model

First we reduce the input embeddings to four dense layers of size 6 and concatenate them differently to capture sentiment in humor, sentiment in irony, sentiment in political stance, humor in irony and irony in political stance. We build up few more dense layers upon these. To prevent over-fitting we use dropout layers appropriately. [LINK] to view the model diagram.

6.4 Results

The classification report of the model on test set(from train_test_split) is attached below.

['Class0: 117', 'Class1: 167', 'Class2: 115']

	precision	recall	f1-score	support
0	0.42	0.40	0.41	121
1	0.41	0.50	0.45	138
2	0.43	0.36	0.39	140
accuracy			0.42	399
macro avg	0.42	0.42	0.42	399
weighted avg	0.42	0.42	0.42	399